TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Diseño y Desarrollo de un Sistema de Control de Reservas Hoteleras

Curso 2018/2019

Alumno/a:

Adrián Rodríguez Escudero

Director/es:

Rafael Guirado Clavijo

Ana Olivares García

Índice de contenido

Índice de figuras

Índice de códigos

1. Introducción

Este proyecto se va a desarrollar en el seno de una Beca Talento D-UAL en la empresa Senator Hotels & Resorts [1]. Esta empresa se dedica al sector hotelero y gestiona una gran cantidad de hoteles por toda España y el Caribe.

Estos hoteles reciben miles de reservas a través de diversas plataformas tales como *Booking* [2], *Expedia* [3], *Mirai* [4], *OTS* [5]... Para llevar un control eficiente de las reservas, la empresa dispone de un servidor que las recoge desde las diferentes plataformas que les dan servicio para su posterior gestión centralizada.

Debido a la importancia de las reservas y su eficiente gestión, si el servidor falla durante un tiempo (por ejemplo, 10 minutos), se pueden llegar a perder reservas, con el gran impacto económico que ello conlleva. Para evitar estas pérdidas, la empresa dispone de servidores en la Nube en los que se realizan réplicas. Aun así, puede haber pérdidas, ya que la aplicación que realiza dichas réplicas puede fallar también. El objetivo de este proyecto consiste en comprobar que las plataformas web de venta como Booking, *Expedia*, *Mirai* y *OTS*, entre otras, tengan la información de los establecimientos de la empresa Senator Hotels & Resorts realmente actualizada.

Para ello se tendrá que comparar, de manera continuada, los datos de precios y disponibilidad de los servidores con las distintas plataformas de venta.

Únicamente se comprobarán ciertas "fechas calientes" que corresponderán a eventos importantes en la zona (fiestas, ferias, congresos...), que son las más susceptibles a pérdidas si ocurriera cualquier disparidad en la información. Por ejemplo, si no se ha actualizado correctamente un cierre de ventas, se puede dar lugar a lo que se conoce como "overbooking".

Cada plataforma posee su propio protocolo de comunicación, por lo que, en base a su API particular, se debe desarrollar una solución personalizada. La comunicación va a consistir en el envío y recepción de mensajes XML a unos determinados puntos de control previamente establecidos.

	Obtención	**Actualización**
Mirai	getinventory	webservice_updater
Booking	roomrateavailability	availability
OTS	OTA_HotelAvailGetRQ	OTA_HotelRatePlanNotifRQ
Expedia	ProductAvailRateRetrievalRS	AvailRateUpdateRQ

Tabla 1: Lista de APIs utilizadas para las diferentes plataformas.

En la tabla 1 podemos observar los diferentes nombres que tienen las distintas APIs que utilizaremos, ya sean los de la API de *Booking* [6], de *Expedia* [7], de *Mirai* [8] o la de *OTS* [9].

Las APIs destinadas a la obtención de datos nos devolverán en un XML, pasándole previamente unos datos, la información sobre las tarifas y la disponibilidad del hotel seleccionado en unas fechas concretas.

Por el contrario, las APIs destinadas a la actualización de datos nos devolverán en un XML indicando si la actualización ha sido correcta o no. Para ello tendremos que crear un documento XML concreto para cada plataforma al que le indicaremos el hotel, habitación, fechas, disponibilidad y tarifa a actualizar.

Por otro lado, está aplicación también comprobará la información dentro del sistema. En esta empresa hay varios niveles de base de datos:

- Servidor local bajo un entorno de AS400 [10], con base de datos basada en DB2 [11].
- Servidor en la nube con SQL Server.

Siguiendo la misma idea de "fechas calientes", se comprobará que los datos en esas fechas sean los mismos en ambos servidores (AS400 con Nube).

La aplicación notificará las diferencias que encuentre en AS400 respecto a la nube y en la nube respecto a las plataformas y las arreglará, insertando nuevos registros, eliminándolos o modificándolos según corresponda. Se programará un registro para informar sobre las acciones realizadas, las advertencias encontradas, así como los errores de base de datos.

La aplicación está basada en una estructura modular [12] la cual le dota de una capacidad de extensión de nuevas funciones e incorporaciones de nuevas plataformas por si, por ejemplo, la empresa realiza altas de hoteles en nuevos sitios web de reserva online y hay que conectar la aplicación a esta web, dotándole de un mejor soporte y mantenimiento al software. La estructuración modular es un paradigma de programación que consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable.

Senator Hotels & Resorts

La entidad colaboradora en la que me encuentro disfrutando de mi Beca Talento D-UAL se conoce comúnmente como **Grupo Hoteles Playa S.A.**, pero en 2017 la identidad de la marca corporativa pasó a ser Senator Hotels & Resorts.

La organización de la marca estará en dos grupos preestablecidos que son:

- **Producto Hotelero.** La principal oferta está basada en hoteles vacacionales pero como la empresa se ha expandido cuenta con hoteles de línea urbana conocidos como Senator y los centros de salud y belleza Senzia Spa & Wellness.
- **Marcas del grupo.** Incursión en otros segmentos de negocio como una agencia de viajes Viajes Rossell, el parque temático de Oasys Minihollywood que se encuentra en Tabernas o el Aquarium de Roquetas de Mar.

Nos centraremos en clasificar los diferentes tipos de hoteles:

- **Hoteles Senator.** Son hoteles urbanos cuya localización es estratégica, situándose en pleno centro de las ciudades. Están repartidos por toda España: Barcelona, Cádiz, Granada, Huelva, Madrid, Málaga, Murcia, Sevilla y Valencia.
- **Hoteles Playa.** Son hoteles vacacionales caracterizados por su ubicación en primera linea de playa y sus grandes piscinas tematizadas con jacuzzis. Solo se encuentran en la comunidad andaluza.
- **Diverhoteles.** Son hoteles tematizados en Almería y Marbella que ofrecen la mejor diversión para la familia.
- **Apartamentos Playa.** Son apartamentos vacacionales de nueva construcción que se encuentran junto a la playa. Hay uno en Vera y el otro está en Huelva.
- **Caleia.** Se encuentra en Mallorca, en primera línea de la playa de Cala Millor. Es un hotel dirigido para adultos, ideal para estancias con pareja y amigos.

Características de la empresa

Dentro de este punto podemos explicar la actividad profesional, las instalaciones y también el número de trabajadores que tiene la empresa.

El carácter profesional de Senator Hotels & Resorts es la de una cadena de gestión y mantenimiento hotelero que empezó con hoteles vacacionales y cuya expansión de Almería al ámbito nacional le hizo incluir hoteles urbanos como los que nos encontramos en Madrid. Posteriormente se ha expandido a República Dominicana y a Cancún.

Debido a lo extenso que resultaría detallar cada una de las instalaciones de las que dispone la empresa usaré un listado con los diferentes hoteles de cada una de las provincias. Este listado se puede observar en la figura 1.

Figura 1: Hoteles de Senator Hotels & Resorts.

Más adelante se incluirán aquí los hoteles de República Dominicana y Cancún.

En cuanto a recursos humanos podemos comentar el número de trabajadores de la empresa, donde se realiza una media de los mismos al haber bastantes hoteles de carácter vacacional, donde oscila sobre los 1250 trabajadores durante todo el año. Su pico más alto se produce en verano, alcanzando los 4500 trabajadores.

2. Planificación del proyecto

Hemos tenido en cuenta ciertas estrategias de planificación de proyectos informáticos para poder optimizar y gestionar de la manera más eficiente el tiempo dedicado al proyecto en todas sus fases. Para ello, hemos utilizado técnicas basadas en procedimientos ágiles [13] que facilitan la gestión organizativa del proyecto y sus tareas. Una de estas técnicas disponibles en la actualidad y que hemos utilizado en este proyecto ha sido *Scrum* [14].

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario

identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija (iteraciones que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4 semanas, límite máximo de feedback de producto real y reflexión). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

2.1. Planificación de la iteración

El primer día de la iteración se realiza la reunión de planificación de la iteración. Tiene dos partes:

1. **Selección de requisitos** (2 horas). El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que prevé que podrá completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

2. **Planificación de la iteración** (2 horas). El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos seleccionados. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se autoasignan las tareas, se autoorganizan para trabajar incluso en parejas (o grupos mayores) con el fin de compartir conocimiento (creando un equipo más resiliente) o para resolver juntos objetivos especialmente complejos.

2.2. Inspección y adaptación

El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

1. **Revisión (demostración)** (1,5 horas). El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.

2. **Retrospectiva** (1,5 horas). El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Facilitador se encargará de eliminar o escalar los obstáculos identificados que estén más allá del ámbito de acción del equipo.

2.3. Tiempos por fase

Para la realización del proyecto se ha estimado ciertas fases con sus tiempos dedicados. Estas fases son las siguientes:

- **Planificación** (28 horas). Fase en la que se realizará una planificación correcta y eficiente del proyecto.
- **Análisis** (20 horas). Fase en la que se analizará el problema detenidamente para poder abordarlo eficaz y eficientemente, evitando así estar continuamente buscando la solución.
- **Diseño** (10 horas). Fase en la que se planteará un diseño de la interfaz optimizando los componentes gráficos y teniendo en cuenta los puntos más importantes de las interfaces de usuario, tales como la usabilidad de la aplicación y más aspectos de diseño.
- **Desarrollo** (90 horas). Fase en la que se creará el código que generará la solución al problema.
- **Depuración** (36 horas). Fase en la que se analizan y corrigen errores del desarrollo.
- **Optimización** (37 horas). Fase en la que se busca la eficiencia del desarrollo para determinar una solución más rápida y "ligera" (menor consumo de recursos).
- **Desarrollo de la memoria del proyecto** (81 horas). Fase en la que se elaborará la memoria completa del proyecto, así como el anteproyecto.
- **Mantenimiento** (indefinido). Fase indefinida temporalmente, ya que siempre se realizará un soporte del proyecto mientras dure mi contrato laboral.

Para nuestro caso, hemos utilizado la tecnología SCRUM mediante la herramienta online ScrumDo a la cual se puede acceder mediante el siguiente enlace:

https://www.scrumdo.com

Para el desarrollo del proyecto solo se ha necesitado de un desarrollador y un supervisor o jefe de proyecto, rol que ha desempeñado mi tutora de la Beca Talento D-UAL. Mi labor ha sido la de desarrollador, creando toda la aplicación, y la de mi tutora ha sido, mediante reuniones periódicas con ella, la de supervisar que mi trabajo era correcto, tanto para el cálculo de los precios como para el de disponibilidades o consultas sobre la estructura de las bases de datos de la empresa.

Hemos necesitado tres iteraciones, la primera de ellas para preparar la clase principal con la interfaz y la estructura para realizar las comprobaciones. La segunda para la comprobación local y la tercera para la comprobación web.

Tal y como podemos observar en la figura 2 tenemos nuestro tablero incompleto, puesto que se trata de una captura realizada en la mitad del desarrollo del proyecto. En este tablero se puede ver a simple vista las principales tareas y las subtareas del desarrollo del proyecto.

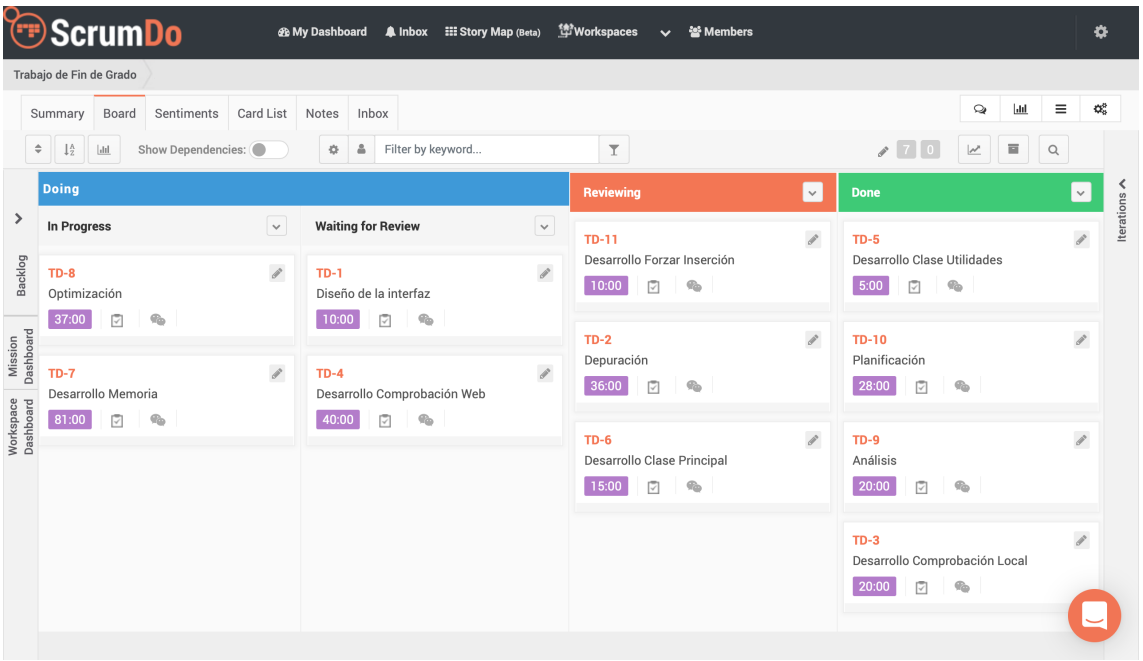


Figura 2: Herramienta ScrumDo.

3. Hardware utilizado

Todo el desarrollo basado en el servidor local a través de la aplicación es gracias al hardware utilizado y facilitado por la empresa IBM debido a que Senator Hotels & Resorts mantiene una estrecha relación con dicha empresa ya que posee uno de sus servidores.

International Business Machines Corporation (IBM) [15] es una reconocida empresa multinacional estadounidense de tecnología y consultoría con sede en Armonk, Nueva York. IBM fabrica y comercializa hardware y software para computadoras, y ofrece servicios de infraestructura, alojamiento de Internet, y consultoría en una amplia gama de áreas relacionadas con la informática, desde computadoras centrales hasta nanotecnología.

3.1. Servidor AS400

IBM introdujo el sistema AS/400 [16] en 1988. Era un sistema integrado, compuesto por un hardware (AS/400) y un sistema operativo (OS/400), junto a muchas funcionalidades centrales, como una base de datos integrada.

Con el paso de los años, el hardware y el software han atravesado muchas actualizaciones, revisiones, y cambios de nombre. Mientras que muchos todavía se refieren al sistema como AS/400 o, a veces, como servidor iSeries de IBM, hoy el hardware es -técnicamentePower Systems, y ejecuta un sistema operativo llamado IBM i.

Desde el comienzo, una de las principales funcionalidades de esta plataforma ha sido su creciente compatibilidad. Es posible ejecutar un programa creado para el AS/400 de 1988 en un servidor Power Systems actual con muy pocos o incluso ningún cambio.

Esta compatibilidad continua es una de las razones por las que muchas compañías que compraron un AS/400 años atrás, todavía lo llaman AS/400. Aunque los servidores Power son más rápidos y cuentan con una tecnología de vanguardia.

IBM continúa actualizado la plataforma a diario y tiene grandes planes para el futuro de IBM i. Cada dos o tres años, lanza nuevas versiones de hardware y software que ofrecen avances en la potencia de procesamiento y sus funcionalidades.

En la figura 3 se muestra una fotografía del servidor AS400 de la empresa Senator Hotels & Resorts.

Figura 3: Servidor AS400 de Senator Hotels & Resorts.

3.1.1. Características de IBM i

El sistema IBM i posee muchas cualidades y características pero se mostrarán a continuación las más destacables:

- **IBM i es escalable.** Las empresas pueden empezar con un servidor económico 4-core y fácilmente ascender a máquinas de hasta 256-core. Es extraño que las necesidades de procesamiento de una empresa superen a IBM i.
- **IBM i es seguro.** Cuando está correctamente configurado y equipado con el software adecuado, IBM i puede ser una plataforma muy segura.
- **IBM i es confiable.** Se suele decir que esta plataforma trabaja todo el tiempo. De hecho, ofrece confiabilidad de nivel empresarial y herramientas de alta disponibilidad para operar 100 % de manera continua.
- **IBM i es moderno.** Además de ser capaz de ejecutar los programas existentes, IBM i soporta un saludable mix de lenguajes de desarrollo nativos y de código abierto, incluyendo RPG, SQL, Java, .NET, PHP y C++.
- **IBM i es compatible.** Igual de verdadero que en 1988, la creciente compatibilidad de esta plataforma protege su inversión al evitar costosas migraciones de código cuando las plataformas son actualizadas.

3.1.2. Instalación de IBM i

Para acceder a la plataforma y, con ello, a nuestro servidor AS400, deberemos instalar la aplicación "iSeries Access". Esta aplicación viene instalada por defecto en todos los equipos de la empresa a través de la red local. Para acceder a ella deberemos hacer click con el botón derecho del ratón sobre el escritorio y seleccionar Nuevo>Icono de escritorio de IBM i, tal y como podemos observar en la figura 4.

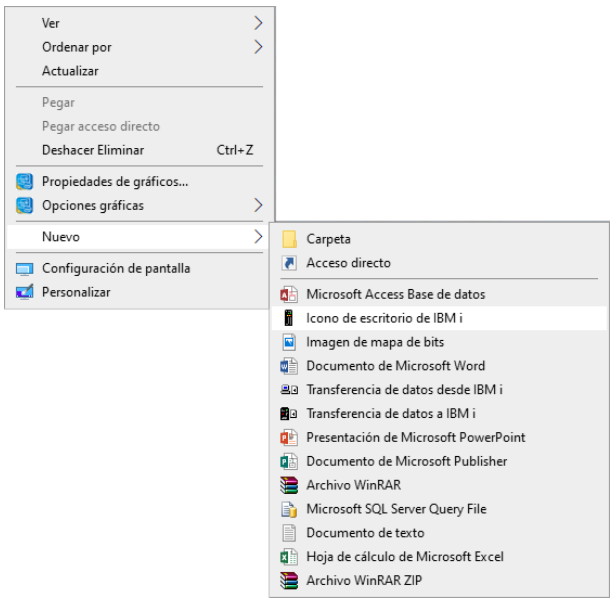


Figura 4: Crear icono de escritorio de IBM i.

Una vez hacemos click sobre el icono, la aplicación, nos pedirá que introduzcamos nuestros credenciales, los cuales tienen que estar creados previamente, tal y como podemos observar en la figura 5.

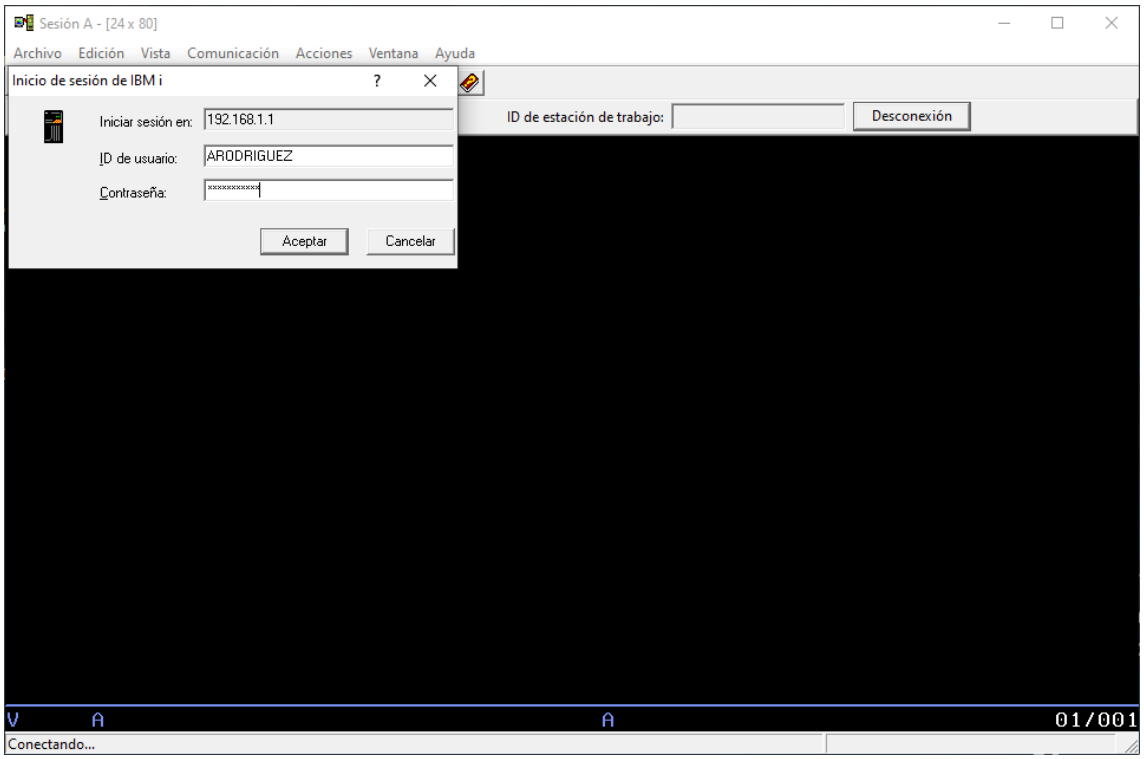


Figura 5: Iniciar sesión para acceder al AS400.

En la figura 6 podemos observar cómo es la interfaz de usuario de IBM i.

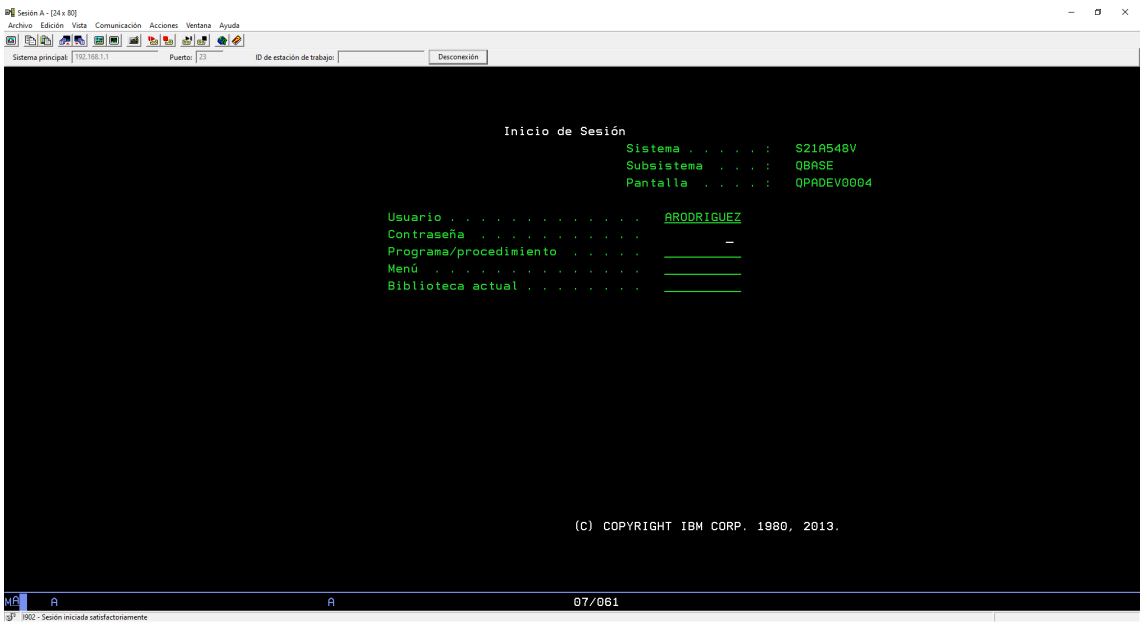


Figura 6: Interfaz de usuario de IBM i.

Tras volver a introducir nuestros credenciales, podremos acceder a varias opciones, entre ellas la opción de SQL, que es la que más nos interesa, ya que para ciertos usuarios orientados al soporte de informática, por ejemplo, tiene está opción capada pero para el área de desarrollo esta opción sí está disponible. En la figura 7 podemos observar un ejemplo de cómo nos muestra el AS400 las columnas de una tabla denominada RESPLAHT.

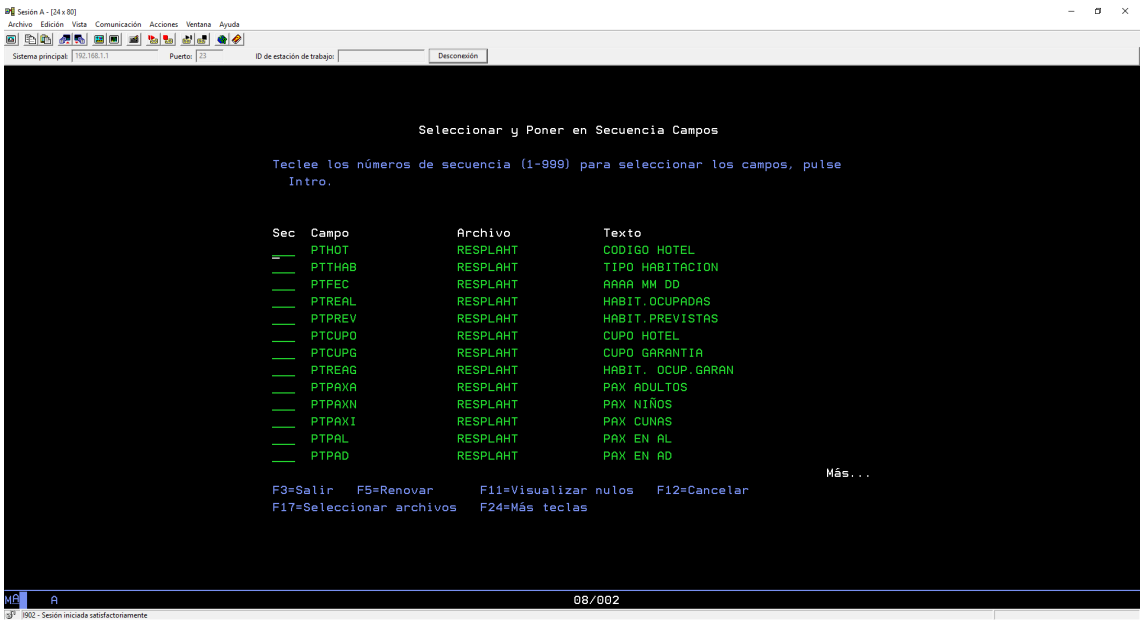


Figura 7: Vista de la tabla RESPLAHT en el AS400.

3.2. Sistema DB2

DB2 [17] es una familia de productos de sistema de gestión de bases de datos relacionales (RDBMS) de IBM que sirven a varias plataformas diferentes de sistemas operativos. Según IBM, DB2 lidera en términos de participación y rendimiento en el mercado de bases de datos. Aunque los productos DB2 se ofrecen para sistemas basados en UNIX y sistemas operativos de computadoras personales, DB2 sigue a productos de base de datos de Oracle en sistemas basados en UNIX y a Access de Microsoft en sistemas Windows.

Además de su oferta para el OS/390 para mainframe y los sistemas operativos VM y sus sistemas AS/400 de su gama media, IBM ofrece productos DB2 para un espectro multiplataforma que incluye Linux basado en UNIX, HP-UX, Sun Solaris y SCO UnixWare; y para su sistema operativo de computadoras personales OS/2, así como para los sistemas Windows 2000 de Microsoft y versiones anteriores. Las bases de datos DB2 se puede acceder desde cualquier programa de aplicación mediante el uso de la interfaz Open Database Connectivity (ODBC) de Microsoft, la interfaz Java Database Connectivity (JDBC) o un corredor de interfaz CORBA.

3.2.1. Versión de DB2 y compatibilidad

DB2 versión 9 [18] es un motor de base de datos relacional que integra XML de manera nativa, lo que IBM ha llamado pureXML, que permite almacenar documentos completos dentro del tipo de datos xml para realizar operaciones y búsquedas de manera jerárquica dentro de éste, e integrarlo con búsquedas relacionales.

La compatibilidad implementada en la última versión, hace posible la importación de los datos a DB2 en una media de 1 o 2 semanas, ejecutando PL/SQL de forma nativa en el gestor IBM DB2.

La automatización es una de sus características más importantes, ya que permite eliminar tareas rutinarias y permitiendo que el almacenamiento de datos sea más ligero, utilizando menos hardware y reduciendo las necesidades de consumo de alimentación y servidores.

DB2 para Linux, UNIX y Windows permite la automatización de tareas, reducción de las necesidades de consumo de alimentación, un alto rendimiento que reduce los servidores necesarios para ejecutar la base de datos, escalabilidad sencilla y alta disponibilidad en su arquitectura de discos de datos y otras soluciones que facilitan la colaboración entre profesionales.

Con aplicaciones que se despliegan y desarrollan de forma sencilla incluso si han sido creadas para utilizarse con otros software de bases de datos.

3.2.2. Características de DB2

En esta sección se detallarán las características de DB2 [19], el cual es capaz de recopilar y analizar información específica, incluyendo información sobre los siguientes elementos:

- Aplicaciones con el mayor porcentaje de sentencias SQL fallidas, desbordamientos de clasificación, tiempos de espera de bloqueo excedidos y puntos muertos, y la proporción más baja de aciertos de agrupación de almacenamientos intermedios.
- Bases de datos con el porcentaje más alto de sentencias SQL fallidas, la proporción más baja de aciertos de agrupación de almacenamientos intermedios y el número más alto de conexiones, tiempos de espera excedidos de bloqueo y puntos muertos.
- Información de configuración y tiempo de ejecución de "HADR" (Database High Availability and Disaster Recovery).
- Uso de tamaño y de página de los espacios de tablas.
- Información de registro, como la cantidad de espacio de registro activo, el porcentaje de registro secundario y la cantidad de espacio utilizando por los registros de archivado.

- Mensajes de diagnósticos del archivo "db2diag.log" de la base de datos de DB2, incluyendo información sobre el registro de notificación de administración, registro de sucesos y registro de diagnósticos.
- Definiciones personalizadas de la sentencia de SQL, estados de última ejecución y resultados.

