Iván Menacho Domínguez

Integración de “Ponient Hotels” en el sistema de ticketing de PortAventura

TRABAJO FINAL DE GRADO

dirigido por Jordi Massaguer Pla

Grado en Técnicas de Desarrollo de Aplicaciones Web y Móviles

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2024

Resumen

PortAventura World inicia en 2022 la adquisición de hoteles en la Costa Daurada con el objetivo de ampliar y mejorar su oferta hotelera, creando la marca Hoteles Ponient.

Estos hoteles, al ser comprados y no construidos por la misma empresa, ya cuentan con su propio sistema de llaves para las habitaciones, lo cual resulta costoso de reemplazar por el sistema utilizado en los hoteles del resort.

Dado que las llaves de estos hoteles externos no permiten a los clientes acceder a las diferentes áreas del resort, se les proporcionan entradas impresas desde un punto de venta estándar de la compañía. Sin embargo, estas entradas impresas, aunque válidas en todo momento, no ofrecen ninguna trazabilidad, lo que puede generar problemas de control y seguridad.

Para resolver este inconveniente, se propone una solución que incluye el desarrollo de dos aplicaciones: una automática de ejecución diaria y otra manual para situaciones excepcionales. Estas aplicaciones permitirán generar reservas a través de peticiones API en el sistema de ticketing al migrar los datos de las reservas de los hoteles, haciendo una prueba piloto en el hotel menos concurrido antes de su paso a producción. De esta manera, se asegura la trazabilidad de las entradas, evitando errores humanos y reduciendo la posibilidad de fraudes.

Esta propuesta no solo mejora la eficiencia operativa, sino que también incrementa la seguridad y satisfacción de los clientes, al permitirles disfrutar de una experiencia más integrada y sin inconvenientes dentro del resort. Además, la implementación de esta solución fortalece el control interno y optimiza los procesos administrativos relacionados con el acceso a las distintas áreas del resort.

Resum

PortAventura World inicia l'any 2022 l'adquisició d'hotels a la Costa Daurada amb l'objectiu d'ampliar i millorar la seva oferta hotelera, creant la marca Hotels Ponient.

Aquests hotels, al ser comprats i no construïts per la mateixa empresa, ja compten amb el seu propi sistema de claus per a les habitacions, cosa que resulta costós de reemplaçar pel sistema utilitzat als hotels del resort.

Com que les claus d'aquests hotels externs no permeten als clients accedir a les diferents àrees del resort, se'ls proporcionen entrades impreses des d'un punt de venda estàndard de la companyia. Tot i això, aquestes entrades impreses, encara que són vàlides en tot moment, no ofereixen cap traçabilitat, cosa que pot generar problemes de control i seguretat.

Per resoldre aquest inconvenient, es proposa una solució que inclou el desenvolupament de dues aplicacions: una automàtica d'execució diària i una altra manual per a situacions excepcionals. Aquestes aplicacions permetran generar reserves a través de peticions API al sistema de ticketing migrant les dades de les reserves dels hotels, fent una prova pilot a l'hotel menys concorregut abans del seu pas a producción. D´aquesta manera, s´assegura la traçabilitat de les entrades, evitant errors humans i reduint la possibilitat de fraus.

Aquesta proposta no només millora l'eficiència operativa, sinó que també incrementa la seguretat i la satisfacció dels clients, en permetre'ls gaudir d'una experiència més integrada i sense inconvenients dins del resort. A més, la implementació d'aquesta solució enforteix el control intern i optimitza els processos administratius relacionats amb l'accés a les diferents àrees del resort.

Abstract

PortAventura World began in 2022 the acquisition of hotels on the Costa Daurada with the aim of expanding and improving its hotel offer, creating the brand Ponient Hotels.

These hotels, being bought and not built by the same company, already have their own key system for the rooms, which is costly to replace by the system used in the resort hotels.

Since the keys to these external hotels do not allow customers to access the different areas of the resort, they are provided with printed tickets from a company's standard point of sale. However, these printed entries, although valid at all times, do not offer any traceability, which can generate control and safety problems.

To solve this problem, a solution is proposed that includes the development of two applications: an automatic daily run and a manual for exceptional situations. These applications will allow you to generate bookings through API requests in the ticketing system by migrating hotel reservation data, doing a pilot test in the least busy hotel before going into production. This ensures the traceability of entries, avoiding human error and reducing the possibility of fraud.

This proposal not only improves operational efficiency, but also increases the safety and satisfaction of customers, allowing them to enjoy a more integrated and seamless experience within the resort. In addition, the implementation of this solution strengthens internal control and optimizes administrative processes related to access to the different areas of the resort.

Índice

1 Introducción

1.1 PortAventura World

PortAventura World es un complejo de ocio ubicado en la Costa Dorada de España, en las localidades de Salou y Vila-Seca, en la provincia de Tarragona. Es uno de los principales destinos turísticos de Europa y atrae a visitantes de todo el mundo.

Figura 1. Logo de PortAventura World

El complejo consta de varios componentes principales:

1. **PortAventura Park**: Es el parque temático principal y el corazón del complejo. Está dividido en cinco áreas temáticas que representan diferentes partes del mundo: Mediterráneo, Polinesia, China, México y Far West. Cada una de ellas ofrece atracciones, espectáculos en vivo, restaurantes y tiendas.
2. **Ferrari Land**: Este parque temático un poco más pequeño está dedicado a la marca italiana de automóviles, Ferrari. Este segundo parque ofrece otro tipo de experiencias más relacionadas con la velocidad o el mundo del motor.
3. **Caribe Aquatic Park**: El parque acuático es una opción para los meses de verano. Con una temática caribeña, ofrece toboganes de agua, piscinas de olas, áreas para niños y más.
4. **Hoteles**: Además de los parques temáticos, PortAventura World ofrece una experiencia de alojamiento a través de su cadena de hoteles temáticos. Estos hoteles están diseñados para complementar la experiencia de los visitantes, ofreciendo una gama de opciones de alojamiento que se adaptan a diferentes preferencias y presupuestos.

En resumen, PortAventura World ofrece una experiencia completa de entretenimiento y diversión para toda la familia, con una combinación de parques temáticos, parque acuático, hoteles temáticos y áreas de entretenimiento.

1.2 Sistema de admisiones

El sistema de admisiones en PortAventura World se erige sobre la base de la utilización de códigos QR o de barras, los cuales son escaneados por tornos de admisión. Este sistema, aparentemente simple en su concepción, constituye la piedra angular de la interacción inicial entre el parque y sus visitantes. Sin embargo, su funcionamiento implica una serie de procesos complejos que abarcan desde la generación de los códigos hasta su validación en los tornos de ingreso.

En primer lugar, es crucial comprender el proceso de generación de estos códigos de barras o QR, que se inicia en el momento de la venta de las entradas. Los sistemas informáticos de PortAventura World están diseñados para generar códigos únicos asociados a cada compra, los cuales contienen información relevante como la fecha de entrada, el tipo de admisión y cualquier otro dato pertinente para la gestión del acceso al parque.

La estructura de estos códigos es de vital importancia, ya que debe permitir una identificación rápida y precisa por parte de los tornos de admisión. Por lo tanto, se utilizan estándares reconocidos en la industria que garantizan la integridad y la legibilidad de la información contenida en los códigos, asegurando así una experiencia fluida para los visitantes.

Una vez que los visitantes llegan al parque y presentan sus códigos de barras o QR en los tornos de admisión, se desencadena un proceso de validación instantáneo. Los tornos están equipados con lectores de alta velocidad y precisión que escanean los códigos y verifican su autenticidad en cuestión de milisegundos. Este aspecto es crucial, ya que cualquier demora o fallo en el proceso de admisión podría generar molestias en los visitantes y afectar negativamente su experiencia.

La importancia de un funcionamiento impecable de este sistema radica en su papel como primer punto de contacto entre el parque y sus visitantes. Cualquier percance o retraso en el proceso de admisión no solo puede causar frustración en los clientes, sino también afectar la percepción general del parque y, en última instancia, provocar pérdidas económicas significativas.

Por lo tanto, la rapidez y la eficiencia son aspectos fundamentales en cada etapa del proceso, desde la compra de la entrada hasta la activación y validación de los códigos de barras o QR en los tornos de admisión. Solo mediante la implementación de un sistema robusto y ágil se puede garantizar una experiencia positiva para los visitantes y, en consecuencia, el éxito continuo de PortAventura World en el mercado del entretenimiento.

En los siguientes puntos podremos ver un poco más a fondo el funcionamiento de los puntos más importantes a la hora de poder acceder al parque, profundizando en la generación de códigos, su estructura y el proceso de admisión en los tornos.

1.2.1 Códigos de barras y QR's

Para empezar, un código de barras o QR es una representación gráfica de datos que se utiliza para identificar de manera única un artículo o producto. El primero consiste en una serie de barras paralelas de diferentes anchuras y espacios entre ellas que codifican información como números o letras, el segundo es igual pero con patrones de píxeles. Esta información es interpretada por un lector de códigos de barras, que la convierte en datos legibles por un sistema informático. En nuestro caso, nuestro producto principal son las entradas con su correspondiente código de barras o QR y el lector de código de barras es el torno, el cual tiene implementado el sistema informático que decide admitir o no ese código.

Figura 2. Código de barras y código QR

En PortAventura World predefinimos estos códigos de barras con una estructura o "media", las cuales tienen diferentes variables según el canal que los emita, los cuales son varios, pero los principales son Galaxy y Experticket para entradas u Opera para los hoteles.

Galaxy es un software de la empresa Gateway Ticketing Systems utilizado en todo el mundo. Este sistema permite desde la configuración de productos de Ticketing hasta su misma venta, siendo un software tan potente que llega a ser utilizado por grandes empresas alrededor de todo el planeta, pudiendo llegar a encontrarlo en sitios tan conocidos como el Museo de Historia Natural de Londres o Disneyland París.

Otro canal que tenemos integrado para la venta de tickets en nuestra empresa es Experticket, el cual es un sistema de distribución de tickets especializado, sobre todo en parques temáticos, disponiendo de una API que permite a los clientes integrarse con ellos para vender entradas a través de esta plataforma. En nuestro caso, nuestra integración hace que los tickets emitidos a través Experticket no pasen por Galaxy, teniendo una media estructura diferente para poder identificarlos, pasando directamente a nuestro Sistema Integral de Control de accesos, en adelante SICA.

Por la parte de Opera, lo utilizamos en nuestro sistema de hoteles internos, los cuales hacen que la misma llave de habitación sirva para entrar a los parques del resort. Estas llaves son una tarjeta que una banda magnética para entrar a la habitación, y el código de barras impreso, que se envía a SICA y permite el acceso al parque con el mismo.

1.2.2 Sistema Integral de Control de Accesos

SICA es, como he dicho anteriormente, el Sistema Integral de Control de Accesos, el cual es el entorno de gestión de accesos al parque, gestionado por Inetum, siendo este mismo los tornos que permiten o no entrar, según tenemos nosotros configurado en su aplicación web el acceso del producto.

Inetum es una empresa global de servicios y soluciones digitales que ofrece una amplia gama de servicios en áreas como consultoría, tecnología, y gestión de negocios. Anteriormente conocida como Gfi Informatique, la compañía cambió su nombre a Inetum en 2020. Su objetivo principal es ayudar a las empresas y organizaciones a digitalizarse y transformarse para adaptarse a los desafíos del mundo moderno.

La empresa cuenta con una amplia red de expertos y profesionales en tecnología y negocios, y su enfoque se centra en la innovación, la calidad y la satisfacción del cliente. Inetum tiene presencia en varios países alrededor del mundo y trabaja en estrecha colaboración con sus clientes para ayudarles a alcanzar sus objetivos empresariales mediante soluciones digitales adaptadas a sus necesidades específicas.

Figura 3. Logotipo de Inetum

En nuestro caso, nos ofrece la aplicación web SICA un amplio sistema en el cual podemos gestionar diferentes apartados relacionados con las admisiones, como los códigos de acceso, los quioscos de autoventa, los calendarios para las entradas o incluso el visionado de diferentes informes.

1.3 Sistema de hoteles

Los visitantes tienen la opción de elegir entre dos tipos de hoteles: los In-Park, que ofrecen acceso directo a los parques de atracciones para una conveniencia inigualable, y los Ponient Hotels, que consisten en tres establecimientos ubicados en las ciudades circundantes. Estos últimos han sido adquiridos para ampliar la oferta hotelera de PortAventura de una manera más asequible, permitiendo a los huéspedes disfrutar de una experiencia vacacional completa tanto en el resort como en la Costa Daurada.

1.3.1 Operativa de los hoteles In-Park

Actualmente, son cinco los hoteles que forman parte de esta experiencia de inmersión total: el Hotel PortAventura, el Hotel Gold River, el Hotel El Paso, el Hotel Caribe y el Hotel Mansión de Lucy.

Una de las características distintivas de estos hoteles es su sistema de acceso a PortAventura Park. Cada huésped recibe una llave de habitación en forma de tarjeta, que no solo sirve para abrir las puertas de su alojamiento gracias a una banda magnética en su parte posterior, sino que también contiene un código de barras. Este código de barras está vinculado al sistema de reservas del hotel, que se gestiona a través del sistema Opera.

Durante la duración de la estancia del cliente en el hotel, este código de barras habilita el acceso ilimitado a los tornos del parque temático. Esto significa que los huéspedes pueden disfrutar de todas las atracciones, espectáculos y entretenimiento que ofrece PortAventura Park tantas veces como deseen durante su alojamiento, sin tener que preocuparse por la compra de entradas adicionales o la espera en las colas de acceso.

Este sistema proporciona una experiencia fluida y sin complicaciones para los huéspedes, permitiéndoles aprovechar al máximo su tiempo en el parque temático mientras disfrutan de las comodidades y servicios de los hoteles del resort.

Figura 4. Logo de PortAventura Hotels

1.3.2 Operativa de los Ponient Hotels

Los hoteles que forman parte del resort de PortAventura ofrecen a los huéspedes una experiencia integrada y sin fisuras, pero es importante destacar que algunos de estos hoteles, como los Ponient Hotels adquiridos posteriormente, tienen una operativa ligeramente diferente debido a su historia previa antes de ser adquiridos por la compañía.

Antes de que estos hoteles fueran adquiridos por PortAventura World, ya estaban en funcionamiento y tenían su propio sistema de llaves de habitaciones establecido. En lugar de cambiar completamente este sistema, lo que implicaría costos significativos, la decisión se tomó para mantenerlo intacto. Como resultado, estos hoteles conservaron sus métodos originales de generación de llaves.

En el caso específico de la distribución de accesos al parque, en lugar de utilizar el sistema de llaves con códigos de barras como en los hoteles del resort, se optó por una solución diferente. A través del sistema de ticketing Galaxy, se proporcionaban entradas impresas a los clientes como parte de su paquete de alojamiento. Estas entradas le otorgaban acceso al parque temático durante su estancia en el hotel.

Esta adaptación inteligente permitió que los hoteles recién adquiridos mantuvieran su funcionamiento sin interrupciones importantes mientras se integraban en la oferta general de PortAventura World.

Figura 5. Logo de Ponient Hotels

2 Objetivos

2.1 Problemas para abordar

Desde la adquisición del primer hotel Ponient, se ha enfrentado a una operativa desafiante en cuanto al acceso de los clientes al parque. Aunque los hoteles adquiridos ya contaban con un sistema de gestión de reservas, al pasar a formar parte de PortAventura World, se busca ofrecer un servicio que esté a la par con los estándares de calidad establecidos. Para abordar esta necesidad, se implementó el sistema Opera para la gestión de reservas, pero surgió un obstáculo: la falta de integración con el sistema de llaves de habitación del parque, lo que impedía proporcionar a los clientes una llave multifuncional. Esta limitación condujo a la adopción de un sistema de impresión de tickets Galaxy como solución provisional.

Sin embargo, esta solución presentaba varias deficiencias:

- Posibilidad de errores humanos en el proceso de impresión.
- Riesgo de fraude debido a la falta de control y trazabilidad.
- Ausencia de vinculación entre los tickets impresos y las reservas en Opera.

Al depender de la intervención humana en la recepción del hotel para imprimir los tickets, se incrementaba la probabilidad de errores y confusiones. Además, al carecer de identificación nominal, los tickets podrían ser utilizados de manera fraudulenta en cualquier momento durante la estancia del cliente. Por último, la falta de trazabilidad de los tickets limitaba las oportunidades de análisis de datos y otros fines.

2.2 Propuestas de solución

Para abordar estas deficiencias, se propone una solución integral mediante dos aplicaciones:

1. **Aplicación automatizada de migración de reservas:** Se desarrollará una aplicación que ejecute automáticamente cada día a una hora predeterminada. Esta aplicación transferirá las reservas desde el sistema de reservas de hoteles Opera al sistema de ticketing Galaxy. De esta manera, se crearán en Galaxy reservas completas con todos los datos necesarios del cliente, lo que permitirá al recepcionista del hotel acceder a ellas con el mismo número de referencia utilizado en Opera. Así, se podrán imprimir los tickets correspondientes de forma eficiente y precisa.
2. **Aplicación de escritorio para gestión manual:** Además, se desarrollará una aplicación de escritorio que pueda ser utilizada excepcionalmente en casos de errores o reservas no transferidas correctamente. Esta aplicación manual proporcionará la capacidad de transferir las reservas del día seleccionado y generar un documento con todas ellas para mantener un registro detallado y garantizar la integridad de los datos.

Con estas aplicaciones, se busca mejorar significativamente la eficiencia operativa, reducir los errores humanos, mitigar el riesgo de fraude y aumentar la trazabilidad de las reservas y los tickets, lo que contribuirá a una experiencia más satisfactoria para los clientes y facilitará el análisis de datos para diversas áreas de interés.

3 Planificación

Para estructurar el proyecto, se han establecido distintos períodos de tiempo, cada uno con objetivos específicos. En este contexto, optamos por dividirlo en cinco metas más generales con plazos más amplios. Esta elección se fundamenta en la posibilidad de que surjan variaciones menores durante el desarrollo, permitiendo así una mayor flexibilidad para cumplir con todas las etapas del proyecto.

