

EasySQL

PROYECTO DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

CURSO 2017/18

ELABORADO POR: ALEJANDRO ANTONIO DEL RÍO MORENO.



CENEC
MALAGA

ÍNDICE DE CONTENIDO

1. DESCRIPCIÓN.....	3
1.1. INTRODUCCIÓN.....	3
1.2. MOTIVACIÓN.....	5
2. ANÁLISIS DEL MERCADO ACTUAL.....	6
2.1. ESTUDIO DE PRODUCTOS EXISTENTES.....	7
2.2. CONCLUSIONES DEL ESTUDIO.....	10
3. PROPUESTA INNOVADORA.....	11
4. ESTUDIO DE VIABILIDAD.....	13
4.1. RECURSOS Y TECNOLOGÍAS A EMPLEAR.....	13
4.2. EVALUACIÓN DE COSTES.....	18
4.3. PLANIFICACIÓN DEL TIEMPO.....	19
4.4. ALCANCE DEL PROYECTO.....	20
5. ARQUITECTURA DE LA APLICACIÓN.....	21
5.1. ANÁLISIS Y DIAGRAMAS UML.....	23
5.2. ANÁLISIS Y DIAGRAMAS E/R.....	33
6. FUTURAS MEJORAS Y CONCLUSIONES.....	34
7. PRUEBAS Y RESULTADOS.....	35
8. MANUAL USUARIO.....	40
8.1. VENTANA INICIO.....	40

8.2. VENTANA REGISTRO.....	41
8.3. VENTANA CONEXIONES (MODO INVITADO).....	42
8.4. VENTANA CONEXIONES (MODO USUARIO).....	43
8.5. VENTANA OPERACIONES.....	44
8.6. VENTANA CREATE DATABASE.....	45
8.7. VENTANA DROP (DATABASE / TABLE).....	45
8.8. VENTANA CREATE TABLE.....	46
8.9. VENTANA SHOW TABLES.....	47
8.10. VENTANA RESULTADO DATOS.....	48
8.11. VENTANA ALTER TABLE.....	49
8.12. VENTANA SELECT.....	50
8.13. VENTANA INSERT.....	51
8.14. VENTANA UPDATE.....	52
8.15. VENTANA DELETE.....	53
9. BIBLIOGRAFÍA.....	54
10. ÍNDICE DE ELEMENTOS.....	55
10.1. ÍNDICE DE IMÁGENES.....	55
10.2. ÍNDICE DE UML.....	56
ANEXO I.- MANUAL DEL CÓDIGO.....	57

1. DESCRIPCIÓN

En esta memoria pretende ser un sumario en el que plasmar y dejar reflejado el trabajo realizado a lo largo del desarrollo del proyecto de final de Formación Profesional de Grado Superior, así de sus tecnologías usadas.

1.1.INTRODUCCIÓN

EasySQL es una herramienta pensada para facilitar el aprendizaje del lenguaje SQL al usuario, por medio de una interfaz gráfica amigable y simple, el manejo de datos de los dos grandes sistemas gestores de bases de datos que existen en la actualidad.



Imagen 1. Logotipo de EasySQL.

Esto permite un acercamiento más acogedor a un usuario novato o inexperto, que pueda llegar a verse abrumado por la cantidad de herramientas y opciones que poseen las completas soluciones que ofrecen hoy en día los principales desarrolladores, pensadas para grandes empresas o complejos proyectos, con una curva de aprendizaje mucho más dura.

El mundo del aprendizaje siempre ha sido un mundo en constante evolución, con el tiempo van apareciendo nuevas metodologías y maneras de enseñanza para adaptarse a los nuevos tiempos. Sobre todo en la informática, campo en el que normalmente unos contenidos de hace 5 años quedarían obsoletos en la actualidad.

En la actualidad, estamos viviendo la transición en el ámbito de la educación dentro de los centros del tradicional formato de papel y bolígrafo, aquellos que nos han acompañado desde largo tiempo atrás, a métodos más interactivos y tecnológicos, como el caso de la implantación de tablets y ordenadores para los alumnos, que aunque muy poco a poco pero en mayor medida en campos como la informática estamos viendo que cada vez ocupan una mayor importancia dentro de las aulas. Por ello, es necesario que tanto docentes como alumnos cuenten siempre con las mejores herramientas para favorecer un rápido aprendizaje, más en los tiempos que corren, que cada minuto cuenta y hay tanta materia y conocimiento que cualquier mejora en la manera de transmitir información puede ser muy bien recibida.

Es por ello que gracias a nuestra formación, debemos crear herramientas que aprovechen la disponibilidad de recursos que existen y ayuden a allanar el terreno a aquellos alumnos que quedan por venir.

La herramienta, especialmente pensada para los estudiantes, permite al usuario con una forma visual y fácil ir realizando las operaciones básicas de creación y manejo de datos, además de poder ver al instante el resultado de éstas, así como la sentencia SQL equivalente.

Normalmente, un usuario debe tener cierta preparación e instrucción a la hora de poder empezar a utilizar sistemas gestores de bases de datos. Este programa intenta acortar el tiempo necesario de preparación previa de dicho usuario y animarlo a ver resultados rápidos sin tener que lidiar con la exigente sintaxis del lenguaje SQL, permite un acercamiento básico y seguro.

1.2.MOTIVACIÓN

Dado que la informática es una rama de la ciencia en la que continuamente se está avanzando, creando nuevos productos, surgiendo necesidades y adaptando tecnologías a ellas, los desarrolladores de aplicaciones son aquellos que deben cargar con el peso de esto. Al contrario que otras ciencias como la medicina, el mundo de la informática avanza muy rápido, constantemente se están produciendo cambios, y nosotros debemos adaptarnos a ellos.

Es por ello que cuando uno se enfrenta a una tecnología, que, aunque en este caso tenga ya largo recorrido y esté bastante establecido como es el Lenguaje de Consulta Estructurada, del inglés **Structured Query Language** o por sus siglas, SQL, siempre es de agradecer tener una herramienta para los primeros momentos del aprendizaje, aquellos en los que el usuario se pueda sentir más perdido y sea más propenso a fallos y errores como aquellos producidos por, en este caso, el punto y coma (;), o sentencias peligrosas de ejecutar ya sea por pérdidas masivas de datos sin que el usuario sea consciente de ellas.

Otra de las motivaciones que me llevaron a hacer el proyecto sobre esta tecnología en concreto, es la robustez y lo sorprendentemente bien que ha aguantado los años dicha tecnología sin convertirse en obsoleta. La primera versión del lenguaje data de inicios de 1974, y a día de hoy sigue siendo el lenguaje número uno en manejo de datos a través de Sistemas Gestores de Bases de Datos, sin previsión de que esto vaya a cambiar en corto plazo, ya que constantemente el estándar se va actualizando para adaptarse a las nuevas necesidades que van surgiendo con el tiempo.

2. ANÁLISIS DEL MERCADO ACTUAL

La principal apuesta del proyecto es el uso del sólido y consolidado lenguaje SQL, utilizado por las principales y empresas a nivel mundial.

A día de hoy, está emergiendo una prometedora nueva tecnología llamada NoSQL, que aún teniendo gran recepción por parte del público de desarrolladores indies y compañías pequeñas, está teniendo problemas en acercarse al ámbito de las grandes empresas, ya que un cambio de tecnología supondría el rehacer y trasladar los datos de las ya robustas y fiables bases de datos SQL tradicionales a estas más modernas.

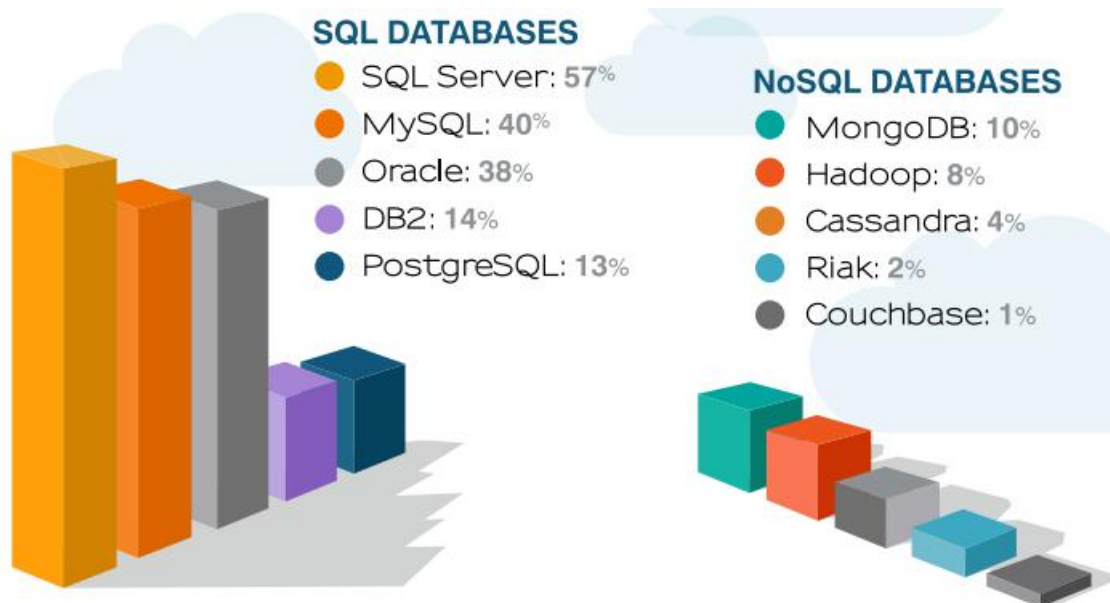


Imagen 2. Comparación proporción de uso actual entre diferentes tecnologías.

Como podemos observar en la gráfica de la página anterior, el gran grueso del mercado actual utiliza las clásicas bases de datos SQL. Los porcentajes del gráfico no son excluyentes, esto quiere decir que en la práctica las compañías según sus necesidades utilicen varias de estas tecnologías o Sistemas Gestores de Bases de datos para aprovechar las particularidades y beneficios de cada uno. También se puede observar que, dentro de las bases de datos tradicionales, las dos grandes bazas son la apuesta de Microsoft con Microsoft SQL Server, y la apuesta de Sun Microsystems (ahora parte de Oracle Corporation).

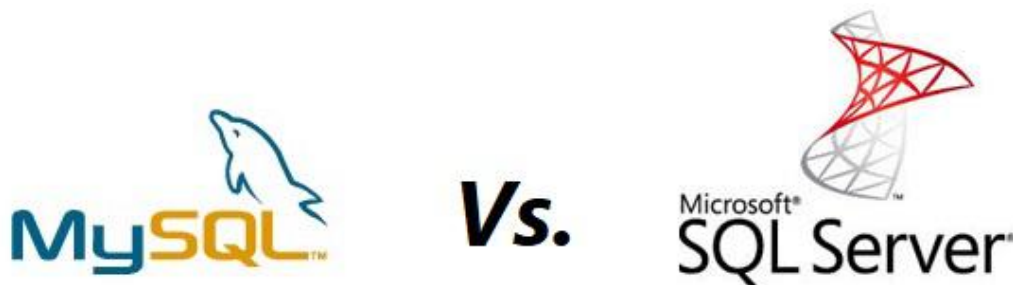


Imagen 3. Las dos principales marcas dominantes del mercado.

Estas dos grandes marcas unidas copan una gran cuota de mercado entre ellas. Es por ello que el proyecto se centra en dar soporte a ellas.

Además, ambas cuentan con versiones gratuitas específicas para que en caso de ser estudiante o pequeño desarrollador, podamos utilizarlas y aprender con ellas sin tener que asumir costes iniciales.

2.1. ESTUDIO DE PRODUCTOS EXISTENTES

Para los Sistemas Gestores de Bases de Datos a utilizar, existen dos completas y complejas soluciones en el mercado para su manejo y utilización: MySQL WorkBench y SQL Server Management. Las aplicaciones fueron creadas y son a día de hoy mantenidas por los mismos creadores de dichos Sistemas Gestores, los mencionados anteriormente Microsoft y Sun Microsystems.

La primera aplicación a comentar es MySQL WorkBench. A destacar sobre su competidora, esta dispone de dos versiones; una Community, licenciada sobre GPL, de código abierto, gratuita y almacenada en GitHub para que cualquier usuario pueda contribuir a su desarrollo, y otra versión Standard de código cerrado, destinada a empresas, con soporte y diferentes planes de pago según las necesidades del negocio, así como añade algunas series de plugins y módulos de código cerrado.

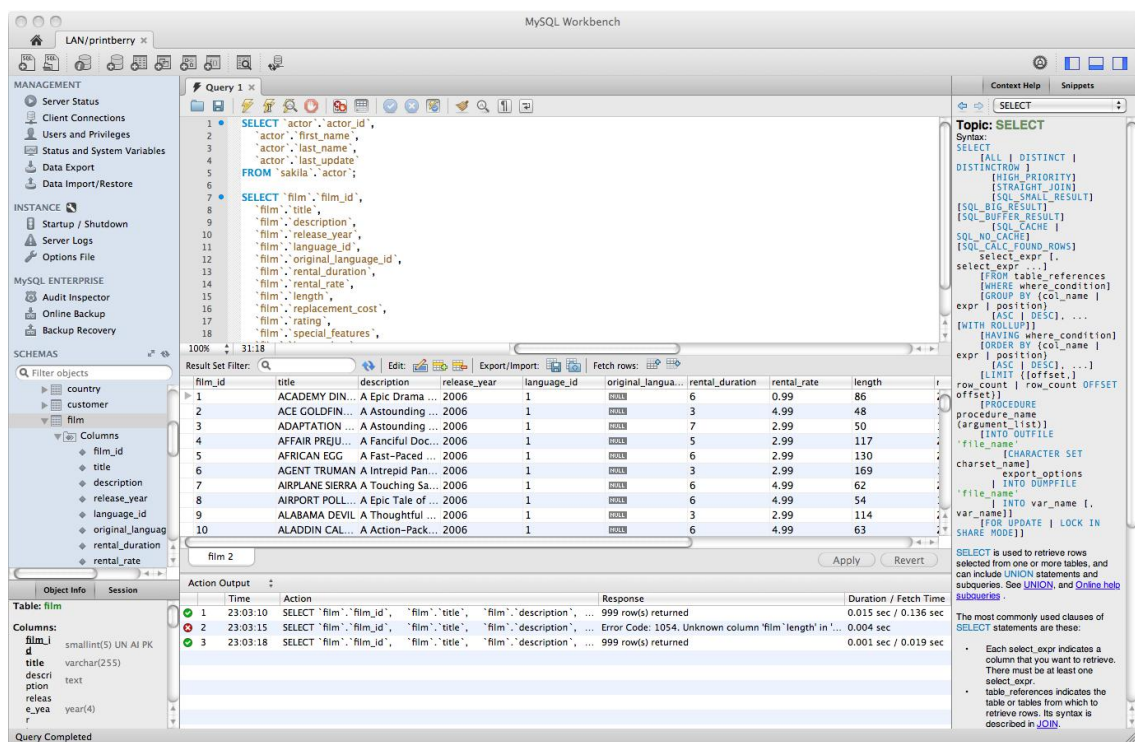


Imagen 4. Interfaz de usuario de MySQL WorkBench (MacOs)

Esta herramienta dispone de un potente editor gráfico específico para diseñar estructuras de bases de datos, tablas y relaciones entre ellas. Desde su introducción se ha vuelto muy popular dentro de la comunidad MySQL. Actualmente es el segundo producto más descargado desde la web de MySQL con más de 250.000 descargas al mes, a parte de haber sido calificado por revistas open-source como el mejor manejador de bases de datos de código abierto existente.

SQL Server Management Studio es la respuesta de Microsoft a la necesidad de un software de manejo sus bases de datos, ya que la primera versión del Servidor SQL Server fue lanzada en 1989 pero no sería hasta 2005 cuando sacaron una aplicación gráfica para su manejo, unos años más tarde que MySQL. En este caso, todas las versiones de este producto están bajo licencia propietaria, y en ningún caso el código está disponible al usuario.

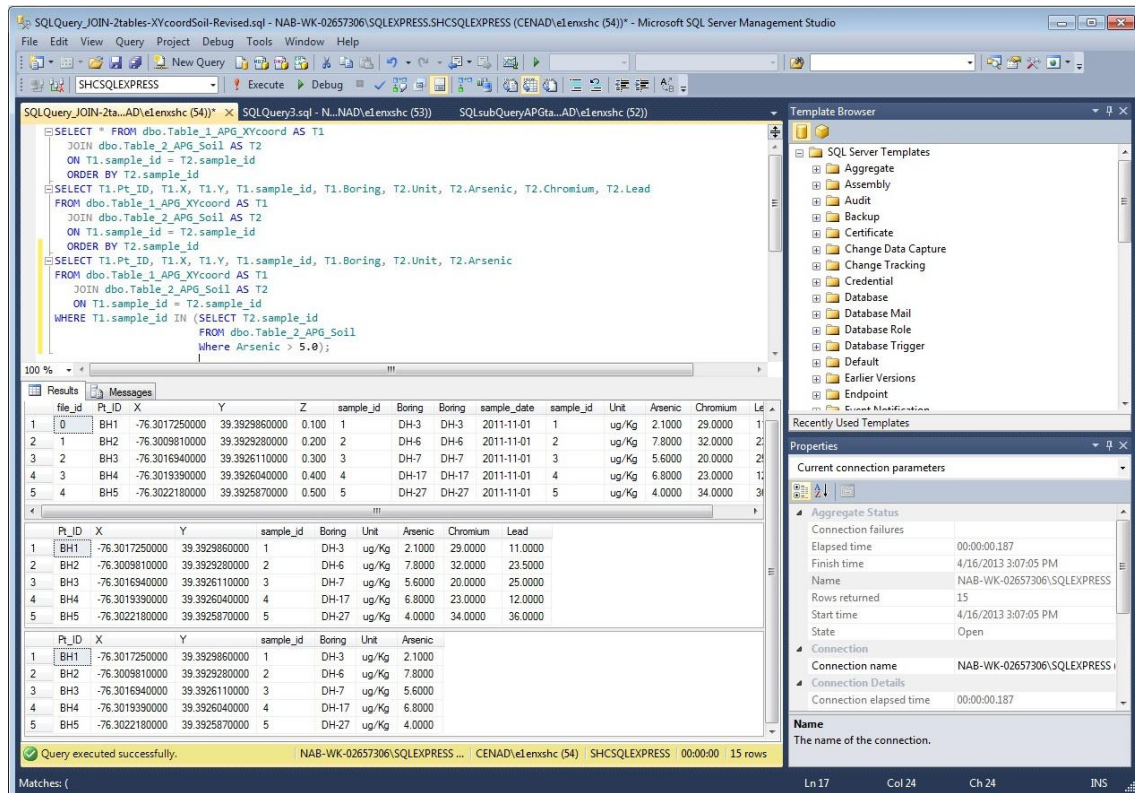


Imagen 5. Interfaz de usuario de SQL Server Management Studio (Windows)

La gran ventaja de esta herramienta es que si estás desarrollando en un lenguaje .NET y usas el entorno de desarrollo Visual Studio tienes gran integración en ambas, encontrarás a mano la mayoría de las herramientas que puedas necesitar usar para tu proyecto .NET, como podrías encontrar en Visual Studio. Provee de métodos y maneras de acceder a los datos de manera segura y probada, y ahorra al desarrollador de tener que escribir sus propias funciones para el acceso o manipulación de datos.

2.2. CONCLUSIONES DEL ESTUDIO

En los tiempos que corren, el poder consultar, almacenar y modificar información es algo totalmente necesario para cualquier empresa, esto se traduce como que en la práctica mayoría de empresas de una envergadura considerable se necesita personal dedicado al tratamiento de grandes datos.

Constantemente se generan nuevos puestos de trabajo relacionados con este ámbito, y para poder rellenar esos huecos necesitan personas que previamente reciban una formación al respecto.

Imagen 6. Logotipo de SQL

A día de hoy las herramientas destinadas para este fin pecan de los mismos fallos. Aún habiendo visto las dos más importantes del mercado, las demás soluciones en mayor o menor medida tienen un mismo



denominador común: son bastante grandes y complejas, requieren de una buena formación previa para poder empezar a ejecutar operaciones sencillas (casi todo debe hacerse por comandos escritos sin mucha ayuda por el entorno) y en definitiva son poco intuitivas y complejas de usar para un usuario novato, que suele esperar ver resultados rápidos y avances en poco tiempo.

Es por ello, unido a la larga vida y el gran futuro de esta tecnología, de la que se tiene la necesidad constante de aprendizaje por parte de nuevos usuarios de ella, que existe un hueco para este nicho de usuarios del que esta aplicación puede sacar provecho para establecerse y ser una herramienta útil para un primer acercamiento del usuario al mundo del tratamiento de datos por medio de consultas con el lenguaje SQL.

3. PROPUESTA INNOVADORA

En la actualidad, docentes y alumnos carecen de una herramienta de fácil manejo, portable, sin instalación, gratuita, de código abierto y casi nula necesidad de formación previa como la desarrollada en este proyecto.

Es por ello que, esta aplicación puede entrar en un mercado específico que apenas está cubierto, y atraer a cierta cantidad de usuarios ya que no tiene competidores directos.

La herramienta, está pensada para los primeros momentos de la enseñanza de un alumno, por lo que se prevee que éste en un periodo de tiempo no muy largo, cuando avance en su aprendizaje y tenga más comprensión del lenguaje SQL, pase a utilizar herramientas más sofisticadas y que ofrezcan soluciones más completas.

Es por ello que se ha tenido en cuenta el peso de esta, reduciendo a mínimos el tamaño que ocupa el ejecutable, y la sencillez de su comienzo de uso, así como lo fácil de deshacerse de ella una vez el usuario lo estime conveniente. La aplicación final para el alumno consta de un archivo ejecutable .exe que solo necesita abrir y listo, nada de instaladores o desinstaladores ni tampoco necesidad de dependencias de terceros programas u librerías de componentes.

Aun así, el usuario final terminará comprobando que para ciertas operaciones rápidas y sencillas de consulta e inserción de datos este programa sigue siendo mucho más rápido de usar, ya que los tiempos de carga de la aplicación, como por tiempos de creación de un comando son bastante menores en comparación con las grandes soluciones de los fabricantes, que en equipos modestos suelen tardar bastante en hacer la carga inicial.