4. Software utilizado

En esta sección se indicará todo el software utilizado para el desarrollo del proyecto. El software utilizado pasa desde el mismo IDE hasta los lenguajes de programación que me han dotado de herramientas suficientes para poder llevar a cabo las operaciones de sincronización de las plataformas así como las actualizaciones y obtención de la información de tanto las plataformas web como del servidor local y en la nube.

Todo ello basándonos en los distintos protocolos de comunicación que poseen cada plataforma ya que, como hemos indicado anteriormente, cada plataforma posee su propio protocolo de comunicación y hay que adaptar las peticiones a dichos protocolos haciendo uso de sus APIs.

4.1. Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado (IDE) [20], es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Un IDE debe tener las siguientes características:

- Multiplataforma.
- Soporte para diversos lenguajes de programación.
- Integración con Sistemas de Control de Versiones.
- Reconocimiento de Sintaxis.
- Extensiones y Componentes para el IDE.
- Integración con Framework populares.
- Depurador.
- Importar y Exportar proyectos.
- Múltiples idiomas.
- Manual de Usuarios y Ayuda.

4.1.1. Microsoft Visual Studio

El entorno de trabajo que he utilizado se denomina Microsoft Visual Studio en su versión Professional 2013 aunque actualmente está disponible hasta la versión de 2017.

Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes utilizan las funciones de .NET Framework, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y Servicios Web XML.

La nueva versión de Visual Studio contiene las siguientes características:

- **Mayor productividad:** correcciones y mejoras de código, navegación y depurado. Ahorra tiempo y esfuerzo en las tareas diarias sin importar el lenguaje o la plataforma. En equipos DevOps, Visual Studio 2017 agiliza en inner loop y acelera el flujo de código con nuevas características en tiempo real.
- **Azure:** integrado en la suite de las herramientas de Azure, permite a los desarrolladores crear fácilmente aplicaciones "cloud first" bajo Microsoft Azure, facilitando la configuración, compilación, depurado y el package.
- **Desarrollo móvil:** Visual Studio 2017 con Xamarin hace más rápido y fácil para los desarrolladores compilar, conectar y ajustar aplicaciones móviles para Android, iOS y Windows.

4.1.2. Microsoft .NET Framework

.NET Framework es una tecnología que admite la compilación y ejecución de la última generación de aplicaciones y Servicios web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.
- Ofrecer un entorno de ejecución de código que promueva la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan scripts o intérpretes de comandos.

- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en Web.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

4.1.3. Microsoft SQL Server 2012 Management Studio

SQL Server es un sistema de gestión de bases de datos relacionales (RDBMS) de Microsoft que está diseñado para el entorno empresarial. SQL Server se ejecuta en T-SQL (Transact -SQL), un conjunto de extensiones de programación de Sybase y Microsoft que añaden varias características a SQL estándar, incluyendo control de transacciones, excepción y manejo de errores, procesamiento fila, así como variables declaradas.

Bajo el nombre código Yukon en su etapa de desarrollo, SQL Server 2005 fue lanzado en noviembre de 2005. Se dice que el producto 2005 proporcionó una mayor flexibilidad, escalabilidad, confiabilidad y seguridad a las aplicaciones de base de datos, y permitió que fueran más fáciles de crear y desplegar, lo que reduce la complejidad y el tedio involucrado en la gestión de bases de datos. SQL Server 2005 también incluía más soporte administrativo.

El código original de SQL Server ha sido desarrollado por Sybase; a finales de 1980, Microsoft, Sybase y Ashton-Tate colaboraron para producir la primera versión del producto, SQL Server 4.2 para OS/2. Posteriormente, tanto Sybase como Microsoft ofrecieron productos de SQL Server. Sybase cambió después el nombre de su producto a Adaptive Server Enterprise.

4.1.4. SoapUI

SoapUI [21] es una aplicación de pruebas para servicios web, es un programa de código abierto para arquitecturas orientadas a servicios SOA y REST. Su funcionalidad abarca la inspección de servicios web, llamada, simulación y consumo. Se usa para pruebas funcionales, pruebas de carga y chequeo.

SoapUI fue lanzado inicialmente en SourceForge en septiembre de 2005 . Es un software libre, licenciado bajo los términos de la Licencia Pública General GNU. Está desarrollado por Smarbear sobre Java, y es multiplataforma. Hoy en día, también es compatible con SoapUI IDEA, Eclipse y NetBeans. En este artículo veremos un ejemplo básico de consumo de servicio web através de este programa.

Una vez iniciada la aplicación debemos crear un nuevo proyecto SOAP, para ello pulsaremos sobre “Projects” y “New SOAP Project”, tal y como podemos observar en la figura 8.

Figura 8: Creación de un nuevo proyecto SOAP.

Debemos indicar la descripción del servicio, definida bajo el protocolo WSDL, y que define la interfaz pública a los servicios Web. Esta descripción se puede indicar al programa SoapUI, mediante una URL, o bien mediante un fichero almacenado en local. En nuestro caso utilizaremos una URL y será la que nos facilita la API de la plataforma web OTS. Esta URL es la siguiente:

http://integrations.axisdata.net:54321/b2b-OTAProviderConnector?WSDL

Una vez introducida la URL en el campo "Initial WSDL", nos pondrá por defecto el nombre asociado a dicho servicio web, en nuestro caso, integrations.axisdata. Podemos ver en la figura 9 cómo quedaría dicha descripción.

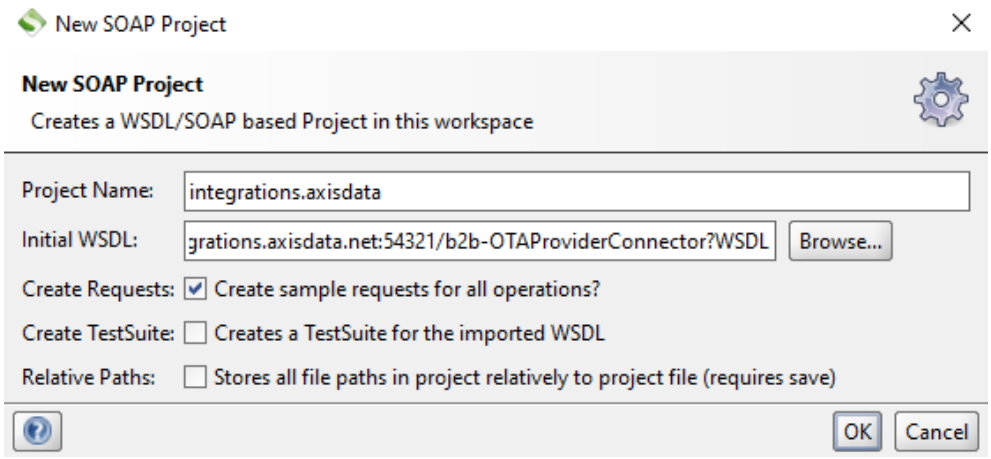


Figura 9: Información del proyecto SOAP.

La URL anterior, si la introducimos en nuestro navegador y accedemos a ella, nos devolverá un XML como el que se puede observar en la figura 10.

Figura 10: XML devuelto por la URL del proyecto SOAP en un navegador.

Una vez configurado y conectado nuestro proyecto SOAP, nos aparecerá en el menú de la izquierda los nombres de sus posibles peticiones. En nuestro caso, haremos click sobre la petición llamada “hotelRatePlan” y nos mostrará una plantilla de petición SOAP. Esta plantilla no es más que un XML con todos sus atributos marcados con una “?” los cuales deberíamos sustituir por nuestros valores reales. No es necesario rellenar todos los datos, la petición será válida si incluimos solo los esenciales.

Para realizar la petición satisfactoriamente deberemos rellenar los datos correctamente y pulsar sobre el botón de ejecutar situado en la esquina superior izquierda de nuestra petición. Una vez hemos ejecutado nuestra petición SOAP, la ventana se divide en dos zonas, una con los datos de consulta, situada a la izquierda, y otra con los datos de la respuesta, situada a la derecha. Debajo de estas dos ventanas podemos ver el tiempo de respuesta de la consulta por ejemplo. Podemos ver el resultado de nuestra petición en la figura 11.

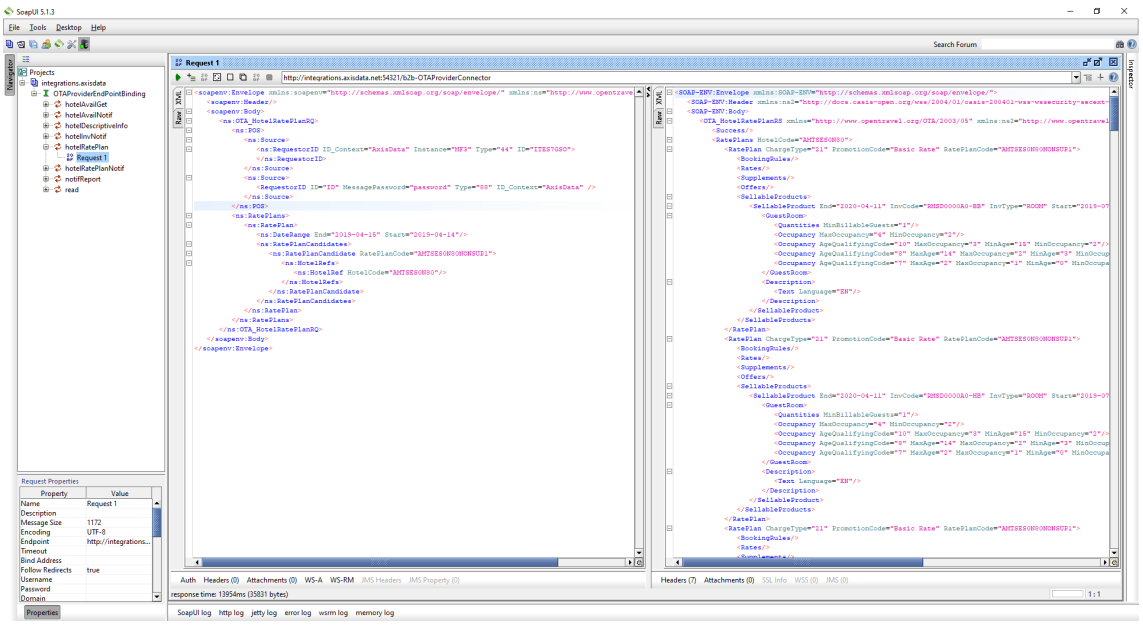


Figura 11: Petición SOAP completada.

Como hemos podido observar, no es complicado ejecutar peticiones XML a los diferentes servicios webs que hay en el mercado. Además, estas peticiones nos devuelven información adicional como los tiempos de respuesta que son muy útiles ya que un tiempo exagerado puede provocar un fallo en la aplicación o "timeout". Estas peticiones realmente serán llevadas a cabo mediante programación, pero este programa resulta verdaderamente útil si queremos realizar diversas pruebas en las que gráficamente podemos ver con simpleza las respuestas de las peticiones, para saber, por ejemplo, la estructura del XML devuelto.

4.2. Lenguajes de programación utilizados

En informática, se conoce como lenguaje de programación [22] a un programa destinado a la construcción de otros programas informáticos. Su nombre se debe a que comprende un lenguaje formal que está diseñado para organizar algoritmos y procesos lógicos que serán luego llevados a cabo por un ordenador o sistema informático, permitiendo controlar así su comportamiento físico, lógico y su comunicación con el usuario humano.

La implementación de lenguajes de programación permite el trabajo conjunto y coordinado, a través de un conjunto afín y finito de instrucciones posibles, de diversos programadores o arquitectos de software, para lo cual estos lenguajes imitan, al menos formalmente, la lógica de los lenguajes humanos o naturales.

Para el desarrollo de las apps se han utilizado diversos lenguajes de programación. Esto se debe a que cada lenguaje ofrece unas características diferentes a los demás puesto que, en el paradigma de la programación orientada a objetos, por ejemplo, no es conveniente ni lógico expresar las sentencias en un lenguaje específico para la ges-

tión de las bases de datos como lo es SQL.

Por ello, hemos utilizado tres lenguajes de programación, los cuales son Visual Basic .NET, XML y SQL. A modo de resumen, estos lenguajes de programación tienen los siguientes objetivos:

- **Visual Basic .NET:** se ha utilizado para el desarrollo de todas las funciones de las Apps. Es el motor del proyecto ya que es la base sobre la que se sostiene toda la estructura del código. Se encarga de ejecutar todas las operaciones referentes a la App y su interacción gráfica.
- **XML:** se ha utilizado para el envío de peticiones web con el objetivo de enviar y recibir datos desde las plataformas web. Sin este lenguaje de programación no se podría interactuar con las diferentes plataformas de reserva online.
- **SQL:** se ha utilizado para la gestión de todas las bases de datos debido a que es el lenguaje por referencia en este campo. Tanto para el trato con las bases de datos en el servidor en la nube mediante SQL Server como con la base de datos local basada en el servidor de IBM AS/400, aunque en este último varía un poco las intrucciones, la base de las peticiones es SQL.

Tal y como hemos podido ver, cada lenguaje de programación tienen su función específica y, por lo tanto, no son compatibles entre sí pero sí permiten la colaboración entre ellos.

4.2.1. Visual Basic .NET

En el mundo de la programación informática, uno de los lenguajes más populares y conocidos es el de Visual Basic [23]. Creado en 1991 por Alan Cooper para Microsoft, este paquete permite programar contenidos informáticos gráficos de manera simple y accesible.

Visual Basic ha sido desarrollado con el objetivo de entregar a los usuarios de programación informática un paquete de utilidades simples y accesibles. Es por esto que el Visual Basic puede ser usado y fácilmente comprendido por expertos como también por usuarios principiantes. Su base parte del dialecto BASIC pero con componentes novedosos que lo adaptan a los lenguajes informáticos modernos. A esto se suma que el Visual Basic es además un lenguaje de programación guiado por eventos que permite mayor operatibilidad y mejores resultados.

4.2.2. SQL

El lenguaje de consulta estructurado o SQL [24] (por sus siglas en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como

hacer cambios en ella.

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones.

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

4.2.3. XML

XML [25] son las siglas del Lenguaje de Etiquetado Extensible. La expresión se forma a partir del acrónimo de la expresión inglesa eXtensible Markup Language. Se trata también de un lenguaje estándar que posee una Recomendación del World Wide Web Consortium: Extensible Markup Languajes (XML). Con la palabra "Extensible" se alude a la no limitación en el número de etiquetas, ya que permite crear aquellas que sean necesarias.

XML es un lenguaje que permite jerarquizar y estructurar la información y describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo. La información estructurada presenta varios contenidos (texto, imágenes, audio, etc.) y formas: hojas de cálculo, tablas de datos, libretas de direcciones, parámetros de configuración, dibujos técnicos, etc. La forma da alguna indicación de qué papel puede jugar el contenido (por ejemplo, el contenido de una sección encabezada con un significado difiere del contenido de una nota a pie de página, lo que significa algo diferente que el contenido de un pie de foto o el contenido de una tabla de datos). Más o menos todos los documentos tienen la misma estructura.

4.3. Application Programming Interface

Application Programming Interface o comúnmente abreviado "API" [26] es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.

Las API son valiosas, ante todo, porque permiten hacer uso de funciones ya existentes en otro software (o de la infraestructura ya existente en otras plataformas) para no estar reinventando la rueda constantemente, reutilizando así código que se sabe que está probado y que funciona correctamente. En el caso de herramientas propietarias (es decir, que no sean de código abierto), son un modo de hacer saber a los programadores de otras aplicaciones cómo incorporar una funcionalidad concreta sin por ello tener que proporcionar información acerca de cómo se realiza internamente el proceso.

En nuestro caso, hemos utilizado diferentes APIs para poder obtener y actualizar información de las diferentes plataformas web a las que tenemos que acceder.

Las APIs están separadas, es decir, tendremos una API específica para obtener información y otra para actualizarla. Para ambas APIs tendremos que enviar un XML concreto con la información de usuario y password de la plataforma y el hotel y/o la habitación (no todas las APIs te piden la habitación, algunas te devuelven todas las habitaciones). En el caso concreto de la API de actualización, además hay que añadir los valores a actualizar, ya sea el precio y la disponibilidad.

4.3.1. API de Mirai

Para Mirai hemos utilizado la API "getInventory" para obtener los datos de disponibilidad, tarifas y precios, y la API "webservice_updater" para actualizar tanto la disponibilidad como el precio asociado a una tarifa concreta.

Para realizar una petición a la API "getInventory" solamente hay que hacer una llamada a la siguiente dirección URL con los datos que queramos pedir:

https://api.mirai.com/XMLIntegrationx/webservice_getinventory.apro?login=**username**&password=**password**&hotelId=**hotelId**&month=**month**&year=**year**

Como podemos observar se le pasarán cinco datos clave para la obtención de los datos de Mirai. Estos son los siguientes:

- **Username:** este será nuestro username que utilizamos para realizar un inicio de sesión en la plataforma de Mirai.
- **Password:** este será nuestra contraseña que utilizamos para realizar un inicio de sesión en la plataforma de Mirai.
- **HotelId:** este será el código del hotel a examinar que nos proporciona la plataforma.
- **Month:** este será el mes que queremos examinar. Mirai solo acepta llamadas por mes y no por rango de fechas (menor eficiciencia en comparación con las demás plataformas ya que si se pide un rango entre meses tendríamos que realizar dos llamadas).
- **Year:** este será el año que queremos examinar. Mirai solo acepta llamadas por mes y año y no por rango de fechas (menor eficiciencia en comparación con las demás plataformas ya que si se pide un rango entre años tendríamos que realizar dos llamadas).

Esta petición nos devolverá un enorme XML que nuestra aplicación recorrerá y comprobará. Si encuentra disparidades, tanto para precio como para disponibilidad las corregirá creando un XML similar al que se puede observar en el código 1 del Anexo.

Este XML tiene seis valores clave para la actualización de los datos de Booking. Estos son los siguientes:

- **HotelId:** este será el código del hotel a examinar que nos proporciona la plataforma.
- **RoomID:** este será el código de la habitación a examinar que nos proporciona la plataforma.
- **Currency:** este valor indicará la moneda corresponidiente, en nuestro caso, los euros (EUR).
- **Date:** este valor siempre será 1.0 en nuestro caso.
- **UnitPrice:** este valor será un entero que corresponderá al nuevo precio.
- **MinimumStay:** este valor será un entero y hará referencia al mínimo de noches de estancia en la habitación.

Una vez construido el XML con los valores correspondientes se enviarán a la siguiente dirección URL para que la plataforme los procese y actualice correctamente:

https://api.mirai.com/XMLIntegrationx/webservice_updater.apro?
login=**user**&password=**password**

Para poder comprobar los cambios en la plataforma deberemos de ir a la misma, la cual se encuentra en la siguiente dirección URL:

https://es.mirai.com

Una vez hagamos inicio de sesión como unidad hotelera accederemos a la extranet de Mirai. En dicha extranet deberemos seleccionar uno de los hoteles asociados a la cuenta logueada, en nuestro caso hemos seleccionado el hotel "Playaballena Aquapark & Spa". Una vez seleccionado se nos mostrará la interfaz que muestra la figura 12.

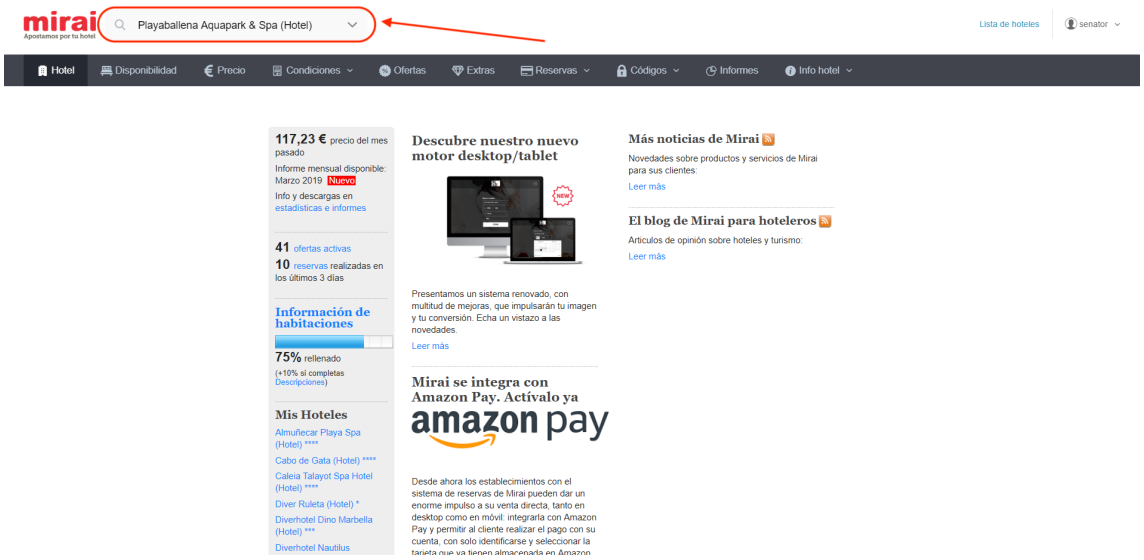


Figura 12: Extranet de Mirai.

Si queremos observar los precios asociados a dicho hotel, deberemos pulsar sobre la opción “Precio” y nos aparecerá la ventana que podemos observar en la figura 13.

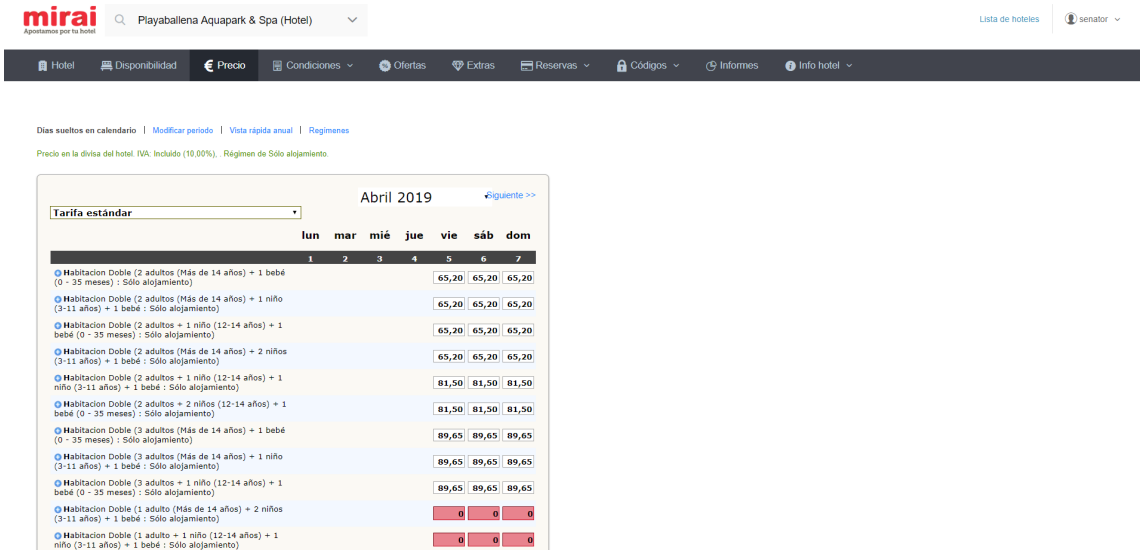


Figura 13: Precio en la extranet de Mirai.

Si queremos observar las disponibilidades asociadas a dicho hotel, deberemos pulsar sobre la opción “Disponibilidad” y nos aparecerá la ventana que se puede observar en la figura 14.

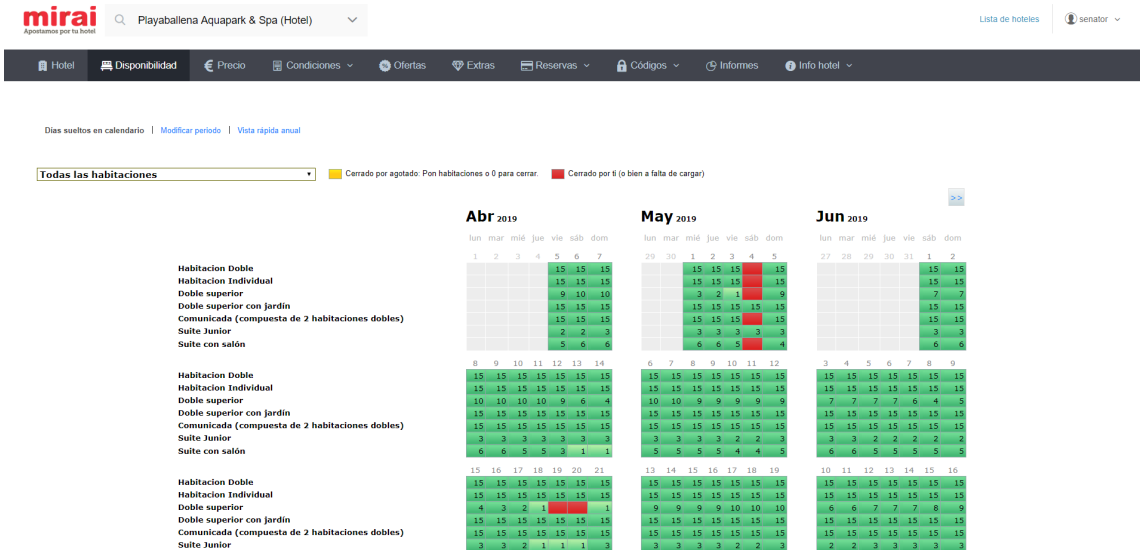


Figura 14: Disponibilidad en la extranet de Mirai.

4.3.2. API de Booking

Para Booking hemos utilizado la API "roomrateavailability" para obtener los datos de disponibilidad, tarifas y precios, y la API "availability" para actualizar tanto la disponibilidad como el precio asociado a una tarifa concreta.

Para realizar una petición a la API "roomrateavailability" solamente hay que hacer una llamada a la siguiente dirección URL con los datos que queramos pedir:

https://supply-xml.booking.com/hotels/xml/roomrateavailability

Esta petición nos devolverá un enorme XML que nuestra aplicación recorrerá y comprobará. Si encuentra disparidades, tanto para precio como para disponibilidad las corregirá creando un XML similar al que se puede observar en el código 2 del Anexo.

Como podemos observar se le pasarán siete datos clave para la obtención de los datos de Booking. Estos son los siguientes:

- **Username:** este será nuestro username que utilizamos para realizar un inicio de sesión en la plataforma de Booking.
- **Password:** este será nuestra contraseña que utilizamos para realizar un inicio de sesión en la plataforma de Booking.
- **Hotel_Id:** este será el código del hotel a examinar que nos proporciona la plataforma.
- **Version:** este valor siempre será 1.0 en nuestro caso.

- **Number_of_days:** este valor es el que determinará la longitud de la fecha. A la fecha inicial o "start_date" se le suma este valor y tendremos el rango de fechas a examinar.
- **Start_Date:** esta será la fecha del primer día a examinar, a partir de este día comenzará la búsqueda de información.
- **Room_level:** este valor es importante y varía mucho en la información obtenida ya que en función de este dato se mostrarán más detalles de la habitación o no. En nuestro caso el valor siempre será 1 ya que no requerimos de tantos detalles, los imprescindibles que son el precio y la disponibilidad se encuentran en el primer nivel.

Esta petición nos devolverá un enorme XML que nuestra aplicación recorrerá y comprobará. Si encuentra disparidades, tanto para precio como para disponibilidad las corregirá creando un XML similar al código 3.

Este XML tiene seis valores clave para la actualización de los datos de Booking. Estos son los siguientes:

- **Username:** este será nuestro username que utilizamos para realizar un inicio de sesión en la plataforma de Booking.
- **Password:** este será nuestra contraseña que utilizamos para realizar un inicio de sesión en la plataforma de Booking.
- **Hotel_Id:** este será el código del hotel a examinar que nos proporciona la plataforma.
- **Version:** este valor siempre será 1.0 en nuestro caso.
- **Room_Id:** este será el código de la habitación a examinar que nos proporciona la plataforma.
- **From:** esta será la fecha de inicio para la petición.
- **To:** esta será la fecha de fin para la petición.
- **Rate_Id:** este será el código de la tarifa a examinar que nos proporciona la plataforma.
- **Price:** este será el nuevo precio a actualizar (entero).
- **Closed:** este valor indica si está cerrada mediante un 0 o no la habitación mediante un 1 (boolean).
- **Quantity:** este valor será la nueva cantidad de habitaciones disponibles.
- **MinimumStay:** este valor será un entero y hará referencia al mínimo de noches de estancia en la habitación (0 indica que no hay límite).

- **MaximumStay:** este valor será un entero y hará referencia al máximo de noches de estancia en la habitación (0 indica que no hay límite).

Una vez construido el XML con los valores correspondientes se enviarán a la siguiente dirección URL para que la plataforme los procese y actualice correctamente:

https://supply-xml.booking.com/hotels/xml/availability

Si queremos comprobar que se han actualizado correctamente los cambios, deberemos dirigirnos a la plataforma de Booking accediendo a la siguiente dirección URL:

https://admin.booking.com/

Una vez hagamos inicio de sesión como unidad hotelera accederemos a la extranet de Booking. En dicha extranet deberemos seleccionar el hotel, habitación y tarifa correspondiente para poder observar tanto el precio asociado como la disponibilidad.

4.3.3. API de OTS

Para OTS hemos utilizado la API "OTA_HotelAvailGetRQ" para obtener los datos de disponibilidad, tarifas y precios, y la API "OTA_HotelRatePlanNotifRQ" para actualizar tanto la disponibilidad como el precio asociado a una tarifa concreta.

En este caso, hemos utilizado un Servicio de Referencia en el cual se generan los XMLs en base a objetos predefinidos por la propia compañía. Para utilizar dicho servicio, debemos primero agregarlo a nuestro proyecto. La agregación se realizará dirigiendonos hacia el explorador de soluciones y haciendo click derecho sobre el proyecto y pulsaremos sobre la opción "Agregar referencia de servicio..." tal y como podemos observar en la figura 15 (en nuestro caso, hemos creado una carpeta en el proyecto denominada "Service References y aquí se ha incluido").

