Departamento de Sistemas Informáticos y Comutación

Universitat Politècnica de València

CASO REAL DE LAS FASES DEL DESARROLLO SOFTWARE DE UN SISTEMA DE GESTIÓN DE DATOS PARA LA TOMA DE DECISIONES DENTRO DE PROCESOS DE INGENIERIA RELACIONADOS CON LA FABRICACIÓN Y LANZAMIENTO DE MODELOS DE VEHÍCULOS EN EL MUNDO DE LA AUTOMOCIÓN

Tesis Final de Máster

Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información

2014 - 2015

Autor: Jorge Laguna Hernández

Director: Santiago Escobar Román

Compañía: Itera Soluciones de Ingeniería

14 de Septiembre de 2015

Resumen

El presente proyecto se ha realizado en el marco de una relación laboral del autor del este proyecto con la empresa ITERA Soluciones de Ingeniería SL que tienen entre sus clientes a la planta de fabricación de Ford Motor Company en Almussafes, España y que ha permitido trabajar al autor de este proyecto con información real de los problemas que aparecen al desarrollar un sistema de gestión de datos interno de la empresa ITERA para mejorar los procesos de decisiones que requiere el trabajo en sus clientes.

Actualmente las distintas ramas de ingeniería basan muchas de sus decisiones en los datos que proporciona el desempeño del trabajo que se esté realizando. Pero el gran problema que suele aparecer es que los sistemas que gestionan la información de estos procesos de ingeniería son viejos, desfasados y están diseñados y pensados bajo requisitos que no son acordes a los requisitos que pueden aparecer actualmente.

Para realizar con éxito la gestión de la información disponible y que esta gestión permita tomar decisiones es indispensable disponer de herramientas que permitan manejar esa gran cantidad de datos provenientes de diferentes fuentes. La finalidad de este proyecto es precisamente desarrollar estas herramientas para aunar todos los datos relevantes en un solo sistema de gestión.

En este proyecto abordaremos todas las fases del desarrollo software desde la toma de requisitos hasta el mantenimiento del sistema de gestión de información resultante, viendo todos los problemas que han surgido a lo largo del ciclo de desarrollo y como se han tenido que ir solucionando.

Palabras clave: Calidad, Ford, Itera, PVT, Desarrollo de Producto, Lanzamiento, C#, WPF, Procesos Corporativos, Office, IBM Personal Communications, Emulador terminal 3270, Ingeniería Software, Requisitos, Diseño, Análisis, Implementación, Mantenimiento, Almussafes.

Tabla de contenidos (1)

Contenido

Tabla de contenidos (2)

Tabla de contenidos (3)

1. Resumen

El presente proyecto se ha realizado en el marco de una relación laboral del autor del este proyecto con la empresa ITERA Soluciones de Ingeniería SL que tienen entre sus clientes a la planta de fabricación de Ford Motor Company en Almussafes, España y que ha permitido trabajar al autor de este proyecto con información real de los problemas que aparecen al desarrollar un sistema de gestión de datos interno de la empresa ITERA para mejorar los procesos de decisiones que requiere el trabajo en sus clientes.

Actualmente las distintas ramas de ingeniería basan muchas de sus decisiones en los datos que proporciona el desempeño del trabajo que se esté realizando. Pero el gran problema que suele aparecer es que los sistemas que gestionan la información de estos procesos de ingeniería son viejos, desfasados y están diseñados y pensados bajo requisitos que no son acordes a los requisitos que pueden aparecer actualmente.

Para realizar con éxito la una gestión de la información disponible y que esta gestión permita tomar decisiones es indispensable disponer de herramientas que permitan manejar esa gran cantidad de datos provenientes de diferentes fuentes. La finalidad de este proyecto es precisamente desarrollar estas herramientas para aunar todos los datos relevantes en un solo sistema de gestión.

Como hemos comentado, la información obtenida durante el proceso de ingeniería así como la procedente del cliente, nos proporciona datos sobre si se está realizando las distintas fases de los proyectos correctamente y nos permite tomar decisiones al respecto. En un cliente, como por ejemplo Ford Motor Co, todos estos datos son tratados y centralizados en distintas bases de datos a nivel corporativo y no suelen tener una relación robusta entre sí.

Para comprobar si la toma de decisiones y el trabajo que se realiza diariamente es óptimo, necesitamos trabajar con estas bases de datos, por lo que se plantean las siguientes cuestiones: ¿cómo relacionar estos datos?, ¿cómo actualizar datos aislados con respecto al global?, ¿cómo crear estadísticas, resúmenes, etc., de datos aislados?, ¿cómo informar del significado de datos aislados?, ¿cómo presentar esta información al equipo de trabajo correspondiente?, ¿cómo se puede mejorar el proceso?,…

En resumen, en este proyecto abordaremos todas las fases del desarrollo software desde la toma de requisitos hasta el mantenimiento del sistema de gestión de información

resultante, viendo todos los problemas que han surgido a lo largo del ciclo de desarrollo y como se han tenido que ir solucionando.

1.1. Resumen del documento

Apartado 1: Se realiza una introducción al proyecto y se resume a grandes rasgos el contenido de este proyecto.

Apartado 2: Se exponen las motivaciones tanto del autor del proyecto como de la empresa a la hora de afrontar este proyecto de desarrollo Software para mejorar la eficiencia en su principal área de negocio.

Apartado 3: En este apartado se enumerarán y definirán algunos de los acrónimos utilizados que son de utilidad para este proyecto

Apartado 4: Breve resumen de la empresa la cual tiene los derechos de la aplicación

Apartado 5: Breve tecto de donde se va autilizar la aplicación desarrollada en este proyecto

Apartado 6: Proyecto IMS, cómo se ha desarrollado, y que etapas se han seguido en el propio desarrollo, detallando desde las teclogías utilizadas hasta el mantenimiento del software

Apartado 7: Manual de usuario de la aplicación

Apartado 8: Método para actualizar la versión de la aplicación

Apartado 9: Enumaración de las mejoras que se han ido implementando en la aplicación a lo largo del proyecto

Apartado 10: Conclusiones del proyecto

Apartado 11: Donde podemos encontrar ejemplos de plantillas de código

Apartado 12: Bibliografía

Apartado 13: Otros documentos que acompañarán a este proyecto.

2. Motivaciones

Después de una serie de años trabajando en aspectos relacionados con la calidad y los procesos de fabricación en el mundo de la automoción, vi la oportunidad de enriquecer mis conocimientos sobre el Desarrollo Software apuntándome al Master de Ingeniería del Software, Métodos formales y Sistemas de Información, el cual está distinguido con una Mención de Calidad, impartido en el Departamento de Sistemas Informáticos y Computación en la Universidad Politécnica de Valencia.

Durante la realización de este Máster recibí una propuesta de trabajo para incorporarme a la empresa Itera Soluciones de Ingeniería, que es una empresa de ingeniería con una fuerte presencia en los departamentos de PVT y Lanzamiento en la planta de fabricación de Ford Motor Company en Almussafes, España.

La idea de esta incorporación fue hacerme cargo del desarrollo de una herramienta que permitiese a los ingenieros de Itera, que participaban en los dos departamentos de Ford mencionados anteriormente, realizar un trabajo más eficiente gestionando la información interna de Ford con un sistema de gestión de información flexible y rápido que redujese el tiempo que se necesita para recopilar información y por lo tanto el tiempo que se necesita para la toma de decisiones.

3. Acrónimos utilizados en el proyecto

A continuación se enumeran y definen una serie de acrónimos que se utilizan en el presente proyecto o son de interés para entender procesos utilizados.

- BSAQ System (Balanced Single Agenda for Quality): Programa de equilibrado único para la Calidad. Es un sistema de seguimiento que contiene los datos de calidad de los indicadores externos y la hoja de ruta.
- ICA (Interim Corrective Action): Acción Correctiva Provisional, son acciones que se aplican a consecuencia de un problema de fabricación y que se comprueba si es efectiva.
- MY (Model Year): Año del Modelo, es un período de 12 meses a partir de la primera unidad fabricada para un nuevo modelo.
- New Model Launch Whiteboard: Pizarra Lanzamiento del nuevo modelo, es un proceso para interconectar el actual modelo VRT con el lanzamiento del nuevo

modelo. Se utiliza para conducir la reacción urgente y obtener posibles fallos pudiendo determinar la contención y la aplicación de las medidas correctivas permanente.

- OKTB + 90 days: 90 días después de la confirmación de que los indicadores de satisfacción del cliente se han cumplido define como bien comprar.
- PCA (Permanent Corrective Action) Acción Correctiva Permanente, solución definitiva a un problema de garantías.
- PIC Room – Area de reuniones
- PMT (Program Management Team): Un equipo del programa de gestión del proceso de desarrollo de productos para un grupo de partes relacionadas, o módulos (s). El PMT es responsable de la calidad del módulo, la función, coste, peso y distribución.
- PD (Product Development Function Group): Desarrollo de productos.
- QLS-VO (Quality Leadership System for Vehicle Operations): Sistema de calidad para el liderazgo de las operaciones de vehículos, que es un sistema interno de calidad donde se recopilan todos los datos del vehículo durante la fabricación.
- Subsystem: Un desglose de los datos de calidad total de vehículos aprobados, en grupos más pequeños basados en un mapa de códigos de preocupación del cliente (CCC) y los grupos de vehículos funcionales (VFG).
- FORM 4: Es una matriz de seguimiento que se genera a partir de la base de datos BSAQ y se utiliza para documentar un proceso de resolución de problemas. Contiene las acciones y el calendario adoptado para hacer frente tanto a las preocupaciones internas y externas con datos de verificación (interna, externa y si está disponible) para apoyar la resolución del problema.
- VIN (Vehicle Identification Number) Número de Identificación de Vehículo.
- VRT (Variability Reduction Team): Equipo de Reducción de la variabilidad, es un grupo de personal de planta o equipo asignado para trabajar en la solución de problemas de los subsistemas y en los problemas para mejorar la calidad.
- CR: Corncern, nombre que se adjudica a la información relacionada con un problema y su solución dentro de Ford Motor Company
- WERS: Base de datos de Ford Motor Company accesible a través de un terminal Mainframe
- WOW: WERS On the Web
- PCOMM: IBM Personal Communications. Emulador de un terminal 3270
- IMS: Infomration Management System, nombre que recibe la aplicación desarrollada en este proyecto

4. ITERA

ITERA ofrece una gama de soluciones globales de ingeniería. Más de 10 años en el mercado (Inglaterra, Alemania y España) avalan nuestra EXPERIENCIA en el desarrollo y lanzamiento de productos, especialmente en el sector del automóvil. La FLEXIBILIDAD de nuestro equipo, altamente cualificado, y nuestro sistema de trabajo, basado en la consecución de objetivos, nos permite conseguir la máxima EFICIENCIA en cada proyecto.

4.1. Profesionalidad

Contamos con un equipo de especialistas altamente capacitados en el campo de la ingeniería. El 98% de nuestros empleados son ingenieros titulados. Se trata de un grupo joven, con entusiasmo, liderado por profesionales experimentados, y en FORMACION CONTINUA. Para ITERA el conocimiento del sector y de las últimas tecnologías es clave. La empresa ofrece a sus trabajadores oportunidades de formación individualizada potenciando las competencias de cada uno y añadiendo valor a la empresa.

4.2. Flexibilidad

Adaptamos nuestros servicios a las demandas de nuestros clientes. Respondemos con rapidez a los cambios y a los retos de la actualidad, trabajando por objetivos que permiten aligerar la estructura y los costes de nuestros clientes. Les ofrecemos en cada momento nuevas soluciones de ingeniería que resuelven los desafíos a los que se enfrentan en un mercado cada vez más competitivo y exigente.

4.3. Eficiencia

Hacemos nuestros los objetivos y metas de cada cliente. Analizamos y proponemos para encontrar la solución más eficaz y adecuada. Valoramos las estructuras horizontales, donde todas las opiniones son escuchadas y tenidas en cuenta, y que permiten una EFICIENCIA operacional para alcanzar la máxima calidad optimizando los recursos y con el mejor servicio posible.

4.4. Diseño de producto

Contamos con ingenieros con gran experiencia en el sector de la automoción, preparados para ayudar en el diseño de piezas a terceros. Ofrecemos soporte de factibilidad, selección de materiales y seguimiento de utillajes, así como la asistencia en el diseño de producto utilizando sistemas CAD 3D. Diseñamos galgas de control dimensional, asesoramos en la definición de tolerancias geométricas y nos encargamos de la creación de planes de control. Lideramos desde el diseño inicial hasta su industrialización, así como posibles cambios en serie.

4.5. Prototipado

Disponemos de instalaciones propias de fabricación con las últimas tecnologías (FDM y SLS) y sistemas del mercado. La calidad en la ejecución y los cortos plazos de entrega nos diferencian de nuestros competidores. Las piezas prototipo o series cortas pueden utilizarse en un gran abanico de sectores y aplicaciones.

4.6. Soporte de calidad

Nos encargamos del diseño, implementación y mantenimiento de sistemas de gestión integrado basado en los requerimientos UNE-EN ISO 9001 / ISO/TS 16949, UNE-EN ISO 14001, OHSAS 18001 Y UNE-EN ISO/IEC 17025, requisitos de cliente, legislación interna y requisitos legales. Sistemas avanzados sector automoción: VDA, QS9000 & 6Sigma. Ofrecemos diseño e implementación del Scorecard y auditorías internas. Supervisión, gestión y control de indicadores internos y externos, trabajando siempre bajo los estándares UNE 160000 (I+D+i). Lideramos la acreditación de Q1 en plantas de producción.

4.7. Ingeniería de lanzamiento

Somos expertos en el soporte de ingeniería de desarrollo en lanzamientos de nuevos vehículos. Supervisamos el proyecto y nos encargamos del asesoramiento e interface con los Centros de Desarrollo, Finanzas, Compras, Ensamblaje y Logística. Hacemos el seguimiento de proveedores y timings. Utilización de metodología 5D para la resolución de problemas desde su definición y causa raíz hasta la implementación de la solución.

5. Factoría Ford en Almussafes

5.1. Planta piloto y Equipo de Lanzamiento

La meta principal de este desarrollo Software es crear una herramienta para poder ayudar a los equipo de Itera que están presentes en los equipos de PVT y Lanzamiento la planta de Fabricación de Ford Almussafes.

Estos equipos se encuentran ubicados en la zona de Planta Piloto de la planta donde se realizan pruebas de fabricación antes de que se realicen en la planta de montaje. El objetivo principal de la planta piloto es fabricar los vehículos por primera vez, antes de que comience la producción de volumen, lo que ayuda en la reducción de tiempo de producción y la detección de posibles errores de montaje y diseño.

6. IMS (Information Management System) ITERA

IMS viene de las siglas en inglés Information Management System, es decir, un sistema que es el resultado de la interacción colaborativa entre personas, tecnologías y procedimientos a los que llamamos en su conjunto un “Sistemas de información” y que está orientado a solucionar problemas empresariales. Estos sistemas se diferencian de los sistemas de información comunes en que para analizar la información utilizan otros sistemas que se usan en las actividades operacionales de la organización. Académicamente, el término es comúnmente utilizado para referirse al conjunto de los métodos de gestión de la información vinculada a la automatización o apoyo humano de la toma de decisiones.

6.1. Objetivos

Los objetivos que se plantean dentro de este proyecto son los siguientes:

- Crear un software para unir en una sola base de datos la información de distintas fuentes de información dentro de la empresa, las cuales, si bien están relacionadas en

un plano teórico no tienen una relación explicita cuando el cliente proporciona esta información desde sus diversas fuentes.

- El segundo objetivo será crear una aplicación que incluya la capa de presentación y la capa de la lógica de negocio y que sea fácilmente instalable y accesible por parte de aquellos empleados de la compañía que necesiten usarla.
- Al tener la información centralizada y accesible, el siguiente objetivo será conseguir relacionar la cada información individual con el ingeniero al cual le afecte con lo que se conseguirá tener un "tracking" personal y en consecuencia poder realizar una evaluación de rendimiento basada en este "tracking".
- Cada fuente de información proporciona su propio formato y método de consulta, con lo que uno de los objetivos principales es abstraer la información y presentarla toda con el mismo formato con lo que se pretende que la curva de aprendizaje para cada nueva fuente información que un ingeniero tenga que consultar y que no conozca sea mínima.
- Al centralizar la información y disponer de ella desde una sola aplicación, otro objetivo será el de visualizar la mayor cantidad de información de un ítem sin perder el contexto global.
- Otro de los puntos más importantes de este proyecto es la gestión de información y el poder proporcionar herramientas que filtren la información utilizando búsquedas personalizadas para conseguir una respuesta más rápida a la hora de tomar decisiones basándose en la información del cliente.
- Muchas veces, debido a las propias limitaciones y requisitos que tiene el propio cliente a la hora de almacenar su información, la información está limitada con lo que en el marco de este proyecto también se hará hincapié en poder ampliar la información base añadiendo nueva información relacionada pero solo accesible desde nuestro Sistema de información

6.2. Tecnologías utilizadas

6.2.1. C#

6.2.1.1. Historia

Durante el desarrollo de .NET Framework, las bibliotecas de clases fueron escritos originalmente usando un sistema compilador de código manejado llamado "Simple Managed C (SMC)". En enero de 1999, Anders Hejlsberg formó un equipo para construir un nuevo

lenguaje “Cool” (C-like Object Oriented Language). A la vez que se anunciaba el proyecto .NET en Julio de 2000, este nuevo lenguaje se había renombrado como C# y las bibliotecas de clases y el Runtime de ASP.NET habían sido portados a C #.

El principal diseñador y arquitecto principal de C# en Microsoft es Anders Hejlsberg, que participó previamente con el diseño de Turbo Pascal, Embarcadero Delphi y Visual J ++. En entrevistas y documentos técnicos, Anders Hejlsberg indicó que los defectos en la mayoría de los principales lenguajes de programación llevaron a crear los fundamentos de la Common Language Runtime (CLR), que, a su vez, llevó al diseño del propio lenguaje C#.

C# y Java son dos lenguajes de programación que han nacido basándose en las mismas premisas, muchas veces se ha criticado a C# de ser una copia de Java pero desde el lanzamiento de la versión 2.0 en Noviembre de 2005, C# y Java han sido lenguajes que han ido evolucionado en trayectorias cada vez más divergentes. Una de las primeras grandes diferencias vino con la incorporación de los genéricos. C# hace uso de “Reification” para proporcionar objetos genéricos “First-Class” que se pueden utilizar como cualquier otra clase, generando el código durante el tiempo de carga de la clase. Por el contrario, los genéricos de Java son esencialmente una función de la sintaxis del lenguaje, y no afectan al código generado.

Por su parte, C# ha añadido varias características importantes para dar cabida a la programación de estilo funcional, que culminó con las extensiones de LINQ lanzadas con la versión 3.0 y su framework de expresiones lambda, métodos de extensión y los tipos anónimos. Estas características permiten a los programadores usar técnicas de programación funcionales. Las extensiones de LINQ y de las importaciones funcionales ayudan a los desarrolladores a reducir la cantidad de código "repetitivo" que se incluye en las tareas comunes como la consulta de una base de datos, análisis de un archivo XML, o buscando a través de una estructura de datos, cambiando el énfasis en la lógica del programa real para ayudar a mejorar la legibilidad y mantenibilidad.

C# fue presentada originalmente para su revisión al subcomité ISO JTC 1/ SC 22 bajo la norma ISO/IEC 23270:2003, que ya está retirada, y posteriormente fue aprobada bajo la norma ISO/IEC 23270:2006.

6.2.1.2. Origen del nombre

El nombre de "C#" se inspiró en la notación musical, donde el símbolo de sostenido indica que en la nota en la que se ha escrito debe hacerse un semitono más agudo. Debido a

las limitaciones técnicas de visualización (fuentes estándar, navegadores, etc.) y el hecho de que el símbolo de sostenido no está presente en el teclado estándar, el símbolo # fue elegido para representar el nombre escrito del lenguaje de programación, aunque Microsoft sigue utilizando el símbolo musical en temas de publicidad, marketing, etc.

Este sufijo se ha utilizado por otros lenguajes .NET que son variantes de lenguas existentes, como J# (un lenguaje .NET también diseñado por Microsoft que se deriva de Java 1.1) o A# (de Ada). El sufijo también se ha utilizado para las bibliotecas, como GTK# (un contenedor .NET para GTK + y otros GNOMElibraries) y Cocoa# (un contenedor para Cocoa).

6.2.1.3. Versiones

Version	**Language specification**			**Date**	**.NET Framework**	**Visual Studio**
	ECMA	**ISO/IEC**	**Microsoft**			
C# 1.0	December 2002	April 2003	January 2002	January 2002	.NET Framework 1.0	VS .NET 2002
C# 1.2			October 2003	April 2003	.NET Framework 1.1	VS .NET 2003
C# 2.0	June 2006	September 2006	September 2005	November 2005	.NET Framework 2.0	VS 2005
C# 3.0	None		August 2007	November 2007	.NET Framework 2.0 (Except LINQ / Query Extensions) .NET Framework 3.0 (Except LINQ / Query Extensions) .NET Framework 3.5	VS 2008 & VS 2010
C# 4.0			April 2010	April 2010	.NET Framework 4	VS 2010
C# 5.0			June 2013	August 2012	.NET Framework 4.5	VS 2012 VS 2013
C# 6.0			Status	TBD	.NET Framework 4.6	VS 2015

Tabla 1. Versiones de C#

6.2.1.4. Resumen de la versiones

FEATURES ADDED	
C# 2.0	• Generics • Partial types • Anonymous methods • Iterators • Nullable types • Getter/setter separate accessibility • Method group conversions (delegates)

	• Co- and Contra-variance for delegates • Static classes
C# 3.0	• Implicitly typed local variables • Object and collection initializers • Auto-Implemented properties • Anonymous types • Extension methods • Query expressions • Lambda expressions • Expression tres • Partial methods
C# 4.0	• Dynamic binding • Named and optional arguments • Generic co- and contravariance • Embedded interop types ("NoPIA")
C# 5.0	• Asynchronous methods • Caller info attributes
FUTURE	
C# 6.0 Planned/Done	• Compiler-as-a-service • Import of static type members into namespace • Exception filters • Await in catch/finally blocks • Auto property initializers • Default values for getter-only properties • Expression-bodied members • Null propagator (Succinct null checking) • String Interpolation • nameof operator • Parameterless struct constructors • Dictionary initializer
C# 7.0 Proposals	• Declaration expressions • Parameter arrays for IEnumerable interfaces

Tabla 2. Presente y futuro de las versiones de C#

El lenguaje de programación C# es un lenguaje moderno de programación orientado a objetos, de propósito general, creado y desarrollado por Microsoft junto con la plataforma .NET. El software desarrollado con C# y con la plataforma .NET se utiliza para un abanico muy amplio de soluciones: aplicaciones de oficina, aplicaciones web, sitios web, aplicaciones de escritorio, aplicaciones móviles, juegos y muchos otros. C# es un lenguaje de alto nivel que es similar a Java y C++. Hoy en día C# es uno de los lenguajes de programación más populares. Es utilizado por millones de desarrolladores en todo el mundo.

Para bien o para mal, el lenguaje C # y la plataforma .NET están mantenidos y administrados íntegramente por Microsoft y no están abiertos a terceros. Debido a esto, el resto de las grandes empresas de software como IBM, Oracle y SAP basan sus soluciones en la plataforma Java y el uso de Java como su idioma principal para el desarrollo de sus propios productos de software. A diferencia de C# y .NET Framework, el lenguaje Java y la

plataforma son proyectos de código abierto en la que toda una comunidad de empresas de software, organizaciones y desarrolladores individuales participan. Las normas, las especificaciones y todas las novedades en el mundo de Java son desarrollados por los grupos de trabajo formados de toda la comunidad de Java, en lugar de una sola empresa (como el caso de C # y .NET Framework). El lenguaje C# se distribuye junto con Common Language Runtime (CLR). Este entorno es parte de la plataforma .NET Framework, un conjunto de librerías estándar que ofrecen funcionalidad básica, compiladores, depuradores y otras herramientas de desarrollo.

Gracias al framework CLR los programas que se desarrollen son portables por lo que podrán funcionar con poco o ningún cambio en diversas plataformas hardware y sistemas operativos como MS Windows, teléfonos móviles y otros dispositivos portátiles basados en Windows Mobile, Windows Phone y Windows 8. Los programas en C# no se crean mediante un producto independiente, se utiliza la plataforma Microsoft .NET Framework, que consiste generalmente en un entorno para el desarrollo y ejecución de programas, y se compone de:

- Lenguajes de programación .NET (C#, VB.NET y otros)
- Un entorno para la ejecución de código administrado (CLR), que ejecuta programas de una manera controlada
- Un conjunto de herramientas de desarrollo, como el compilador csc, que convierte los programas de C# en código intermedio (llamado MSIL) que el CLR puede entender
- Un conjunto de librerías estándar, como ADO.NET, que permiten el acceso a las bases de datos (como MS SQL Server o MySQL) y WCF que conecta las aplicaciones a través de marcos de comunicación estándar y protocolos como HTTP zócalos, REST, JSON, SOAP y TCP.

El Framework .NET es parte de todas las distribuciones modernas de Windows y está disponible en diferentes versiones.

6.2.2. ¿C# o Java?

C# es un lenguaje de programación moderno, ampliamente difundido, utilizado por millones de programadores de todo el mundo. Al mismo tiempo C# es más simple y fácil de aprender que, por ejemplo, C y C++. Al ser, a la vez, un lenguaje adecuado para principiantes y un que sigue siendo ampliamente utilizado en la industria por muchas grandes empresas, por lo que es uno de los lenguajes de programación más populares hoy en día.

Java es el competidor directo para C# y, aunque es discutible, sus características lo definen como más poderoso, más rico y mejor diseñado. Elegimos C #, porque es más fácil de aprender y se distribuye con un la posibilidad de un entorno de desarrollo libre (aunque existen versiones de pago que introducen más características en el entorno de desarrollo).

6.2.2.1. Ventajas de C#

C# es un lenguaje de programación orientado a objetos como otros lenguajes de programación modernos (Java y C ++). Para ver las ventajas de la programación orientada a objetos podemos pensar en lenguajes orientados a objetos como lenguajes que permiten trabajar con objetos del mundo real (por ejemplo, estudiante, escuela, libro y otros). Los objetos tienen propiedades (por ejemplo, nombre, color, etc.) y pueden realizar acciones (por ejemplo, mover, hablar, etc.).

6.2.2.2. Comparación C# vs Java

Para explicar la elección del uso del lenguaje C# en este proyecto en vez de otros lo vamos a comparar con su más directo competidor que además es uno de los lenguajes más extendidos y utilizados dentro de cómo es Java.