Otro punto importante es el hecho que este proyecto desde el minuto 0 de su creación ha estado siendo almacenado y actualizado en la plataforma online número uno en cantidad de usuarios y proyectos de código abierto y colaborativos conocida como GitHub.



Imagen 7. Logotipo de GitHub

Esto, dado que es una aplicación de desarrolladores para desarrolladores, es una ventaja importante frente a las alternativas de código cerrado y propietarias, ya que permite que el proyecto se pueda mantener a lo largo del tiempo gracias a contribuciones o aportaciones de personas anónimas que quieran ayudar de manera desinteresada con tan sólo descargarse la solución para Visual Studio completa, que contiene los archivos de código fuente sin compilar, imágenes de prueba, ejemplos, archivos documentados y scripts SQL de pruebas. Todo ello se encuentra alojado en la misma web para una mayor comodidad para el desarrollador final, que encontrará con la documentación y ejemplos una manera sencilla de continuar con la labor de desarrollo.

Aquel alumno que aprenda y se beneficie con el uso de esta herramienta también podría, si está interesado, observar y mejorar el funcionamiento interno y aprender cómo la aplicación trabaja en su interior, esto es: las llamadas a bases de datos, la manera de gestionar las conexiones, las diferencias entre los conectores MySQL y SQL Server... También podría tener un efecto llamada y animar a esa persona a acercarse al mundo de las tecnologías .NET como WPF y el lenguaje C# con los que está construida la aplicación.

4. ESTUDIO DE VIABILIDAD

En este apartado se tienen en cuenta y se analizan los gastos materiales necesarios para la implantación del software de cara al usuario final. También se examinan las tecnologías usadas para la creación y desarrollo del proyecto.

Se tiene en cuenta que, en caso de tratar de analizar los costes de los recursos humanos, estos son soportados de manera íntegra por el autor del proyecto y no se cobrará ningún importe en la realidad, ya que se realiza sin ánimo de lucro, abierto y será cedido a la comunidad de desarrolladores de software libre por lo que no tendrá un beneficio económico.

4.1. RECURSOS Y TECNOLOGÍAS A EMPLEAR

A la hora de analizar los medios empleados para el desarrollo y uso de este proyecto, podemos diferenciar dos grandes campos. Por un lado examinaremos los recursos utilizados en el lado del desarrollador, aquellos que han hecho posible que el proyecto cobre vida y haya llegado hasta donde está ahora, y por el otro indagaremos las tecnologías o bienes necesarios por parte del usuario final para poder hacer uso de este software.

Empezando por el lado del desarrollador, las siguientes tecnologías ahora listadas han sido utilizadas a lo largo del desarrollo de este producto:

- **Visual Studio:** es el entorno de desarrollo integrado (IDE) por excelencia para crear aplicaciones nativas para el Sistema Operativo Windows. Su desarrollador es Microsoft, autor del nombrado S.O. Soporta múltiples lenguajes de programación, y su uso no sólo se limita a aplicaciones para Windows, también tiene grandes capacidades a la hora de crear aplicaciones Web. Cuenta con un muy potente editor gráfico de ventanas que permite en poco tiempo obtener grandes resultados, abstrayendo al desarrollador de escribir gran parte del código

relacionado con la Vista de la aplicación. Tiene gran integración con bases de datos (sobretudo SQL Server) e incluye herramientas para el manejo de Git y GitHub. Existe una versión Community gratuita que es la que ha sido elegida para esta ocasión.



Imagen 8. Logo de la aplicación Visual Studio.

- **Windows Presentation Foundation:** También conocido como WPF, es una tecnología de Microsoft. Presentada junto con el sistema operativo Windows Vista, es el relevo de la antigua tecnología usada para crear ventanas en anteriores sistemas operativos Windows, conocida como Windows Forms, que aún sigue siendo soportada. La principal ventaja es que separa, usando el lenguaje de marcas XAML y los lenguajes .NET la interfaz de interacción de la parte del código, propiciando una arquitectura Modelo Vista Controlador. Esta tecnología permite crear unas interfaces de usuario que se adaptan independientemente de la resolución de la pantalla, y todos los gráficos se representan usando Direct3D, por lo que están optimizadas y ganan en velocidad por aceleración de la tarjeta gráfica del equipo.



Imagen 9. Logo de la tecnología WPF.

- **Git con GitHub:** Git es un software de control de versiones diseñado por el creador del kernel de Linux, Linus Torvalds. Su propósito es llevar registro de los cambios en archivos del proyecto y coordinar el trabajo en caso de que varias personas trabajen con los mismos archivos.

Permite al desarrollador ver un historial de los cambios producidos y volver en cualquier momento a un estado anterior del proyecto. También permite separar en ramas de desarrollo el código, por si se necesitan hacer pruebas o incorporar cambios que no se saben si pueden afectar a la estabilidad del software. Al estar configurado con GitHub, permite que el proyecto con su historial de cambios esté alojado en la nube, fuera de nuestro equipo local (a pesar de que se mantiene una copia local). Esto permite la posibilidad de trabajar en el código desde más de un ordenador, al mismo tiempo que al tener una copia siempre actualizada en la nube da la seguridad de que nuestro trabajo no va a desaparecer en ningún momento. Desde el primer momento del desarrollo de este software se lleva dejando constancia en la plataforma GitHub, para no dejar posibilidad a ninguna pérdida de datos que pudiese producirse en la copia local.

Una de las grandes ventajas que ofrece GitHub a la hora de depositar tu proyecto en su plataforma es la gran cantidad de usuarios activos existentes. Es la mayor plataforma de código abierto colaborativa que existe en la actualidad. Estos usuarios pueden colaborar y ser partícipes de tu proyecto, ayudando a mejorar el código, implementando nuevas funciones, o dándote consejos sobre cómo mejorar el software.



Imagen 10. Logo de la tecnología Git y la plataforma GitHub

- **SQL Server:** es un sistema gestor de base de datos relacional desarrollado por Microsoft , como se ha hablado de ella previamente en este manual. Se ha seleccionado por su gran integración y facilidad de uso en Visual Studio, que hace que ayuda al desarrollador de tener que lidiar con algunos de los problemas como seguridad de conexión y número de conexiones permitidas. Esta herramienta se utiliza a la hora de relacionar y almacenar con persistencia los usuarios con su contraseña para la función de guardado de conexiones de la aplicación.

- **BCrypt para C#:** es uno de los mejores algoritmos de hasheo de contraseñas que existen en la actualidad. Sus creadores fueron Niels Provos y David Mazières, y fue presentado en 1999. Es el algoritmo de contraseñas por defecto para el sistema operativo OpenBSD, así como muchas otras distribuciones Linux como SUSE Linux. Se diferencia del resto en la velocidad en la que hashea. Esta es conocida por tardar alrededor de 100ms por defecto en hashear una contraseña aunque se pueden configurar a más. Esto hace que a un posible atacante le sea inviable por el tiempo y costo computacional que conlleva para sacar una contraseña por fuerza bruta comparado con los otros algoritmos que obtienen resultados en tiempo mucho menores.

Se ha utilizado una implementación en forma de librería para C# escrita por Damien Miller.

\$2y\$10\$6z7GKa9kpDN7KC3ICW1Hi.f0/to7Y/x36WUKNP0IndHdkdR9Ae3K

- Algoritmo
- Algorithm options (eg cost)
- Salt
- Hashed password

Imagen 11. Anatomía del hash devuelto por el algoritmo BCrypt.

- **SQL Server Management Studio, Bases de datos MySQL y MySQL**

WorkBench: Las anteriores herramientas, que proveen de una interfaz gráfica para utilizar las bases de datos, se han utilizado a lo largo del desarrollo para probar que las funcionalidades de la aplicación de los comandos lanzados contra bases de datos han tenido una ejecución correcta. También han permitido crear y definir las propias bases de datos de pruebas y sus tablas.

- **Conector MySQL para .NET:** Propiedad de la compañía Oracle, se ha utilizado en el proyecto la nueva versión 8, recién salida este año, para asegurar tener el máximo rendimiento y soporte. Se encarga de proveer funcionalidades críticas a la hora de establecer conexiones y ejecutar comandos contra bases de datos MySQL, algo que las tecnologías .NET no proveen de forma nativa.

- **Otros:** durante el desarrollo del software se han utilizado otros recursos que se listan a continuación:

- Ordenador con el Sistema Operativo Windows 10.
- Suite ofimática WPS con la que se ha redactado la Memoria.
- Notepad++ para el manejo de scripts SQL.

La lista de tecnologías que el usuario final debe tener para hacer uso de la aplicación es considerablemente menor, ya que éste sólo necesita un ordenador personal con una versión de Windows superior a Vista. En el caso de querer disponer de las funciones de almacenado de datos de conexiones para un determinado usuario, parte importante de la aplicación pero no imprescindible, también necesitaría tener instalada una instancia de SQL Server en su máquina o en un servidor externo que gestione los datos. También necesitaría como parte evidente la base de datos MySQL o SQL Server contra la que ejecutar los comandos que genere con la aplicación.

4.2. EVALUACIÓN DE COSTES

Podemos dividir la parte de la evaluación de costes en tres grandes ámbitos. El coste de los recursos humanos, el coste de las licencias, y el coste del mantenimiento futuro del proyecto.

Para empezar, ya que como se ha recalcado previamente, la aplicación ha sido en todo momento de código abierto y sin ánimo de lucro, no ha existido hasta el momento del lanzamiento del software ningún tipo de coste humano. Todo el trabajo lo ha llevado el autor del proyecto (yo) sin recibir retribución.

Una vez comentado el tema del coste humano, se entra a valorar el tema de las licencias del software necesarios a utilizar. En este proyecto se ha optado por utilizar las versiones gratuitas que existían para las tecnologías usadas ya que perfectamente cumplen para nuestros requerimientos. No se ha tenido que pagar ninguna licencia. Con Microsoft Visual Studio se ha optado por la Community Edition. Esta versión es de licencia gratuita para desarrolladores. A pesar de no tener todos los contenidos, como los tiene la versión Professional, se podemos realizar todas las tareas de desarrollo que se han requerido. Con Sistema Gestor de Base de Datos de SQL Server se ha optado por la Developer Edition. Este tipo de licencias, no contienen soporte técnico pero ese no ha sido problema ya que todo el soporte y dudas que he ido pudiendo necesitar las he resuelto con la ayuda de información de internet.

Para terminar se analiza el tema del mantenimiento futuro del producto. Ya que desde el principio se ha apostado por la filosofía open-source y de trabajo colaborativo sin ánimo de lucro, el proyecto no va a requerir de costes futuros. Siguiendo futuras versiones y nuevas mejoras se esperan que sean entregadas de manera gratuita al público, así como que los costes de mantenimiento del alojamiento de los ficheros de código y la aplicación en sí se realizan de manera gratuita y de por vida por la plataforma GitHub.

4.3. PLANIFICACIÓN DEL TIEMPO

Una vez realizada la exposición inicial de ideas del proyecto al tutor de seguimiento, se comienza con su planificación. Esta planificación se ha realizado teniendo en cuenta que el autor de la aplicación está realizando unas prácticas de formación en centro de trabajo al mismo tiempo que elabora el proyecto.

De esta planificación inicial se obtienen los siguientes puntos importantes que resumen el tiempo de trabajo total que puede llevar tardar en realizarse.

- Reuniones con el tutor: se han dedicado unas 5 horas en todas las reuniones con el tutor, junto con otras 5 añadidas por el desplazamiento más 2 horas extras resolviendo dudas junto a otros profesores o alumnos. En total 12.
- Estudio y análisis del proyecto: se han dedicado alrededor de unas 15 horas en el estudio y análisis previo del proyecto.
- Diseño de la base de datos: un total de unas 2 horas se han utilizado para decidir las tecnologías apropiadas para la base de datos del programa así como su diseño interno.
- Creación de la base de datos: aproximadamente se han tardado 3 horas en la creación del esquema, tablas y sus relaciones en el sistema gestor de bases de datos utilizado, SQL Server.
- Diseño y codificación de la interfaz de usuario: esta es la segunda parte del proyecto que más tiempo me ha llevado, al rededor de 30 horas se han utilizado para el diseño y su implementación de todas las ventanas de la interfaz de usuario que tiene la aplicación, ellas codificadas en XAML haciendo uso de la tecnología WPF.

- Diseño y codificación de la parte lógica de la aplicación: el grueso de la dedicación del tiempo se lo lleva, como no podía ser de otra manera, la parte del diseño y codificación de la lógica de negocio, todo aquello relacionado con el funcionamiento del software de manera interna. Aproximadamente un tiempo total de 60 horas.
- Instalación de software necesario: en este apartado se recoge el tiempo necesario utilizado para la instalación de las herramientas de software usadas para el desarrollo del proyecto. Unas 5 horas han sido invertidas entre el tiempo de descarga y tiempo de instalación de las tecnologías.
- Memoria del proyecto: por ultimo se tiene en cuenta el tiempo total utilizado para hacer esta memoria. Aproximadamente 20 horas se han necesitado para elaborar el contenido de la misma, sumando tiempos de análisis y búsqueda de datos así como el tiempo de plasmarlos y darle un formato correcto a este documento.

Según la planificación resultante, el proyecto ha tenido una duración de 177 horas. Teniendo en cuenta que para el desarrollo del proyecto se ha dispuesto de unos 3 meses de trabajo, Este se ha desarrollado con una dedicación aproximada de 12 horas semanales.

4.4. ALCANCE DEL PROYECTO

En este apartado se evalúa el alcance actual y futuro a medio y largo plazo de este proyecto.

A la hora de entregar el proyecto, se tiene idea de que esté disponible para el usuario final en el mismo momento que la entrega. Este podrá hacer uso de él sin necesidad que el autor le asista personalmente e instruya a utilizarlo, ya que encontrará ejemplos de uso e instalación el servidor donde se encuentra la solución, en este caso GitHub.

A medio plazo se espera que con parte de ayuda de la comunidad de software libre y siendo liderado por el autor del proyecto, se sigan añadiendo mejoras, depurando funcionalidades y realizando optimizaciones que hagan de este producto uno más sencillo de usar si cabe. Es posible que algún alumno que utilice la aplicación se interese por el funcionamiento interna de esta y debe saber que tendrá la facilidad y la disponibilidad de aportar contenido y ayudar con el desarrollo en todo momento a través de GitHub.

A largo plazo se ha planteado un posible fork del proyecto hacia una versión Web, que contaría con los mismos principios que este proyecto. Debería ser gratuito, de código abierto y hosteado en GitHub y primar siempre la facilidad de uso. Esto permitiría utilizar el software desde cualquier dispositivo, sin tener en cuenta el sistema operativo (actualmente limitado a Windows) ni tampoco dónde se ejecuta (ahora sólo en ordenadores), ya que el dispositivo únicamente necesitaría de un navegador web.

5. ARQUITECTURA DE LA APLICACIÓN

La aplicación es una intermediaria gráfica entre el usuario que la maneje y el host final para peticiones que siguen la arquitectura de Cliente - Servidor.

Esta es un modelo de diseño de software por capas en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.

Un usuario que haga uso de ella la utilizará para generar consultas SQL de manera sencilla, y ejecutarlas contra un servidor de bases de datos que devolverá una respuesta en forma de datos o en forma de información como el número de filas afectadas.

Algunos otros ejemplos de aplicaciones computacionales que usen el modelo cliente-servidor son el correo electrónico, un Servidor de impresión y la World Wide Web.

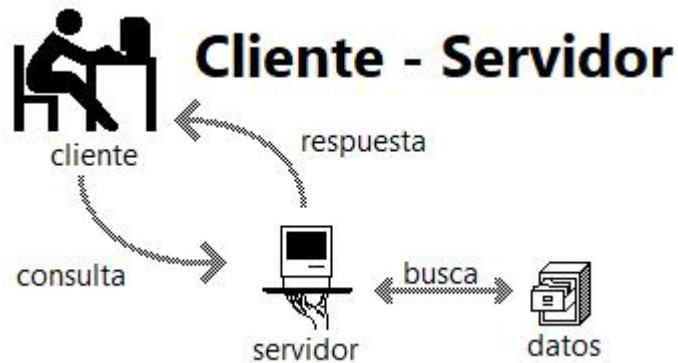


Imagen 12. Arquitectura Cliente - Servidor

A la hora del desarrollo y la manera de codificar la aplicación, se ha apostado por el uso del patrón de arquitectura de software llamado Modelo-vista-Controlador (MVC). Este patrón separa los datos y la lógica de negocio de una aplicación de la manera en la que se representa visualmente al usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

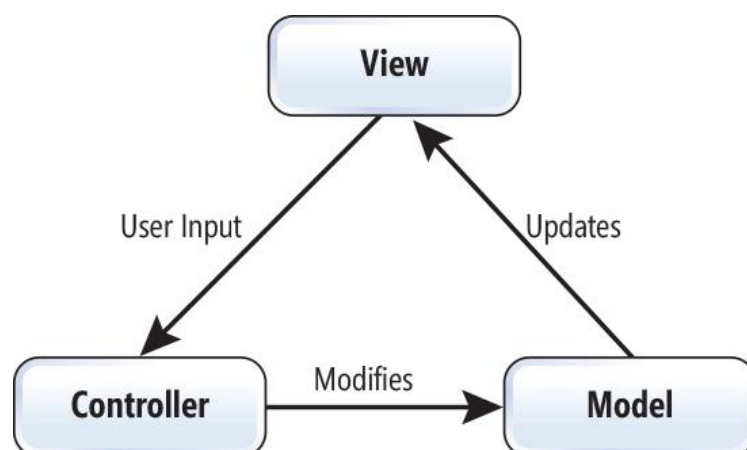
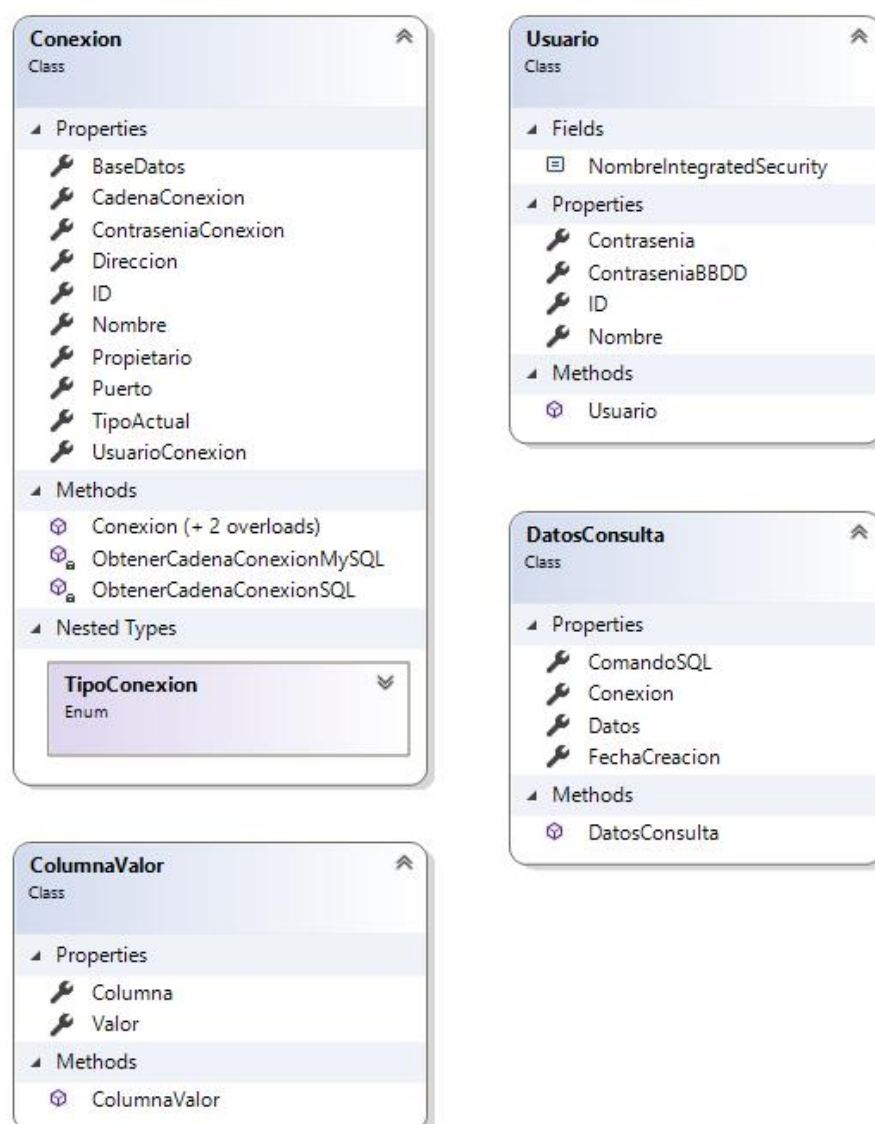


Imagen 13. Colaboración típica entre componentes en un patrón MVC.