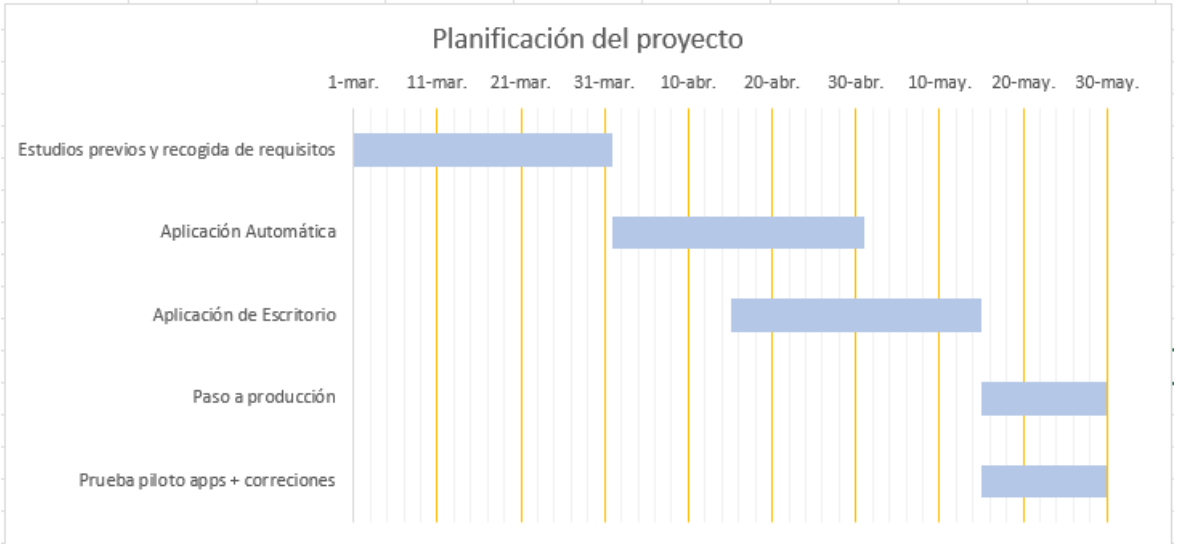

Figura 6. Diagrama de Gantt con planificación

3.1 Estudios previos y recogida de requisitos

La previsión se establece para marzo. Periodo para estudiar la API de Galaxy, recoger los requisitos y preparación de la base de datos donde se almacena toda la información que se trata en las aplicaciones, pidiendo una vista al técnico responsable del sistema de hoteles para la extracción de las reservas de hotel y el desarrollo del procedimiento almacenado que trata los datos de la vista introduciéndolos en una tabla.

3.2 Desarrollo de la aplicación automática

Programado para realizarse en abril. Durante este tiempo se completa el desarrollo de la aplicación automática en un entorno de pruebas. Se comienza por esta aplicación porque no tiene interfaz gráfica y es más rápida de desarrollar.

3.3 Desarrollo de la aplicación de escritorio

Cuando ya se tiene la mitad de la aplicación automática, se empieza a implementar la aplicación de escritorio. Se realiza de mitades de abril a mitades de mayo. Al tener la funcionalidad desarrollada en la automática, para esta se dedica más tiempo a la interfaz gráfica y a la usabilidad.

3.4 Paso a producción

Con previsión de ser realizado durante la segunda mitad de mayo, se instala la aplicación automática en un servidor de la compañía programando una tarea que ejecuta la aplicación cada día a la misma hora, instalando también la de escritorio en los diferentes hoteles Ponient.

3.5 Prueba piloto y correcciones

También teniendo la previsión de realizarse durante la segunda mitad de mayo, se da permisos de ejecución a los usuarios del hotel Ponient Vila-Centric (hotel menos concurrido) y se empieza a ejecutar la nueva operativa. Se revisan logs y se detectan las incidencias para posteriores correcciones

4 Requisitos del proyecto

Ambas aplicaciones tienen un propósito común: migrar reservas desde el sistema Opera hacia Galaxy y facilitar el acceso de los clientes. Sin embargo, presentan requisitos distintos. En este apartado se analizará el diagrama de casos de uso del proyecto y sus requisitos:

4.1 Diagrama de casos de uso

En el siguiente diagrama de casos de uso podemos identificar dos actores, los cuales se pueden diferenciar por ser las dos aplicaciones, la automática y la manual, compartiendo tres casos de uso diferentes y añadiéndose dos más a la manual:

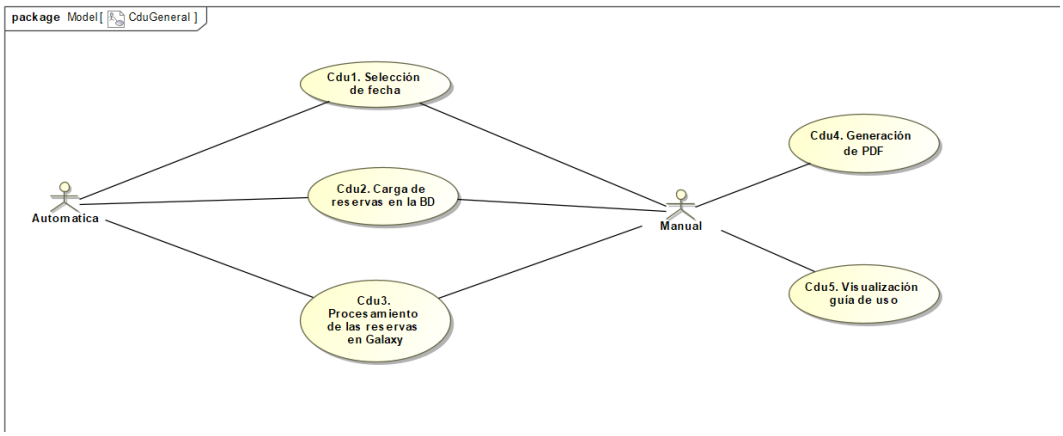

Figura 7. Diagrama de casos de uso de la aplicación de escritorio

Los casos de uso que podemos encontrar son los siguientes:

4.1.1 CDU 1. Selección de fecha:

Ejecutado por las dos aplicaciones, donde se guarda en una variable una fecha que será utilizada posteriormente en el "*CDU 2*", "*CDU 3*" y "*CDU 4*".

4.1.2 CDU 2. Carga de reservas en la BD

Ejecutado por las dos aplicaciones, utilizando el valor de la variable del "*CDU 1*", ejecuta un procedimiento almacenado de la base de datos que consulta una vista en Oracle (Base de datos de Opera) e inserta las reservas sin procesar y no existentes en una tabla.

4.1.3 CDU 3. Procesamiento de las reservas en Galaxy

Ejecutado por las dos aplicaciones, consulta las reservas que no están procesadas en Galaxy, envía las peticiones API necesarias según la variable recogida en el "*CDU 1*", generando así reservas en Galaxy. Una vez generadas, consulta la respuesta de la petición y

actualiza la tabla marcando las reservas procesadas y añadiendo el identificador de reserva de Galaxy.

4.1.4 CDU 4. Generación de PDF

Ejecutado únicamente por la aplicación de escritorio (manual), consulta en la tabla filtrando por la variable recogida en el "*CDU 1*" y el hotel deseado, genera y guarda un archivo PDF con los datos de la reserva, incluyendo el identificador de reserva de Opera en código de barras para poder escanearse posteriormente y no tener errores a la hora de la impresión de los tickets.

4.1.5 CDU 5. Visualización de la guía de uso

Ejecutado únicamente por la aplicación manual, el usuario puede visualizar un cuadro de texto que incluye una guía de uso en caso de no saber cómo utilizar la aplicación.

El diagrama permite visualizar las interacciones de ambas aplicaciones, mostrando un claro flujo de trabajo donde, al seleccionar la fecha, se asigna este valor como parámetro del procedimiento almacenado de la base de datos. Luego, al ejecutar el "*CDU 3*", se realizan las peticiones API con los datos de la tabla, lo que resulta en la creación de reservas en Galaxy. Por último, los dos casos de uso restantes involucran la generación del PDF de las reservas y la visualización de la guía de uso a través de una caja de texto, que representan interacciones simples del usuario con la aplicación de escritorio.

4.2 Requisitos generales

Existen unos que no son específicamente de ninguna aplicación en concreto, es decir, en este apartado podremos ver los requisitos funcionales externos al funcionamiento de las dos aplicaciones:

4.2.1 Tipo de base de datos

Base de datos relacional (RDBMS) en SQL Server, ya proporcionado y gestionado por la empresa, garantizando la integridad, seguridad y facilidad de acceso a los datos.

4.2.2 Tablas

Una tabla general en la que se guarden los datos principales de las reservas de Opera, teniendo como clave primaria el identificador de reserva de Opera añadiendo campos de control e identificación.

4.2.3 Vistas

Vista en Oracle que recoja todas las reservas de los Hoteles Ponient utilizadas por un procedimiento almacenado que seleccionará y filtrará las necesarias para gestionarlas posteriormente.

4.2.4 Procedimientos

Procedimiento almacenado con un parámetro de tipo DATE que, consultando la vista en Oracle (SGBD de Opera), filtra e inserta los datos en la tabla principal, comprobando que no se introducen valores repetidos en el identificador del número de reserva de Opera.

Estos requisitos generales establecen un marco sólido para la gestión y operación de las bases de datos y vistas asociadas, asegurando la integridad y consistencia de los datos en todo momento.

4.3 Requisitos funcionales aplicación automática

La aplicación automática tiene el objetivo de ejecutarse cada día a las 00:30 a través de una tarea creada en un servidor de la empresa migrando las reservas con entradas del día de Opera de todos los Hoteles Ponient hacia Galaxy, para ello son necesarios diferentes puntos:

4.3.1 Extracción de datos de Opera

Conexión con la base de datos de Opera a través de un procedimiento almacenado para extraer todas las reservas que contienen una tarifa que incluye entradas.

4.3.2 Transformación de los datos

Adaptación de los datos extraídos al formato requerido por Galaxy.

4.3.3 Migración de los datos hacia Galaxy

Peticiones API con los datos de las reservas de Opera para generarlas en Galaxy.

4.3.4 Actualización del estado de los datos

Gestión del estado de las reservas, indicando si ya se han procesado.

4.3.5 Registro de operaciones (Log)

Creación y mantenimiento de un log detallado y diario de todas las operaciones realizadas durante el proceso de migración y actualización de estado de los datos.

4.3.6 Gestión de errores y recuperación:

Detección y manejo de errores durante la migración de datos.

4.3.7 Programación de la ejecución automática

Configuración y mantenimiento de la tarea programada para asegurar la ejecución diaria de la aplicación.

La implementación de los requisitos funcionales para la aplicación automática de migración de reservas desde Opera hacia Galaxy establece un sólido marco para garantizar la eficiencia y la fiabilidad del proceso automatizado. Estos elementos son fundamentales para el correcto funcionamiento y la gestión óptima de las operaciones diarias de los Hoteles Ponient.

4.4 Requisitos no funcionales aplicación automática

La calidad y el rendimiento de una aplicación automática que se ejecuta diariamente son elementos cruciales que van más allá de su funcionalidad básica. Para garantizar un desempeño óptimo y una ejecución fiable en diversas condiciones, es fundamental definir y cumplir con una serie de requisitos no funcionales. Estos requisitos abarcan aspectos que influyen directamente en la eficacia y la aceptación del software por parte de los usuarios finales y administradores del sistema:

4.4.1 Rendimiento

Debe ser capaz de manejar grandes volúmenes de datos sin experimentar una degradación significativa del rendimiento. La ejecución diaria no debe exceder los 10 minutos para evitar impactos negativos en el rendimiento de los sistemas y garantizar que no interfiera con otros procedimientos operativos.

4.4.2 Seguridad

Se debe proporcionar seguridad integrada en el momento de la conexión a la base de datos y no debe tener credenciales en claro.

4.4.3 Fiabilidad

La aplicación debe ser estable y no debe experimentar caídas inesperadas o errores graves durante el uso normal y debe tener una tasa de disponibilidad del 99.9 %, lo que significa que estará operativa durante al menos el 99.9 % del tiempo de servicio planificado.

4.4.4 Mantenibilidad

El código debe seguir estándares de codificación limpios y ser fácilmente legible y mantenible por otros desarrolladores y debe ser modular y extensible para facilitar la incorporación de nuevas características y corrección de errores.

4.4.5 Compatibilidad

Debe ser compatible con las diferentes configuraciones de hardware para garantizar una experiencia consistente en diferentes entornos.

4.4.6 Implementación técnica de la aplicación

La aplicación automática será desarrollada como aplicación de consola utilizando el lenguaje de programación Visual Basic (VB) dentro del entorno de desarrollo proporcionado por .NET Framework 4.8.

La definición y el cumplimiento de estos requisitos no funcionales son aspectos críticos en el desarrollo de una aplicación automática que busca ofrecer un rendimiento fiable y constante. A través de este análisis detallado de los requisitos no funcionales, se ha establecido un marco sólido que guiará el diseño, desarrollo y evaluación de la aplicación, asegurando su calidad y eficacia en diversas dimensiones.

4.5 Requisitos funcionales aplicación de escritorio

La aplicación de escritorio está diseñada para intervenir en situaciones excepcionales, mientras que la aplicación automática se encarga de simplificar al máximo las tareas diarias de los recepcionistas. Sin embargo, en caso de surgir alguna de estas excepciones, la aplicación de escritorio debe ser capaz de generar la reserva en Galaxy y emitir las entradas correspondientes para el cliente de manera rápida y eficiente. Para ello, se proponen diferentes puntos:

4.5.1 Calendario

Implementar un *widget* de calendario que permita la selección del día para extraer las reservas de Opera.

4.5.2 Carga remota de reservas

Facilitar la ejecución remota del procedimiento almacenado para extraer las reservas, utilizando como parámetro la fecha seleccionada en el calendario.

4.5.3 Bloqueo de interacciones

Garantizar que durante la ejecución del procedimiento almacenado no se pueda interactuar con la aplicación para evitar casos de error.

4.5.4 Procesamiento de reservas

Ejecutar las peticiones API necesarias para la generación de reservas en Galaxy de manera eficaz y sin demoras.

4.5.5 Guía de uso integrada

Incluir una caja de texto en un botón que proporcione una guía paso a paso sobre cómo utilizar la aplicación de manera efectiva.

4.5.6 Impresión de reservas

Generar un archivo PDF que contenga el número de reserva en texto y código de barras, junto con los datos del cliente, para su escaneo y posterior impresión en Galaxy.

4.5.7 Tabla de reservas

Mostrar en una cuadrícula los datos de las reservas del día seleccionado en el calendario, facilitando así la visualización y gestión de estas.

4.5.8 Gestión de color

Implementar un sistema de coloración que resalte el estado de las reservas, utilizando verde para indicar reservas procesadas y migradas correctamente a Galaxy, y rojo para aquellas que presenten algún inconveniente.

4.5.9 Registro de operaciones (Log)

Creación y mantenimiento de un log detallado y diario de todas las operaciones realizadas durante el proceso de migración y actualización de estado de los datos.

La implementación de estos requisitos funcionales en la aplicación de escritorio no solo garantizará una gestión eficiente de reservas en situaciones excepcionales, sino que también contribuirá a optimizar las operaciones diarias de los recepcionistas. Además, estos elementos ofrecen una experiencia de usuario intuitiva y fluida, mejorando la productividad y la satisfacción tanto del personal como de los clientes.

4.6 Requisitos no funcionales aplicación de escritorio

La calidad y el rendimiento de una aplicación de escritorio son elementos cruciales que van más allá de su funcionalidad básica. Para garantizar una experiencia de usuario satisfactoria y un desempeño óptimo en diversas condiciones, es fundamental definir y cumplir con una serie de requisitos no funcionales. Estos requisitos abarcan aspectos que influyen directamente en la eficacia y la aceptación del software por parte de los usuarios finales:

4.6.1 Usabilidad

La interfaz de usuario debe ser intuitiva y fácil de navegar, con un diseño limpio y coherente. El tiempo de aprendizaje para utilizar la aplicación no debe exceder los 30 minutos para un usuario promedio, debe permanecer en una carpeta de red en la que los usuarios tendrán acceso previamente,

4.6.2 Rendimiento

Debe ser capaz de manejar grandes volúmenes de datos sin experimentar degradación del rendimiento exponencial.

4.6.3 Seguridad

Se debe proporcionar seguridad integrada en el momento de la conexión a la base de datos y no debe tener credenciales en claro.

4.6.4 Fiabilidad

La aplicación debe ser estable y no debe experimentar caídas inesperadas o errores graves durante el uso normal y debe tener una tasa de disponibilidad del 99.9 %, lo que significa que estará operativa durante al menos el 99.9 % del tiempo de servicio planificado.

4.6.5 Mantenibilidad

El código debe seguir estándares de codificación limpios y ser fácilmente legible y mantenible por otros desarrolladores y debe ser modular y extensible para facilitar la incorporación de nuevas características y corrección de errores.

4.6.6 Compatibilidad

Debe ser compatible con diferentes resoluciones de pantalla y configuraciones de hardware para garantizar una experiencia consistente en diferentes entornos.

4.6.7 Implementación técnica de la aplicación

La aplicación de escritorio se desarrollará con el lenguaje de programación Visual Basic (VB) en el entorno de desarrollo proporcionado por .NET Framework 4.8.

La definición y el cumplimiento de los requisitos no funcionales son aspectos críticos en el desarrollo de una aplicación de escritorio que busque ofrecer una experiencia de usuario óptima y un rendimiento confiable. A través de este análisis detallado de los requisitos no funcionales, se ha establecido un marco sólido que guiará el diseño, desarrollo y evaluación de la aplicación, asegurando su calidad y eficacia en diversas dimensiones.

5 Diseño

En este apartado se analizará el comportamiento de la aplicación, comenzando con un diagrama de secuencias general que muestra el flujo necesario para integrar las reservas de hoteles en el sistema de ticketing, facilitando así el trabajo del recepcionista. Este diagrama ilustrará cómo la aplicación se comunica con la base de datos de Opera para extraer las reservas pertinentes, cómo las procesa y las envía al sistema Galaxy mediante peticiones API personalizadas, y cómo maneja las confirmaciones recibidas de Galaxy. Posteriormente, podremos ver también observar el diseño de la aplicación de escritorio y lo que encontrarán los usuarios cuando vayan a ejecutar la operativa.

5.1 Diagrama de secuencias general

En el diagrama de secuencias vemos el flujo de interacción de las dos aplicaciones con los demás sistemas cumpliendo los objetivos generales. Podemos observar 4 actores/sistemas involucrados:

- **Usuario**: En el caso de la aplicación automática sería la tarea automática que la ejecuta, en el caso de la de escritorio el recepcionista.
- **Aplicación**: Cualquiera de las dos aplicaciones.
- **BD (Base de datos)**: SGBD incluyendo aquí vistas, tablas y procedimientos almacenados.
- **Galaxy**: Sistema de ticketing utilizado por la compañía.