6.2.2.3. Tipos

Data types	**Java**	**C#**
Single-root (unified) type system	No; but wrapper types	Yes
Signed integers	Yes; 8, 16, 32, 64 bits	Yes; 8, 16, 32, 64 bits
Unsigned integers	No; but some method support.	Yes; 8, 16, 32, 64 bits
Character	Yes	Yes
Date/time	Yes; reference type	Yes; value type
IEEE 754 binary32 floating point number	Yes	Yes
IEEE 754 binary64 floating point number	Yes	Yes

Boolean type	Yes	Yes
High precision decimal number	No; but see 'Arbitrary size decimals' below	128-bit (28 digits) Decimal type
Arbitrary size decimals	Reference type; no operators	No
Strings	Immutable reference type, Unicode	Immutable reference type, Unicode
Arbitrary size integers	Reference type; no operators	Yes
Complex numbers	No	Yes
Reference types	Yes	Yes
Arrays	Yes	Yes
Value types	No; only primitive types	Yes
Type annotations	Yes	Yes
Enumerated types	Yes; reference type	Yes; scalar
Lifted (nullable) types	No; but wrapper types	Yes
Tuples	No; limited 3rd party available.	No; but standard library support with limited arity
Pointers	No; only method references	Yes

Tabla 3. Comparación de tipos entre Java y C#

6.2.2.4. Sintaxis

En general, las sintaxis de las lenguas son muy similares. La sintaxis a nivel de estado y de expresión es casi idéntica a la inspiración obvia de la C / C ++ tradición. A nivel de definición de tipo existen (clases e interfaces) algunas pequeñas diferencias. Java es explícita sobre extender las clases y la implementación de interfaces, mientras que C# deduce esto de la clase de tipos de la clase/interfaz de la que deriva.

C# soporta más características que Java, que en cierta medida es también evidente en la sintaxis que especifica más palabras clave y más reglas gramaticales que Java.

6.2.2.5. Programación orientada a objetos

Tanto C # y Java están diseñados desde el principio como lenguajes orientados a objetos mediante el Dynamic Dispatch, con una sintaxis similar a C ++ (C ++, a su vez deriva de C).

Object orientation	**Java**	**C#**
Classes	mandatory	mandatory
Interfaces	Yes	Yes
Abstract classes	Yes	Yes
Member accessibility levels	Yes; public, package, protected, private	Yes; public, internal, protected, private, protected internal
Class-level inner classes	Yes;static inner classes are class level	Yes; all inner classes are class level
Instance-level inner classes	Yes	No
Statement-level (local) anonymous classes	Yes	Yes; Without methods
Partial classes	No; but see AspectJ	Yes
Implicit (inferred) anonymous classes	No	Yes
Deprecation/obsolescence	Yes	Yes
Overload versioning	Some	Yes
Enums can implement interfaces	Yes	No
Properties	No, but see JavaBeans spec	Yes
Events	Provided by standard libraries	Built-in language feature
Operator overloading	No	Yes
Indexers	No	Yes

Implicit conversions	No	Yes
Explicit conversions	No	Yes

Tabla 4. Comparación entre Java y C#

6.2.2.6. Exceptions

Exceptions	**Java**	**C#**
Checked exceptions	Yes	No
Try-catch-finally	Yes	Yes

Tabla 5. Comparación excepciones entre Java y C#

6.2.2.7. Try-catch-finally

También hay diferencias entre las dos lenguas en el tratamiento de la try-finally. El bloque finally se ejecuta siempre, incluso si el bloque try contiene sentencias de control de paso como "throw" o "return". En Java, esto puede provocar un comportamiento inesperado, si el bloque try y en el bloque finally se escriben instrucciones "return" con valores diferentes, en cambio en C# no se permite esto mediante la prohibición de las sentencias de control de paso en el bloque finally.

Una razón común para el uso de try-finally bloques es proteger los recursos de gestión de código, lo que garantiza la liberación de esos recurso. C# permite el uso del método Dispose() del objeto para la liberación de recursos.

Una diferencia más sutil es el momento en que se crea una traza de la pila cuando se produce una excepción. En Java, el "stack-trace" se crea en el momento en que se crea la excepción, en cambio en C# se crea el "stack-trace" el momento de ejecutar el "throw".

6.2.2.8. Soporte en distintas plataformas

Platform support	**Java**	**C#**
Linux	Yes	via Mono
Mac OS X	Yes	via Mono
Solaris	Yes	via Mono
FreeBSD	Yes	via Mono

AIX	Yes	Partial?
iOS	via RoboVM or Codename One	via Xamarin MonoTouch
Windows	Yes	Yes
Windows Mobile	Yes	Yes
Windows Phone	via Codename One	Yes
Windows 8 Modern	Yes	Yes
Android	via Dalvik, ART	via Xamarin MonoDroid
Feature phones	Yes	No
Symbian	Yes	Deprecated
Blackberry	Yes	via cross-compiler

Tabla 6. Soporte de plataformas para Java y C#

6.2.2.9. Entornos de tiempo de ejecución

Java (lenguaje de programación) está diseñado para ejecutarse en la plataforma Java a través del Java Runtime Environment (JRE). La plataforma Java incluye la Máquina Virtual Java (JVM), así como un conjunto común de las bibliotecas. El JRE se diseñó originalmente para apoyar la ejecución interpretarse con compilación final como opción. La mayoría de los entornos JRE ejecutan totalmente o al menos parcialmente los programas compilados, posiblemente con la optimización adaptativa. El compilador de Java genera el bytecode de Java.

C# está diseñado para ejecutarse en el Common Language Runtime (CLR). El CLR está diseñado para ejecutar código totalmente compilado. El compilador de C# produce instrucciones Common Intermediate Language (CLI). Entonces se carga este código y compila para instrucciones de la máquina en la arquitectura objetivo.

6.2.3. WPF

Es un subsistema gráfico para la prestación de interfaces de usuario en aplicaciones basadas en Windows de Microsoft. WPF, anteriormente conocido como "Avalon", fue lanzado inicialmente como parte de .NET Framework 3.0. En lugar de confiar en el viejo subsistema GDI, WPF utiliza DirectX. WPF intenta proporcionar un modelo de

programación coherente para la creación de aplicaciones y separa la interfaz de usuario de la lógica de negocio. Se asemeja a otros objetos de modelos basados en XML, como las implementadas en XUL y SVG.

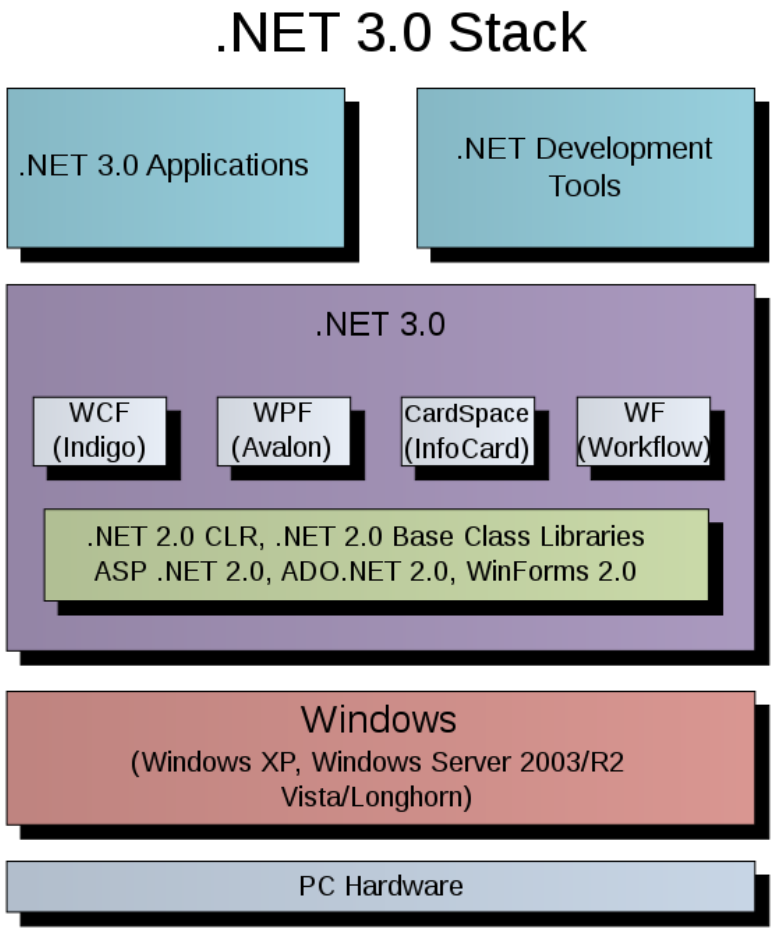


Ilustración 1. Arquitectura WPF

WPF XAML emplea, un lenguaje basado en XML, para definir y vincular diferentes elementos de la interfaz. Las aplicaciones WPF también se pueden implementar como programas de escritorio independientes o como objetos embebidos en una página web. WPF pretende unificar una serie de elementos comunes de la interfaz de usuario, tales como la representación 2D/3D, los documentos fijos y adaptativos, tipografía, gráficos vectoriales, animaciones en tiempo de ejecución. Estos elementos pueden ser vinculados y manipulados basándose en varios eventos, las interacciones del usuario y enlaces de datos.

XAML forma parte de Microsoft Windows Presentation Foundation (WPF). WPF es la categoría de características de Microsoft .NET Framework 3.5 relacionadas con la presentación visual de aplicaciones basadas en Windows y de aplicaciones cliente basadas en exploradores web.

Las aplicaciones basadas en WPF se pueden ejecutar en Windows Vista o en versiones anteriores de Windows siempre que esté instalado Microsoft .NET Framework 3.5 (e Internet Explorer 7.0 en el caso de aplicaciones cliente basadas en exploradores web).

6.2.3.1. Historia y futuro

WPF fue lanzado inicialmente como parte de .NET Framework 3.0. Desde entonces, Microsoft ha publicado cinco versiones principales: WPF 3.0 (noviembre 2006), WPF 3.5 (noviembre 2007), WPF 3.5sp1 (agosto 2008), WPF 4 (abril de 2010), y WPF 4.5 (agosto de 2012).

De acuerdo con los desarrolladores de Microsoft respondiendo preguntas durante Build 2014, WPF está siendo desarrollado activamente, pero no fue diseñado para dar una respuesta rápida, ni adaptado a dispositivos de baja potencia de consumo, tales como tabletas o teléfonos inteligentes. Debido a eso, no se esperan cambios importantes más por hacer en el Framework.

6.2.3.2. Características

Los Gráficos se representan usando Direct3D. Esto permite la visualización de los más gráficos complejos y temas personalizados. Permite a Windows enviar algunas tareas de gráficos directamente a la GPU con lo que se reduce la carga de trabajo de la CPU.

Windows Presentation Foundation (WPF) es el marco de interfaz de usuario de Microsoft para crear aplicaciones con una experiencia de usuario rica. Es parte del marco .NET 3.0 y superiores. El énfasis de WPF en gráficos vectoriales permite la mayoría que los controles y elementos se puedan escalar sin pérdida de calidad o pixelización, aumentando así la accesibilidad. Con la excepción de Silverlight, la integración Direct3D permite representación 3D simplificada. Además, el contenido interactivo en 2D puede superponerse a superficies 3D de forma nativa.

6.2.3.3. Data Binding

- WPF tiene un conjunto integrado de servicios de datos para permitir a los desarrolladores de aplicaciones manipular datos dentro de las aplicaciones. Es compatible con cuatro tipos de enlace de datos:

    - One Time: cuando el cliente hace caso omiso de las actualizaciones en el servidor.
    - One Way: de sólo lectura, donde el cliente tiene acceso a los datos.
    - Two Way: donde el cliente puede leer y escribir datos en el servidor
    - One Way To Source: donde el cliente tiene sólo acceso de escritura a los datos
- Consultas LINQ, incluyendo LINQ to XML, también pueden actuar como fuente de datos para el enlace de datos.
- La unión de los datos no tiene relación con su presentación. WPF proporciona plantillas de datos para controlar la presentación de los datos.
- Un conjunto de controles integrados se proporciona como parte de WPF, que contiene elementos tales como botones, menús, rejillas, y cuadro de lista.
- Un concepto de WPF es la separación lógica entre un control y su aspecto.
- Una plantilla de un control puede ser completamente reescrita y cambiar su aspecto visual.
- Un control puede contener cualquier otro control o diseño, lo que permite un alto grado de control sobre la composición.
- Repintado la pantalla no siempre es necesario.

6.2.3.4. Media Services

- El WPF proporciona un sistema integrado para la creación de interfaces de usuario con elementos comunes de diseño imágenes vectoriales y raster, audio y vídeo. WPF también proporciona un sistema de animación y un sistema de renderizado 2D/3D.
- WPF proporciona primitivas para gráficos en 2D junto con un conjunto integrado de pinceles, lápices, geometrías y transformaciones.
- Las capacidades 3D de WPF son un subconjunto del conjunto de funciones completas proporcionada por Direct3D. Sin embargo, WPF proporciona una mayor integración con otras características como interfaces de usuario, documentos y medios de comunicación. Esto hace que sea posible tener interfaces de usuario 3D, documentos en 3D,....
- Existe soporte para los formatos más comunes de imagen: BMP, JPEG, PNG, TIFF, Windows Media Photo, GIF e ICO.
- WPF soporta video en formato WMV, MPEG y algunos archivos AVI por defecto. WPF puede usar todos los codecs instalados.

6.2.3.5. XAML

WPF introduce XMAL como código para su programación. XAML está diseñado como un método más eficiente de desarrollar interfaces de usuario de aplicaciones y está basado en el lenguaje XML.

La ventaja específica que aporta es que es un lenguaje totalmente declarativo, permitiendo al desarrollador describir el comportamiento y la integración de los componentes sin el uso de la programación mediante procedimientos. El uso de XAML para desarrollar interfaces de usuario también permite la separación del modelo y de la vista, que es uno de los principios que estamos siguiendo en el este proyecto software.

WPF usa XAML para crear interfaces de usuario (UI) de una excelente calidad visual en lenguaje de marcado en lugar de lenguaje de programación como, por ejemplo, C#. Puede crear documentos elaborados de UI totalmente en XAML con elementos tales como controles, texto, imágenes, formas, animación y mucho más. Como XAML es declarativo (al igual que HTML), se requerirá la adición de código si fuera necesario agregar lógica en tiempo de ejecución a la aplicación. Por ejemplo, si la aplicación sólo usa XAML, se pueden crear y animar elementos de la interfaz de usuario y configurarlos para que respondan de un modo limitado a los datos proporcionados por el usuario (mediante desencadenadores de eventos), pero la aplicación no podrá realizar cálculos ni responderlos, ni podrá crear espontáneamente nuevos elementos de UI sin la adición de código. El código de una aplicación XAML se almacena en un archivo distinto del documento XAML. El hecho de que el diseño de la interfaz de usuario esté separado del código subyacente permite a programadores y diseñadores trabajar juntos en el mismo proyecto sin interferir mutuamente en su trabajo.

Aunque XAML no es necesario para crear la interfaz ya que todos sus elementos de WPF pueden ser escritos en un lenguaje .NET (C #, VB.NET). El código XAML se puede convertir en código ensamblador de la misma manera todos los lenguajes .NET.

6.2.3.6. Extensible Application Markup Language (XAML)

Filename extension	.xaml
Internet media type	application/xaml+xml
Developed by	Microsoft

Initial release	v1.0 / June 2008
Latest release	v2009 (16 April 2010; 5 years ago[2][3])
Type of format	User interface markup language
Extended from	XML

Tabla 7. Resumen XAML

6.2.3.7. Architecture

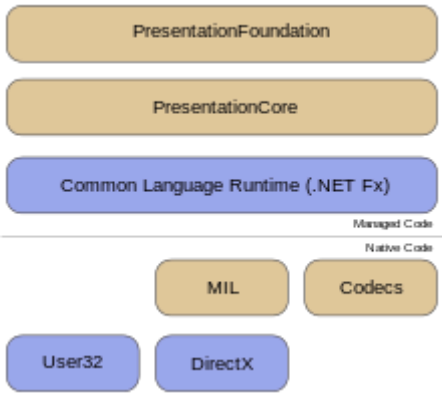


Ilustración 2. Arquitectura WPF

La mayoría de WPF es en código administrado, pero el "composition engine" que hace el renderizado de las aplicaciones WPF es un componente nativo. Lleva el nombre de Media Integration Layer (MIL) y reside en la librería milcore.dll. Este conecta directamente con DirectX y proporciona soporte básico para superficies 2D y 3D, la manipulación controlada por temporizador del contenido de una superficie con el fin de exponer las construcciones de animación a un nivel superior, y la composición de los elementos individuales de una aplicación WPF en un escena 3D final que representa la interfaz de usuario de la aplicación y la renderiza en la pantalla.

Los elementos de la interfaz de una aplicación WPF se mantienen como una clase de objetos visuales. Los objetos visuales proporcionan una interfaz administrada en un "Composition Tree" que es mantenido por la Media Integration Layer (MIL). Cada elemento de WPF crea y añade uno o más nodos a ese árbol. Los nodos de composición contienen instrucciones de representación, como las instrucciones de clipping y transformación, además de otros atributos visuales.

Por lo tanto toda la aplicación se representa como una colección de nodos de composición, que se almacenan en una memoria intermedia en la memoria del sistema. Periódicamente, MIL recorre el árbol y ejecuta las instrucciones de renderizado en cada nodo que se representa a continuación en la pantalla. MIL utiliza el algoritmo del pintor (Painter's algorithm) en el que todos los componentes se renderizan desde la parte posterior de la pantalla hasta la parte frontal, lo que permite efectos complejos como transparencias. Este proceso de renderizado es acelerado por hardware usando la GPU. El árbol de la composición se almacena en caché por el MIL, por lo que cualquier cambio en el árbol de composición sólo necesita ser incrementalmente comunicado a MIL. Esto también libera a las aplicaciones de gestión de repintar la pantalla; MIL puede hacer eso en sí ya que tiene toda la información necesaria. Las animaciones pueden ser implementados como cambios de tiempo activadas en el árbol de composición. En el lado visible de usuario, las animaciones se especifican de forma declarativa, mediante el establecimiento de algún efecto de animación en algún elemento a través de una propiedad y especificando la duración. El código subyacente actualiza los nodos específicos del árbol, a través de objetos visuales, para representar tanto los estados intermedios a intervalos de tiempo especificados, así como el estado final del elemento. MIL hará que los cambios en el elemento de forma automática.

Todas las aplicaciones WPF comienzan con dos hilos: uno para la gestión de la interfaz de usuario y otro subproceso de fondo para el manejo de la representación y el repintado Renderezida y repintado está gestionado en WPF por sí mismo, sin la intervención del desarrollador. El hilo de interfaz de usuario contiene el Dispatcher (a través de una instancia de DispatcherObject), que mantiene una cola de las operaciones de interfaz de usuario que deban llevarse a cabo clasificadas por prioridad.

Los eventos de interfaz de usuario, incluyendo el cambio de una propiedad que afecta a la distribución, y eventos de interacción de usuario planteadas están en cola del Dispatcher(), que invoca los manejadores de los eventos. Microsoft recomienda que en los controladores de eventos sólo se actualicen las propiedades para reflejar nuevos contenidos como respuesta de las aplicaciones; el nuevo contenido se genera o se recupera en un subproceso en segundo plano. El hilo de renderizado también almacena en caché el árbol visual, por lo que sólo los cambios en el árbol deben ser comunicados, lo que se traducirá en la actualización únicamente de los píxeles modificados.

WPF apoya un modelo de diseño extensible. El layout se divide en dos fases: Medir y Organizar. La fase de Medida llama de forma recursiva a todos los elementos y determina el tamaño que tendrán. En la fase de Organizar, los elementos secundarios se organizan de forma recurrente por sus padres.

6.2.4. Mahapps

Mahapps.Metro es un toolkit de Interfaz de Usuario para WPF que le da una apariencia Modern UI (anteriormente llamado Metro). Modern UI es el nombre de la interfaz de usuario (UI), desarrollado inicialmente por Microsoft para su uso en Windows Phone. Podemos decir que es una interfaz plana, con colores básicos y diseños geométricos, con una movilidad horizontal (en PC) o vertical (en móviles). Está optimizada para su uso en pantallas táctiles. Más tarde, los principios de la interfaz de usuario Modern se introdujeron en la Xbox 360 y Windows 8.

Mahapps anula el estilo de los controles por defecto y les da un aspecto de parecido al estilo Modern UI. Este toolkit también incluye algunos controles personalizados basados en los conceptos de Windows Phone y Windows 8 Apps.

Mahapps es código abierto y está alojado GitHub para su libre distribución. Este proyecto, que Paul Jenkins comenzó en 2011, intenta ser una forma sencilla de llevar a una interfaz de usuario de estilo Modern a la aplicación WPF. Desde entonces ha evolucionado y ha tenido numerosas contribuciones de otros usuarios de la comunidad (en el último recuento había más de 600).

6.3. Capas de la aplicación

6.3.1. Introducción

En la ingeniería de software, la arquitectura de varios niveles (arquitectura n-tier) es una arquitectura cliente-servidor en el que las funciones de presentación, procesamiento de aplicaciones y gestión de datos están separados física o lógicamente. El uso más extendido de la arquitectura de varios niveles es la arquitectura de tres capas.

La arquitectura de varios niveles de la aplicación proporciona un modelo por el cual los desarrolladores pueden crear aplicaciones flexibles y reutilizables. Gracias a la división en varios niveles, los desarrolladores adquieren la opción de modificar o añadir una capa específica, en lugar de volver a trabajar toda la aplicación. Una arquitectura de tres niveles se compone típicamente de un nivel de presentación, una capa de lógica de dominio, y un nivel de almacenamiento de datos.

6.3.2. Interfaz

Este es el nivel superior de la aplicación. El nivel de presentación muestra la información. En términos simples, es una capa en que los usuarios pueden acceder directamente, como una página web, o un GUI sistemas operativos.

6.3.3. Lógica de negocio

Nivel de aplicación es la lógica de negocio. El nivel lógico controla la funcionalidad de una aplicación mediante el procesamiento detallado de datos.

6.3.4. Datos

El nivel de datos incluye los mecanismos de persistencia de datos (servidores de bases de datos, archivos compartidos, etc.) y la capa de acceso a datos que encapsula los mecanismos de persistencia y expone los datos. La capa de acceso de datos debe proporcionar una interfaz de programación de aplicaciones (API) para el nivel de aplicación que expone los métodos de gestión de los datos almacenados sin exponer o la creación de dependencias de los mecanismos de almacenamiento de datos. Al igual que con la separación de cualquier nivel, hay costos de implementación y, a menudo cuesta al rendimiento a cambio de una mejor escalabilidad y facilidad de mantenimiento.

6.3.5. Arquitectura 3 capas

La arquitectura de tres niveles es un patrón de arquitectura de software cliente-servidor en el que la interfaz de usuario (“presentación”), la lógica de proceso funcional ("reglas de negocio"), el almacenamiento de los datos en ordenadores con su respectivo acceso a datos está desarrollada y mantenida como módulos independientes.

Aparte de las ventajas habituales de software modular con interfaces bien definidas, la arquitectura de tres niveles tiene la intención de permitir que, debido a los cambios de requisitos o tecnología, la modificación solo afecte a la capa implicada. Por ejemplo, un cambio de sistema operativo en el nivel de presentación sólo afectaría el código de interfaz de usuario.

Típicamente, la interfaz de usuario se ejecuta en un PC de escritorio o estación de trabajo y utiliza una interfaz estándar gráfica de usuario, la lógica de proceso funcional que

puede consistir en uno o más módulos separados que se ejecutan en un servidor de estación de trabajo o aplicación, y un sistema de gestión en un servidor de base de datos o una unidad central que contiene la lógica de almacenamiento de datos informáticos.

Ilustración 3. Arquitectura 3 capas

Comúnmente esta arquitectura de tres capas se usa en el campo del desarrollo web, los tres niveles se utiliza a menudo para referirse a sitios web, comúnmente sitios web de comercio electrónico:

- Un servidor web front-end proporciona el contenido estático, y potencialmente algún contenido dinámico en caché. En aplicación basada en web, front-end es el contenido representado por el navegador.
- Un servidor de aplicaciones de procesamiento de contenido y nivel de generación dinámica, por ejemplo, Ruby on Rails, Java EE, ASP.NET, PHP, ColdFusion, Perl, Python plataforma.
- Una base de datos back-end o almacén de datos, que comprende dos conjuntos de datos y el software de sistema de gestión de base de datos que gestiona y proporciona acceso a los datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares. El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

- Presentación. (Conocida como capa Web en aplicaciones Web o como capa de usuario en Aplicaciones Nativas)
- Lógica de Negocio. (Conocida como capa Aplicativa)
- Datos. (Conocida como capa de Base de Datos)

En cambio, el término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice que la arquitectura de la solución es de tres capas y *un nivel*.
- Una solución de tres capas (presentación, lógica del negocio, datos) que residen en dos ordenadores (Presentación+lógica por un lado; lógica+datos por el otro lado). Se dice que la arquitectura de la solución es de tres capas y *dos niveles*.

Para nuestra solución vamos a adaptar la solución de una arquitectura de tres capas a los requisitos que marca la realización de este proyecto. Utilizaremos una aproximación de tres capas y dos niveles. La capa de datos estará ubicada en un servidor accesible a través de la red local y la capa de lógica de negocio (aplicación realizada en C#) y la capa de presentación (Interfaz de Usuario en WPF) se encontrarán en cada ordenador de cada usuario.

6.4. 1ª FASE (primera aproximación, requisitos + diseño)

La solución software que se ha creado ha sido el resultado final de un proceso de desarrollo software. En este proceso de desarrollo software se ha seguido un "ciclo de vida en cascada", donde las flechas indican el orden en que se van realizando las actividades. Este modelo está en desuso, pero sigue siendo adecuado para identificar las actividades principales y el orden natural entre ellas.

Se utiliza este ciclo ya que, aún en desuso como hemos comentado, es el más aproximado para el desarrollo software a pequeña escala, en el que cada fase tiene la opción de volver a la anterior mostrando que en los desarrollos software a pequeña escala es muy complicado, debido a la falta de recursos y medios, poder diferenciar y validar cada fase.

Análisis de requisitos

Se estudian las necesidades de los usuarios, se decide qué debe hacer la aplicación informática para satisfacerlas en todo o en parte, y se genera un Documento de Requisitos.

Diseño de la arquitectura

Se estudia el Documento de Requisitos y se establece la estructura global de la aplicación, descomponiéndola en partes (módulos, subsistemas) relativamente independientes. Se genera un Documento de Diseño.