5.1. ANÁLISIS Y DIAGRAMAS UML

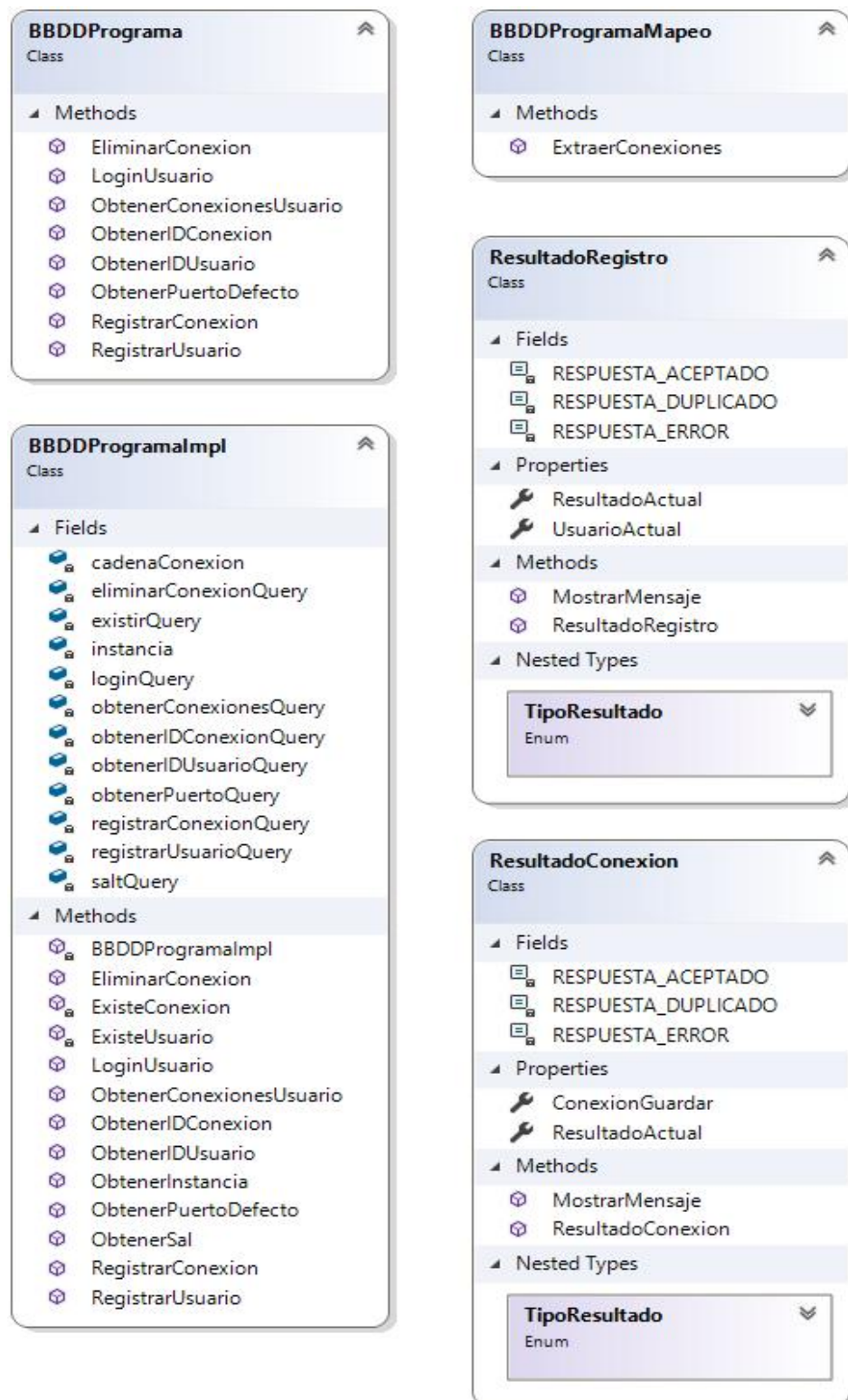
En las siguientes páginas se muestran en lista las tablas UML extraídas de las clases utilizadas durante el desarrollo del proyecto. Estas permiten ver de un vistazo rápido la manera en la que están compuestas las clases, y en ellas aparecen sus métodos y campos.

Las primeras tablas UML corresponden a los modelos de datos utilizados.



UML 1. Modelos de datos.

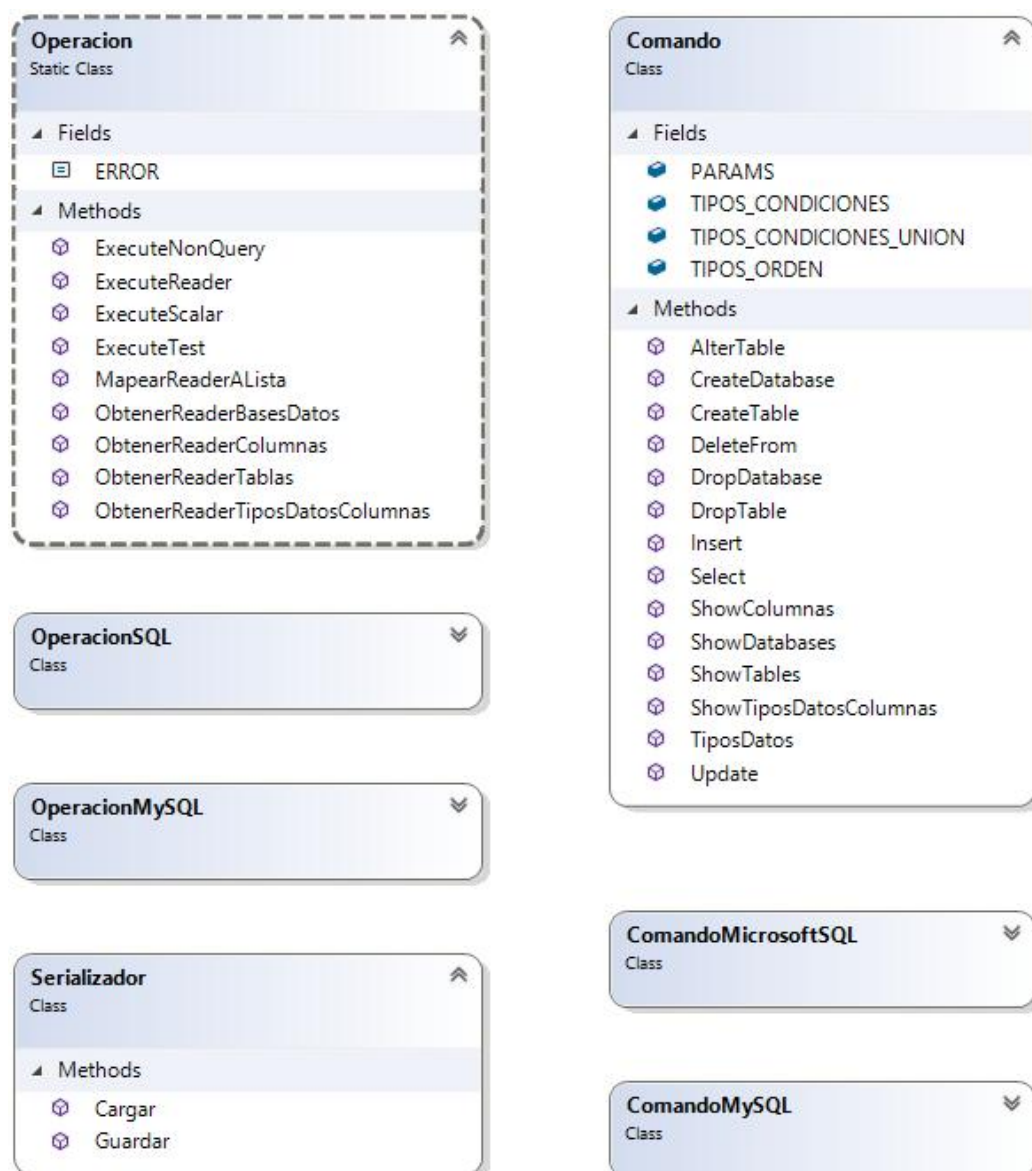
Las siguientes corresponden a las clases utilizadas para el manejo contra la base de datos utilizada por el programa para guardar usuarios y contraseñas.



UML 2. Manejo base de datos del programa.

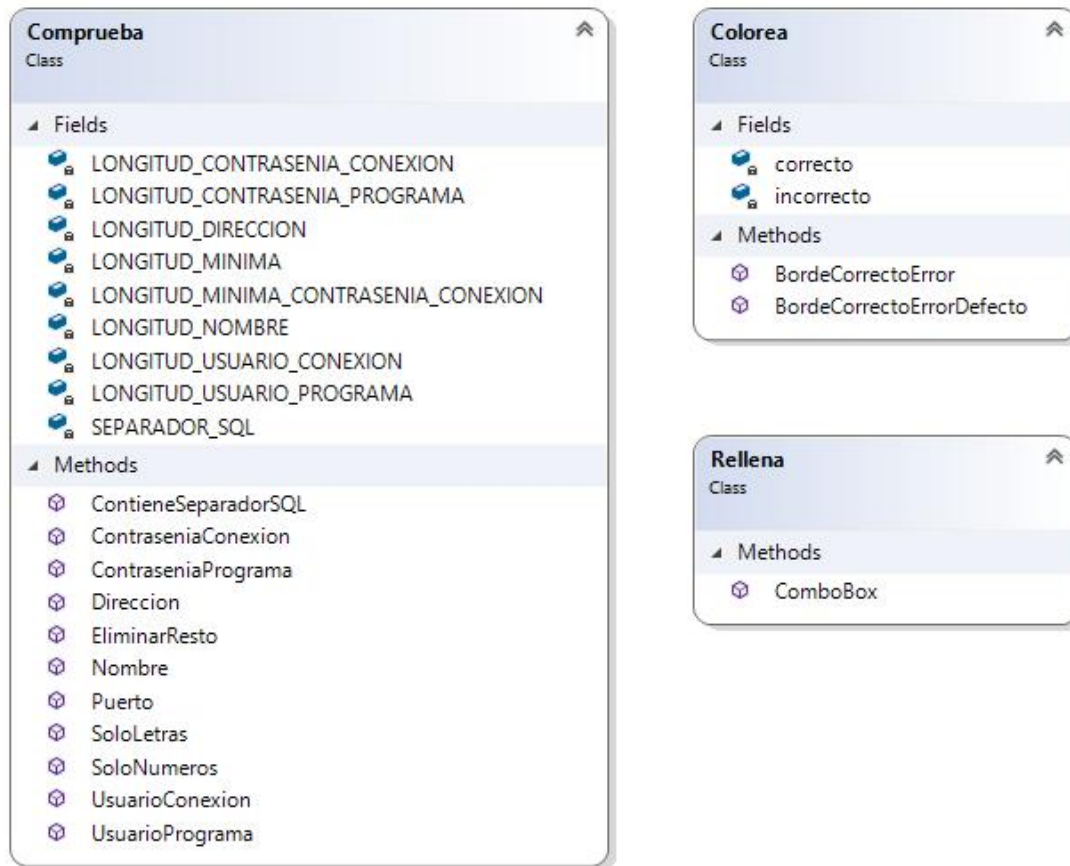
Ahora les toca el turno a aquellas usadas para lanzar consultas y comandos hacia las bases de datos que se quieran consultar. Estas clases son gran parte del núcleo del planteamiento inicial del proyecto. No se muestran aquellas llamadas *MySQL y *MicrosoftSQL ya que consisten únicamente en implementaciones específicas de la clase padre según la tecnología.

También se incluye la clase utilizada para serializar en disco de manera persistente los datos de las consultas en archivos con extensión *.easy.



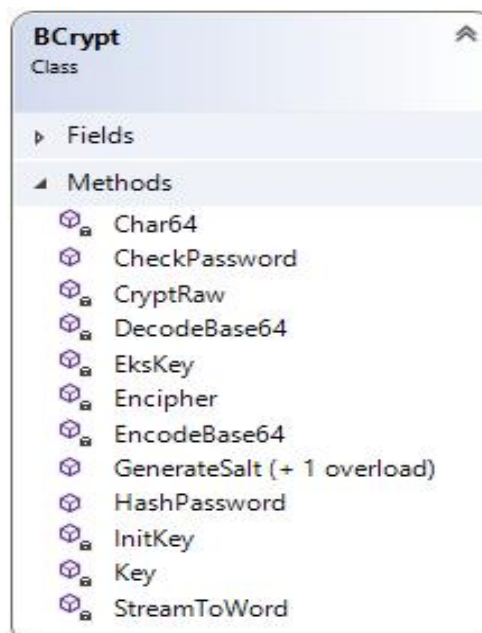
UML 3. Operaciones de manejo y consulta contra Bases de Datos externas.

Se sigue con estas clases de ayuda común para las interfaces gráficas.



UML 4. Clases del paquete Utils de código común.

A continuación, la librería utilizada para el hash de contraseñas en BBDD.

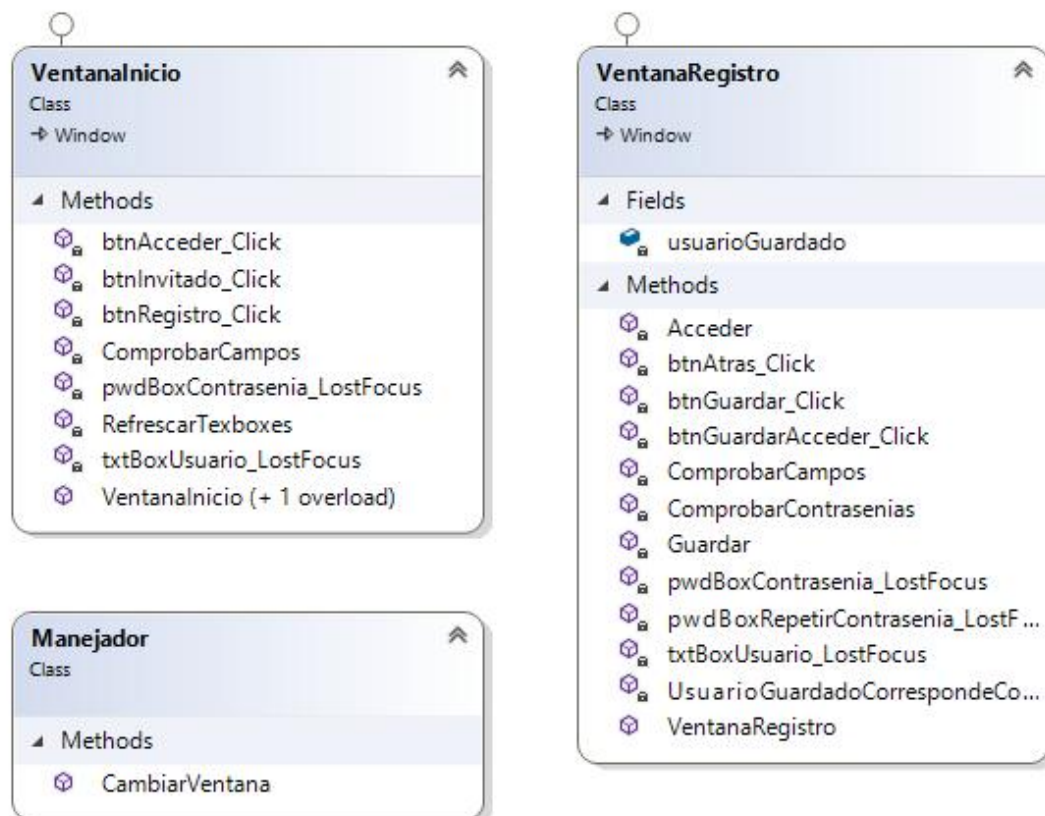


UML 5. Librería BCrypt.

Se prosigue con aquellas tablas UML relacionadas con la interfaz gráfica del usuario, o sea, las ventanas que se verán junto con su comportamiento.

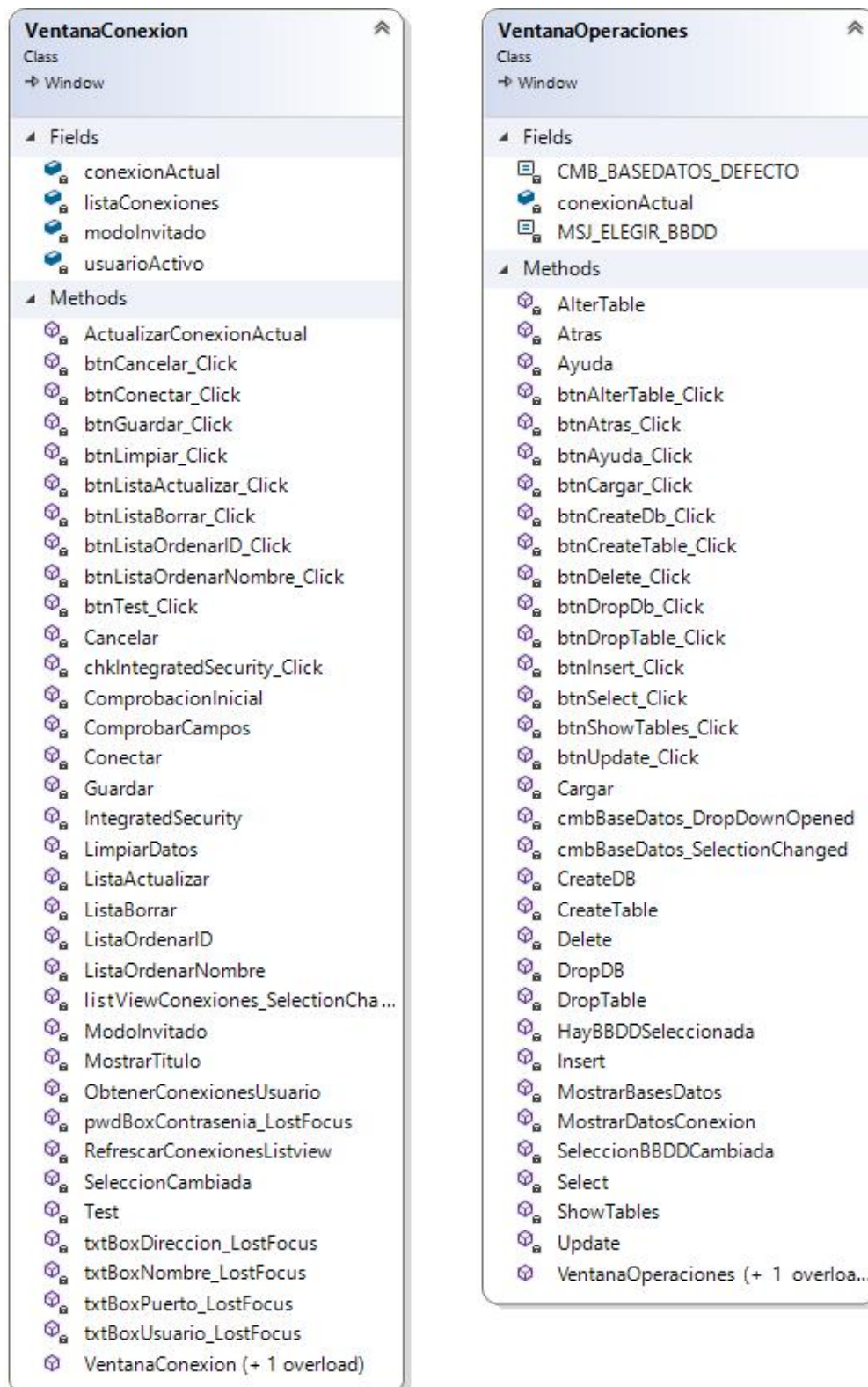
Estas se corresponden a las dos primeras ventanas que se muestran al usuario, la correspondiente con el Inicio y la ventana de Registro.

También se incluye la clase ayudante para gestionar el cambio de una ventana a otra durante toda la aplicación.



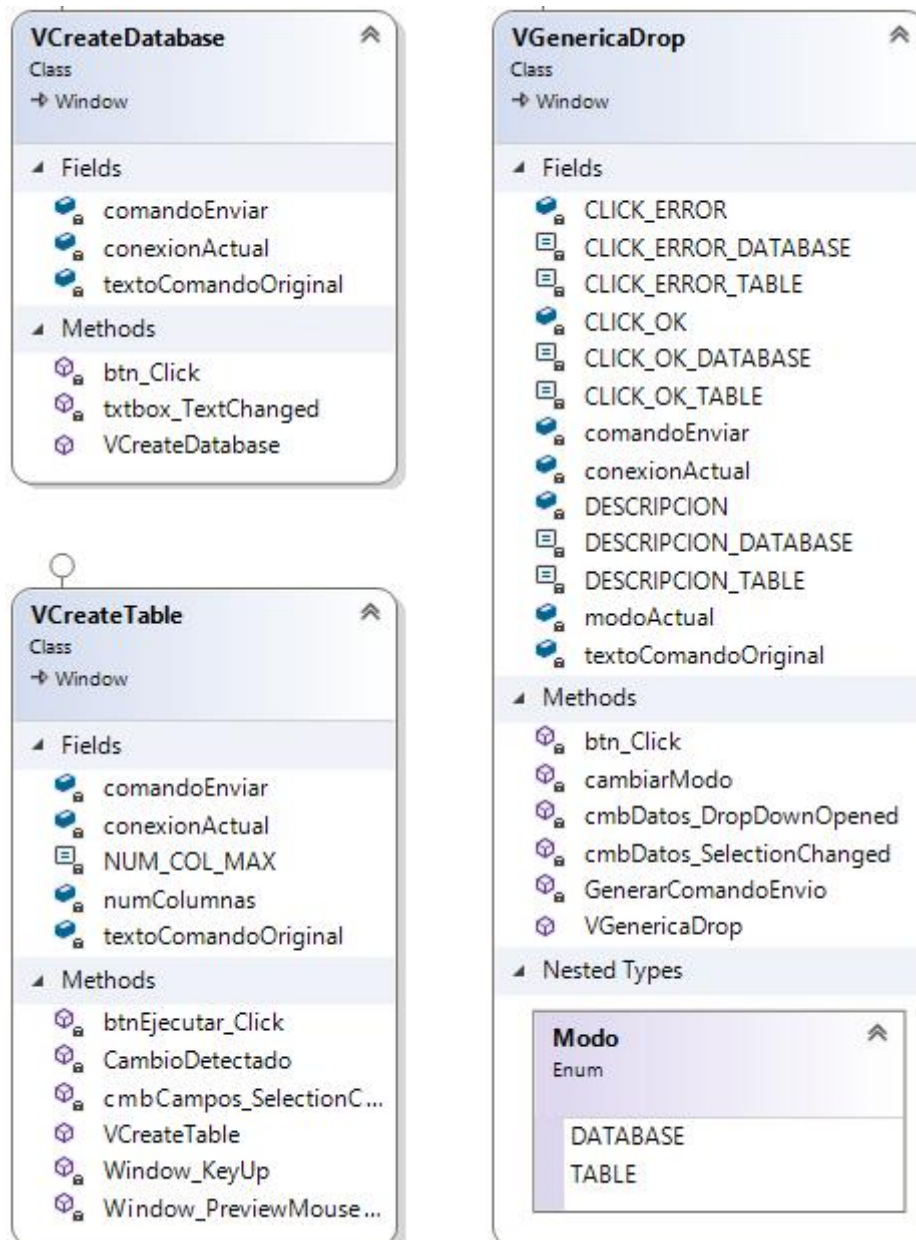
UML 6. Ventanas iniciales.