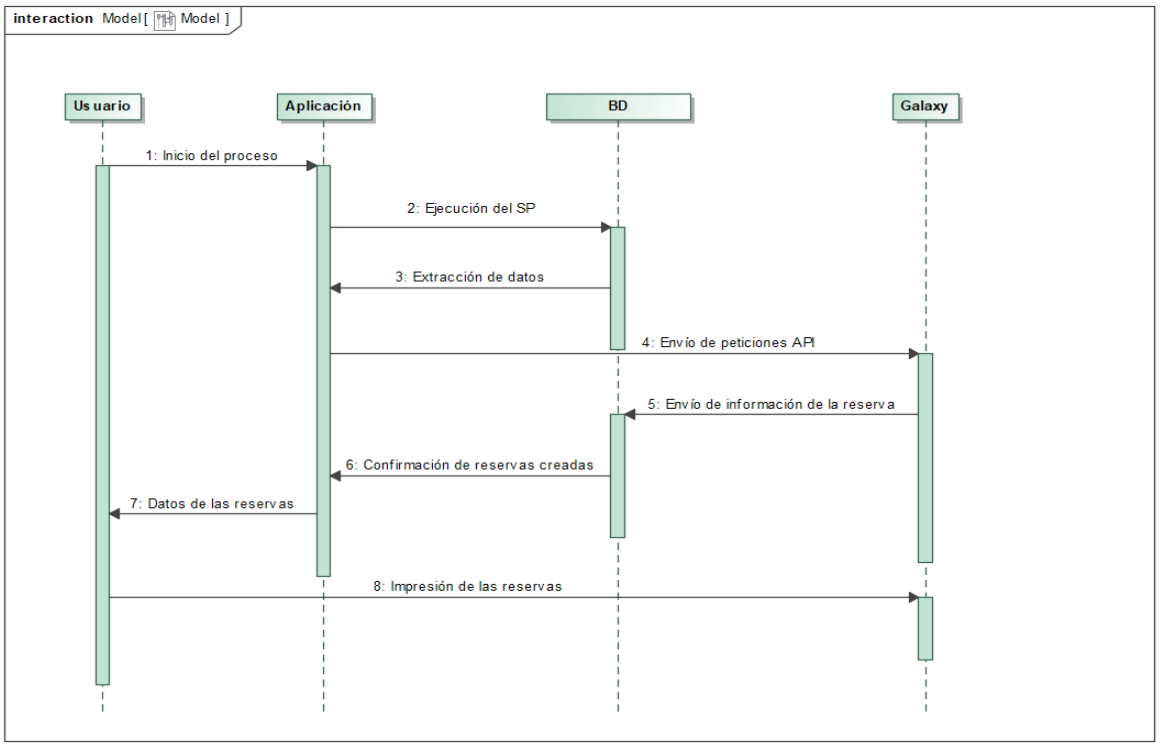

Figura 8. Diagrama de secuencias

El flujo de interacciones empezaría desde la parte izquierda del diagrama, concatenando cada una de las interacciones:

1. **Inicio del proceso.**
2. **Ejecución del SP (“*Stored procedure”, procedimiento almacenado en castellano*).**
3. **Extracción de datos.**
4. **Envío de peticiones API.**
5. **Envío de la información de las reservas.**
6. **Confirmación de las reservas creadas.**
7. **Envío de datos de las reservas.**
8. **Impresión de las reservas.**

El flujo comienza cuando el usuario inicia el archivo “.exe” de la aplicación. Al hacer esto, se activa un procedimiento almacenado que se encarga de manejar la interacción con nuestra base de datos. Este procedimiento almacenado está configurado para conectarse a una vista específica en Oracle, que es la base de datos utilizada por el sistema Opera.

El objetivo del procedimiento almacenado es recoger las reservas del día actual (o el seleccionado en caso de la aplicación manual) que aún no se encuentran en nuestra tabla interna y que contienen tarifas que incluyen entradas desde la vista de Opera hecha por el técnico encargado del sistema de hoteles. Si las reservas ya existen en la tabla o no contienen una tarificación con entradas, no se insertarán. En este punto de la ejecución se verían incluidos los casos de uso “*CDU 1*” y “CDU 2” y se cumplirían los requisitos funcionales 4.2.1, 4.2.2, 4.2.3, 4.2.4, 4.3.1, 4.3.2, 4.5.1, 4.5.2 y 4.5.3.

Una vez que las reservas han sido almacenadas en la tabla, la aplicación procede a la siguiente etapa: Ejecuta una consulta para recoger estas reservas, filtrando específicamente por aquellas que aún no han sido procesadas. Este filtrado garantiza que solo se trabaje con las reservas nuevas y pendientes. Con los datos de las reservas en mano, la aplicación comienza a generar las peticiones API necesarias para la creación de las reservas en el sistema Galaxy. Cada petición API es personalizada y contiene toda la información necesaria para que Galaxy pueda crear las reservas correctamente. Estas peticiones son enviadas a Galaxy de manera secuencial o concurrente.

Al recibir las peticiones, procesa cada una y crea las reservas en su sistema. Una vez que las reservas han sido creadas, Galaxy envía una confirmación de vuelta a la aplicación. La aplicación recoge estas confirmaciones y actualiza la tabla marcando cada reserva de las enviadas como procesada. En este punto de la ejecución se vería incluido el caso de uso “*CDU 3*” y los requisitos funcionales 4.3.4, 4.3.4 y 4.5.4.

Finalmente, el usuario recibe los datos de las reservas procesadas. Utilizando la función de impresión rápida de entradas de Galaxy, el usuario puede imprimir las entradas y entregarlas al cliente sin ningún problema. Este proceso asegura que todas las reservas que incluyen entradas se gestionen de manera eficiente y que los clientes reciban sus entradas de forma oportuna.

En el caso específico de la aplicación de escritorio, el usuario también podrá generar un archivo PDF con los datos de las reservas, consultar una guía de uso y visualizar una tabla de reservas del día seleccionado, viendo las procesadas de color verde y las no procesadas de color rojo, cumpliéndose así los casos de uso “*CDU 4*” y “*CDU 5*” y los requisitos funcionales 4.5.5 y 4.5.6, 4.5.7 y 4.5.8.

Durante todo el código de las dos aplicaciones se tiene un control de errores muy extenso y una gestión de un log muy completo, cumpliendo así los requisitos funcionales 4.3.5, 4.3.6 y 4.5.9 y teniendo en cuenta todos los requisitos no funcionales presentados en el anterior apartado.

5.2 Diseño gráfico de la aplicación de escritorio

El diseño gráfico de la aplicación de escritorio es sencillo e intuitivo. La ventana tiene una resolución fija de 1028x522 píxeles, lo que significa que no se puede cambiar su tamaño. Esta restricción elimina la necesidad de anclar los elementos de la interfaz a los bordes de la ventana, garantizando un diseño consistente. Dado que los dispositivos proporcionados por la empresa para ejecutar la aplicación tienen resoluciones adecuadas, se asegura una visualización correcta y óptima en todas las circunstancias, teniendo una disposición de los elementos de izquierda a derecha, haciendo que sea aún más intuitiva.

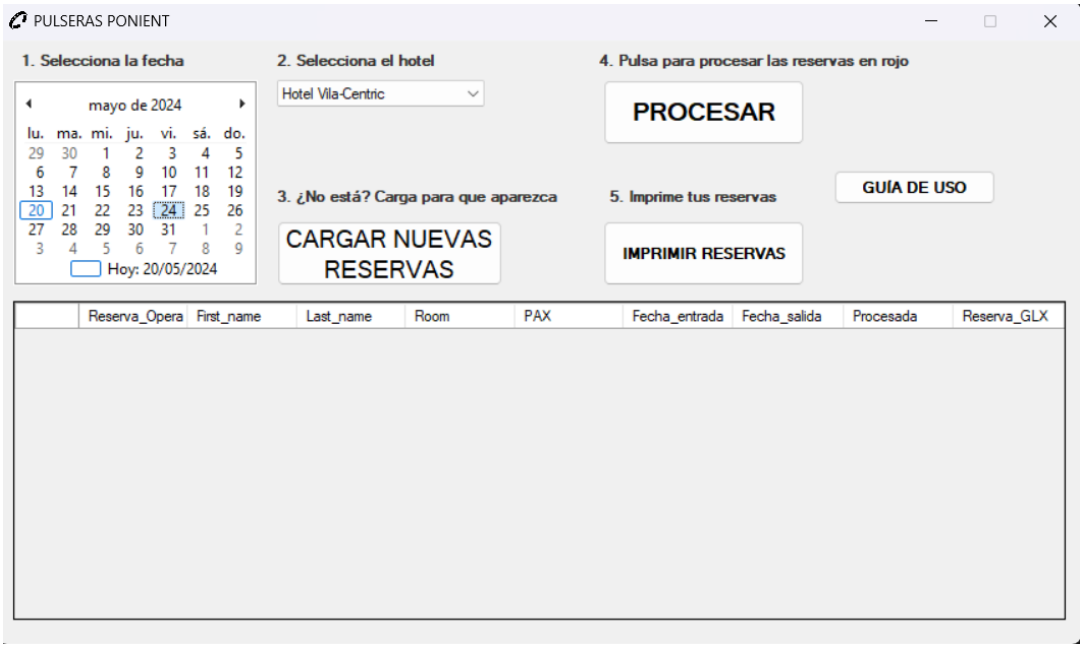

Figura 9. Diseño gráfico de la aplicación

Como se puede observar en la figura previa, tenemos diferentes elementos en la ventana principal, entre ellos etiquetas de texto, un calendario, un desplegable, cuatro botones y una tabla de datos:

- **Etiquetas de texto**: Muestran los pasos a seguir para que el usuario no se pierda durante el uso de la aplicación.
- **Calendario**: Permite seleccionar una fecha que será guardada en una variable, siendo utilizada posteriormente como parámetro en la ejecución del procedimiento almacenado.
- **Desplegable**: Permite seleccionar el hotel, guardando el valor en una variable para elegir las reservas que se van a procesar y las que se van a mostrar en la tabla de

datos. Normalmente estará bloqueado a un solo hotel en el archivo de configuración para evitar errores humanos a la hora del uso de la aplicación.

- **Botón "CARGAR NUEVAS RESERVAS"**: Ejecuta el procedimiento almacenado pasando como parámetro la fecha seleccionada en el calendario, guardando así las reservas en la tabla de la base de datos y mostrándolas en la tabla de datos de la parte de abajo.
- **Botón "PROCESAR"**: Envía peticiones API hacia Galaxy con los datos recogidos en una lista enlazada provenientes de la tabla de la base de datos donde las reservas de hoteles con tarifas que incluyen entradas han sido cargadas gracias al procedimiento almacenado, pudiendo posteriormente crear las reservas en el sistema Galaxy.
- **Botón "IMPRIMIR RESERVAS"**: Genera y almacena un PDF que contiene varios datos de la reserva:
    - **Reserva de Opera**: Mostrando el valor en formato código de barras y en texto para que se pueda escanear y evitar errores humanos a la hora de introducir el identificador en Galaxy.
    - **Nombre**: Mostrando el nombre del titular de la reserva.
    - **Apellidos**: Mostrando los apellidos del titular de la reserva.
    - **Reserva de Galaxy**: Mostrando el identificador de la reserva en Galaxy.

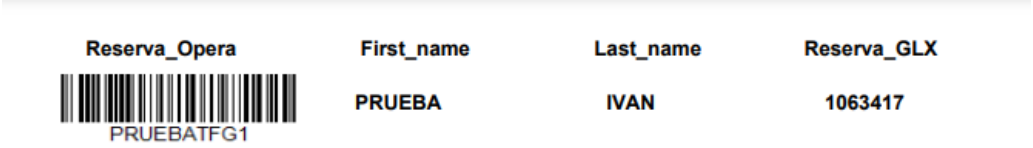

Figura 10. Ejemplo de cómo se mostraría el PDF generado

- **Botón "GUÍA DE USO"**: Muestra una ventana de texto en la que se pueden observar las instrucciones, permitiendo al usuario guiarse cuando se encuentre perdido:

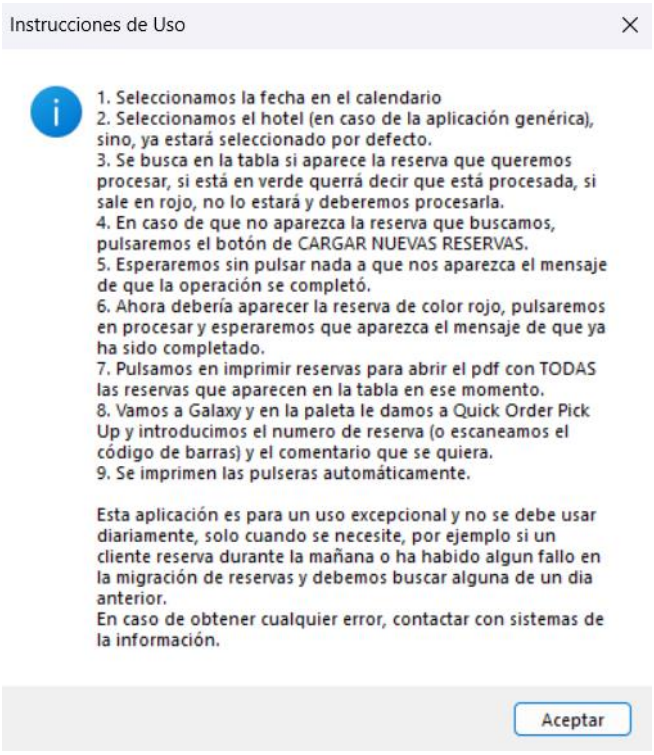

Figura 11. Ventana de texto con las instrucciones de la aplicación de escritorio

- **Tabla de datos**: Conecta con la base de datos y muestra unas columnas de la tabla principal de reservas según el día y el hotel seleccionados, actualizándose constantemente después de cada interacción del usuario. Muestra en verde las reservas procesadas y en rojo las que no se han procesado todavía:

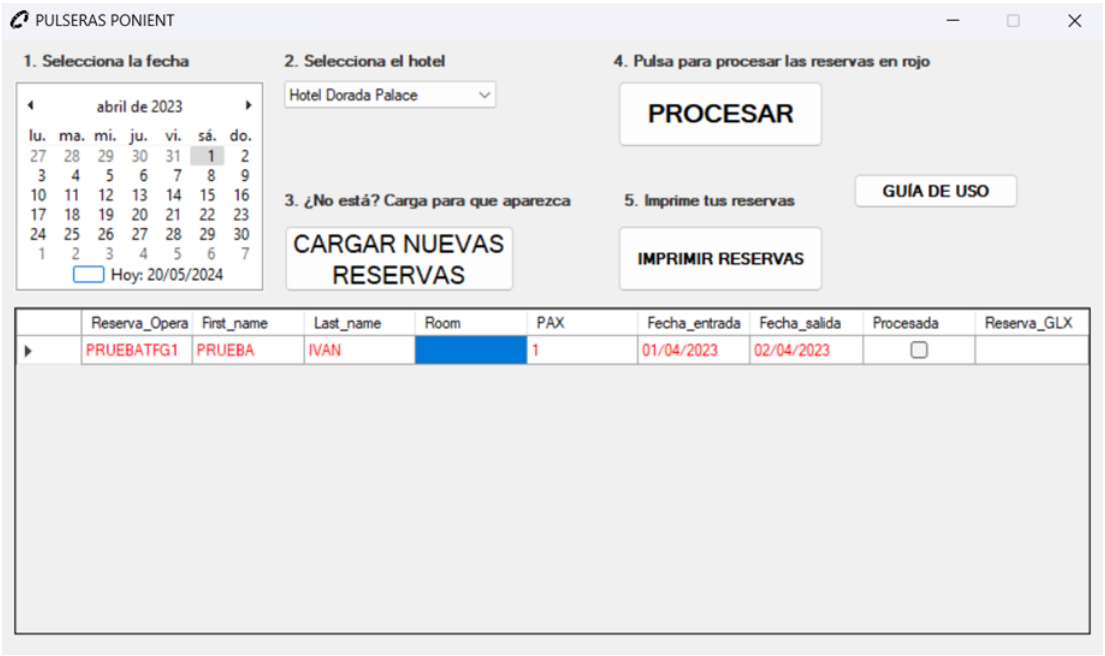

Figura 12. Aplicación de escritorio con una reserva sin procesar

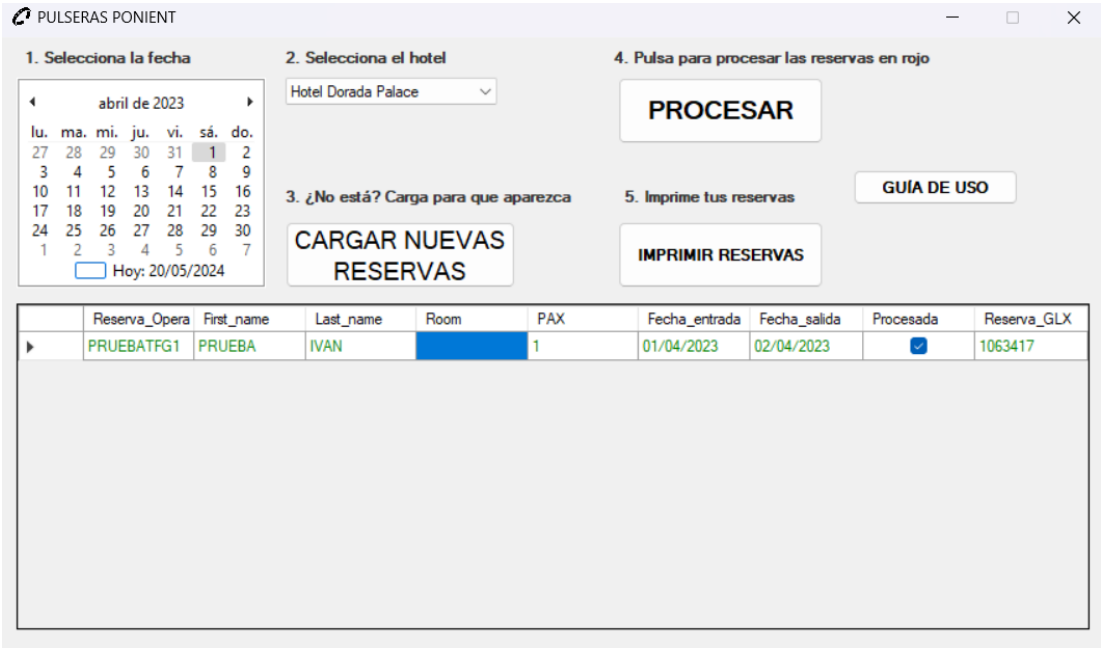

Figura 13. Aplicación de escritorio con una reserva procesada

6 Implementación

6.1 Sistema de base de datos

Para el desarrollo de este proyecto, se ha utilizado un software llamado SQL Server Management Studio 20 (SSMS en adelante) ya proporcionado por la empresa.