Diseño detallado

En esta segunda parte de la actividad de diseño se fijan las funciones de cada módulo, con el detalle de su interfaz. Se genera el código de declaración (o especificación) de cada módulo.

Codificación

Se desarrolla el código de cada módulo.

Pruebas de unidades

Como complemento de la codificación, cada módulo o grupo de módulos se prueba por separado. En las pruebas se comprueba si cada módulo cumple con su especificación de diseño detallado.

Pruebas de integración

Se hace funcionar la aplicación completa, combinando todos sus módulos. Se realizan ensayos para comprobar que el funcionamiento de conjunto cumple lo establecido en el documento de diseño.

Pruebas de validación

Como paso final de la integración se realizan nuevas pruebas de la aplicación en su conjunto. En este caso el objetivo es comprobar que el producto desarrollado cumple con lo establecido en el documento de requisitos, y satisface por tanto las necesidades de los usuarios en la medida prevista.

Fase de mantenimiento

No hay actividades diferenciadas de las anteriores. El mantenimiento del producto exige rehacer parte del trabajo inicial, que puede corresponder a cualquiera de las actividades de las etapas anteriores.

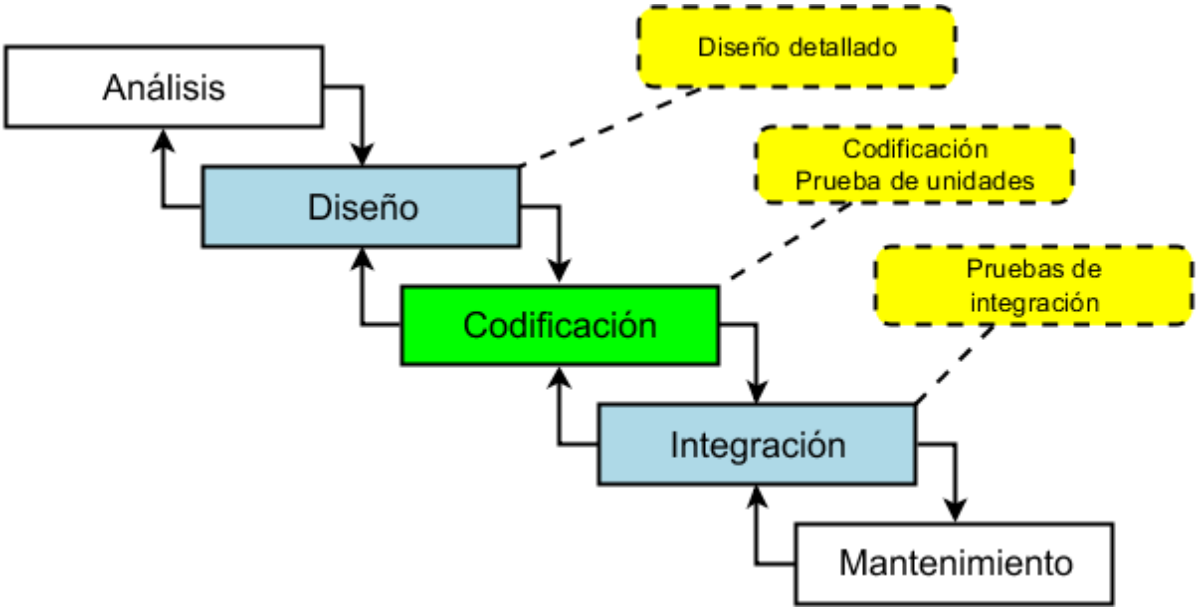


Ilustración 4. Ciclo desarrollo software

Cabe diferenciar que se va detallar los puntos más importantes de cada etapa de desarrollo diferenciándola de las fases de desarrollo, es decir, la redacción de este proyecto se va a centrar en explicar en qué punto estaba cada fase del desarrollo software durante los cambios más significativos de la creación de la aplicación.

6.4.1. Toma de requisitos

A la hora de empezar este proyecto se mantuvieron una serie de reuniones para establecer los requisitos de la misma. Estas reuniones fueron el comienzo del desarrollo software y estuvieron presentes los responsables de la empresa, los cuales conocían las particularidades del entorno en donde se iba a implementar la aplicación.

De las primeras reuniones se concluyó que el proyecto necesitaría tener los siguientes requisitos funcionales:

- La aplicación debe poder gestionar una gran cantidad de información
- Debido a las limtaciones de Ford Motor Company que es donde se ejecutará la aplicación se utilizará Access y todas sus opciones.
- La aplicación tiene que poder ejecutarse como minimo en los ordenadores que pertenezcan al sitema donde los usuarios vayan a desarrollar su trabajo
- No se podrá tener acceso a BBDD ubicadas en servidores
- No se podrá instalar como software independiente en los ordenadores de los trabajadores
- Debe tener un formato sencillo y fácil de entender

- Debe recuperar información de diferentes fuentes (terminales Mainframe PC3270, archivos Excel, …)
- Se almacenará dos tipos de información, la que provenga de la fuente de información y la que proporcione el usuario. La primera será de solo lectura y la segunda de lectura y escritura.
- La aplicación debe tener un control de usuarios
- La aplicación debe poder mostrar la información filtrada según solicite el usuario

6.4.2. Requisitos librerías ehllapi

Para realizar la actualización de información se intentó acceder directamente a las BBDD que disponía Ford pero no se consiguieron los permisos por lo que las únicas formas de obtener la información eran a través de la lectura de archivos Microsoft Office Excel y la lectura de los emuladores de terminales Mainframe.

Para esto último se necesitaba disponer del software de emulación IBM Personal Communications que dispone de las librerías ehllapi con las que se puede actuar sobre el emulador para mandar códigos, funciones o recuperar información.

EHLLAPI es una interfaz de programación estándar que permite el acceso mediante programación a una sesión del emulador. Las funciones se proporcionan para poder hacer una lectura de los datos de la pantalla (tales como los caracteres y atributos), para el envío de pulsaciones de teclado, y la realización de otras funciones relacionadas con el propio emulador.

La interfaz EHLLAPI es interfaz “single-call point” . Hay una sola API a través del cual se solicita a todas las funciones EHLLAPI . En cada llamada a la interfaz la aplicación proporciona un número de función que identifica la función solicitada, un puntero a un búfer de datos, un puntero a la longitud de la memoria intermedia de datos, y un puntero a un código de retorno.

En el anexo de este documento podéis encontrar las guías de referencia de la librería Ehllapi proporcionadas por Ford y por IBM.

6.4.3. Diseño en Access (BBDD + Formularios)

Teniendo en cuenta los requisitos mencionados en el punto 6.4.1, se decidió que la primera aproximación sería mediante el uso de Microsoft Access, tanto para guardar como

mostrar y modificar los datos mediante formularios e informes. Esto, pese a ser una solución factible, presentaba grades limitaciones a la hora de diseñar todas las funciones que requería la aplicación.

El primer problema es que desde Microsoft Access 2007, el software ya no implementaba un control de usuarios nativo. La siguiente cosa que cabía señalar era la pobre implementación del apartado de interfaz gráfica, muy limitada.

Al estudiar esta opción se tomaron dos vías de desarrollo

1. Seguir creando la estructura de tablas de BBDD en Access.
2. En paralelo intentar conseguir la autorización para poder instalar algún entorno de desarrollo software en ordenadores pertenecientes a la estructura Ford para poder implantar una aplicación nativa y desde cero.

6.4.4. Diseño gestor de actualización de contenido dentro de Access

Como uno de los requisitos es no poder acceder a las BBDD corporativas de Ford de forma directa, a la hora de actualizar el contenido se necesita crear un proceso con formatos previamente validados.

En esta primera etapa solo se ha planteado la actualización del contenido que se recupera a través de conectarse al emulador del terminal Mainframe. Estas actualizaciones son completas sin ninguna opción para el usuario. Cuando el usuario lanza la actualización, se actualiza todo lo que corresponda que esté en la BBDD.

6.5. 2ª FASE (revisión y aproximación solución consensuada, requisitos + diseño)

En esta segunda fase se produce un cambio significativo al cambiar varios requisitos de la primera fase, el más importante es conseguir las herramientas dentro de Ford Motor Company para poder desarrollar una aplicación. Estas modificaciones se detallan a continuación.

6.5.1. Modificación de requisitos

Los responsables de la empresa consiguieron la autorización por parte de Ford para poder instalar un entorno de desarrollo software. Esto implicaba que uno de los requisitos que teníamos inicialmente en cuanto a las limitaciones que nos encontrábamos dentro de Ford Motor Company cambiaba por lo que los requisitos también cambian.

El nuevo requisito es:

- Se puede hacer uso de entornos de desarrollo software autorizados por Ford Motor Company como VS Express C++ 2010.

6.5.2. Estudio soluciones desarrollo software a utilizar (VS 2010, VS 2013 Express, ...)

El poder utilizar un entorno de desarrollo como VS cambiaba sustancialmente la idea inicial al tener mucha más libertad de desarrollo pero la versión que ofrecía Ford era una versión antigua y con muy pocas características y se llegó a un segundo acuerdo con Ford que permitió poder utilizar la versión más reciente de Visual Studio (VS 2013 Express).

6.5.3. Diseño capas (DB, software, interfaz)

Este cambio significaba que se podía pensar en el diseño de la aplicación desde otro punto de vista y se optó por un diseño de dos niveles y tres capas como ya hemos visto en el punto 6.3.

Como se ha comentado vamos a adaptar la solución de una arquitectura de tres capas a los requisitos que marca la realización de este proyecto. Utilizaremos una aproximación de tres capas y dos niveles. La capa de datos estará ubicada en un servidor accesible a través de la red local y la capa de lógica de negocio (aplicación realizada en C#) y la capa de presentación (Interfaz de Usuario en WPF) se encontrarán en cada ordenador de cada usuario.

6.5.4. Rediseño gestor de actualización de contenidos integrándolo con la app

La parte más importante del gestor de contenidos es el poder hacer uso de las funciones de la librería EHLLAPI de IBM que nos permite interactuar con el emulador del terminal Mainframe para acceder a la información de Ford Motor Company.

A parte se decidió poder actualizar más información mediante el uso de archivos Access y Excel generados mediante sistemas internos de Ford Motor Company.

Para el uso de la librería EHLLAPI dentro de la aplicación se comprobó la compatibilidad con el proyecto C# y se creó una clase que fuese la encargada de llamar a las funciones de la librería externa. Esta solución se basa en que si aparece una nueva librería o esta sufre alguna modificación tan solo tendremos que cambiar la importación de la librería pero nada más en todo el código.

A continuación está la clase creada a tal propósito en la que tenemos todas las funciones necesarias para interactuar con el emulador del terminal como Conectar, Desconectar, Escribir, Leer, ...

```
using System;
using System.Runtime.InteropServices;
using System.Text;

namespace EHLLAPI
{
    /// <summary>
    ///
    /// </summary>
    public class EhllapiFunc
    {
        /// <summary>
        /// Librería DLL EHLAPI
        /// </summary>
        /// <param name="Func">Función que se envia</param>
        /// <param name="Data">Datos</param>
        /// <param name="Length">Longitud de los datos</param>
        /// <param name="RetC">Valor devuelto</param>
        /// <returns>Código de ejecución (OK, ERROR, ... - Mirar Documentación ELHAPI
Return Codes)</returns>
        [DllImport("EHLAPI32.DLL")]
        public static extern UInt32 hllapi(out UInt32 Func, StringBuilder Data, out
UInt32 Length, out UInt32 RetC);
    }

    /// <summary>
    /// Clase que implementa las funciones a utilizar con la librería DLL
    /// </summary>
    public class EhllapiWrapper
    {
```

```
#region EHLLAPI FUNC CODES

const UInt32
    HA_CONNECT_PS = 1; /*000 Connect PS*/
const UInt32
    HA_DISCONNECT_PS = 2; /*000 Disconnect PS*/
const UInt32
    HA_SENDKEY = 3; /*000 Sendkey function*/
const UInt32
    HA_WAIT = 4; /*000 Wait function*/
const UInt32
    HA_COPY_PS = 5; /*000 Copy PS function*/
const UInt32
    HA_SEARCH_PS = 6; /*000 Search PS function*/
const UInt32
    HA_QUERY_CURSOR_LOC = 7; /*000 Query Cursor*/
const UInt32
    HA_COPY_PS_TO_STR = 8; /*000 Copy PS to String*/
const UInt32
    HA_SET_SESSION_PARMS = 9; /*000 Set Session*/
const UInt32
    HA_QUERY_SESSIONS = 10; /*000 Query Sessions*/
const UInt32
    HA_RESERVE = 11; /*000 Reserve function*/
const UInt32
    HA_RELEASE = 12; /*000 Release function*/
const UInt32
    HA_COPY_OIA = 13; /*000 Copy OIA function*/
const UInt32
    HA_QUERY_FIELD_ATTR = 14; /*000 Query Field*/
const UInt32
    HA_COPY_STR_TO_PS = 15; /*000 Copy string to PS*/
const UInt32
    HA_STORAGE_MGR = 17; /*000 Storage Manager*/
const UInt32
    HA_PAUSE = 18; /*000 Pause function*/
const UInt32
    HA_QUERY_SYSTEM = 20; /*000 Query System*/
const UInt32
    HA_RESET_SYSTEM = 21; /*000 Reset System*/
const UInt32
    HA_QUERY_SESSION_STATUS = 22; /*000 Query Session*/
const UInt32
    HA_START_HOST_NOTIFY = 23; /*000 Start Host*/
const UInt32
    HA_QUERY_HOST_UPDATE = 24; /*000 Query Host Update*/
const UInt32
    HA_STOP_HOST_NOTIFY = 25; /*000 Stop Host*/
const UInt32
    HA_SEARCH_FIELD = 30; /*000 Search Field*/
const UInt32
    HA_FIND_FIELD_POS = 31; /*000 Find Field*/
const UInt32
    HA_FIND_FIELD_LEN = 32; /*000 Find Field Length*/
const UInt32
    HA_COPY_STR_TO_FIELD = 33; /*000 Copy String to*/
const UInt32
    HA_COPY_FIELD_TO_STR = 34; /*000 Copy Field to*/
const UInt32
    HA_SET_CURSOR = 40; /*000 Set Cursor*/
const UInt32
    HA_START_CLOSE_INTERCEPT = 41; /*000 Start Close Intercept*/
```

```
const UInt32
    HA_QUERY_CLOSE_INTERCEPT = 42; /*000 Query Close Intercept*/
const UInt32
    HA_STOP_CLOSE_INTERCEPT = 43; /*000 Stop Close Intercept*/
const UInt32
    HA_START_KEY_INTERCEPT = 50; /*000 Start Keystroke*/
const UInt32
    HA_GET_KEY = 51; /*000 Get Key function*/
const UInt32
    HA_POST_INTERCEPT_STATUS = 52; /*000 Post Intercept*/
const UInt32
    HA_STOP_KEY_INTERCEPT = 53; /*000 Stop Keystroke*/
const UInt32
    HA_LOCK_PS = 60; /*000 Lock Presentation*/
const UInt32
    HA_LOCK_PMSVC = 61; /*000 Lock PM Window*/
const UInt32
    HA_SEND_FILE = 90; /*000 Send File function*/
const UInt32
    HA_RECEIVE_FILE = 91; /*000 Receive file*/
const UInt32
    HA_CONVERT_POS_ROW_COL = 99; /*000 Convert Position*/
const UInt32
    HA_CONNECT_PM_SRVCS = 101; /*000 Connect For*/
const UInt32
    HA_DISCONNECT_PM_SRVCS = 102; /*000 Disconnect From*/
const UInt32
    HA_QUERY_WINDOW_COORDS = 103; /*000 Query Presentation*/
const UInt32
    HA_PM_WINDOW_STATUS = 104; /*000 PM Window Status*/
const UInt32
    HA_CHANGE_SWITCH_NAME = 105; /*000 Change Switch List*/
const UInt32
    HA_CHANGE_WINDOW_NAME = 106; /*000 Change PS Window*/
const UInt32
    HA_START_PLAYING_MACRO = 110; /*000 Start playing macro*/
const UInt32
    HA_START_STRUCTURED_FLD = 120; /*000 Start Structured*/
const UInt32
    HA_STOP_STRUCTURED_FLD = 121; /*000 Stop Structured*/
const UInt32
    HA_QUERY_BUFFER_SIZE = 122; /*000 Query Communications*/
const UInt32
    HA_ALLOCATE_COMMO_BUFF = 123; /*000 Allocate*/
const UInt32
    HA_FREE_COMMO_BUFF = 124; /*000 Free Communications*/
const UInt32
    HA_GET_ASYNC_COMPLETION = 125; /*000 Get Asynchronous*/
const UInt32
    HA_READ_STRUCTURED_FLD = 126; /*000 Read Structured Field*/
const UInt32
    HA_WRITE_STRUCTURED_FLD = 127; /*000 Write Structured*/

#endregion

#region EHLLAPI RETURN CODES

const UInt32
    HARC_SUCCESS = 0; /*000 Good return code.*/
const UInt32
    HARC99_INVALID_INP = 0; /*000 Incorrect input*/
const UInt32
```

```
    HARC_INVALID_PS = 1; /*000 Invalid PS, Not*/
const UInt32
    HARC_BAD_PARM = 2; /*000 Bad parameter, or*/
const UInt32
    HARC_BUSY = 4; /*000 PS is busy return*/
const UInt32
    HARC_LOCKED = 5; /*000 PS is LOCKed, or*/
const UInt32
    HARC_TRUNCATION = 6; /*000 Truncation*/
const UInt32
    HARC_INVALID_PS_POS = 7; /*000 Invalid PS*/
const UInt32
    HARC_NO_PRIOR_START = 8; /*000 No prior start*/
const UInt32
    HARC_SYSTEM_ERROR = 9; /*000 A system error*/
const UInt32
    HARC_UNSUPPORTED = 10; /*000 Invalid or*/
const UInt32
    HARC_UNAVAILABLE = 11; /*000 Resource is*/
const UInt32
    HARC_SESSION_STOPPED = 12; /*000 Session has*/
const UInt32
    HARC_BAD_MNEMONIC = 20; /*000 Illegal mnemonic*/
const UInt32
    HARC_OIA_UPDATE = 21; /*000 A OIA update*/
const UInt32
    HARC_PS_UPDATE = 22; /*000 A PS update*/
const UInt32
    HARC_PS_AND_OIA_UPDATE = 23; /* A PS and OIA update*/
const UInt32
    HARC_STR_NOT_FOUND_UNFM_PS = 24; /*000 String not found,*/
const UInt32
    HARC_NO_KEYS_AVAIL = 25; /*000 No keys available*/
const UInt32
    HARC_HOST_UPDATE = 26; /*000 A HOST update*/
const UInt32
    HARC_FIELD_LEN_ZERO = 28; /*000 Field length = 0*/
const UInt32
    HARC_QUEUE_OVERFLOW = 31; /*000 Keystroke queue*/
const UInt32
    HARC_ANOTHER_CONNECTION = 32; /*000 Successful. Another*/
const UInt32
    HARC_INBOUND_CANCELLED = 34; /*000 Inbound structured*/
const UInt32
    HARC_OUTBOUND_CANCELLED = 35; /*000 Outbound structured*/
const UInt32
    HARC_CONTACT_LOST = 36; /*000 Contact with the*/
const UInt32
    HARC_INBOUND_DISABLED = 37; /*000 Host structured field*/
const UInt32
    HARC_FUNCTION_INCOMPLETE = 38; /*000 Requested Asynchronous*/
const UInt32
    HARC_DDM_ALREADY_EXISTS = 39; /*000 Request for DDM*/
const UInt32
    HARC_ASYNC_REQUESTS_OUT = 40; /*000 Disconnect successful.*/
const UInt32
    HARC_MEMORY_IN_USE = 41; /*000 Memory cannot be freed*/
const UInt32
    HARC_NO_MATCH = 42; /*000 No pending*/
const UInt32
    HARC_OPTION_INVALID = 43; /*000 Option requested is*/
const UInt32
```

```
            HARC99_INVALID_PS = 9998; /*000 An invalid PS id*/
        const UInt32
            HARC99_INVALID_CONV_OPT = 9999; /*000 Invalid convert*/

        #endregion

        /// <summary>
        /// Conexión a la ventana PCOMM
        /// </summary>
        /// <param name="sessionID">Nombre de la sesión de la ventana</param>
        /// <returns>Código de ejecución (OK, ERROR, ... - Mirar Documentación ELHAPI
Return Codes)</returns>
        public static UInt32 Connect(string sessionID)
        {
            StringBuilder Data = new StringBuilder(4);
            Data.Append(sessionID);
            UInt32 rc = 0;
            UInt32 f = HA_CONNECT_PS;
            UInt32 l = 4;
            return EhllapiFunc.hllapi(out f, Data, out l, out rc);
        }

        /// <summary>
        /// Desconexión de la ventana PCOMM
        /// </summary>
        /// <param name="sessionID">Nombre de la sesión de la ventana</param>
        /// <returns>Código de ejecución (OK, ERROR, ... - Mirar Documentación ELHAPI
Return Codes)</returns>
        public static UInt32 Disconnect(string sessionID)
        {
            StringBuilder Data = new StringBuilder(4);
            Data.Append(sessionID);
            UInt32 rc = 0;
            UInt32 f = HA_DISCONNECT_PS;
            UInt32 l = 4;
            return EhllapiFunc.hllapi(out f, Data, out l, out rc);
        }

        /// <summary>
        /// Establece la posición del cursor en la ventana del Terminal
        /// </summary>
        /// <param name="p">Posición absoluta en la ventana</param>
        /// <returns>Código de ejecución (OK, ERROR, ... - Mirar Documentación ELHAPI
Return Codes)</returns>
        public static UInt32 SetCursorPos(int p)
        {
            StringBuilder Data = new StringBuilder(0);
            UInt32 rc = (UInt32)p;
            UInt32 f = HA_SET_CURSOR;
            UInt32 l = 0;
            UInt32 result = EhllapiFunc.hllapi(out f, Data, out l, out rc);
            Wait();
            return result;
        }

        /// <summary>
        /// Obtiene la posición del cursor en la ventana del Terminal
        /// </summary>
        /// <param name="p">Posición absoluta en la ventana</param>
        /// <returns>Código de ejecución (OK, ERROR, ... - Mirar Documentación ELHAPI
Return Codes)</returns>
        public static UInt32 GetCursorPos(out int p)
```

```
        {
            StringBuilder Data = new StringBuilder(0);
            UInt32 rc = 0;
            UInt32 f = HA_QUERY_CURSOR_LOC;
            UInt32 l = 0; //return position
            UInt32 r = EhllapiFunc.hllapi(out f, Data, out l, out rc);
            p = (int)l;
            Wait();
            return r;
        }

        /// <summary>
        /// Envio de una cadena de carácteres a la ventana del Terminal
        /// </summary>
        /// <param name="cmd">Cadena de carácteres que se envia al Terminal</param>
        /// <returns>Código de ejecución (OK, ERROR, ... - Mirar Documentación ELHAPI
Return Codes)</returns>
        public static UInt32 SendStr(string cmd)
        {
            StringBuilder Data = new StringBuilder(cmd.Length);
            Data.Append(cmd);
            UInt32 rc = 0;
            UInt32 f = HA_SENDKEY;
            UInt32 l = (UInt32)cmd.Length;
            UInt32 result = EhllapiFunc.hllapi(out f, Data, out l, out rc);
            Wait();
            return result;
        }

        /// <summary>
        /// Lectura de una parte específica dentro del Terminal
        /// </summary>
        /// <param name="position">Psoción absoluta donde empieza la lectura</param>
        /// <param name="len">Longitud a recuperar</param>
        /// <param name="txt">Cadena donde se guardará el texto recuperado</param>
        /// <returns>Código de ejecución (OK, ERROR, ... - Mirar Documentación ELHAPI
Return Codes)</returns>
        public static UInt32 ReadScreen(int position, int len, out string txt)
        {
            StringBuilder Data = new StringBuilder(len);
            UInt32 rc = (UInt32)position;
            UInt32 f = HA_COPY_PS_TO_STR;
            UInt32 l = (UInt32)len;
            UInt32 r = EhllapiFunc.hllapi(out f, Data, out l, out rc);
            txt = Data.ToString();
            return r;
        }

        /// <summary>
        /// The Wait function checks the status of the host-connected presentation
space. If the session is waiting for a host response (indicated by XCLOCK (X []) or
XSYSTEM), the Wait function causes EHLLAPI to wait up to 1 minute to see if the
condition clears.
        /// </summary>
        /// <returns>
        /// 0 - The keyboard is unlocked and ready for input.
        /// 1 - Your application program is not connected to a valid session.
        /// 4 - Timeout while still in XCLOCK (X []) or XSYSTEM.
        /// 5 - The keyboard is locked.
        /// 9 - A system error was encountered.
        /// </returns>
        public static UInt32 Wait()
```

```
        {
            StringBuilder Data = new StringBuilder(0);
            UInt32 rc = 0;
            UInt32 f = HA_WAIT;
            UInt32 l = 0;
            UInt32 r = EhllapiFunc.hllapi(out f, Data, out l, out rc);
            return r;
        }
    }
}
```

6.6. 3ª FASE (implementación versión alfa)

6.6.1. Implementación BBDD

La BBDD se diseñó como una capa que se ubicaría en un recurso de red. Inicialmente se diseñó para que cada fuente de información pudiese relacionarse con un usuario y con una parte del coche. Tenemos 7 diferentes fuentes de información (Events, Concerns, Alerts, AIMS, BSAQ, Issues Itera, BOM) y cada usuario y cada parte del coche tiene una relación 1 a muchos con cada fuente de información.