Estas corresponden a las dos grandes ventanas de entrada de datos del usuario.



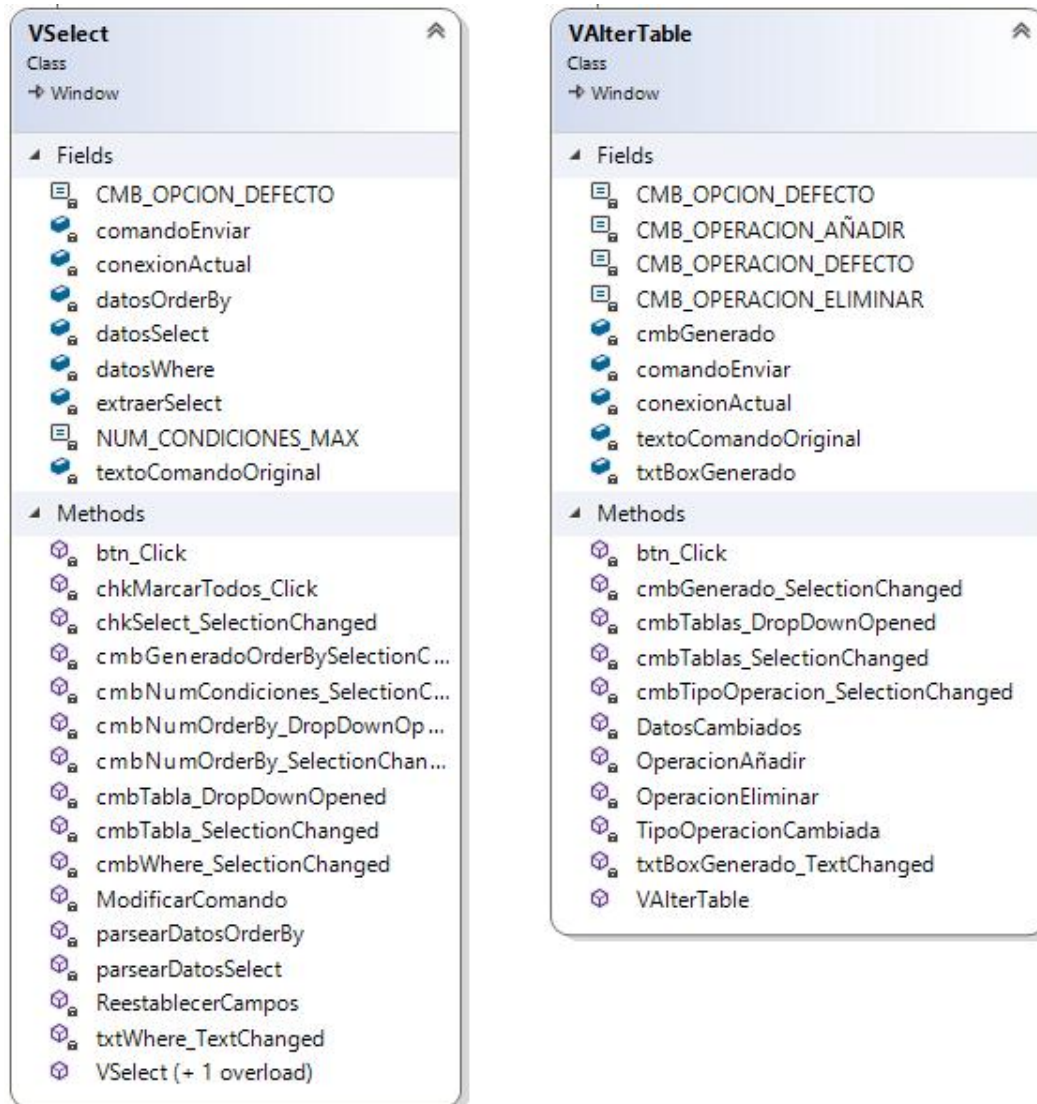
UML 7. Ventanas Conexión y Operaciones.

Se muestran ahora los UML relacionados con las ventanas relacionadas con la ejecución de comandos y consultas contra bases de datos.

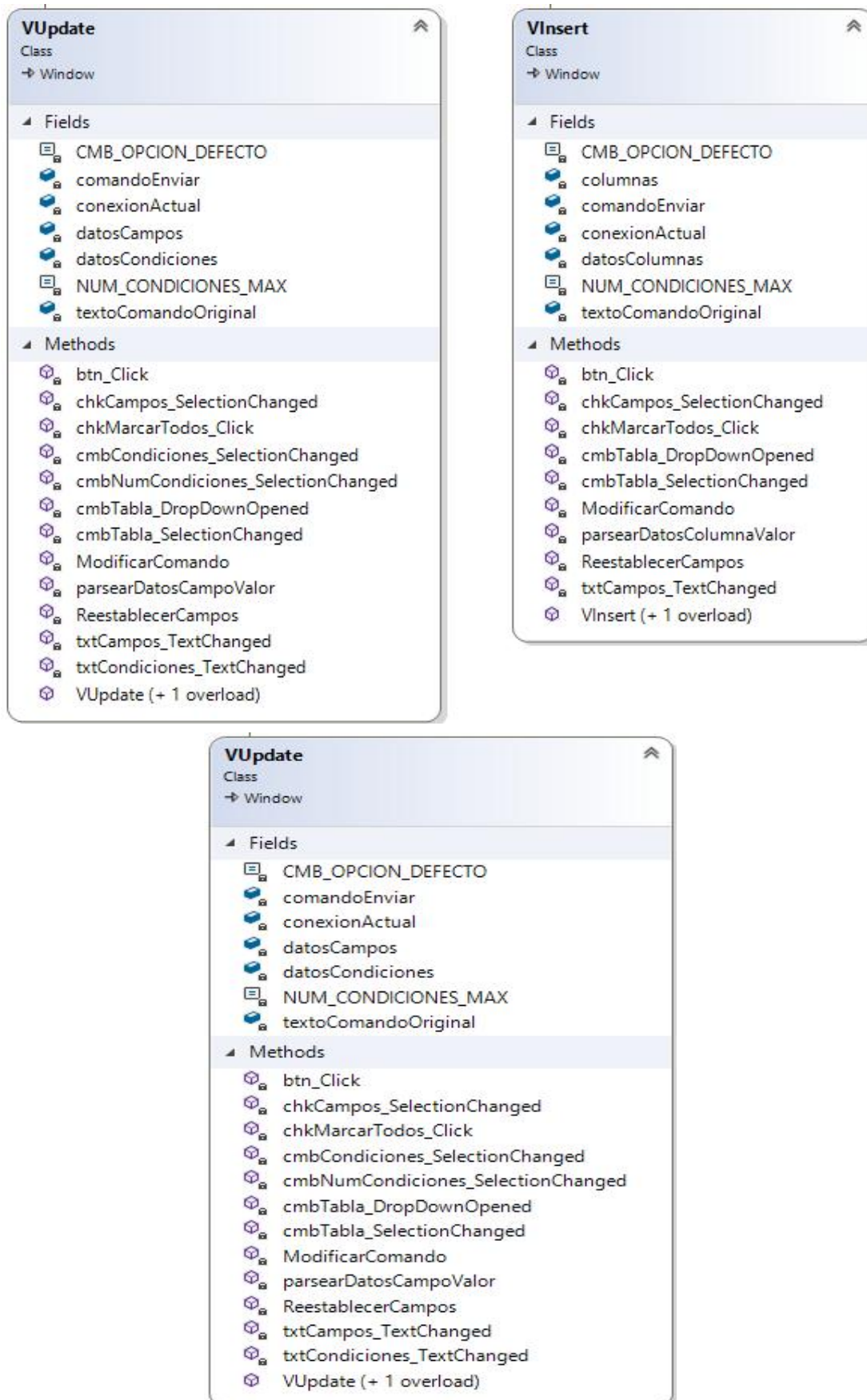


UML 8. Ventanas de comandos I.

Segunda parte de UML relacionados con ventanas de comandos

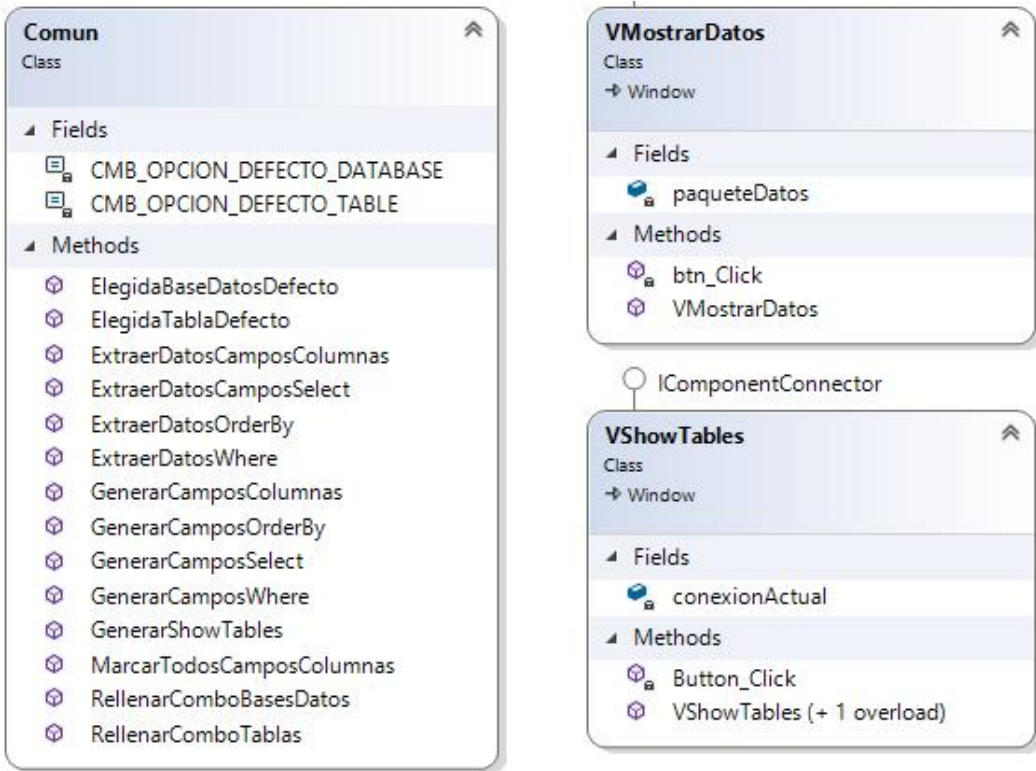


UML 9. Ventana comandos II.

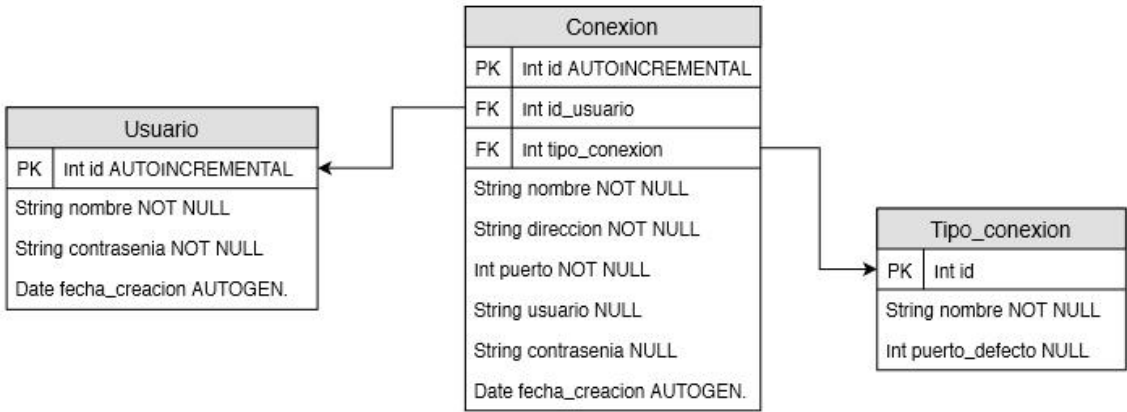


UML 10. Ventanas comandos III.

Por último se muestran las tablas UML de las dos últimas ventanas y de la clase Común que define métodos utilizados entre todas las ventanas. Al final de la página aparece el diagrama de tablas utilizado en la BBDD del programa.



UML 11. Ventanas comandos IV y Común.cs.



UML 12. BBDD del programa.

No se ha parado en entrar en detalle del comportamiento de las clases ya que estas están extensamente documentadas en el código fuente que se proporciona en las últimas páginas de esta Memoria.

5.2. ANÁLISIS Y DIAGRAMAS E/R

La base de datos de este programa se ha reducido a mínimos ya que se usa sólo para una parte opcional de éste. Permite almacenar una lista de conexiones con sus datos de conexión a cada usuario. También permite conocer el tipo de puerto por defecto que debería tener una base de datos en caso del usuario no asignar uno.

Los usuarios y conexiones tienen almacenada de manera automática la fecha en la que se crearon para poder llevar con esos datos a cabo operaciones de estadística o ordenación por fecha de creación.

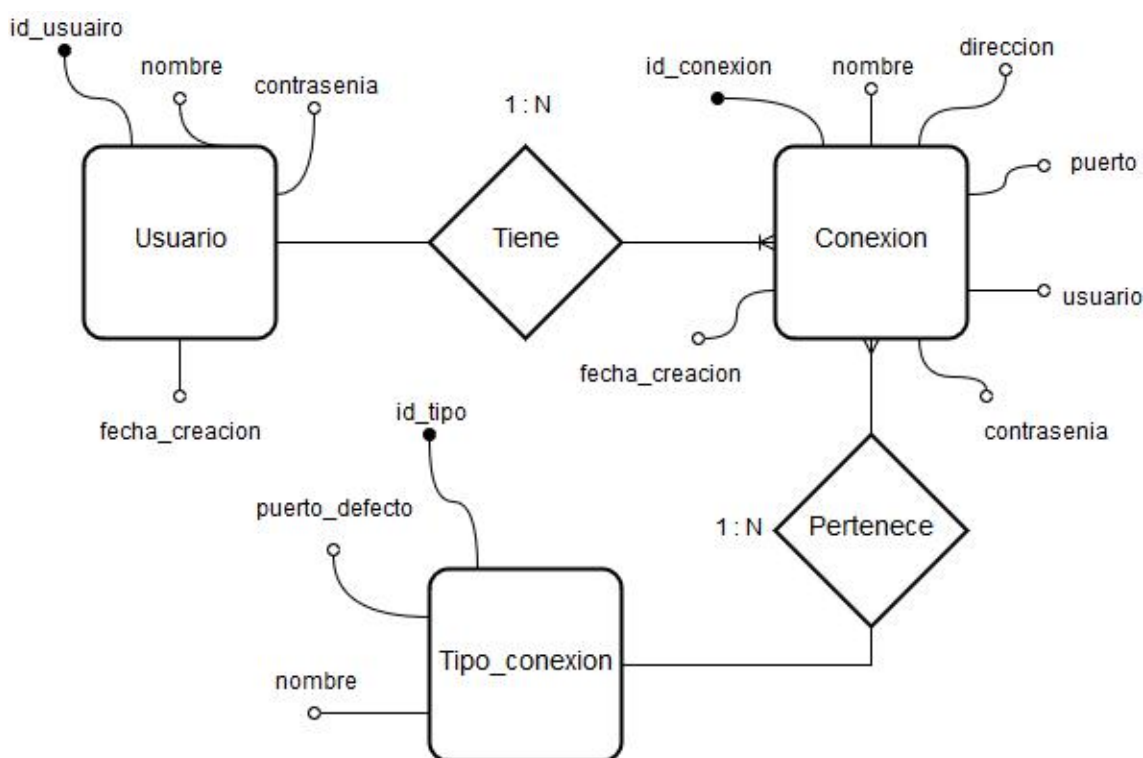


Imagen 14. Diagrama Entidad Relación de la BBDD.

6. FUTURAS MEJORAS Y CONCLUSIONES

En este apartado se exponen las posibles futuras mejoras que el autor ha pensado que el proyecto pueda tener para versiones posteriores.

- Funcionalidad para añadir soporte de Foreign Keys. Estas, aunque son totalmente necesarias en bases de datos relacionales maduras, no se han decidido añadir al programa por creer que era una funcionalidad que queda más allá del ámbito de alcance de este software.

Se ha optado por incluir sólo aquellas funcionalidades básicas para la creación y manejo, para no liar al usuario final (alumnos) con conceptos más avanzados. Lo mismo se podría aplicar para constraints tipo checks, o funciones de agregación.

- Soporte para diferentes tipos de bases de datos. Aunque se ha decidido dar soporte a las dos más grandes del mercado, sería muy bien recibido el poder trabajar con otras marcas de bases de datos, como PostgreSQL.
- Añadir funcionalidad de almacenado de conexiones en modo local sin tener que estar registrado como usuario. Estas conexiones se serializarían y almacenarían en un fichero de datos.
- Posibilidad de almacenar la configuración de una consulta para no tener que repetirla de nuevo. Esto permitiría al usuario tener una lista con las consultas SQL más utilizadas y agilizaría su uso.

7. PRUEBAS Y RESULTADOS

En las siguientes páginas se muestran una serie de pruebas de control que se han realizado con el software. Estas muestran algunas de las implementaciones de seguridad y tolerancia a fallos que se han desarrollado.

El primer caso muestra un intento de registrar un usuario con un nombre que ya se encuentre registrado previamente. El software maneja la situación impidiendo crear dicho usuario y avisando a éste del fallo.

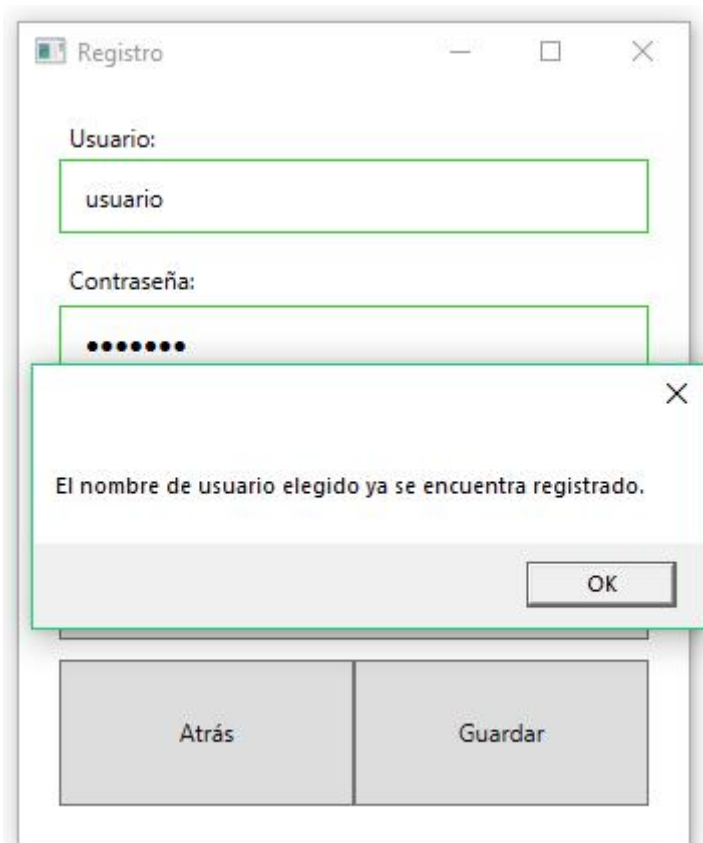


Imagen 15. Aviso de usuario ya registrado.

Este software, aun siendo una aplicación de escritorio tradicional, tiene métodos de validación que muestran de manera visual al usuario con colores como el verde o rojo cuando uno datos introducidos en los campos de tipo input tiene un fallo de formato.

Esta característica suele ser más propia de aplicaciones para móviles o web, que se centran en el diseño.

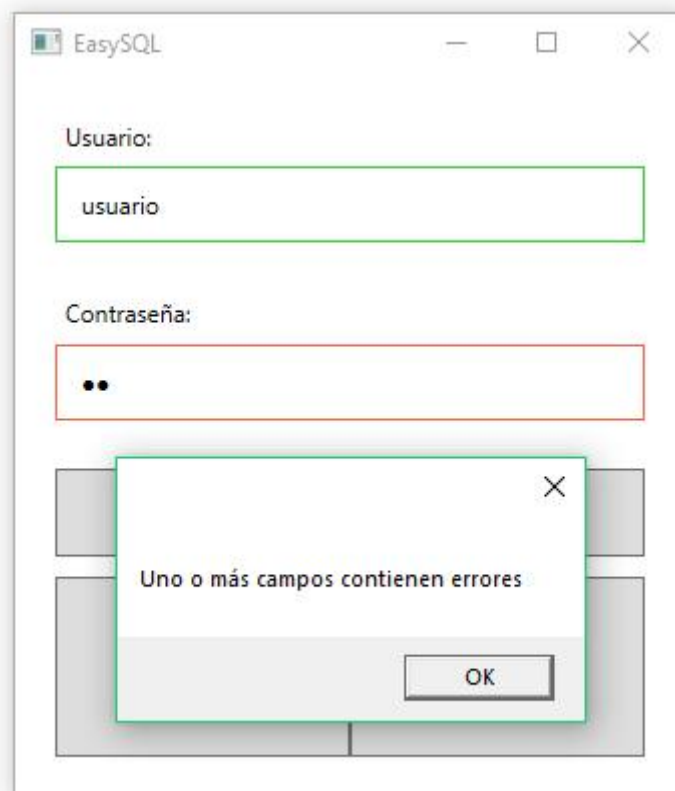


Imagen 16. Validación visual.

En la siguiente captura pantalla del software se pueden apreciar varios detalles. Por una parte, la comentada anterior validación visual. Por otro, podemos ver que al marcar el campo "Integrated Security", una opción de autenticación para bases de datos SQL Server, inhabilita los campos usuario, contraseña y tipo de conexión, ya que todos ellos ya lo gestiona de manera automática la aplicación sin que el usuario haga nada. Por tanto, elimina la posibilidad de introducción de datos erróneos por parte del usuario.

Como mención especial, el valor del campo "Puerto" no se ha introducido, lo asigna por defecto el software teniendo en cuenta el tipo de conexión elegida.