Figura 14. Icono del software SQL Server Management Studio

Esta es una herramienta desarrollada por Microsoft para la gestión de bases de datos SQL Server que proporciona al usuario diferentes características como pueden ser una interfaz gráfica de usuario, monitoreo del sistema, gestión de copias de seguridad e integración con otros SGBD.

Utilizando esta herramienta se han implementado diferentes partes del proyecto, siendo estas una base muy importante para el funcionamiento correcto de las dos aplicaciones, puesto que las dos deben tener conexión directa con la base de datos.

6.1.1 Base de datos y servidor utilizados

Con el objetivo de minimizar la utilización de nuevos recursos y evitar la generación de costes adicionales para la empresa, se ha decidido aprovechar la infraestructura ya existente. Esta estrategia implica el uso de un servidor y una base de datos ya disponibles, asegurando una gestión eficiente y económica de los recursos.

Para la gestión de datos, se utilizará un servidor denominado “R2D2”. Este servidor ya está en operación y se encarga de gestionar diversos datos relacionados con hoteles. Al reutilizar este servidor, se evita la necesidad de invertir en hardware adicional, aprovechando así los recursos ya disponibles de manera eficiente.

La información se integrará en una base de datos existente llamada “ClausOperaSICA”. Esta base de datos ya gestiona datos cruciales de hoteles, siendo la que aloja la gestión de datos de la interface que une Opera y SICA, gestionando las llaves de hoteles in-park y sus códigos de barras para que los clientes puedan entrar al parque, proporcionando un entorno adecuado y estructurado para almacenar y administrar la nueva información relacionada

con los accesos de los clientes de hoteles, en este caso Ponient, sin necesidad de crear una nueva base de datos desde cero.

6.1.2 Tabla Opera_Pulseras_HUM

Las aplicaciones deberán tener una conexión directa con los datos, y esta se hará a través de una tabla para el cumplimiento del requisito 4.2.2. El código de creación de la tabla proporcionado por SSMS es el siguiente:

```
USE [ClausOperaSICA]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Opera_Pulseras_HUM](
	[Hotel] [varchar](20) NOT NULL,
	[Reserva_Opera] [varchar](50) NOT NULL,
	[First_name] [varchar](80) NULL,
	[Last_name] [varchar](80) NULL,
	[Room] [varchar](20) NULL,
	[PAX] [int] NULL,
	[Fecha_entrada] [datetime] NULL,
	[Fecha_salida] [datetime] NULL,
	[PLU] [varchar](6) NULL,
	[Procesada] [bit] NULL,
	[Fecha_proceso] [datetime] NULL,
	[Reserva_GLX] [varchar](20) NULL,
	[Pensio] [varchar](10) NULL,
 CONSTRAINT [PK_Opera_Pulseras_HUM] PRIMARY KEY CLUSTERED
(
	[Reserva_Opera] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

A continuación, la explicación de los campos:

- **Hotel**: Cadena de texto indicando el hotel (variando entre MH1, MHB y MHD actualmente). Sin posibilidad de valores nulos ya que serán determinantes para el algoritmo a la hora de decidir qué producto le pertenece, ya que cada hotel tiene productos (entradas) diferentes.
- **Reserva_Opera**: Cadena de texto identificadora de la reserva en el sistema Opera. Sin posibilidad de valores nulos por ser el identificador único de la reserva en el sistema de hoteles.
- **First_name**: Cadena de texto indicando el nombre del titular de la reserva.
- **Last_name**: de texto indicando los apellidos del titular de la reserva.
- **Room**: Cadena de texto indicando el nombre/número de la habitación.
- **PAX**: Entero indicando el número de personas de la reserva.
- **Fecha_entrada**: Fecha y hora indicando el momento de entrada.
- **Fecha_salida**: Fecha y hora indicando el momento de salida.

- **PLU**: Cadena de texto indicando el identificador del producto perteneciente a la reserva en Galaxy.
- **Procesada**: Booleano indicando si la reserva ha estado creada en Galaxy o no. Este campo será "*False*" por defecto y luego se actualizará con las aplicaciones a True cuando se procese cada reserva.
- **Fecha_proceso**: Fecha y hora indicando el momento en el que se realiza la petición API creando la reserva en Galaxy. Este campo será NULL por defecto hasta que se genere la reserva en Galaxy.
- **Reserva_GLX**: Cadena de texto indicando el identificador de reserva de Galaxy devuelto por la petición API. Campo en NULL hasta su posterior actualización con cualquiera de las dos aplicaciones.
- **Pensio**: Cadena de texto indicando el tipo de pensión perteneciente a la reserva. Utilizado posteriormente por el sistema de restauración y tiendas para su posterior utilización en el sistema de puntos.

Estos campos podrán ser rellenados posteriormente gracias a la vista y al procedimiento almacenado de los siguientes apartados.

6.1.3 Vista PA_EXP_PULSERES_HUM4

El código de esta vista se podrá encontrar en el anexo confidencial del proyecto.

Creada por el técnico del sistema de hoteles, esta vista no se aloja en el mismo sistema de base de datos que la tabla, puesto que proviene de Oracle, el SGBD de Opera, pero sigue siendo esencial su explicación para el desarrollo del proyecto.

Esta vista recoge todas las reservas de Opera, proporcionando los datos principales de las reservas, como puede ser el hotel y el identificador de reserva, pero también realizando cálculos con las tarifas y las duraciones de cada una, proporcionando el PLU (identificador de producto en Galaxy), si es que lo contiene. Gracias a esto, se cumplen los requisitos 4.2.4, 4.4.1 y 4.6.2, puesto que se cumplen los requisitos para la vista y además se les da más rendimiento a las aplicaciones, teniendo solamente que recoger el producto en la tabla y no teniendo que realizar ningún cálculo.

Los campos que se recogen de esta vista son:

- **RESORT**: Cadena de texto indicando el hotel (variando entre MH1, MHB y MHD actualmente).
- **CONFIRMATION_NO**: Cadena de texto identificadora de la reserva en el sistema Opera.
- **GUEST_FIRST_NAME**: Cadena de texto indicando el nombre del titular de la reserva.
- **GUEST_LAST_NAME**: Cadena de texto indicando los apellidos del titular de la reserva.
- **ROOM**: Cadena de texto indicando el nombre/número de la habitación.
- **PAX**: Entero indicando el número de personas de la reserva.
- **BEGIN_DATE**: Fecha y hora indicando el momento de entrada.

- **END_DATE**: Fecha y hora indicando el momento de salida.
- **PLU**: Cadena de texto indicando el identificador del producto perteneciente a la reserva en Galaxy. Este campo deberá calcularse a partir de los datos incluidos en la reserva completa de Opera. En caso de no contenerlo, muestra "*ND*" (No dispone).
- **PENSIO**: Cadena de texto indicando el tipo de pensión perteneciente a la reserva.

Gracias a un "*linked server*" podremos utilizar los datos de esta vista con un procedimiento almacenado e insertarlos en la tabla para su consulta directa, explicando este en el siguiente apartado.

6.1.4 Procedimiento almacenado SP_GeneraReservasHUM

El código de este procedimiento almacenado se podrá encontrar en el anexo confidencial del proyecto.

Una vez tenemos un origen de datos (vista) y un destino donde almacenarlos (tabla) confeccionados, podemos ver el proceso de extracción y almacenaje de estos. Para ello, se desarrolla un Stored Procedure (Procedimiento almacenado en inglés) que necesitará de una fecha como parámetro para ejecutarse.

Una vez se obtiene esa fecha, se crean dos variables @InicioDelDia y @FinDelDia como DATETIME, que posteriormente serán convertidas al mismo día a las 00h y al mismo día a las 23:59h. Estas variables se utilizarán para filtrar el día en el que se querrán procesar las reservas posteriormente en las aplicaciones.

Seguidamente, se comienza con la sentencia de inserción en la tabla, generando una consulta en la que, mediante a un servidor vinculado, se obtienen todos los datos de la vista PA_EXP_PULSERES_HUM4 y se incluye como false el campo Procesada, la fecha y hora del momento de ejecución en Fecha_proceso y NULL en el campo de Reserva_GLX incluyendo el siguiente filtro:

```
WHERE
    PULS.PLU <> 'ND' AND
    PULS.BEGIN_DATE BETWEEN CONVERT(DATETIME, @InicioDelDia, 102) AND
CONVERT(DATETIME, @FinDelDia, 102) AND
    NOT EXISTS (
        SELECT 1
        FROM dbo.Opera_Pulseras_HUM AS OP
        WHERE OP.Reserva_Opera = PULS.CONFIRMATION_NO
    );
```

Este filtro lo que nos permite es solo recoger las reservas que contienen productos, en caso de no disponer de ninguno, se descartan y luego filtra según el día introducido como parámetro con las variables convertidas explicadas anteriormente.

Después se puede observar una subconsulta, con la que se asegura el no introducir valores repetidos en la tabla, para evitar errores de duplicidad de la clave primaria.

Con la ejecución de este procedimiento almacenado, completamos la extracción de datos de las reservas de Opera para poder tratarlos de manera adecuada con las aplicaciones y terminamos con los cimientos del proyecto, siendo estos todo el sistema de base de datos. También cumplimos los requisitos 4.2.4, 4.3.1, 4.5.2 e incluimos los casos de uso "*CDU 1*" y "*CDU 2*".

6.2 API de Galaxy

Gateway Ticketing Systems nos ofrece Galaxy, siendo este un conjunto de tecnologías preparado para la venta de entradas. Una parte de estas tecnologías es una amplia API SOAP integrada con el sistema que permite interactuar con el de una amplia gama de opciones.

6.2.1 API SOAP

SOAP (Simple Object Access Protocol) es un protocolo de mensajería basado en XML que se utiliza para intercambiar información estructurada entre aplicaciones a través de redes, típicamente usando HTTP/HTTPS.

Es utilizado principalmente en aplicaciones empresariales y sistemas que requieren alta seguridad, fiabilidad, y transacciones distribuidas, como servicios bancarios, integración de sistemas heredados y servicios web en grandes organizaciones.

Sus características principales son las siguientes:

- **Formato de mensaje**: Utiliza XML para definir el formato del mensaje. Un mensaje SOAP se compone de un sobre (Envelope) que contiene un encabezado (Header) opcional y un cuerpo (Body) obligatorio.
- **Protocolo independiente**: Aunque comúnmente se transporta sobre HTTP/HTTPS, SOAP puede ser usado con otros protocolos de transporte como SMTP, JMS, y más.
- **Estándares**: SOAP es un estándar de protocolo definido por la W3C, lo que asegura su interoperabilidad entre diferentes sistemas y plataformas.
- **Extensibilidad**: Permite la adición de características y funcionalidades mediante el uso de encabezados SOAP personalizados sin romper la compatibilidad con las aplicaciones existentes.
- **Seguridad y transacciones**: SOAP soporta protocolos de seguridad y transacciones (como WS-Security y WS-AtomicTransaction), lo que lo hace adecuado para aplicaciones empresariales que requieren niveles altos de seguridad y fiabilidad.

Gracias a estas características se cumplen varios requisitos no funcionales como el 4.4.2, 4.4.3, 4.6.4 o 4.6.5.

6.2.2 Comunicación con Galaxy

El servidor de Galaxy admite el intercambio de mensajes en tiempo real a través del Protocolo de transporte de hipertexto (HTTP) y admite comunicación en tiempo real a través de un "servicio web" sencillo respaldado por un servidor web integrado contenido dentro del servicio eGalaxy Server.

Es decir, la comunicación de las aplicaciones con Galaxy es a través de mensajes de solicitud en tiempo real a eGalaxy Server en formato XML mediante el comando HTTP POST y reciben una respuesta XML.

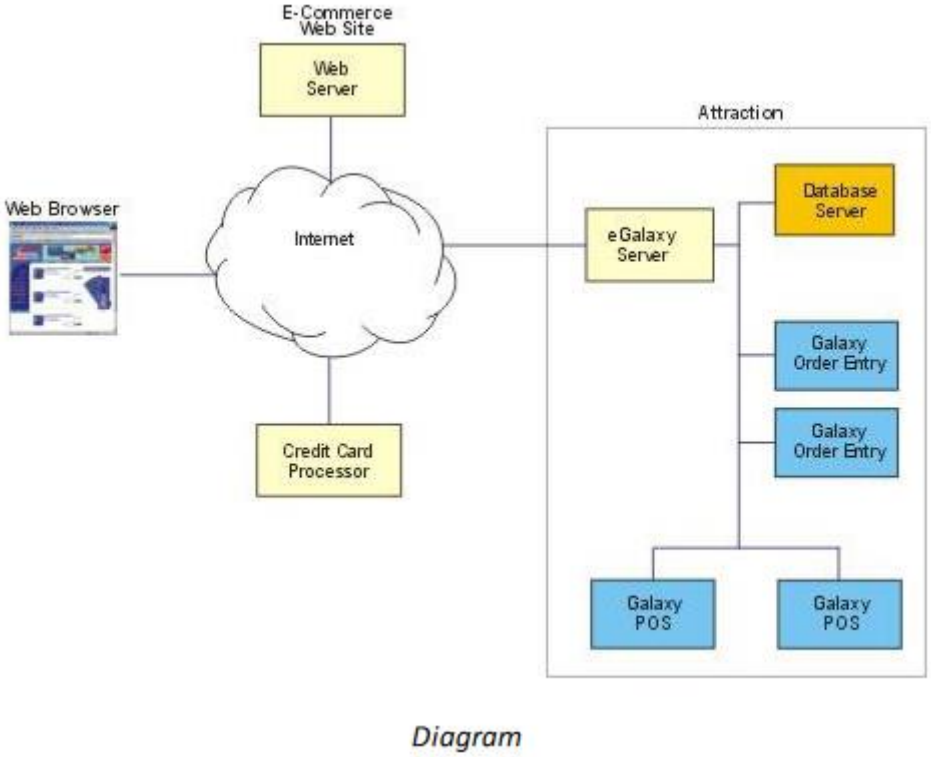

Figura 15. Diagrama de comunicación de Galaxy

Hay muchas opciones y posibilidades interesantes de interacción, pero la que nos interesa únicamente es la de creación de reservas.

Para crear la reserva, primero deberemos formar el encabezado de la petición:

```
<Header>
    <SourceID>TestSystem</SourceID>
    <MessageID>0</MessageID>
    <MessageType>Test</MessageType>
    <TimeStamp>2002-01-20 17:33</TimeStamp>
</Header>
```

Este encabezado puede tener más campos de texto, pero estos son los necesarios:

- **SourceID**: Identificador del sistema origen, configurado previamente en el servidor.

- **MessageID**: Número de mensaje incremental, que usaremos como un campo identificador.
- **MessageType**: Tipo de la petición que se envía, en el caso de la creación de reservas será Orders.
- **TimeStamp**: Momento del envío del mensaje.

Como las peticiones se realizarán desde dentro de la misma red de la compañía, no habrá ninguna necesidad de identificación previa. En el caso de que se hiciera desde una red externa, deberían introducirse unas credenciales, configuradas previamente, en el encabezado.

Para la parte del cuerpo tenemos el siguiente formato:

```
<Body>
  <Orders>
    <Order>
      <OrderID>"Identificador de reserva de Opera"</OrderID>
      <OrderCommand>ADD</OrderCommand>
      <CustomerID>"Número de cliente del hotel en Galaxy"</CustomerID>
      <OrderDate>"Fecha de creación de la reserva"</OrderDate>
      <OrderTotal>0</OrderTotal>
      <OrderStatus>2</OrderStatus>
      <PO>" Campo con la reserva de Opera utilizado para el cuadre de ventas "</PO>
      <OrderContact>
        <Contact>
          <FirstName>" Nombre del titular de la reserva "</FirstName>
          <LastName>" Apellido del titular de la reserva "</LastName>
        </Contact>
      </OrderContact>
      <PaymentContracts>
        <PaymentContract>
          <PaymentPlanID>10</PaymentPlanID>
        </PaymentContract>
      </PaymentContracts>
      <OrderProcessing>YES</OrderProcessing>
      <OrderLines>
        <OrderLine>
          <DetailType>1</DetailType>
          <Description>" Nombre del producto "</Description>
```

```
                <PLU>" Identificador de producto "</PLU>
                <Qty>" Numero de tickets "</Qty>
                <Amount>" Precio del ticket "</Amount>
                <Total>" Precio del ticket * PAX "</Total>
                <DiscountAmount>" Descuento por ticket (100% al ir empaquetado en Opera)
"</DiscountAmount>
            </OrderLine>
            <OrderLine>
                <DetailType>2</DetailType>
                <Amount>0</Amount>
                <PaymentCode>64</PaymentCode>
                <PaymentDate>" Fecha de pago "</PaymentDate>
            </OrderLine>
        </OrderLines>
    </Order>
  </Orders>
</Body>
</Envelope>
```

Los marcados en amarillo y los que varían durante cada reserva son:

- **OrderID**: Número de reserva de Opera. Será guardada en el ExternalID de la base de datos de Galaxy.
- **CustomerID**: Número de cliente del hotel en la configuración de Galaxy. Tiene las configuraciones necesarias para que se le puedan aplicar los descuentos del 100% posteriores para su producción y facturación en Opera.
- **OrderDate**: Fecha de creación de la reserva.
- **PO**: Campo “Purchase Order”, utilizado manualmente para cuadrar ventas. Aquí se introduce el número de reserva de Opera.
- **FirstName**: Nombre del titular de la reserva.
- **LastName**: Apellidos del titular de la reserva.
- **Description**: Descripción del producto vendido en la reserva.
- **PLU**: Identificador del producto vendido en la reserva.
- **Qty**: Cantidad de tickets. Igual al número de personas de la reserva, recogido del campo PAX de la tabla.
- **Amount**: Precio por ticket.
- **Total**: Precio total de la reserva.
- **DiscountAmount**: Descuento a aplicar en cada ticket.
- **PaymentDate**: Fecha y hora del pago (mismo que la creación al realizarse por petición API)

Otros campos importantes en la petición que siempre permanecerán iguales son:

- **OrderCommand**: Tipo de comando para la petición, en este caso, ADD, sirve para la creación de reservas.
- **OrderTotal**: Precio final de la reserva, siempre será 0, puesto que debe producir y facturar por Opera.
- **OrderStatus**: Estado de la reserva, siempre será 2, que quiere decir "Open", permitiendo así su impresión.
- **PaymentCode**: Código de la forma de pago, siendo 72 la forma de pago "OPERA" para su tratado en el cuadre de ventas.

Creando una reserva con los siguientes datos:

- **Hotel**: MH1
- **Reserva_Opera**: 123456Test
- **First_name**: Ivan
- **Last_name**: Test
- **Room**: 001
- **Fecha_entrada**: 2022-03-26 00:00:00.000
- **Fecha_salida**: 2022-03-27 00:00:00.000
- **PLU**: 954147
- **Procesada**: False
- **Fecha_proceso**: 2024-03-22 00:00:00.000
- **Reserva_GLX**: NULL
- **Pensio**: 1

Y realizando la petición API, la comunicación quedaría de la siguiente manera:

Petición:

```
<Envelope>
  <Header>
    <SourceID>API</SourceID>
    <MessageID>123456Test</MessageID>
    <TimeStamp>2024-05-27 17:46</TimeStamp>
    <MessageType>ORDERS</MessageType>
  </Header>
  <Body>
    <Orders>
      <Order>
        <OrderID>123456Test</OrderID>
        <OrderCommand>ADD</OrderCommand>
        <CustomerID>2667</CustomerID>
        <OrderDate>2024-05-27 17:46:22</OrderDate>
```

<OrderTotal>0</OrderTotal>
<OrderStatus>2</OrderStatus>
<ShipToContact>
<SameAsOrderContact>YES</SameAsOrderContact>
</ShipToContact>
<BillToContact>
<SameAsOrderContact>YES</SameAsOrderContact>
</BillToContact>
<UserFields>
<UserField1>NO</UserField1>
<UserField5></UserField5>
</UserFields>
<PO>123456Test</PO>
<OrderReference></OrderReference>
<OrderContact>
<Contact>
<FirstName>Ivan</FirstName>
<MiddleName></MiddleName>
<LastName>Test</LastName>
</Contact>
</OrderContact>
<PaymentContracts>
<PaymentContract>
<PaymentPlanID>10</PaymentPlanID>
</PaymentContract>
</PaymentContracts>
<OrderProcessing>YES</OrderProcessing>
<OrderLines>
<OrderLine>
<DetailType>1</DetailType>
<Description>Entrada MH1 2D2P FL</Description>
<PLU>954147</PLU>
<Qty>3</Qty>
<Amount>25,80</Amount>
<Total>77,4</Total>
<DiscountAmount>25,80</DiscountAmount>

```
            </OrderLine>
            <OrderLine>
               <DetailType>2</DetailType>
               <Amount>0</Amount>
               <PaymentCode>64</PaymentCode>
               <PaymentDate>2024-05-27 17:46:22</PaymentDate>
               <Endorsement>123456Test</Endorsement>
            </OrderLine>
         </OrderLines>
      </Order>
   </Orders>
</Body>
</Envelope>
```

Recibiendo esta respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope>
   <Header>
      <MessageID>-1</MessageID>
      <MessageType>SetOrderStatus</MessageType>
      <SourceID>API</SourceID>
      <TimeStamp>2024-05-27 17:46:39</TimeStamp>
      <EchoData></EchoData>
      <SystemFields></SystemFields>
   </Header>
   <Body>
      <OrderID>123456Test</OrderID>
      <GalaxyOrderID>1074471</GalaxyOrderID>
      <Status>2</Status>
      <Retry>NO</Retry>
      <PaymentContracts/>
   </Body>
</Envelope>
```

Y llegando a Galaxy de esta forma:

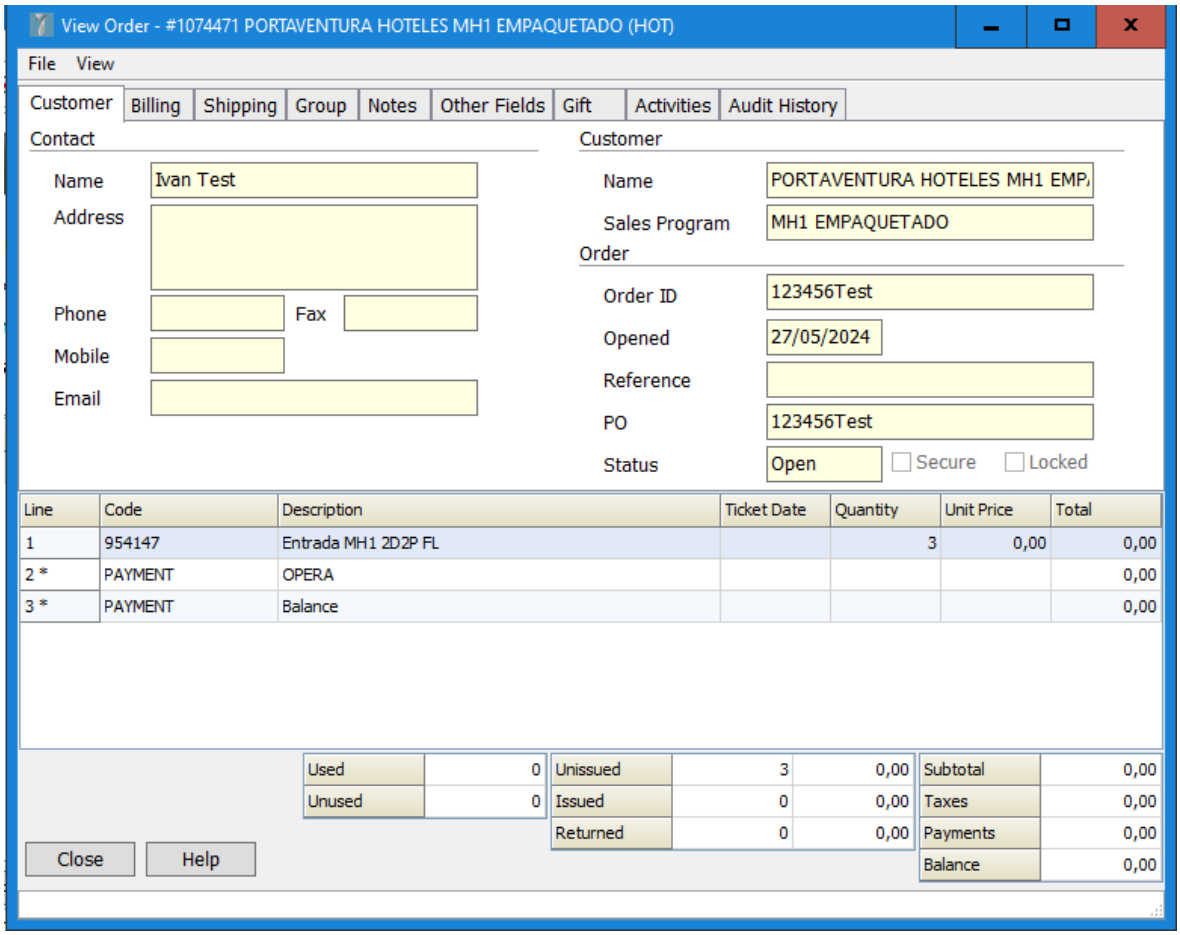

Figura 16. Reserva de prueba creada a través de una petición API.

De esta manera podemos observar en el marco de la parte superior el número de reserva de Galaxy 1074471, en el contacto el nombre del titular de la reserva, los datos del cliente de Galaxy configurados para el hotel MH1 y el número de reserva de Opera en el campo OrderID o PO.

Con este flujo se produce una comunicación entre las aplicaciones y Galaxy, pudiendo así crear de manera automática las reservas. Gracias a esto, cada ticket estará relacionado con la reserva de Galaxy donde podremos encontrar los datos necesarios y tener trazabilidad sobre estas entradas.

6.3 Entorno de desarrollo

Para el desarrollo de ambas aplicaciones de ha utilizado Visual Studio 2022.

Figura 17. Logo de Visual Studio 2022

Este es el IDE (Integrated Development Enviroment) de Microsoft, diseñado para desarrolladores profesionales y equipos de todos los tamaños. Este IDE dispone de numerosas mejoras en rendimiento, funcionalidad y facilidad de uso, adaptándose a las necesidades de desarrollo modernas.

Es una herramienta muy completa para el desarrollo de aplicaciones que nos permite múltiples características como pueden ser:

- Desarrollo multiplataforma.
- Depuración y diagnóstico.
- Colaboración y productividad.
- Compatibilidad con múltiples versiones de .NET.
- Extensibilidad.

Es el IDE elegido por la comodidad que presenta a la hora de desarrollar y el más familiarizado para la empresa a la hora de desarrollar aplicaciones con .NET Framework.

6.4 Código de la aplicación automática

El código de los archivos de la aplicación se encuentra en el anexo confidencial.

La aplicación automática empieza siendo un proyecto de aplicación de consola con Visual Basic .NET Framework 4.8 en Visual Studio 2022.

Esta tiene diferentes archivos que han sido parte del desarrollo:

- Module1.vb
- Reserva.vb
- PersonalException.vb
- App.config

6.4.1 Module1.vb

Este archivo es fundamental para el funcionamiento de la aplicación, ya que contiene el punto de entrada principal del programa, el main. En este archivo se encuentra la sucesión de código que va ejecutándose de manera secuencial y que define la lógica central de la aplicación.

Comienza con la importación de librerías que serán utilizadas a lo largo de la ejecución, las cuales son las siguientes:

- **System.Data.SqlClient**: Proporciona acceso a bases de datos SQL Server desde aplicaciones .NET. Utilizada para las consultas y ejecución del procedimiento almacenado en la base de datos.
- **System.Net.Http**: Facilita el envío de solicitudes HTTP y la recepción de respuestas de recursos web. Utilizado para la comunicación con Galaxy a través de peticiones API.
- **System.Text**: Ofrece clases para trabajar con codificaciones de texto y manipulación de cadenas. Utilizado para el tratado en UTF de las cadenas de texto.
- **System.Configuration**: Permite la gestión de configuraciones y ajustes de aplicaciones a través de archivos de configuración. Utilizado para complementarse con el archivo app.config explicado posteriormente.
- **System.IO**: Proporciona clases para trabajar con operaciones de entrada/salida, incluyendo archivos y flujos de datos. Utilizado tanto para la entrada/salida de la consola como para la gestión del log.

Gracias a estas librerías, se puede estructurar y ejecutar el código en varias etapas claramente definidas. El proceso comienza con la declaración de las variables necesarias para la ejecución del programa. Se inicia un registro de eventos (log) para documentar el proceso y facilitar la detección de problemas.

A continuación, se establece una conexión con la base de datos SQL Server utilizando la biblioteca **System.Data.SqlClient**. Se ejecuta un procedimiento almacenado, pasando como parámetro la fecha del día de ejecución, para obtener las reservas del día. Estas reservas se almacenan en una lista enlazada. Este tipo de estructura de datos se elige porque su tamaño puede adaptarse dinámicamente, evitando problemas de sobrecarga que podrían ocurrir con una tabla de tamaño fijo.

El código entra entonces en un bucle que envía peticiones API a Galaxy para crear las reservas, utilizando **System.Net.Http** para gestionar las solicitudes y respuestas HTTP. La respuesta de cada petición API es procesada para extraer el número de reserva asignado, y la tabla en la base de datos se actualiza con el número de reserva obtenido.

Una vez que todas las reservas han sido procesadas y la tabla actualizada, el programa finaliza. Durante todo el proceso, se manejan posibles excepciones. Si ocurre un error con alguna reserva, se registra el error y el programa continúa con la siguiente reserva.

Este flujo de trabajo asegura que las reservas diarias se gestionen de manera eficiente y robusta, con registro detallado de la ejecución y manejo de excepciones para asegurar la continuidad del proceso incluso en caso de errores.

6.4.2 Reserva.vb (Aplicación automática)

Este archivo se utiliza para definir la clase del objeto “Reserva”, fundamental en la gestión de las reservas en nuestro sistema. Al definir esta clase, podemos trabajar de manera estructurada y eficiente con objetos del tipo Reserva, cada uno de los cuales contiene una serie de propiedades esenciales para el proceso de creación de reservas en el sistema Galaxy a partir de las peticiones API.

Las propiedades definidas en la clase Reserva son las siguientes:

- **resort**
- **confirmationNo**
- **guestFirstName**
- **guestLastName**
- **pax**
- **plu**

Además de estas propiedades, la clase Reserva incluye un constructor que facilita la creación de instancias del objeto Reserva, inicializando las propiedades con los datos proporcionados. Este constructor asegura que todos los objetos Reserva se creen con la información necesaria desde el principio, evitando inconsistencias o datos incompletos.

Para facilitar el acceso y manipulación de los datos almacenados en los objetos Reserva, la clase también dispone de diferentes getters. Estos métodos permiten acceder de manera segura y controlada a las propiedades de la reserva, proporcionando una interfaz clara y consistente para interactuar con los datos.

En resumen, este archivo define una clase Reserva completa y bien estructurada, que encapsula toda la información relevante para la gestión de reservas en Galaxy. Al trabajar con objetos Reserva, podemos asegurar que los datos se manejen de manera eficiente y coherente, mejorando tanto la robustez del sistema como la experiencia del usuario.

6.4.3 PersonalException.vb

Este archivo define una clase especializada que hereda de la clase base **Exception**. Su propósito es permitir la creación y el manejo de excepciones personalizadas dentro del programa, proporcionando una manera más precisa y controlada de gestionar errores y condiciones excepcionales que pueden ocurrir durante la ejecución.

Al utilizar excepciones personalizadas, se mejora la claridad del código y se facilita la identificación de problemas. Cada excepción específica puede incluir mensajes detallados que describen la naturaleza del error, lo que permite una mejor comprensión y resolución de los problemas que puedan surgir, por lo que es una buena opción para el mantenimiento de la aplicación y la identificación de errores.

6.4.4 App.config (Aplicación automática)

Este archivo en formato XML despliega un papel crítico dentro de la arquitectura de la aplicación. Su contenido no solo abarca la cadena de conexión a la base de datos, sino que también contiene una serie de configuraciones vitales que pueden evolucionar con el tiempo, garantizando así la adaptabilidad y flexibilidad del sistema. Entre estas configuraciones se incluyen elementos como la URL a la cual se dirigen las peticiones API, la lista de hoteles disponibles, los clientes configurados en el sistema Galaxy para cada hotel y los productos disponibles en cada establecimiento.

Esta estructura modular y organizada permite una administración eficiente del mantenimiento de la aplicación. En lugar de tener que realizar cambios directamente en el código fuente, el responsable del mantenimiento puede simplemente editar este archivo

XML, siguiendo su estructura predefinida. Esto facilita enormemente la legibilidad y comprensión de las modificaciones realizadas, reduciendo así el riesgo de introducir errores durante el proceso de actualización.

La inclusión de la URL de la petición API en este archivo garantiza que cualquier cambio en la ubicación o en la configuración de la API pueda ser fácilmente reflejado en la aplicación sin necesidad de realizar cambios en el código. Del mismo modo, la lista de hoteles disponibles y los clientes configurados en el sistema Galaxy permiten una gestión centralizada de las entidades relevantes para la aplicación, simplificando el proceso de incorporación de nuevos hoteles, clientes o productos.

Además, la especificación de los productos disponibles por cada hotel dentro de este archivo ofrece una visión completa y detallada de la oferta de cada establecimiento, lo que facilita la gestión de inventario y la personalización de la experiencia del usuario.

En resumen, este archivo sirve como un punto central de configuración para la aplicación, proporcionando una plataforma flexible y fácil de mantener que se adapte a los cambios en la infraestructura de la aplicación y en las necesidades del negocio. Su estructura bien definida y su capacidad para reflejar los cambios en tiempo real hacen de él una herramienta indispensable para garantizar la eficacia y la escalabilidad del sistema a lo largo del tiempo.

6.5 Código de la aplicación de escritorio

El código de los archivos de la aplicación se encuentra en el anexo confidencial.

La aplicación automática empieza siendo un proyecto de aplicación de Windows Forms con Visual Basic .NET Framework 4.8 en Visual Studio 2022.

Esta tiene diferentes archivos que han sido parte del desarrollo:

- Form1.vb
- Reserva.vb
- Logger.vb
- App.config

6.5.1 Form1.vb

Este archivo es fundamental para la ejecución de la aplicación. En él se detallan todas las librerías y procesos que se ejecutarán desde el inicio hasta el fin de la ejecución.

Todo empieza con las librerías que se usan a lo largo de la ejecución de la aplicación:

- **System.Data.SqlClient**: Proporciona acceso a bases de datos SQL Server desde aplicaciones .NET. Utilizada para las consultas y ejecución del procedimiento almacenado en la base de datos.
- **System.Configuration**: Permite la gestión de configuraciones y ajustes de aplicaciones a través de archivos de configuración. Utilizado para complementarse con el archivo app.config explicado posteriormente.

- **System.Net.Http**: Facilita el envío de solicitudes HTTP y la recepción de respuestas de recursos web. Utilizado para la comunicación con Galaxy a través de peticiones API.
- **System.Text**: Ofrece clases para trabajar con codificaciones de texto y manipulación de cadenas. Utilizado para el tratado en UTF de las cadenas de texto.
- **System.Collections.Specialized**: Proporciona clases especializadas para la manipulación de colecciones de datos, enfocándose principalmente en estructuras de datos como diccionarios, listas y conjuntos que son dinámicas y pueden cambiar en tamaño durante la ejecución del programa. Utilizado para el tratado de la lista enlazada.
- **System.ComponentModel**: Proporciona herramientas y clases para el desarrollo de aplicaciones basadas en componentes. Su propósito principal es facilitar la creación, configuración, y gestión de componentes reutilizables en aplicaciones de escritorio, web y móviles. Utilizado para la ejecución del procedimiento almacenado en segundo plano bloqueando los botones.
- **System.IO**: Proporciona clases para trabajar con operaciones de entrada/salida, incluyendo archivos y flujos de datos. Utilizado tanto para la entrada/salida de la consola como para la gestión del log. Utilizado para la generación de los códigos de barras.
- **System.Drawing.Imaging**: Proporciona un conjunto de clases y funciones para trabajar con formatos de imagen, realizar operaciones de procesamiento de imágenes y manipular metadatos asociados a archivos de imagen. Utilizada para añadir los códigos de barra a los archivos PDF generados por la aplicación.
- **ZXing**: Proporciona herramientas para la creación y edición de códigos de barras. Utilizado para su creación
- **PdfSharp**: Proporciona diferentes elementos relacionados con la creación, edición y visualización de documentos PDF. Utilizada para la creación del PDF con los datos de las reservas.