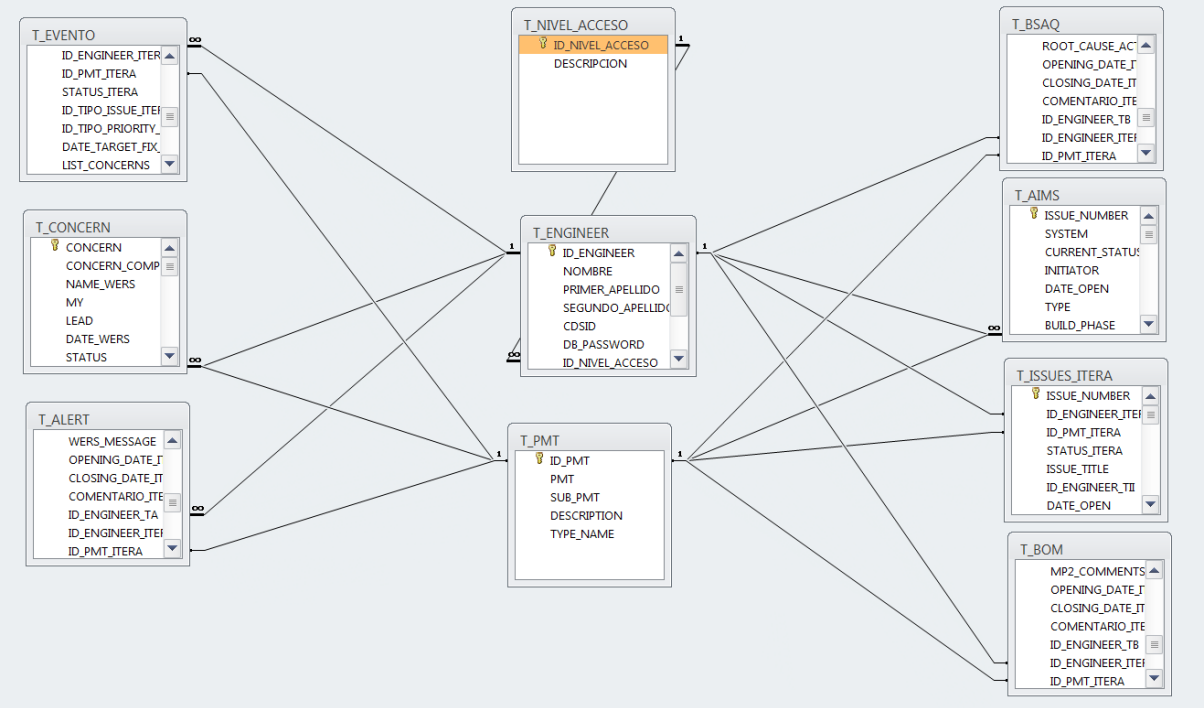


Ilustración 5. Detalle BBDD

6.6.2. Implementación plantillas software

Para la capa de lógica de negocio, que es donde nos encontramos las funciones con las que gestionamos la información, se diseñó únicamente para un tipo de fuente de información, en este caso para la parte de Concerns. Se hizo de esta manera porque el diseño estaba planteado para que la implementación utilizase la misma plantilla software para todas las fuentes de información minimizando así la personalización de cada fuente y pueden ser flexibles y rápidos a la hora de añadir, modificar o quitar diferentes fuentes de información.

La plantilla software tiene 8 regiones:

- Variables de clase: contiene todas las variables globales que se van a utilizar dentro de la clase
- Constructor de clase
- Inicialización: contiene todas las funciones que se van a ejecutar cuando se invoque
- Reset: contienes las funciones, uno vez invocado el objeto, para poder resetear las variables de la clase a su situación inicial (como a la hora de resetear los filtros que se incluyen dentro de la clase)
- Load: llama a las funciones que, según la elección del usuario, devolverán la información para poder visualizarla y trabajar con ella en la interfaz
- Eventos: contiene todas las funciones de eventos relacionados con la capa de interfaz de usuario
- Backgroundworker: función que consulta a la BBDD y carga la información en memoria. Se ejecuta en un hilo separado del hilo principal para evitar que el programa se quede congelado mientras se realiza la búsqueda de información.
- Otras funciones: Conjunto de otras funciones de ayuda que no están categorizadas en ninguno de los apartados anteriores

En los archivos adjuntos a este proyecto se puede encontrar un ejemplo de la plantilla software utilizado para el tipo de datos “Concerns”, así como la implementación del proyecto software.

6.6.3. Implementación interfaz

Para la capa de interfaz se usó WPF del que hemos hablado en el punto 6.2.3. Al igual que para la capa de lógica de negocio, el diseño de la interfaz se realizó de acuerdo a una plantilla para que todas las fuentes de información externas que se introdujesen en la

aplicación tuviesen una gestión de la información como se puede ver más adelante en el punto 7.

Al igual que en el punto anterior, en los archivos adjuntos a este proyecto se puede encontrar un ejemplo de la plantilla software utilizado para la interfaz del tipo de datos "Concerns", así como la implementación del proyecto software.

6.7. 4ª Fase del proyecto (cambios importantes en la implementación)

6.7.1. Modificación diseño capa interfaz

En las primeras versiones de la aplicación se usó el estilo por defecto de WPF, este estilo, si bien proporciona todas las características deseadas es simple y antiguo. Así que se optó por realizar un cambio profundo a la interfaz utilizando la librería externa Mahapps que mejora notablemente el aspecto visual acercando los componentes utilizados a los últimos estilos de Microsoft.

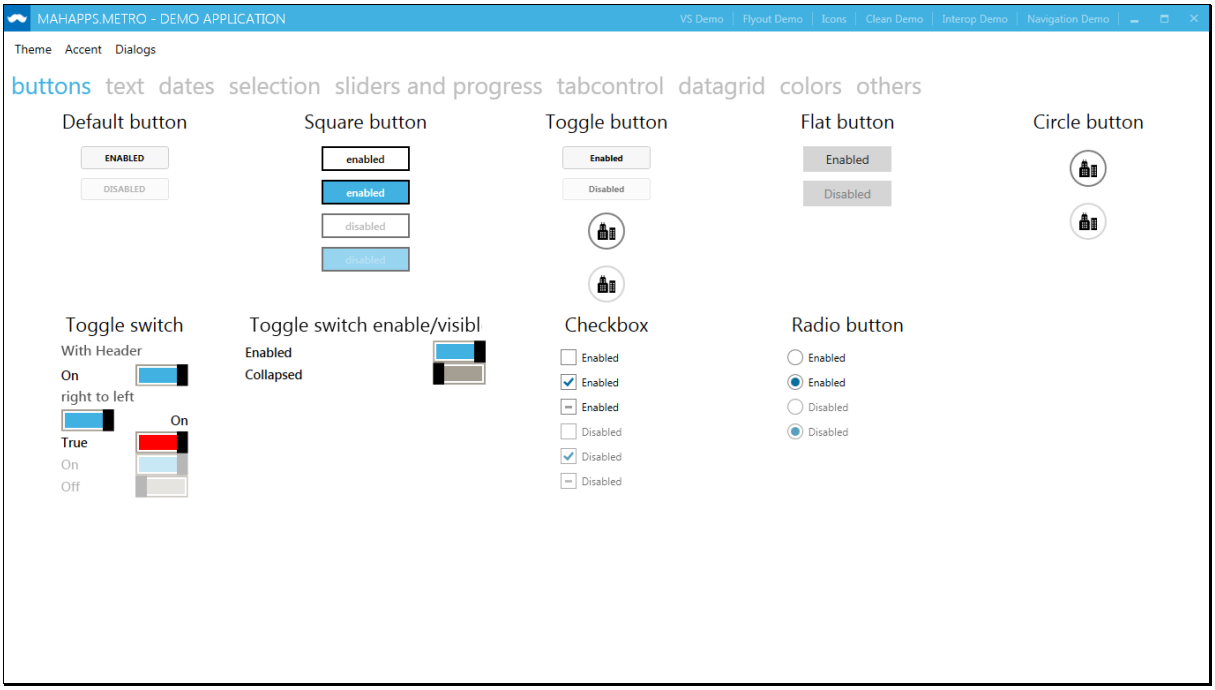


Ilustración 6. Ejemplo de diferentes componentes utilizando Mahapps.

6.7.2. Uso librería mahapps

Se puede instalar MahApps.Metro a través de la interfaz gráfica de usuario NuGet (Administrar Paquetes NuGet dentro de VS).

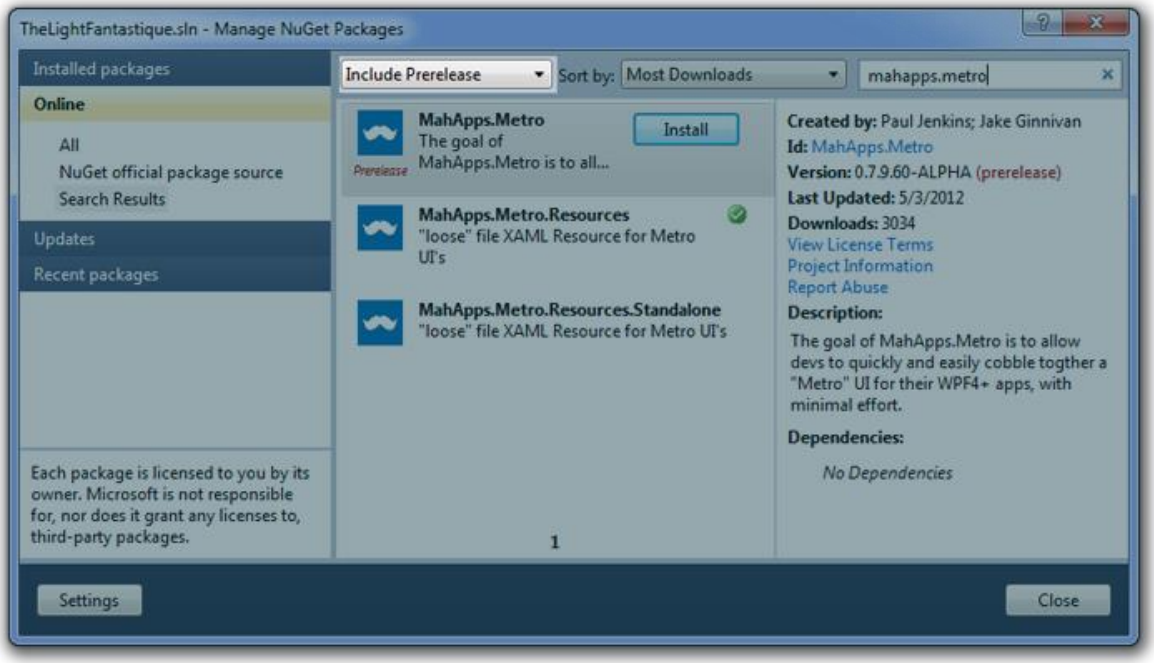


Ilustración 7. Administrador de paquetes VS

O utilizar la consola de Administrador de paquetes:

```
Install-Package MahApps.Metro -Pre
```

Existen dos formas para cambiar el estilo de su ventana usando MahApps.Metro:

- Utilizando el control incluido “MetroWindow”
- Diseñando su propia ventana

Por ahora vamos a utilizar MetroWindow ya que este enfoque funciona para un buen porcentaje de aplicaciones y es la manera más rápida y fácil de continuar el proyecto.

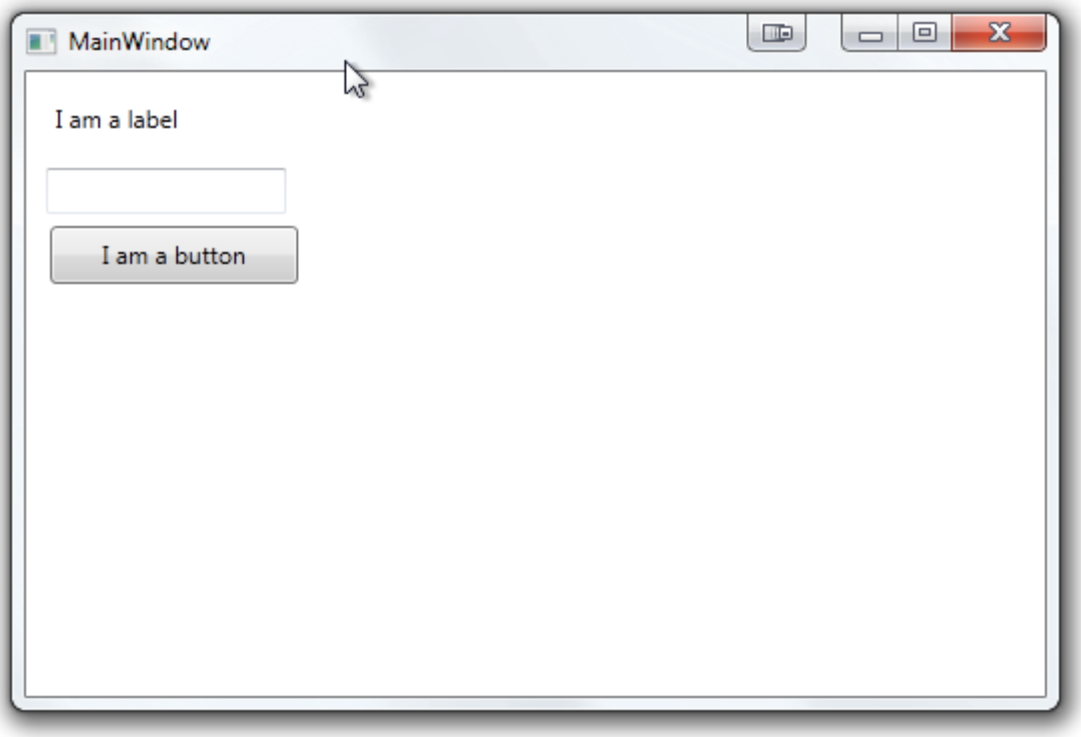


Ilustración 8. Ejemplo ventana WPF

Podemos realizar cambios sobre la ventana modificando el archivo XAML, después de instalar MahApps.Metro:

- Abrir MainWindow.xaml
- Añadir este atributo en la etiqueta de la ventana de apertura. (Así es como se hace referencia a otros espacios de nombres en XAML):
  `xmlns:Controls="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"`
  o
  `xmlns:Controls="http://metro.mahapps.com/winfx/xaml/controls"`
- cambiar la etiqueta "<Window…" por "<Controls:MetroWindow…" (cambiar la etiqueta de cierre también)

Un ejemplo de como quedaría sería:

```
<Controls:MetroWindow x:Class= "WpfApplication2.MainWindow" xmlns= "http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x= "http://schemas.microsoft.com/winfx/2006/xaml" xmlns:Controls= "clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro" Title= "MainWindow" Height= "350" Width= "525" > <!-- your content here --> </Controls:MetroWindow>
```

Para que todo funcione también se necesitará modificar el “MainWindow.xaml.cs” archivo de la clase base “MainWindow”. Para acceder a MetroWindow agregue la siguiente referencia en primer lugar.

```
// using statements...
using MahApps.Metro.Controls

public partial class MainWindow : public partial class MainWindow { // ... }
```

El resultado final será como la Ilustración 9 a continuación.

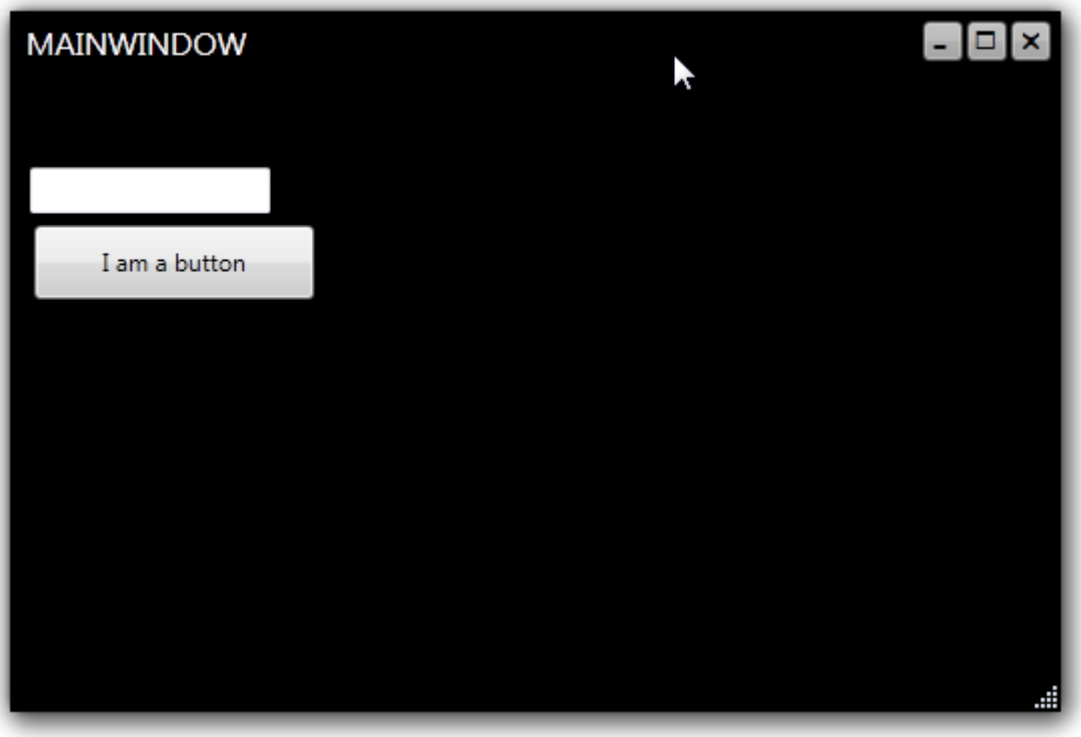


Ilustración 9. Ejemplo ventan Mahapps 1

Todos los recursos de MahApp.Metro están contenidos dentro de diccionarios de recursos independientes. Por lo que habrá que añadir los siguientes “ResourceDictionaries” al archivo “App.xaml”

```
<Application.Resources> <ResourceDictionary> <ResourceDictionary.MergedDictionaries>
<ResourceDictionary Source=
"pack://application:,,,/MahApps.Metro;component/Styles/Controls.xaml" />
```

```
<ResourceDictionary Source=
"pack://application:,,,/MahApps.Metro;component/Styles/Fonts.xaml" />
<ResourceDictionary Source=
"pack://application:,,,/MahApps.Metro;component/Styles/Colors.xaml" />
<ResourceDictionary Source=
"pack://application:,,,/MahApps.Metro;component/Styles/Accents/Blue.xaml" />
<ResourceDictionary Source=
"pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseLight.xaml" />
</ResourceDictionary.MergedDictionaries> </ResourceDictionary>
</Application.Resources>
```

El resultado final será como la Ilustración 10 a continuación.

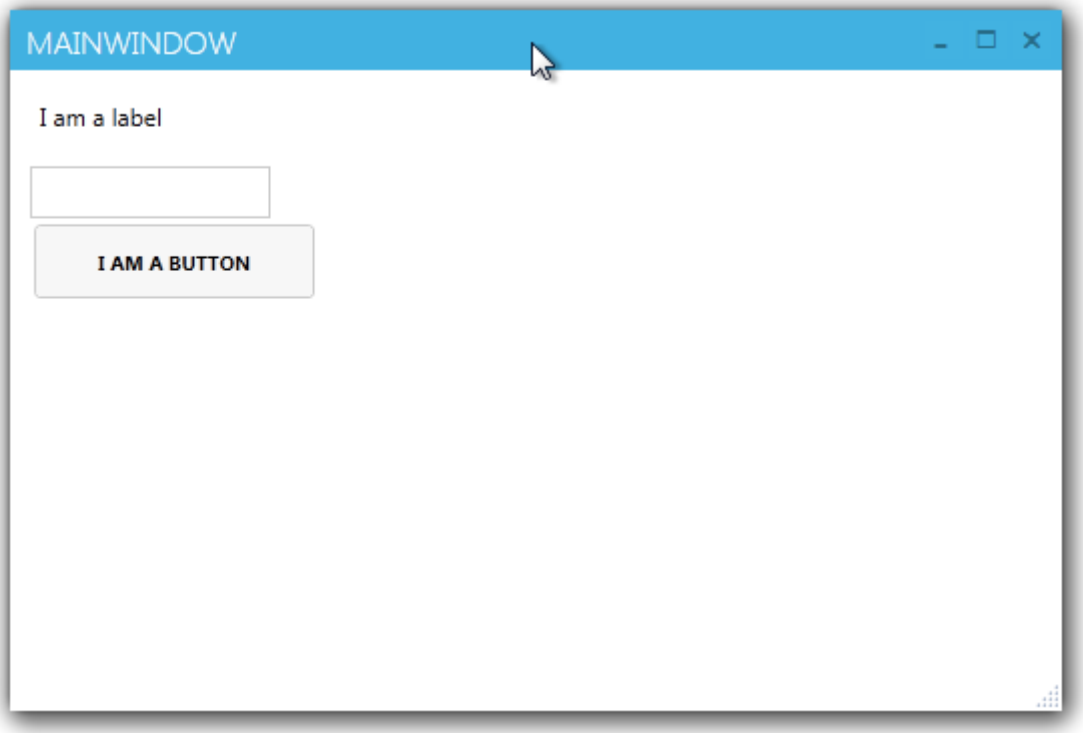


Ilustración 10. Ejemplo ventana Mahapps 2

El valor predeterminado MetroWindow se compone de unos pocos componentes:

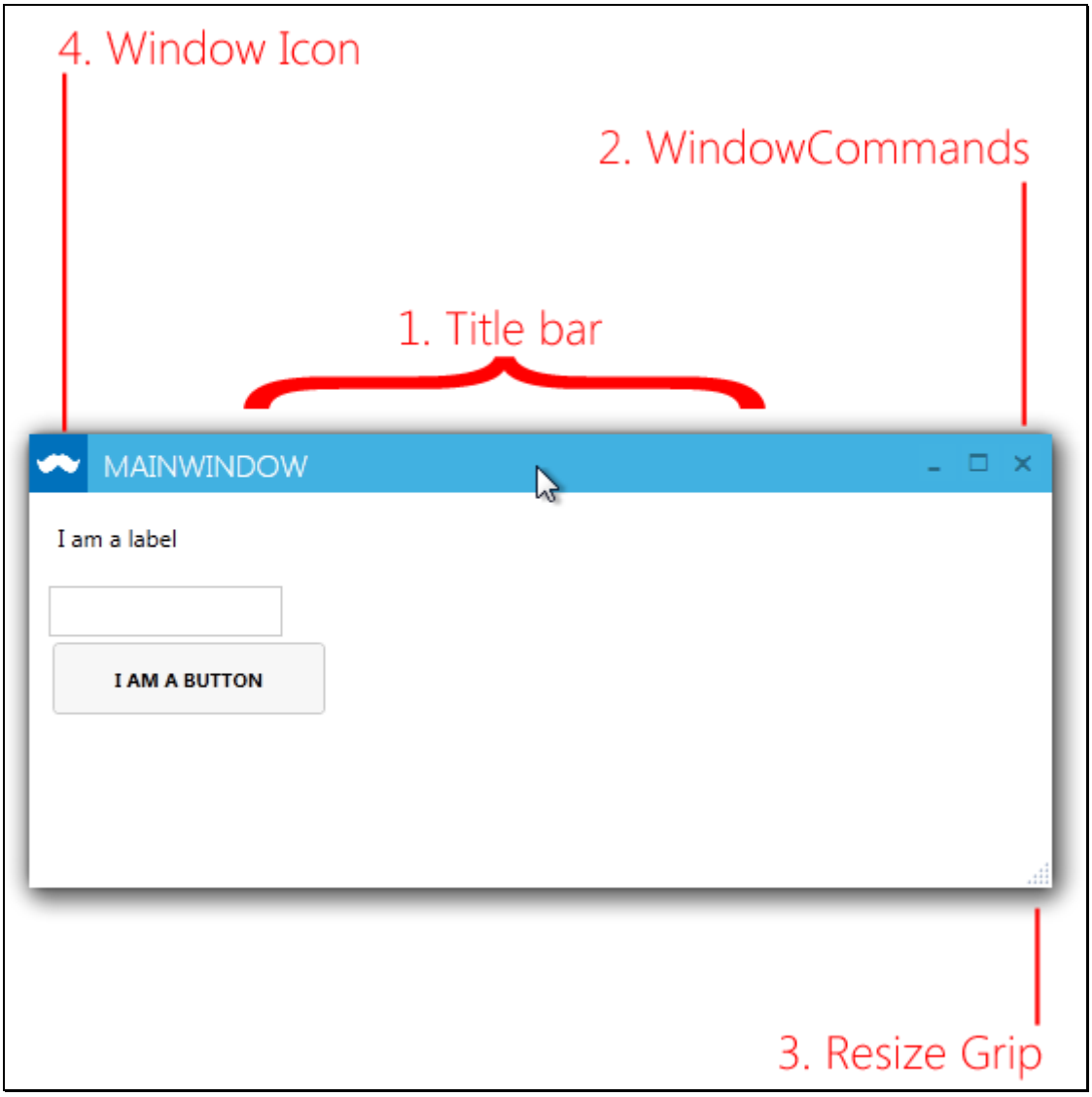


Ilustración 11. Detalle ventana Mahapps

- La barra de título es lo que diferencia a “MetroWindow”.
    - ShowTitleBar="True|False"
- El cambio de tamaño no es la única manera de cambiar el tamaño, todos los bordes y las esquinas se pueden seleccionar

6.8. 5ª Fase del proyecto (fase lanzamiento)

6.8.1. Uso por parte de usuarios finales para obtener un feedback

Al tener la aplicación bastante avanzada en su desarrollo e implementación, se distribuyó entre los usuarios. La idea inicial era la de utilizar algunos usuarios para realizar tests pero se realizó una reunión sobre este tema y se concluyó que se debía lanzar para todos los usuarios a la misma vez por el poco tiempo que disponían para hacer labores de testing.

Así pues todos los usuarios finales (alrededor de unos 60) se convirtieron también en los usuarios testing encargados de proporcionar el primer feedback, informando de fallos, mejoras, etc.

6.8.2. Modificaciones de requisitos y diseño según feedback usuarios

Para tener una vía sencilla para proporcionar este feedback, se optó por añadir en el menú de la aplicación de un apartado “Report a problem” que lleva al usuario a un formulario de Google.