Conexiones || Conectado usuario: contoso con ID: 7

Conexión actual: Mi conexión SQL

Lista Conexiones

Dirección	Usuario	Puerto
localhost\SQLE	Integrated S	1433

* Nombre de la conexión: Mi conexión SQL

* Dirección: localhost\SQLE

Puerto: 1433

* Usuario: Integrated Security

Contraseña:

Guardar contraseña: ☐

* Tipo de conexión: ☒ Microsoft SQL ☒ Integrated Security ☐ MySQL

* Campos marcados con un asterisco deben ser rellenados.

Ordenar - Creación | Ordenar - Nombre | Actualizar | Borrar | Guardar | Limpiar | Cancelar | Test | Conectar

Imagen 17. Integrated Security y puerto por defecto.

A continuación, se pasa a explicar una característica que para el usuario final puede no tener demasiada importancia, pero a la hora del desarrollo de esta fue una de las partes con mayor dificultad de realizar.

Existe una característica con las bases de datos SQL Server y las aplicaciones .NET. Para agilizar y mejorar el rendimiento de las consultas, la primera vez que el software realiza una consulta contra una base de datos SQL Server, de manera transparente y automática mantiene la conexión abierta y un flujo de comprobación de estado constante corriendo. Esto provocaba que a la hora de intentar eliminar una instancia de base de datos, y en caso de haber realizado una consulta previamente contra ella, no se pudiese efectuar la operación debido a que existía una conexión abierta con ella. Esto llevó a la idea de una alerta para el usuario que mostrase un error y la pregunta de si se deseaba forzar el borrado. Al hacerlo, la aplicación manda unos comandos a la BBDD obligarla a cerrar las conexiones que estén activas en el momento con todos los usuarios. Después de eso, acaba borrando la BBDD de manera efectiva.

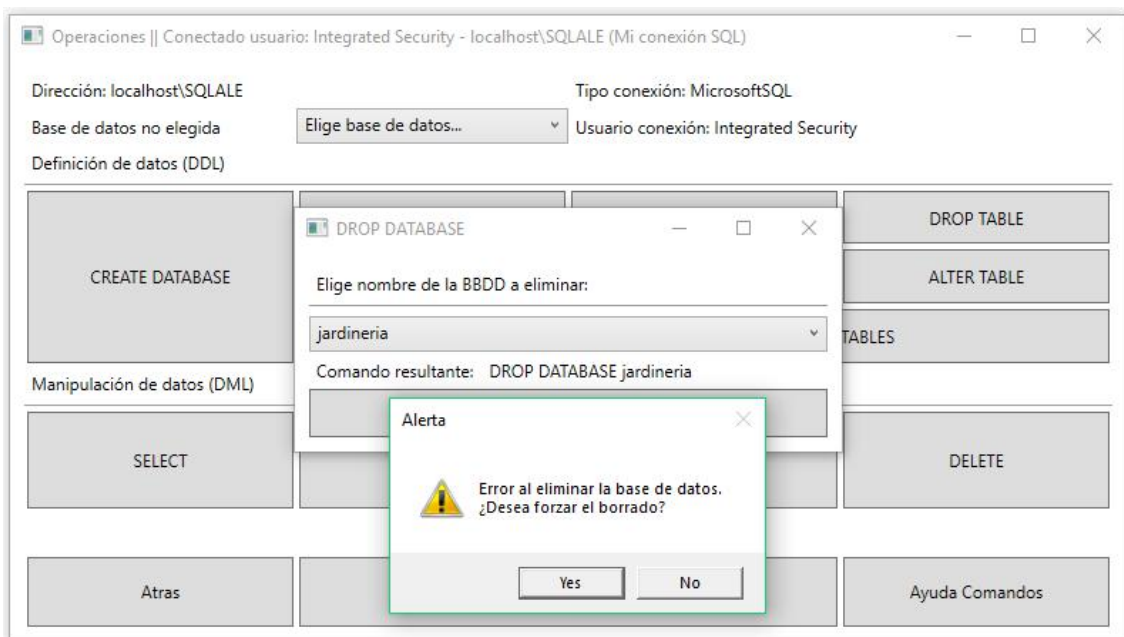


Imagen 18. Forzado de borrado base de datos.

Como dato añadido, esto no ocurre en las bases de datos MySQL.

Ahora se expone un caso de uso y prueba. Una de las funcionalidades que tiene la aplicación es la de poder actualizar datos existentes en la BBDD. Esto se hace mediante el comando UPDATE.

Aquí se muestra un ejemplo en el que, a remarcar, informa al usuario de la cantidad de filas afectadas por el uso de la consulta.

UPDATE || Base de datos "jardineria"

Elige tabla de la que eliminar datos:

producto

Elige campos a actualizar:

<input type="checkbox"/>	Nombre columna:	Tipo dato:	Valor:
<input type="checkbox"/>	id	int	
<input type="checkbox"/>	nombre	nvarchar	
<input checked="" type="checkbox"/>	precio	float	5

Elige cantidad condiciones WHERE: 1

Nombre columna:	Operador:	Valor:
nombre	LIKE	'm%'

3 filas de la tabla "producto" en base de datos "jardineria" modificadas con éxito.

OK

Comando resultante: UPDATE producto SET precio = 5 WHERE nombre LIKE 'm%'

Ejecutar comando

Imagen 19. Update e información de filas afectadas.

8. MANUAL USUARIO

Este apartado de la Memoria pretende ser un resumen de todo lo que el usuario se va a encontrar a la hora de ejecutar la aplicación, y un sumario de las funcionalidades que ofrece y la manera de utilizarla.

8.1. VENTANA INICIO

Es la primera ventana que se va a encontrar el usuario a la hora de ejecutar la aplicación. Esta permite logear con unas credenciales a un usuario, para acceder a la Ventana de Conexiones en modo Usuario. También permite acceder la Ventana de Registro con el botón de la esquina inferior derecha. En caso de que no se quisiese logear, puede utilizar el software en modo Invitado.

The image shows a screenshot of a Windows application window titled "EasySQL". The window has a standard title bar with minimize, maximize, and close buttons. The main content area contains a login form. It starts with a label "Usuario:" followed by a text input field. Below that is a label "Contraseña:" followed by a password input field. Under the password field is a wide, grey button labeled "Acceder". At the bottom of the window, there are two smaller, grey buttons side-by-side: "Acceder como invitado" on the left and "Registrarse" on the right.

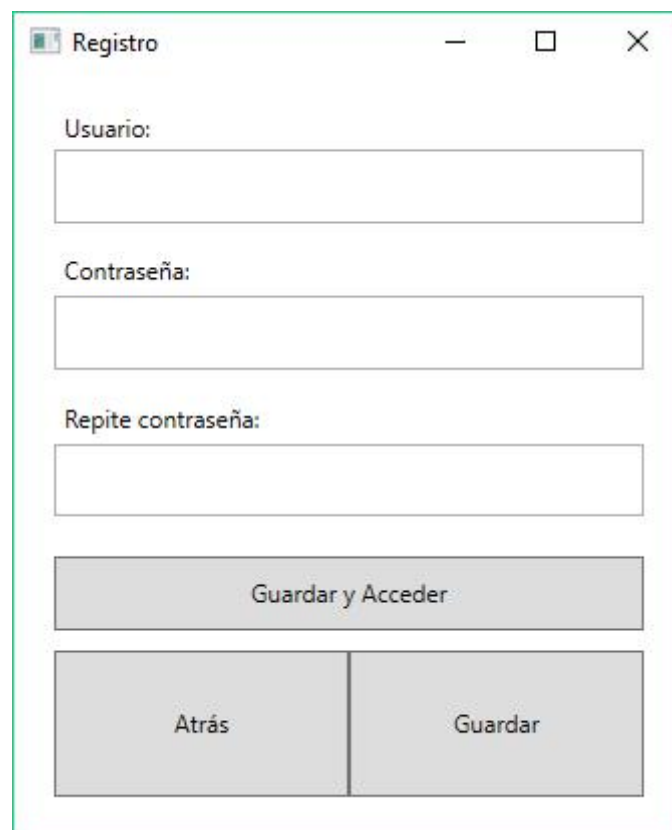
Imagen 20. Ventana Inicio.

8.2. VENTANA REGISTRO

En esta ventana un usuario puede introducir unas credenciales con las que se autenticará en el futuro para usar la aplicación en modo Usuario. Esto lo hará por medio de la introducción de datos en los campos de Usuario y Contraseña.

A la hora de introducir los datos, el programa mostrará con colores rojo y verde si los datos introducidos son correctos.

Podemos añadir varios usuarios a la vez, uno detrás de otro haciendo uso del botón Guardar, o acceder directamente con los campos introducidos sin tener que volver a pasar por la Ventana de Inicio para logearse.



The image shows a software window titled "Registro". Inside the window, there are three text input fields arranged vertically. The first field is labeled "Usuario:", the second "Contraseña:", and the third "Repita contraseña:". Below the input fields, there are three buttons. The first button, "Guardar y Acceder", is a single wide button. Below it are two buttons side-by-side: "Atrás" on the left and "Guardar" on the right. The window has a standard title bar with minimize, maximize, and close buttons.

Imagen 21. Ventana Registro.

8.3. VENTANA CONEXIONES (MODO INVITADO)

Esta es la Ventana de Conexiones en modo Invitado, tiene restringidas las funciones de almacenamiento de datos de conexiones para un usuario en concreto. Estas características se inhabilitan de manera visual al usuario para que no pueda interactuar con los controles de la interfaz gráfica destinados a esa funcionalidad.

Permite introducir datos referentes con una conexión de base de datos y acceder a la Ventana Operaciones, o realizar un Test de conexión para saber si los datos introducidos son correctos.

Conexiones || Conectado como invitado

Conexión actual:

Lista Conexiones

Nombre	Dirección	Tipo Conexión	Usuario	Puerto
--------	-----------	---------------	---------	--------

Ordenar - Creación Actualizar Borrar

Ordenar - Nombre

* Nombre de la conexión:

* Dirección:

Puerto:

* Usuario:

Contraseña:

Guardar contraseña: ☐

* Tipo de conexión: ☒ Microsoft SQL ☐ Integrated Security ☐ MySQL

* Campos marcados con un asterisco deben ser rellenos.

Guardar Limpiar Cancelar Test Conectar

Imagen 22. Ventana Conexiones modo Invitado.

8.4. VENTANA CONEXIONES (MODO USUARIO)

Ahora se muestra la misma Ventana de Conexiones, esta vez habiendo accedido a ella a través del logeo con credenciales de la aplicación. Ahora permite hacer uso de todos los controles visuales de la interfaz gráfica.

Permite a un usuario almacenar direcciones correspondientes con bases de datos y se asocian automáticamente a su cuenta. Estas direcciones se muestran en un panel de Lista de Conexiones en la parte izquierda de la pantalla. Al hacer clic sobre uno de los items de la lista se actualizan los campos de la derecha, reflejando la información de esta. Los elementos de la lista se pueden ordenar de manera alfabética o por fecha de creación, así como eliminarlos.

Conexión actual:

Lista Conexiones

Nombre	Dirección	Tipo Conexión
casa	localhost\SQLALE	MicrosoftSQL
pruebas	62.123.22.14\MySt	MySQL
jardineri	www.miservidor.es	MySQL

* Nombre de la conexión: jardineria

* Dirección: www.miservidor.es\basedatos

Puerto:

* Usuario: jardinero

Contraseña:

Guardar contraseña: ☐

* Tipo de conexión: ☐ Microsoft SQL ☐ Integrated Security ☒ MySQL

* Campos marcados con un asterisco deben ser rellenados.

Ordenar - Creación Actualizar Borrar Guardar Limpiar Cancelar Test Conectar

Ordenar - Nombre

Imagen 23. Ventana Conexiones modo Usuario.

Una funcionalidad añadida es la del campo Puerto. En caso de no ser provisto de ningún número, o este ser 0, el programa buscará en la base de datos el número correspondiente al puerto por defecto en ese tipo de Base de Datos y se lo asignará automáticamente.

8.5. VENTANA OPERACIONES

Esta ventana está diseñada para servir de puerta de entrada a todas las funciones de consulta y modificación de datos y estructuras contra la base de datos que nos estamos conectando.

Permite elegir la instancia de la base de datos con la que trabajaremos, y con ella elegida, nos permitirá con el uso de unos grandes botones, organizados por el tipo de comando que se quiera ejecutar, y separados en dos grandes campos (Definición de datos, DDL y Manipulación de datos, DML), realizar las tareas de gestión y las operaciones que el usuario estime conveniente.

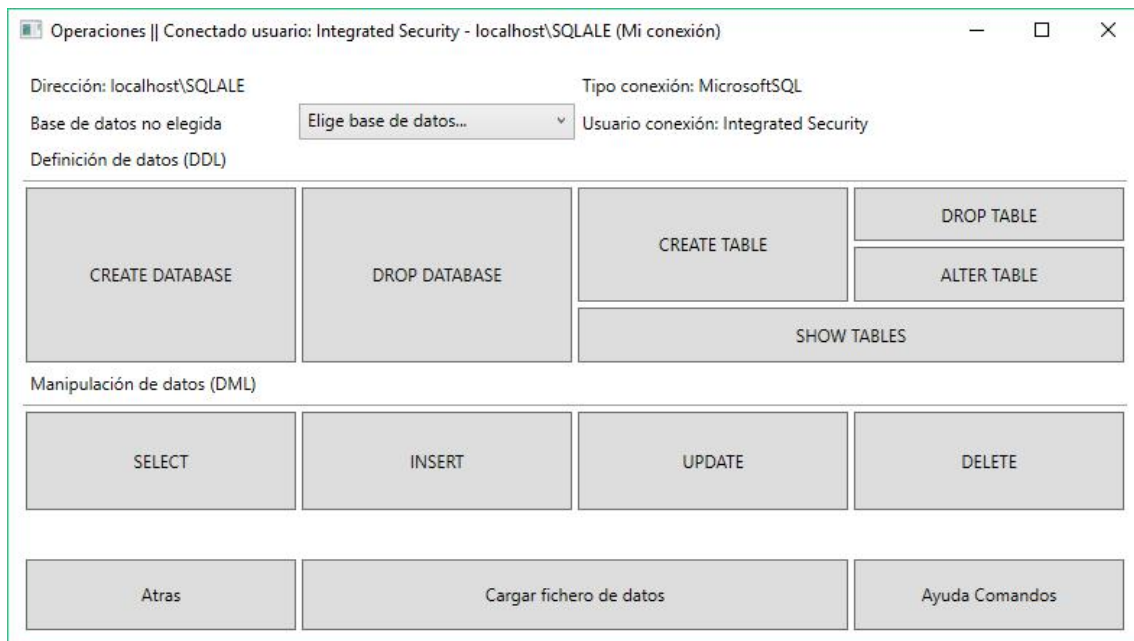


Imagen 24. Ventana Operaciones.

También permite, en la parte inferior de la pantalla, cargar un fichero de datos *.easy que haya sido previamente almacenado en algún dispositivo de memoria no volátil con la Ventana de Resultado Datos. También podremos abrir una conexión Web que llevará al usuario a una página de referencia para comandos SQL, utilizando el botón Ayuda situado en la parte inferior derecha.

8.6. VENTANA CREATE DATABASE

Esta ventana permite al usuario crear una nueva instancia de base de datos. Para ello este sólo necesita introducir el nombre que va a tener la instancia en el input de datos y pulsar el botón inferior.



Imagen 25. Ventana Create Database.

8.7. VENTANA DROP (DATABASE / TABLE)

Esta ventana, que se reutiliza para dos funciones similares, permite dos modos, eliminar una base de datos o tabla de la base de datos. Para ello el usuario seleccionará de la lista desplegable el nombre de la tabla o base de datos que desee eliminar, y al pulsar el botón enviará el comando al servidor.

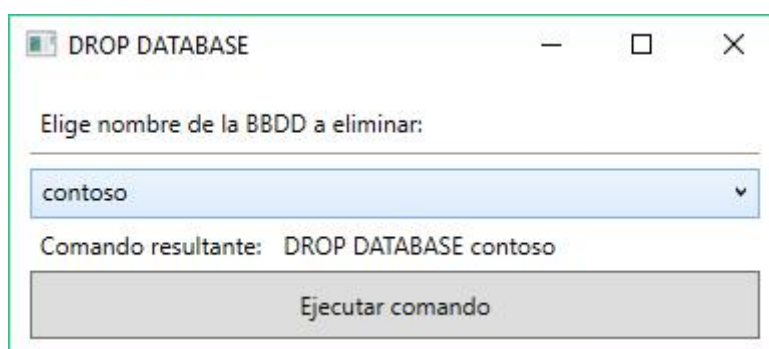


Imagen 26. Ventana Drop Database.

8.8. VENTANA CREATE TABLE

Permite crear una nueva tabla dentro de la instancia de base de datos al usuario. En la parte de arriba, éste debe asignar el nombre que tendrá la tabla. Luego, elegir el número de campos (columnas) que esta va a tener.

Una vez elegido, se generarán de manera automática 3 controles de interfaz gráfica que permitirán al usuario decidir si cada columna será primary key, el nombre de esta y el tipo de dato que se asignará. Una vez comprobado que todos los datos sean correctos, se ejecutará el comando.

CREATE TABLE || Base de datos "contoso"

Introduce nombre de la tabla a crear:

producto

Elige cantidad campos: 3

PK:	Nombre de la columna:	Tipo de dato:
<input checked="" type="checkbox"/>	id	INT
<input checked="" type="checkbox"/>	nombre	NVARCHAR(50)
<input checked="" type="checkbox"/>	precio	FLOAT

Comando resultante: CREATE TABLE producto (id INT, nombre NVARCH

Ejecutar Comando

Imagen 27. Ventana Create Table.

8.9. VENTANA SHOW TABLES

Esta ventana permite, de un vistazo rápido, observar una lista con los nombres de todas las tablas que existan en la instancia de la base de datos previamente elegida en la Ventana Operaciones.

Al pulsar sobre el botón "Ver Datos" que tiene cada columna en su lado izquierdo, se ejecutará un comando `SELECT * FROM (nombre tabla)` que permitirá de manera rápida echar un vistazo a los datos que contenga la tabla. Estos datos se muestran a través de una ventana Resultado Datos.

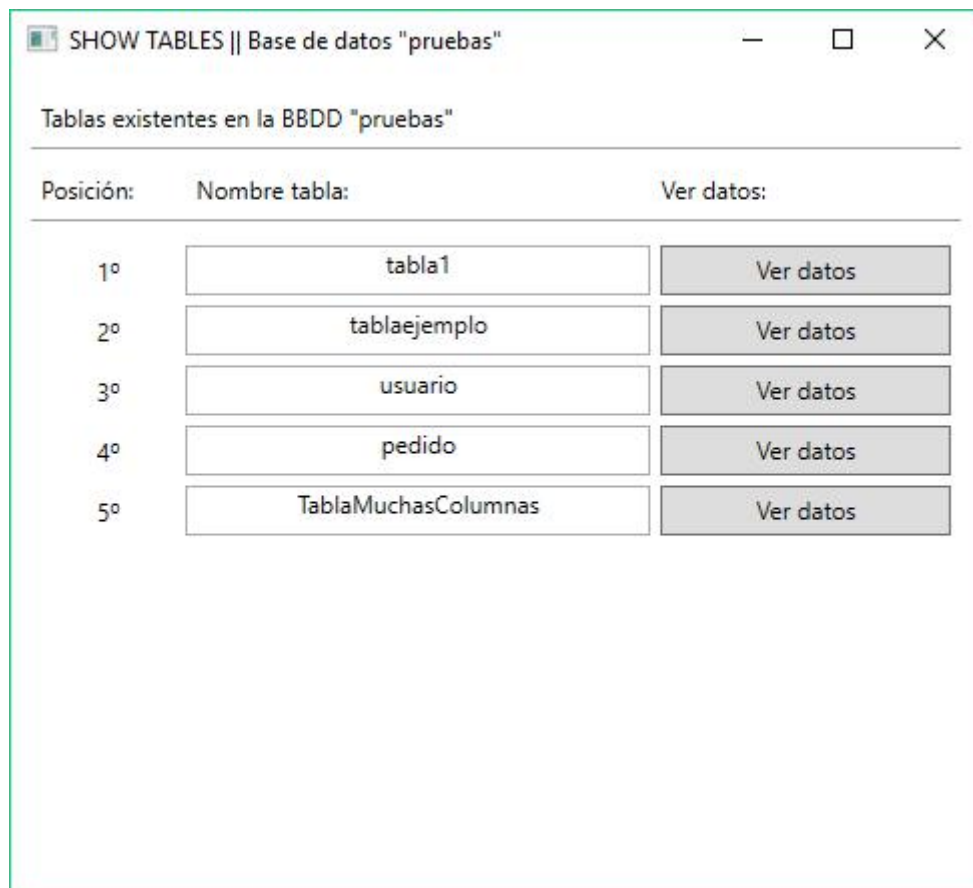
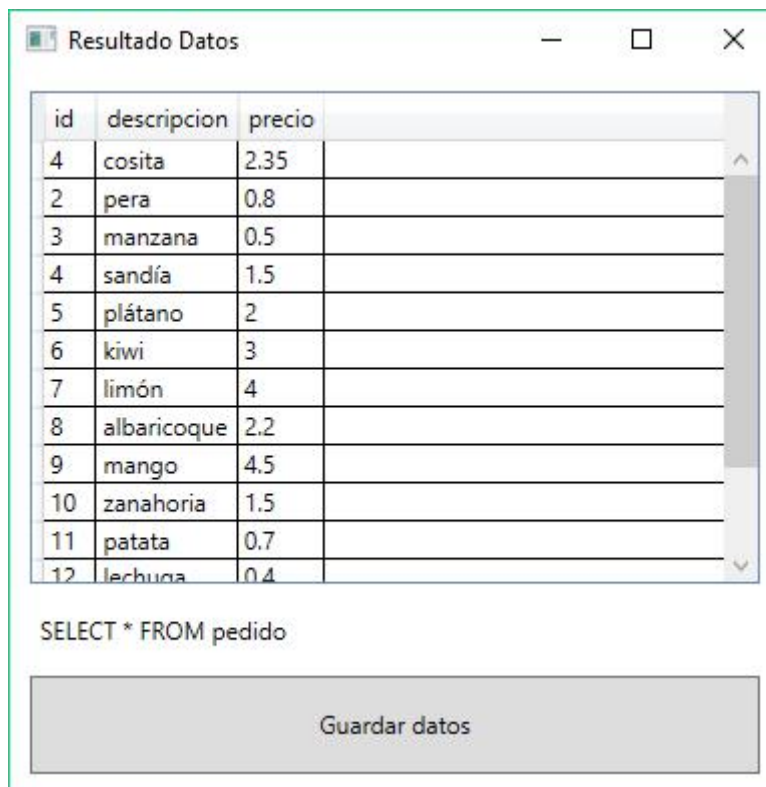


Imagen 28. Ventana Show Tables.