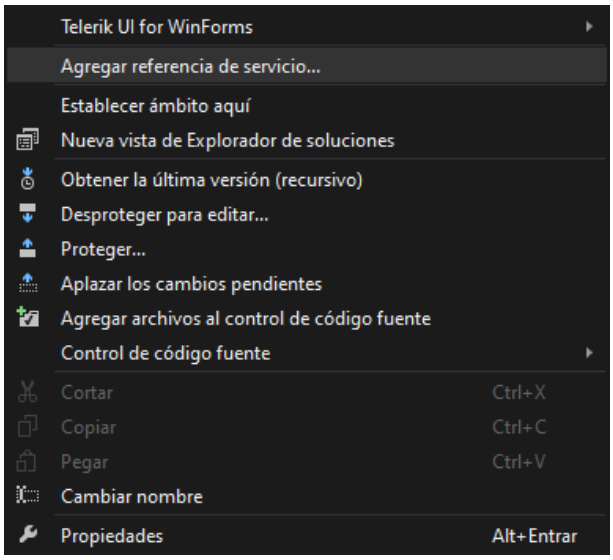


Figura 15: Agregar Referencia de Servicio.

Una vez hayamos hecho click sobre dicha opción se nos abrirá una ventana en la que tendremos que indicar la dirección URL en la que se encuentra dicho servicio, la cual, para nuestro caso es la siguiente:

http://integrations.axisdata.net:54321/b2b-OTAProviderConnector?WSDL

Si nuestra dirección es correcta, le daremos al botón "Ir" y nos mostrará toda la información asociada a dicho servicio abajo, junto con todas las operaciones de las que dispone. Le escribimos un nombre abajo tal como "ServiceReferenceOTS". En nuestro caso, la ventana nos quedaría tal y como se muestra en la figura 16.

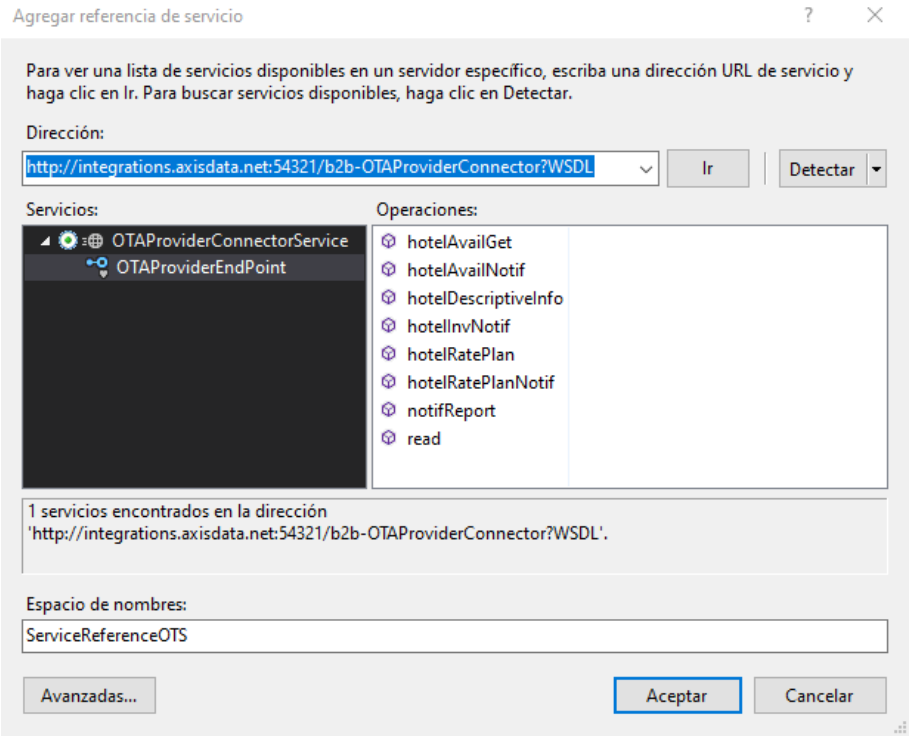


Figura 16: Información recopilada de la Referencia de Servicio.

Una vez que pulsemos el botón "Aceptar", este servicio de referencia se agregará a nuestro proyecto y podremos utilizarlo. Podemos comprobar cómo queda una vez agregado en la figura 17.

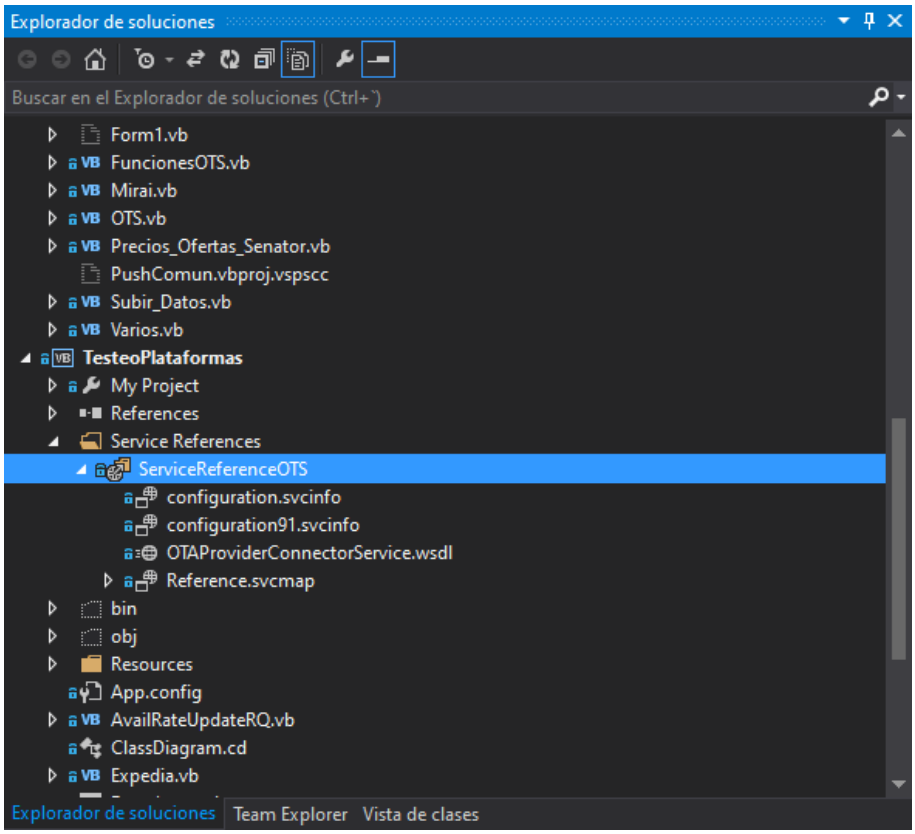


Figura 17: Referencia de Servicio agregada.

En este caso, tenemos que serializar los XML de petición y de subida en función de los objetos que podemos utilizar a través de la referencia.

Si queremos comprobar que se han actualizado correctamente los cambios, deberemos dirigirnos a la plataforma de OTS accediendo a la siguiente dirección URL:

https://provider.otsglobe.com/

Una vez hagamos inicio de sesión como unidad hotelera accederemos a la extranet de OTS. En dicha extranet deberemos seleccionar previamente en la opción “Contract List” o lista de contratos y nos aparecerán todos los contratos asociados a la cuenta registrada.

Una vez seleccionado el contrato, en nuestro caso, hemos seleccionado el contrato asociado al hotel “Playaballena”, veremos toda una lista de información asociada a las tarifas, fechas y disponibilidades de las habitaciones, así como gráficas de rendimiento del hotel. Si pulsamos sobre la “i” de información sobre una de las celdas de la primera tabla, se nos mostrará una pequeña ventana en la que podemos observar el precio base y la disponibilidad. Podemos observar esta ventana a continuación en la figura 18.

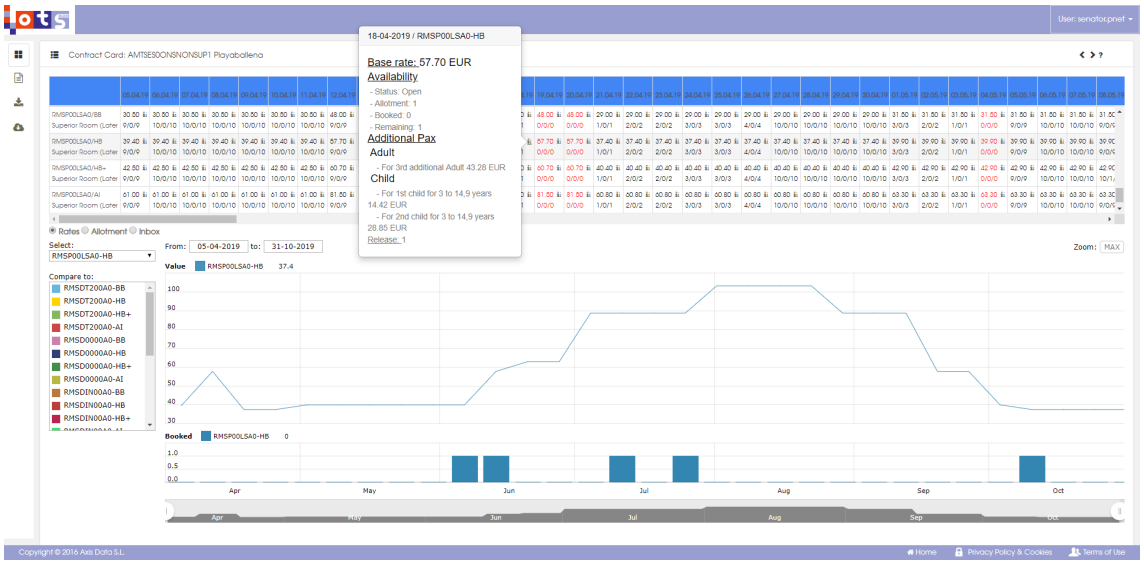


Figura 18: Información de precios y disponibilidad en OTS.

4.3.4. API de Expedia

Para Expedia hemos utilizado la API "ProductAvailRateRetrievalRS" para obtener los datos de disponibilidad, tarifas y precios, y la API "AvailRateUpdateRQ" para actualizar tanto la disponibilidad como el precio asociado a una tarifa concreta.

En este caso, hemos utilizado una clase denominada "AvailRateUpdateRQ" que hará como Servicio de Referencia al igual que para el caso de la plataforma OTS ya que sus métodos generan los XMLs en base a objetos predefinidos por la propia compañía.

En este caso, tendremos que deserializar primero el objeto para transformarlo en un XML para poder enviarlo directamente a la plataforma para que obtenga los datos o los actualice. Estas peticiones se envían a la siguiente dirección URL:

https://services.expediapartnercentral.com/eqc/parr

Si queremos comprobar que se han actualizado correctamente los cambios, deberemos dirigirnos a la plataforma de Expedia accediendo a la siguiente dirección URL:

https://join.expediapartnercentral.com/es/

Una vez hagamos inicio de sesión como unidad hotelera accederemos a la extranet de Expedia. En dicha extranet deberemos seleccionar el hotel, habitación y tarifa correspondiente para poder observar tanto el precio asociado como la disponibilidad.

4.4. Interfaces de usuario

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar.

Las interfaces básicas de usuario son aquellas que incluyen elementos como menús, ventanas, teclado, ratón, los beeps y algunos otros sonidos que la computadora hace, y en general, todos aquellos canales por los cuales se permite la comunicación entre el ser humano y la computadora. La mejor interacción humano-máquina a través de una adecuada interfaz (Interfaz de Usuario), que le brinde tanto comodidad, como eficiencia.

4.4.1. Tipos de interfaces de usuario

Dentro de las Interfaces de Usuario se puede distinguir básicamente tres tipos:

- **Una interfaz de hardware:** a nivel de los dispositivos utilizados para ingresar, procesar y entregar los datos: teclado, ratón y pantalla visualizadora.
- **Una interfaz de software:** destinada a entregar información acerca de los procesos y herramientas de control, a través de lo que el usuario observe a habitualmente en la pantalla.
- **Una interfaz de Software-Hardware:** que establece un puente entre la máquina y las personas, permite a la máquina entender la instrucción y a el hombre entender el código binario traducido a información legible.

4.4.2. Características de las interfaces de usuario

En el diseño de una interfaz de usuario en sitios web, ordenadores y aplicaciones el usuario es de vital importancia y en todo momento se tiene que tener cuenta la experiencia e interacción del usuario.

El principal objetivo del diseño de interfaz de usuario es lograr que la interacción del usuario sea lo más eficiente y simple posible. Al fin y al cabo, los diseños que se centran en el usuario se convierten en las mejores interfaces.

Diseñar una buena interfaz de usuario requiere excelentes habilidades de diseño, amplio conocimiento del mundo del diseño y una comprensión de las necesidades del usuario. Hay muchos procesos previos que tienen que ser monitorizados antes de pasar al diseño propiamente dicho.

Estas son algunas de las características clave para desarrollar una buena interfaz de usuario:

- **Claridad:** la claridad evita que el usuario cometa errores al lidiar con dicha interfaz y da una perfecta experiencia al usuario. Claridad significa que la información es transmitida de manera precisa.
- **Interactividad:** Una buena interfaz tiene que ser rápida. Esta característica aumenta grandemente la experiencia del usuario.
- **Flexibilidad:** La mejor interfaz de usuario es aquella que puede deshacer las acciones erróneas del usuario sin mayor problema.
- **Eficiente:** La interfaz de usuario debería entender el objetivo del usuario y no impedírselo de ningún modo.
- **Atractiva:** Un diseño atractivo puede provocar que los usuarios no solo utilicen la aplicación sino que también desarrollen una especie de conexión con ésta.

4.4.3. Componentes Incorporados a la interfaz de usuario

En esta sección detallaremos los componentes adicionales que hemos requerido e incorporado a la interfaz de usuario de nuestro proyecto. Estos componentes son necesarios para el correcto funcionamiento del código. En primer lugar debemos conocer cómo se añaden los componentes. Esta opción se encuentra en la vista de diseñador de nuestro formulario en la barra izquierda con el nombre de "Cuadro de herramientas". Este cuadro de herramientas nos muestra una lista de los componentes que podemos incluir en nuestra aplicación. Para hacer la inclusión del componente seleccionado, únicamente deberemos de arrastrarlo hasta la interfaz y ya quedará incorporado a la misma. Esta lista de componentes tiene la interfaz que se observa en la figura 19.

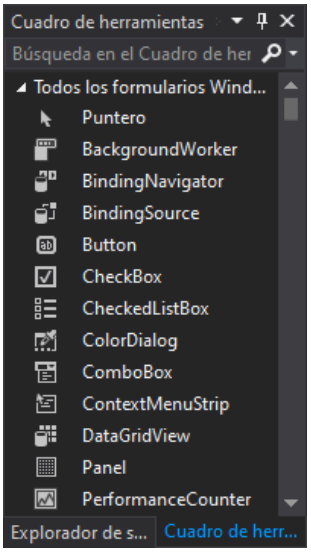


Figura 19: Cuadro de Herramientas.

A continuación detallamos los componentes que hemos requerido incluir para nuestro proyecto:

- **BackgroundWorker:** este componente se trata de un hilo en segundo plano y nos servirá para poder ejecutar acciones sin que afecte a las demás interacciones permitidas por nuestra aplicación. Además es muy útil para evitar "crash" en la App. En nuestro caso la hemos utilizado para las comprobaciones.

  Para poder hacer un uso completo de este componente debemos irnos a sus propiedades y activar las opciones de "ProgressChanged" y "RunWorkerCompleted" ya que vienen desactivadas por defecto. Estas opciones se pueden observar en la figura 20.

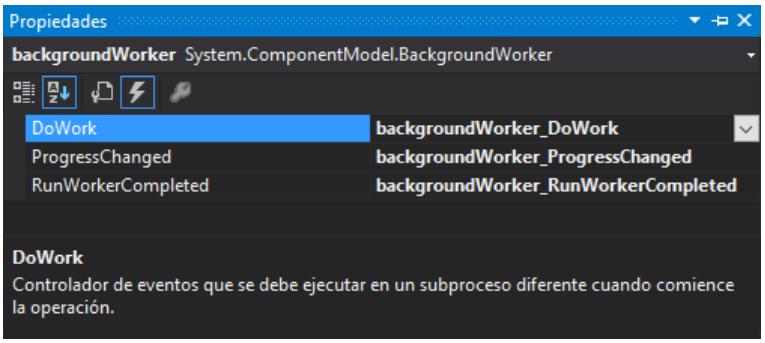


Figura 20: Propiedades del BackgroundWorker.

- **Timer:** este componente se trata de un temporizador que ejecutará una acción en bucle cada cierto tiempo determinado. Esto nos servirá por ejemplo, para crear un bucle de comprobaciones.

- **ImageList:** este componente se trata de una colección de imágenes y será necesario para poder incorporar iconos personalizados a nuestro proyecto. Una vez incorporado a nuestro proyecto, haremos click sobre él para ver sus propiedades y seleccionaremos la propiedad "images" localizada donde se muestra en la figura 21.

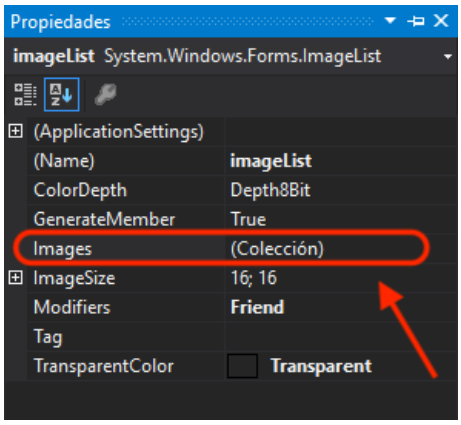


Figura 21: Propiedades de una colección de imágenes.

Una vez hayamos hecho click sobre dicha colección nos aparecerá la siguiente ventana en la que podremos agregar las imágenes las cuales tienen como identificador para poder acceder a ellas el número que le acompaña a la izquierda siendo incremental desde la primera imagen hasta la última. Podemos observar cómo queda nuestra colección de imágenes ya rellena en la figura 22.

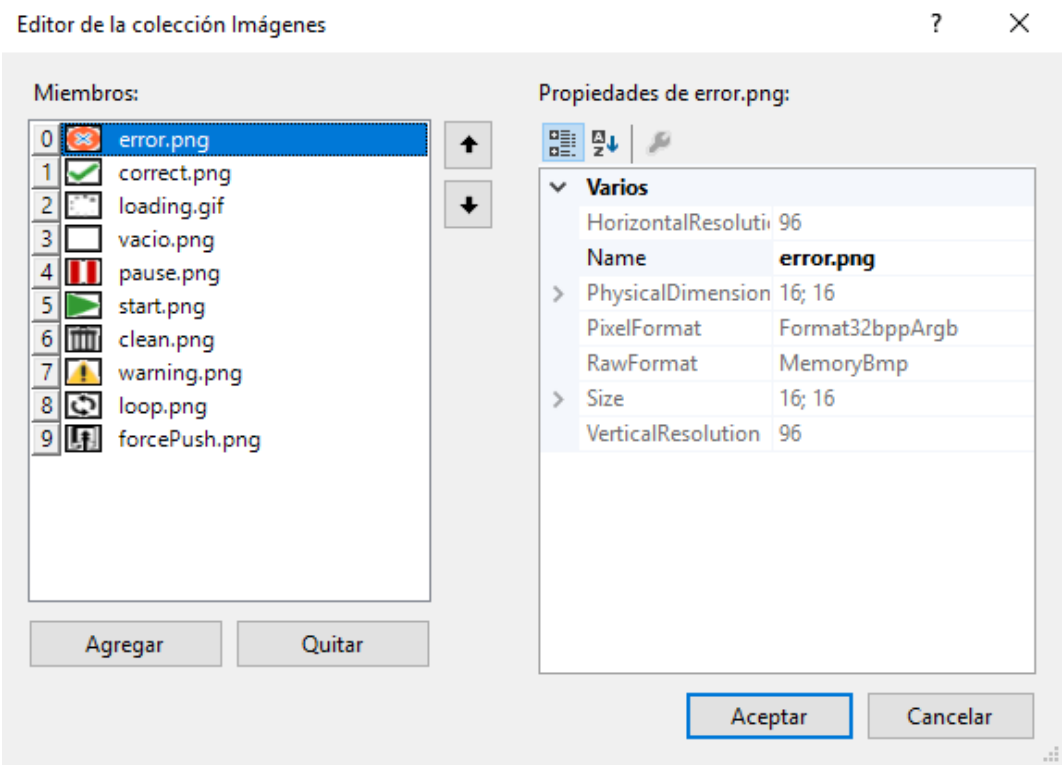


Figura 22: Colección de imágenes del proyecto.

- **VisualStudio2012LightTheme:** este tema será necesario para mejorar la interfaz del seleccionable de tablas y plataformas del proyecto, entre otras cosas.

5. Metodología de desarrollo

Una metodología de desarrollo de software brinda al equipo de trabajo un marco para construir aplicaciones de manera eficiente y rigurosa, garantizando un producto cercano al esperado. Si no se desarrolla a partir de una metodología, el resultado final será impredecible y no se podrá controlar el avance del proyecto.

5.1. Programación Modular

Un mismo programa puede ser creado de muchas maneras distintas, dependiendo del estilo de cada uno de los usuarios programadores. Sin lugar a duda, un buen programa no es solo aquel que tiene un numero reducido de líneas de código, sino también una buena organización y suficientes comentarios descriptivos que expliquen que es lo que esta haciendo en cada parte del programa. Desde hace ya algunos años se han ido creando y perfeccionando diferentes técnicas de programación, con el objetivo de crear programas con un alto grado de organización, haciéndolos fáciles de leer y modificar.

La programación modular [27] es una técnica que consiste en dividir un problema en distintos módulos con el fin de que cada uno realice una única actividad o tarea. De esta manera cada uno de los módulos se analizan, codifican y ponen a punto por separado de los demás módulos. Cada programa contiene un programa principal que controla todo lo que sucede; este transfiere el control a módulos de manera que ellos puedan ejecutar sus propios submodulos. Los submodulos son independientes en el sentido en que ninguno de ellos puede tener acceso directo a cualquier otro módulo ó submódulo excepto el módulo que llama y sus propios submódulos. programa).

5.2. Ciclo de vida del software

El término ciclo de vida del software [28] describe el desarrollo de software, desde la fase inicial hasta la fase final. El propósito de este programa es definir las distintas fases intermedias que se requieren para validar el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo: se asegura de que los métodos utilizados son apropiados.

En los modelos clásicos de ciclo de vida del software (Modelo en cascada, Modelo incremental, Modelo de prototipo, Modelo en espiral...) las etapas de desarrollo consisten básicamente en cuatro actividades principales: Análisis, Diseño, Desarrollo y Evaluación. Estas actividades se llevan a cabo con un orden definido que suele vincular el proceso de desarrollo. Sin embargo, si se opta por seguir la metodología SCRUM, el ciclo de vida no se trata como un flujo lineal en el que se puedan distinguir exactamente estas actividades; además no se precisa seguir un orden preciso en el proceso de desarrollo. El proyecto puede empezar con cualquier actividad y se puede pasar de una actividad a otra en cualquier momento maximizando la flexibilidad y la productividad del equipo.

6. Estructura de la Extensión

Nuestro proyecto se desglosa, como hemos comentado con anterioridad, en una aplicación completa que realizará todas las tareas basándose en una tabla de fechas de eventos y una extensión de una aplicación completa donde introduciremos los datos de dicha tabla.

6.1. Estructura Externa

En nuestro caso, la extensión de la aplicación no recibió cambios en su interfaz, debido en parte a que se trata de una interfaz menos compleja. Debido a que se trata de una extensión y no de una aplicación como tal, solo explicaremos dicha extensión pero si detallaremos como deberemos navegar a través de la App para poder llegar hasta nuestros componentes gráficos.

En primer lugar, abriríamos la aplicación Peticiones XML y tendríamos que introducir los credenciales de Administrador, cuya interfaz se puede observar en la figura 23.

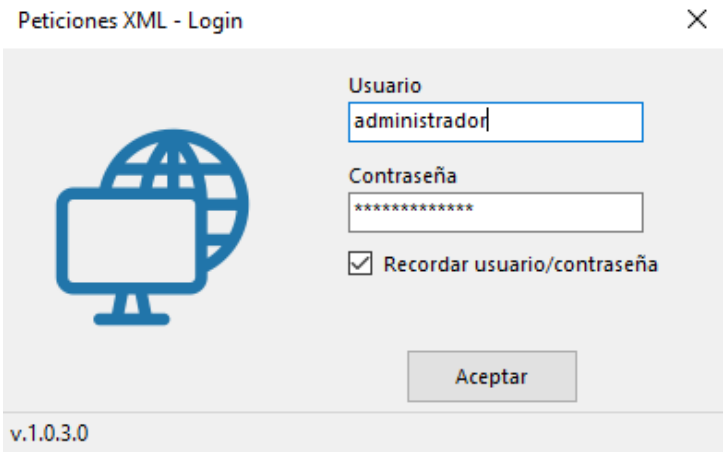


Figura 23: Interfaz de usuario del login de la aplicación Peticiones XML.

Una vez ingresamos los credenciales le daremos a iniciar sesión y nos aparecerá la interfaz que se muestra en la figura 24.

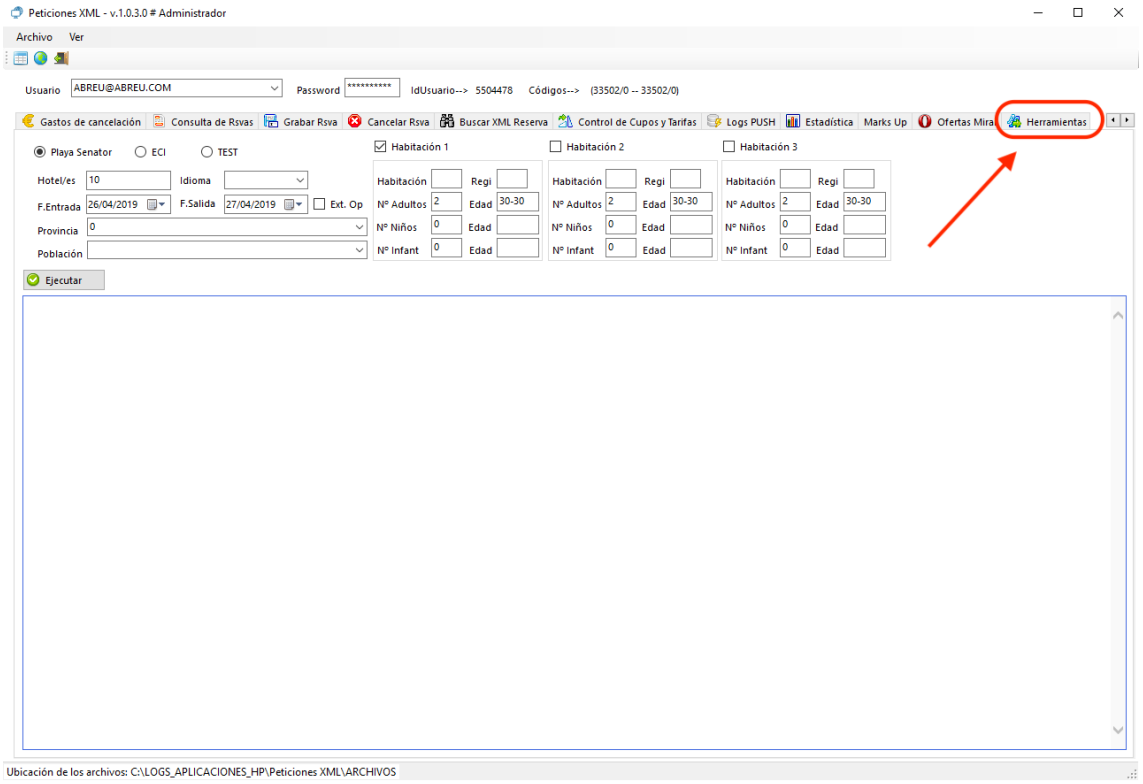


Figura 24: Interfaz de usuario de la aplicación Peticiones XML.

Podemos observar que la interfaz de la aplicación en general consta de una navegación basada en pestañas y, evidentemente, cada pestaña contiene una nueva interfaz. Para llegar a nuestra interfaz deberemos hacer click sobre la pestaña “Herramientas”.

La pestaña herramientas, a su vez, tiene dos pestañas más, nuestra interfaz se localizará, evidentemente, en una de ellas, concretamente en la pestaña “Mantenimiento PUSH”. En dicha pestaña se puede observar cuatro bloques los cuales se llaman “Caducidad Ofertas”, “Cierres de ventas”, “Bloqueo de cupo” y, la que verdaderamente nos interesa, “Fecha Eventos”. Esta interfaz se puede observar en la figura 25.

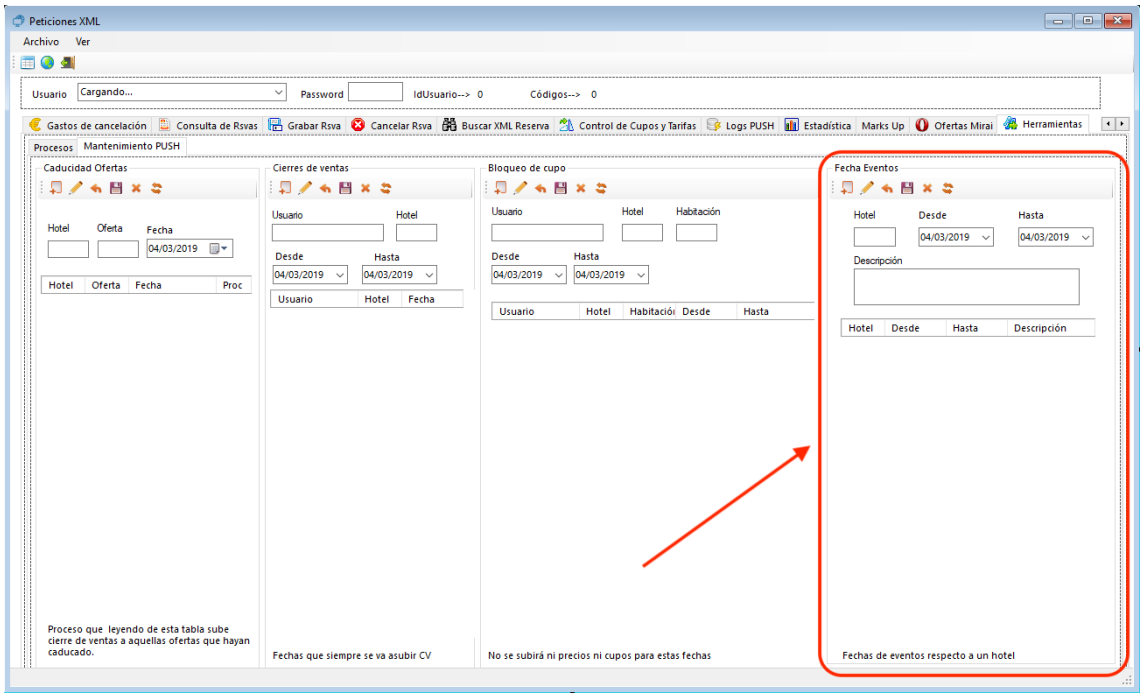


Figura 25: Interfaz de usuario de la extensión de la aplicación.