En este formulario, el usuario de la aplicación, puede informar de un error de un elemento inexistente, de un resultado que no es correcto, o también puede informar de mejoras como son añadir filtros en los documentos, mejoras la usabilidad,…

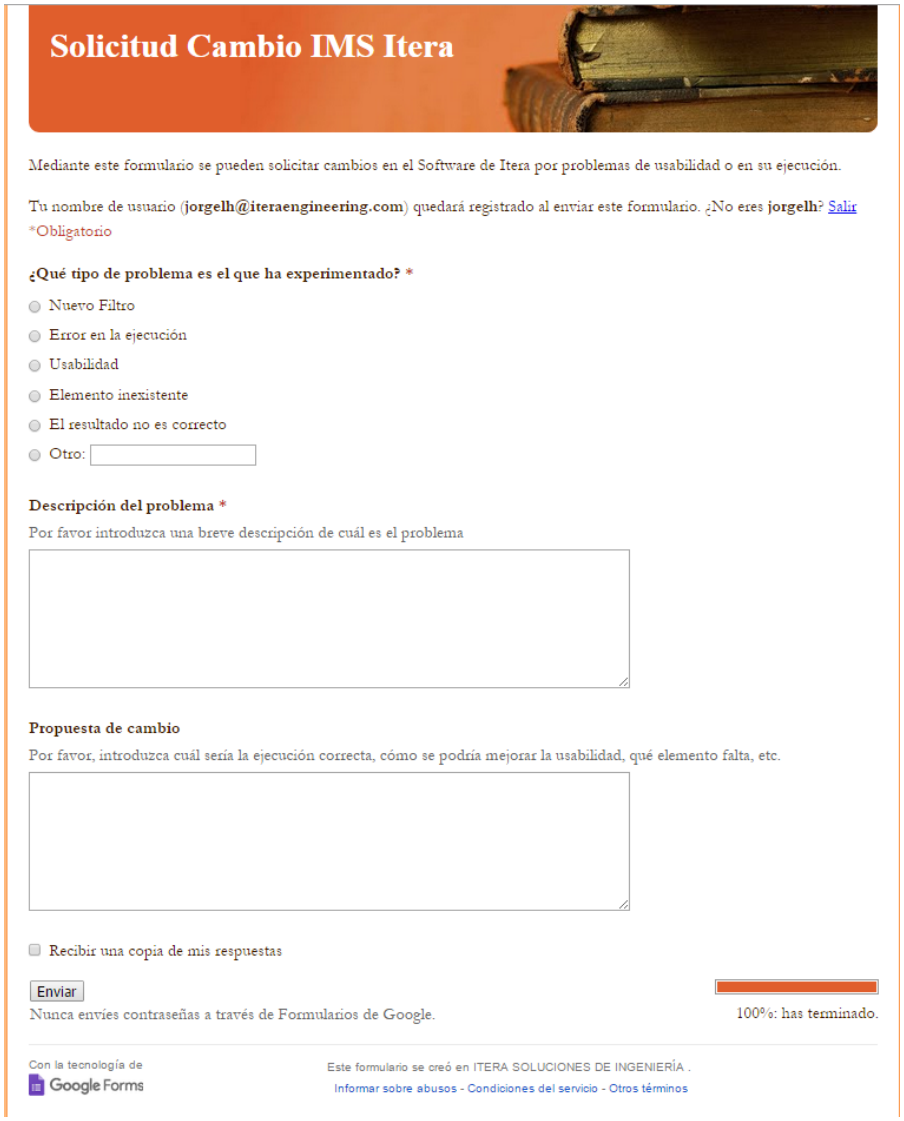


Ilustración 12. Formulario feedback IMS

6.9. 6ª Fase del proyecto

6.9.1. Validación, verificación y mantenimiento

Una vez lanzada la aplicación y estando en uso por parte de los usuarios finales se pasó a la parte de validación y verificación. La versión 1.8.1.0 se convirtió en la versión estable con todas las funciones que se acordaron en los requisitos para la primera parte del proyecto. Esta versión también se verificó comprobando que sus funciones no presentaban ningún fallo importante.

La versión 1.8.1.0 se establece como la versión correspondiente a la primera parte del proyecto. A la hora de mejorar la aplicación y hacerla más potente se ha empezado a preparar una segunda versión que incorporará mejoras, sobretodo, en los reportes que puede obtener la aplicación. Esta segunda versión será la base para establecer una serie de indicadores automáticos del trabajo de los usuarios y se añadirán en la parte "Dashboard" de la aplicación.

Por último es importante señalar que la aplicación requiere un mantenimiento, sobretodo del archivo Access, el cual es importante compactar y reparar con cierta asiduidad para que el tamaño de la BBDD no crezca demasiado.

7. MANUAL DE USO

Durante los siguientes puntos se detallará la aplicación en su capa de presentación. Esto va dirigido para los usuarios finales de la aplicación para que conozcan el funcionamiento y posibilidades que ofrece la aplicación. Cómo hemos comentado en el punto 6, se ha optado por una arquitectura de tres capas y dos niveles en la que, tanto la capa de presentación como la de lógica de negocio, se ejecutan desde el ordenador del usuario final.

Esto no es una buena aproximación y provoca que se ralentice el sistema considerablemente ya que hay una gran cantidad de información que se tiene que transmitir mediante la red local desde la capa de datos ubicada en un servidor hasta las capas de lógica de negocio y presentación ubicados de forma local en cada ordenador de cada usuario. Cada vez que el usuario hace uso de la aplicación, la información se tiene que transmitir, procesarla y presentarla de forma local en vez de hacer el proceso en el servidor y solo presentarla en el ordenador del usuario.

Aún sabiendo que no es una buena solución, es la solución que se ha tenido que utilizar debido a las requisitos que se establecieron para este proyecto.

7.1. Pantalla principal

La pantalla principal es la pantalla de presentación de la aplicación y esta está basada en páginas (como si estuviésemos dentro de un navegador con la diferencia que no tenemos los botones de anterior y siguiente escondidos a propósito) donde iremos pasando de página en página según la información que queramos consultar pero, salvo contadas excepciones,

siempre interactuaremos en el contexto de la misma ventana, la ventana principal de la aplicación.

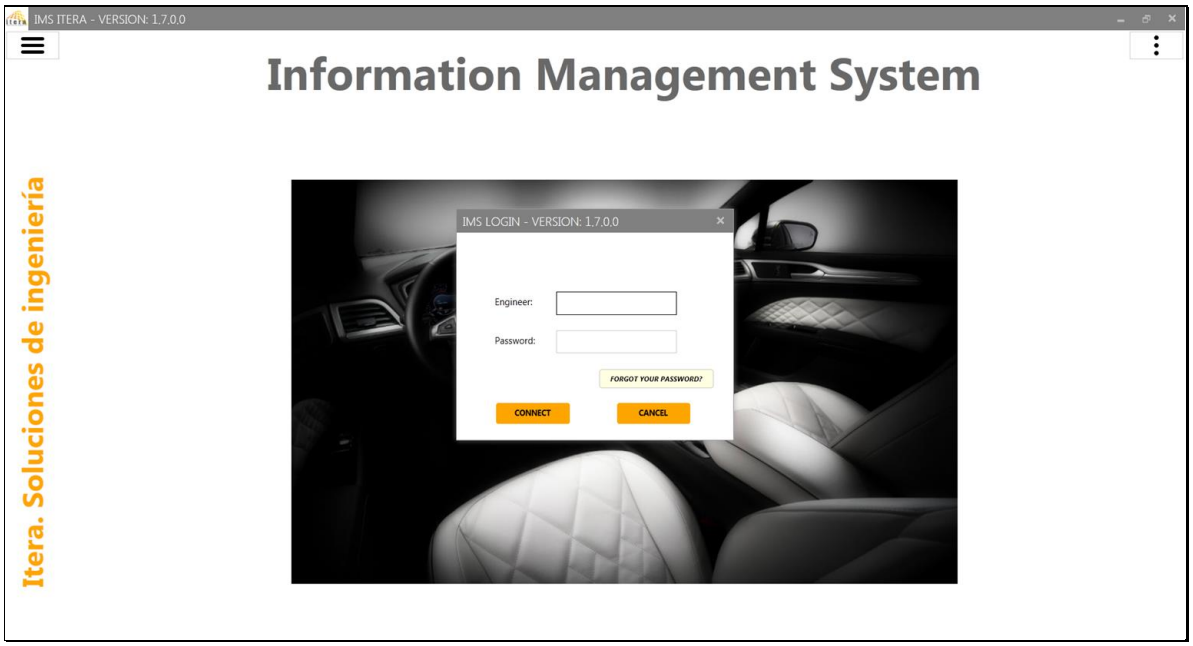


Ilustración 13. Ventana principal IMS

Al iniciar la aplicación, las primeras acciones que realizará serán intentar conectarse a la BBDD. En caso de no poder conectarse a la BBDD se mostrará una ventana con un mensaje de error.

Cabe destacar, que si bien hemos dicho que todas las acciones ocurrían dentro de la ventana principal, los mensajes de error se mostrarán en ventanas externas. Esto ocurre así ya que uno de los requisitos que tiene la aplicación es que funcione con la versión 4.5 de .Net Framework, en esta versión todavía no se han implementado las páginas async que serían los equivalentes a las ventanas de error que se muestran y que no bloquean el hilo de ejecución principal pero si crean una nueva capa para mostrar un mensaje. Esto último se implementa con la versión 4.5.1 de .Net.

7.2. Login

Una de las excepciones a navegar en la ventana principal es la ventana que se utiliza para autentificar al usuario. Esta ventana aparece al iniciar el programa o al llamar a la función “Log In” desde el menú de usuario de la aplicación (ver punto ¿?). Se ha establecido un formato de usuarios para que sea más intuitivo para el usuario final que consiste en utilizar el mismo usuario que utiliza el empleado en el cliente donde está trabajando (en este

caso el cliente es Ford y el código que utiliza para identificar a un usuario tiene el nombre de CDSID).

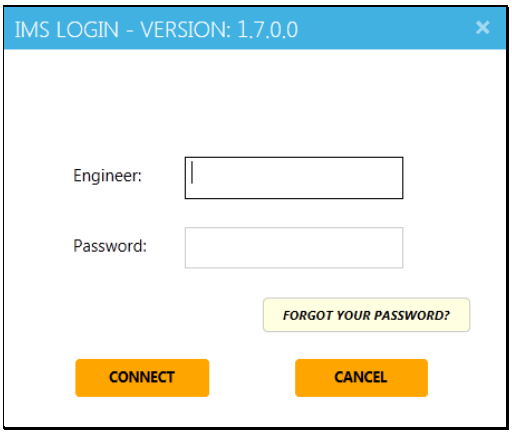


Ilustración 14. Ventana login IMS

En esta ventana externa a la ventana principal tenemos los siguientes campos:

- El usuario (será el CDSID de FORD)
- La contraseña (para el primer uso o al resetear la contraseña se utilizará el mismo CDSID del usuario o una contraseña en blanco)
    - Después del primer uso se pedirá la introducción de una contraseña propia
- En caso de olvidar la contraseña se puede pedir el reseteo mandando un correo mediante el botón “Forgot your password?”.
- Los botones “Connect” and “Cancel” sirven para verificar datos con la BBDD o para cerrar la ventana de login.

A continuación se puede ver un flujo de control de cómo se identifica un usuario en la aplicación:

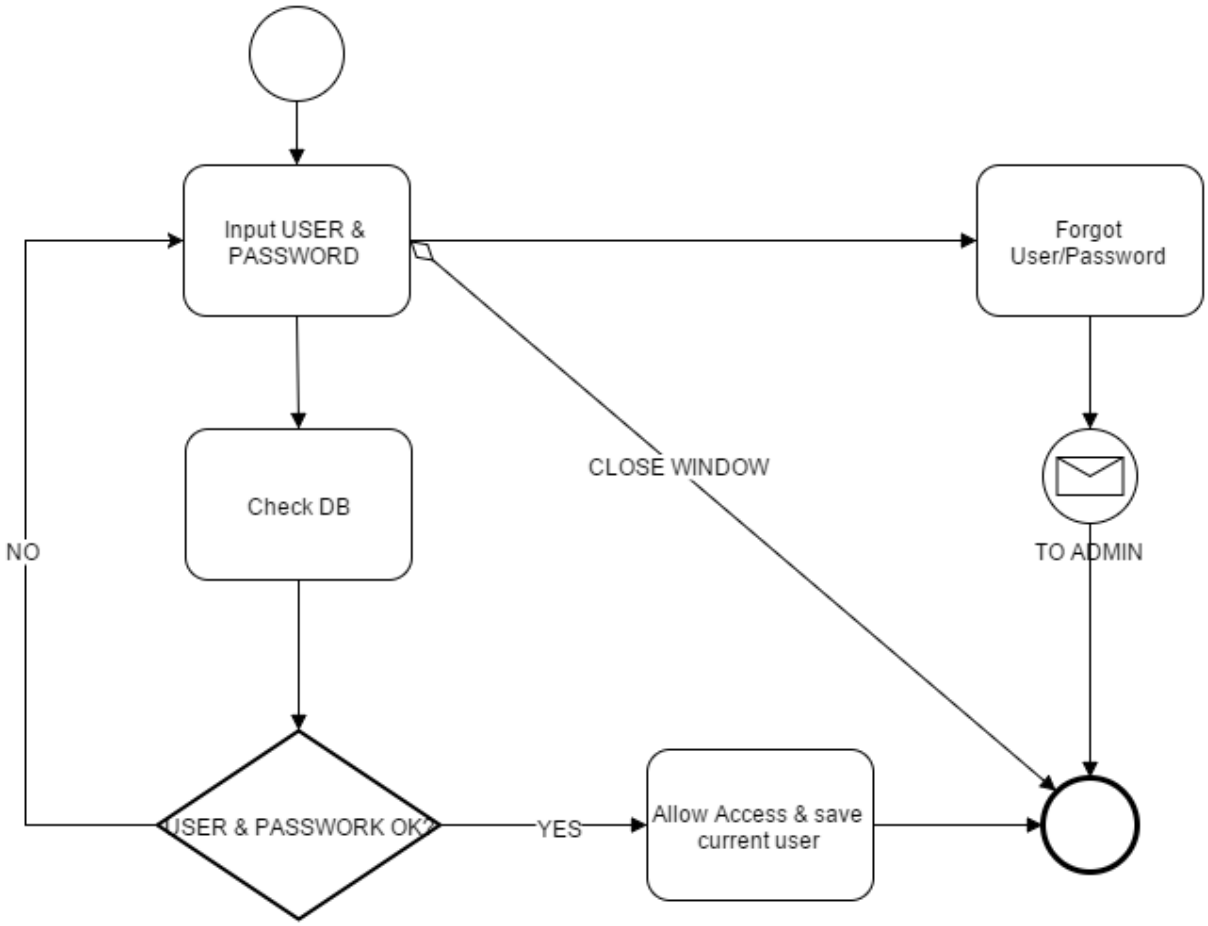


Ilustración 15. Diagrama login IMS

7.3. Menú usuario & menú de aplicación

Ilustración 16. Menu usuario IMS

Existen dos menús principales en la aplicación, el menú de usuario y el menú de la aplicación. Estos menús se encuentran en los laterales de la ventana principal, el menú de usuario en la parte izquierda y el menú de aplicación en la parte derecha. Para abrir estos menús basta con pasar el ratón por encima del botón correspondiente o hacer clic sobre él y para cerrarlos basta con situar el ratón fuera del menú o pulsar el botón de cierre que se encuentra siempre en el interior del marco del menú.

Los menús se han programado como como un marco tipo "flyout" el cual crea una capa que se sitúa por encima del contenido de la ventana que lo invoque. Esto permite utilizar toda el área disponible de la ventana principal para los datos.

Menú de usuario: se encuentran todas las opciones de gestión de información disponibles para el usuario.

- Concerns: Documento que permite mostrar/guardar información relacionada con problemas del vehículo.
- Alerts: Documento que permite mostrar/guardar información relacionada con problemas del vehículo.
- Event: Documento que permite mostrar/guardar información relacionada con problemas del vehículo.
- Issues: Submenú que contiene documentos similares en definición (AIMS, BSAQ) que se utilizan bien para el equipo de lanzamiento o para el equipo de PVT y que permite mostrar/guardar información relacionada con problemas del vehículo.
- Issues Itera: Documento que permite crear/mostrar/guardar información relacionada con cualquier tipo de problema aparecido en el trabajo del día a día.
- BOM: Documento que permite mostrar/guardar información relacionada con partes del vehículo.
- Graphs: Submenú que contiene informes disponibles para estudiar/gestionar la información guardada en base de datos como el informe Dashboard y Reports que permite la creación de informes personalizados.
- DB Tables: Submenú que contiene un listado de las principales tablas de la BBDD que sean creado para realizar un mantenimiento de la aplicación. La aplicación permite para todas estas tablas crear/modificar/borrar los datos.
    - Engineers: Listado de usuarios dados de alta en la aplicación junto a toda su información
    - PMTs/VFGs/Other: Listado de las distintas áreas del coche según la división establecida por Ford.

    - MY/CODE: Listado de los años y códigos de modelos de vehículos según el formato establecido por Ford.
    - Activity: Listado de las diferentes actividades que actúan sobre el vehículo según el formato establecido por Ford.
    - Issue Type Itera: Listado de las diferentes Issues que previamente se han establecido por los responsables de la empresa Itera que pueden aparecer durante el trabajo de cada día.
    - Auto Assign: Documento que permite establecer las reglas que se deben cumplir para que un ítem se asigne a un usuario.
- DB Other: Submenú que contiene un listado de tablas secundarias de la BBDD que sean creado para realizar un mantenimiento de la aplicación. La aplicación permite para todas estas tablas crear/modificar/borrar los datos.
    - Historial Acceso: Muestra la información de todos los logins (correctos o incorrectos) que realizan los usuarios al intentar entrar a la aplicación. Esta tabla solo se puede visualizar pero no se puede ni añadir, ni modificar información.
    - Vehicle Line: Listado de los nombre que se proporcionan a los modelos independientemente de los submodelos que se fabriquen con sus diferentes opciones y de que fábrica según el formato establecido por Ford.
    - CPSC: Listado de las diferentes áreas del coche según el formato establecido por Ford.
    - Vehicle Program: Listado de los nombres que se proporcionan a los submodelos según el formato establecido por Ford.
    - Plants: Listados de las plantas de fabricación utilizadas en la aplicación según el formato establecido por Ford.
- Get Data: Documento que permite a administradores de la aplicación poder lanzar las funciones que actualizan la información de la BBDD. Estas funciones permiten actualizar las información de los documentos: Conerns, Alerts, Event, BSAQ, AIMS, BOM y además permite lanzar la función que auto asigna los ítems a usuarios según las reglas creadas previamente.
- Settings: Documento que permite modificar diferentes opciones de la aplicación, como los datos de conexión a la BBDD, los datos de usuario para acceder a datos en terminales 3270 y otros datos de la aplicación.

Menú de usuario: se encuentran todas las opciones de gestión de aplicación disponibles para el usuario.

- IMS: Permite cerrar la aplicación con la opción “Exit” del submenú.
- DB: Permite la conexión, desconexión de la BBDD y mostrar la ventana de login.
- MEMO: Permite mostrar la ventana principal y mostrar u ocultar el memo de la aplicación donde se van registrando usos y problemas o errores.
- USER: Permite volver a poner las opciones por defecto de la aplicación para el usuario activo, también permite cambiar la contraseña del usuario activo y mandar un informe de mejoras, errores al administrador de la aplicación.
- HELP: Muestra la ventana de “About” de la aplicación.

7.3.1. Documentos

Llamaremos documento a la plantilla creada en fase de diseño para mostrar la información sin tener en cuenta la fuente ni los formatos originales. Esta solución sirve para que la curva de aprendizaje para la nueva información que se introduzca en la BBDD sea mínima.

Todos los campos que estén marcados con un asterisco y de color naranja significan que son campos obligatorios a la hora de rellenar información. Por otro lado los campos con un fondo gris significan que son de solo lectura y aquellos campos que tengan un fondo blanco significan que pueden ser de lectura y escritura.

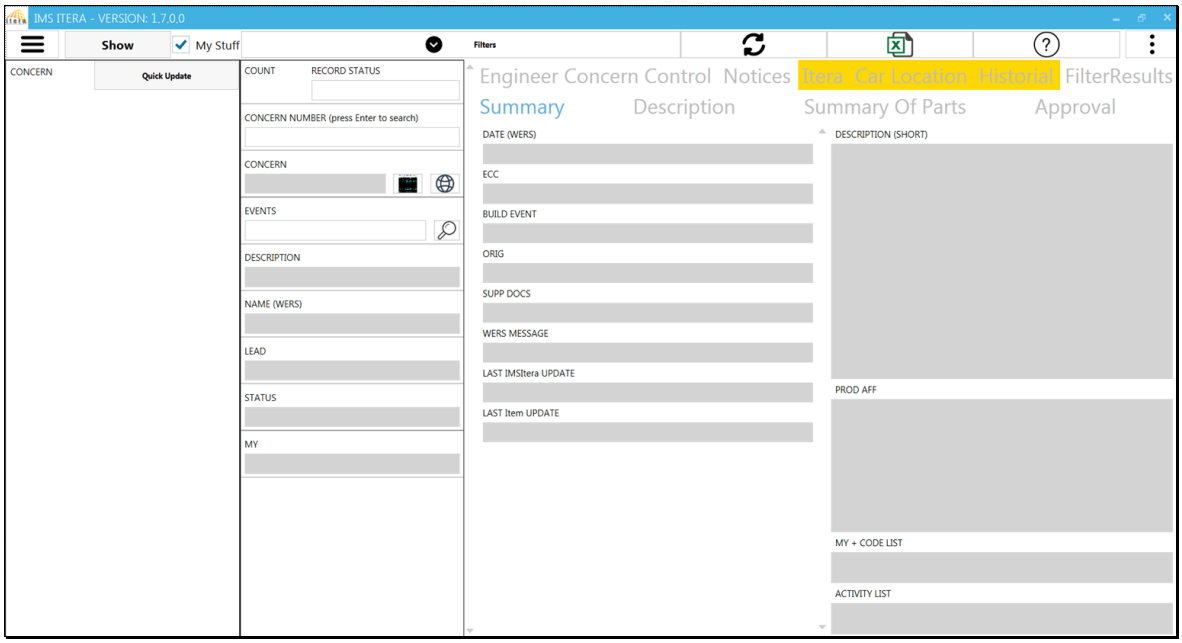


Ilustración 17. Ventana documento IMS

La plantilla documento consta de 6 partes principales:

- Botones de acción.
- Listado de ítems.
- Datos Fijos.
- Botones de información local.
- Pestañas de información.
- Datos de pestañas.

7.3.1.1. Botones acción

La barra de botones de acción permite al usuario solicitar acciones a la aplicación.

- Botón Show: Recupera de BBDD los datos de acuerdo a los filtros seleccionados.
- My Stuff: Al seleccionarlo solo tiene en cuenta los datos del ingeniero logueado.
- Menú Filtros: Expande el menú para seleccionar los filtros a aplicar a los datos (Ver punto 7.3.1.2).
- Botón Refresh: Resetea la ventana de documentos.
- Botón Tareas (*si disponible*): Permite a un usuario crear una tarea asignándola a uno o varios usuarios al mismo tiempo (Ver punto 7.3.1.3).
- Botón Excel: Genera un documento Excel con toda la información correspondiente al listado actual.
- Botón Ayuda: Muestra ventanas de ayuda para utilizar la app.
- Botón Recargar BBDD (*si disponible*): Permite realizar una desconexión y conexión rápida a la BBDD en los casos en los que puedan existir problemas de conexión durante el uso de la aplicación.

7.3.1.2. Filtros

Los filtros son el corazón de la gestión de la información dentro de la aplicación. Los filtros se muestran gracias a un menú tipo “flyout” al igual que los menús de usuario y de aplicación. Todos los filtros que se han añadido en cada documento son debidos a los requisitos que se plantearon en la fase de toma de requisitos en las reuniones con los responsables de cada área.

Dentro del menú de filtros tenemos cuatro divisiones principales, estas divisiones están implementadas mediante “expanders” y su función es la de cargar los filtros con respecto a los datos de BBDD solo la primera vez que se abra el expander, así se consigue mejorar la experiencia de usuario al gastar solo el tiempo de consulta a BBDD para rellenar solo los filtros que el propio usuario necesite.

Todos los filtros que estén en blanco no se tendrán en cuenta a la hora de consultar la BBDD por lo que es como decir que se tienen que seleccionar todas las instancias de ese filtro. Además para que los filtros relacionados con campos que son modificables por el usuario puedan cumplir su función se necesita que el usuario proporcione a la aplicación toda la información que pueda.

Para abrir este menú basta con pasar el ratón por encima del botón correspondiente o hacer clic sobre él y para cerrarlo basta con situar el ratón fuera del menú o pulsar el botón de cierre que se encuentra siempre en el interior del marco del menú. Las cuatro divisiones principales son:

- ITERA: Los filtros principales que pertenecen a los datos locales de la aplicación, es decir, los datos que no son del cliente.

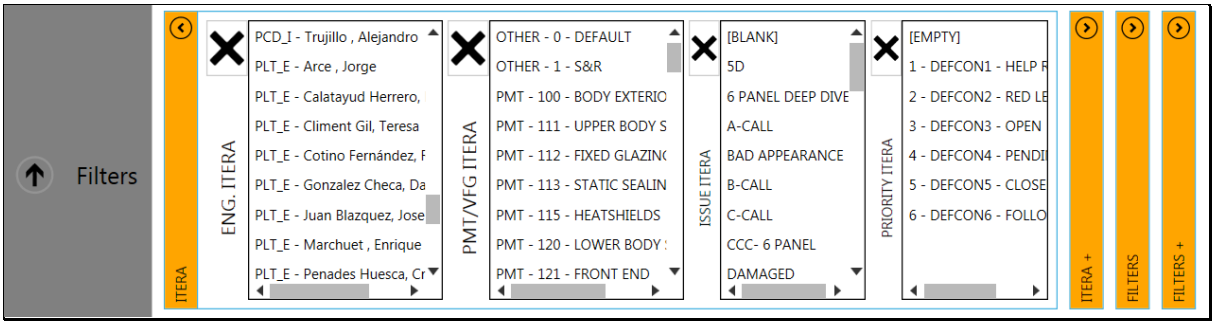


Ilustración 18. Filtros en IMS 1

- Eng. Itera es un filtro tipo listado en el que se pueden seleccionar uno o varios elementos y son los usuarios dados de alta en la aplicación y están ordenados en grupos según los equipos/proyectos a los que están asignados para facilitar la selección de todos los grupos de usuarios.
- PMT/VFG ITERA es un filtro tipo listado en el que se pueden seleccionar uno o varios elementos y son todas las áreas de un vehículo utilizando el formato de Ford y están ordenados mediante grupos según cual sea el origen del formato, además se han añadido áreas genéricas bajo el grupo “OTHER”.
- ISSUE ITERA es un filtro tipo listado en el que se pueden seleccionar uno o varios elementos y son los distintos Issues que se han identificado junto a los responsable que pueden aparecer a la hora de gestionar la información de problemas dentro de la aplicación.

- PRIORITY ITERA es un filtro tipo listado en el que se pueden seleccionar uno o varios elementos y especifica la prioridad de un elemento para su resolución.

- ITERA+: Son filtros secundarios que pertenecen a los datos locales de la aplicación, es decir, los datos que no son del cliente (Ford).