8.10. VENTANA RESULTADO DATOS

Esta es la ventana utilizada para representar las consultas SELECT al usuario. Consta de un control tipo DataTable que se popula de manera dinámica con los datos de las filas correspondientes a la consulta ejecutada. En la parte superior de la tabla podemos observar que se reflejan los nombres de las columnas, tal cual como están almacenados en la base de datos. Abajo de ellos se muestran los datos de manera ordenada.



The screenshot shows a window titled "Resultado Datos" with a table of data and a SQL query below it. The table has four columns: "id", "descripcion", "precio", and an empty column. The data rows are as follows:

id	descripcion	precio	
4	cosita	2.35	
2	pera	0.8	
3	manzana	0.5	
4	sandía	1.5	
5	plátano	2	
6	kiwi	3	
7	limón	4	
8	albaricoque	2.2	
9	mango	4.5	
10	zanahoria	1.5	
11	patata	0.7	
12	lechuga	0.4	

Below the table, the SQL query "SELECT * FROM pedido" is displayed. At the bottom of the window is a button labeled "Guardar datos".

Imagen 29. Ventana Resultado Datos.

8.11. VENTANA ALTER TABLE

Esta ventana permite ejecutar el comando ALTER TABLE para modificar una tabla de una instancia de base de datos seleccionada. El usuario deberá al comienzo seleccionar desde la lista desplegable un nombre de tabla válido que desee modificar. Luego de ello, en la segunda lista desplegable, elegirá el tipo de operación a realizar.

Permite tanto la creación de nuevas columnas para insertarlas en la tabla como eliminar alguna columna ya existente. Según la opción elegida en la segunda lista desplegable, la ventana que se está mostrando se modificará de manera dinámica para hacer aparecer los controles de interfaz gráfica necesarios para llevar a cabo la operación.

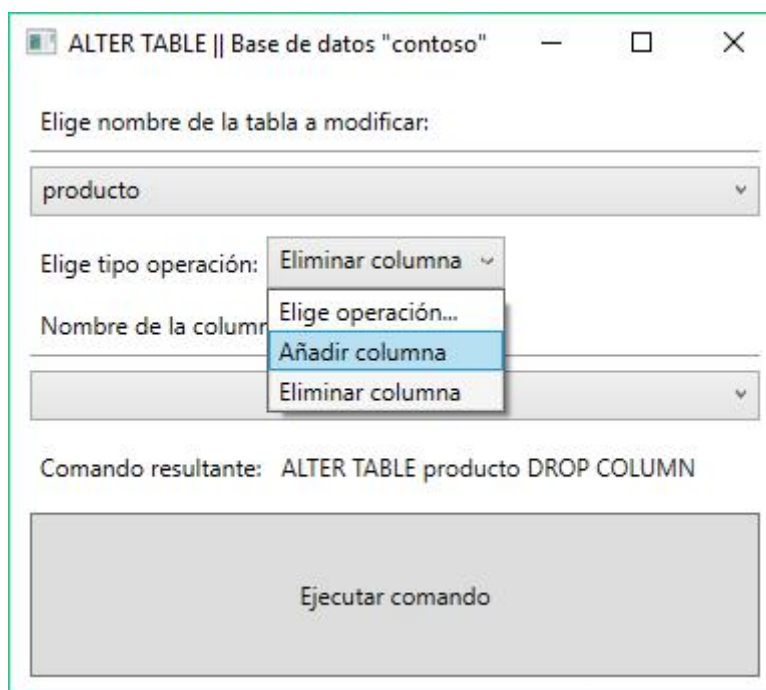


Imagen 30. Ventana Alter Table.

8.12. VENTANA SELECT

Permite al usuario realizar una consulta de datos existente en la base de datos. Primero se elige el nombre de la tabla a consultar, y luego se añade cuantas condiciones WHERE se estimen. Por último, se fija un orden de los datos con las cláusulas ORDER BY.

SELECT || Base de datos "pruebas"

Elige tabla de la que mostrar datos:

TablaMuchasColumnas

Elige campos a mostrar:

☐ Nombre columna: Tipo dato:

<input checked="" type="checkbox"/>	dinero	bigint
<input checked="" type="checkbox"/>	nombre	nvarchar
<input type="checkbox"/>	apellido	nvarchar

Elige cantidad condiciones WHERE: 2

Nombre columna:	Operador:	Valor:
dinero	>	5
AND		
apellido	LIKE	'%dez'

Elige cantidad cláusulas ORDER BY: 1

Posición:	Columna:	Sentido:
1º	nombre	ASC

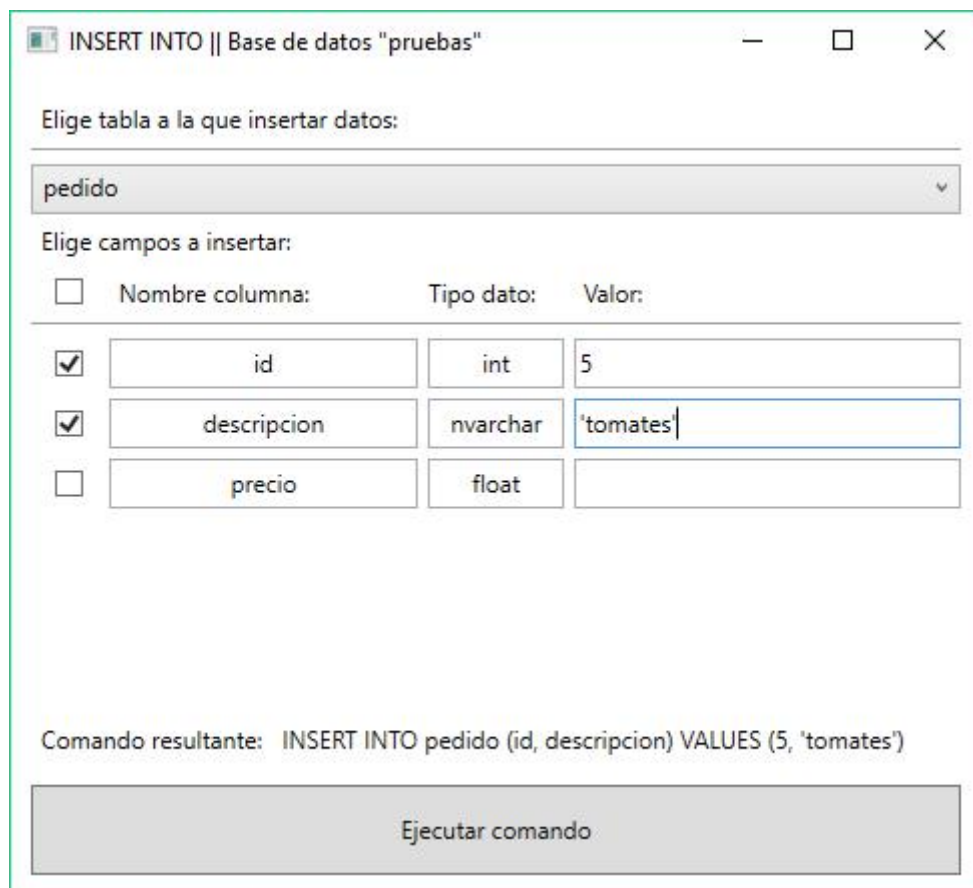
Comando resultante: SELECT dinero, nombre FROM TablaMuchasColumnas WHERE d

Ejecutar comando

Imagen 31. Ventana Select.

8.13. VENTANA INSERT

Con la ayuda de esta ventana, podremos hacer uso de la inserción de datos en nuestra base de datos. Primero, el usuario debe elegir una tabla válida desde la lista desplegable de la parte superior de la ventana. Una vez se elija, la aplicación generará de manera automática tantos campos como columnas tuviese esa tabla, a parte de mostrar el tipo de dato que contiene para ayudar al usuario a la hora de insertar los datos. Una vez el usuario ha marcado las columnas que desee insertar y le asigne un valor apropiado en el campo destinado para ello, podrá hacer uso del botón ubicado en la parte inferior de esta ventana para lanzar la orden a ejecutar contra la base de datos.



INSERT INTO || Base de datos "pruebas"

Elige tabla a la que insertar datos:

pedido

Elige campos a insertar:

<input type="checkbox"/>	Nombre columna:	Tipo dato:	Valor:
<input checked="" type="checkbox"/>	id	int	5
<input checked="" type="checkbox"/>	descripcion	nvarchar	'tomates'
<input type="checkbox"/>	precio	float	

Comando resultante: INSERT INTO pedido (id, descripcion) VALUES (5, 'tomates')

Ejecutar comando

Imagen 32. Ventana Insert.

8.14. VENTANA UPDATE

Con esta ventana, se pueden realizar actualizaciones de datos ya existentes en tablas de dos modos. Asignando un valor a una o varias columnas de manera indiscriminada a todas las filas, o utilizando condiciones WHERE para realizar la actualización de datos de una manera más selectiva.

UPDATE || Base de datos "pruebas"

Elige tabla de la que eliminar datos:

pedido

Elige campos a actualizar:

<input type="checkbox"/>	Nombre columna:	Tipo dato:	Valor:
<input type="checkbox"/>	id	int	
<input checked="" type="checkbox"/>	descripcion	nvarchar	'pera'
<input type="checkbox"/>	precio	float	

Elige cantidad condiciones WHERE: 3

Nombre columna:	Operador:	Valor:
id	=	5
OR		
precio	IS NULL	
OR		
descripcion	LIKE	'm%'

Comando resultante: UPDATE pedido SET descripcion = 'pera' WHERE id = 5 OR prec

Ejecutar comando

Imagen 33. Ventana Update.

8.15. VENTANA DELETE

Para concluir con esta Memoria de proyecto, se muestra de las ventanas que faltaban por dejar plasmada en este documento.

La ventana DELETE FROM permite al usuario borrar filas de datos de una tabla ya existente. Primero éste debe seleccionar una tabla de la lista desplegable de la parte de arriba de la pantalla. Luego, si desea hacer un borrado selectivo de algunas filas, introducirá las clausulas WHERE correspondientes en la parte media de la ventana. En caso contrario, se eliminarán todos los datos que contengan las filas de la tabla.

DELETE FROM || Base de datos "pruebas"

Elige tabla de la que eliminar datos:

pedido

Elige cantidad condiciones WHERE: 2

Nombre columna: Operador: Valor:

precio < 4

AND

id > 15

Comando resultante: DELETE FROM pedido WHERE precio < 4 AND id > 15

Ejecutar comando

Imagen 34. Ventana Delete.

9. BIBLIOGRAFÍA

Las siguientes páginas web se han utilizado a la hora de realizar el proyecto y los análisis y estudios necesarios para esta Memoria.

- <https://msdn.microsoft.com/es-es/library/>
- <https://docs.microsoft.com/en-us/sql/?view=sql-server-2017>
- <https://docs.microsoft.com/en-us/dotnet/framework/wpf/>
- <https://dev.mysql.com/doc/>
- <https://stackoverflow.com/>
- <https://es.wikipedia.org/>

Los siguientes enlaces corresponden a las fuentes de donde se han tomado algunas de las imágenes que han acompañado esta Memoria.

- 1.- <http://bigdatahadooppro.com/sql-vs-nosql-all-you-need-to-know/>
- 2.- <http://learn-php-online.top5reviews.com/php-and-mysql-vs-asp-net-and-sql-server/>
- 3.- https://www.mysql.com/common/images/products/MySQL_Workbench_Editor_General_Mac.png
- 4.- <http://social.msdn.microsoft.com/Forums/getfile/272678>
- 5.- https://cdn.macworld.co.uk/cmsdata/features/3638150/setup_learn_sql_mac_thumb800.jpg
- 6.- https://studyguide.itu.dk/~media/studyguide/student-life/facilities-at-itu/it-facilities/github/github_logo.png
- 7.- https://upload.wikimedia.org/wikipedia/commons/thumb/6/61/Visual_Studio_2017-logo_and_wordmark.svg/2000px-Visual_Studio_2017_logo_and_wordmark.svg.png
- 8.- <https://javiersuarezruiz.files.wordpress.com/2018/01/wpf-logo.png>
- 9.- http://www.palermo.edu/Archivos_content/ingenieria/top/130712_git_github_topdenota1.jpg
- 10.- <https://i.stack.imgur.com/PNYHA.png>
- 11.- <http://3.bp.blogspot.com/-iCTBNggzQtU/VJxJcvKAKrI/AAAAAAAAAHc/2Vnwfs9bmw/s1600/client-server-illustration.gif> (traducción propia)
- 12.- <http://codingornot.com/wp-content/uploads/2017/10/mvc-modelo-vista-controlador.png>

10. ÍNDICE DE ELEMENTOS

10.1. ÍNDICE DE IMÁGENES

Imagen 1 . Logotipo de EasySQL.....	3
Imagen 2 . Comparación proporción de uso actual entre diferentes tecnologías.....	6
Imagen 3 . Las dos principales marcas dominantes del mercado.....	7
Imagen 4 . Interfaz de usuario de MySQL WorkBench (MacOs).....	8
Imagen 5 . Interfaz de usuario de SQL Server Management Studio (Windows).....	9
Imagen 6 . Logotipo de SQL.....	10
Imagen 7 . Logotipo de GitHub.....	12
Imagen 8 . Logo de la aplicación Visual Studio.....	14
Imagen 9 . Logo de la tecnología WPF.....	14
Imagen 10 . Logo de la tecnología Git y la plataforma GitHub.....	15
Imagen 11 . Anatomía del hash devuelto por el algoritmo BCrypt.....	16
Imagen 12 . Arquitectura Cliente - Servidor.....	22
Imagen 13 . Colaboración típica entre componentes en un patrón MVC.....	22
Imagen 14 . Diagrama Entidad Relación de la BBDD.....	33
Imagen 15 . Aviso de usuario ya registrado.....	35
Imagen 16 . Validación visual.....	36
Imagen 17 . Integrated Security y puerto por defecto.....	37
Imagen 18 . Forzado de borrado base de datos.....	38
Imagen 19 . Update e información de filas afectadas.....	39
Imagen 20 . Ventana Inicio.....	40
Imagen 21 . Ventana Registro.....	41
Imagen 22 . Ventana Conexiones modo Invitado.....	42
Imagen 23 . Ventana Conexiones modo Usuario.....	43

Imagen 24 . Ventana Operaciones.....	44
Imagen 25 . Ventana Create Database.....	45
Imagen 26 . Ventana Drop Database.....	45
Imagen 27 . Ventana Create Table.....	46
Imagen 28 . Ventana Show Tables.....	47
Imagen 29 . Ventana Resultado Datos.....	48
Imagen 30 . Ventana Alter Table.....	49
Imagen 31 . Ventana Select.....	50
Imagen 32 . Ventana Insert.....	51
Imagen 33 . Ventana Update.....	52
Imagen 34 . Ventana Delete.....	53

10.2. ÍNDICE DE UML

UML 1 . Modelos de datos.....	23
UML 2 . Manejo base de datos del programa.....	24
UML 3 . Operaciones de manejo y consulta contra Bases de Datos externas.....	25
UML 4 . Clases del paquete Utils de código común.....	26
UML 5 . Librería BCrypt.....	26
UML 6 . Ventanas iniciales.....	27
UML 7 . Ventanas Conexión y Operaciones.....	28
UML 8 . Ventanas de comandos I.....	29
UML 9 . Ventana comandos II.....	30
UML 10 . Ventanas comandos III.....	31
UML 11 . Ventanas comandos IV y Común.cs.....	32
UML 12 . BBDD del programa.....	32

ANEXO I.- MANUAL DEL CÓDIGO

En este apartado están incluidas el código C# de las clases más importantes de la aplicación. Estas son: clase de dedicada al manejo de la base de datos interna del programa, clase que devuelve comandos SQL bien formados (implementación de SQL Server), clase que ejecuta comandos contra base de datos externas (SQL Server) y la clase Común donde se encuentran los métodos de generación de campos para las ventanas de manera dinámica.

Existen en total +50 clases C#, con un total aproximado de 7,500 líneas de código. Plasmarlo todo aquí haría de esta memoria un documento exageradamente largo, es por ello que muestro sólo 3 de las más relevantes.

```
public class BBDDProgramaImpl
{
    /// <summary>
    /// Cadena de conexión con los datos necesarios para la conexión con la BBDD del programa.
    /// </summary>
    private string cadenaConexion;

    /**
     * Declaraciones para las Query que se van a ejecutar
     */
    private string registrarUsuarioQuery, registrarConexionQuery, existirQuery, loginQuery, saltQuery, obtenerIDUsuarioQuery,
        obtenerIDConexionQuery, obtenerConexionesQuery, eliminarConexionQuery, obtenerPuertoQuery;

    /// <summary>
    /// Construye un objeto BBDDProgramaImpl, asigna la cadena de conexión y las query.
    /// </summary>
    private BBDDProgramaImpl()
    {
        // 1. Crea la cadena de conexión.
        cadenaConexion =
            "Data Source=localhost\\SQLALE;"
            + "Initial Catalog=usuarios;"
            + "Integrated Security=True";

        // 2. Crea las query.
        loginQuery = "SELECT nombre FROM Usuario WHERE nombre =@usuario " +
            "AND contrasenia =@contrasenia";
        saltQuery = "SELECT contrasenia_salt FROM Usuario WHERE nombre =@usuario ";
        registrarUsuarioQuery = "INSERT INTO Usuario (nombre, contrasenia, contrasenia_salt) VALUES " +
            "@usuario,@contrasenia,@contrasenia_salt";
        registrarConexionQuery = "INSERT INTO Conexion (id_tipo_conexion, id_usuario, nombre, direccion, puerto, usuario, contrasenia)"
+
            " VALUES (@tipo_conexion, @id_usuario, @nombre, @direccion, @puerto, @usuario, @contrasenia)";
        existirQuery = "SELECT nombre FROM Usuario WHERE nombre =@usuario";
        obtenerIDUsuarioQuery = "SELECT id_usuario FROM Usuario WHERE nombre=@usuario AND contrasenia=@contrasenia";
        obtenerIDConexionQuery = "SELECT id_conexion FROM Conexion WHERE nombre=@nombredir " +
            "AND direccion=@direccion AND usuario=@usuario";
        obtenerConexionesQuery = "SELECT id_conexion, id_tipo_conexion, nombre, direccion, puerto, usuario, contrasenia " +
            "FROM conexion WHERE id_usuario = @id_usuario";
        eliminarConexionQuery = "DELETE FROM Conexion WHERE id_conexion = @id_conexion";
        obtenerPuertoQuery = "SELECT puerto_defecto FROM tipo_conexion WHERE id_tipo = @id_tipo";
    }

    /// <summary>
    /// Usada para almacenar una instancia de esta clase.
    /// </summary>
    private static BBDDProgramaImpl instancia;

    /// <summary>
    /// Devuelve una única instancia de esta clase siguiendo el patrón Singleton.
    /// </summary>
    /// <returns>Una instancia bien construida de la clase.</returns>
    public static BBDDProgramaImpl ObtenerInstancia()
    {
        if (instancia == null)
        {
```

```

        instancia = new BBDDProgramaImpl();
    }
    return instancia;
}

/// <summary>
/// Implementación de la lógica de Login.
/// </summary>
public ResultadoLogin LoginUsuario(string usuario, string contrasenia)
{
    string contraseniaSalt = ObtenerSal(usuario);
    string contraseniaBcrypt = "";
    if (!String.IsNullOrEmpty(contraseniaSalt))
        contraseniaBcrypt = BCrypt.HashPassword(contrasenia, contraseniaSalt);
    SqlCommand loginCmd = new SqlCommand(loginQuery);
    loginCmd.Parameters.AddWithValue("@usuario", usuario);
    loginCmd.Parameters.AddWithValue("@contrasenia", contraseniaBcrypt);

    // Obtiene el resultado
    object resultado = OperacionSQL.ExecuteScalar(cadenaConexion, loginCmd);

    // Si el resultado es nulo, no existe el usuario.
    if (resultado == null)
    {
        return new ResultadoLogin(ResultadoLogin.TipoResultado.DENEGADO, null);
    }
    else if (resultado.Equals(Operacion.ERROR))
    {
        return new ResultadoLogin(ResultadoLogin.TipoResultado.ERROR, null);
    }
    {
        // Se le asigna la ID de la base de datos al usuario creado.
        Usuario creado = new Usuario(usuario, contrasenia, contraseniaBcrypt);
        creado.ID = BBDDPrograma.ObtenerIDUsuario(creado);
        return new ResultadoLogin(ResultadoLogin.TipoResultado.ACEPTADO, creado);
    }
}

/// <summary>
/// Obtiene la Sal almacenada en BBDD de un usuario dado.
/// </summary>
/// <param name="usuario">El usuario del que obtener la sal.</param>
/// <returns>String con la sal de la contraseña del usuario.</returns>
public string ObtenerSal(string usuario)
{
    SqlCommand saltCmd = new SqlCommand(saltQuery);
    saltCmd.Parameters.AddWithValue("@usuario", usuario);
    // Obtiene el resultado
    object resultado = OperacionSQL.ExecuteScalar(cadenaConexion, saltCmd);
    if (resultado != null)
        return resultado.ToString();
    else
        return "";
}

/// <summary>
/// Implementación de la lógica de Registro.
/// </summary>
public ResultadoRegistro RegistrarUsuario(string usuario, string contrasenia)
{
    if (ExisteUsuario(usuario))
    {
        return new ResultadoRegistro(ResultadoRegistro.TipoResultado.DUPLICADO, null);
    }
    else
    {
        // Generar sal
        string contraseniaSalt = BCrypt.GenerateSalt();
        //mySalt == "$2a$10$rBV2JDwW3.vKyeQcM8fFO"
        string contraseniaBcrypt = BCrypt.HashPassword(contrasenia, contraseniaSalt);
        //myHash == "$2a$10$rBV2JDwW3.vKyeQcM8fFO477714bVeQgDL6VikxqlzQ7TCa1Qv1a"
        bool doesPasswordMatch = BCrypt.CheckPassword(contrasenia, contraseniaBcrypt);

        SqlCommand registrarCmd = new SqlCommand(registrarUsuarioQuery);
        registrarCmd.Parameters.AddWithValue("@usuario", usuario);
        registrarCmd.Parameters.AddWithValue("@contrasenia", contraseniaBcrypt);
        registrarCmd.Parameters.AddWithValue("@contrasenia_salt", contraseniaSalt);

        // Obtiene el resultado
        int resultadoFilasSQL = OperacionSQL.ExecuteNonQuery(cadenaConexion, registrarCmd);

        // Si es distinto mayor a 0, se habrá registrado el usuario
        if (resultadoFilasSQL > 0)
        {
            Usuario creado = new Usuario(usuario, contrasenia, contraseniaBcrypt);
            creado.ID = BBDDPrograma.ObtenerIDUsuario(creado);
            return new ResultadoRegistro(ResultadoRegistro.TipoResultado.ACEPTADO, creado);
        }
        else
        {
            return new ResultadoRegistro(ResultadoRegistro.TipoResultado.ERROR_CONEXION, null);
        }
    }
}