Podemos observar que todos los bloques contienen arriba una serie de botoneras. Para mantener una homogeniedad con la interfaz, hemos desarrollado la misma botonera con las mismas funciones. Se puede observar la botonera en la figura 26.

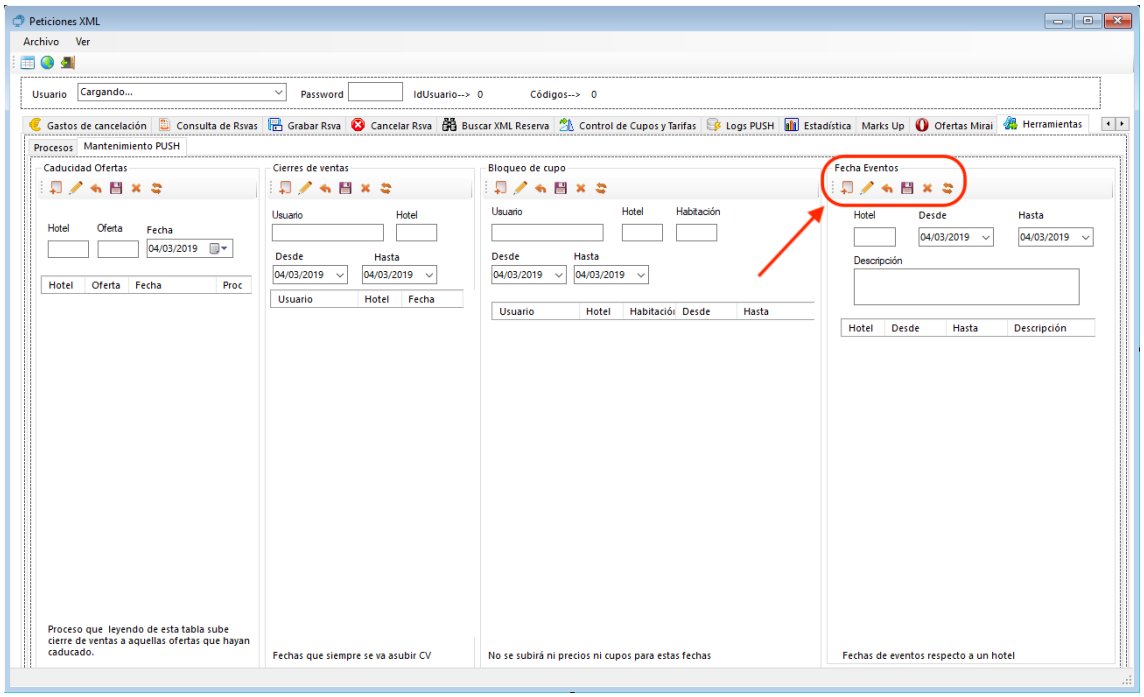


Figura 26: Botonera de la extensión de la aplicación.

Como es natural, cada botón tiene una función diferente dentro de la botonera, no obstante todas las botoneras mantienen una línea de funciones, es decir, por ejemplo, el primer boón que nos encontramos dentro de la botonera tiene la función de crear un nuevo objeto o registro, pues todos esos botones con la mismo icono harán exactamente lo mismo pero referente a su bloque. El icono de crear nuevo registro es el que se muestra en la figura 27.

Figura 27: Crear un nuevo registro.

La extensión permite editar registros seleccionados pulsando sobre el botón editar que se puede observar en la figura 28.

Figura 28: Editar un registro.

La extensión permite cancelar la edición de registros seleccionados pulsando sobre el botón cancelar que se puede pbservar en la figura 29.

Figura 29: Cancelar edición de un registro.

Si queremos guardar los cambios realizados en el registro seleccionado en la base de datos deberemos pulsar sobre el botón guardar que se puede observar en la figura 30.

Figura 30: Guardar el registro en la base de datos.

La extensión permite eliminar registros seleccionados pulsando sobre el botón eliminar que se puede observar en la figura 31.

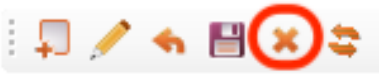


Figura 31: Eliminar un registro.

La extensión permite actualizar la tabla de fecha de eventos pulsando sobre el botón editar que se puede observar en la figura 32.

Figura 32: Actualizar tabla de fecha eventos.

En la figura 33 observaremos un ejemplo de mensaje de error para cuando se detecta una introdución inválida de un rango de fechas.

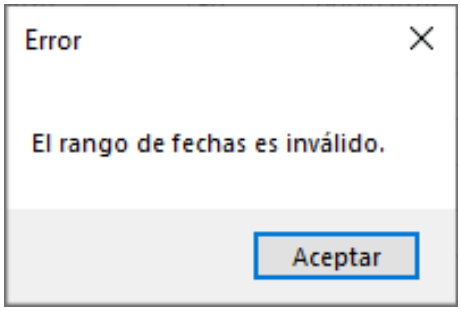
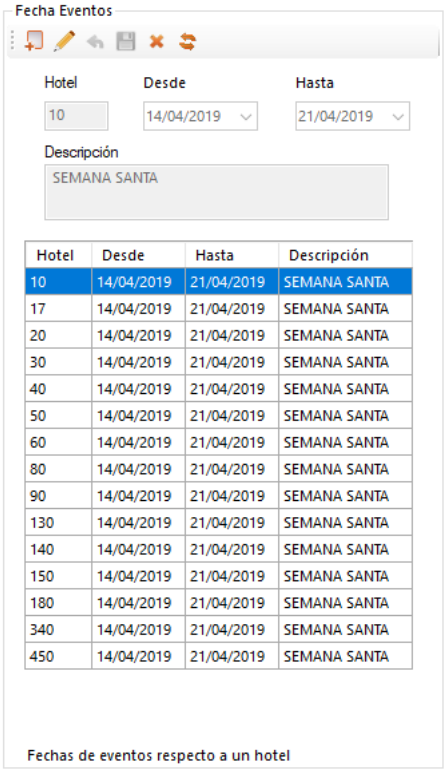


Figura 33: Ejemplo de un mensaje de error indicando un rango inválido de fechas.

Podemos observar en la figura 34 un ejemplo real de cómo estaría cargado el grid de fecha eventos.



Hotel	Desde	Hasta	Descripción
10	14/04/2019	21/04/2019	SEMANA SANTA
17	14/04/2019	21/04/2019	SEMANA SANTA
20	14/04/2019	21/04/2019	SEMANA SANTA
30	14/04/2019	21/04/2019	SEMANA SANTA
40	14/04/2019	21/04/2019	SEMANA SANTA
50	14/04/2019	21/04/2019	SEMANA SANTA
60	14/04/2019	21/04/2019	SEMANA SANTA
80	14/04/2019	21/04/2019	SEMANA SANTA
90	14/04/2019	21/04/2019	SEMANA SANTA
130	14/04/2019	21/04/2019	SEMANA SANTA
140	14/04/2019	21/04/2019	SEMANA SANTA
150	14/04/2019	21/04/2019	SEMANA SANTA
180	14/04/2019	21/04/2019	SEMANA SANTA
340	14/04/2019	21/04/2019	SEMANA SANTA
450	14/04/2019	21/04/2019	SEMANA SANTA

Figura 34: Ejemplo real de la tabla fecha de eventos cargada.

6.2. Estructura Interna

El proyecto, como ya hemos indicado anteriormente, ha sido elaborado con el lenguaje de programación "Visual Basic", el cual nos aporta muchas cualidades y estrategias. Una de ellas es la de mantener las partes del código encerradas en regiones para poder tener bien organizado las partes más sensibles de nuestro código o simplemente para dividir y clasificar funciones distintas. En el caso de la extensión de la aplicación, el código está dividido en regiones que separan las diferentes funciones del sistema permitiendo así identificar con mayor facilidad las diferentes pestañas que contiene el programa, por ejemplo.

Una región se crea utilizando la etiqueta "Region" seguido del nombre de la región. En nuestro caso, para dicha extensión, hemos denominado a nuestra región "Fecha Eventos" quedando tal y como se observa en el código 4 del Anexo.

Explicaremos el código con detalle a continuación. En primer lugar, nuestra estructura del código está basada en 7 métodos los cuales son los encargados de mantener todas las funcionalidades de la extensión. Tenemos una variable global para determinar si el usuario está editando o no. A continuación, explicaremos con detalle los diferentes métodos que contiene nuestra extensión:

- **Método CargarFechaEventos():** este método será el encargado de cargar inicialmente todas las configuraciones de la extensión así como el grid. Para esta extensión simula el comportamiento de un método main pero en este caso el método es llamado en cuanto la pestaña es cargada. En el código 5 del Anexo podemos observar su estructura completa.
- **Método HabilitarEdicionFechaEventos(ByVal habilitar As Boolean):** este método es el encargado de habilitar la botonera para que ésta permita la las opciones de edición o no. Por defecto, los botones de "nuevo evento", "editar evento" y "eliminar evento" están activados, mientras que el botón de "cancelar evento" y "guardar evento" están desactivados. En el código 6 del Anexo podemos observar su estructura completa.
- **Método RecuperarTextBoxFechaEventos():** este método obtiene los datos de la fila seleccionada en el grid y los incorpora a nuestros componentes gráficos. En el arranque de la pestaña si hace una llamada a este método para que esté el primer registro ya introducido en los componentes. En el código 7 del Anexo podemos observar su estructura completa.
- **Método HabilitarBotonesFechaEventos(ByVal habilitar As Boolean):** este método es el encargado de habilitar los componentes gráficos del hotel, fechas y descripción del evento. Por defecto estos componentes están desactivados. En el código 8 del Anexo podemos observar su estructura completa.
- **Método CargarGridFechaEventos():** este método es el encargado de rellenar la tabla con todos los eventos mediante SQL. En el código 9 del Anexo podemos observar su estructura completa.

- **Método AddFilaGridFechaEventos(ByVal hotel As Integer, ByVal desde As Date, ByVal hasta As Date, ByVal descripcion As String):** este método es el encargado de añadir una fila más a nuestra tabla de eventos mediante los valores pasados por parámetro. En el código 10 del Anexo podemos observar su estructura completa.

- **Método ActualizarFilaSeleccionadaGridFechaEventos(ByVal hotel As Integer, ByVal desde As Date, ByVal hasta As Date, ByVal descripcion As String):** este método será el encargado de actualizar los valores en la fila seleccionada, ya que los valores que hemos modificado, una vez enviados a la base de datos, hay que añadirlos a su fila correspondiente. En el código 11 del Anexo podemos observar su estructura completa.

- **Método BorrarTextBoxFechaEventos():** este método es el encargado de limpiar todos los componentes gráficos dejándolos vacíos y en el caso de las fechas poniendo la actual. En el código 12 del Anexo podemos observar su estructura completa.

- **Método EliminarFechaEventos():** este método será el encargado de eliminar de la base de datos todos los registros que tengamos seleccionados en el grid. En el código 13 del Anexo podemos observar su estructura completa.

- **Método RefrescarFechaEventos():** este método será el encargado de actualizar el grid de nuevo. Esto nos servirá para poder observar si los cambios se han realizado correctamente, en caso de un corte de red, por ejemplo. En el código 14 del Anexo podemos observar su estructura completa.

Todos estos métodos anteriores no tienen sentido si no están enlazados a los componentes gráficos o si no capturamos los eventos asociados como los click a los botones, el pegado de texto, la presión de las teclas... Estos eventos son capturados a través de **"Handles"** y a continuación detallaremos todos los que hemos utilizado en nuestro código.

- **Evento tsbNuevoFechaEventos.Click:** este evento capturará el click sobre el botón "tsbNuevoFechaEventos" y será el encargado de deshabilitar las opciones correspondientes de la botonera y habilitar los componentes gráficos. En el código 15 del Anexo podemos observar su estructura completa.

- **Evento tsbEditFechaEventos.Click:** este evento capturará el click sobre el botón "tsbEditFechaEventos" y será el encargado de controlar los posibles errores en la edición. En el código 16 del Anexo podemos observar su estructura completa.

- **Evento tsbCancelarFechaEventos.Click:** este evento capturará el click sobre el botón "tsbCancelarFechaEventos" y será el encargado de controlar la habilitación de los controles según corresponda. En el código 17 del Anexo podemos observar su estructura completa.

- **Evento tsbEliminarFechaEventos.Click:** este evento capturará el click sobre el botón "tsbEliminarFechaEventos" y será el encargado de controlar la verificación de las eliminaciones de los eventos seleccionados. En caso de que el usuario verifique la eliminación, se efectuará la acción. En el código 18 del Anexo podemos observar su estructura completa.
- **Evento tsbRefrescarFechaEventos.Click:** este evento capturará el click sobre el botón "tsbRefrescarFechaEventos" y será el encargado de realizar una actualización de la tabla fecha eventos. En el código 19 del Anexo podemos observar su estructura completa.
- **Evento tsbGuardarFechaEventos.Click:** este evento capturará el click sobre el botón "tsbGuardarFechaEventos" y será el encargado de controlar todos los posibles errores existentes realizando validaciones tales como detectar rangos de fechas inválidos, comprobar que el hotel exista... Después de realizar todas las validaciones, el programa guardará en la base de datos la fecha de evento con los datos actualizados, ya sea porque se ha creado uno nuevo o porque se ha editado uno. Además, este evento. En el código 20 del Anexo podemos observar su estructura completa.
- **Evento dgvFechaEventos.SelectionChanged:** este evento capturará el cambio de celda seleccionada sobre la tabla "dgvFechaEventos" y será el encargado de habilitar la edición que corresponda en los componentes gráficos y recuperar los nuevos datos de la fila seleccionada para incorporarlos en dichos componentes gráficos. En el código 21 del Anexo podemos observar su estructura completa.
- **Evento hotelFechaEventos.KeyPress:** este evento capturará la presión de cualquier tecla sobre el campo "hotelFechaEventos" y será el encargado de restringir que se introduzcan carácteres no numéricos. En el código 22 del Anexo podemos observar su estructura completa.

7. Estructura del Proyecto

En esta sección se explicará toda la estructura de forma detallada del proyecto, cómo ha sido elaborada y todas las funciones que tiene, así como las interfaces de usuario.

Para explicar el funcionamiento del proyecto así como todos sus detalles, dividiremos las dos estructuras: la externa y la interna.

7.1. Estructura Externa

En esta sección se detallará todo lo relacionado con la interfaz de usuario y los componentes incorporados al proyecto para el correcto funcionamiento del mismo.

7.1.1. Interfaz de usuario

La interfaz de usuario del proyecto ha sufrido muchas versiones durante todo el desarrollo del proyecto por diversos motivos entre los cuales se encuentran los siguientes:

- **Mejorar la interfaz de usuario:** se ha tenido en cuenta todas las características anteriormente detalladas para desarrollar una buena interfaz de usuario.
- **Implementación de nuevas funciones:** conforme se avanzaba en el proyecto también se le han ido añadiendo nuevas funciones que requerían de estar presentes en la interfaz de usuario de la aplicación.
- **Simplificar la programación:** simplificar la programación reduciendo o modificando elementos de la interfaz de usuario hace posible la disminución de la programación debido a que no todos los componentes de la interfaz tienen el mismo manejo y cada cual tiene sus restricciones.

Para poder utilizar los nuevos componentes hemos tenido que utilizar una librería externa denominada "Telerik". La versión más reciente de nuestra interfaz es la que se puede observar en la figura 35.

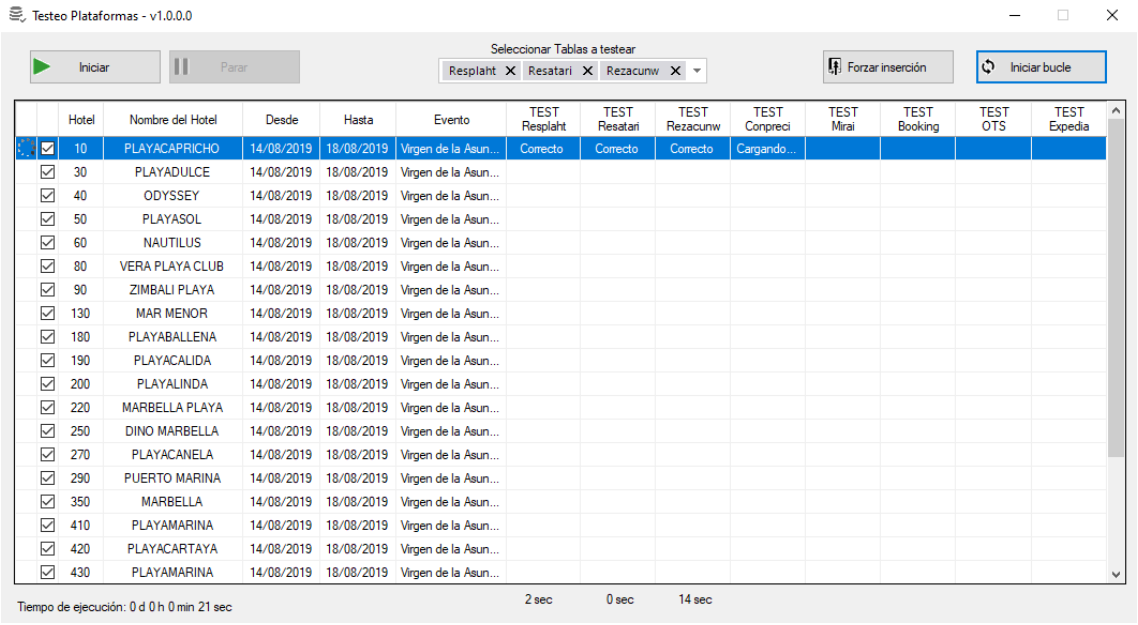


Figura 35: Interfaz del proyecto.

Para explicarla, la desglosaremos en hasta 16 puntos distintos, los cuales se muestran en la figura 36.

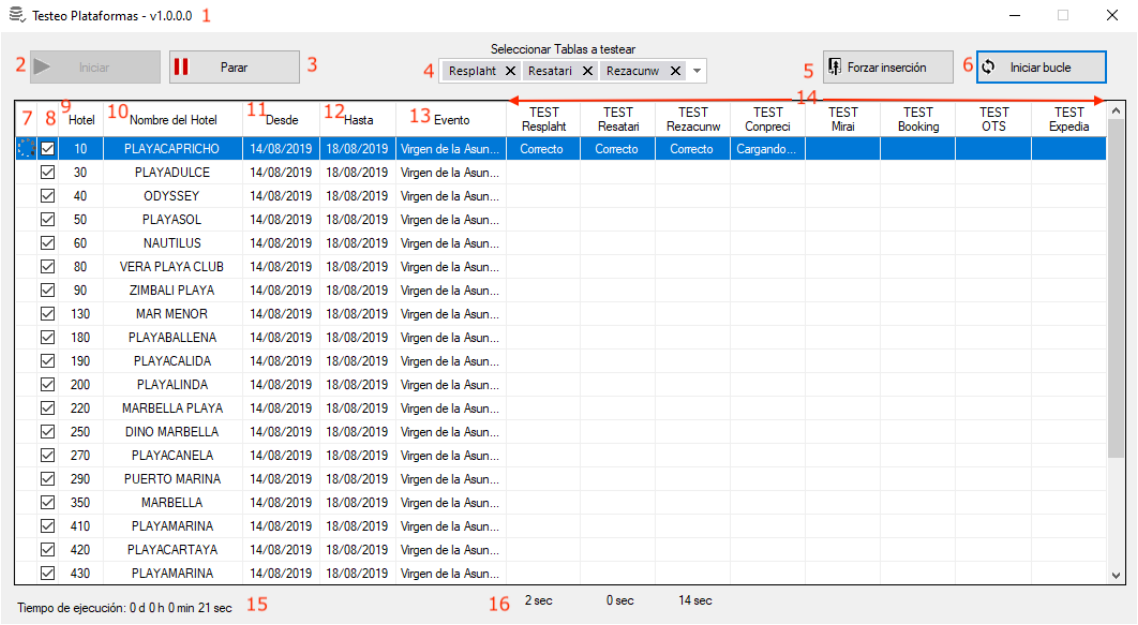


Figura 36: Interfaz del proyecto desglosada.

Estos puntos se explicarán a continuación:

1. **Versión del proyecto:** Este componente mostrará la versión del proyecto, en nuestro caso, nos encontramos con la primera versión, la versión v1.0.0.0.

2. **Botón Iniciar:** Este componente es el botón Iniciar, una vez lo pulsemos se ejecutarán las comprobaciones.

3. **Botón Parar:** Este componente es el botón Parar, una vez lo pulsemos se pararán las comprobaciones.

4. **Combobox de tablas y plataformas:** Este componente es un combobox que contiene las tablas y plataformas que vamos a comprobar (se puede ir cambiando en plena ejecución, ya que el hilo en segundo plano nos permite hacerlo).

5. **Botón Forzar Inserción:** Este componente es un botón que abrirá la ventana correspondiente para forzar la inserción de nuevos registros en función de las tablas que elijamos (este componente se explicará con más detalle en el punto "Forzar Inserción").

6. **Botón Iniciar/Parar Bucle:** Este componente es un botón que será el encargado de activar o desactivar el temporizador para poder ejecutar las comprobaciones en bucle.

7. **Columna de verificación:** Esta columna será la que indicará, a simple vista, si se está realizando esa comprobación, si han ocurrido problemas, errores, o si está todo correcto para todas las comprobaciones de ese rango de fechas.

8. **Columna de checkeo:** Esta columna estará por defecto activada y podremos desactivarla si queremos que esa comprobación en concreto no se ejecute (se puede ir cambiando en plena ejecución, al igual que el combobox de tablas y plataformas).

9. **Columna "Hotel":** Esta columna nos indicará el código del hotel que vamos a comprobar.

10. **Columna "Nombre del Hotel":** Esta columna solo nos mostrará a simple vista el nombre del hotel.

11. **Columna "Desde":** Esta columna nos mostrará el rango de fecha desde el que partimos.

12. **Columna "Hasta":** Esta columna nos mostrará el rango de fecha hasta que finalizamos.

13. **Columna "Evento":** Esta columna nos mostrará una descripción del evento por el cual se realiza la comprobación.

14. **Columnas de comprobación:** Estas columnas son las más importantes y nos mostrarán a simple vista si ha habido warnings, errores o si está todo correcto para cada TEST.

15. **Tiempo total de comprobación:** Este componente nos mostrará el tiempo total de las comprobaciones.

16. **Tiempo de comprobación por tabla/plataforma:** Este componente nos mostrará el tiempo medio de comprobación por tabla/plataforma. Nos servirá de bastante ayuda para poder determinar qué tablas o plataformas tardan más en darnos los datos por si nuestra red interna está fallando y el servidor AS400 no nos aporta la información con suficiente rapidez, por ejemplo, o por si hay problemas de conexión con ciertas plataformas.

7.1.2. Iconos utilizados

En esta sección, se indicarán todos los iconos utilizados dentro del proyecto, así como una breve descripción del objetivo de cada uno. Estas descripciones se pueden observar en la tabla 2.

Icono	Descripción
	Este icono será mostrado en la celda correspondiente cuando haya ocurrido un error.
	Este icono será mostrado en la celda correspondiente cuando haya ocurrido una incidencia o disparidad.
	Este icono será mostrado en la celda correspondiente cuando no haya ocurrido ni un error ni una incidiencia o disparidad.
	Este icono será mostrado en la celda correspondiente cuando esté realizándose el proceso de comprobación.
	Este icono es una imagen transparente y se utiliza para simular que no hay nada.
	Este icono será el que hará referencia al forzado del registro.
	Este icono será el que hará referencia a la activación y desactivación del bucle de comprobación.
	Este icono será el que hará referencia al “pause” o parado de las comprobaciones o del bucle.
	Este icono será el que hará referencia al “start” o comienzo de las comprobaciones.
	Este es el icono de la aplicación, se muestra tanto en la esquina superior izquierda de la ventana como en los accesos directos de la aplicación.

Tabla 2: Iconos de la aplicación.

Para nuestro proyecto hemos utilizado una imagen animada o también denominado GIF, el cual se mostrará para indicar que se está ejecutando el test para el evento y hotel seleccionados, pero este elemento no puede ser introducido en la colección de imágenes, pues Visual Studio lo detectaría como una imagen estática y no es nuestra intención. Para poder incluirlo en nuestro proyecto requerimos de introducirlo como un recurso y para ello debereos dirigirnos a las propiedades del proyecto, en la opción “Recursos” y pulsar sobre el botón “Agregar Recurso”. Podemos observar nuestro recurso loading cargado en el proyecto en la figura 37.

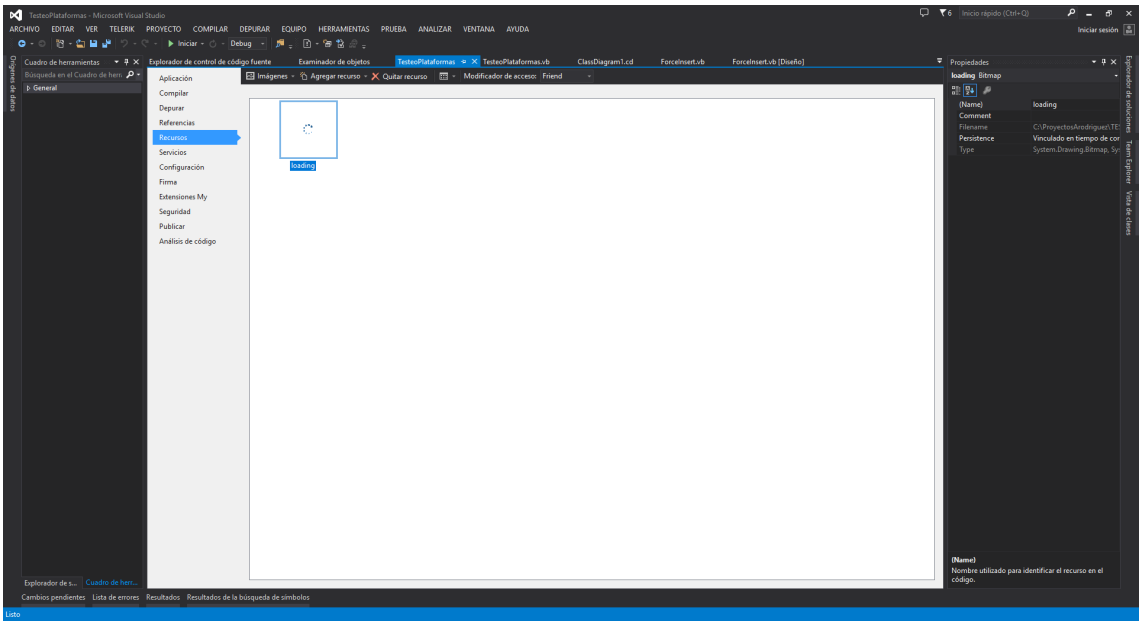


Figura 37: Colección de recursos del proyecto.

7.2. Estructura Interna del proyecto

La estructura interna del proyecto consta de toda la programación y organización de las clases creadas así como toda su funcionalidad. Nuestro proyecto consta de tres bloques/clases fundamentales, la estructura principal centrada en el main de la aplicación, la comprobación local y la comprobación web.

7.2.1. Estructura principal

En esta sección se detellará todas las funciones más importantes de la estructura principal de proyecto. Esta estructura es el motor de la aplicación ya que es la que se encarga de toda la interacción gráfica y de hacer las llamadas correspondientes a las comprobaciones.

En el código 23 del Anexo se pueden observar todas las variables globales que se utilizarán para el desarrollo de nuestro proyecto.

A continuación se explicarán todos los "Handles" de la aplicación:

- **Evento platformTesting_Load:** este evento se ejecutará al cargar el formulario principal de la aplicación. Será el encargado de llamar al método loadForm(). En el código 24 del Anexo podemos observar su estructura completa.
- **Evento platformTesting_FormClosing:** este evento se ejecutará al cerrar el formulario principal de la aplicación. Será el encargado de llamar al método stopTesting(). En el código 25 del Anexo podemos observar su estructura completa.

- **Evento btnStartTesting_Click:** este evento se ejecutará al hacer click sobre el botón "btnStartTesting". Será el encargado de llamar al método startTesting(). En el código 26 del Anexo podemos observar su estructura completa.
- **Evento btnStopTesting_Click:** este evento se ejecutará al hacer click sobre el botón "btnStopTesting". Será el encargado de llamar al método stopTesting(). En el código 27 del Anexo podemos observar su estructura completa.
- **Evento btnLoopTesting_Click:** este evento se ejecutará al hacer click sobre el botón "btnLoopTesting". Será el encargado de llamar al método loopTesting(). En el código 28 del Anexo podemos observar su estructura completa.
- **Evento btnForceInsert_Click:** este evento se ejecutará al hacer click sobre el botón "btnForceInsert". Será el encargado de llamar al método forceInsert(). En el código 29 del Anexo podemos observar su estructura completa.
- **Evento timer_Tick:** este evento se ejecutará en bucle en intervales de tiempos establecidos en nuestro timer. Será el encargado de llamar al método actionTimer(). En el código 30 del Anexo podemos observar su estructura completa.
- **Evento backgroundWorker_RunWorkerCompleted:** este evento se ejecutará una vez que nuestro hilo principal haya concluido. Detectará si ha habido errores y lo grabará en la variable "existsError", en caso contrario llamará al método backgroundWorkerCompleted(). En el código 31 del Anexo podemos observar su estructura completa.
- **Evento backgroundWorker_ProgressChanged:** este evento se ejecutará una vez que hayamos indicado a nuestro hilo principal que debe realizar una acción sobre la interfaz gráfica de nuestro proyecto. Sele pasará una variable compuesta de tres valores, los cuales son "row", "column" y "report" que equivalen a la celda correspondiente a través de las dos primeras y el reporte a indicar en la misma gracias a la llamada del método updateCell(). En el código 32 del Anexo podemos observar su estructura completa.
- **Evento backgroundWorker_DoWork:** este evento se ejecutará una vez que hayamos indicado a nuestro hilo principal que se ejecute. Será el encargado de llamar al método backgroundWorkerAction(). En el código 33 del Anexo podemos observar su estructura completa.