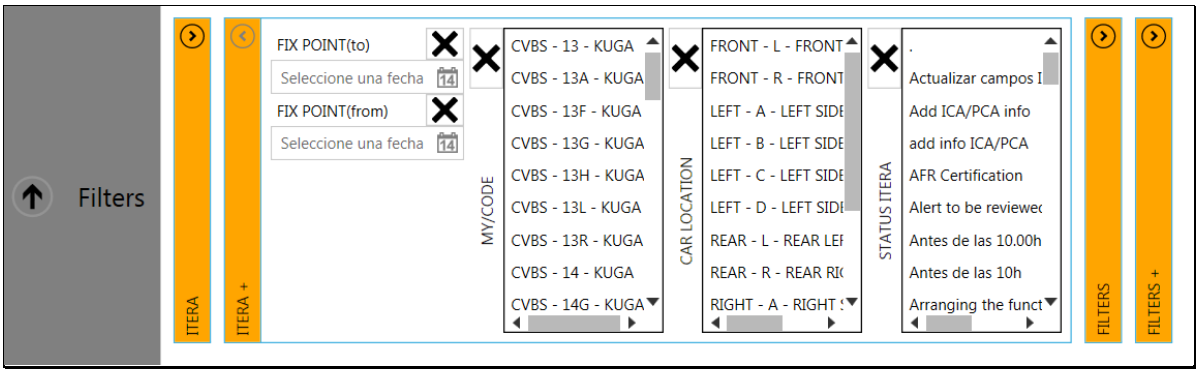


Ilustración 19. Filtros en IMS 2

- FIX POINT (to) / FIX POINT (from) son filtros tipo fecha que permiten introducir una fecha, bien directamente o bien mediante un calendario que se muestra en un pop-up y permite mostrar elementos los cuales se ha marcado dentro del intervalo de fechas seleccionada.
- MY/CODE es un filtro tipo listado en el que se pueden seleccionar uno o varios elementos y especifica que modelos y que años se quiere recuperar.
- STATUS ITERA es un filtro tipo listado en el que se pueden seleccionar uno o varios elementos y viene de un campo abierto que introduce el usuario. Las distintas opciones que se ven en este listado son todos los textos diferentes con los usuarios han introducido como información en sus ítem. Para este campo es importante recordar como se ve en el punto de datos en pestañas de Itera que es conveniente que exista un formato previo para indicar que información se está añadiendo.

- FILTERS: Los filtros principales que pertenecen a los datos de cliente (Ford). Cada uno de estos filtros depende del documento al que se refieran (Concerns, Alerts,...). Para entrar en detalle en estos filtros se debe preguntar al responsable directo o al administrador de la aplicación.
- FILTERS+: Son filtros secundarios que pertenecen a los datos de cliente (Ford). Cada uno de estos filtros depende del documento al que se refieran (Concerns, Alerts,...). Para entrar en detalle en estos filtros se debe preguntar al responsable directo o al administrador de la aplicación.

7.3.1.3. Tareas

La función Tareas permite a un usuario asignar una tarea a uno o varios usuarios a la vez. Esta función se ejecuta desde una ventana independiente a la ventana principal y se puede utilizar haciendo clic sobre el botón "Tareas" de la barra de botones de acción.

Este función solo podrá utilizarse por parte de usuarios con permisos habilitados, en este caso serán aquellos que tengan como mínimo un rol de responsable de equipo.

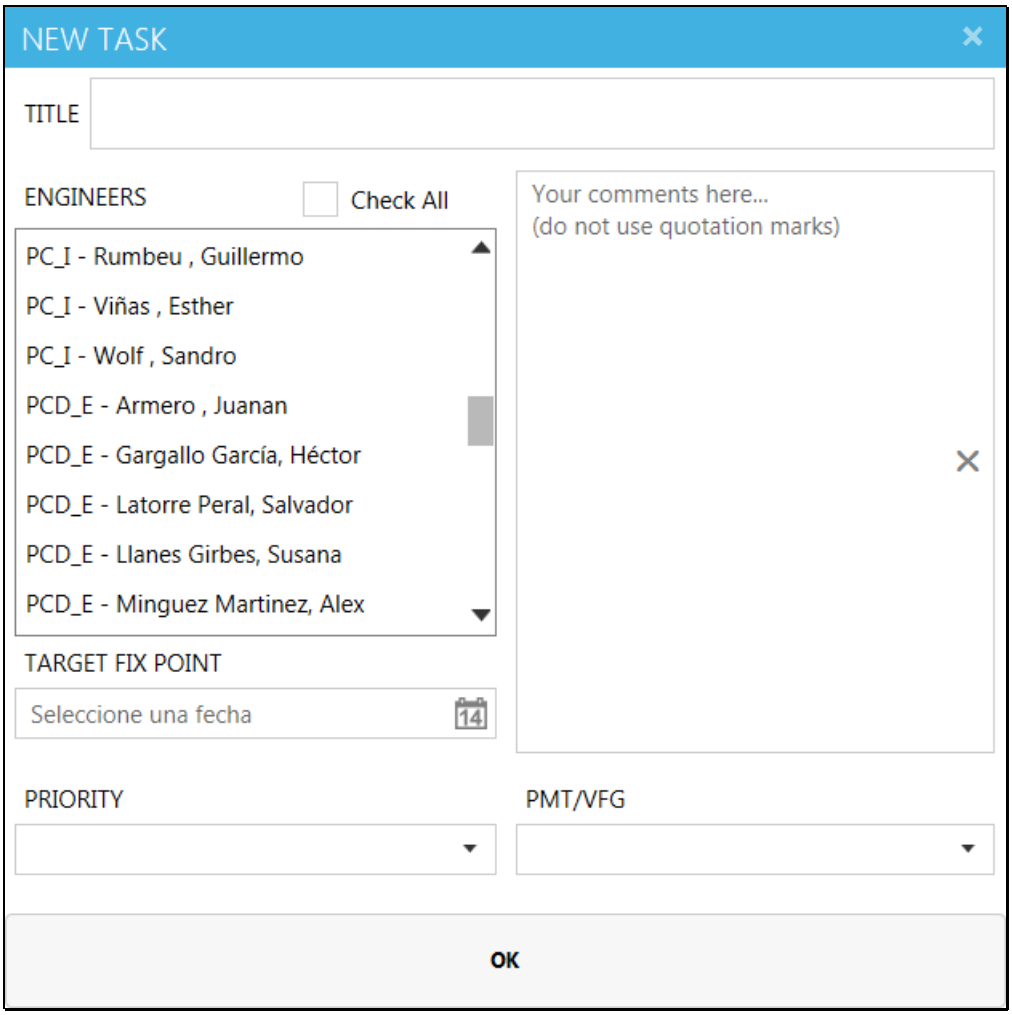


Ilustración 20. Ventana de tareas en IMS

Rellenar todos los campos de esta ventana es obligatorio para poder crear las tareas. Los campos son los siguientes:

- Title: título de la tarea (con un máximo de 140 carácteres).
- Engineers: listado de usuarios a los que se les asignará la tarea en el documento desde donde se ejecute el botón "Tareas". Este campo tiene la opción de seleccionar todo el listado de usuarios seleccionando la opción "Check All".

- Target Fix Point: es un campo fecha que indica la fecha límite para tener esta tarea solucionada.
- Priority: es el nivel de prioridad que el creador dela tarea considera correcto para el inicio de esta.
- Comments: Donde el iniciador de la tarea especificará para que se ha creado la tarea y cuál es su finalidad.
- PMT/VFG: campo donde se especifica cual es la relación de la tarea con el vehículo, si la tiene.

Los pasos para crear una tarea serían los siguientes:

1. Comprobar que la opción crear "Tarea" está disponible en la barra de botones de acción de acción del documento donde queramos crear la tarea.
2. Pulsar el botón "Tarea" y rellenar todos los campos de la ventana emergente.
3. Al pulsar el botón "Ok" se creará la tarea para el documento activo en cada usuario seleccionado en la lista de usuarios.

En caso de querer cancelar la tarea, basta con cerrar la ventana emergente en cualquier punto del proceso de creación de una tarea.

7.3.1.4. Listado

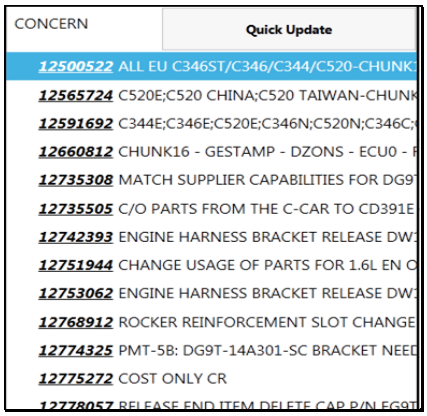


Ilustración 21. Ejemplo listado en un documento en IMS

Esta parte de un Documento es simplemente el listado con la información principal (normalmente identificador + una breve descripción) de los datos obtenidos en BBDD de acuerdo a los filtros seleccionados. Este listado sirve para obtener un vistazo rápido al ítem mediante la pequeña descripción que acompaña al identificador y para seleccionar

cualquiera de los ítems y que nos muestre toda su información en las partes de Datos Fijos y Datos de Pestañas que veremos más adelante.

En algunos documentos está disponible la opción “Quick Update” mediante un botón en la parte superior del listado y permite la actualización de los elementos que se encuentran en el listado hasta un máximo de 30.

Para utilizarlo con la fuente de datos WERS se debe tener instalado el programa IBM PERSONAL COMMUNICATIONS.

7.3.1.5. Datos fijos

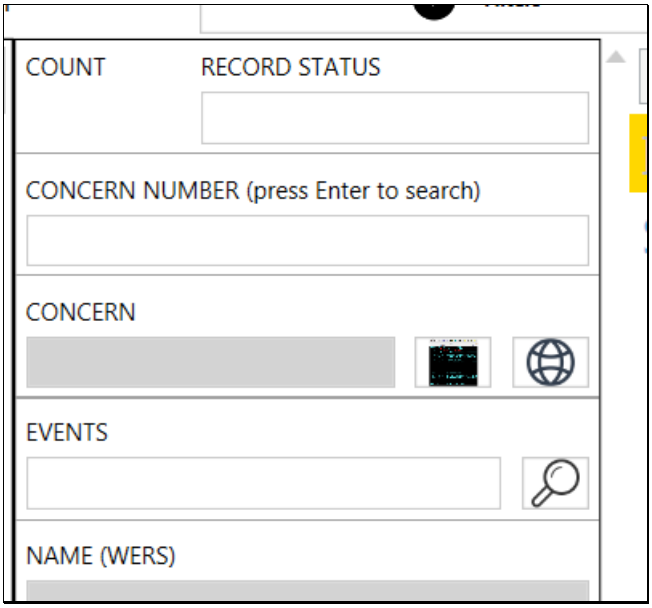


Ilustración 22. Ejemplo de datos fijos en ventana Documento en IMS

La sección de datos fijos de un documento es la parte donde se podrán consultar la información más importante de un ítem previamente seleccionado en el listado y siempre estarán visibles.

Estos datos dependerán de la información que contenga el ítem, pero independientemente de eso habrá una serie de características que se compartirán en los datos fijos:

- Campo “Count” muestra el número de resultados obtenidos en el listado
- Record Status muestra el estado del ítem en BBDD y tiene 3 estados:
    - ACTIVE: El ítem ha aparecido en la última actualización.

    - ON HOLD: El ítem lleva entre 1 y 10 días sin aparecer en una actualización.
    - DELETED: El ítem lleva más de 10 días sin aparecer en una actualización.
- Botón Mundo (*si disponible*): Muestra el ítem seleccionado en la fuente de datos WEB original
- Botón Terminal (*si disponible*): Muestra el ítem seleccionado en la fuente de datos de Terminal Mainframe 3270 original.
- Botón Lupa (*si disponible*): Muestra el ítem seleccionado en otro documento de la aplicación.

7.3.1.6. Datos variables

La sección de Datos Variables del documento contiene información secundaria del ítem previamente seleccionado en el listado y que se muestra según que pestaña esté activa.

Dependiendo del color de fondo de la pestaña, tenemos dos tipos de información que se pueden visualizar:

- (Sin color de fondo): Datos recogidos de la fuente, son datos de solo lectura.
- (Con color de fondo): Datos de la aplicación, son datos de lectura y escritura para añadir información personal a un ítem seleccionado (Ingeniero asignado, prioridad, etc.)

7.3.1.7. Pestaña Itera

Dentro de la sección de Datos Variables tenemos la pestaña Itera (con color de fondo), esta pestaña contiene los datos principales que se pueden añadir a un ítem sea cual sea la fuente de información de donde provengan.

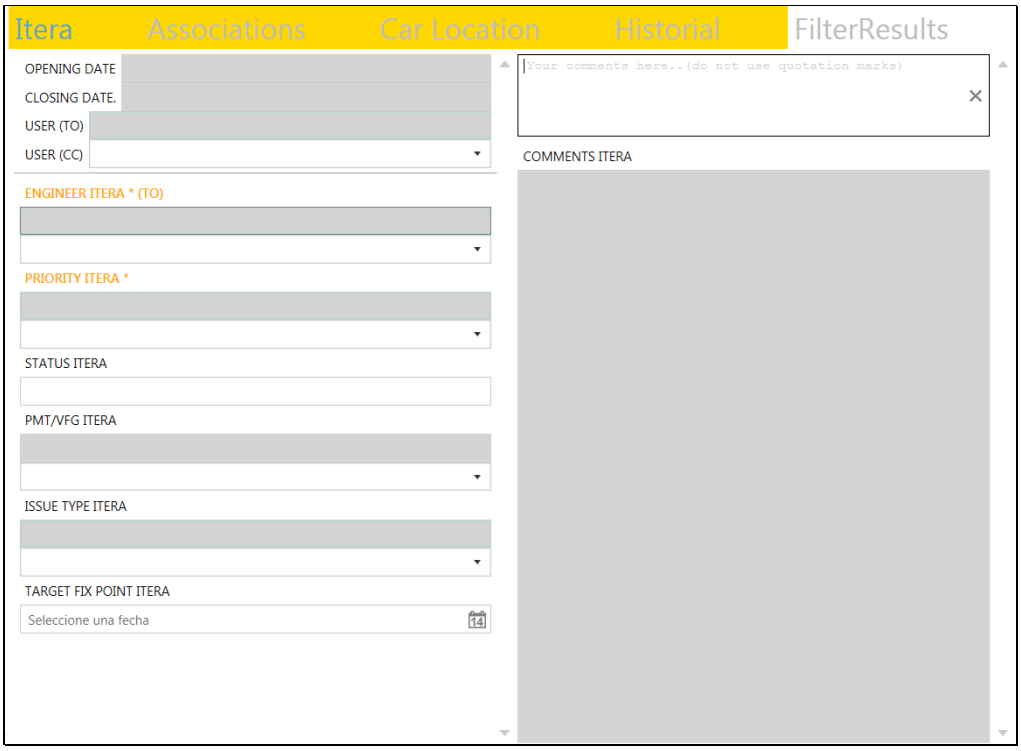


Ilustración 23. Datos adicionales en la pestaña Itera.

Estos datos que se pueden añadir como información adicional se detallan a continuación:

- Opening Date es la fecha en la de la última asignación de el ítem actual a un usuario (no se puede modificar).
- Closing Date es la última fecha en la que el ítem paso a “DEFCON 5 – CLOSED” en el campo “Priority Itera” (no se puede modificar).
- User (to): Último ingeniero en modificar algún dato (no se puede modificar).
- User (cc): Usuario que va a recibir una copia del correo de notificación al guardar la información. Es un desplegable donde se puede seleccionar solo un usuario.
- Engineer Itera (to): Ingeniero asignado al ítem y es un campo obligatorio.
- Priority Itera: Campo obligatorio que especifica el estado de un ítem. Este campo es obligatorio.
- Status Itera: Campo libre que amplía la información de priority
- PMT/VFG Itera: Campo con el listado de PMTs, VFGs y Otros de las distintas áreas del coche según la notación Ford.
- IssueType Itera: Problema genérico para clasificar el ítem
- Target Fix Point Itera: Fecha prevista de resolución de la incidencia
- Comments Itera: Campo libre para incluir comentarios sobre el trabajo realizado en el ítem. Tenemos que tener en cuenta que los comentarios no se podrán

eliminar una vez guardados y que todos los comentarios incluirán la fecha y el usuario que los realizó.

7.3.1.8. Botones de información local

Los botones de información local solo estarán disponibles cuando sea posible utilizarlos y solo gestionan información local que se pueda crear/modificar/eliminar (pestañas con color de fondo).

Ilustración 24. Botones de información local en IMS

Existen cuatro botones para gestionar la información local:

- Botón NEW (*si activo*): Crea una nueva instancia en el documento.
- Botón UPDATE (*si activo*): Guarda la información en BBDD (mandará un correo de aviso siempre que se cambie el ingeniero asignado, en otro caso no)
- Botón UPDATE And SEND (*si activo*): Guarda la información en BBDD y siempre envía un correo de aviso
- Botón DELETE (*si activo*): Marca como RECORD STATUS "Deleted" el ítem seleccionado en el listado, pero no borra la información de BBDD.

7.3.1.9. Correo automático de nueva información

Una parte importante de un sistema de gestión es poder informar a los usuarios de información relevante que tenga relación con ellos, en este aspecto la aplicación dispone de una función de notificación que envía un correo electrónico automático utilizando la cuenta predeterminada configurada en el programa Microsoft Outlook.

Esta notificación se enviará siempre que, al guardar, el campo de Engineer Itera cambie en la pestaña Itera en los Datos Variables o siempre que un usuario guarde utilizando el botón "Update and Send". Esto se hace así porque la asignación de un ítem a un usuario es una información prioritaria y según requisitos siempre debe ser notificada.

El correo es autogenerado e informará al usuario añadiendo unos campos en concreto para poder identificar cual es el ítem que ha cambiado y en la mayoría de casos que

información ha cambiado. Los destinatarios del correo serán por defecto el usuario que figure en el campo "Engineer Itera" y el último usuario que haya hecho cambios sobre ese ítem, adicionalmente se podrá enviar también a un

- Título donde se indica que tipo de documento ha generado el aviso y es el campo asunto del correo
- Ítem: Número identificativo del ítem que ha generado la nueva información y se muestra en el cuerpo del correo.
- Description: Campo descripción del ítem y se muestra en el cuerpo del correo.
- Priority: Última prioridad asignada al ítem (en caso de que se haya modificado sino se quedará en blanco) y se muestra en el cuerpo del correo.
- Status: Campo libre Status asignada al ítem y se muestra en el cuerpo del correo.
- Comments: Solo el último comentario añadido al ítem y se muestra en el cuerpo del correo.

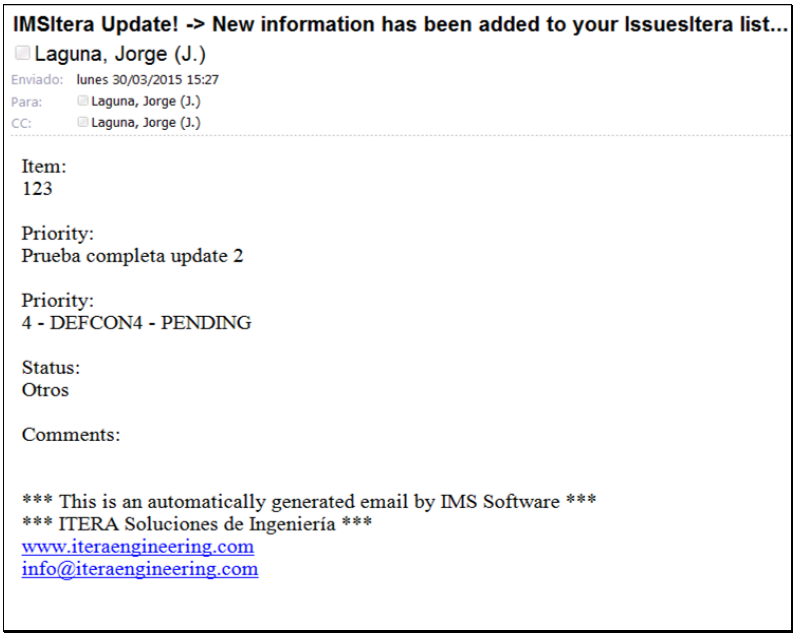


Ilustración 25. Ejemplo correo automático para notificar nuevo información con respecto a un ítem de la BBDD

7.3.2. Graphs

Esta parte de la aplicación corresponde al estudio de los datos dentro del Sistema de Información que se está creando al usar la aplicación. Utilizando los datos almacenados en la capa de información se buscará dar sentido a la información de una forma que sirva para evaluar las acciones que se están realizando sobre todos esos problemas.

Esta al ser la última parte en el proceso en la que se ha trabajado en este proceso de desarrollo, está limitada y solo se ofrece un resumen de los estados de los ítems por documento y usuario.

En futuras mejoras de la aplicación se añadirán generadores de informes para extraer información concreta que ayude a la toma de decisiones.

7.3.2.1. Dashboard

El Dashboard es el documento que muestra el resultado global de un usuario con respecto a los ítems que tiene asignados en cada documento. La información que presenta está dividida en varias pestañas siendo la principal la pestaña "Summary" donde el resumen se hace en base a las diferentes posibilidades que existe a la hora de seleccionar el campo "Priority Itera".

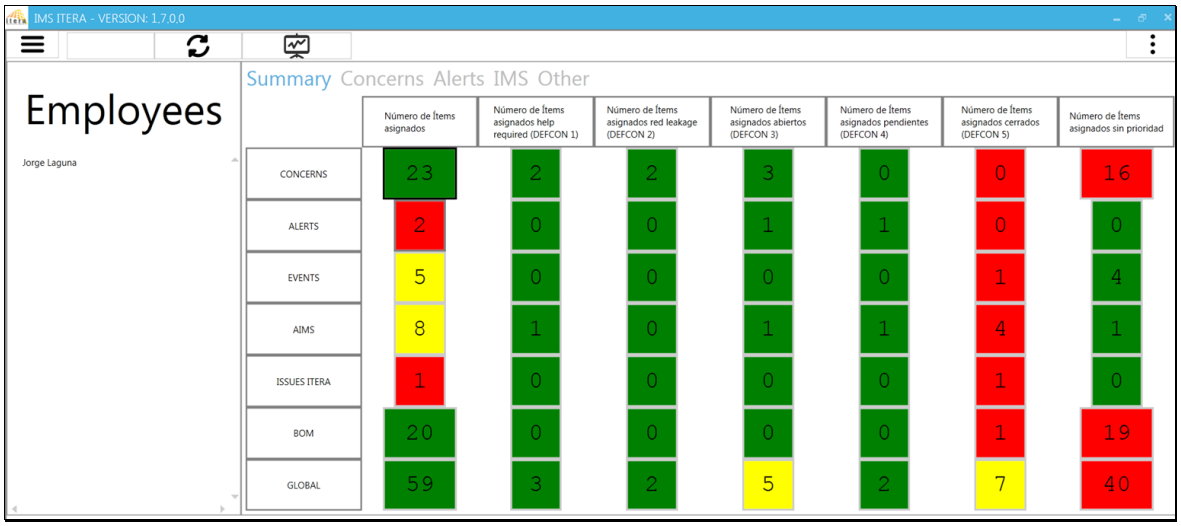


Ilustración 26. Ejemplo de la pantalla Dashboard

- Summary: Resumen de las prioridades del ingeniero logueado que se dividen en los ítems de mayor prioridad hasta los ítems de menos prioridad (Help Required, Red Leakage, Open, Pending, Closed, Follow Up) añadiendo como primera columna todos los ítems asignados y como última los ítems que no tienen una prioridad asignada.
- Concerns
    - Status A (número de ítems + listado), muestra el número total y el listado de todos los ítems que el valor de "Status" en los datos del cliente es "A".
    - Status C (número de ítems + listado), muestra el número total y el listado de todos los ítems que el valor de "Status" en los datos del cliente es "C".

    - Status W (número de ítems + listado), muestra el número total y el listado de todos los ítems que el valor de "Status" en los datos del cliente es "W".

- Alerts
    - Sin autorizar (número de ítems + listado), muestra el número total y el listado de todos los ítems que el valor de "Authorized" en los datos del cliente es "N" o "_".
    - Status A (número de ítems + listado), muestra el número total y el listado de todos los ítems que el valor de "Status" en los datos del cliente es "A".
    - Status C (número de ítems + listado), muestra el número total y el listado de todos los ítems que el valor de "Status" en los datos del cliente es "C".
    - Status W (número de ítems + listado), muestra el número total y el listado de todos los ítems que el valor de "Status" en los datos del cliente es "W".
    -
- IMS
    - Número de horas totales utilizando la aplicación.
    - Número de acceso a la aplicación en el último mes.
- Other
    - Otros indicadores de prueba que desaparecerán en futuras versiones o pasarán a la pestaña correspondiente. No es relevante indicarlos en el manual

Para cambiar los datos en el Dashboard se debe trabajar sobre los ítems de cada documento ya que la información mostrada en este informe es solo de salida. Es obligatorio como uso del programa asignar una prioridad a cada ítem asignado a un usuario para que la información de los informes sea lo más coherente posible y además estos informes permitirán ver a un usuario todos sus datos y todos los datos de personas que estén bajo su responsabilidad.

Sabiendo que una de las facetas más importantes de un sistema de información es la presentación de datos para la toma de decisiones se añadirán en un futuro muchos más informes basándose en los requisitos establecidos según las reuniones realizadas con los responsables de equipo y empresa para mejorar la visión global.

Todos estos informes que se recogerán de la etapa de recogida de requisitos son indicadores que podremos dividir en dos grandes áreas:

- Actuales (muestran el estado actual de la información)
- Periódicos (muestran el resultado durante un periodo de tiempo)

7.3.3. Auto asignación

Existe un proceso de actualización que asigna ítems automáticamente a los ingenieros. Esto se basa en crear reglas en las que los datos de varios campos identifican a que usuario tiene que asignarse a un ítem.

Para crear las reglas se ha creado un asistente que muestra los pasos y solicita la información para crear la regla.