Gracias a estas librerías se pueden crear las diferentes funciones que, combinadas, permiten la ejecución de la aplicación:

6.5.1.1 Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Esta es la función que se ejecuta con el inicio de la aplicación y cumple diferentes funciones, como el inicio de la gestión del log y la carga de los componentes principales.

Entre sus funciones se encuentra el bloqueo del tamaño de la ventana de la aplicación, la recogida de los datos del archivo de configuración y la carga de la cuadrícula de datos.

6.5.1.2 Form1_FormClosing(sender As Object, e As FormClosingEventArgs) Handles MyBase.FormClosing

Esta función espera a la finalización de la ejecución del thread del logger para completarse y cerrar la aplicación.

6.5.1.3 MonthCalendar1_DateChanged(sender As Object, e As DateRangeEventArgs) Handles MonthCalendar1.DateChanged

Esta función inicia el calendario guardando en la variable de fecha el día actual, pero detecta si hay algún cambio para almacenar ese dato para la posterior ejecución del Stored Procedure.

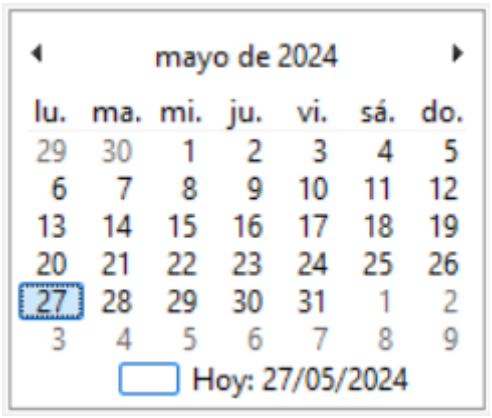

Figura 18. Calendario de la aplicación de escritorio

6.5.1.4 Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click (Botón de la guía de uso)

Controla cuando se pulsa sobre el botón de guía de inicio mostrando una ventana con el texto de guía.

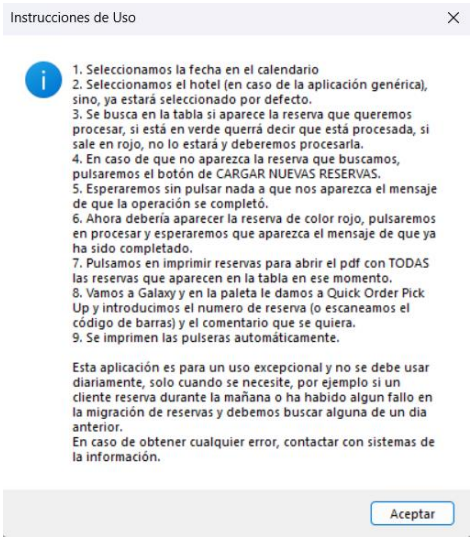

Figura 19. Ventana de la guía de uso

6.5.1.5 ComboBox1_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ComboBox1.SelectedIndexChanged

Controla si hay alguna selección diferente en la selección del desplegable de hoteles.

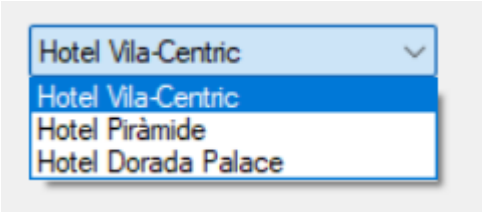

Figura 20. Desplegable de los hoteles

6.5.1.6 Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click

Esta función se controla cuando se hace clic en el botón de ”CARGAR RESERVAS”. Verifica si las variables de hotel y fecha tienen valor antes de ejecutar el procedimiento almacenado para evitar errores. Luego, deshabilita los componentes con los que se puede interactuar, excepto el de guía de uso. Finalmente, configura y ejecuta el trabajo en segundo plano a través del objeto backgroundWorker.

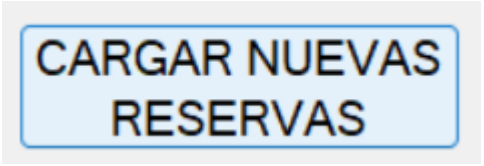

Figura 21. Botón de ejecución del procedimiento almacenado

6.5.1.7 backgroundWorker_DoWork(sender As Object, e As DoWorkEventArgs) Handles backgroundWorker.DoWork

Se ejecuta en segundo plano cuando se llama al método RunWorkerAsync() desde la función anterior. Realiza las operaciones de ejecución del procedimiento almacenado, donde se establece la conexión con la base de datos, se configura el comando SQL para llamar al procedimiento, se ejecuta y se informa sobre el progreso. En caso de errores, se registra un mensaje de error y se muestra una alerta al usuario.

6.5.1.8 backgroundWorker_RunWorkerCompleted(sender As Object, e As RunWorkerCompletedEventArgs) Handles backgroundWorker.RunWorkerCompleted

Esta función se activa cuando el trabajo en segundo plano ha terminado. Restaura el estado de los componentes deshabilitados a su estado original, muestra un mensaje de completado al usuario y carga la cuadrícula de datos.

6.5.1.9 Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click

Esta función controla la ejecución del botón “PROCESAR”. Una vez se pulsa, se consulta la base de datos buscando las reservas del día y del hotel seleccionados anteriormente y se guardan en una lista enlazada de tipo Reserva. A partir de ahí se recorre la lista realizando las peticiones hacia Galaxy creando las reservas, posteriormente consultando la respuesta recibida y actualizando la tabla de la base de datos, marcando las reservas como procesadas e indicando el número identificador de reserva en Galaxy.

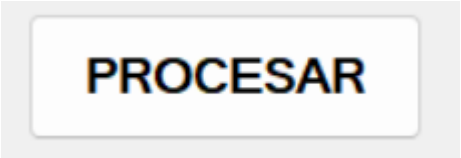

Figura 22. Botón de procesar de la aplicación de escritorio

6.5.1.10 DataGridView1_RowPrePaint(sender As Object, e As DataGridViewRowPrePaintEventArgs) Handles DataGridView1.RowPrePaint

Esta función consulta la columna Reserva_GLX de la cuadrícula de datos de reservas y comprueba si es nulo o contiene valor. En caso de que contenga valor, se pinta toda la fila de color verde; en caso de ser nulo, se pinta de color rojo, indicando si las reservas están o no procesadas de una manera más visual.

6.5.1.11 CargarTablaInicio()

Función que se ejecuta en la carga de la aplicación consultando la base de datos y mostrando las reservas disponibles del día y hotel en la tabla de la base de datos.

6.5.1.12 CargarTablaCambio(fechaSeleccionada As DateTime, resort As String)

Esta función se ejecuta al cambiar de fecha en el calendario o al cambiar de hotel en el desplegable. Como la función anterior, muestra las reservas disponibles del día y del hotel en la tabla de la base de datos, pero esta vez cuando hay algún cambio, pasando por parámetro estos posibles cambios.

6.5.1.13 GenerarPDF()

Esta función utiliza las librerías de PDFSharp permitiendo la creación y almacenaje de un archivo PDF con los datos de 8 reservas por página, mostrando el identificador de reserva de Opera como un código de barras.

6.5.1.14 GenerateBarcode(data As String, width As Integer, height As Integer) As Bitmap

Función que permite la creación de códigos de barras a través de los valores pasados como parámetro y utilizada en la función anterior.

6.5.1.15 Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click

Función que controla interacción con el botón de “IMPRIMIR RESERVAS” donde se genera el PDF ejecutando la función de GenerarPDF().

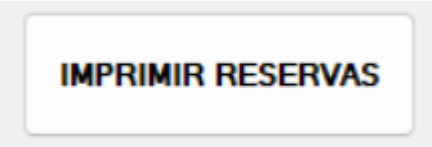

Figura 23. Botón de imprimir reservas de la aplicación de escritorio

6.5.2 Reserva.vb (Aplicación de escritorio)

Cuenta con la misma estructura que en la aplicación automática, incluyendo las mismas propiedades, constructor y funciones “getters”. Estructura base del objeto Reserva que nos permite el tratado de cada una de las reservas durante toda la ejecución.

6.5.3 Logger.vb

Esta clase proporciona funcionalidades para el registro de eventos y la escritura de registros en archivos de registro.

Sus variables son:

- **logQueue**: Cola utilizada para almacenar la información del registro antes de que se escriba en el archivo de registro.
- **isRunning**: Flag que indica si el hilo encargado de escribir en el archivo de registro está en ejecución.
- **logThread**: Un objeto de hilo que se utilizará para ejecutar el método ***EscribirEnLog***.
- **soloFecha**: Una cadena que almacena la fecha actual en formato "yyyy-MM-dd".

Sus funciones son las siguientes:

6.5.3.1 AgregarAlLog()

Permite agregar información al registro. Toma un parámetro de tipo String que representa la información a ser registrada y la encola en logQueue.

6.5.3.2 EscribirEnLog()

Este método se ejecuta en un hilo dedicado y se encarga de escribir la información en el archivo de registro, verifica si la carpeta de registros (LOGS) existe en el directorio de la aplicación. Si no existe, la crea, itera continuamente mientras isRunning sea verdadero y verifica si hay información en logQueue, si hay información en la cola, la extrae y la escribe en el archivo de registro utilizando un StreamWriter.

6.5.3.3 StopLogging()

Detiene la escritura en el archivo de registro estableciendo isRunning en falso. Si el hilo de registro (logThread) está vivo, espera a que termine.

6.5.3.4 WaitForThreadToFinish

Espera a que el hilo de registro (logThread) termine. Esto se logra llamando al método Join**()** en el hilo de registro.

6.5.3.5 StartLoggingThread

Inicia el hilo para escribir en el archivo de registro llamando al método EscribirEnLog.

Este conjunto de funciones proporciona una forma segura y eficiente de registrar información en archivos de registro, utilizando un hilo dedicado para escribir los registros y asegurando la integridad de los datos registrados.

6.5.4 App.config (Aplicación de escritorio)

Sigue una estructura muy similar a la de la aplicación automática, como se mencionó anteriormente. Al igual que en la aplicación automática, este archivo XML mantiene una organización fácil, sencilla y centralizada de los datos con posibles variaciones. Esto significa que se utiliza para almacenar configuraciones y datos que pueden cambiar con el tiempo, como las cadenas de conexión a bases de datos, las descripciones de los productos y otros ajustes de la aplicación.

Al separar estas configuraciones del código fuente de la aplicación, el archivo app.config permite ajustar el comportamiento de la aplicación sin necesidad de recompilar el código. Esta flexibilidad facilita la adaptación de la aplicación a diferentes entornos y requisitos específicos sin afectar su funcionalidad principal.

Además, al seguir una estructura similar a la de la aplicación automática, el archivo app.config de la aplicación de escritorio promueve la coherencia y la reutilización de configuraciones entre diferentes partes de la solución de software. Esto contribuye a una gestión más eficiente y consistente de la configuración en toda la aplicación, lo que simplifica el mantenimiento y la personalización de esta.

7 Juego de pruebas

Para el juego de pruebas se proponen un total de 18 casos: 9 casos diferentes para cada aplicación. En los diferentes casos presentados se va a simular la creación de diferentes reservas de Opera con diferentes características, en las que se va a explicar su ejecución, el resultado esperado y el resultado obtenido, apoyándonos en los resultados del log.

7.1 Juego de pruebas de la aplicación automática

7.1.1 Caso 1: Reserva individual existente

Se intenta procesar una reserva la cual su número de reserva de Opera ya existe en Galaxy.

Datos de la reserva en base de datos:

- Hotel: MH1
- Reserva_Opera: TestIvan_1
- First_name: Prueba
- Last_name: Ivan
- Room: 111
- PAX: 1
- Fecha_entrada: 28-05-2024 00:00:00
- Fecha_salida: 29-05-2024 00:00:00
- PLU: 954116
- Procesada: False
- Fecha_proceso: 28-05-2025 00:00:00
- Reserva_GLX: NULL
- Pensio: 1

Resultado esperado: La reserva TestIvan_1 ya existe en Galaxy y no se crea una nueva.

Resultado obtenido: Correcto: La reserva TestIvan_1 ya existe en Galaxy y no se crea una nueva.

Respuestas en el log: <ErrorText>Order (TestIvan_1) already exists in Galaxy</ErrorText>

7.1.2 Caso 2: Grupo de reservas con una existente

Se intentan procesar 3 reservas con las mismas características, pero con un identificador de Opera diferentes. Para la reserva ya existente se utilizará TestIvan_1, ya utilizada en el caso anterior y dos más que serán TestIvan_2 y TestIvan_3.

Resultado esperado: La reserva TestIvan_1 ya existe en Galaxy y no se crea una nueva, pero las reservas TestIvan_2 y TestIvan_3 se crean sin ningún problema.

Resultado obtenido: Correcto: La reserva TestIvan_1 ya existe en Galaxy y no se crea una nueva, pero las reservas TestIvan_2 y TestIvan_3 se crean sin ningún problema.

Respuestas en el LOG:

<ErrorText>Order (TestIvan_1) already exists in Galaxy</ErrorText>

<GalaxyOrderID>944057</GalaxyOrderID>

<GalaxyOrderID>944058</GalaxyOrderID>

7.1.3 Caso 3: Reserva individual con producto inexistente

Se intenta procesar una reserva TestIvan_4 normal y corriente, pero se incluye un PLU inexistente:

Datos de la reserva:

- Hotel: MH1
- Reserva_Opera: TestIvan_4
- First_name: Prueba
- Last_name: Ivan
- Room: 111
- PAX: 1
- Fecha_entrada: 28-05-2024 00:00:00
- Fecha_salida: 29-05-2024 00:00:00
- PLU: 909090
- Procesada: False
- Fecha_proceso: 28-05-2025 00:00:00
- Reserva_GLX: NULL
- Pensio: 1

Resultado esperado: La reserva TestIvan_4 no se crea en Galaxy porque contiene un producto inexistente en el archivo app.config.

Resultado obtenido: Correcto: La reserva TestIvan_4 no se crea en Galaxy porque contiene un producto inexistente en el archivo app.config

Respuestas en el LOG:

La descripción no existe en la configuración.

7.1.4 Caso 4: Grupo de reservas y una con producto inexistente

Se intentan procesar las reservas TestIvan_4, TestIvan_5 y TestIvan_6, siendo la primera la anterior con producto inexistente y las otras dos reservas correctas.

Resultado esperado: La reserva TestIvan_4 no se crea en Galaxy porque contiene un producto inexistente en el archivo app.config, pero las reservas TestIvan_5 y TestIvan_6 se crean correctamente en Galaxy.

Resultado esperado: Correcto: La reserva TestIvan_4 no se crea en Galaxy porque contiene un producto inexistente en el archivo app.config. pero las reservas TestIvan_5 y TestIvan_6 se crean correctamente en Galaxy.

Respuestas en el LOG:

La descripción no existe en la configuración.

<GalaxyOrderID>944061</GalaxyOrderID>

<GalaxyOrderID>944062</GalaxyOrderID>

7.1.5 Caso 5: Reserva individual con hotel inexistente

Se intenta procesar una reserva TestIvan_7 que contiene un hotel inexistente “MHZ” en la configuración.

Datos de la reserva:

- Hotel: MHZ
- Reserva_Opera: TestIvan_7
- First_name: Prueba
- Last_name: Ivan
- Room: 111
- PAX: 1
- Fecha_entrada: 28-05-2024 00:00:00
- Fecha_salida: 29-05-2024 00:00:00
- PLU: 909090
- Procesada: False
- Fecha_proceso: 28-05-2025 00:00:00
- Reserva_GLX: NULL
- Pensio: 1

Resultado esperado: La reserva no se crea en Galaxy porque contiene un hotel inexistente en el archivo app.config.

Resultado obtenido: Correcto: La reserva no se crea en Galaxy porque contiene un hotel inexistente en el archivo app.config.

Resultados en el LOG:

El resort no existe en la configuración.

7.1.6 Caso 6: Grupo de reservas y una con hotel inexistente

Se intentan procesar la reserva TestIvan_7 del caso anterior con hotel erróneo y TestIvan_8 y TestIvan_9 siendo estas dos últimas correctas.

Resultado esperado: La reserva no se crea en Galaxy porque contiene un hotel inexistente en el archivo app.config, pero las reservas TestIvan_7 y TestIvan_8 se crean sin ningún problema.

Resultado obtenido: Correcto: La reserva no se crea en Galaxy porque contiene un hotel inexistente en el archivo app.config, pero las reservas TestIvan_7 y TestIvan_8 se crean sin ningún problema.

Resultados en el LOG:

El resort no existe en la configuración.

<GalaxyOrderID>944063</GalaxyOrderID>

<GalaxyOrderID>944064</GalaxyOrderID>

7.1.7 Caso 7: Intento de ejecución fuera de la red de PortAventura

Se intenta procesar una reserva TestIvan_10 totalmente correcta pero fuera de la red de PortAventura, habiendo copiado la carpeta de la aplicación en local.

Resultado esperado: La aplicación no llega a ejecutar el Procedimiento Almacenado y finaliza.

Resultado obtenido: Correcto: La aplicación no llega a ejecutar el Procedimiento Almacenado y finaliza.

Resultados en el LOG:

Algo ha fallado en la ejecución del Stored Procedure: Error relacionado con la red o específico de la instancia mientras se establecía una conexión con el servidor SQL Server. No se encontró el servidor o éste no estaba accesible. Compruebe que el nombre de la instancia es correcto y que SQL Server está configurado para admitir conexiones remotas. (provider: Named Pipes Provider, error: 40 - No se pudo abrir una conexión con SQL Server)

7.1.8 Caso 8: Reserva individual correcta

Se intenta procesar una reserva individual TestIvan_10 correcta.

Resultado esperado: La aplicación se ejecuta correctamente sin ningún problema y la reserva se crea en Galaxy.

Resultado obtenido: Correcto: La aplicación se ejecuta correctamente sin ningún problema y la reserva se crea en Galaxy.

Resultados en el LOG:

<GalaxyOrderID>944065</GalaxyOrderID>

7.1.9 Caso 9: Grupo de reservas correctas

Se intentan procesar tres reservas TestIvan_10, TestIvan_11 y TestIvan_12 correctas.

Resultado esperado: La aplicación se ejecuta correctamente sin ningún problema y las reservas se crean en Galaxy.

Resultado obtenido: Correcto: La aplicación se ejecuta correctamente sin ningún problema y las reservas se crean en Galaxy.