A continuación se detallarán los métodos del proyecto:

- **Método loadForm():** este método realizará la función de cargar la versión del programa, establecer qué tipo de aplicación se trata (hay funciones en la librería comun capadas por IP, se tuvo que introducir esta opción para indicar si se trata de esta aplicación haga una excepción), crear la ruta de los logs y cargar el grid de fecha de eventos. En el código 34 del Anexo podemos observar su estructura completa.

- **Método startTesting():** este método realizará la función de cargar de nuevo el grid, limpiar los tiempos, establecer el tiempo actual y arrancar la acción del hilo principal. En el código 35 del Anexo podemos observar su estructura completa.
- **Método stopTesting():** este método realizará la función de parar la acción del hilo principal, indicándole, esté donde esté, ya sea en un bucle o no que debe volver al handle evento "backgroundWorker_RunWorkerCompleted". En el código 36 del Anexo podemos observar su estructura completa.
- **Método forceInsert():** este método será el encargado de mostrar el formulario de forzado de inserción. En el código 37 del Anexo podemos observar su estructura completa.
- **Método loadGrid():** este método será el encargado de cargar el grid de fechas eventos. Primero obtendrá todos los registros chequeados en el datatable dtCheckedTest para poder seguir manteniendo esta configuración al recargar la tabla, si no es la primera vez. Una vez obtenidos los rgistros chequeados, se procederá a limpiar y cargar la tabla con los datos obtenidos de la tabla de fecha de eventos. Se desactivarán ciertas funciones de la interfaz como la ordenación por columnas ya que puede alterar el correcto funcionamiento de las comprobaciones. Solo se añadirán los eventos si la fecha actual es igual o superior a la de los mismos. En el código 38 del Anexo podemos observar su estructura completa.
- **Método clearTimes():** este método será el encargado de limpiar todos los tiempos, ya tanto las variables como los componentes gráficos (label). En el código 39 del Anexo podemos observar su estructura completa.
- **Método actionTimer():** este método será el que se ejecute tras el evento "timer_Tick". Será el encargado de llamar al método de iniciar de nuevo todas las comprobaciones mediante el método startTesting() si no está en una comprobación, es decir, esperará su tiempo y si no se han terminado las comprobaciones, no ejecutará nada. Este es el método fundamental para la ejecución en bucle. En el código 40 del Anexo podemos observar su estructura completa.
- **Método loopTesting():** este método será el que se ejecute tras el evento "btnLoopTesting_Click". Será el encargado de arrancar o parar el temporizador, así como el encargado de poner la imagen de iniciar o parar junto a un texto aclaractivo cuando corresponda. En el código 41 del Anexo podemos observar su estructura completa.
- **Método backgroundWorkerAction():** este método es muy importante, pues es el que ejecuta nuestro hilo principal. Este recorrerá nuestra tabla y, si está chequeado un registro realizará las comprobaciones que estén chequeadas, ya sean las tablas del AS400 o las plataformas. También es el encargado animar el recurso (GIF) de carga. En el código 42 del Anexo podemos observar su estructura completa.

- **Método backgroundWorkerCompleted():** este método será llamado una vez que el hilo principal ha sido completado, es decir, cuando haya realizado todas sus tareas. Habilitará el botón de Start, deshabilitará el de Stop y escribirá los tiempos de ejecución. En el código 43 del Anexo podemos observar su estructura completa.
- **Método platformTesting():** este método será el encargado de clasificar si una comprobación es local o web, por tanto, lo derivará a su correspondiente método. También será el encargado de escribir sobre la celda correspondiente el estado de la comprobación, ya sea un error, una advertencia, correcto o cargando... En el código 44 del Anexo podemos observar su estructura completa.
- **Método addTestTime():** este método será el encargado de añadir los tiempos de ejecución a las correspondientes variables en función del tiempo inicial a la plataforma pasada por parámetro. En el código 45 del Anexo podemos observar su estructura completa.
- **Método updateCell():** este método será el encargado de escribir el reporte correspondiente sobre la celdas, ya sean en la primera columna (columna "state") para mostrar los iconos indicados o en las columnas de las comprobaciones. En el código 46 del Anexo podemos observar su estructura completa.
- **Método writeExecutionTime():** este método será el encargado de escribir los tiempos de ejecución en la interfaz, mediante sus correspondientes label. En el código 47 del Anexo podemos observar su estructura completa.
- **Métodos onFrameChanged() y dgvPlatformTesting_Paint():** estos métodos son auxiliares para la animación del recurso "loading". Solo se permitirá la animación de la primera columna que es en la que se encontrará nuestro icono animado. En los códigos 48 y 49 del Anexo podemos observar su estructura completa.
- **Métodos getHotelName():** este método se encargará de dar un formato a los nombres de los hoteles debido a su longitud. En el código 50 del Anexo podemos observar su estructura completa.
- **Métodos getReportCell():** este método transforma un reporte de string a "reportCell". En el código 51 del Anexo podemos observar su estructura completa.
- **Métodos isCheckedToTest():** este método determinará si el registro está chequeado para ser enviado a las comprobaciones o no. En el código 52 del Anexo podemos observar su estructura completa.

7.2.2. Comprobación Local

La comprobación local consta de hacer un "checkeo" o reconocimiento de las tablas albergadas en la base de datos del servidor AS400 y las que se encuentran en SQL Server y comprobar que los datos estén correctamente sincronizados. Debido a la extensión de los métodos, en el Anexo se muestra el código sólo de los más cortos:

- **Método LocalTest():** Este método se encargará de obtener los valores de las tablas del AS400 y de SQL Server en función de la tabla que le llegue por parámetro y validará si tienen datos o no. En caso de tener datos, ejecutará las comprobaciones para AS400 y SQLServer mediante los métodos testAS400() y testSQL() respectivamente. En el código 53 del Anexo podemos observar su estructura.

- **Método doLocalOperation():** Este método es el más importante para la comprobación local, ya que es el que dirige las operaciones y las muestra en el LOG. Su misión consiste en, si no tiene una cancelación pendiente el hilo, insertar los registros si no se encuentran, por el contrario, si sí se encuentran deberá editar una celda del registro denominada "procesado" indicando que necesita una revisión para que, sea Baja o Modificación la operación corresponiente a realizar, un programa de la empresa lo detecte y lo corrija. Esto se hace para integrar mejor aún la aplicación con los programas ya establecidos de la empresa, este en concreto, está en contínua ejecución y, por tanto, la edición es casi instantánea. Todas las operaciones que se realicen serán indicadas en el LOG. En el código 54 del Anexo podemos observar la estructura de este método.

- **Método testAS400():** Este método se encargará de realizar toda la comprobación local de Altas y Modificaciones hacia el servidor AS400 comprobando que los datos alojados en el mismo sean los mismos que los que se alojan en el SQLServer. En caso de haber disparidades, se ejecutará la acción correspondiente mediante el método doLocalOperation().

- **Método testSQL():** Este método se encargará de realizar toda la comprobación local de Bajas hacia el servidor AS400, puesto que es el referente. Se hace en dos comprobaciones debido a que si no encontramos el mismo identificador en la primera puede que sí esté en la segunda, ya que estamos comprobando como referente, en la primera, el AS400 y cuando se acaba la tabla puede que hayan más registros que en la segunda, con lo cual debemos de hacer una segunda comprobación puesto que probablemente, se encuentren casos de Baja. En caso de haber disparidades, se ejecutará la acción correspondiente, en este caso, inserciones, mediante el método doWebOperation().

- **Método getUpdatedRowsSQL():** Este método se encarga de obtener los datos pertenecientes a la tabla del SQLServer que le pasemos.

- **Método getUpdatedRowsAS400():** Este método se encarga de obtener los datos pertenecientes a la tabla del AS400 que le pasemos.

- **Método insertRegistry():** Este método se encarga de insertar un registro en el servidor AS400 en función de la tabla que le pasemos.

- **Método sameFields():** Este método es muy importante, pues es el que comprueba que todos los campos en una fila de una tabla del AS400 y otra del SQLServer están sincronizados, en caso contrario, lo notifica.

- **Método addIdentifier():** Este método se encarga de agregar a la lista "listIdentifier" si no lo contiene, un identificador. Esto nos servirá para saber los identificadores que llevamos procesados para evitar volver a actualizar un mismo registro.

7.2.3. Comprobación Web

La comprobación web consta de hacer un "checkeo" de los precios y las disponibilidades de todas las tarifas de las diferentes plataformas asignadas, en nuestro caso, disponemos de cuatro, Mirai, Booking, OTS y Expedia. A continuación se explicarán los métodos principales de esta comprobación:

- **Método WebTest():** Este método se encargará de obtener los valores de las tablas del AS400 y de SQL Server en función de la tabla que le llegue por parámetro y validará si tienen datos o no. En caso de tener datos, ejecutará las comprobaciones para AS400 y SQLServer mediante los métodos testAS400() y testSQL() respectivamente. En el código 55 del Anexo podemos observar su estructura.

- **Método doWebOperation():** Este método es el más importante para la comprobación web, ya que es el que dirige las operaciones y las muestra en el LOG. Su misión consiste en, si no tiene una cancelación pendiente el hilo, insertar los registros si no se encuentran, por el contrario, si sí se encuentran deberá editar una celda del registro denominada "procesado" indicando que necesita una revisión para que, sea Baja o Modificación la operación corresponiente a realizar, un programa de la empresa lo detecte y lo corrija. Esto se hace para integrar mejor aún la aplicación con los programas ya establecidos de la empresa, este en concreto, está en contínua ejecución y, por tanto, la edición es casi instantánea. Todas las operaciones que se realicen serán indicadas en el LOG. En el código 56 del Anexo podemos observar la estructura de este método.

- **Método updateAvailability():** Este método será el encargado de, en función de la plataforma que le pasemos, generar el XML correspondiente, ya sea mediante una serialización o deserialización de objetos o mediante creación directa de la estructura del XML, actualizar la disponibilidad.

- **Método updatePrice():** Este método será el encargado de, en función de la plataforma que le pasemos, generar el XML correspondiente, ya sea mediante una serialización o deserialización de objetos o mediante creación directa de la estructura del XML, actualizar el precio.

- **Método getSQLPrice():** Este se trata del método más largo de todo el proyecto, pues es el encargado de, mediante una fecha, hotel, habitación, tarifa y plataforma obtener el precio real de la misma navegando a través de las complejas bases de datos que mantiene la compañía. Este calcula todas las ofertas añadidas a esos días y devuelve el precio final, entendiéndose que datos como el número de infantes, la cantidad de días o si se trata de un hotel de costa (Hotel Playa) o de ciudad (Senator), por ejemplo, pueden ser clave para formular el

precio correcto. Este método es el único que contiene parte de código de la empresa. Esto se ha realizado así debido a las constantes variaciones de las tarifas, puesto que de esta forma mantener una sincronización con las aplicaciones de la empresa y esta para el cálculo del precio es mucho más sencillo.

- **Método testMirai():** este método será el encargado de comprobar que toda la base de datos de precios y disponibilidades de la plataforma Mirai, para un rango de fechas concreto, mantenga una sincronización perfecta con los datos y disponibilidades calculadas. Todas las incidencias serán notificadas a través del Log.
- **Método testBooking():** este método será el encargado de comprobar que toda la base de datos de precios y disponibilidades de la plataforma Booking, para un rango de fechas concreto, mantenga una sincronización perfecta con los datos y disponibilidades calculadas. Todas las incidencias serán notificadas a través del Log.
- **Método testOts():** este método será el encargado de comprobar que toda la base de datos de precios y disponibilidades de la plataforma OTS, para un rango de fechas concreto, mantenga una sincronización perfecta con los datos y disponibilidades calculadas. Todas las incidencias serán notificadas a través del Log.
- **Método testExpedia():** este método será el encargado de comprobar que toda la base de datos de precios y disponibilidades de la plataforma Expedia, para un rango de fechas concreto, mantenga una sincronización perfecta con los datos y disponibilidades calculadas. Todas las incidencias serán notificadas a través del Log.

7.2.4. Forzar Inserción

Tenemos creada una clase orientada al forzado de registros para casos excepcionales y pertenecientes a la comprobación local. Esta clase es pequeña pero muy útil y en el código 57 se muestra su estructura interna completa. Esta clase lo que hace es recoger los datos de los registros del servidor “repliSQL3” y los replica en el servidor “repliSQL4” mediante el uso de su indentificador RRNMOD. Le indicaremos la tabla en la que queremos que se inserte el registro a través de un desplegable en la interfaz. Su interfaz es muy pequeña y para acceder a ella deberemos pulsar sobre el botón “Forzar inserción”. Una vez lo hayamos pulsado, nos abrirá la ventana que se aprecia en la figura 38, en la que nos encontramos un desplegable y un campo de texto en el que tendremos que insertar el RRNMOD que queramos forzar para la nueva inserción en el servidor “repliSQL4”.

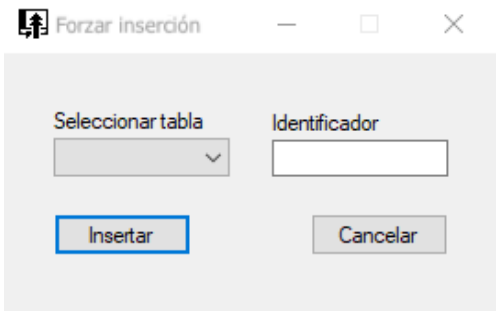


Figura 38: Interfaz de la clase "Forzar inserción".

7.2.5. Utilidades

Hemos desarrollado una clase aparte que será heredada por las demás y que contiene funciones útiles y utilizadas por todas las demás clases. Esta clase es la clase "Utilities.vb" y contiene bastantes funciones, enumeradores y listas que son utilizados para hacer una referencia correcta a las diferentes tablas, plataformas, servidores, operaciones, reportes... Gran cantidad de estas funciones y enumeradores tienen su utilidad orientada al correcto reporte en el LOG.

A continuación explicaremos los diferentes enumeradores que hemos utilizado:

- **Enumerador "Platform":** se utilizará para determinar tanto las diferentes plataformas web como tablas locales (la barra baja seguida de una C implica que se trata del segmento costero). En el código 58 podemos observar su estructura.
- **Enumerador "Server":** se utilizará para determinar si estamos tratando sobre la base de datos alojada en el AS400 o en el SQL Server. En el código 59 podemos observar su estructura.
- **Enumerador "OperationType":** se utilizará para determinar si estamos tratando una operación de actualización de disponibilidad o de precio. En el código 60 podemos observar su estructura.
- **Enumerador "LocalOperationType":** se utilizará para determinar si estamos tratando sobre una acción de Alta, Modificación o Baja para las operaciones del AS400. En el código 61 podemos observar su estructura.
- **Enumerador "ReportCell":** se utilizará para reportar correctamente la acción realizada en la celda, ya sea para mostrar que ha ocurrido un error, que está correcto, que está cargando, que ha ocurrido un fallo o ha habido disparidad, que no existe el hotel en la plataforma o que no hay contrato para ese hotel en la plataforma. En el código 62 podemos observar su estructura.
- **Enumerador "ReportLog":** se utilizará para determinar qué acción ha ocurrido para reportarla en el archivo LOG. En el código 63 podemos observar su estructura.

- **Variables globales de la clase:** En el código 64 del Anexo podemos ver todas las variables globales que hemos utilizado en esta clase.
- **Método WriteLog():** este método será el encargado de escribir en e archivo LOG haciendo uso de una función en la librería común de la empresa. Va teniendo en cuenta todos los parámetros de entrada para establecer un reporte correcto y detallado de la situación. En el código 65 del Anexo podemos observar su estructura.
- **Método haveDataThisDatatable():** este método es un auxiliar que nos determinará si una tabla tiene o no datos. En el código 66 del Anexo podemos observar su estructura.
- **Método DateRange():** este método es un auxiliar que nos obtendrá una lista de fechas en función de un rango. En el código 67 del Anexo podemos observar su estructura.
- **Método getIdUser():** este método es un auxiliar que nos obtendrá el ID de un usuario de plataforma en función de un enumerador "Platform".
- **Método getUserFromString():** este método es un auxiliar que nos devolverá el valor en forma de enumarador "Platform" en función de su valor en String.
- **Método getUser():** este método es un auxiliar que nos obtendrá el string de un usuario de plataforma en función de su ID.
- **Método addCodeCupoRoom():** este método es un auxiliar que añadirá un código de habitación para la comprobación de las disponibilidades si se puede a la lista "listCodeCupoRoom".
- **Método addCodePriceRoom():** este método es un auxiliar que añadirá un código de habitación para la comprobación de las disponibilidades si se puede a la lista "listCodePriceRoom".

7.2.6. Diagrama de clases

En esta sección se mostrará el diagrama de clases, aunque por su extensión, se ha dividido en dos figuras, concretamente en las figuras 39 y 40, para poder visualizarla con más claridad.

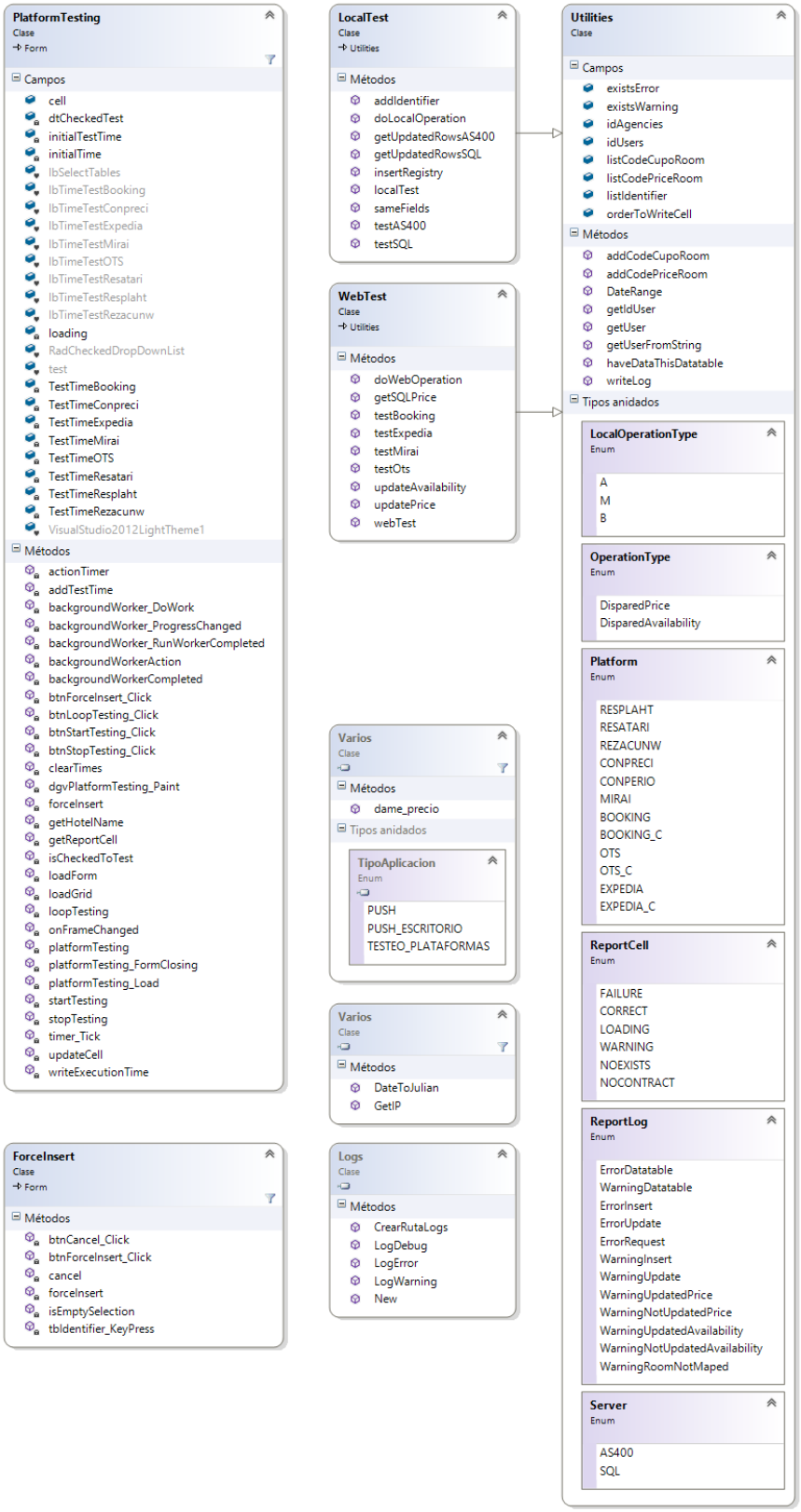


Figura 39: Diagrama de clases.

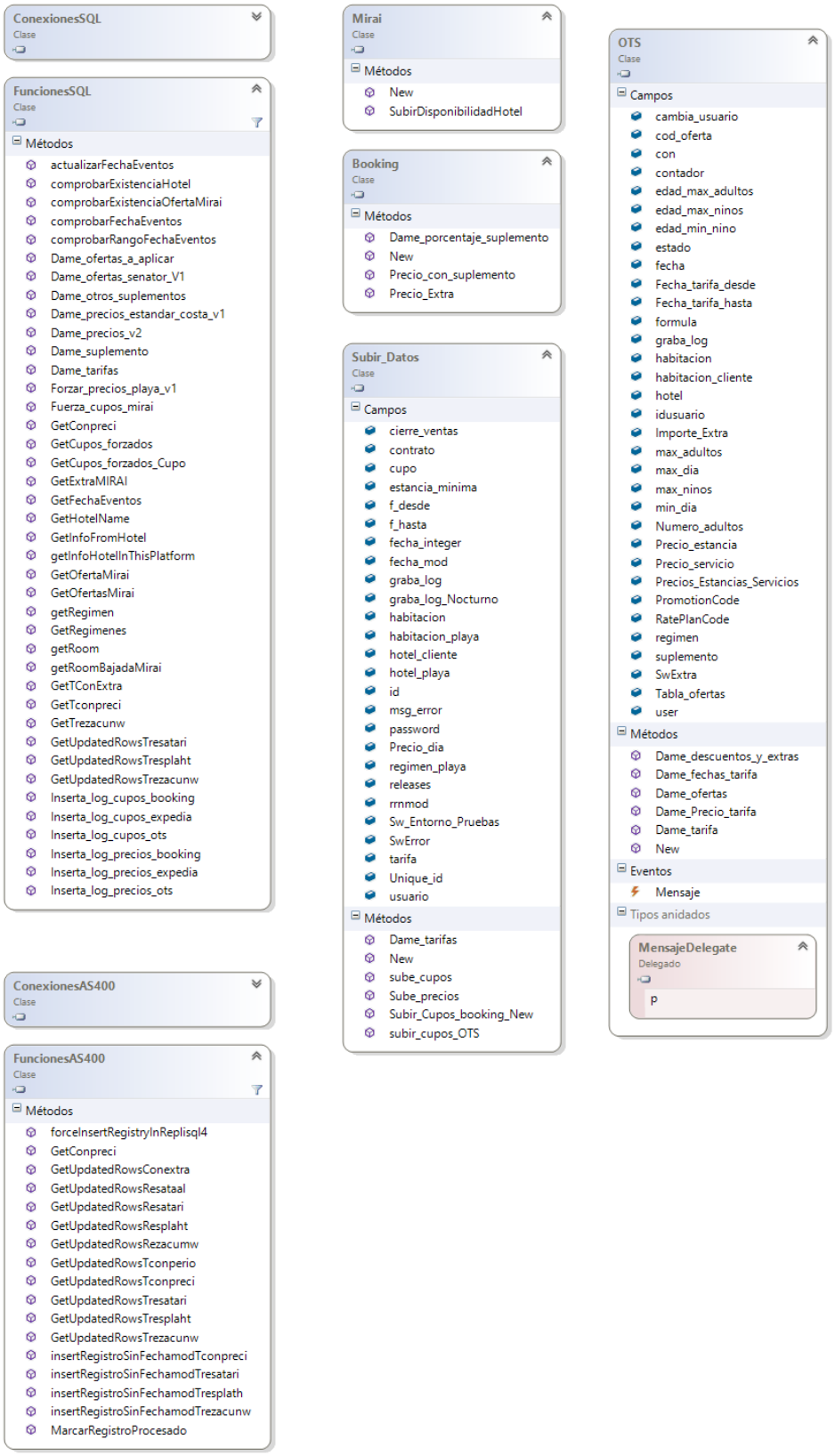


Figura 40: Diagrama de clases.

7.2.7. Log

Los logs [29] son archivos de texto normales. Estos ficheros registran todos los procesos que han sido definidos como relevantes por el programador de la aplicación. Por ejemplo, en el caso de los archivos log de una base de datos se registran todos los cambios de aquellas transacciones completadas exitosamente. Así, en caso de que un fallo del sistema elimine información de la base de datos, el log será la clave para la restauración completa de la base de datos correspondiente.

Dependiendo de su programación, los ficheros log se generan automáticamente. Sin embargo, si se cuenta con los conocimientos necesarios, también será posible crear archivos de registro propios. En general, la línea de un log contiene:

- **Evento**, por ejemplo un inicio o una acción de un programa.
- **Marca de tiempo** en la que queda registrada la fecha y la hora a la que se ha producido el evento.

Por lo general, la marca de tiempo se genera primero, con el fin de reflejar la secuencia cronológica de los eventos.

En nuestro caso, nuestro log indicará la acción que se ha realizado sobre las diversas tablas y los servidores a nivel de comprobación local y los cambios actualizados en las cuatro plataformas incorporadas, ya sea indicando actualizaciones de precios o de disponibilidad registrándose también el día que se ha actualizado. Al comienzo de la línea del log se deja la marca del tiempo para saber a qué hora y día se ha realizado dicha acción. Podemos ver a continuación un ejemplo de un log de tipo "Warning", en el que se muestran incidencias asociadas directamente con disparidades, en la figura 41.

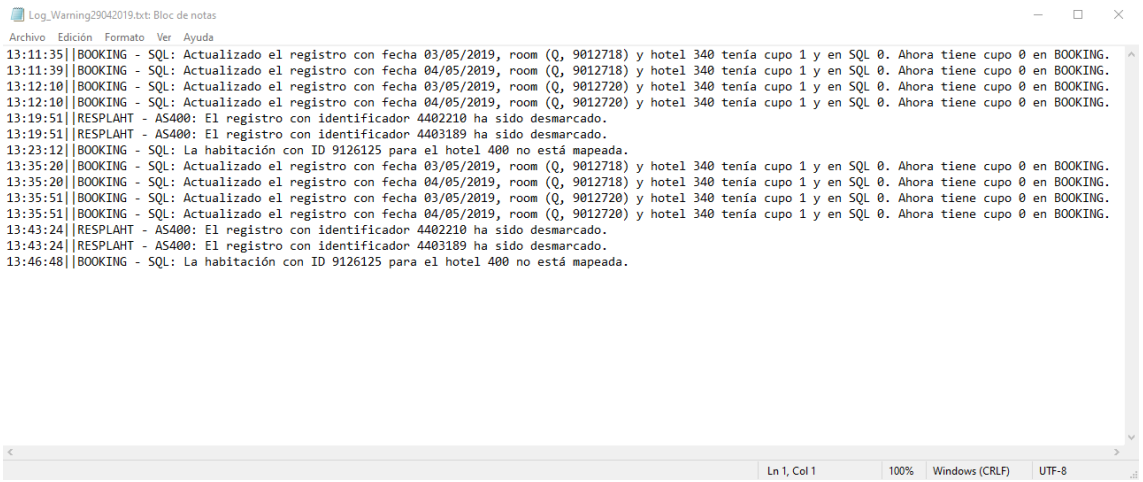


Figura 41: Ejemplo de log de tipo "Warning".

En nuestro caso, podemos observar que no mostramos la fecha sino que solo escribimos la hora debido a que el mismo archivo log contiene en su nombre esta fecha. Esto nos sirve para observar con facilidad si un día contiene o no alguna disparidad, ya que si no existen directamente no se crea este archivo log.

Podemos ver a continuación un ejemplo de un log de tipo "Error", en el que se muestran errores asociados directamente con fallos de base de datos u otros, en la figura 42.

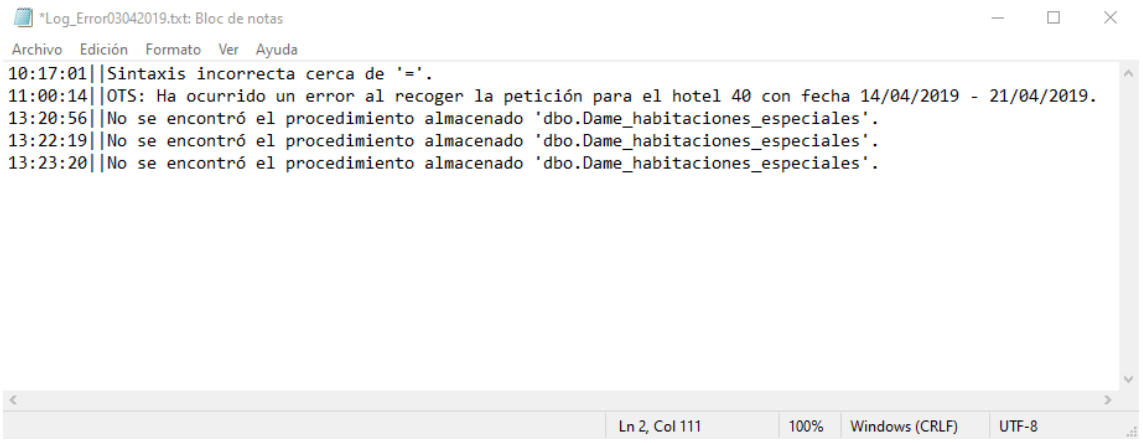


Figura 42: Ejemplo de log de tipo "Error".