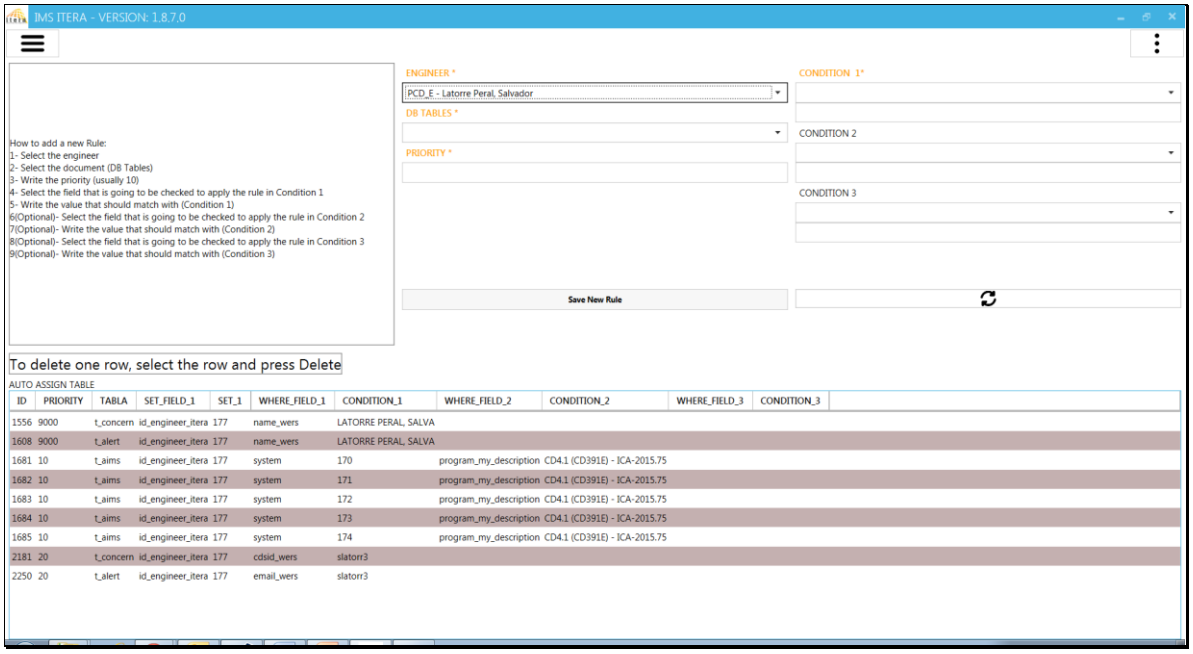


Ilustración 27. Asistente para reglas de auto asignación.

En este asistente podemos seleccionar un usuario en el campo “Engineer” y ver el listado de reglas que contiene. Estas reglas se pueden borrar de manera individual seleccionándolas y pulsando la tecla suprimir.

La parte izquierda del asistente muestra los pasos para añadir una regla:

1. Seleccionar un usuario en el campo “Engineer”. Es un campo obligatorio.
2. Seleccionar un documento en el campo “DB Tables”. Es un campo obligatorio.
3. Escribir una prioridad para esa regla, las prioridades por debajo de 1000 se ejecutarán con la opción de prioridad rápida. Es un campo obligatorio.
4. Seleccionar el campo que se va a consultar y escribir el valor que debe tener para crear la regla en el apartado “Condition 1”. Es un campo obligatorio.
5. Lo mismo que el punto 4 para “Condition 2”.Este valor es opcional.
6. Lo mismo que el punto 4 para “Condition 3”. Este valor es opcional.

Al realizar los pasos anteriormente expuestos podemos borrar la regla utilizando el botón “Save New Rule”.

Muchas veces ocurrirá que como los datos de la fuente no dan la suficiente información como para discernir a qué usuario en concreto se tiene que asignar un ítem, siendo las posibilidades normalmente dos o tres usuarios. Para resolver este problema se ha desarrollado un procedimiento que debe conocer todo usuario que utilice la aplicación y que consiste en los siguientes puntos:

- Asignar prioridad y otros datos si el ítem le corresponde.
- Reasignar el ítem a otro ingeniero si no le pertenece (siempre con el conocimiento de vuestro responsable).
- Reasignar el ítem al usuario “PODGE, HODGE” si el ítem no pertenece a nadie del equipo (siempre con el conocimiento del responsable correspondiente).

7.3.4. Gestión

En este punto trataremos sobre uno de los puntos que debe existir en toda aplicación, que es la gestión y mantenimiento para poder utilizarla a la hora de que aparezcan cambios en el uso habitual sin tener que realizar cambios en el código.

El mantenimiento en esta aplicación se basa en la información que debe conocer previamente para poder hacer las gestiones de gestionar la información y que no se puede sacar de forma directa de ninguna fuente de información. En resumen son todos aquellos listados que diferencian la información que es útil para los usuarios del resto de información que dispone el cliente para otros departamentos, fábricas, modelos, etc.

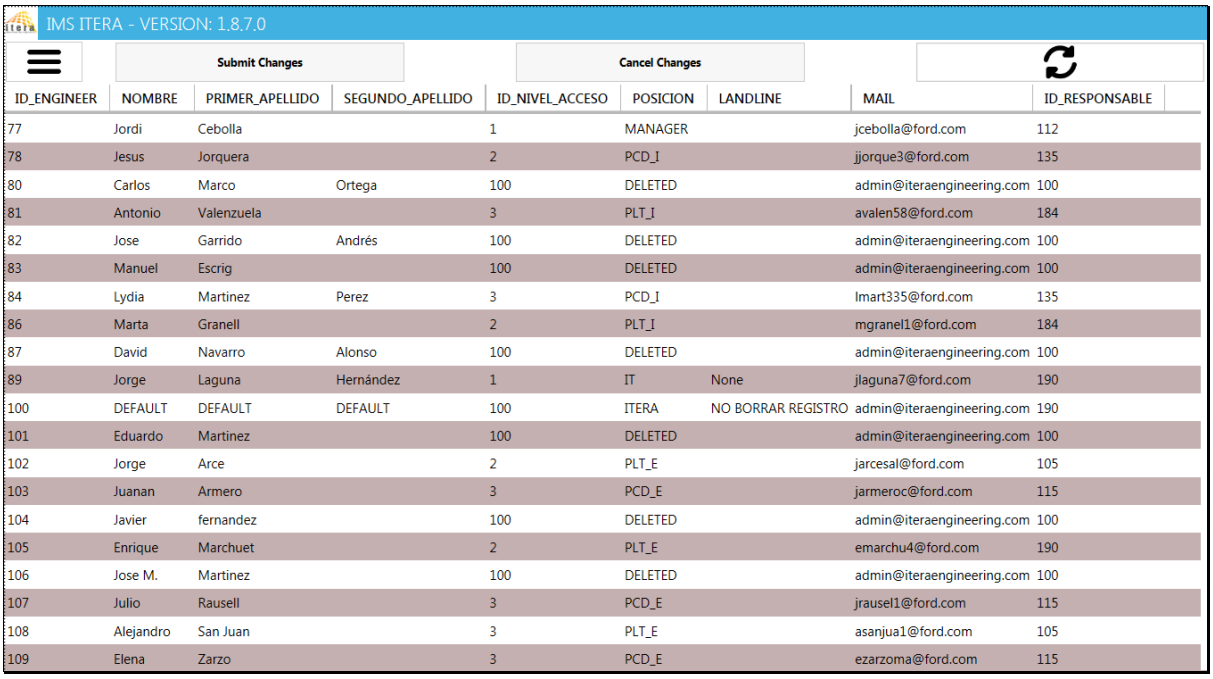
Estos listados se guardan como tablas en la base de datos y la aplicación permite interactuar sobre estos datos, añadiendo, modificando o eliminando la información.

7.3.4.1. Tablas DB principales

DB Tables son las tablas principales de información que pueden ser manipuladas para añadir o eliminar información por parte de usuarios con permisos.

ENGINEERS, listado con todos los ingenieros donde encontramos la siguiente información:

- Nombre, primer apellido, segundo apellido más un número de identificación creado al introducir al usuario por primera vez como datos de identificación.
- Id nivel acceso que indica cuál es su nivel de permisos y por lo tanto que secciones de la aplicación puede utilizar y cuáles no.
- Posición, landline y mail como datos de contacto, donde la posición nos permitirá agrupar a los diferentes empleados por grupos en los listados.
- Id responsable que permitirá crear el árbol de jerarquías que por ejemplo se usa en el Dashboard cuando alguien tiene otros usuarios bajo su responsabilidad. Este árbol se crea de forma recursiva y puede tener tantos niveles como haya en la propia empresa. El 1 se establece como el permiso más alto y aumentando los números los permisos irán descendiendo. Tal cual está ahora los permisos están divididos de la siguiente forma:
    - 1: Superadministrador
    - 2: Administrador
    - 3: Usuario estándar de la compañía
    - 4: Usuario estándar externo



ID_ENGINEER	NOMBRE	PRIMER_APELLIDO	SEGUNDO_APELLIDO	ID_NIVEL_ACCESO	POSICION	LANDLINE	MAIL	ID_RESPONSABLE
77	Jordi	Cebolla		1	MANAGER		jcebolla@ford.com	112
78	Jesus	Jorquera		2	PCD_I		jjorque3@ford.com	135
80	Carlos	Marco	Ortega	100	DELETED		admin@iteraengineering.com	100
81	Antonio	Valenzuela		3	PLT_I		avalen58@ford.com	184
82	Jose	Garrido	Andrés	100	DELETED		admin@iteraengineering.com	100
83	Manuel	Escrig		100	DELETED		admin@iteraengineering.com	100
84	Lydia	Martinez	Perez	3	PCD_I		lmart335@ford.com	135
86	Marta	Granell		2	PLT_I		mgranel1@ford.com	184
87	David	Navarro	Alonso	100	DELETED		admin@iteraengineering.com	100
89	Jorge	Laguna	Hernández	1	IT	None	jlaguna7@ford.com	190
100	DEFAULT	DEFAULT	DEFAULT	100	ITERA	NO BORRAR REGISTRO	admin@iteraengineering.com	190
101	Eduardo	Martinez		100	DELETED		admin@iteraengineering.com	100
102	Jorge	Arce		2	PLT_E		jarcesal@ford.com	105
103	Juanan	Armero		3	PCD_E		jarmeroc@ford.com	115
104	Javier	fernandez		100	DELETED		admin@iteraengineering.com	100
105	Enrique	Marchuet		2	PLT_E		emarchu4@ford.com	190
106	Jose M.	Martinez		100	DELETED		admin@iteraengineering.com	100
107	Julio	Rausell		3	PCD_E		jrausel1@ford.com	115
108	Alejandro	San Juan		3	PLT_E		asanjua1@ford.com	105
109	Elena	Zarzo		3	PCD_E		ezarzoma@ford.com	115

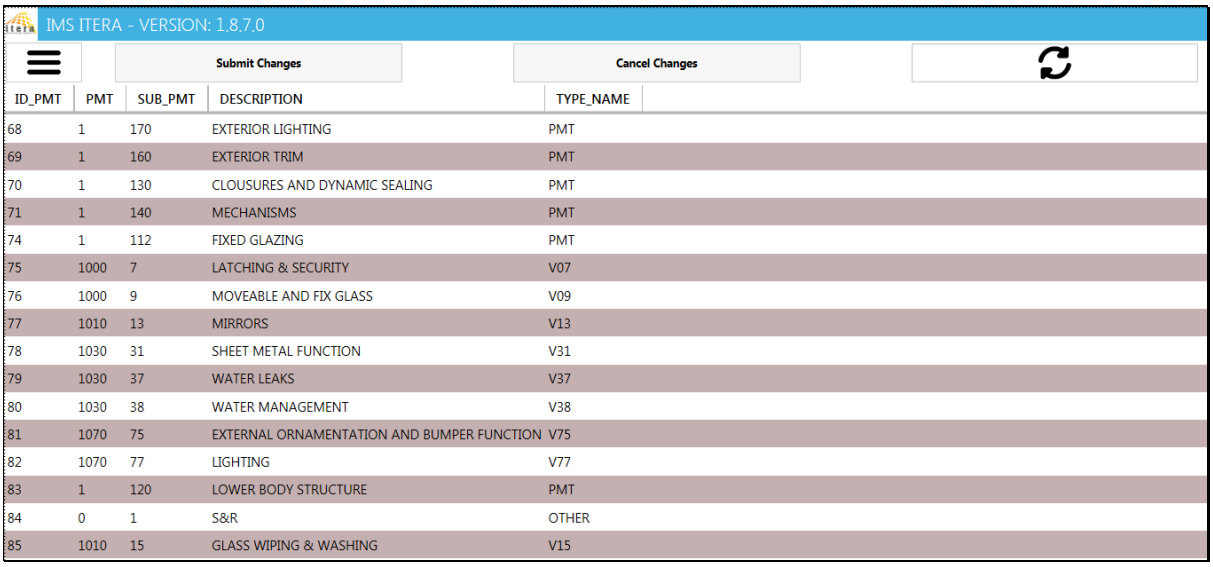
Ilustración 28. Gestión de usuarios.

PMTs/VFGs/Other, listado con diferentes áreas de un vehículo usando el formato que emplea Ford Motor Company.

- PMT, código general que pertenece a Program Management Team (1:Body exterior, 2:Body interior, …) según format Ford Motor Company. No aplicable

para VFG y Other pero se debe rellenar con un identificador (ver ejemplo en la imagen siguiente)

- Sub PMT, código general más commodity (sub-división del área). Para VFG y Other se indica si propio código.
- Description, definición del código "Sub-PMT".
- Type name, identifica a que grupo genérico pertenece (PMT, VFG,...).

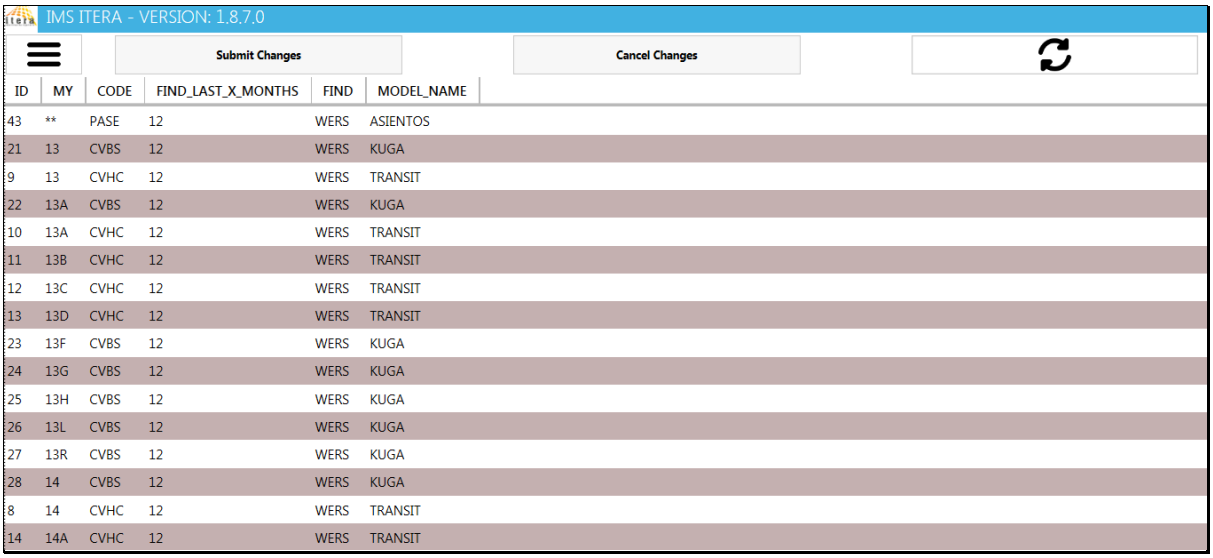


ID_PMT	PMT	SUB_PMT	DESCRIPTION	TYPE_NAME
68	1	170	EXTERIOR LIGHTING	PMT
69	1	160	EXTERIOR TRIM	PMT
70	1	130	CLOUSURES AND DYNAMIC SEALING	PMT
71	1	140	MECHANISMS	PMT
74	1	112	FIXED GLAZING	PMT
75	1000	7	LATCHING & SECURITY	V07
76	1000	9	MOVEABLE AND FIX GLASS	V09
77	1010	13	MIRRORS	V13
78	1030	31	SHEET METAL FUNCTION	V31
79	1030	37	WATER LEAKS	V37
80	1030	38	WATER MANAGEMENT	V38
81	1070	75	EXTERNAL ORNAMENTATION AND BUMPER FUNCTION	V75
82	1070	77	LIGHTING	V77
83	1	120	LOWER BODY STRUCTURE	PMT
84	0	1	S&R	OTHER
85	1010	15	GLASS WIPING & WASHING	V15

Ilustración 29. Ejemplo tabla PMT en IMS

MY/CODE, listado con los diferentes códigos de año y vehículo bajo formato Ford. Aquí se tienen que añadir aquellos códigos de los que se quiera sacar información en la BBDD.

- MY, código del año en formato NNX donde NN son los dos últimos dígitos de un año y X es opcional y es una letra que indica una parte específica dentro del propio año
- CODE es el código de vehículo utilizado por Ford Motor Company
- Find Last X Months es el campo que indica cuantos meses de información se van a solicitar al recoger los datos desde la fuente.
- Model name es un campo que ayuda al usuario a identificar más rápidamente con que modelos se está trabajando, el valor que se guarda es el nombre comercial del vehículo.



ID	MY	CODE	FIND_LAST_X_MONTHS	FIND	MODEL_NAME
43	**	PASE	12	WERS	ASIENTOS
21	13	CVBS	12	WERS	KUGA
9	13	CVHC	12	WERS	TRANSIT
22	13A	CVBS	12	WERS	KUGA
10	13A	CVHC	12	WERS	TRANSIT
11	13B	CVHC	12	WERS	TRANSIT
12	13C	CVHC	12	WERS	TRANSIT
13	13D	CVHC	12	WERS	TRANSIT
23	13F	CVBS	12	WERS	KUGA
24	13G	CVBS	12	WERS	KUGA
25	13H	CVBS	12	WERS	KUGA
26	13L	CVBS	12	WERS	KUGA
27	13R	CVBS	12	WERS	KUGA
28	14	CVBS	12	WERS	KUGA
8	14	CVHC	12	WERS	TRANSIT
14	14A	CVHC	12	WERS	TRANSIT

Ilustración 30. Ejemplo tabla MY/CODE en IMS

7.3.4.1. Tablas DB secundarias

DB Others: Tablas secundarias de información (Issue Type Itera, Historial de Acceso,...) que pueden ser manipuladas para añadir o eliminar información por parte de usuarios con permisos

7.3.5. Settings

La aplicación dispone de un apartado para cambiar parámetros del software y personalizar la aplicación. Los parámetros más importantes a añadir o cambiar son los que se refieren a los datos de usuarios que utilizará el programa para acceder a los emuladores de los terminales Mainframe 3270.

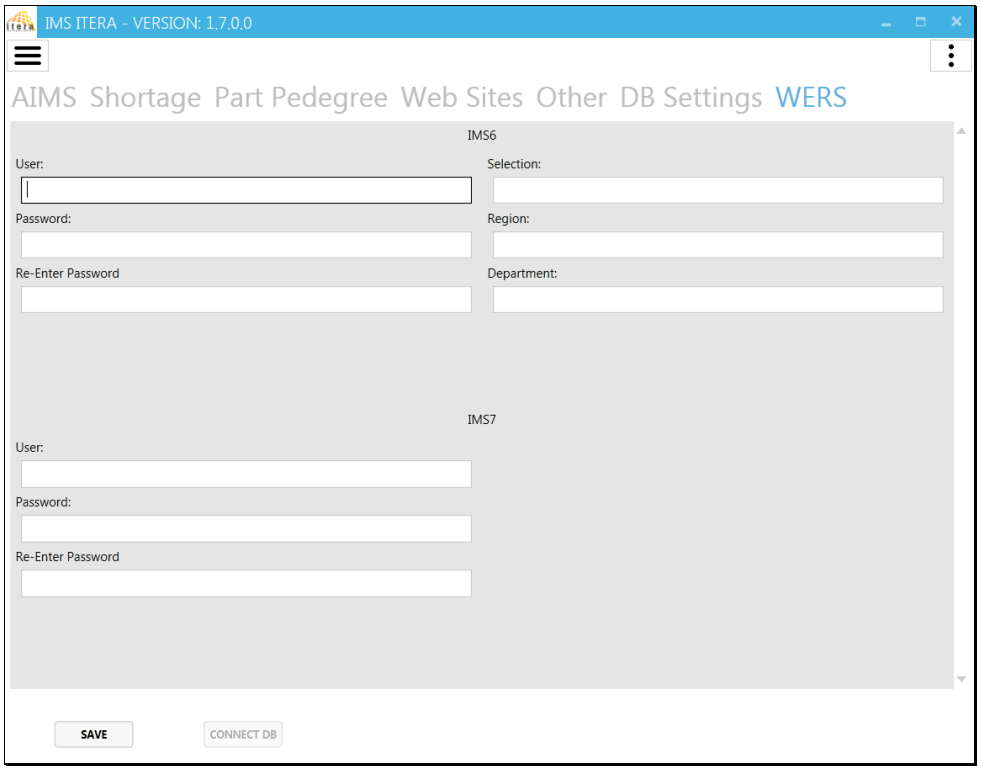


Ilustración 31. Opciones de personalización de la aplicación para cada usuario.

Para poder abrir un ítem del Terminal Mainframe o actualizar datos se debe proporcionar al programa los datos de acceso en la pestaña “WERS”

- User: [usuario en WERS o CMMS3]
- Password: [password en WERS o CMMS3]
- Selection: 16
- Region: E
- Department: BODY

Para poder utilizar estas funciones se debe tener instalado el programa IBM PERSONAL COMMUNICATIONS

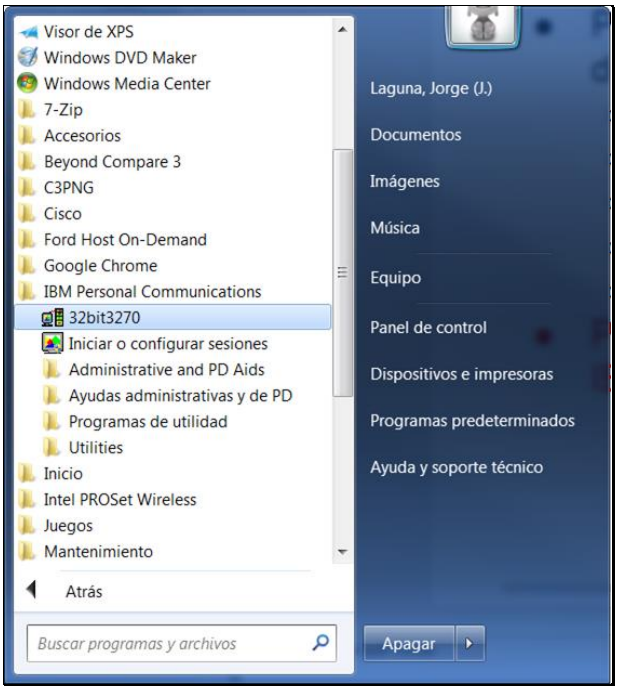


Ilustración 32. IBM Personal Communications instalado en Windows 7.

Las otras dos opciones que tenemos para personalizar la aplicación es la introducción de los datos para la conexión a la capa de datos en los que podremos introducir los datos para crear la cadena de conexión como: provider, initial catalog, user, password y data source que la podremos introducir a mano o bien pulsando el botón “Browse”.

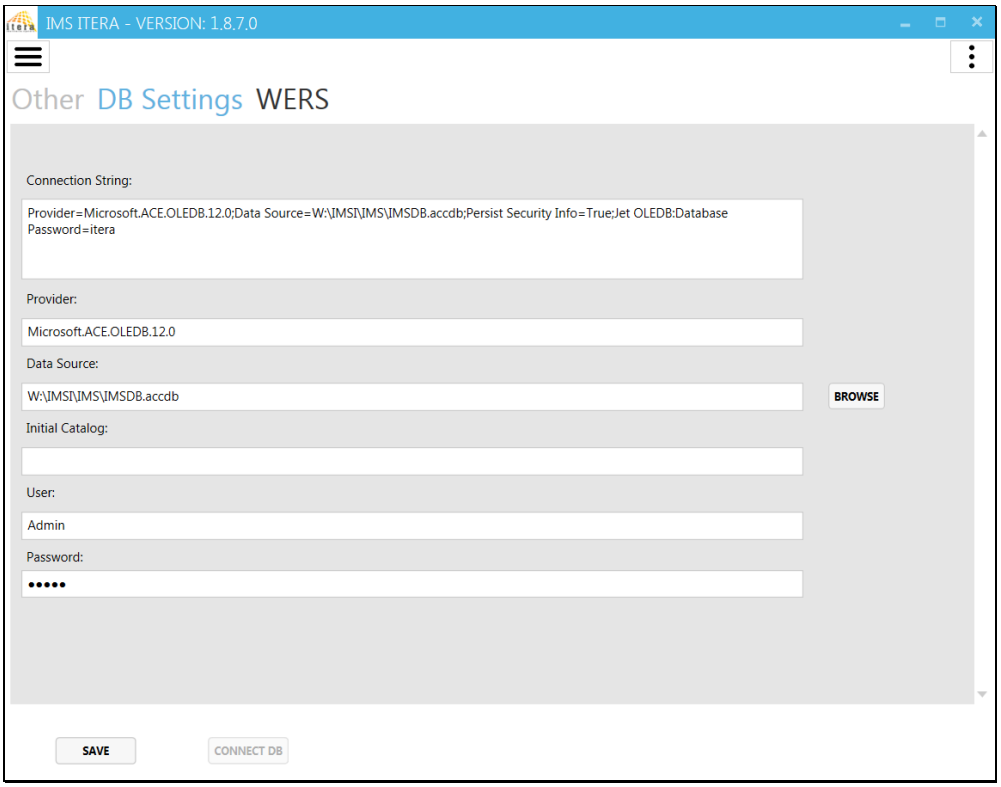


Ilustración 33. Ventana Settings en IMS

La última pestaña de opciones que podremos utilizar es Other en la que podemos personalizar la altura de la fila del archivo Excel que podemos generar con el listado de cada documento, elegir el número de conexión que por defecto es 1 y por último a que correos electrónicos queremos enviar los resultados de las actualizaciones que realicemos ya sea a través de la opción “Get Data” del menú o “Quick Update” de cada documento.

7.3.6. Actualización información

Get Data: Página que permite la actualización de la información de los datos desde la fuente que puede ser utilizada por usuarios con permisos

8. Actualización de la aplicación

Debido a las restricciones de permisos en los ordenadores en cuanto a la instalación de programas y características en Windows que existen en los ordenadores del lugar de trabajo donde los usuarios van a utilizar la aplicación, el método de actualización tiene unas características especiales.

Se ha optado por distribuir el software mediante una solución portable. Los usuarios tienen que ejecutar un ejecutable de un archivo comprimido que contiene todos los archivos necesarios para la ejecución del programa y descomprimirlo en una carpeta local de su ordenador.

Se avisará a todos los usuarios mediante correo electrónico de las nuevas actualizaciones y en este correo electrónico se recordarán los pasos para poder instalarlo en el ordenador y para acceder a él. Además en el propio correo se notificará de los cambios más significativos que tiene la nueva versión.