```

```

    }
}

/// <summary>
/// Implementación de la lógica de ObtenerIDUsuario.
/// </summary>
public int ObtenerIDUsuario(Usuario usuario)
{
    // Crea el comando
    SqlCommand obtenerCmd = new SqlCommand(obtenerIDUsuarioQuery);
    obtenerCmd.Parameters.AddWithValue("@usuario", usuario.Nombre);
    obtenerCmd.Parameters.AddWithValue("@contrasenia", usuario.ContraseniaBBDD);
    // Obtiene el resultado
    object resultado = OperacionSQL.ExecuteScalar(cadenaConexion, obtenerCmd);
    // Si el resultado es nulo, no existe el usuario.
    if (resultado == null)
    {
        return -1;
    }
    else
    {
        return (int) resultado;
    }
}

/// <summary>
/// Implementación de la lógica de ObtenerIDConexion.
/// </summary>
public int ObtenerIDConexion(Conexion conexion)
{
    // Crea el comando
    SqlCommand obtenerIDCmd = new SqlCommand(obtenerIDConexionQuery);
    obtenerIDCmd.Parameters.AddWithValue("@nombredir", conexion.Nombre);
    obtenerIDCmd.Parameters.AddWithValue("@direccion", conexion.Direccion);
    obtenerIDCmd.Parameters.AddWithValue("@usuario", conexion.UsuarioConexion);
    // Obtiene el resultado
    object resultado = OperacionSQL.ExecuteScalar(cadenaConexion, obtenerIDCmd);
    // Si el resultado es nulo, no existe la conexion.
    if (resultado == null)
    {
        return -1;
    }
    else
    {
        return (int)resultado;
    }
}

/// <summary>
/// Implementación de la lógica de RegistrarConexion.
/// </summary>
public ResultadoConexion RegistrarConexion(Conexion guardar)
{
    // Si el puerto no está relleno, usar el por defecto almacenado en la bbdd
    // INSERT INTO conexion VALUES
    // tipo_conexion, id_usuario, nombre, direccion, puerto, usuario, contrasenia
    int tipo_conexion = (int)guardar.TipoActual;
    int id_usuario = guardar.Propietario.ID;
    string nombre = guardar.Nombre;
    string direccion = guardar.Direccion;
    string usuario = guardar.UsuarioConexion;
    string contrasenia = guardar.ContraseniaConexion;
    int puerto = guardar.Puerto;
    // La conexion se registra y luego se obtiene el ID de la BBDD en el constructor de Conexion
    if (ExisteConexion(guardar))
    {
        return new ResultadoConexion(ResultadoConexion.TipoResultado.DUPLICADO, null);
    }
    else
    {
        SqlCommand registrarCmd = new SqlCommand(registrarConexionQuery);
        // @tipo_conexion, @id_usuario, @nombre, @direccion, @puerto, @usuario, @contrasenia
        registrarCmd.Parameters.AddWithValue("@tipo_conexion", tipo_conexion);
        registrarCmd.Parameters.AddWithValue("@id_usuario", id_usuario);
        registrarCmd.Parameters.AddWithValue("@nombre", nombre);
        registrarCmd.Parameters.AddWithValue("@direccion", direccion);
        registrarCmd.Parameters.AddWithValue("@puerto", puerto);
        registrarCmd.Parameters.AddWithValue("@usuario", usuario);
        registrarCmd.Parameters.AddWithValue("@contrasenia", contrasenia);

        // Obtiene el resultado
        int resultadoFilasSQL = OperacionSQL.ExecuteNonQuery(cadenaConexion, registrarCmd);

        // Si es distinto mayor a 0, se habrá registrado la conexion
        if (resultadoFilasSQL > 0)
        {
            // Devuelve la conexión guardada con su ID asignado
            guardar.ID = ObtenerIDConexion(guardar);
            return new ResultadoConexion(ResultadoConexion.TipoResultado.ACEPTADO, guardar);
        }
    }
}

```

```

        else
        {
            return new ResultadoConexion(ResultadoConexion.TipoResultado.ERROR, null);
        }
    }
}

/// <summary>
/// Implementación de la lógica de EliminarConexion.
/// </summary>
public bool EliminarConexion(Conexion eliminar)
{
    SqlCommand eliminarCmd = new SqlCommand(eliminarConexionQuery);
    eliminarCmd.Parameters.AddWithValue("@id_conexion", eliminar.ID);
    // Obtiene resultado
    int resultadoFilasSQL = OperacionSQL.ExecuteNonQuery(cadenaConexion, eliminarCmd);

    // Si es distinto a 0, se eliminado la conexión
    return (resultadoFilasSQL != 0);
}

/// <summary>
/// Implementación de la lógica de ObtenerConexionesUsuario.
/// </summary>
public ObservableCollection<Conexion> ObtenerConexionesUsuario(Usuario usuario)
{
    // Crea el comando
    SqlCommand obtenerConexCmd = new SqlCommand(obtenerConexionesQuery);
    obtenerConexCmd.Parameters.AddWithValue("@id_usuario", usuario.ID);
    using (SqlDataReader lector = OperacionSQL.ExecuteReader(cadenaConexion, obtenerConexCmd))
    {
        // Si el lector no es nulo, parsear las conexiones
        if (lector != null)
        {
            ObservableCollection<Conexion> retorno = new ObservableCollection<Conexion>(BBDDProgramaMapeo.ExtraerConexiones(lector,
usuario));
            return retorno;
        }
        else
        {
            return null;
        }
    }
}

//// Métodos privados ////

/// <summary>
/// Comprueba si existe el nombre de un usuario en BBDD
/// </summary>
/// <param name="usuario">El nombre del usuario a buscar</param>
/// <returns>True si existe el usuario.</returns>
private bool ExisteUsuario(string usuario)
{
    // Crea el comando
    SqlCommand existirCmd = new SqlCommand(existirQuery);
    existirCmd.Parameters.AddWithValue("@usuario", usuario);
    // Obtiene resultado
    object resultado = OperacionSQL.ExecuteScalar(cadenaConexion, existirCmd);

    // Si el resultado es nulo, no existe el usuario.
    return (resultado != null);
}

/// <summary>
/// Comprueba si existe una conexión en BBDD
/// </summary>
/// <param name="usuario">La conexión</param>
/// <returns>True si existe la conexión.</returns>
private bool ExisteConexion(Conexion guardar)
{
    int resultado = ObtenerIDConexion(guardar);

    // Si el resultado es distinto a -1, existe la conexion
    return (resultado != -1);
}

/// <summary>
/// Implementación de ObtenerPuertoDefecto.
/// </summary>
public int ObtenerPuertoDefecto(Conexion.TipoConexion tipo)
{
    // Crea el comando
    SqlCommand obtenerPuerto = new SqlCommand(obtenerPuertoQuery);
    obtenerPuerto.Parameters.AddWithValue("@id_tipo", (int)tipo);
    // Obtiene y devuelve el resultado
    return (int) OperacionSQL.ExecuteScalar(cadenaConexion, obtenerPuerto);
}
}

```

```

public class ComandoMicrosoftSQL
{
    /// <summary>
    /// Define los tipos de datos disponibles para una BBDD SQL Server
    /// </summary>
    public static readonly string[] TIPOS_DATOS =
    { "INT", "FLOAT", "NVARCHAR(50)", "NVARCHAR(MAX)", "DATETIME" };

    /**
     * Las siguientes líneas definen los comandos SQL con la sintaxis específica de SQL Server.
     */
    private const string SHOW_DATABASES = "SELECT name FROM master.sys.databases";
    private const string CREATE_DATABASE = "CREATE DATABASE ";
    private const string DROP_DATABASE_FORCE = "ALTER DATABASE @1param SET SINGLE_USER WITH ROLLBACK IMMEDIATE; DROP DATABASE @1param";
    private const string DROP_DATABASE = "DROP DATABASE ";
    private const string SHOW_TABLES =
    "SELECT TABLE_NAME FROM @1param.INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE'";
    private const string SHOW_COLUMNS =
    "SELECT COLUMN_NAME FROM @1param.INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = '@2param'";
    private const string SHOW_COLUMNS_DATA_TYPE =
    "SELECT DATA_TYPE FROM @1param.INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = '@2param'";
    private const string CREATE_TABLE = "CREATE TABLE ";
    private const string DROP_TABLE = "DROP TABLE ";
    private const string ALTER_TABLE = "ALTER TABLE ";
    private const string DELETE_FROM = "DELETE FROM @1param ";
    private const string UPDATE = "UPDATE @1param SET ";
    private const string INSERT = "INSERT INTO @1param (@2param) VALUES (@3param)";
    private const string SELECT = "SELECT @1param FROM @2param @3param @4param";

    /// <summary>
    /// Devuelve un comando SQL Server construido para mostrar bases de datos,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand ShowDatabases()
    {
        return new SqlCommand(SHOW_DATABASES);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para crear bases de datos,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand CreateDatabase()
    {
        return new SqlCommand(CREATE_DATABASE);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para eliminar bases de datos,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand DropDatabase(bool forzar)
    {
        if (!forzar)
            return new SqlCommand(DROP_DATABASE);
        else
            return new SqlCommand(DROP_DATABASE_FORCE);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para crear tables,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand CreateTable()
    {
        return new SqlCommand(CREATE_TABLE);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para borrar tablas,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand DropTable()
    {
        return new SqlCommand(DROP_TABLE);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para alterar tablas,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand AlterTable()
    {
        return new SqlCommand(ALTER_TABLE);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para mostrar tablas,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand ShowTables()

```

```

    {
        return new SqlCommand(SHOW_TABLES);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para mostrar columnas de tablas,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand ShowColumnas()
    {
        return new SqlCommand(SHOW_COLUMNS);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para mostrar los tipos de las columnas de tablas,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand ShowTiposDatosColumnas()
    {
        return new SqlCommand(SHOW_COLUMNS_DATA_TYPE);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para eliminar datos,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand DeleteFrom()
    {
        return new SqlCommand(DELETE_FROM);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para actualizar datos,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand Update()
    {
        return new SqlCommand(UPDATE);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para añadir datos,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand Insert()
    {
        return new SqlCommand(INSERT);
    }

    /// <summary>
    /// Devuelve un comando SQL Server construido para mostrar datos,
    /// como se especifica en Comando.cs.
    /// </summary>
    public static DbCommand Select()
    {
        return new SqlCommand(SELECT);
    }
}

public class OperacionSQL
{
    /// <summary>
    /// Mensaje a devolver en caso que la consulta sea fallida.
    /// </summary>
    private const string MSJ_ERROR =
        "Error en la operación, compruebe los datos o la conexión con la base de datos.";

    /// <summary>
    /// Conexión única para evitar conexiones redundantes al usar ExecuteReader
    /// </summary>
    private static SqlConnection sqlConnReader;

    /// <summary>
    /// Implementa para una conexión SQL Server el método ExecuteTest como se especifica en Operacion.cs
    /// </summary>
    public static bool ExecuteTest(string cadenaConexion)
    {
        using (SqlConnection sqlCon = new SqlConnection(cadenaConexion))
        {
            try
            {
                // Abre y cierra la conexión, si no salta excepción, la conexión es correcta.
                sqlCon.Open();
                sqlCon.Close();
                return true;
            }
            catch (SqlException s)
            {
                MessageBox.Show(s.Message);
                return false;
            }
        }
    }
}

```

```

        catch (Exception s)
        {
            MessageBox.Show(MSG_ERROR);
            Console.WriteLine(s.Message);
            return false;
        }
    }
}

/// <summary>
/// Implementa para una conexión SQL Server el método ExecuteScalar como se especifica en Operacion.cs
/// </summary>
public static object ExecuteScalar(string cadenaConexion, SqlCommand comando)
{
    using (SqlConnection sqlCon = new SqlConnection(cadenaConexion))
    {
        try
        {
            // 1. Abre the la conexión
            sqlCon.Open();
            comando.Connection = sqlCon;
            // 2. Ejecuta y devuelve un objeto resultado
            return comando.ExecuteScalar();
        }
        catch (SqlException s)
        {
            MessageBox.Show(s.Message);
            return Operacion.ERROR;
        }
        catch (Exception s)
        {
            MessageBox.Show(MSG_ERROR);
            Console.WriteLine(s.Message);
            return Operacion.ERROR;
        }
    }
}

/// <summary>
/// Implementa para una conexión SQL Server el método ExecuteNonQuery como se especifica en Operacion.cs
/// </summary>
public static int ExecuteNonQuery(string cadenaConexion, SqlCommand comando)
{
    using (SqlConnection sqlCon = new SqlConnection(cadenaConexion))
    {
        try
        {
            // 1. Abre the la conexión
            sqlCon.Open();
            comando.Connection = sqlCon;
            // 2. Ejecuta y devuelve el número de filas afectadas
            return comando.ExecuteNonQuery();
        }
        catch (SqlException s)
        {
            MessageBox.Show(s.Message);
            return Operacion.ERROR;
        }
        catch (Exception s)
        {
            MessageBox.Show(MSG_ERROR);
            Console.WriteLine(s.Message);
            return Operacion.ERROR;
        }
    }
}

/// <summary>
/// Implementa para una conexión SQL Server el método ExecuteReader como se especifica en Operacion.cs
/// </summary>
public static SqlDataReader ExecuteReader(string cadenaConexion, SqlCommand comando)
{
    try
    {
        sqlConnReader = new SqlConnection(cadenaConexion);
        sqlConnReader.Open();
        comando.Connection = sqlConnReader;
        return comando.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
    }
    catch (SqlException s)
    {
        MessageBox.Show(s.Message);
        return null;
    }
    catch (Exception s)
    {
        MessageBox.Show(MSG_ERROR);
        Console.WriteLine(s.Message);
        return null;
    }
}
}

```



```

    }

    public class Comun
    {
        /// <summary>
        /// Opción por defecto al elegir una base de datos
        /// </summary>
        private const string CMB_OPCION_DEFECTO_DATABASE = "Elige base de datos...";

        /// <summary>
        /// Opción por defecto al elegir una tabla
        /// </summary>
        private const string CMB_OPCION_DEFECTO_TABLE = "Elige tabla...";

        /// <summary>
        /// Comprueba si está elegida la opción por defecto.
        /// </summary>
        /// <param name="combo">El ComboBox que se desea comprobar.</param>
        /// <returns>True si está elegida la opción por defecto.</returns>
        public static bool ElegidaTablaDefecto(ComboBox combo)
        {
            return combo.SelectedItem == null
                || combo.SelectedItem.Equals(CMB_OPCION_DEFECTO_TABLE);
        }

        /// <summary>
        /// Comprueba si está elegida la opción por defecto.
        /// </summary>
        /// <param name="combo">El ComboBox que se desea comprobar.</param>
        /// <returns>True si está elegida la opción por defecto.</returns>
        public static bool ElegidaBaseDatosDefecto(ComboBox combo)
        {
            return combo.SelectedItem == null
                || combo.SelectedItem.Equals(CMB_OPCION_DEFECTO_DATABASE);
        }

        /// <summary>
        /// Rellena un ComboBox con el nombre de las tablas de la BBDD actual.
        /// </summary>
        /// <param name="actual">La conexión actual.</param>
        /// <param name="aRellenar">El ComboBox a rellenar.</param>
        public static void RellenarComboTablas(Conexion actual, ComboBox aRellenar)
        {
            List<string> nombresTablas =
                Operacion.MapearReaderALista(Operacion.ObtenerReaderTablas(actual));
            // Rellena el combobox con las bases de datos o tablas pertinentes
            nombresTablas.Insert(0, CMB_OPCION_DEFECTO_TABLE);
            aRellenar.Items.Clear();
            Rellena.ComboBox(aRellenar, nombresTablas);
            aRellenar.SelectedIndex = 0;
        }

        /// <summary>
        /// Rellena un ComboBox con el nombre de las BBDD de la conexión actual.
        /// </summary>
        /// <param name="actual">La conexión actual.</param>
        /// <param name="aRellenar">El ComboBox a rellenar.</param>
        public static void RellenarComboBasesDatos(Conexion actual, ComboBox aRellenar)
        {
            List<string> nombresBBDD =
                Operacion.MapearReaderALista(Operacion.ObtenerReaderBasesDatos(actual));
            // Rellena el combobox con las bases de datos o tablas pertinentes
            nombresBBDD.Insert(0, CMB_OPCION_DEFECTO_DATABASE);
            aRellenar.Items.Clear();
            Rellena.ComboBox(aRellenar, nombresBBDD);
            aRellenar.SelectedIndex = 0;
        }

        /// <summary>
        /// Genera una serie de controles que se insertan en un contenedor.
        /// Usado para mostrar los campos de un SELECT
        /// </summary>
        /// <param name="contenedor">El contenedor donde serán los controles insertados.</param>
        /// <param name="actual">La conexión actual.</param>
        /// <param name="nombreTabla">Nombre de la tabla actual.</param>
        /// <param name="handlerEventosCheckboxes">Evento al que llamarán los checkboxes
        /// generados.</param>
        public static void GenerarShowTables(StackPanel contenedor, Conexion actual,
            Action<object, RoutedEventArgs> handlerEventosButtons)
        {
            List<string> nombre_tablas =
                Operacion.MapearReaderALista(
                    Operacion.ObtenerReaderTablas(actual));

            // Se vacía de contenido el stackpanel
            contenedor.Children.Clear();

            for (int i = 0; i < nombre_tablas.Count; i++)
            {
                // GRID PADRE
                Grid contenedorHijo = new Grid();
            }
        }
    }

```

```

        contenedorHijo.Margin = new Thickness(0, 5, 0, 0);
        // Se le asignan las columnas 1* 3* 2*
        ColumnDefinition gridCol0 = new ColumnDefinition();
        gridCol0.Width = new GridLength(1, GridUnitType.Star);
        ColumnDefinition gridCol1 = new ColumnDefinition();
        gridCol1.Width = new GridLength(3, GridUnitType.Star);
        ColumnDefinition gridCol2 = new ColumnDefinition();
        gridCol2.Width = new GridLength(2, GridUnitType.Star);

        contenedorHijo.ColumnDefinitions.Add(gridCol0);
        contenedorHijo.ColumnDefinitions.Add(gridCol1);
        contenedorHijo.ColumnDefinitions.Add(gridCol2);

        // LBL POSICION
        Label lblPosicion = new Label();
        lblPosicion.Height = 25;
        lblPosicion.Content = i + 1 + "°";
        lblPosicion.VerticalAlignment = VerticalAlignment.Center;
        lblPosicion.HorizontalAlignment = HorizontalAlignment.Center;

        // TXT COLUMNA
        TextBox txtTabla = new TextBox();
        txtTabla.Height = 25;
        txtTabla.IsReadOnly = true;
        txtTabla.Text = nombre_tablas[i];
        txtTabla.HorizontalAlignment = HorizontalAlignment.Center;

        // BTN DATOS
        Button btnDatos = new Button();
        btnDatos.Height = 25;
        btnDatos.Content = "Ver datos";
        btnDatos.Margin = new Thickness(5, 0, 5, 0);
        btnDatos.VerticalContentAlignment = VerticalAlignment.Center;
        btnDatos.HorizontalAlignment = HorizontalAlignment.Center;

        // Se asignan posiciones para los hijos del grid padre
        Grid.SetColumn(lblPosicion, 0);
        Grid.SetColumn(txtTabla, 1);
        Grid.SetColumn(btnDatos, 2);

        // Se asignan eventos a los controles dinámicos
        btnDatos.Click += new RoutedEventHandler(handlerEventosButtons);

        // Se añaden los elementos a sus respectivas posiciones
        contenedorHijo.Children.Add(lblPosicion);
        contenedorHijo.Children.Add(txtTabla);
        contenedorHijo.Children.Add(btnDatos);

        contenedor.Children.Add(contenedorHijo);
    }
}

/// <summary>
/// Genera una serie de controles que se insertan en un contenedor.
/// Usado para mostrar los campos de un SELECT
/// </summary>
/// <param name="contenedor">El contenedor donde serán los controles insertados.</param>
/// <param name="actual">La conexión actual.</param>
/// <param name="nombreTabla">Nombre de la tabla actual.</param>
/// <param name="handlerEventosCheckboxes">Evento al que llamarán los checkboxes
/// generados.</param>
public static void GenerarCamposSelect(StackPanel contenedor, Conexion actual, string nombreTabla,
    Action<object, RoutedEventArgs> handlerEventosCheckboxes)
{
    List<string> nombre_columnas =
        Operacion.MapearReaderALista(
            Operacion.ObtenerReaderColumnas(actual, nombreTabla));
    List<string> tipo_datos =
        Operacion.MapearReaderALista(
            Operacion.ObtenerReaderTiposDatosColumnas(actual, nombreTabla));

    int numCamposGenerar = nombre_columnas.Count;

    // Se vacía de contenido el stackpanel
    contenedor.Children.Clear();

    for (int i = 0; i < numCamposGenerar; i++)
    {
        // GRID PADRE
        Grid contenedorHijo = new Grid();
        contenedorHijo.Margin = new Thickness(0, 5, 0, 0);
        // Se le asignan las columnas 1* 5* 4*
        ColumnDefinition gridCol0 = new ColumnDefinition();
        gridCol0.Width = new GridLength(1, GridUnitType.Star);
        ColumnDefinition gridCol1 = new ColumnDefinition();
        gridCol1.Width = new GridLength(5, GridUnitType.Star);
        ColumnDefinition gridCol2 = new ColumnDefinition();
        gridCol2.Width = new GridLength(4, GridUnitType.Star);

        contenedorHijo.ColumnDefinitions.Add(gridCol0);
        contenedorHijo.ColumnDefinitions.Add(gridCol1);