7.2.8. ClickOnce

ClickOnce [30] es una tecnología de implementación que permite crear aplicaciones basadas en Windows de actualización automática que pueden instalarse y ejecutarse con una mínima interacción del usuario. Visual Studio proporciona soporte completo para publicar y actualizar aplicaciones implementadas con tecnología ClickOnce si se ha desarrollado el proyecto con Visual Basic y Visual C .

A continuación mostraremos cómo se activa ClickOnce a través de Visual Studio.

En primer lugar deberemos dirigirnos a las propiedades del proyecto sobre la opción "Firma", situada en la barra lateral, y activar la firma de de los manifiestos de ClickOnce. Una vez activado, deberemos introducir un certificado si lo tenemos, en caso contrario, crearemos uno de prueba pulsando sobre el botón "Crear certificado de prueba" tal y como podemos observar en la figura 43.

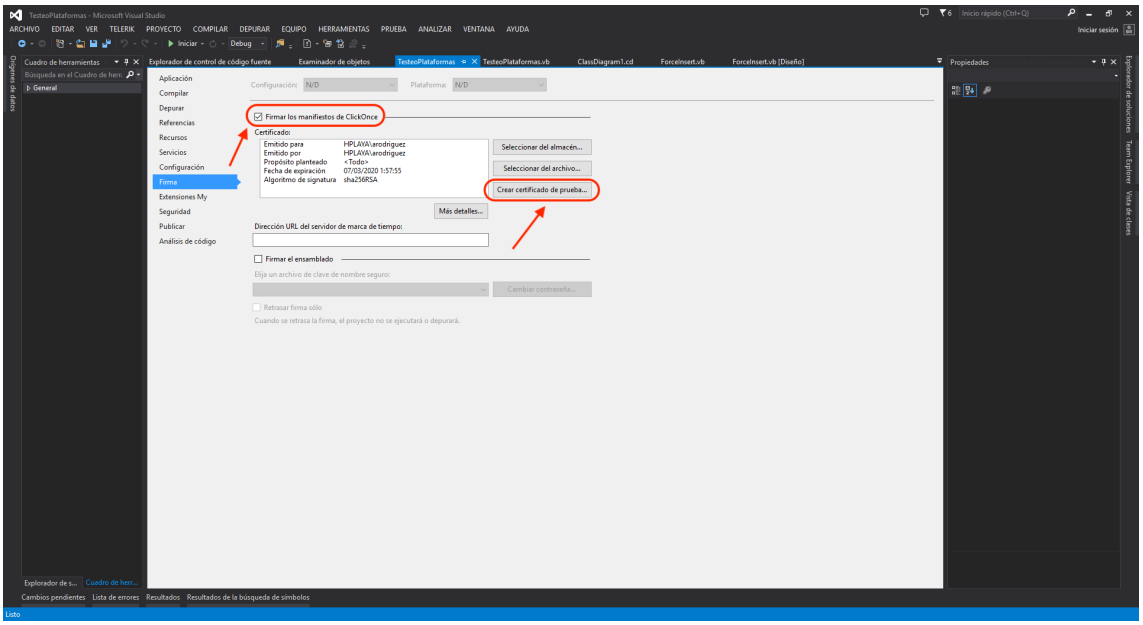


Figura 43: Firmar los manifiestos de ClickOnce.

Una vez firmados los manifiestos de ClickOnce deberemos dirigirnos esta vez a la opción "Publicar". Una vez allí deberemos indicar donde estará la carpeta de la publicación. La publicación cobra mayor sentido si se realiza sobre un repositorio web o carpeta en red ya que podrá actualizarse en todos los ordenadores que puedan acceder a este, de lo contrario, si se realiza sobre un repositorio local, en el que solo tiene acceso el ordenador local, evidentemente, este será el único que se actualizará ya que los demás no tienen acceso. En nuestro caso se realizará sobre una carpeta en red cuya dirección es la siguiente:

\\192.168.3.70\C$\Repositorio\TesteoPlataformas\

Una vez introducimos la dirección pulsaremos sobre el botón "Publicar ahora". Veremos los requisitos previos de nuestra aplicación mediante el botón "Requisitos previos...". Esta opción es muy interesante y necesaria en nuestro caso ya que se puede ver qué archivos externos se van a adjuntar con el proyecto. En general, Visual Studio adjuntará todos los que considere necesarios y no habrá ningún problema pero no siempre es así, un ejemplo de ello es nuestro caso, ya que utilizamos la librería "Telerik" para componentes gráficos y no se adjunta automáticamente, para remediarlo, pulsaremos sobre el estado de la publicación y seleccionaremos la opción "Incluir", tal y como podemos observar en la figura 44.

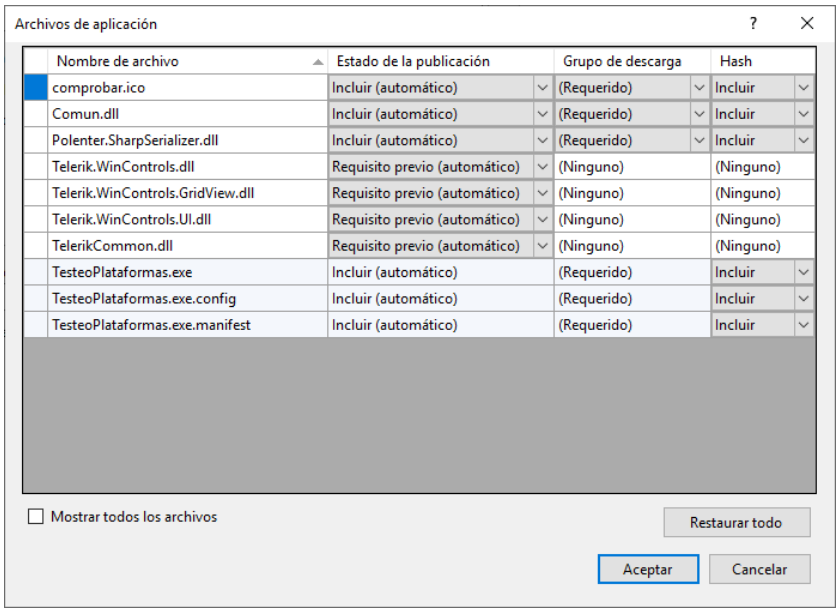


Figura 44: Requisitos previos de la aplicación.

Para nuestra App deseamos que esta busque nuevas actualizaciones al inicio de su ejecución, para ello deberemos presionar el botón “Actualizaciones...” y en la ventana que nos aparece deberemos activar la primera casilla que indica que la aplicación debe buscar actualizaciones y seleccionar la opción “Antes de que se inicie la aplicación”.

Para editar la información que se adjuntará a la aplicación pulsaremos sobre el botón “Opciones...” y editaremos todos los campos que nos aparezcan. En nuestro caso hemos introducido como nombre del editor “Senator Hotels & Resorts”, pues esta aplicación ha sido creada para ellos y como trabajador de la empresa, como nombre del conjunto de aplicaciones y de producto utilizaremos un nombre más corto, utilizaremos “Testeo Plataformas”. Esta ventana es la que se aprecia en la figura 45.

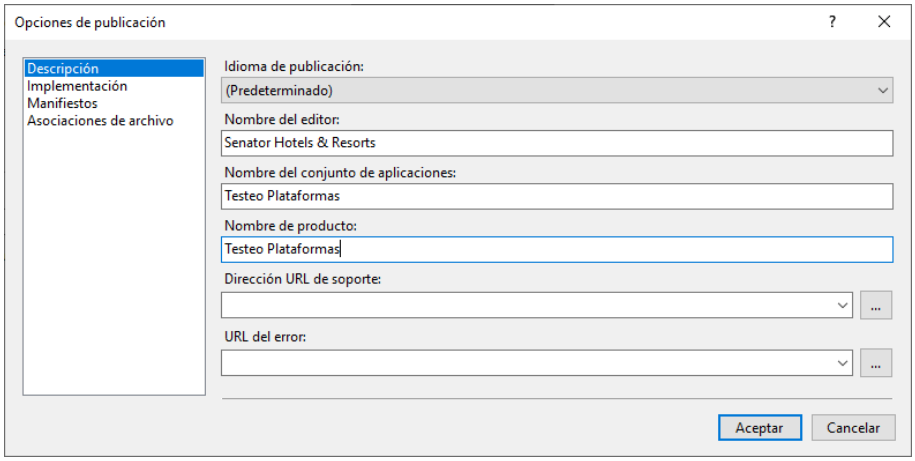


Figura 45: Información de la aplicación.

Nos dirigiremos a la opción “Manifiestos” y activaremos la opción que nos creará un acceso directo en el escritorio para que en la instalación se genere dicho archivo.

Una vez hayamos realizado los pasos anteriores, prodecederemos a realizar el proceso de publicación, para ello pulsaremos sobre el botón “Asistente para publicación”, el cual nos guiará durante todo el proceso de publicación. En su inicio nos nos abrirá la ventana que se muestra en la figura 46, la cual nos indica la ruta de acceso.

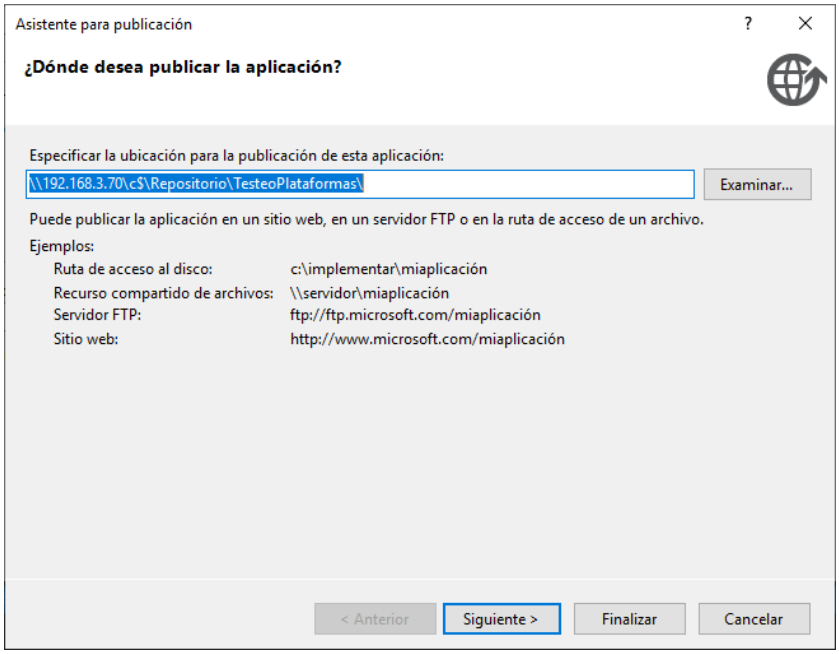


Figura 46: Comienzo de la publicación de la aplicación.

Pulsaremos sobre el botón “Siguiente” y nos aparecerá una nueva ventana en la que deberemos indicar que la aplicación estará disponible en línea y sin conexión.

Pulsaremos sobre el botón “Siguiente” y nos aparecerá otra ventana en la que el asistente nos informa de la ruta de publicación de la aplicación.

Una vez pulsado el botón “Finalizar” la aplicación será publicada. Tras la publicación, prodederemos a instalarla como haría cualquier usuario que necesitase acceder a ella. Nos dirigiremos al repositorio a través del sistema de archivos de nuestro ordenador en la misma ruta de instalación que antes:

\\192.168.3.70\C$\Repositorio\TesteoPlataformas\

Una vez aquí, nos encontraremos con que la carpeta no está vacía, sino que contiene dos archivos de instalación y una carpeta en la que se almacenarán todos los archivos necesarios para la instalación, en función de la versión, ya que en esta carpeta se almacenarán todas las versiones publicadas de la aplicación.

Para instalar la aplicación deberemos abrir su archivo de instalación, el cual tiene, en nuestro caso, el nombre "TesteoPlataformas.application", y deberemos seguir los pasos que nos indica dicho instalador. Al abrir dicho archivo deberemos de indicar que queremos instalar la aplicación aunque nos informe de que no se puede comprobar el fabricante.

En la figura 47 podemos observar que se trata de una instalación normal de cualquier otra aplicación.

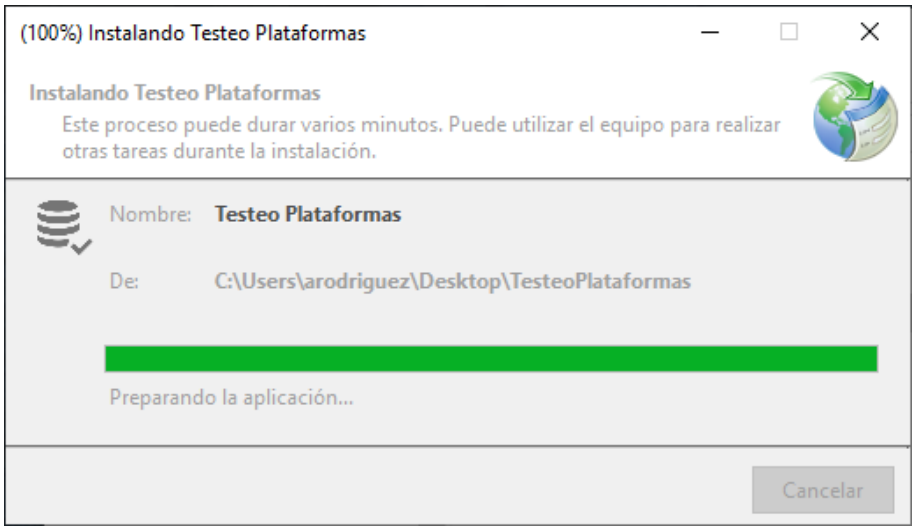


Figura 47: Progreso de instalación de la aplicación Testeo Plataformas.

Una vez hayamos instalado la aplicación se generará un acceso directo de la misma en el escritorio. Abrimos la aplicación y nos aparecerá una ventana en la que nos está indicando que está buscando una nueva actualización. Esta ventana se puede apreciar en la figura 48.

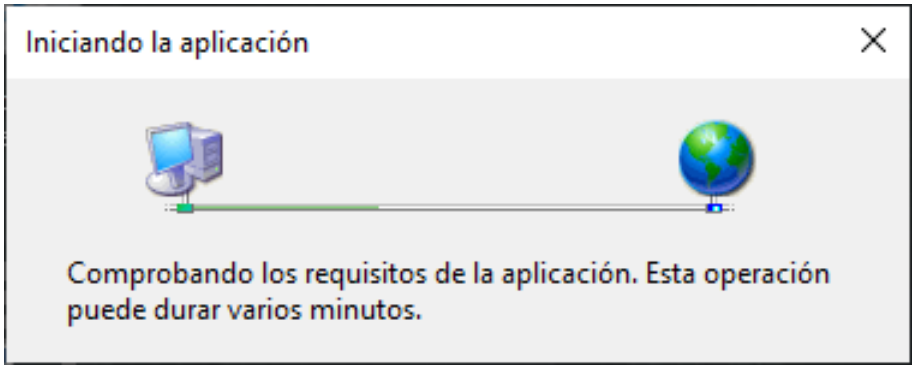


Figura 48: Buscando nuevas actualizaciones.

7.2.9. Microsoft Team Foundation Server

Team Foundation Server [31] no es símplemente una herramienta para el control de versiones de código fuente: su alcance va mucho más allá, facilitando la gestión completa del ciclo de vida de la aplicación, desde la fase de diseño hasta las pruebas, pasando por la integración continua o la calidad del código.

TFS incorpora varios sistemas integrados: por un lado una base de datos en SQL Server que contiene no sólo el código fuente de nuestras aplicaciones sino los elementos de trabajo que posibilitan el seguimiento del desarrollo; los datos se integran en un Data Warehouse de SQL Server Analysis Services que proporciona información sobre el estado del proyecto mediante informes en Reporting Services o Excel; el motor de compilación Team Build permite la compilación desatendida de los proyectos y genera informes de calidad de la compilación; y todo ello se puede integrar en portales de colaboración de Microsoft Sharepoint para que todo el equipo pueda compartir información, documentos o calendarios asociados al proyecto.

El acceso a TFS se realiza desde el add-in Team Explorer incluido en Visual Studio, pero además es posible acceder a los datos a traves de un web propio o a través de la integración con Excel y Microsoft Project. Team Foundation Server es la herramienta definitiva para la gestión completa de todos los aspectos de una aplicación de cualquier tamaño.

En nuestro caso, tenemos nuestro proyecto enlazado al Team Foundation Server del departamento de desarrollo de la empresa. Podemos proteger nuestro proyecto dirigiéndonos a la opción "Team Explorer". En esta opción tenemos la posibilidad de escribir un comentario y seleccionar los cambios que queremos subir y los que no. Podemos observar esta opción en la figura 49.

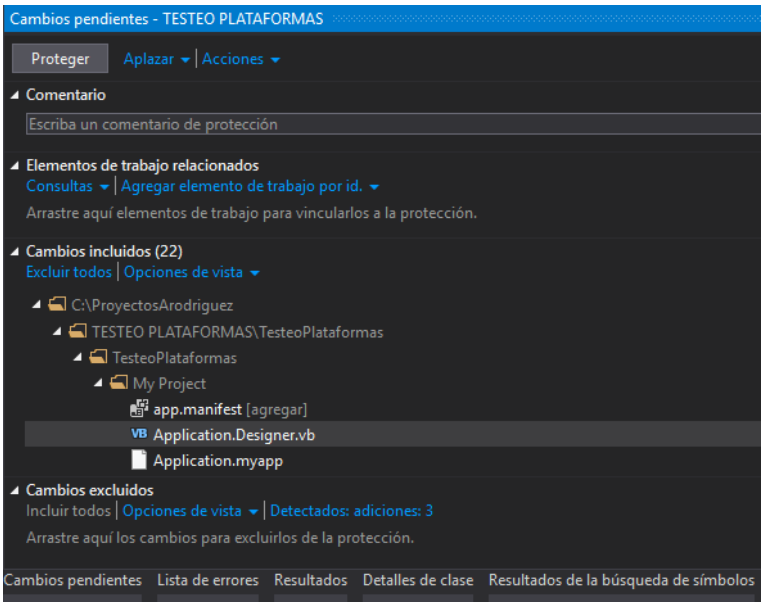


Figura 49: Proteger cambios mediante Microsoft Team Foundation Server.

Si pulsamos sobre la opción "Explorador de control de código fuente" nos aparecerá toda la jerarquía de carpetas y proyectos enlazados al Team Foundation Server. En la figura 50 se muestra nuestra jerarquía de carpetas y proyectos enlazados.

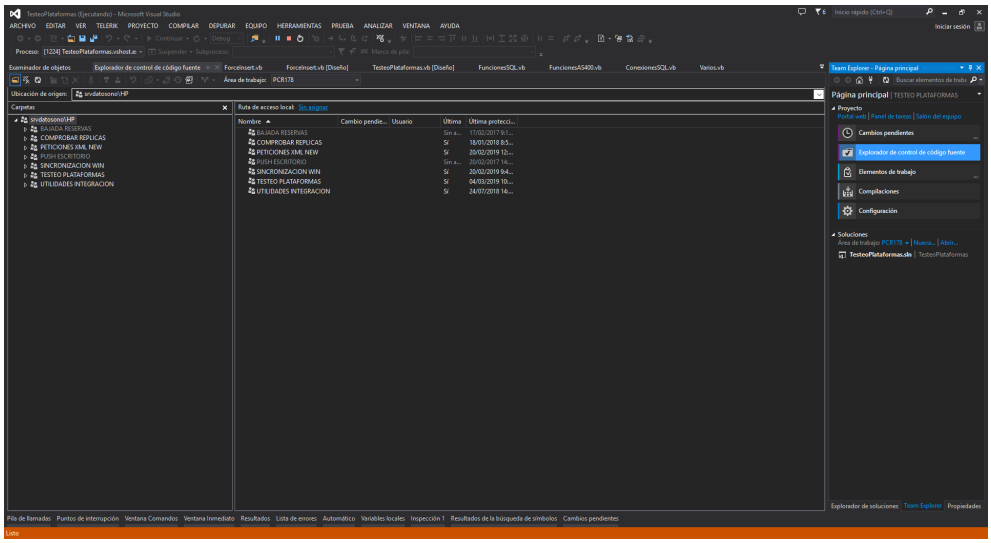
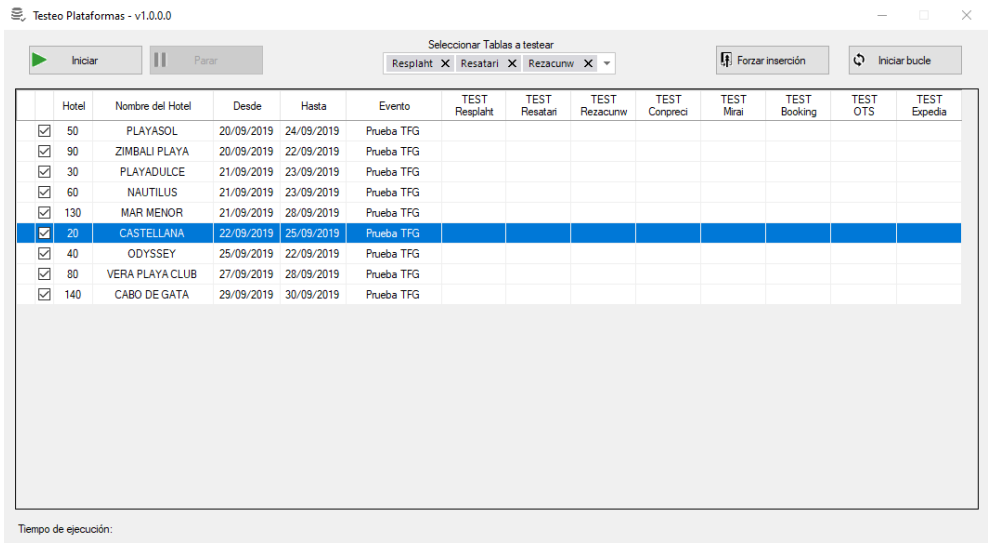


Figura 50: Explorador de control de código fuente.

7.3. Funcionamiento general

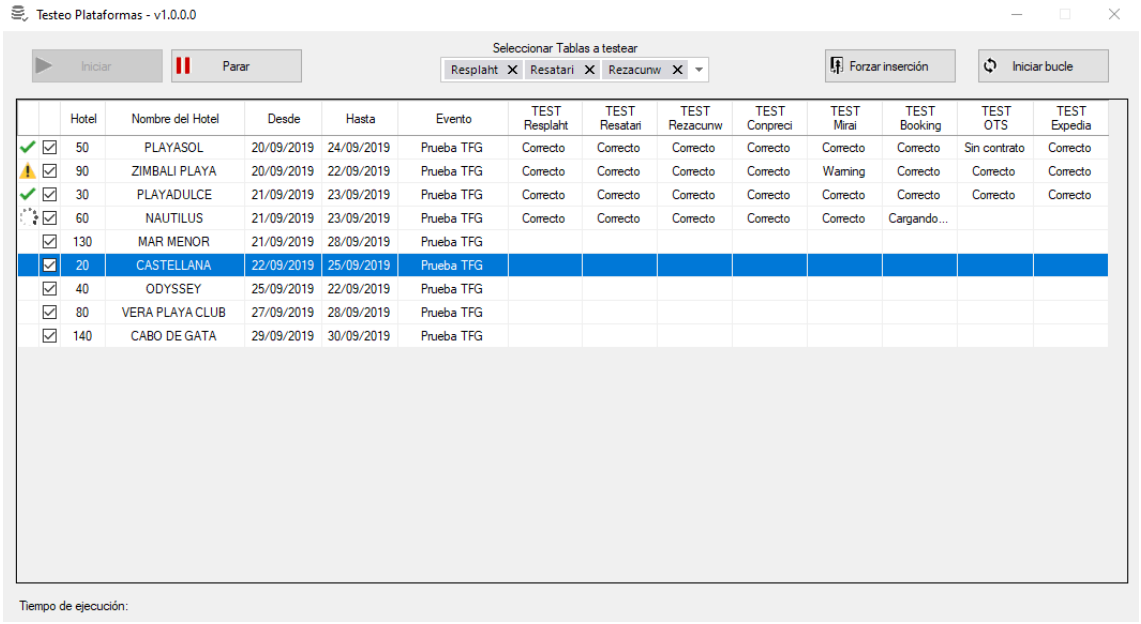
Nuestro proyecto tiene un funcionamiento general sencillo para el usuario, pues hemos prestado especial interés en que esto sea así, ya que en la empresa se requiere de aplicaciones con interfaces sencillas aunque con operaciones internas complejas para no "perder el tiempo" si se puede simplificar. El proyecto tiene en cuenta, como ya sabemos, la tabla fecha eventos para operar y, en su inicio, la carga en el grid, estando paradas las comprobaciones tal y como podemos observar en la figura 51.



	Hotel	Nombre del Hotel	Desde	Hasta	Evento	TEST Resplaht	TEST Resatari	TEST Rezacunw	TEST Conpreci	TEST Mirai	TEST Booking	TEST OTS	TEST Expedia
☑	50	PLAYASOL	20/09/2019	24/09/2019	Prueba TFG								
☑	90	ZIMBALI PLAYA	20/09/2019	22/09/2019	Prueba TFG								
☑	30	PLAYADULCE	21/09/2019	23/09/2019	Prueba TFG								
☑	60	NAUTILUS	21/09/2019	23/09/2019	Prueba TFG								
☑	130	MAR MENOR	21/09/2019	28/09/2019	Prueba TFG								
☑	20	CASTELLANA	22/09/2019	25/09/2019	Prueba TFG								
☑	40	ODYSSEY	25/09/2019	22/09/2019	Prueba TFG								
☑	80	VERA PLAYA CLUB	27/09/2019	28/09/2019	Prueba TFG								
☑	140	CABO DE GATA	29/09/2019	30/09/2019	Prueba TFG								

Figura 51: Proyecto con las comprobaciones paradas.

Una vez pulsemos el botón start, se irán rellenando las celdas que están vacías de izquierda a derecha y de arriba a abajo hasta completar el grid. Si en alguna comprobación se realiza alguna operación para ese registro en esa tabla/plataforma, se generará una línea en el log, se escribirá en la celda correspondiente a esa tabla/plataforma - registro y se escribirá, una vez pase al siguiente registro del grid, en la columna el icono correspondiente. Si, por el contrario, no se ha tenido que realizar ninguna operación significa que todo está correcto y se indicará. Una vez paren las comprobaciones se escribirá el tiempo total de las comprobaciones en la parte inferior izquierda y debajo de cada columna tabla/plataforma aparecerá un tiempo medio de ejecución, para poder evaluar a simple vista la tardanza de peticiones web, por ejemplo. Podemos ver en la figura 52 cómo quedaría un ejemplo de ejecución.



		Hotel	Nombre del Hotel	Desde	Hasta	Evento	TEST Resplaht	TEST Resatari	TEST Rezacunw	TEST Conpreci	TEST Mirai	TEST Booking	TEST OTS	TEST Expedia
✓	☑	50	PLAYASOL	20/09/2019	24/09/2019	Prueba TFG	Correcto	Correcto	Correcto	Correcto	Correcto	Correcto	Sin contrato	Correcto
⚠	☑	90	ZIMBALI PLAYA	20/09/2019	22/09/2019	Prueba TFG	Correcto	Correcto	Correcto	Correcto	Warning	Correcto	Correcto	Correcto
✓	☑	30	PLAYADULCE	21/09/2019	23/09/2019	Prueba TFG	Correcto	Correcto	Correcto	Correcto	Correcto	Correcto	Correcto	Correcto
◌	☑	60	NAUTILUS	21/09/2019	23/09/2019	Prueba TFG	Correcto	Correcto	Correcto	Correcto	Correcto	Cargando...		
	☑	130	MAR MENOR	21/09/2019	28/09/2019	Prueba TFG								
	☑	20	CASTELLANA	22/09/2019	25/09/2019	Prueba TFG								
	☑	40	ODYSSEY	25/09/2019	22/09/2019	Prueba TFG								
	☑	80	VERA PLAYA CLUB	27/09/2019	28/09/2019	Prueba TFG								
	☑	140	CABO DE GATA	29/09/2019	30/09/2019	Prueba TFG								

Figura 52: Ejemplo de ejecución del proyecto.

En el caso excepcional de que hubieran problemas con algún registro en concreto de las comprobaciones locales y no se solucionase mediante nuestras operaciones, existe la opción de forzar una inserción, tal y como se ha explicado con anterioridad.

8. Conclusiones y trabajos futuros

Gracias a este proyecto he podido aprender con mayor profundidad sobre los lenguajes Visual Basic .NET, SQL y XML, desarrollar un mejor manejo del entorno de desarrollo Visual Studio y de los servicios web y comprender la importancia que tienen en la actualidad.

Este proyecto me ha enseñado la importancia que tiene una perfecta sincronización entre las diferentes herramientas que hay en el mercado y las bases de datos, puesto que gracias a esto, se puede crear un ecosistema, en nuestro caso basado en las reservas online, con mucha robustez y a la vez flexible, pues la programación así lo permite. Para lograr este tipo de ecosistema se necesita muchas horas de trabajo probando hasta que las cosas salgan al 100 %, puesto que, tratándose de precios y disponibilidades, en un sistema informático en el cual tras un segundo (literalmente) se pueden modificar 100 precios y disponibilidades diferentes, puede provocar una pérdida de dinero impresionante a la compañía, por lo tanto, el trabajo debe ser evaluado con mucha cautela y perfección.