Resultados en el LOG:

<GalaxyOrderID>944066</GalaxyOrderID>

<GalaxyOrderID>944067</GalaxyOrderID>

<GalaxyOrderID>944068</GalaxyOrderID>

7.2 Juego de pruebas de la aplicación de escritorio

7.2.1 Caso 1: Reserva individual existente

Se intenta procesar una reserva la cual su número de reserva de Opera ya existe en Galaxy.

Datos de la reserva en base de datos:

- Hotel: MH1

- Reserva_Opera: TestIvan_13
- First_name: Prueba
- Last_name: Ivan
- Room: 111
- PAX: 1
- Fecha_entrada: 28-05-2024 00:00:00
- Fecha_salida: 29-05-2024 00:00:00
- PLU: 954116
- Procesada: False
- Fecha_proceso: 28-05-2025 00:00:00
- Reserva_GLX: NULL
- Pensio: 1

Resultado esperado: La reserva TestIvan_13 ya existe en Galaxy y no se crea una nueva.

Resultado obtenido: Correcto: La reserva TestIvan_13 ya existe en Galaxy y no se crea una nueva.

Respuestas en el log: <ErrorText>Order (TestIvan_13) already exists in Galaxy</ErrorText>

7.2.2 Caso 2: Grupo de reservas con una existente

Se intentan procesar 3 reservas con las mismas características, pero con un identificador de Opera diferentes. Para la reserva ya existente se utilizará TestIvan_13, ya utilizada en el caso anterior y dos más que serán TestIvan_14 y TestIvan_15. Luego se generará el PDF.

Resultado esperado: La reserva TestIvan_13 ya existe en Galaxy y no se crea una nueva, pero las reservas TestIvan_14 y TestIvan_15 se crean sin ningún problema.

Resultado obtenido: Correcto: La reserva TestIvan_13 ya existe en Galaxy y no se crea una nueva, pero las reservas TestIvan_14 y TestIvan_15 se crean sin ningún problema.

Respuestas en el LOG:

<ErrorText>Order (TestIvan_13) already exists in Galaxy</ErrorText>

<GalaxyOrderID>944069</GalaxyOrderID>

<GalaxyOrderID>944070</GalaxyOrderID>

PDF Generado.

7.2.3 Caso 3: Reserva individual con producto inexistente

Se intenta procesar una reserva TestIvan_16 normal y corriente, pero se incluye un PLU inexistente:

Datos de la reserva:

- Hotel: MH1
- Reserva_Opera: TestIvan_16
- First_name: Prueba
- Last_name: Ivan

- Room: 111
- PAX: 1
- Fecha_entrada: 28-05-2024 00:00:00
- Fecha_salida: 29-05-2024 00:00:00
- PLU: 909090
- Procesada: False
- Fecha_proceso: 28-05-2025 00:00:00
- Reserva_GLX: NULL
- Pensio: 1

Resultado esperado: La reserva TestIvan_16 no se crea en Galaxy porque contiene un producto inexistente en el archivo app.config.

Resultado obtenido: Correcto: La reserva TestIvan_16 no se crea en Galaxy porque contiene un producto inexistente en el archivo app.config

Respuestas en el LOG:

La descripción no existe en la configuración.

7.2.4 Caso 4: Grupo de reservas y una con producto inexistente

Se intentan procesar las reservas TestIvan_16, TestIvan_17 y TestIvan_18, siendo la primera del anterior caso con producto inexistente y las otras dos reservas correctas. A continuación se intentará generar el PDF.

Resultado esperado: La reserva TestIvan_16 no se crea en Galaxy porque contiene un producto inexistente en el archivo app.config, pero las reservas TestIvan_17 y TestIvan_18 se crean correctamente en Galaxy.

Resultado esperado: Correcto: La reserva TestIvan_16 no se crea en Galaxy porque contiene un producto inexistente en el archivo app.config, pero las reservas TestIvan_17 y TestIvan_18 se crean correctamente en Galaxy.

Respuestas en el LOG:

La descripción no existe en la configuración.

<GalaxyOrderID>944071</GalaxyOrderID>

<GalaxyOrderID>944072</GalaxyOrderID>

PDF Generado.

7.2.5 Caso 5: Reserva individual con hotel inexistente

Se intenta procesar una reserva TestIvan_19 que contiene un hotel inexistente “MHZ” en la configuración.

Datos de la reserva:

- Hotel: MHZ
- Reserva_Opera: TestIvan_19
- First_name: Prueba
- Last_name: Ivan

- Room: 111
- PAX: 1
- Fecha_entrada: 28-05-2024 00:00:00
- Fecha_salida: 29-05-2024 00:00:00
- PLU: 909090
- Procesada: False
- Fecha_proceso: 28-05-2025 00:00:00
- Reserva_GLX: NULL
- Pensio: 1

Resultado esperado: La reserva no se crea en Galaxy porque contiene un hotel inexistente en el archivo app.config.

Resultado obtenido: Correcto: La reserva no se crea en Galaxy porque contiene un hotel inexistente en el archivo app.config.

Resultados en el LOG:

El resort no existe en la configuración.

7.2.6 Caso 6: Grupo de reservas y una con hotel inexistente

Se intentan procesar la reserva TestIvan_19 del caso anterior con hotel erróneo y TestIvan_20 y TestIvan_21 siendo estas dos últimas correctas. Luego se intenta generar el PDF.

Resultado esperado: La reserva TestIvan_19 no se crea en Galaxy porque contiene un hotel inexistente en el archivo app.config, pero las reservas TestIvan_20 y TestIvan_21 se crean sin ningún problema y se genera el PDF.

Resultado obtenido: Correcto: La reserva TestIvan_19 no se crea en Galaxy porque contiene un hotel inexistente en el archivo app.config, pero las reservas TestIvan_20 y TestIvan_21 se crean sin ningún problema. Y se genera el PDF.

Resultados en el LOG:

El resort no existe en la configuración.

<GalaxyOrderID>944073</GalaxyOrderID>

<GalaxyOrderID>944074</GalaxyOrderID>

PDF Generado.

7.2.7 Caso 7: Intento de ejecución fuera de la red de PortAventura

Se intenta procesar una reserva TestIvan_22 totalmente correcta pero fuera de la red de PortAventura, habiendo copiado la carpeta de la aplicación en local.

Resultado esperado: La aplicación no llega a ejecutar el Procedimiento Almacenado y finaliza.

Resultado obtenido: Correcto: La aplicación no llega a ejecutar el Procedimiento Almacenado y finaliza.

Resultados en el LOG:

Algo ha fallado en la ejecución del Stored Procedure: Error relacionado con la red o específico de la instancia mientras se establecía una conexión con el servidor SQL Server. No se encontró el servidor o éste no estaba accesible. Compruebe que el nombre de la instancia es correcto y que SQL Server está configurado para admitir conexiones remotas. (provider: Named Pipes Provider, error: 40 - No se pudo abrir una conexión con SQL Server)

7.2.8 Caso 8: Reserva individual correcta

Se intenta procesar una reserva individual TestIvan_23 correcta y se intenta generar el PDF.

Resultado esperado: La aplicación se ejecuta correctamente sin ningún problema y la reserva se crea en Galaxy y se genera el PDF.

Resultado obtenido: Correcto: La aplicación se ejecuta correctamente sin ningún problema y la reserva se crea en Galaxy y se genera el PDF.

Resultados en el LOG:

<GalaxyOrderID>944075</GalaxyOrderID>

PDF Generado.

7.2.9 Caso 9: Grupo de reservas correctas

Se intentan procesar tres reservas TestIvan_24, TestIvan_25 y TestIvan_26 correctas y se intenta generar el PDF.

Resultado esperado: La aplicación se ejecuta correctamente sin ningún problema, las reservas se crean en Galaxy y se genera el PDF.

Resultado obtenido: Correcto: La aplicación se ejecuta correctamente sin ningún problema, las reservas se crean en Galaxy y se genera el PDF.

Resultados en el LOG:

<GalaxyOrderID>944076</GalaxyOrderID>

<GalaxyOrderID>944077</GalaxyOrderID>

<GalaxyOrderID>944078</GalaxyOrderID>

PDF Generado.

8 Prueba piloto

Primero, se pasa a producción la aplicación automática, la cual se ejecuta en un servidor llamado LISA. El equipo de sistemas asigna una tarea de ejecución diaria a esta aplicación, programada para que se ejecute cada día a las 00:30 horas. Además, se le asigna un sistema de alertas propio de la compañía, diseñado para enviar un correo automático de aviso en caso de que aparezca la palabra "error" en algún archivo de la carpeta de logs.

Una vez realizada esta implementación en producción, durante los primeros 15 días, se revisan diariamente los logs para comprobar que la aplicación se ha ejecutado con éxito. Si se detectan errores, se procede a buscar y aplicar las correcciones necesarias para resolverlos.

En cuanto a la aplicación de escritorio, se crea una carpeta de red específica y se otorgan los permisos necesarios a los trabajadores que la utilizarán. La aplicación se instala en el hotel Ponient Vila-Centric, elegido por su menor volumen de clientes, lo que permite una ejecución más controlada y sin grandes volúmenes de datos.

Se explica a los responsables del hotel y a los trabajadores la nueva operativa a seguir y se inicia su uso. Durante los primeros 15 días, se revisan diariamente los logs y se buscan soluciones para cualquier incidencia reportada, asegurando así una implementación suave y efectiva de la aplicación de escritorio.

El resultado de estas pruebas piloto fue positivo, ya que solo se encontró un error en el periodo principal de 15 días. Este error era en ambas aplicaciones y el origen se encontraba en el procedimiento almacenado que extrae las reservas de Opera. Las fechas desde Oracle tenían un formato incorrecto de fechas en comparación al usado por SQL-Server por lo que un simple formateo de fechas corrigió el error y no se encontró ninguno más. Después de esto se pudo seguir con la nueva operativa a la perfección.

9 Evaluación de costes

En esta sección se detallan los costes asociados al desarrollo de dos aplicaciones, especificando tanto las tarifas horarias como el tiempo total estimado de trabajo. Este análisis proporciona una visión clara de los recursos financieros necesarios para llevar a cabo el proyecto, permitiendo una planificación adecuada y una evaluación realista de los gastos. Además, se consideran los gastos adicionales en licencias y hardware necesarios en caso de que la empresa no disponga de estos recursos previamente.

9.1 Desglose de Costes

El coste del desarrollo de cada aplicación se ha calculado en función de una tarifa horaria de 50 euros. Se estima que el desarrollo completo de cada aplicación requiere un total de 150 horas de trabajo aproximadamente, siendo un total de 300h, incluyendo requerimientos y toma de decisiones.

Las licencias, hardware y otro tipo de costes no se tienen en cuenta porque la empresa ya dispone de estos, pero se detallan a continuación:

- **Licencias de Galaxy**: Cada licencia de este sistema tiene un coste de 1980 euros, incluyendo el sistema de punto de venta, el módulo de reservas, el gestor de eventos y el módulo de pases.
- **Licencia de SQL-Server**: Las licencias de este sistema general de bases de datos en el paquete standard tienen un coste de 3600 euros por dos cores. El servidor utilizado por PortAventura para estas aplicaciones dispone de 4 cores, por lo que el coste total es de 7200 euros.

A continuación, se presenta un desglose detallado de estos costes.

9.2 Tarifa Horaria

La tarifa horaria de 50 euros se ha determinado como decisión del director del trabajo de fin de grado.

9.3 Tiempo de Desarrollo

El tiempo estimado para completar el proyecto es de 300 horas aproximadamente. Este cálculo se basa en una planificación aproximada de las tareas involucradas, que incluyen:

- **Análisis y Planificación**: 20 horas
- **Diseño de la Aplicación**: 30 horas
- **Desarrollo de la Aplicación**: 70 horas
- **Pruebas y Depuración**: 20 horas
- **Implementación y Revisión Final**: 10 horas

9.4 Cálculo Total de Costes

El coste total para el desarrollo de cada aplicación se puede calcular de la siguiente manera:
Coste Total Desarrollo= Tarifa Horaria × Número de Horas
Coste Total Desarrollo = 50 euros/hora × 150 horas = 7500 euros

Por lo tanto, el desarrollo cada aplicación tiene un coste de 7500 euros. Dado que se trata de un desarrollo de dos aplicaciones, el coste total combinado sería:
Coste Total Desarrollo Combinado = 7500 euros × 2 = 15000 euros

En caso de haber tenido en cuenta las licencias necesarias para el desarrollo y paso a producción del proyecto se podría calcular de la siguiente manera:
Coste Total Galaxy = Coste Licencia Galaxy × Número De Puntos De Venta Galaxy
Coste Total Galaxy = 1980 × 3 = 5940 euros

Coste Total BD = Cose Licencia Core × Número Cores
Coste Total BD = 1800 × 4 = 7200 euros

Coste Total Licencias = Coste Total Galaxy + Coste Total BD
Coste Total Licencias = 5940 + 7200 = 13140 euros

Sumando los costes del desarrollo con la tarifa horaria y los costes de licencias:
Coste Total = Coste Total Desarrollo Combinado + Coste Total Licencias
Coste Total = 15000 + 13140 = 28140 euros.

En resumen, el coste del proyecto es de 15000 euros sin contar las licencias, por lo que la compañía, en lugar de pagar 28140 euros, se ha ahorrado 13140 euros teniendo únicamente el coste del primer importe.

9.5 Justificación de los Costes

Es fundamental justificar estos costes para asegurar la viabilidad y la sostenibilidad del proyecto. La tarifa de 50 euros por hora es razonable y competitiva en el contexto del desarrollo de aplicaciones, teniendo en cuenta la calidad del trabajo y el nivel de especialización requerido. Además, el tiempo estimado de 150 horas por aplicación es adecuado para cubrir todas las fases del desarrollo de manera exhaustiva.
Se tienen en cuenta únicamente los costes de desarrollo por las horas empleadas, puesto que la empresa ya dispone de las licencias y el hardware necesario para la implementación del proyecto.

10 Legislación y protección de datos

En este proyecto, la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSI) no aplica. La LSSI regula aspectos legales relacionados con los servicios de la sociedad de la información y del comercio electrónico en España, como la gestión de sitios web, tiendas en línea y comunicaciones comerciales electrónicas. Sin embargo, este proyecto no involucra ninguno de estos elementos, ya que no se trata de un sitio web, ni de un comercio electrónico, ni de una comunicación comercial electrónica. Por lo tanto, las disposiciones de la LSSI no son relevantes en este contexto.

En cuanto al Reglamento General de Protección de Datos (RGPD), este proyecto no introduce nuevos datos personales. El RGPD es una normativa de la Unión Europea que establece cómo deben tratarse los datos personales de los ciudadanos, cubriendo aspectos como la recolección, almacenamiento y uso de datos personales. En este proyecto, solo se consultan datos ya existentes en el sistema de hoteles. Este sistema ya cumple con la normativa del RGPD, lo que significa que los datos están siendo tratados de acuerdo con las regulaciones vigentes. Por lo tanto, al no manejar nuevos datos y utilizar únicamente datos que ya están protegidos según la normativa, el proyecto cumple con los requisitos del RGPD.

Finalmente, en lo que respecta a la propiedad intelectual, las dos aplicaciones desarrolladas en este proyecto son creación del Iván Menacho Domínguez y los derechos de explotación y copia pertenecen a la compañía PortAventura World.

11 Conclusiones

El desarrollo de dos aplicaciones, tanto automática como manual, ha resultado ser un acierto en múltiples aspectos. La aplicación automática, programada para ejecutarse diariamente y agilizar las operaciones de los recepcionistas, ha sido fundamental para mejorar la eficiencia operativa. Por otro lado, la aplicación manual, inicialmente concebida para casos excepcionales, ha demostrado ser excepcionalmente útil en el día a día, facilitando la revisión de reservas y la impresión de códigos de barras para agilizar aún más las operaciones.

Este proyecto ha sido una oportunidad invaluable para seguir desarrollando habilidades como programador, explorando campos como bases de datos y comunicación entre sistemas. Aunque enfrenté desafíos significativos durante el desarrollo de la aplicación manual, con cada prueba superada hemos logrado mejorar y optimizar la aplicación paso a paso. Especialmente, el paso a producción en el hotel Ponient Vila-Centric, al ser un hotel con menos afluencia, ha permitido una implementación más tranquila y efectiva, lo que ha facilitado la solución de errores y la mejora continua.

Quizá se podría decir que la parte más complicada de todas ha sido el desarrollo de la aplicación de escritorio. Aun teniendo la funcionalidad hecha con anterioridad en la aplicación automática, el tener que controlar tantas escenas que se pueden producir al ser esta una aplicación hecha para que la use un trabajador se ha hecho difícil porque a medida que avanzaba la aplicación, salían más posibilidades a cubrir. Aun así es realmente gratificante poder ver el impacto positivo que ha tenido el primer proyecto que he propuesto y desarrollado en la empresa en un corto período de tiempo desde mi incorporación (menos de un año). Recibir la confianza de la compañía para el desarrollo de un proyecto con tal impacto es realmente gratificante.

Este proyecto ha tenido un impacto significativo en varios aspectos, tanto para el cliente como para la empresa y los trabajadores involucrados. Desde el punto de vista del cliente, el impacto se refleja en la mejora de la experiencia de usuario al poder tener sus entradas listas en el momento del check-in del hotel. Esta agilidad operativa genera una impresión positiva en el cliente y contribuye a una mejor percepción de los servicios ofrecidos.

Además, el proyecto ha permitido mejorar la trazabilidad de las entradas tarificadas en las reservas de los hoteles Ponient. Esta trazabilidad es fundamental para la empresa, ya que facilita el control y análisis del uso de estas entradas, al tiempo que ayuda a prevenir fraudes como el cambio de productos a formato de pulsera.

Desde mi perspectiva, el impacto más significativo radica en la mejora de la eficiencia y productividad de los trabajadores de recepción. Gracias a las aplicaciones desarrolladas, los trabajadores ya no tienen que realizar tareas manuales y tediosas como consultar las entradas en el sistema Opera y luego registrarlas en el sistema Galaxy. Esto ha permitido agilizar enormemente sus operaciones diarias y les ha dado la oportunidad de ser mucho más productivos durante su jornada laboral.

Personalmente, recibir el reconocimiento y los elogios de los trabajadores por la utilidad de la aplicación que he desarrollado es una de las mejores sensaciones que puedo experimentar. Saber que mi trabajo está teniendo un impacto positivo en la vida laboral de mis compañeros y en la operativa general de la empresa es increíblemente gratificante y me motiva aún más a seguir desarrollando soluciones innovadoras y útiles en el futuro.