```

```

        contenedorHijo.ColumnDefinitions.Add(gridCol2);

        // Debe tener
        // chkElegir
        // txtColumna
        // txtTipoDato

        // CHK ELEGIR
        CheckBox chkElegir = new CheckBox();
        chkElegir.HorizontalAlignment = HorizontalAlignment.Center;
        chkElegir.VerticalAlignment = VerticalAlignment.Center;

        // TXT COLUMNA
        TextBox txtColumna = new TextBox();
        txtColumna.Height = 25;
        txtColumna.IsReadOnly = true;
        txtColumna.Text = nombre_columnas[i];
        txtColumna.VerticalContentAlignment = VerticalAlignment.Center;
        txtColumna.HorizontalContentAlignment = HorizontalAlignment.Center;

        // TXT TIPOS DATOS
        TextBox txtTipoDato = new TextBox();
        txtTipoDato.Height = 25;
        txtTipoDato.IsReadOnly = true;
        txtTipoDato.Text = tipo_datos[i];
        txtTipoDato.Margin = new Thickness(5, 0, 5, 0);
        txtTipoDato.VerticalContentAlignment = VerticalAlignment.Center;
        txtTipoDato.HorizontalContentAlignment = HorizontalAlignment.Center;

        // Se asignan posiciones para los hijos del grid padre
        Grid.SetColumn(chkElegir, 0);
        Grid.SetColumn(txtColumna, 1);
        Grid.SetColumn(txtTipoDato, 2);

        // Se asignan eventos a los controles dinámicos
        chkElegir.Checked += new RoutedEventHandler(handlerEventosCheckboxes);
        chkElegir.Unchecked += new RoutedEventHandler(handlerEventosCheckboxes);

        // Se añaden los elementos a sus respectivas posiciones
        contenedorHijo.Children.Add(chkElegir);
        contenedorHijo.Children.Add(txtColumna);
        contenedorHijo.Children.Add(txtTipoDato);

        contenedor.Children.Add(contenedorHijo);
    }
}

/// <summary>
/// Extrae los datos de un contenedor con controles generados.
/// </summary>
/// <param name="contenedor">El control del que extraer los datos.</param>
/// <returns>Una lista de datos.</returns>
public static async Task<List<ColumnaValor>> ExtraerDatosCamposSelect(StackPanel contenedor)
{
    // Espera 5ms para que de tiempo a repintar los componentes
    await Task.Delay(5);
    // Comprobar los campos de las condiciones generados y extraer datos
    List<ColumnaValor> datos = new List<ColumnaValor>();
    var generados = contenedor.Children;

    // ||| Recorrer el grid |||
    // +Grid[i]
    // -chkElegir[i][0] -> Comprobar
    // -txtColumna[i][1] -> Obtener nombre
    // -txtTipoDato[i][2]
    for (int i = 0; i < generados.Count; i++)
    {
        if (generados[i] is Grid)
        {
            Grid grid = generados[i] as Grid;
            CheckBox chkElegir = grid.Children[0] as CheckBox;
            TextBox txtColumna = grid.Children[1] as TextBox;

            // Por cada fila con chkElegir marcado, extraer datos columna y valor
            if (chkElegir.IsChecked.Value)
            {
                string columna = txtColumna.Text?.ToString();
                ColumnaValor filaDatos = new ColumnaValor(columna, "");

                datos.Add(filaDatos);
            }
        }
    }
    return datos;
}

/// <summary>
/// Genera una serie de controles que se insertan en un contenedor.
/// </summary>
/// <param name="contenedor">El contenedor donde serán los controles insertados.</param>

```

```

/// <param name="actual">La conexión actual.</param>
/// <param name="nombreTabla">Nombre de la tabla actual.</param>
/// <param name="handlerEventosCheckboxes">Evento al que llamarán
/// los checkboxes generados.</param>
/// <param name="handlerEventosTextboxes">Evento al que llamarán
/// los textboxes generados.</param>
public static void GenerarCamposColumnas(StackPanel contenedor, Conexion actual, string nombreTabla,
    Action<object, RoutedEventArgs> handlerEventosCheckboxes, Action<object, TextChangedEventArgs> handlerEventosTextboxes)
{
    List<string> nombre_columnas =
        Operacion.MapearReaderALista(
            Operacion.ObtenerReaderColumnas(actual, nombreTabla));
    List<string> tipo_datos =
        Operacion.MapearReaderALista(
            Operacion.ObtenerReaderTiposDatosColumnas(actual, nombreTabla));

    int numCamposGenerar = nombre_columnas.Count;

    // Se vacía de contenido el stackpanel
    contenedor.Children.Clear();

    for (int i = 0; i < numCamposGenerar; i++)
    {
        // GRID PADRE
        Grid contenedorHijo = new Grid();
        contenedorHijo.Margin = new Thickness(0, 5, 0, 0);
        // Se le asignan las columnas 1* 4* 2* 5*
        ColumnDefinition gridCol0 = new ColumnDefinition();
        gridCol0.Width = new GridLength(1, GridUnitType.Star);
        ColumnDefinition gridCol1 = new ColumnDefinition();
        gridCol1.Width = new GridLength(4, GridUnitType.Star);
        ColumnDefinition gridCol2 = new ColumnDefinition();
        gridCol2.Width = new GridLength(2, GridUnitType.Star);
        ColumnDefinition gridCol3 = new ColumnDefinition();
        gridCol3.Width = new GridLength(5, GridUnitType.Star);

        contenedorHijo.ColumnDefinitions.Add(gridCol0);
        contenedorHijo.ColumnDefinitions.Add(gridCol1);
        contenedorHijo.ColumnDefinitions.Add(gridCol2);
        contenedorHijo.ColumnDefinitions.Add(gridCol3);

        // Debe tener
        // chkElegir
        // txtColumna
        // txtTipoDato
        // txtValor

        // CHK ELEGIR
        CheckBox chkElegir = new CheckBox();
        chkElegir.HorizontalAlignment = HorizontalAlignment.Center;
        chkElegir.VerticalAlignment = VerticalAlignment.Center;

        // TXT COLUMNA
        TextBox txtColumna = new TextBox();
        txtColumna.Height = 25;
        txtColumna.IsReadOnly = true;
        txtColumna.Text = nombre_columnas[i];
        txtColumna.VerticalContentAlignment = VerticalAlignment.Center;
        txtColumna.HorizontalContentAlignment = HorizontalAlignment.Center;

        // TXT TIPOS DATOS
        TextBox txtTipoDato = new TextBox();
        txtTipoDato.Height = 25;
        txtTipoDato.IsReadOnly = true;
        txtTipoDato.Text = tipo_datos[i];
        txtTipoDato.Margin = new Thickness(5, 0, 5, 0);
        txtTipoDato.VerticalContentAlignment = VerticalAlignment.Center;
        txtTipoDato.HorizontalContentAlignment = HorizontalAlignment.Center;

        // TXT VALOR
        TextBox txtValor = new TextBox();
        txtValor.Height = 25;
        txtValor.VerticalContentAlignment = VerticalAlignment.Center;

        // Se asignan posiciones para los hijos del grid padre
        Grid.SetColumn(chkElegir, 0);
        Grid.SetColumn(txtColumna, 1);
        Grid.SetColumn(txtTipoDato, 2);
        Grid.SetColumn(txtValor, 3);

        // Se asignan eventos a los controles dinámicos
        chkElegir.Checked += new RoutedEventHandler(handlerEventosCheckboxes);
        chkElegir.Unchecked += new RoutedEventHandler(handlerEventosCheckboxes);
        txtColumna.TextChanged += new TextChangedEventHandler(handlerEventosTextboxes);
        txtTipoDato.TextChanged += new TextChangedEventHandler(handlerEventosTextboxes);
        txtValor.TextChanged += new TextChangedEventHandler(handlerEventosTextboxes);

        // Se añaden los elementos a sus respectivas posiciones
        contenedorHijo.Children.Add(chkElegir);
        contenedorHijo.Children.Add(txtColumna);
        contenedorHijo.Children.Add(txtTipoDato);
    }
}

```

```

        contenedorHijo.Children.Add(txtValor);

        contenedor.Children.Add(contenedorHijo);
    }
}

/// <summary>
/// Extrae los datos de un contenedor con controles generados.
/// </summary>
/// <param name="contenedor">El control del que extraer los datos.</param>
/// <returns>Una lista de datos.</returns>
public static async Task<List<ColumnaValor>> ExtraerDatosCamposColumnas(StackPanel contenedor)
{
    // Espera 5ms para que de tiempo a repintar los componentes
    await Task.Delay(5);
    // Comprobar los campos de las condiciones generados y extraer datos
    List<ColumnaValor> datos = new List<ColumnaValor>();
    var generados = contenedor.Children;

    // ||| Recorrer el grid |||
    // +Grid[i]
    // -chkElegir[i][0] -> Comprobar
    // -txtColumna[i][1] -> Obtener nombre
    // -txtTipoDato[i][2]
    // -txtValor[i][3] -> Obtener datos
    for (int i = 0; i < generados.Count; i++)
    {
        if (generados[i] is Grid)
        {
            Grid grid = generados[i] as Grid;
            CheckBox chkElegir = grid.Children[0] as CheckBox;
            TextBox txtColumna = grid.Children[1] as TextBox;
            TextBox txtValor = grid.Children[3] as TextBox;

            // Por cada fila con chkElegir marcado, extraer datos columna y valor
            if (chkElegir.IsChecked.Value)
            {
                string columna = txtColumna.Text?.ToString();
                string valor = txtValor.Text?.ToString();
                ColumnaValor filaDatos = new ColumnaValor(columna, valor);

                datos.Add(filaDatos);
            }
        }
    }
    return datos;
}

/// <summary>
/// Genera una serie de controles que se insertan en un contenedor.
/// </summary>
/// <param name="contenedor">El contenedor donde serán los controles insertados.</param>
/// <param name="actual">La conexión actual.</param>
/// <param name="numCondiciones">Número de condiciones a generar.</param>
/// <param name="nombreTabla">Nombre de la tabla actual.</param>
/// <param name="handlerEventosCheckboxes">Evento al que llamarán los checkboxes generados.</param>
/// <param name="handlerEventosTextboxes">Evento al que llamarán los textboxes generados.</param>
public static void GenerarCamposWhere(StackPanel contenedor, Conexion actual, int numCondiciones, string nombreTabla,
    Action<object, SelectionChangedEventArgs> handlerEventosCombos, Action<object, TextChangedEventArgs> handlerEventosTextboxes)
{
    string[] tipos_operadores = Comando.TIPOS_CONDICIONES;
    string[] tipos_operadores_union = Comando.TIPOS_CONDICIONES_UNION;
    List<string> nombre_columnas =
        Operacion.MapearReaderALista(
            Operacion.ObtenerReaderColumnas(actual, nombreTabla));
    nombre_columnas.Insert(0, "");

    // Se vacía de contenido el stackpanel
    contenedor.Children.Clear();

    for (int i = 0; i < numCondiciones; i++)
    {
        // GRID PADRE
        Grid contenedorSuperior = new Grid();
        // Se le asignan las columnas 4* 2.5* 5*
        ColumnDefinition gridCol1 = new ColumnDefinition();
        gridCol1.Width = new GridLength(4, GridUnitType.Star);
        ColumnDefinition gridCol2 = new ColumnDefinition();
        gridCol2.Width = new GridLength(2.5, GridUnitType.Star);
        ColumnDefinition gridCol3 = new ColumnDefinition();
        gridCol3.Width = new GridLength(5, GridUnitType.Star);

        contenedorSuperior.ColumnDefinitions.Add(gridCol1);
        contenedorSuperior.ColumnDefinitions.Add(gridCol2);
        contenedorSuperior.ColumnDefinitions.Add(gridCol3);

        // CMB COLUMNA
        ComboBox cmbColumna = new ComboBox();

```

```

        cmbColumna.Height = 25;
        foreach (var nombre in nombre_columnas)
        {
            cmbColumna.Items.Add(nombre);
        }

        // CMB OPERADORES
        ComboBox cmbOperadores = new ComboBox();
        cmbOperadores.Height = 25;
        cmbOperadores.Margin = new Thickness(0, 5, 0, 5);
        foreach (var tipoDato in tipos_operadores)
        {
            cmbOperadores.Items.Add(tipoDato);
        }

        // TXTVALOR
        TextBox txtValor = new TextBox();
        cmbOperadores.Height = 25;

        // CMB AND OR
        ComboBox cmbAndOr = new ComboBox();
        cmbAndOr.Margin = new Thickness(0, 5, 0, 5);
        cmbAndOr.HorizontalContentAlignment = HorizontalAlignment.Center;
        foreach (var tipoDato in tipos_operadores_union)
        {
            cmbAndOr.Items.Add(tipoDato);
        }

        // Se asignan posiciones para los hijos del grid padre
        Grid.SetColumn(cmbColumna, 0);
        Grid.SetColumn(cmbOperadores, 1);
        Grid.SetColumn(txtValor, 2);

        // Se asignan eventos a los controles dinámicos
        cmbColumna.SelectionChanged += new SelectionChangedEventHandler(handlerEventosCombos);
        cmbOperadores.SelectionChanged += new SelectionChangedEventHandler(handlerEventosCombos);
        txtValor.TextChanged += new TextChangedEventHandler(handlerEventosTextboxes);
        cmbAndOr.SelectionChanged += new SelectionChangedEventHandler(handlerEventosCombos);

        // Se añaden los elementos a sus respectivas posiciones
        contenedorSuperior.Children.Add(cmbColumna);
        contenedorSuperior.Children.Add(cmbOperadores);
        contenedorSuperior.Children.Add(txtValor);

        contenedor.Children.Add(contenedorSuperior);
        contenedor.Children.Add(cmbAndOr);
    }
    // Elimina el último ComboBox AND OR
    int tamaño = contenedor.Children.Count;
    if (tamaño > 0)
        contenedor.Children.RemoveAt(tamaño - 1);
}

/// <summary>
/// Extrae los datos de un contenedor con controles generados.
/// </summary>
/// <param name="contenedor">El control del que extraer los datos.</param>
/// <returns>Una lista de datos.</returns>
public static async Task<string> ExtraerDatosWhere(StackPanel contenedor)
{
    // Espera 5ms para que de tiempo a repintar los componentes
    await Task.Delay(5);
    // Comprobar los campos de las condiciones generados y extraer datos
    string datos = " WHERE ";
    var generados = contenedor.Children;

    // Por cada condicion existente, extraerlas y emparejarlas con un AND u OR
    // +Grid[i]
    // -cmbColumna[i][0]
    // -cmbOperadores[i][1]
    // -txtValor[i][2]
    // +cmbAndOr[i]
    for (int i = 0; i < generados.Count; i++)
    {
        if (generados[i] is Grid)
        {
            Grid grid = generados[i] as Grid;
            ComboBox cmbColumna = grid.Children[0] as ComboBox;
            ComboBox cmbOperadores = grid.Children[1] as ComboBox;
            TextBox txtValor = grid.Children[2] as TextBox;

            // Datos
            datos += cmbColumna.SelectedItem?.ToString() + " ";
            datos += cmbOperadores.SelectedItem?.ToString() + " ";
            datos += txtValor.Text?.ToString() + " ";
        }
        else if (generados[i] is ComboBox)
        {
            ComboBox cmbAndOr = generados[i] as ComboBox;
            datos += cmbAndOr.SelectedItem?.ToString() + " ";
        }
    }
}

```

```

    }
    return datos;
}

/// <summary>
/// Genera una serie de controles que se insertan en un contenedor.
/// </summary>
/// <param name="contenedor">El contenedor donde serán los controles insertados.</param>
/// <param name="actual">La conexión actual.</param>
/// <param name="nombreTabla">Nombre de la tabla actual.</param>
/// <param name="handlerEventosCheckboxes">Evento al que llamarán
/// los checkboxes generados.</param>
/// <param name="handlerEventosTextboxes">Evento al que llamarán
/// los textboxes generados.</param>
public static void GenerarCamposOrderBy(StackPanel contenedor, Conexion actual, int numCampos,
    List<ColumnaValor> camposElegidos, string nombreTabla, Action<object, SelectionChangedEventArgs> handlerEventosCombos)
{
    string[] tipos_operadores = Comando.TIPOS_ORDEN;

    // Se vacía de contenido el stackpanel
    contenedor.Children.Clear();

    for (int i = 0; i < numCampos; i++)
    {
        // GRID PADRE
        Grid contenedorSuperior = new Grid();
        // Se le asignan las columnas 1* 4* .5*
        ColumnDefinition gridCol1 = new ColumnDefinition();
        gridCol1.Width = new GridLength(1, GridUnitType.Star);
        ColumnDefinition gridCol2 = new ColumnDefinition();
        gridCol2.Width = new GridLength(4, GridUnitType.Star);
        ColumnDefinition gridCol3 = new ColumnDefinition();
        gridCol3.Width = new GridLength(1.5, GridUnitType.Star);

        contenedorSuperior.ColumnDefinitions.Add(gridCol1);
        contenedorSuperior.ColumnDefinitions.Add(gridCol2);
        contenedorSuperior.ColumnDefinitions.Add(gridCol3);

        // LBL POSICION
        Label lblPosicion = new Label();
        lblPosicion.Height = 25;
        lblPosicion.Content = i + 1 + "º";
        lblPosicion.VerticalAlignment = VerticalAlignment.Center;
        lblPosicion.HorizontalAlignment = HorizontalAlignment.Center;

        // CMB COLUMNA
        ComboBox cmbColumna = new ComboBox();
        cmbColumna.Height = 25;
        cmbColumna.Items.Add("");
        foreach (var columna in camposElegidos)
        {
            cmbColumna.Items.Add(columna.Columna);
        }

        // CMB OPERADORES
        ComboBox cmbOperadores = new ComboBox();
        cmbOperadores.Height = 25;
        cmbOperadores.Margin = new Thickness(0, 5, 0, 5);
        foreach (var tipoDato in tipos_operadores)
        {
            cmbOperadores.Items.Add(tipoDato);
        }

        // Se asignan posiciones para los hijos del grid padre
        Grid.SetColumn(lblPosicion, 0);
        Grid.SetColumn(cmbColumna, 1);
        Grid.SetColumn(cmbOperadores, 2);

        // Se asignan eventos a los controles dinámicos
        cmbColumna.SelectionChanged += new SelectionChangedEventHandler(handlerEventosCombos);
        cmbOperadores.SelectionChanged += new SelectionChangedEventHandler(handlerEventosCombos);

        // Se añaden los elementos a sus respectivas posiciones
        contenedorSuperior.Children.Add(lblPosicion);
        contenedorSuperior.Children.Add(cmbColumna);
        contenedorSuperior.Children.Add(cmbOperadores);

        contenedor.Children.Add(contenedorSuperior);
    }
}

/// <summary>
/// Extrae los datos de un contenedor con controles generados.
/// </summary>
/// <param name="contenedor">El control del que extraer los datos.</param>
/// <returns>Una lista de datos.</returns>
public static async Task<List<ColumnaValor>> ExtraerDatosOrderBy(StackPanel contenedor)
{
    // Espera 5ms para que de tiempo a repintar los componentes
    await Task.Delay(5);
    // Comprobar los campos de las condiciones generados y extraer datos

```

```

var generados = contenedor.Children;

List<ColumnaValor> datos = new List<ColumnaValor>();

// Por cada campo a ordenar, emparejarlo con su sentido ASC|DESC
// +Grid[i]
// -lblPosicion[i][0]
// -cmbColumna[i][1]
// -cmbOperadores[i][2]
for (int i = 0; i < generados.Count; i++)
{
    if (generados[i] is Grid)
    {
        Grid grid = generados[i] as Grid;
        ComboBox cmbColumna = grid.Children[1] as ComboBox;
        ComboBox cmbOperadores = grid.Children[2] as ComboBox;

        // Datos
        string columna = cmbColumna.SelectedItem?.ToString();
        string operador = cmbOperadores.SelectedItem?.ToString();
        datos.Add(new ColumnaValor(columna, operador));
    }
}
return datos;
}

/// <summary>
/// Marca o no todos los checkboxes del contenedor proporcionado.
/// </summary>
/// <param name="contenedor">El contenedor del que marcar los checkboxes.</param>
/// <param name="marcado">True si se desean marcar, false los desmarca.</param>
public static void MarcarTodosCamposColumnas(StackPanel contenedor, bool marcado)
{
    var generados = contenedor.Children;

    // Por cada condicion existente, extraerlas y emparejarlas con un AND u OR
    // +Grid[i]
    // -chkElegir[i][0] -> Marcar
    // -txtColumna[i][1]
    // -txtTipoDato[i][2]
    // -txtValor[i][3]
    for (int i = 0; i < generados.Count; i++)
    {
        if (generados[i] is Grid)
        {
            Grid grid = generados[i] as Grid;
            CheckBox chkElegir = grid.Children[0] as CheckBox;

            chkElegir.IsChecked = marcado;
        }
    }
}
}

```


ELABORADO POR: ALEJANDRO ANTONIO DEL RÍO MORENO.

FECHA DE ENTREGA: 18/06/2018.

EMAIL: alejandro.rio@outlook.es