Este proyecto también me ha enseñado que hay muchas más cosas por aprender todavía y que estoy esperando poder adquirir esas nuevas competencias y enseñanzas. Además, el desarrollo de todo el proyecto dentro de la empresa me ha ayudado mucho a comprender cómo es el trabajar como uno más dentro del mundo laboral. El apoyo consistente de mi tutora de prácticas y cotutora del proyecto, sobretodo a comprender la estructura de las bases de datos de la compañía y a poder seguir adelante sin complicaciones extremas.

El proyecto será de ayuda para la empresa para poder tener un mejor control sobre la sincronización de las reservas, aunque en un futuro será transformado en un servicio web para que esté siendo ejecutado en segundo plano y así, mediante la aplicación "Peticiones XML" tener un control absoluto de todo solo incorporando las nuevas fechas y automáticamente el proyecto realizaría las comprobaciones correspondientes. Esto forma parte del mantenimiento y soporte que tendrá la aplicación, pues su desarrollo es contínuo por mi parte puesto que siempre se le van incorporando nuevas funciones o nuevos cálculos de precios. Esto es algo nuevo para mí pues, hasta ahora, los trabajos requeridos tenían un principio y un fín, cuando conseguías el objetivo habías terminado la aplicación o proyecto que se te haya mandado durante la carrera universitaria pero, en el mundo laboral, te das cuenta de que no es así, pues los requisitos de la aplicación pueden variar incluso cuando ya está terminada.

9. Anexo

En este Anexo se muestran algunos de los códigos desarrollados en este trabajo en los distintos lenguajes de programación utilizados.

API de Mirai

```xml
<?xml version="1.0" encoding="utf-8"?>
<inventoryUpdate hotelId="HotelId">
  <RoomID id="RoomID">
    <rate currency="EUR">
      <planning date="date" unitPrice="unitPrice" minimumStay="minimumStay"/>
    </rate>
    <inventory><quantity>quantity</quantity></inventory>
  </RoomID>
</inventoryUpdate>
```

Código 1: Estructura XML para la actualización de valores de Mirai.

API de Booking

```xml
<request>
  <username>username</username>
  <password>password</password>
  <hotel_id>hotel_id</hotel_id>
  <version>1.0</version>
  <number_of_days>number_of_days</number_of_days>
  <start_date>start_date</start_date>
  <room_level>1</room_level>
</request>
```

Código 2: Parámetros a enviar a la API “roomrateavailability”.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<request>
  <username>username</username>
  <password>password</password>
  <hotel_id>hotel_id</hotel_id>
  <version>1.0</version>
  <room id="room_id">
    <date from ="from" to="to">
      <rate id="rate_id"/>
      <price>price</price>
      <closed>closed</closed>
      <quantity>quantity</quantity>
      <minimumStay>minimumStay</minimumStay>
      <maximumStay>maximumStay</maximumStay>
    </date>
  </room>
</request>
```

Código 3: Estructura XML para la actualización de valores de Booking.

Estructura Interna de la Extensión de la Aplicación - Métodos

```
#Region "Fecha Eventos"
...
#End Region
```

Código 4: Creación de una región en Visual Basic.

```
Private Sub CargarFechaEventos()
  HabilitarEdicionFechaEventos(True)
  RecuperarTextBoxFechaEventos()
  HabilitarBotonesFechaEventos(False)
  CargarGridFechaEventos()
End Sub
```

Código 5: Método CargarFechaEventos.

```
Private Sub HabilitarEdicionFechaEventos(ByVal habilitar As Boolean)
  If habilitar Then
    tsbNuevoFechaEventos.Enabled = True
    tsbEditFechaEventos.Enabled = True
    tsbCancelarFechaEventos.Enabled = False
    tsbGuardarFechaEventos.Enabled = False
    tsbEliminarFechaEventos.Enabled = True
  Else
    tsbNuevoFechaEventos.Enabled = False
    tsbEditFechaEventos.Enabled = False
    tsbCancelarFechaEventos.Enabled = True
    tsbGuardarFechaEventos.Enabled = True
    tsbEliminarFechaEventos.Enabled = False
  End If
End Sub
```

Código 6: Método HabilitarEdicionFechaEventos.

```
Private Sub RecuperarTextBoxFechaEventos()
  If dgvFechaEventos.SelectedRows.Count <> 0 Then
    Dim row As DataGridViewRow = dgvFechaEventos.SelectedRows.Item(0)
    hotelFechaEventos.Text = row.Cells("Hotel").Value
    DateTimePickerDesdeFechaEventos.Value = Convert.ToDateTime(row.Cells("Desde").Value)
    DateTimePickerHastaFechaEventos.Value = Convert.ToDateTime(row.Cells("Hasta").Value)
    descripcionFechaEventos.Text = row.Cells("Descripcion").Value
  End If
End Sub
```

Código 7: Método RecuperarTextBoxFechaEventos.

```
Private Sub HabilitarBotonesFechaEventos(ByVal habilitar As Boolean)
  If habilitar Then
    hotelFechaEventos.Enabled = True
    DateTimePickerDesdeFechaEventos.Enabled = True
    DateTimePickerHastaFechaEventos.Enabled = True
    descripcionFechaEventos.Enabled = True
  Else
    hotelFechaEventos.Enabled = False
    DateTimePickerDesdeFechaEventos.Enabled = False
    DateTimePickerHastaFechaEventos.Enabled = False
    descripcionFechaEventos.Enabled = False
  End If
End Sub
```

Código 8: Método HabilitarBotonesFechaEventos.

```
Private Sub CargarGridFechaEventos()
  Dim dt As DataTable = FuncionesSQL.GetFechaEventos()
  For Each row As DataRow In dt.Rows
    AddFilaGridFechaEventos(row.Item("hotel"), row.Item("fecha_inicio"), _
                            row.Item("fecha_fin"), row.Item("descripcion"))
  Next
End Sub
```

Código 9: Método CargarGridFechaEventos.

```
Private Sub AddFilaGridFechaEventos(ByVal hotel As Integer, ByVal desde As Date, ByVal
    hasta As Date, ByVal descripcion As String)
  Dim desdeAux As String() = desde.ToString.Split(New Char() {" "c})
  Dim hastaAux As String() = asta.ToString.Split(New Char() {" "c})
  dgvFechaEventos.Rows.Add(hotel, desdeAux.GetValue(0), _
                           hastaAux.GetValue(0), descripcion)
End Sub
```

Código 10: Método AddFilaGridFechaEventos.

```
Private Sub ActualizarFilaSeleccionadaGridFechaEventos(ByVal hotel As Integer, ByVal desde
    As Date, ByVal hasta As Date, ByVal descripcion As String)
  Dim desdeAux As String() = desde.ToString.Split(New Char() {" "c})
  Dim hastaAux As String() = hasta.ToString.Split(New Char() {" "c})

  dgvFechaEventos.SelectedRows.Item(0).Cells("Hotel").Value = hotel
  dgvFechaEventos.SelectedRows.Item(0).Cells("Desde").Value = desdeAux.GetValue(0)
  dgvFechaEventos.SelectedRows.Item(0).Cells("Hasta").Value = hastaAux.GetValue(0)
  dgvFechaEventos.SelectedRows.Item(0).Cells("Descripcion").Value = descripcion
End Sub
```

Código 11: Método ActualizarFilaSeleccionadaGridFechaEventos.

```
Private Sub BorrarTextBoxFechaEventos()
  If dgvFechaEventos.SelectedRows.Count <> 0 Then
    hotelFechaEventos.Clear()
    DateTimePickerDesdeFechaEventos.Value = Date.Today
    DateTimePickerHastaFechaEventos.Value = Date.Today
    descripcionFechaEventos.Clear()
  End If
End Sub
```

Código 12: Método BorrarTextBoxFechaEventos.

```
Private Sub EliminarFechaEventos()
  If dgvFechaEventos.SelectedRows.Count = 0 Then
    MessageBox.Show("No hay filas seleccionadas.", "Error", MessageBoxButtons.OK)
  Else
    For Each row As DataGridViewRow In dgvFechaEventos.SelectedRows

      FuncionesSQL.EliminarFechaEventos(row.Cells("Hotel").Value, _
                                        row.Cells("Desde").Value, _
                                        row.Cells("Hasta").Value)
    Next
  End If
End Sub
```

Código 13: Método EliminarFechaEventos.

```
Private Sub RefrescarFechaEventos()
  dgvFechaEventos.Rows.Clear()
  CargarGridFechaEventos()
End Sub
```

Código 14: Método RefrescarFechaEventos.

Estructura Interna de la Extensión de la Aplicación - Eventos

```
Private Sub tsbNuevoFechaEventos_Click_1(sender As Object, e As EventArgs) Handles
    tsbNuevoFechaEventos.Click
  estaEditando = False
  HabilitarEdicionFechaEventos(False)
  HabilitarBotonesFechaEventos(True)
  BorrarTextBoxFechaEventos()
End Sub
```

Código 15: Evento tsbNuevoFechaEventos.Click.

```
Private Sub tsbEditFechaEventos_Click_1(sender As Object, e As EventArgs) Handles
    tsbEditFechaEventos.Click
  If dgvFechaEventos.SelectedRows.Count = 0 Then
    MessageBox.Show("No hay ninguna fila seleccionada.", "Error", MessageBoxButtons.OK)
  ElseIf dgvFechaEventos.SelectedRows.Count > 1 Then
    MessageBox.Show("Solo se puede editar una fila.", "Error", MessageBoxButtons.OK)
  Else
    estaEditando = True
    HabilitarEdicionFechaEventos(False)
    HabilitarBotonesFechaEventos(True)
    RecuperarTextBoxFechaEventos()
  End If
End Sub
```

Código 16: Evento tsbNuevoFechaEventos.Click.

```
Private Sub tsbCancelarFechaEventos_Click_1(sender As Object, e As EventArgs) Handles
    tsbCancelarFechaEventos.Click
  HabilitarEdicionFechaEventos(True)
  HabilitarBotonesFechaEventos(False)
End Sub

```

Código 17: Evento tsbCancelarFechaEventos.Click.

```
Private Sub tsbEliminarFechaEventos_Click_1(sender As Object, e As EventArgs) Handles
    tsbEliminarFechaEventos.Click
  Dim mensaje As String="Esta seguro de querer eliminar la fecha de evento seleccionada?"
  If dgvFechaEventos.SelectedRows.Count > 1 Then
    mensaje = "Esta seguro de querer eliminar las fechas de evento seleccionadas?"
  End If
  If MessageBox.Show(mensaje, "Confirmacion", MessageBoxButtons.YesNo) = Windows.Forms.
      DialogResult.Yes Then
    EliminarFechaEventos()
    RefrescarFechaEventos()
  End If
End Sub
```

Código 18: Evento tsbEliminarFechaEventos.Click.

```
Private Sub tsbRefrescarFechaEventos_Click_1(sender As Object, e As EventArgs) Handles
    tsbRefrescarFechaEventos.Click
  RefrescarFechaEventos()
End Sub
```

Código 19: Evento tsbRefrescarFechaEventos.Click.

```
Private Sub tsbGuardarFechaEventos_Click_1(sender As Object, e As EventArgs) Handles
    tsbGuardarFechaEventos.Click
  If hotelFechaEventos.Text = "" Then
    MessageBox.Show("Por favor, rellene correctamente el campo hotel.", "Error",
      MessageBoxButtons.OK)
  ElseIf DateTime.Compare(DateTimePickerDesdeFechaEventos.Value, (
      DateTimePickerHastaFechaEventos.Value)) = 1 Or _
    DateTime.Compare(DateTimePickerDesdeFechaEventos.Value, DateTime.Today) = -1 Then
    MessageBox.Show("El rango de fechas es invalido.", "Error", MessageBoxButtons.OK)
  End If

  Dim bSuccess As Boolean = True
  Dim editarDescripcion As Boolean = False
    bSuccess = FuncionesSQL.comprobarExistenciaHotel(hotelFechaEventos.Text)

  'ESTO SE HACE POR SI SOLO SE QUIERE MODIFICAR LA DESCRIPCION'
  If estaEditando Then
    Dim row As DataGridViewRow = dgvFechaEventos.SelectedRows.Item(0)
    If hotelFechaEventos.Text = dgvFechaEventos.SelectedRows.Item(0).Cells("Hotel").Value
      And DateTimePickerDesdeFechaEventos.Value.ToString = Convert.ToDateTime(
      dgvFechaEventos.SelectedRows.Item(0).Cells("Desde").Value.ToString) And
      DateTimePickerHastaFechaEventos.Value.ToString = Convert.ToDateTime(
      dgvFechaEventos.SelectedRows.Item(0).Cells("Hasta").Value.ToString) Then
      editarDescripcion = True
      bSuccess = True
    End If
  End If

  If bSuccess Then
    bSuccess = FuncionesSQL.comprobarFechaEventos(hotelFechaEventos.Text,
      DateTimePickerDesdeFechaEventos.Value, DateTimePickerHastaFechaEventos.Value)
    If bSuccess And editarDescripcion = False Then
      MessageBox.Show("Ya existe esa fecha de evento.", "Error", MessageBoxButtons.OK)
    Else
      bSuccess = FuncionesSQL.comprobarRangoFechaEventos(hotelFechaEventos.Text,
      DateTimePickerDesdeFechaEventos.Value, DateTimePickerHastaFechaEventos.Value)
      If bSuccess = False Then
        MessageBox.Show("El rango de fechas para este hotel es invalido.", "Error",
      MessageBoxButtons.OK)
      ElseIf estaEditando Then
        Dim row As DataGridViewRow = dgvFechaEventos.SelectedRows.Item(0)
        bSuccess = FuncionesSQL.actualizarFechaEventos(hotelFechaEventos.Text,
      DateTimePickerDesdeFechaEventos.Value, DateTimePickerHastaFechaEventos.Value,
      descripcionFechaEventos.Text, CInt(row.Cells("Hotel").Value), Convert.ToDateTime(row.
      Cells("Desde").Value), Convert.ToDateTime(row.Cells("Hasta").Value))
        ActualizarFilaSeleccionadaGridFechaEventos(hotelFechaEventos.Text,
      DateTimePickerDesdeFechaEventos.Value, DateTimePickerHastaFechaEventos.Value,
      descripcionFechaEventos.Text)
      Else
        bSuccess = FuncionesSQL.insertarFechaEventos(hotelFechaEventos.Text,
      DateTimePickerDesdeFechaEventos.Value, DateTimePickerHastaFechaEventos.Value,
      descripcionFechaEventos.Text)
        AddFilaGridFechaEventos(hotelFechaEventos.Text, DateTimePickerDesdeFechaEventos.
      Value, DateTimePickerHastaFechaEventos.Value, descripcionFechaEventos.Text)
      End If
```

```
38      End If
39    Else
40      MessageBox.Show("No existe el hotel.", "Error", MessageBoxButtons.OK)
41    End If
42  End Sub
```

Código 20: Evento tsbGuardarFechaEventos.Click.

```
1  Private Sub dgvFechaEventos_SelectionChanged(sender As Object, e As EventArgs) Handles
       dgvFechaEventos.SelectionChanged
2    HabilitarEdicionFechaEventos(True)
3    RecuperarTextBoxFechaEventos()
4    HabilitarBotonesFechaEventos(False)
5  End Sub
```

Código 21: Evento dgvFechaEventos.SelectionChanged.

```
1  Private Sub hotelFechaEventos_KeyPress(sender As Object, e As KeyPressEventArgs) Handles
       hotelFechaEventos.KeyPress
2    If Char.IsDigit(e.KeyChar) Then
3      e.Handled = False
4    ElseIf Char.IsControl(e.KeyChar) Then
5      e.Handled = False
6    Else
7      e.Handled = True
8    End If
9    If (e.KeyChar = Convert.ToChar(Keys.Return)) Then
10     e.Handled = True
11     SendKeys.Send("{TAB}")
12   End If
13 End Sub
```

Código 22: Evento hotelFechaEventos.KeyPress.

Estructura Interna del Proyecto - Eventos

```
1  #Region "Global variables"
2      Private loading As Bitmap = My.Resources.loading 'EJECUCION DEL GIF LOADING'
3      Private initialTime As DateTime 'TIEMPO INICIAL AL PULSAR INICIAR'
4      Private initialTestTime As DateTime 'TIEMPO INICIAL DEL TEST'
5      Private dtCheckedTest As New DataTable 'GUARDARA LOS TEST CHEQUEADOS'
6      Public Shared cell As String = "" 'DETERMINARA LA CELDA CORRESPONDIENTE'
7
8      Private TestTimeResplaht As New List(Of Double) 'TIEMPO DE MEDIO RESPAHT'
9      Private TestTimeResatari As New List(Of Double) 'TIEMPO DE MEDIO RESATARI'
10     Private TestTimeRezacunw As New List(Of Double) 'TIEMPO DE MEDIO REZACUNW'
11     Private TestTimeConpreci As New List(Of Double) 'TIEMPO DE MEDIO CONPRECI'
12     Private TestTimeMirai As New List(Of Double) 'TIEMPO DE MEDIO MIRAI'
13     Private TestTimeBooking As New List(Of Double)'TIEMPO DE MEDIO BOOKING'
14     Private TestTimeOTS As New List(Of Double) 'TIEMPO DE MEDIO OTS'
15     Private TestTimeExpedia As New List(Of Double) 'TIEMPO DE MEDIO EXPEDIA'
16 #End Region
```

Código 23: Variables globales de la aplicación.

```
Private Sub platformTesting_Load(sender As Object, e As EventArgs) Handles MyBase.Load
  loadForm()
End Sub
```

Código 24: Evento platformTesting_Load.

```
Private Sub platformTesting_FormClosing(sender As Object, e As FormClosingEventArgs)
    Handles MyBase.FormClosing
  stopTesting()
End Sub
```

Código 25: Evento platformTesting_FormClosing.

```
Private Sub btnStartTesting_Click(sender As Object, e As EventArgs) Handles btnStartTesting.
    Click
  startTesting()
End Sub
```

Código 26: Evento btnStartTesting_Click.

```
Private Sub btnStopTesting_Click(sender As Object, e As EventArgs) Handles btnStopTesting.
    Click
  stopTesting()
End Sub
```

Código 27: Evento btnStopTesting_Click.

```
Private Sub btnLoopTesting_Click(sender As Object, e As EventArgs) Handles btnLoopTesting.
    Click
  loopTesting()
End Sub
```

Código 28: Evento btnLoopTesting_Click.

```
Private Sub btnForceInsert_Click(sender As Object, e As EventArgs) Handles btnForceInsert.
    Click
  forceInsert()
End Sub
```

Código 29: Evento btnForceInsert_Click.

```
Private Sub timer_Tick(sender As Object, e As EventArgs) Handles timer.Tick
  actionTimer()
End Sub
```

Código 30: Evento timer_Tick.

```
Private Sub backgroundWorker_RunWorkerCompleted(sender As Object, e As System.
    ComponentModel.RunWorkerCompletedEventArgs) Handles backgroundWorker.
    RunWorkerCompleted
  If e.Error IsNot Nothing Then
    Utilities.existsError = True
  Else
    backgroundWorkerCompleted()
  End If
End Sub
```

Código 31: Evento backgroundWorker_RunWorkerCompleted.

```
Private Sub backgroundWorker_ProgressChanged(sender As Object, e As System.
    ComponentModel.ProgressChangedEventArgs) Handles backgroundWorker.ProgressChanged
  Dim row As Integer = CInt(e.UserState.ToString.Split(New Char() {" "c})(0))
  Dim column As String = e.UserState.ToString.Split(New Char() {" "c})(1)
  Dim report As String = e.UserState.ToString.Split(New Char() {" "c})(2)
  updateCell(row, column, getReportCell(report))
End Sub
```

Código 32: Evento backgroundWorker_ProgressChanged.

```
Private Sub backgroundWorker_DoWork(sender As Object, e As System.ComponentModel.
    DoWorkEventArgs) Handles backgroundWorker.DoWork
  backgroundWorkerAction()
End Sub
```

Código 33: Evento backgroundWorker_DoWork.

Estructura Interna del Proyecto - Métodos

```
Private Sub loadForm()
  Me.Text = Me.Text & " - v" & My.Application.Info.Version.ToString
  PushComun.Varios.SetAplication(PushComun.Varios.TipoAplicacion.
      TESTEO_PLATAFORMAS)

  Comun.Logs.CrearRutaLogs()

  dtCheckedTest.Columns.Add(New DataColumn("test"))
  dtCheckedTest.Columns.Add(New DataColumn("hotel"))
  dtCheckedTest.Columns.Add(New DataColumn("eventName"))

  loadGrid(True)

  If dgvPlatformTesting.RowCount > 1 Then
    btnStartTesting.Enabled = True
  Else
    btnStartTesting.Enabled = False
  End If
  btnStopTesting.Enabled = False
End Sub
```

Código 34: Método loadForm().

```
Private Sub startTesting()
  If Not backgroundWorker.IsBusy Then
    loadGrid(False)
    clearTimes()
    initialTime = DateTime.Now

    If Not backgroundWorker.IsBusy Then
      If Not My.Computer.FileSystem.FileExists(My.Application.Info.DirectoryPath & "\Logs")
      Then
        My.Computer.FileSystem.CreateDirectory(My.Application.Info.DirectoryPath & "\Logs")
      End If
      backgroundWorker.RunWorkerAsync()
      btnStartTesting.Enabled = False
      btnStopTesting.Enabled = True
    End If
  End If
End Sub
```

Código 35: Método startTesting().

```
Private Sub stopTesting()
  backgroundWorker.CancelAsync()
End Sub
```

Código 36: Método stopTesting().

```
Private Sub forceInsert()
  Dim forceInsert As New ForceInsert
  forceInsert.Show()
End Sub
```

Código 37: Método forceInsert().

```
Private Sub loadGrid(ByVal firstTime As Boolean)
  If Not firstTime Then
    dtCheckedTest.Rows.Clear()
    For Each row As DataGridViewRow In dgvPlatformTesting.Rows
      dtCheckedTest.Rows.Add(row.Cells("test").Value, row.Cells("hotel").Value, row.Cells("
      eventName").Value)
    Next
  End If

  dgvPlatformTesting.Rows.Clear()
  dgvPlatformTesting.BackgroundColor = System.Drawing.SystemColors.Control

  Dim dt As DataTable = FuncionesSQL.GetFechaEventosCompleto()
  For Each column As DataGridViewColumn In dgvPlatformTesting.Columns()
    column.SortMode = DataGridViewColumnSortMode.NotSortable
    column.Resizable = DataGridViewTriState.False
  Next

  If Not dt Is Nothing Then
    For Each row As DataRow In dt.Rows
```

```
20        If DateTime.Compare(row.Item("fecha_inicio"), DateTime.Now) = 1 Then
21          dgvPlatformTesting.Rows.Add(imageList.Images(3), isCheckedToTest(firstTime, row),Trim
          (row.Item("hotel").ToString), getHotelName(Trim(row.Item("hotel").ToString)), row.Item("
          fecha_inicio").ToString.Split(New Char() {" "c})(0), row.Item("fecha_fin").ToString.Split(
          New Char() {" "c})(0), Trim(row.Item("descripcion").ToString))
22        End If
23      Next
24    End If
25  End Sub
```

Código 38: Método loadGrid().

```
1   Private Sub clearTimes()
2     TestTimeResplaht.Clear()
3     TestTimeResatari.Clear()
4     TestTimeRezacunw.Clear()
5     TestTimeConpreci.Clear()
6     TestTimeMirai.Clear()
7     TestTimeBooking.Clear()
8     TestTimeOTS.Clear()
9     TestTimeExpedia.Clear()
10
11    lbTimeTestResplaht.Text = ""
12    lbTimeTestResatari.Text = ""
13    lbTimeTestRezacunw.Text = ""
14    lbTimeTestConpreci.Text = ""
15    lbTimeTestMirai.Text = ""
16    lbTimeTestBooking.Text = ""
17    lbTimeTestOTS.Text = ""
18    lbTimeTestExpedia.Text = ""
19  End Sub
```

Código 39: Método clearTimes().

```
1   Private Sub actionTimer()
2     If Not backgroundWorker.IsBusy Then
3       startTesting()
4     End If
5   End Sub
```

Código 40: Método actionTimer().

```
1   Private Sub loopTesting()
2     If timer.Enabled = True Then
3       btnLoopTesting.Text = "Start Loop"
4       timer.Enabled = False
5       btnLoopTesting.Image = imageList.Images(8)
6       timer.Stop()
7     Else
8       btnLoopTesting.Image = imageList.Images(4)
9       btnLoopTesting.Text = "Stop Loop"
10      timer.Enabled = True
11      timer.Start()
12    End If
13  End Sub
```

Código 41: Método loopTesting().

```
Private Sub backgroundWorkerAction()
  If RadCheckedDropDownList.CheckedItems.Count <> 0 Then 'Si hay elementos checkeados..'
    Dim index As Integer = 0

    ImageAnimator.Animate(loading, New EventHandler(AddressOf Me.onFrameChanged))
    For Each row As DataGridViewRow In dgvPlatformTesting.Rows
      If backgroundWorker.CancellationPending Then Exit For
      If row.Cells("test").Value Then 'Si estÁ marcada para realizar el test entraremos'
        cell = index & " state"

        backgroundWorker.ReportProgress(0, cell & " LOADING") 'Poner estado cargando'

        'TESTEO LOCAL'
        If RadCheckedDropDownList.Items(0).Checked Then platformTesting(index, Utilities.Platform.RESPLAHT)
        If RadCheckedDropDownList.Items(1).Checked Then platformTesting(index, Utilities.Platform.RESATARI)
        If RadCheckedDropDownList.Items(2).Checked Then platformTesting(index, Utilities.Platform.REZACUNW)
        If RadCheckedDropDownList.Items(3).Checked Then platformTesting(index, Utilities.Platform.CONPRECI)

        'TESTEO WEB'
        If RadCheckedDropDownList.Items(4).Checked Then platformTesting(index, Utilities.Platform.MIRAI)
        If RadCheckedDropDownList.Items(5).Checked Then platformTesting(index, Utilities.Platform.BOOKING)
        If RadCheckedDropDownList.Items(6).Checked Then platformTesting(index, Utilities.Platform.OTS)
        If RadCheckedDropDownList.Items(7).Checked Then platformTesting(index, Utilities.Platform.EXPEDIA)

        If Utilities.existsError Then
          backgroundWorker.ReportProgress(0, cell & " ERROR") 'Poner estado error'
        ElseIf Utilities.existsWarning Then
          backgroundWorker.ReportProgress(0, cell & " WARNING") 'Poner estado aviso'
        ElseIf Not backgroundWorker.CancellationPending Then
          backgroundWorker.ReportProgress(0, cell & " CORRECT") 'Poner estado correcto'
        End If

        Utilities.existsError = False
        Utilities.existsWarning = False
      End If
      index += 1
    Next
    ImageAnimator.StopAnimate(loading, New EventHandler(AddressOf Me.onFrameChanged))
  End If
End Sub
```

Código 42: Método backgroundWorkerAction().

```
Private Sub backgroundWorkerCompleted()
  btnStartTesting.Enabled = True
  btnStopTesting.Enabled = False
  writeExecutionTime()
End Sub
```

Código 43: Método backgroundWorkerCompleted().

```
Private Sub platformTesting(ByVal index As Integer, ByVal platform As Utilities.Platform)
  If Not backgroundWorker.CancellationPending Then 'Si tenemos una cancelacion pendiente no
      haremos nada'
  If dgvPlatformTesting.RowCount.Equals(index) Then Exit Sub ' Si el grid no contiene ningun
      registro no haremos nada'
  initialTestTime = DateTime.Now

  Dim hotel As String = dgvPlatformTesting.Rows(index).Cells("hotel").Value
  Dim startEvent As String = dgvPlatformTesting.Rows(index).Cells("startEvent").Value.
      ToString
  Dim endEvent As String = dgvPlatformTesting.Rows(index).Cells("endEvent").Value.ToString
  Dim cell As String = index & " " & platform.ToString

  Utilities.orderToWriteCell = Utilities.ReportCell.CORRECT ' Suponemos en principio que
      todo esta correcto'
  backgroundWorker.ReportProgress(0, cell & " LOADING") 'Escribimos "Cargando.." en la
      celda ya que se cumple esta acciÓn si o si'

  Select Case platform
    Case Utilities.Platform.RESPLAHT, Utilities.Platform.RESATARI, Utilities.Platform.
      REZACUNW, Utilities.Platform.CONPRECI
      LocalTest.localTest(platform, hotel, startEvent, endEvent, backgroundWorker)
    Case Utilities.Platform.MIRAI, Utilities.Platform.BOOKING, Utilities.Platform.OTS,
      Utilities.Platform.EXPEDIA
      WebTest.webTest(platform, hotel, startEvent, endEvent, backgroundWorker)
  End Select

  If Not backgroundWorker.CancellationPending OrElse (backgroundWorker.CancellationPending
       And Not Utilities.orderToWriteCell.ToString.Equals("CORRECT")) Then
    backgroundWorker.ReportProgress(0, cell & " " & Utilities.orderToWriteCell.ToString) '
      Escribimos en la celda'
  End If
  If Not backgroundWorker.CancellationPending Then addTestTime(platform) 'Si hemos
      cancelado la operacion el ultimo registro no estara completo'
  End If
End Sub
```

Código 44: Método platformTesting().

```
Private Sub addTestTime(ByVal platform As Utilities.Platform)
  Select Case platform
    Case Utilities.Platform.RESPLAHT
      TestTimeResplaht.Add(DateTime.Now.Subtract(initialTestTime).TotalSeconds)
    Case Utilities.Platform.RESATARI
      TestTimeResatari.Add(DateTime.Now.Subtract(initialTestTime).TotalSeconds)
    Case Utilities.Platform.REZACUNW
      TestTimeRezacunw.Add(DateTime.Now.Subtract(initialTestTime).TotalSeconds)
    Case Utilities.Platform.CONPRECI
      TestTimeConpreci.Add(DateTime.Now.Subtract(initialTestTime).TotalSeconds)
    Case Utilities.Platform.MIRAI
      TestTimeMirai.Add(DateTime.Now.Subtract(initialTestTime).TotalSeconds)
    Case Utilities.Platform.BOOKING
      TestTimeBooking.Add(DateTime.Now.Subtract(initialTestTime).TotalSeconds)
    Case Utilities.Platform.OTS
      TestTimeOTS.Add(DateTime.Now.Subtract(initialTestTime).TotalSeconds)
    Case Utilities.Platform.EXPEDIA
      TestTimeExpedia.Add(DateTime.Now.Subtract(initialTestTime).TotalSeconds)
  End Select
End Sub
```

Código 45: Método addTestTime().