12 Índice de figuras

Principal:

Anexos:

13 Recursos utilizados

1. PortAventura World
   https://www.portaventuraworld.com/
2. Ponient Hotels
   https://www.ponienthotels.com/
3. Gateway Ticketing
   https://www.gatewayticketing.com/
4. Inetum
   https://www.inetum.com/es
5. Visual Studio
   https://visualstudio.microsoft.com/es/vs/
6. .NET Framework
   https://learn.microsoft.com/en-us/dotnet/framework/Magic Draw
7. Magic Draw
   https://www.magicdraw.com/shop
8. PDFSharp
   https://docs.pdfsharp.net/PDFsharp/Overview/About.html

14 Anexos

14.1 Manual de uso Aplicación de Escritorio

Para realizar la nueva operativa para la entrega de entradas empaquetadas a los hoteles Ponient, primero debes ir al escritorio y ejecutar el archivo PulserasHUM.exe, una vez abierto, se debe abrir la siguiente ventana:

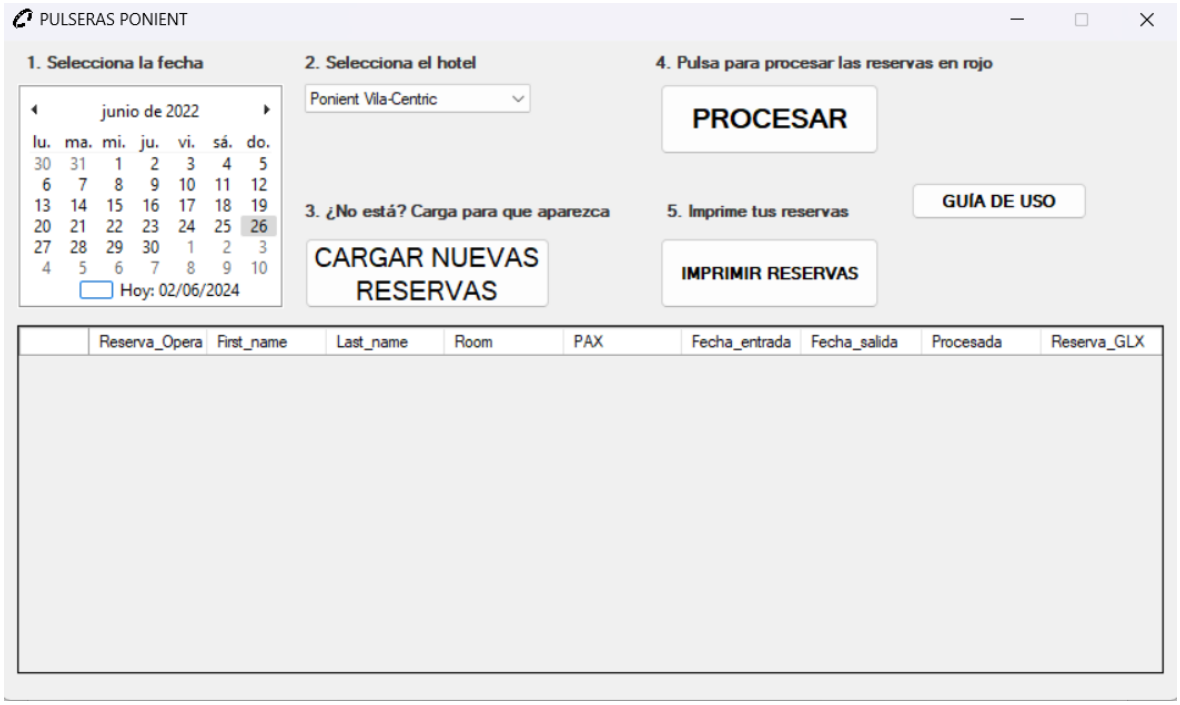

Figura Extra 1. Ventana principal de la aplicación de escritorio

En el desplegable superior ya debe aparecer el hotel en el que estamos operando, en caso contrario, se debe elegir.

En el calendario de arriba a la izquierda debes seleccionar la fecha de entrada de las reservas que se quieran emitir, en caso de no aparecer en la cuadrícula inferior, debes pulsar en el botón “CARGAR NUEVAS RESERVAS”, tras varios minutos aparecerá en la cuadrícula inferior de color rojo:

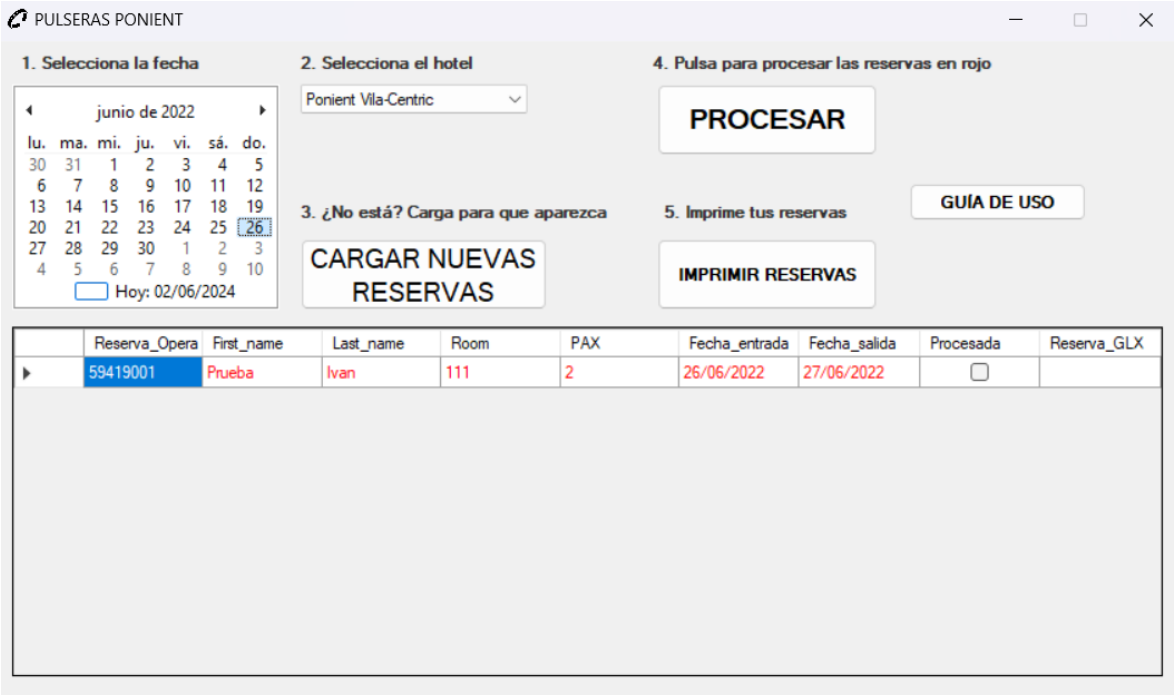

Figura Extra 2. Reserva sin procesar

Una vez se tiene la reserva de Opera deseada en la cuadrícula, es la hora de generarla en Galaxy. Para ello, se pulsa en el botón de procesar y saldrá el mensaje de que la operación se ha completado con éxito:

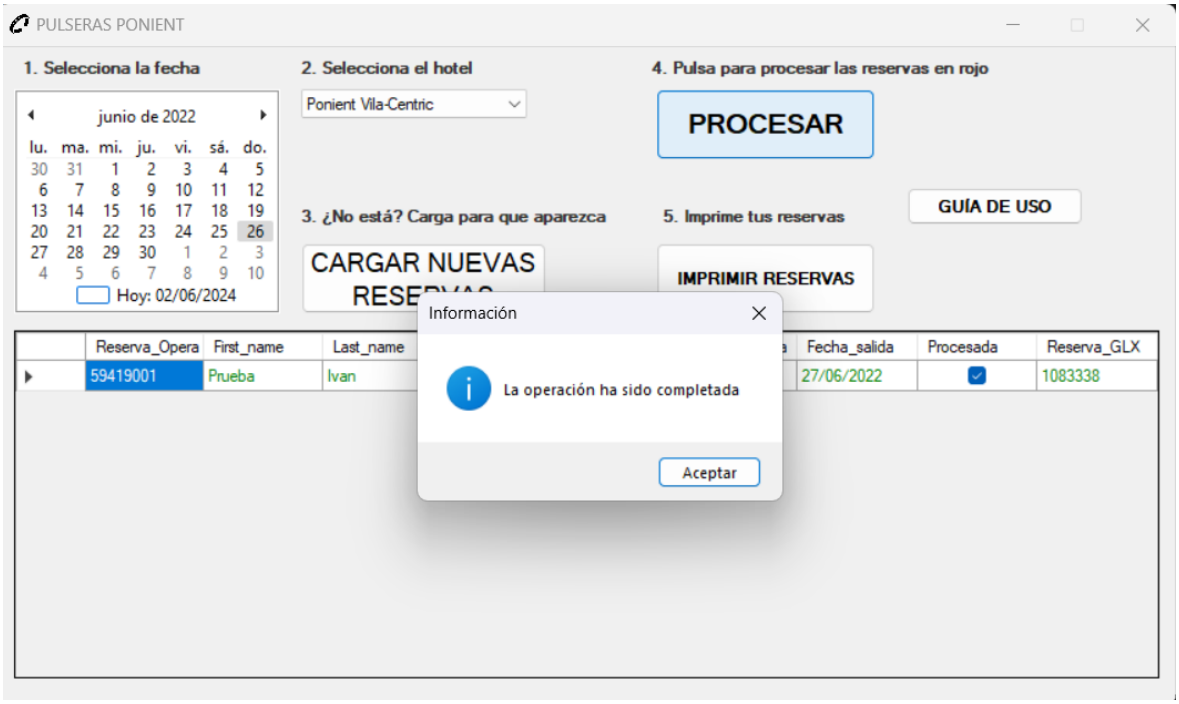

Figura Extra 3. Aparición del mensaje de operación completada

Ahora deberá aparecer de color verde, esto quiere decir que la reserva ya ha estado cargada en Galaxy, por lo tanto, pulsaremos el botón de imprimir reservas y se abrirá un documento PDF de este estilo:

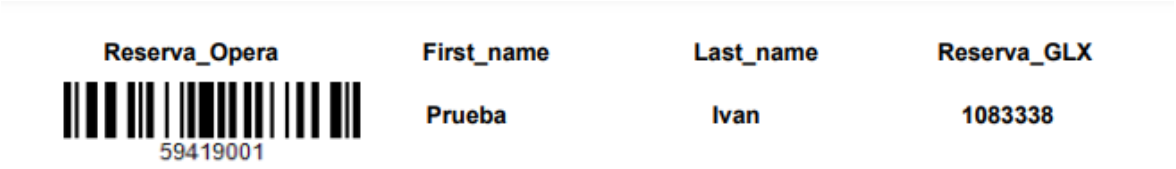

Figura Extra 4. Líneas impresas en el PDF con las reservas

Una vez tenemos estos datos, nos vamos a la paleta de nuestro dispositivo Galaxy y buscamos el botón "Quick Order Pickup":

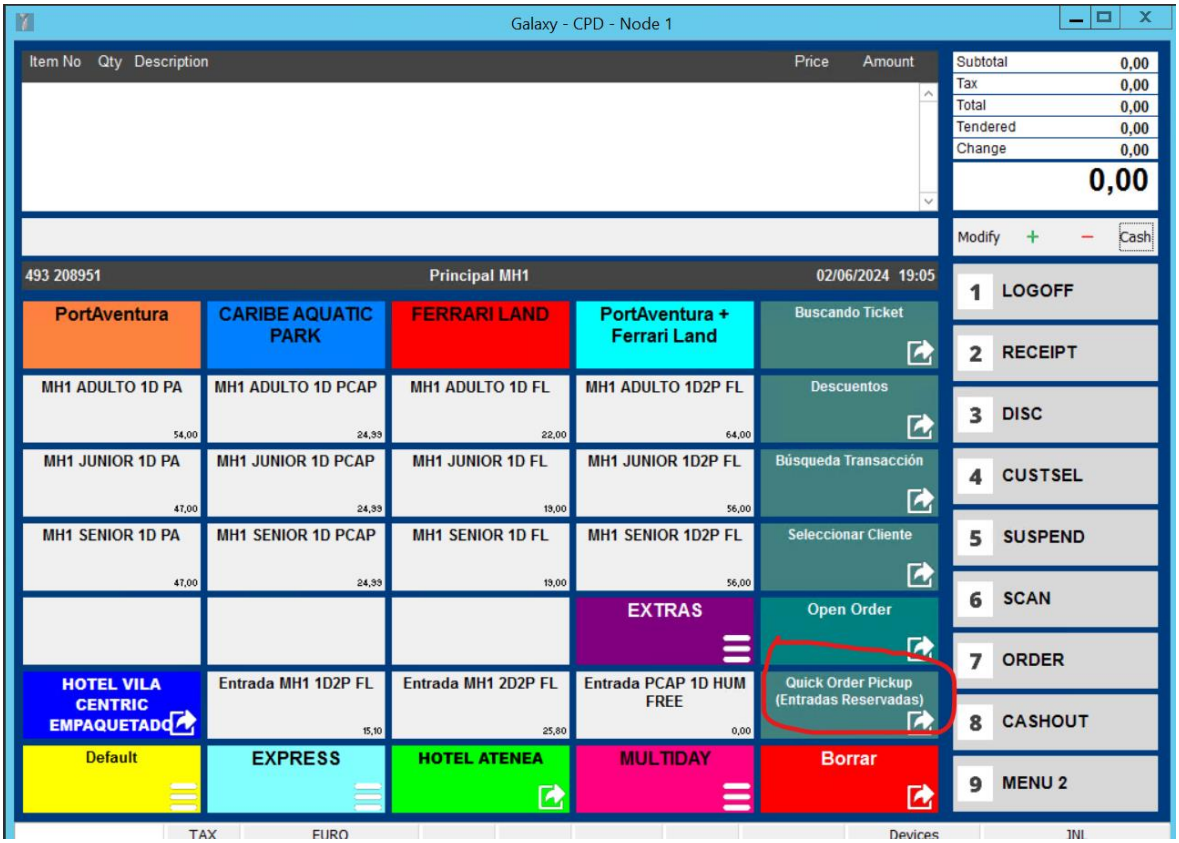

Figura Extra 5. Punto de venta de Galaxy

Una vez pulsado ahí, aparecerá la siguiente ventana, en la que deberemos introducir el número de reserva que queramos, el de Galaxy o el de Opera (pudiendo escanear el QR del PDF).

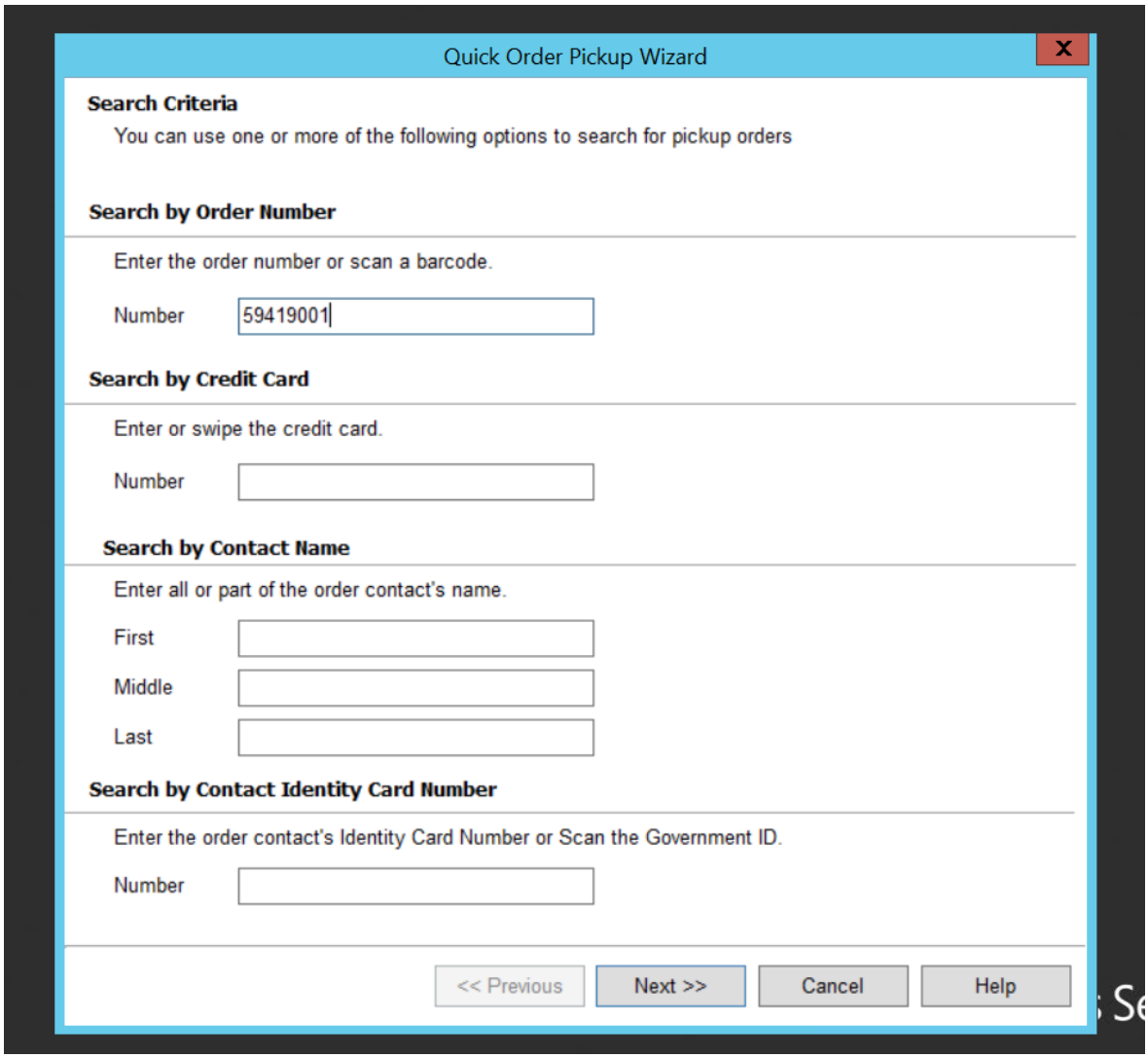

Figura Extra 6. Función Galaxy para impresión rápida

Una vez introducido, pulsaremos finalizar y se imprimirán las entradas por la impresora asignada:

Figura Extra 7. Entradas de ejemplo generadas con la aplicación

En caso de obtener cualquier error durante la operativa se deberá enviar un mail al equipo de Sistemas de la Información o llamar a tecnología. Mientras no se soluciona, se deberá realizar la operativa antigua, imprimiendo las entradas manualmente desde el punto de venta de Galaxy.