```
Private Sub updateCell(ByVal row As Integer, ByVal column As String, ByVal report As
    Utilities.ReportCell)
  If column = "state" Then
    Select Case report
      Case Utilities.ReportCell.FAILURE
        dgvPlatformTesting.Rows(row).Cells(column).Value = imageList.Images(0) 'Error..'
      Case Utilities.ReportCell.CORRECT
        dgvPlatformTesting.Rows(row).Cells(column).Value = imageList.Images(1) 'Correcto..'
      Case Utilities.ReportCell.LOADING
        dgvPlatformTesting.Rows(row).Cells(column).Value = loading 'Cargando..'
      Case Utilities.ReportCell.WARNING
        dgvPlatformTesting.Rows(row).Cells(column).Value = imageList.Images(7) 'Warning..'
    End Select
  Else
    Select Case report
      Case Utilities.ReportCell.FAILURE
        dgvPlatformTesting.Rows(row).Cells(column).Value = "Error"
      Case Utilities.ReportCell.CORRECT
        dgvPlatformTesting.Rows(row).Cells(column).Value = "Correcto"
      Case Utilities.ReportCell.LOADING
        dgvPlatformTesting.Rows(row).Cells(column).Value = "Cargando..."
      Case Utilities.ReportCell.WARNING
        dgvPlatformTesting.Rows(row).Cells(column).Value = "Warning"
      Case Utilities.ReportCell.NOEXISTS
        dgvPlatformTesting.Rows(row).Cells(column).Value = "No existe"
      Case Utilities.ReportCell.NOCONTRACT
        dgvPlatformTesting.Rows(row).Cells(column).Value = "Sin contrato"
    End Select
  End If
End Sub
```

Código 46: Método updateCell().

```
Private Sub writeExecutionTime()
  Dim tiempoEjecucion As TimeSpan = DateTime.Now.Subtract(initialTime)

  lbExecutionTime.Text = "Tiempo de ejecuciÓn: " & tiempoEjecucion.Days & " d " &
      tiempoEjecucion.Hours & " h " & tiempoEjecucion.Minutes & " min " & tiempoEjecucion.
      Seconds & " sec"

  If TestTimeResplaht.Count <> 0 Then lbTimeTestResplaht.Text = Math.Round(
      TestTimeResplaht.Average) & " sec"
  If TestTimeResatari.Count <> 0 Then lbTimeTestResatari.Text = Math.Round(
      TestTimeResatari.Average) & " sec"
  If TestTimeRezacunw.Count <> 0 Then lbTimeTestRezacunw.Text = Math.Round(
      TestTimeRezacunw.Average) & " sec"
  If TestTimeConpreci.Count <> 0 Then lbTimeTestConpreci.Text = Math.Round(
      TestTimeConpreci.Average) & " sec"
  If TestTimeMirai.Count <> 0 Then lbTimeTestMirai.Text = Math.Round(TestTimeMirai.
      Average) & " sec"
  If TestTimeBooking.Count <> 0 Then lbTimeTestBooking.Text = Math.Round(
      TestTimeBooking.Average) & " sec"
  If TestTimeOTS.Count <> 0 Then lbTimeTestOTS.Text = Math.Round(TestTimeOTS.
      Average) & " sec"
  If TestTimeExpedia.Count <> 0 Then lbTimeTestExpedia.Text = Math.Round(
      TestTimeExpedia.Average) & " sec"

End Sub
```

Código 47: Método writeExecutionTime().

```
Private Sub onFrameChanged(ByVal o As Object, ByVal e As EventArgs)
  If Not backgroundWorker.CancellationPending Then
    dgvPlatformTesting.InvalidateColumn(0)
  End If
End Sub
```

Código 48: Método onFrameChanged().

```
Private Sub dgvPlatformTesting_Paint(sender As Object, e As PaintEventArgs) Handles
    dgvPlatformTesting.Paint
  ImageAnimator.UpdateFrames()
End Sub
```

Código 49: Método dgvPlatformTesting_Paint().

```
Private Function getHotelName(ByVal hotel As String) As String
  Dim hotelName As String = ""
  hotelName = FuncionesSQL.GetHotelName(hotel)
  hotelName = hotelName.ToUpper

  hotelName = hotelName.Replace("HOTEL", "")
  hotelName = hotelName.Replace("SUITES", "")
  hotelName = hotelName.Replace("APTOS.", "")
  hotelName = hotelName.Replace("AQUAPARK", "")
  hotelName = hotelName.Replace("DIVER", "")
```

```
    hotelName = hotelName.Replace("AGUADULCE", "")
    hotelName = hotelName.Replace("ROQUETAS", "")
    hotelName = hotelName.Replace("SENATOR", "")
    hotelName = hotelName.Replace("SPA", "")
    hotelName = hotelName.Replace("&", "")
    hotelName = hotelName.Replace("*", "")

    hotelName = Trim(hotelName)
    Return hotelName
End Function
```

Código 50: Método getHotelName().

```
Private Function getReportCell(ByVal report As String) As Utilities.ReportCell
    Dim reportCell As Utilities.ReportCell
    Select Case report
        Case "ERROR"
            reportCell = Utilities.ReportCell.FAILURE
        Case "CORRECT"
            reportCell = Utilities.ReportCell.CORRECT
        Case "LOADING"
            reportCell = Utilities.ReportCell.LOADING
        Case "WARNING"
            reportCell = Utilities.ReportCell.WARNING
        Case "NOEXISTS"
            reportCell = Utilities.ReportCell.NOEXISTS
        Case "NOCONTRACT"
            reportCell = Utilities.ReportCell.NOCONTRACT
    End Select
    Return reportCell
End Function
```

Código 51: Método getReportCell().

```
Private Function isCheckedToTest(ByVal firstTime As Boolean, ByVal row As DataRow) As
    Boolean
    Dim checkedToTest As Boolean = firstTime
    If Not firstTime Then
        For Each rowChecked As DataRow In dtCheckedTest.Rows
            If rowChecked.Item("hotel").ToString.Equals(row.Item("hotel").ToString) And _
                rowChecked.Item("eventName").ToString.Equals(row.Item("descripcion").ToString) Then
                checkedToTest = rowChecked.Item("test")
                Exit For
            End If
        Next
    End If
    Return checkedToTest
End Function
```

Código 52: Método isCheckedToTest().

```
Public Shared Sub localTest(ByVal table As Platform, ByVal hotel As Integer, ByVal startEvent
        As String, ByVal endEvent As String, ByVal backgroundWorker As BackgroundWorker)
  Dim dtAS400 As DataTable = Nothing
  Dim dtSQL As DataTable = Nothing
  If Not table.Equals(Platform.CONPRECI) Then
    dtAS400 = getUpdatedRowsAS400(table, hotel, startEvent, endEvent)
    dtSQL = getUpdatedRowsSQL(table, hotel, startEvent, endEvent)

    dtAS400.PrimaryKey = New DataColumn() {dtAS400.Columns.Item("RRNMOD")}
    dtSQL.PrimaryKey = New DataColumn() {dtSQL.Columns.Item("ID")}
  End If

  If table.Equals(Platform.CONPRECI) OrElse _
   haveDataThisDatatable(dtAS400, True, table, Server.AS400) And _
   haveDataThisDatatable(dtSQL, True, table, Server.SQL) Then

    testAS400(table, dtAS400, dtSQL, hotel, startEvent, endEvent, backgroundWorker)
    testSQL(table, dtAS400, dtSQL, backgroundWorker)
  End If
End Sub
```

Código 53: Método LocalTest().

```
Public Shared Sub doLocalOperation(ByVal platform As Platform, ByVal identifier As String,
      ByVal localOperationType As LocalOperationType, ByVal backgroundWorker As
      BackgroundWorker)
  If Not backgroundWorker.CancellationPending Then
    Dim fechaMod As String = FuncionesAS400.getLastFechamod(platform.ToString, identifier,
      localOperationType.ToString) ' Obtenemos la ultima fecha de modificado'

    If fechaMod.Equals("") Then 'Si fechamod es nula significa que no existe en replisql4, por lo
      tanto se inserta con los datos del replisql3'
      If insertRegistry(platform, identifier) Then ' Intentamos insertar el registro inexistente en el
      replisql4'
        WriteLog(ReportLog.WarningInsert, platform, Server.SQL, identifier) ' El registro ha
      tenido que ser insertado porque no existia en el replisql4'
      Else ' En caso de fallo en la inserciÓn informamos en el log'
        WriteLog(ReportLog.ErrorInsert, platform, Server.SQL, identifier) ' El registro no ha
      podido insertarse indicando un error de bd'
      End If
    Else ' Si la fechamod no es nula, se quita la marca de procesado para que el servidor en
      continua ejecucion lo vuelva a procesar y actualice todos los cambios'
      If FuncionesAS400.desmarcarRegistroProcesado(platform.ToString, identifier, fechaMod)
      Then ' Desmarcamos el registro'
        WriteLog(ReportLog.WarningUpdate, platform, Server.AS400, identifier) ' Indicamos que
      el registro ha sido marcado como no procesado'
      Else
        WriteLog(ReportLog.ErrorUpdate, platform, Server.AS400, identifier) ' El registro no ha
      podido desmarcarse indicando un error de bd'
      End If
    End If
  End If
End Sub
```

Código 54: Método doLocalOperation().

```
Public Shared Sub webTest(ByVal platform As Platform, ByVal SQLHotelID As Integer, ByVal
    startEvent As String, ByVal endEvent As String, ByVal backgroundWorker As
    BackgroundWorker)
  Dim desdeDate = DateTime.ParseExact(startEvent, "dd/MM/yyyy", Nothing)
  Dim hastaDate = DateTime.ParseExact(endEvent, "dd/MM/yyyy", Nothing)
  Select Case platform
    Case platform.MIRAI 'Mirai necesita dos llamadas si los meses son distintos...'
      testMirai(SQLHotelID, startEvent, startEvent, endEvent, backgroundWorker)
      If desdeDate.Year <> hastaDate.Year Or desdeDate.Month <> hastaDate.Month Then
        testMirai(SQLHotelID, endEvent, startEvent, endEvent, backgroundWorker)
      End If
    Case platform.BOOKING
      testBooking(SQLHotelID, startEvent, endEvent, backgroundWorker)
    Case platform.OTS
      testOts(SQLHotelID, startEvent, endEvent, backgroundWorker)
    Case platform.EXPEDIA
      testExpedia(SQLHotelID, startEvent, endEvent, backgroundWorker)
    End Select
End Sub
```

Código 55: Método WebTest().

```
Public Shared Sub doWebOperation(ByVal platform As Platform, ByVal username As String,
    ByVal password As String, ByVal operationType As OperationType, ByVal requestDate As
    String, ByVal SQLHotelID As String, ByVal PlatformHotelID As String, ByVal
    SQLRoomID As String, ByVal PlatformRoomID As String, ByVal platformValue As String,
    ByVal sqlValue As String, ByVal quantity As String, ByVal minimumStay As String, ByVal
    backgroundWorker As BackgroundWorker, Optional ByVal datatable As DataTable =
    Nothing, Optional ByVal PlatformRateID As String = "", Optional ByVal dataSet As
    DataSet = Nothing, Optional ByVal isPlaya As Boolean = False, Optional ByVal
    Occupancy As Integer = 0, Optional ByVal closed As Boolean = False)
  If Not backgroundWorker.CancellationPending Then
    Select Case operationType
      Case operationType.DisparedPrice
        If updatePrice(platform, username, password, requestDate, PlatformHotelID, SQLHotelID,
         PlatformRoomID, quantity, minimumStay, sqlValue, backgroundWorker, PlatformRateID,
        dataSet, isPlaya, Occupancy, closed, "", SQLRoomID) Then
          WriteLog(ReportLog.WarningUpdatedPrice, platform, Server.SQL, PlatformRoomID,
        requestDate, SQLHotelID, Trim(SQLRoomID), platformValue, sqlValue, PlatformRateID)
        Else
          WriteLog(ReportLog.WarningNotUpdatedPrice, platform, Server.SQL, PlatformRoomID,
         requestDate, SQLHotelID, Trim(SQLRoomID), platformValue, sqlValue, PlatformRateID)
        End If
      Case operationType.DisparedAvailability
        If updateAvailability(platform, username, password, requestDate, PlatformHotelID,
        PlatformRoomID, quantity, minimumStay, sqlValue, backgroundWorker, datatable) Then
          WriteLog(ReportLog.WarningUpdatedAvailability, platform, Server.SQL,
        PlatformRoomID, requestDate, SQLHotelID, Trim(SQLRoomID), platformValue, sqlValue)
        Else
          WriteLog(ReportLog.WarningNotUpdatedAvailability, platform, Server.SQL,
        PlatformRoomID, requestDate, SQLHotelID, Trim(SQLRoomID), platformValue, sqlValue)
        End If
    End Select
  End If
End Sub
```

Código 56: Método doWebOperation().

Forzar inserción

```
Public Class ForceInsert
  Private Sub btnForceInsert_Click(sender As Object, e As EventArgs) Handles btnForceInsert.
      Click
    forceInsert()
  End Sub

  Private Sub btnCancel_Click(sender As Object, e As EventArgs) Handles btnCancel.Click
    cancel()
  End Sub

  Private Sub tbIdentifier_KeyPress(sender As Object, e As KeyPressEventArgs) Handles
      tbIdentifier.KeyPress
    If Asc(e.KeyChar) >= 32 And Asc(e.KeyChar) <= 47 Or Asc(e.KeyChar) >= 58 Then
      e.Handled = True
    End If
  End Sub

  Private Sub forceInsert()
    If Not isEmptySelection() Then
      Dim BD As String = cbTables.Text
      Dim RRNMOD As String = tbIdentifier.Text

      'COPIA DE VIAJES A REPLISQL4'
      If FuncionesAS400.forceInsertRegistryInReplisql4(BD, RRNMOD) Then
        MsgBox("El registro se ha insertado correctamente")
      Else
        MsgBox("Ha ocurrido un problema al intentar insertar el registro.")
      End If
    End If
  End Sub

  Private Sub cancel()
    Me.Close()
  End Sub

  Private Function isEmptySelection() As Boolean
    Dim bSuccess As Boolean = False
    If (cbTables.Text Is Nothing Or cbTables.Text.Equals("")) And (tbIdentifier.Text Is Nothing
      Or tbIdentifier.Text.Equals("")) Then
      bSuccess = True
      MsgBox("Introduzca los campos.")
    ElseIf cbTables.Text Is Nothing Or cbTables.Text.Equals("") Then
      bSuccess = True
      MsgBox("Hay que seleccionar una tabla.")
    ElseIf tbIdentifier.Text Is Nothing Or tbIdentifier.Text.Equals("") Then
      bSuccess = True
      MsgBox("Hay que introducir un identificador.")
    End If
    Return bSuccess
  End Function
End Class
```

Código 57: Clase “ForceInsert”.

Clase Utilidades - Enumeradores y variables

```
Public Enum Platform
  RESPLAHT
  RESATARI
  REZACUNW
  CONPRECI
  CONPERIO

  MIRAI
  BOOKING
  BOOKING_C
  OTS
  OTS_C
  EXPEDIA
  EXPEDIA_C
End Enum
```

Código 58: Enumerador “Platform”.

```
Public Enum Server
  AS400
  SQL
End Enum
```

Código 59: Enumerador “Server”.

```
Public Enum OperationType
  DisparedPrice
  DisparedAvailability
End Enum
```

Código 60: Enumerador “OperationType”.

```
Public Enum LocalOperationType
  A
  M
  B
End Enum
```

Código 61: Enumerador “LocalOperationType”.

```
Public Enum ReportCell
  FAILURE
  CORRECT
  LOADING
  WARNING
  NOEXISTS
  NOCONTRACT
End Enum
```

Código 62: Enumerador “ReportCell”.

```
Public Enum ReportLog
  ErrorDatatable
  WarningDatatable

  ErrorInsert
  ErrorUpdate
  ErrorRequest

  WarningInsert
  WarningUpdate
  WarningUpdatedPrice
  WarningNotUpdatedPrice
  WarningUpdatedAvailability
  WarningNotUpdatedAvailability
  WarningRoomNotMaped
End Enum
```

Código 63: Enumerador “ReportLog”.

```
' Expedia (5518623), Booking_C (413414), Booking (413415), Mirai (2450517), OTS (5511600) '
Public Shared ReadOnly idUsers() As Integer = {5518623, 413414, 413415, 2450517, 5511600}

' idAgencias: Expedia, Booking Costas, Booking, Mirai '
Public Shared ReadOnly idAgencies() As Integer = {38783, 1090, 1090, 1143}

' Se usaran para evitar referencias repetidas bajo el mismo identificador '
Public Shared listIdentifier As New ArrayList
Public Shared listCodeCupoRoom As New ArrayList
Public Shared listCodePriceRoom As New ArrayList

' Se usara para controlar si se ha detectado un error '
Public Shared existsError As Boolean = False

' Se usara para controlar si se ha detectado un warning '
Public Shared existsWarning As Boolean = False

' Se usara para ordenar la escritura de la celda suponiendo que esta todo correcto '
Public Shared orderToWriteCell As ReportCell = ReportCell.CORRECT
```

Código 64: Variables globales de la clase Utilities.

Clase Utilidades - Métodos

```
1  Public Shared Sub WriteLog(ByVal report As ReportLog, Optional ByVal platform As Platform
       = Nothing, Optional ByVal server As Server = Nothing, Optional ByVal identifier As String
       = "", Optional ByVal requestDate As String = "", Optional ByVal hotel As String = "",
       Optional ByVal room As String = "", Optional ByVal infoWeb As String = "", Optional
       ByVal infoSQL As String = "", Optional ByVal PlatformRateID As String = "")
2    Dim common As String = ""
3    If Not platform.ToString Is Nothing Then common = platform.ToString & " - " & server.
       ToString & ": "
4
5    Select Case report
6      Case ReportLog.ErrorDatatable
7        Comun.Logs.LogError(common & "Error al cargar un datatable.")
8      Case ReportLog.WarningDatatable
9        Comun.Logs.LogWarning(common & "El datatable cargado no tiene datos.")
10
11     'REPORTES LOCALES'
12     Case ReportLog.ErrorInsert
13       Comun.Logs.LogError(common & "Error al insertar el registro con identificador " &
         identifier & ".")
14     Case ReportLog.ErrorUpdate
15       Comun.Logs.LogError(common & "Error al desmarcar el registro con identificador " &
         identifier & ".")
16     Case ReportLog.WarningInsert
17       Comun.Logs.LogWarning(common & "El registro con identificador " & identifier & " ha sido
          insertado.")
18     Case ReportLog.WarningUpdate
19       Comun.Logs.LogWarning(common & "El registro con identificador " & identifier & " ha sido
          desmarcado.")
20
21     'REPORTES WEB'
22     Case ReportLog.ErrorRequest
23       Comun.Logs.LogError(platform.ToString & ": Ha ocurrido un error al recoger la peticiÓn
         para el hotel " & hotel & " con fecha " & requestDate & ".")
24     Case ReportLog.WarningUpdatedPrice
25       Comun.Logs.LogWarning(common & "Actualizado el registro con fecha " & requestDate & "
         , room (" & room & ", " & identifier & "), rateID " & PlatformRateID & " y hotel " &
         hotel & " tenÍa precio " & infoWeb & " y en SQL " & infoSQL & ". Ahora tiene precio " &
         infoSQL & " en " & platform.ToString & ".")
26     Case ReportLog.WarningNotUpdatedPrice
27       Comun.Logs.LogWarning(common & "No actualizado el registro con fecha " & requestDate
         & ", room (" & room & ", " & identifier & "), rateID " & PlatformRateID & " y hotel " &
         hotel & " tenÍa precio " & infoWeb & " y en SQL " & infoSQL & ".")
28     Case ReportLog.WarningUpdatedAvailability
29       Comun.Logs.LogWarning(common & "Actualizado el registro con fecha " & requestDate & "
         , room (" & room & ", " & identifier & ") y hotel " & hotel & " tenÍa cupo " & infoWeb &
         " y en SQL " & infoSQL & ". Ahora tiene cupo " & infoSQL & " en " & platform.ToString
         & ".")
30     Case ReportLog.WarningNotUpdatedAvailability
31       Comun.Logs.LogWarning(common & "No actualizado el registro con fecha " & requestDate
         & ", room (" & room & ", " & identifier & ") y hotel " & hotel & " tenÍa cupo " & infoWeb
          & " y en SQL " & infoSQL & ".")
32     Case ReportLog.WarningRoomNotMaped
```

```
33          Comun.Logs.LogWarning(common & "La habitaciÓn con ID " & identifier & " para el hotel
             " & hotel & " no estÁ mapeada.")
34          orderToWriteCell = ReportCell.NOEXISTS
35        End Select
36
37        If report.ToString.StartsWith("Error") Then
38          existsError = True
39          orderToWriteCell = ReportCell.FAILURE ' Escribimos Error en la celda'
40        ElseIf report.ToString.StartsWith("Warning") Then
41          existsWarning = True
42          orderToWriteCell = ReportCell.WARNING 'Escribimos Revisar LOG en la celda'
43        End If
44    End Sub
```

Código 65: Método "WriteLog".

```
1   Public Shared Function haveDataThisDatatable(ByVal datatable As DataTable, Optional ByVal
        report As Boolean = False, Optional ByVal platform As Platform = Nothing, Optional
        ByVal server As Server = Nothing) As Boolean
2     Dim haveData As Boolean = True
3
4     If datatable Is Nothing Then ' Si las tablas no han recibido ningun cambio significa que no se
        ha cargado bien por error de bd'
5       If report Then WriteLog(ReportLog.ErrorDatatable, server, platform) ' Informar de error en
        el log al cargar las tablas'
6         haveData = False
7       ElseIf datatable.Rows.Count = 0 Then 'Si las tablas han recibido las columnas pero no tienen
         registros...'
8         If report Then WriteLog(ReportLog.WarningDatatable, server, platform) ' Informar de que
        las condiciones no han devuelto nada'
9           haveData = False
10        End If
11    Return haveData
12  End Function
```

Código 66: Método "haveDataThisDatatable".

```
1   Public Shared Function DateRange(Start As DateTime, Thru As DateTime) As IEnumerable(Of
        Date)
2     Return Enumerable.Range(0, (Thru.Date - Start.Date).Days + 1).Select(Function(i) Start.
        AddDays(i))
3   End Function
```

Código 67: Método "DateRange".

Referencias

[1] Senator Hotels & Resorts, "Descubra nuestros hoteles."
Recuperado el 30 de mayo de 2019 de https://www.playasenator.com, 2019.

[2] Booking, "Encuentra ofertas para todas las temporadas."
Recuperado el 23 de junio de 2019 de https://www.booking.com, (s.f).

[3] Expedia, "Haz tu reserva con expedia."
Recuperado el 25 de junio de 2019 de https://www.expedia.es, (s.f).

[4] Mirai, "We believe in your hotel."
Recuperado el 3 de julio de 2019 de https://mirai.com, (s.f).

[5] OTS, "Discover axis data."
Recuperado el 1 de julio de 2019 de https://axisdata.net, (s.f).

[6] Booking, "Demand api reference."
Recuperado el 29 de mayo de 2019 de https://developers.booking.com/api/index.html, (s.f).

[7] Expedia, "Developer apis."
Recuperado el 13 de junio de 2019 de https://expediaconnectivity.com/developer, (s.f).

[8] Mirai, "Api."
Recuperado el 17 de junio de 2019 de https://mirai.readthedocs.io/en/latest/api.html, (s.f).

[9] OTS, "Axis data solutions."
Recuperado el 17 de julio de 2019 de https://axisdata.net/section/solutions, (s.f).

[10] IBM, "Ibm i on power systems."
Recuperado el 26 de junio de 2019 de https://www.ibm.com/es-es/it-infrastructure/power/os/ibm-i, (s.f).

[11] Marce, "Db2/400 manual de consulta sql."
Recuperado el 17 de julio de 2019 de http://www.marce.com/docs/systems/manuales/db2_400_manual_consulta_sql_v3r7.pdf, 2018.

[12] J.Alvarez, "Diseño estructurado."
Recuperado el 15 de junio de 2019 de https://www.infor.uva.es/∼jvalvarez/docencia/ptema8is1.pdf, 2017.

[13] Jeff Sutherland, *Scrum: El revolucionario método para trabajar el doble en la mitad de tiempo.* Ariel, 2018.

[14] Platzi, "¿cómo funciona la metodología scrum?."
Recuperado el 29 de mayo de 2019 de https://platzi.com/blog/metodologia-scrum-fases, 2015.

[15] Wikipedia, "Ibm."
Recuperado el 7 de julio de 2019 de https://es.wikipedia.org/wiki/IBM, 2018.

[16] Tom Huntington, "¿está muerto el as/400?."
Recuperado el 25 de julio de 2019 de https://www.helpsystems.com/es/blog/esta-muerto-el-as400, 2018.

[17] Margaret Rouse, "Db2."
Recuperado el 23 de junio de 2019 de https://searchdatacenter.techtarget.com/es/definicion/DB2, 2015.

[18] Wikipedia, "Db2.."
Recuperado el 10 de junio de 2019 de https://es.wikipedia.org/wiki/DB2, 2017.

[19] IBM, "Características de db2."
Recuperado el 17 de julio de 2019 de https://www.ibm.com/support/knowledgecenter/es/SS3JRN_7.2.1/com.ibm.itcama.doc_7.2.1/db2/featuresdb2.html, 2017.

[20] F. García, "Entorno de desarrollo integrado (ide)."
Recuperado el 11 de julio de 2019 de https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide, 2013.

[21] LaPolitecnica.Net, "Soapui."
Recuperado el 11 de junio de 2019 de http://www.lapolitecnica.net/blog/soapui, 2014.

[22] María Estela Raffino, "¿qué es un lenguaje de programación?."
Recuperado el 10 de julio de 2019 de https://concepto.de/lenguaje-de-programacion, 2018.

[23] Evangelos Petroutsos, *Mastering Visual Basic.Net.* SYBEX INC, 2002.

[24] Alan Beaulieu, *Aprende SQL (2ª Edición).* ANAYA, 2009.

[25] Erik T. Ray, *Learning XML.* O'REILLY, 2003.

[26] Robert Lowe, "¿qué es una api y para qué sirve?."
Recuperado el 5 de julio de 2019 de https://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve, 2014.

[27] Gerardo Silva, "¿qué es la programación modular?."
Recuperado el 23 de junio de 2019 de https://sites.google.com/site/microcontroladoresmicrochip/programacion/-que-es-la-programacion-modular, (s.f).

[28] Jean Françoise Pillou, "Ciclo de vida del "software"."
Recuperado el 14 de junio de 2019 de https://es.ccm.net/contents/223-ciclo-de-vida-del-software, 2017.

[29] 1&1 IONOS, "Ficheros log: Toda la información de registro en un archivo." Recuperado el 29 de junio de 2019 de https://www.ionos.mx/digitalguide/online-marketing/analisis-web/el-log-el-archivo-de-registro-de-procesos-informaticos, 2016.

[30] Microsoft, "Clickonce security and deployment." Recuperado el 29 de junio de 2019 de https://docs.microsoft.com/en-us/visualstudio/deployment/clickonce-security-and-deployment?view=vs-2019, 2016.

[31] Certia, "Microsoft team foundation server." Recuperado el 12 de junio de 2019 de https://www.certia.net/TFS, (s.f).

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

Este proyecto se va a desarrollar en el seno de una Beca Talento D-UAL en la empresa hotelera Senator Hotels & Resorts.

Estos hoteles reciben miles de reservas a través de diversas plataformas tales como Booking, Expedia, Mirai, OTS… Para llevar un control eficiente de las reservas, la empresa dispone de un servidor que las recoge desde las diferentes plataformas que les dan servicio para su posterior gestión centralizada. Debido a la importancia de las reservas y su eficiente gestión, si el servidor falla durante un tiempo (por ejemplo, 10 minutos), se pueden llegar a perder reservas, con el gran impacto económico que ello conlleva. Para evitar estas pérdidas, la empresa dispone de servidores en la Nube en los que se realizan réplicas. Aun así, puede haber pérdidas, ya que la aplicación que realiza dichas réplicas puede fallar también.

El objetivo de este proyecto consiste en comprobar que las plataformas web de venta como Booking, Expedia, Mirai y OTS, entre otras, tengan la información de los establecimientos de la empresa Senator Hotels & Resorts realmente actualizada.

Por otro lado, está aplicación también comprobará la información dentro del sistema local de la empresa.

La aplicación notificará las diferencias que encuentre dentro del sistema y Nube-Plataformas y las arreglará, insertando nuevos registros, eliminándolos o modificándolos según corresponda.

This project will be developed within a D-UAL Talent Grant in the hotel company Senator Hotels & Resorts.

These hotels receive thousands of reservations through various platforms such like Booking, Expedia, Mirai, OTS… To keep an efficient control of the reservations, the company has a server that picks them up from the different platforms that serve them for their subsequent centralized management. Due to the importance of reservations and efficient management, if the server fails for a while (for example, 10 minutes), reservations may be lost, with the great economic impact that entails. To avoid these losses, the company has servers in the Cloud where replicas are made. Yet thus, there may be losses, since the application that performs these replicas can fail too.

The objective of this project is to verify that the sales web platforms such as Booking, Expedia, Mirai and OTS, among others, have the information of the establishments of the company Senator Hotels & Really updated resorts.

On the other hand, this application will also check the information within the local system of the company.

The application will notify the differences it finds within the system and Cloud-Platforms and fix them, inserting new records, deleting them or modifying them as appropriate.

GRADO EN INGENIERÍA INFORMÁTICA, 2018/2019