8.1. Ejemplo del correo electrónico de notificación de una actualización

Hola a todos,

Hay una nueva actualización del software IMS, para que los Responsables y Team Leaders comprueben los cambios antes de enviar la actualización a todos los usuarios, en:

(a partir de ahora procederemos de esta forma, mandando primero la actualización para que Responsables y Team Leaders comprueben los cambios e informen de posibles errores y unos días después (de 2 a 4 días) se enviará la actualización a todos los usuarios)

W:\IMSI_OTROS\release candidate

IMP-> Tenéis que instalaros la última versión que está en esta carpeta (1.8.1.0 RC) (para recordar cómo se instala buscar en este correo la sección *Cómo lo instalo*).

El cambio más significativo en esta nueva versión es la **modificación de filtros y datos en BSAQ, posibilidad de asignar tareas a varios usuarios a la vez, cambio de ordenación del listado de ingenieros basado en grupos de trabajo, mejoras en la funciones Get Data, uso de Opening Date and Closing Date en la pestaña Itera, nueva información en Dahboard,…**

En la siguiente dirección está un archivo con el Control de Cambios que se está llevando en el Software y en donde podéis ver todos los cambios, mejoras y solución de problemas por versión. Los últimos cambios añadidos son los de las versiones 1.8.0.7 y 1.8.1.0

W:\IMSI\CONTROL DE VERSIONES IMS.docx

El software ha sufrido bastantes cambios internos, os agradecería que si os aparece algún error me lo comentaseis como muchos de vosotros ya soléis hacer.

Recordar que en la carpeta de W lo único que tenéis que hacer es ejecutar el archivo comprimido IMS[version], el resto de acciones (utilizar el programa, …) siempre desde la carpeta local de vuestro ordenador donde lo habéis descomprimido.

Como hablamos en las reuniones os pido que le echéis un vistazo y los que no lo habéis utilizado busquéis al menos un par de Concerns, Alerts, piezas en el BOM, Issues internos, … que llevéis y os lo asignéis utilizando la pestaña de "Itera".

Recordar que el software **"IMS Itera" está desarrollado con el fin de centralizar la información** con la que se trabaja día a día (concerns, alerts, aims, issues internos, …).

La app está en fase de pruebas (fase beta). Para cualquier duda, problema, bug, etc., seleccionar la opción USER -> REPORT A PROBLEM del menú superior de la ventana de la aplicación, también podéis mandarme un correo a jlaguna7@ford.com o jorgelh@iteraengineering.com y os contestaré lo antes posible o podéis venir directamente al primer piso de las casetas de lanzamiento.

A continuación tenéis un breve resumen para la instalación y la ejecución

Cómo lo instalo:

W -> W:\IMSI\IMS Install

En W está el archivo ejecutable, **IMS [version].exe**, que descomprime la app en la carpeta que se especifique (por ejemplo en el escritorio, **NO descomprimir el ejecutable en "W:\"**).

Esta carpeta **debe ser una carpeta LOCAL** del ordenador de cada uno (en el caso que no sea la primera vez que se instala en vuestro ordenador local **tenéis que elegir la misma carpeta que la última vez**. Si la última vez fue vuestro Escritorio entonces volver a seleccionar el Escritorio).

El ejecutable de la aplicación, después de descomprimir el archivo de W, estará en la carpeta IMS/ en vuestro ordenador y se llama también IMS.

**Los usuarios y las contraseñas son ambos el CDSID de cada uno para el primer uso (puede ser que la contraseña esté vacía en algunos casos).*

**La base de datos está ya configurada y está centralizada para todos los datos.*

Cómo lo utilizo:

Ejecutar la aplicación que habéis descomprimido en vuestro ordenador.

Introducir el usuario y contraseña (ambos son el CDSID para el primer uso).

Elegir la información que se desee consultar (concerns, alerts, Issues(AIMS, …), BOM+(BOM, …), …).

Usar los *filtros* y luego pulsar el botón *Show* (la primera vez que se solicitan datos, si son muchos, puede ir un poco lento).

Seleccionar algún ítem en el listado vertical y consultar su información

**Si se quieren ver todos los datos desmarcar la opción My Stuff que se encuentra debajo del botón Show y volverlo a pulsar.*

9. Mejoras por versiones

9.1. Control de versiones IMS Itera

IMS versión 1.8.0.7

- Mejora software. Actualización AIMS con dataadapters y flexibilidad en la plantilla a descargar de la información fuente.
- Mejora software. Control de errores ventana Terminal en actualización Concerns y Alerts.
- Se corrige bug. Se utiliza una nueva función para recoger toda la información de Summary Of Parts en la actualización de Concerns y no solo la información de la actividad por defecto.
- Se añade dato. La identificación en el documento será [modelo] + [num bsaq]
    - BSAQ
- Se añade dato. La fecha Closing Date se actualizará al poner un elemento como Priority Itera CLOSED
    - (TODOS)
- Se añade dato. La fecha Opening Date se actualizará al asignar un elemento a un ingeniero
    - (TODOS)
- Se mejora función. Al actualizar BSAQ se mostrará una ventana para la elección del modelo a actualizar
    - GET DATA
- Se corrige bug. Reseteo correcto de los campos al devolver un resultado con o resultados (no se borraban los campos)
    - (TODOS)
- Se mejora función. Solo mostrará un aviso de versión no actualizada cuando la versión que se está ejecutando es menos a la última versión que indica la BBDD de IMS.
- Se corrige bug. Part number (press enter to search) no devolvía resultados.
- Se añaden filtros. Filtros fechas para los campos de ICA y PCA
    - BSAQ
- Se cambia dato. En la pestaña Alertas se mostrarán “Días hasta expiración” en vez de “Días hasta Status C”
    - DASHBOARD

- Se añade dato. Pestaña BSAQ con ICA/PCA overdue; ICA/PCA actual date; ICA/PCA empty date;
    - DASHBOARD
- Se mejora función. La Auto Asignación tendrá dos versiones (larga y corta).
    - GET DATA
- Se modifica interfaz. Se mueve el filtro Status Itera al expander Itera+
    - (TODOS)
- Se corrige bug. En algunas circunstancias la función User Default Settings, recuperaba los datos de la versión anterior.
- Se mejora función. Los Concerns y Alerts en Status X se tratarán de la misma forma que los que están en Status C.
- Se corrige bug. La búsqueda CPSC (press enter to search) no eliminaba los resultados de la búsqueda anterior.
    - BOM
- Deprecated. No se guardará a partir de esta versión el campo Historial al actualizar Concerns y Alerts desde WERS.
- Mejora software. Control de errores ventana Terminal en actualización Events.
- Se añade función. Botón creación de Tareas para varios usuarios. Se eligen los usuarios, el título, el PMT/VFG, Priority y Comentario
    - ISSUES ITERA
- Interfaz. En la barra de botones de acción se añade el botón Crear Tareas y solo estará habilitado en los documentos en los que esté disponible.
    - (TODOS)

IMS versión 1.8.0.0

- Se añade función. Visualización del log de eventos y log de errores en el memo de la app.
- Mejora usabilidad. Se centraliza la confirmación de modificación de datos Itera en una nueva barra de botones New, Update, UpdateSave, Delete que se encuentran por encima de las pestañas.
- Mejora usabilidad. Los botones de actualización New, Update, Updateave, Delete solo se activarán cuando sea posible utilizarlos.
- Se añade Documento. BSAQ.
- Se añade Actualización. BSAQ.
- Se corrige bug. Mostrar ordenador listado PMT
    - AIMS
- Mejora interfaz. Indicación ratón loading al hacer login con usuario.

- Mejora interfaz. Indicación ratón loading mientras carga información en la selección de ítems.
- Se corrige bug. Mostrar ordenados los subniveles del árbol de empleados
- DASHBOARD
- Mejora programación. Los comentarios se preprocesan sustituyendo comillas dobles por comillas simples
    - TODOS
- Se corrige bug. Se carga correctamente desplegable MY/CODE
    - ISSUES ITERA
- Se añade dato. El listado de PMT pasará a ser PMT + VFG
    - TODOS
- Se corrige bug. Búsqueda “Base (press enter to search)”
    - BOM
- Mejora interfaz. El listado de PMT + VFG se mostrará con el siguiente formato ([tipo] – [código] – [descripción])
    - TODOS
- Se corrige bug. Reset campos ICA y PCA
    - Issues Itera
- Mejora interfaz. Los menús flotantes se mostraran automáticamente al pasar el ratón por encima de sus botones de acción.
- Mejora interfaz. Los menús flotantes se ocultarán al sacar el ratón fuera de la zona del menú.
- Se añade dato. Nueva pestaña Itera Associations, donde se podrán guardar y buscar las relaciones de un ítem con ítems de otros documentos
    - TODOS
- Se corrige bug. Cambio en la nueva ventana de WERS de introducción (texto ims6).
- Se mejora función. Comprobación de conexión correcta a BBDD
- Se mejora función. Las Alertas en X no contarán como alertas sin autorizar
    - DASHBOARD
- Se añade dato. Nueva pestaña multimedia en Itera para aquellos documentos que precisen incorporar datos
    - AIMS, ISSUES ITERA, BOM
- Se añade dato. Se añade campo “Send Email CC” para añadir cualquier ingeniero como copia en el correo automático de actualización de información en la pestaña Itera
    - TODOS
- Mejora interfaz. Los datos requeridos estarán coloreados de naranja e indicados con un asterisco

    - TODOS
- Mejora interfaz. Reordenación de los campos de la pestaña Itera
    - TODOS
- Se corrige bug. Se controla excepción que provocaba el cierre de la app al maximizar una imagen vacía
    - BOM, ISSUES ITERA
- Se corrige bug. Se muestra advertencia al intentar adjuntar una imagen o documento si tener seleccionado un ítem
    - BOM, ISSUES ITERA, AIMS
- Deprecated. Se eliminan tablas t_model, t_alert_part, t_alert_program, t_concern_program, t_part_concern, t_pmt_chunk, t_supplier.
- Deprecated. Se eliminan documentos Model, Supplier.
- Deprecated. Se eliminan campos Concern, Alert.
    - EVENTS
- Deprecated. Se eliminan campos Concern search, Alert search para ICA y PCA
    - AIMS
- DB. Se añaden los nuevos campos de asociación en todas las tablas de documentos.
- Mejora interfaz. Issues Type se pone en grupo DB Tables.
- Mejora interfaz. CPSC, Vehicle Line se ponen en grupo DB Others.
- Se añade dato. Posibilidad de ir al ítem del documento asociado desde la pestaña Associations
    - TODOS
- Se añade dato. Se crea un nuevo historial en una tabla independiente con los cambios realizados por Itera en un ítem de un documento
    - TODOS
- Se corrige bug. Borrar imagen al pulsar el botón Refresh
    - ISSUES ITERA
- Mejora informes. Campo PMT/VFG, Issue Type y Priority se mostrarán como texto descriptivo en vez de como id
    - EXCEL generado
- Se corrige bug. Al generar un Excel vuelve a estar disponible la posibilidad de incluir las imágenes que se han adjuntado a cada ítem
    - EXCEL generado
- Deprecated. El campo Status no se coleará en ningún caso
    - ISSUES ITERA

- Se añade dato. Los títulos de los campos que incluyan el texto “DEPRECATED…” (y en algunos casos también estarán tachados) significan que dejarán de utilizarse en futuras versiones, también se mostrará la explicación como un tool tip del propio campo.
- Mejora usabilidad. Al pulsar Quick Update se solicitará una confirmación adicional para ejecutar la función
    - CONCERNS, ALERTS
- Se añade dato. Follow Up (DEFCON 6) en las prioridades que significará que se realiza un seguimiento sobre un ítem externo a itera pero no se realizan acciones sobre él para solucionarlo
    - TODOS
- Mejora usabilidad. Preproceso del texto Status History para cada comentario en diferentes líneas
    - AIMS
- Se añade dato. Campo Released se muestra en Datos Fijos
    - CONCERNS
- Se añade filtro. Filtro para seleccionar solo los que están hechos Released
    - CONCERNS
- Se añade filtro. Filtro para VFG, CCC, Process-Design-Supplier, Team Leader, Author como listados
    - BSAQ
- Se añade filtro. Filtro para Function como desplegable
    - BSAQ
- Se añade información. Nuevos Issues Type, Glidepath, CCC 6Panel, Deep Dive 6Panel, 5D, FCPA Alert.
- Se añade información. Datos prioridad BSAQ en Summary
    - DASHBOARD
- Se añade dato. Closing date que se pondrá automáticamente al poner una prioridad a CLOSED
    - TODOS
- Se añade dato. Opening Date que se pondrá automáticamente al hacer una asignación a un ingeniero
    - TODOS
- Se corrige bug. Se aumenta el número de pantallas que se leen en WERS (description, etc)
    - GET DATA

IMS versión 1.7.0.0

- Mejora software. Mejora de la velocidad de carga al buscar resultados con los filtros
    - (TODOS)
- Mejora software. Mejora de la velocidad de carga al buscar resultados con “Press enter to search”
    - (TODOS)
- Mejora software. Mejora de la velocidad de guardado de información en BBDD
    - (TODOS)
- Mejora software. Se indica la carga al abrir una página de la app y se mejora la velocidad.
- Se añade función. Auto asignación basada en la tabla de BBDD t_auto_assign
    - GET DATA
- Mejora usabilidad. Ciertos campos de información se autoajustan al espacio disponible según el tamaño de la ventana.
- Mejora programación. Creación de plantillas para la estandarización de documentos.
- Mejora programación. Estandarización de selección de ítem actual.
- Mejora software. Se crean tareas en nuevos hilos para realizar las búsquedas de BBDD
    - (TODOS)
- Se corrige bug. Dirección web del formulario Report a Problem.
- Mejora usabilidad. Se aumenta el tamaño que ocupan los filtros
    - (TODOS)
- Se añade dato Status A, W para Concerns y Alerts
    - DASHBOARD
- Se añade dato. Listado de Concerns y Alerts
    - DASHBOARD
- Mejora usabilidad. Solo se mostrarán los ingenieros activos en los listados de ingenieros (desplegables y filtros)
    - (TODOS)
- Se corrige bug. Colores en el campo priority
    - (TODOS)
- Se añade dato. Se permite buscar por “elemento sin dato” en el filtro Priority Itera
    - (TODOS)
- Se añade dato. Visualización de la versión en los títulos de las ventanas.
- Mejora usabilidad. Se indica app trabajando mientras se está cargando cualquier documento.
- Se añade dato. Campo que indica estado Released o no
    - CONCERNS
- Se añade dato. El correo automático incluirá la prioridad y la descripción o título.
- Mejora programación. Optimización de la búsqueda de un ítem en el listado del treeview

    - (TODOS)
- Mejora usabilidad. Después de la búsqueda de un ítem con el campo “press enter to search” se selecciona el primer elemento encontrado
    - (TODOS)
- Se corrige bug. Se controla la excepción que cerraba la app al intentar abrir un Concern o una Alerta en el terminal sin tener instalado IBM PCOMM.
- Se corrige bug. El filtro Oldest Newest
    - (TODOS)
- Se añade filtro. Oldest/Newest
    - ISSUES ITERA
- Se corrige bug. Calculo de Pending y No asignados en pestaña Summary
    - DASHBOARD
- Deprecated. Botones ordenación de los listados
    - (TODOS)
- Se añade tabla en DB. Tabla auto asignación.

IMS versión 1.6.0.0

- Mejora usabilidad. Estandarización niveles prioridad (Help Required, Red Leakage, Open, Pending, Closed)
    - (TODOS)
- Modificación de campo. El campo Status Itera pasa a ser un campo libre para ampliar la información de Priority
    - ISSUES ITERA
- Mejora seguridad. Solo acceso a usuarios con nivel Itera o superior
    - DASHBOARD
- Nuevo Dato. Relación ingeniero-responsable en tabla t_engineer de BBDD.
- Mejora usabilidad. Los menús y los filtros se añaden como ventanas flotantes dejando el resto del espacio a la información.
- Mejora usabilidad. Se integra en la ventana principal de la app
    - SETTINGS
- Mejora seguridad. Solo acceso a búsqueda de datos ingeniero a nivel admin (responsables) o superior
    - DASHBOARD
- Mejora usabilidad. Se estandarizar menú superior de acciones (Show, filtros, report, …)
    - (TODOS)
- Se añade dato. CPSC code de la primera pieza
    - CONCERNS, ALERTS

- Se añade dato. Activity del CPSC code de la primera pieza
    - CONCERNS, ALERTS
- Mejora usabilidad. Se añade ingeniero HODGE PODGE. Será un cajón de sastre para todos los ítems que sean de otras áreas.

IMS versión 1.5.0.0

- Se añade dato. Un pictograma permite indicar a que parte del vehículo pertenece un ítem (una zona por ítem)
    - (TODOS)
- Se añade filtro. Se puede filtrar por zona del vehículo en la pestaña de filtros Itera+
    - (TODOS)
- DB. Se actualiza campos Car Location en todas las tablas respectivas.

IMS versión 1.4.0.1 (1.4.0.0 HotFix)

- Se corrige bug. Al buscar un ítem ya no se añade a la lista (antes no limpiaba la lista y se añadía al final)
    - CONCERNS

IMS versión 1.4.0.0

- Mejora Software. Optimización recursos al actualizar desde WERS para evitar posibles errores.
- Información Deprecated. MY-Code list y Activity List ya no estarán soportados (se utilizará la información de la pantalla eng. concern control y lead)
    - CONCERNS, ALERTS
- Se corrige bug. Correcta visualización en pestañas Itera al cambiar tamaño de la ventana.
- Se añade dato. Estandarización de la pestaña Itera para tener como mínimo los mismos datos: User, Eng., PMT, Status, Issue, Priority, Fix Date.
    - (TODOS)
- Se añade filtro. Todo los elementos básicos (Eng, PMT, Status, Issue Type, Priority, Fix Date) de la pestaña Itera tendrán sus filtros correspondientes en la sección de filtros de Itera.
    - (TODOS)
- Mejora usabilidad. La información de Record_Status se muestra junto a la información Count al inicio de los datos fijos
    - (TODOS)
- DB. Se actualiza la BBDD para guardar la nueva información de la pestaña Itera.

- Mejora usabilidad. La información de asignaciones pasa de la pestaña Itera a Historial.
    - (TODOS)
- Mejora usabilidad. El campo prioridad de Itera cambiará de color según la prioridad asignada.
- Se añade setting. Se puede poner que correos recibirán los resultados de actualización de IMS.
- Mejora usabilidad. Se añade opción de abrir ítems en WOW (Wers On the Web)
    - CONCERNS, ALERTS

IMS versión 1.3.1.0

- Se añade dato. Planta afectada, Producción afectada, Supp docs
    - CONCERNS, ALERTS
- Se corrige bug. Filtros Itera
    - (TODOS)
- Se añade filtro. Planta Afectada en Filters+
    - ALERTS
- Mejora usabilidad. Se cambia fuente a Courier para una correcta visualización de las tablas descargadas de WERS
    - (TODOS)
- Mejora programación. La conexión es única.
- Mejora programación. Clase base de datos básica.
- Mejora Reports. Aparece el nombre de ingeniero en vez de su ID al generar un informe de resultados.
- Mejora usabilidad. Se omite campo Default Activity
    - CONCERNS
- Se añade opción. Envió de correo a responsables con resultados de cada actualización.
- Se añade indicador. Tiempo que permanece la aplicación abierta.
- Mejora usabilidad. Se separan el log normal (log.txt) y el de errores (errors.txt)
- Mejora usabilidad. Nuevo estado del registro “ON HOLD” (ventana de 5 días desde que desaparece)
    - (TODOS)
- Se corrige bug. Filtro Oldest/Newest
    - (TODOS)
- Mejora usabilidad. Se añade fecha en los logs de actualización
    - GET DATA
- Se añade opción. Filtrar por Id o Fecha introducción en BBDD IMS
    - (TODOS)

- Se corrige bug. Colores en campo Aging
    - AIMS
- Se añade dato. Clousure Statement
    - ALERTS
- Se corrige bug. Datos filtro custom Filters
    - (TODOS)
- Mejora usabilidad. Se cierra ventana terminal 3270 al acabar actualización.
- Se añade dato. Wers mensaje de la parte inferior del terminal, Build event
    - CONCERNS, ALERTS

IMS versión 1.2.1.0

- Nueva Tabla Tipos de Issue Itera para administradores.
- Se corrige bug (textos) en el menú principal.
- Mejora usabilidad. Modificados los layouts de los filtros tipo “List”.
- Mejora usabilidad. Modificados los layouts de los filtros tipo “Custom Filters”.
- Añadido filtros. Mostrar los más X ítems más recientes o más antiguos.
    - AIMS

IMS versión 1.2.0.0

- Añadido filtro para mostrar los X ítems más nuevos o más viejos
    - CONCERNS, ALERTS
- Añadido filtro Activity que se basa en la información Prim Eng de la pantalla Eng de WERS.
    - CONCERNS
- Modificación filtro. Dato PROGRAM se busca en los datos de la pantalla Eng Conc de WERS
    - CONCERNS
- Modificación filtro. Datos PROGRAM se busca en los datos de LEAD en WERS
    - ALERTS
- Se corrige bug en filtro Record Status
    - (TODOS)
- Para el primer acceso se puede poner contraseña en blanco o el mismo CDSIS.
- Añadido campo AIMS para guardar y buscar
    - ISSUES ITERA
- Mejora usabilidad. Solo se muestra fecha sin horas en algunos campos fechas.

- Mejora usabilidad. “Days till fix point” se muestra en rojo y en negativo si la fecha ya ha pasado.
- Mejora usabilidad. No muestra error al cancelar la carga de una foto
    - ISSUES ITERA, BOM
- Nueva opción. Se puede elegir incluir foto al generar Excel.
- Nueva opción. Se pueden maximizar las imágenes guardadas.
- Se corrige bug al buscar por PMT o Default
    - (TODOS)
- Se corrige bug en el orden de los listados PMT e Ingeniero Itera
    - (TODOS)
- Mejora usabilidad. Reorganización datos
    - AIMS
- Se corrige bug que borraba campo texto de Custom Filters
    - (TODOS)
- Nuevo formato archivo Excel generado.
- Nueva función. Se puede guardar y visualizar archivo PDF
    - AIMS
- Se corrige bug al guardar fecha vacía en actualización
    - AIMS
- Se añade campo Part Itera, Issue Itera, Safe vin, safe vin date
    - AIMS
- Nueva tabla tipos Issues Itera.
- El email recordatorio automático también se envía al último usuario que lo modificó.
- Se corrige bug con el formato de fechas
    - (TODOS)
- Se añaden datos de Itera en los Metadatos del Excel generado.
- Mejora usabilidad. Se puede modificar la contraseña desde el menú USER.
- Gestión. Se registra el ID del ordenador en cada acceso.
- Actualizaciones. Se han añadido actividades.

IMS versión 1.1.1.2

- Recordatorio. Se envía un correo electrónico automático al asignar una pieza a un ingeniero o al pulsar sobre el nuevo botón “[Save] and Send”
    - (TODOS)
- Excel se genera con imagen Itera.
- Diseño. Se modifica la portada.
- Mejora usabilidad. Cambio colores pestañas.

    - (TODOS)
- Añadido posibilidad de incluir una imagen
    - ISSUES ITERA

IMS versión 1.1.1.0

- Se añade documento: Issues Itera
- Se muestra descripción en los campos correspondientes a los IDs (Engineer, PMT, Model, User)
    - (TODOS)
- Se añade colores en el campo Current_Status (Posibilidad de añadir colores a cualquier campo bajo petición)
    - AIMS
- Mejora usabilidad. Separación visible entre comentarios
    - (TODOS)
- Mejora usabilidad. Mensaje para no poner comillas dobles en todos los campos comentario
    - (TODOS)
- Se corrige bug parámetros AIMS en actualización
    - GET DATA
- Se corrige bug fechas generación Excel
    - (TODOS)
- Se deshabilita actualizaciones Shortage, Part Pedegree
    - GET DATA

IMS versión 1.1.0.2

- Se añade documento: BOM
- Añadido posibilidad de incluir una imagen para una pieza
    - BOM
- Añadido campos para información local de las fases EPT, TT, …
    - BOM
- Añadida actualización semi-automática del BOM
    - GET DATA
- Añadido filtro por texto del campo Base
    - BOM
- Añadido campo y filtro por texto CPSC + nombre de ingeniero si existe
    - BOM

- Mejora de usabilidad. Aumento velocidad de conexión a BBDD al arrancar.
- Mejora de usabilidad. Campo Custom Filter busca coincidencias exactas o similares
    - (TODOS)
- Mejora de usabilidad. Al seleccionar un filtro (listbox) cambia de color
    - (TODOS)
- Se habilita QuickUpdate y se corrigen bugs.
    - CONCERNS, ALERTS
- Se corrige bug fechas en la generación de archivos Excel
    - (TODOS)
- Ítems en Status C solo se actualizarán la primera vez que se introduzcan en la BBDD, para posteriores actualizaciones se deberá utilizar QuickUpdate
    - CONCERNS, ALERTS
- Añadido aviso si se pierde la conexión con la base de datos.
- Se crean archivos log con un tamaño máximo de 1 MB.
- Añadida pestaña ICA/PCA para añadir la información local de las Alerts y Concerns que correspondan.
    - AIMS
- Los listados resultantes de los filtros se ordenan de menor a mayor.
    - (TODOS)
- Añadido aviso continuar/cancelar para aquellas consultas que superen los 500 ítems.
    - (TODOS)
- Añadida actividad NL00.
- La búsqueda de actualización se basará en MY/CODE, ACTIVITY, HISTORY, PLANT, DATES
    - CONCERNS, ALERTS (no hay planta afectada en la búsqueda)
- No se podrá modificar un comentario guardado, se tratan como un histórico
    - (TODOS)
- Se añade dato “Fecha última vez encontrado” para diferenciarlo de “Fecha última vez actualizado”. Se utiliza para conocer el estado ACTIVE/DELETED de los ítems con Status C
    - CONCERNS, ALERTS
- Se corrige bug en actualización (tiempo espera y sobrescribir datos xRef)
    - CONCERNS, ALERTS
- Los comentarios vacíos no se guardarán
    - (TODOS)
- Se añade tabla Plantas Afectas.
- En tabla MY/CODE se añade el número de meses que se quieren recuperar.

- Añadido campo Build Event
    - ALERTS
- Mejora usabilidad. Campo introducir comentario más grande.
    - (TODOS)
- Añadido campo Status Itera
    - (TODOS)
- Mejora usabilidad. Mejor redistribución pestaña Itera
    - (TODOS)
- Mejora usabilidad. Filtro Record Status pasa a ser un desplegable
    - AIMS
- Habilitada opción de generar Excel mediante selección de filas en tabla de datos
    - (TODOS)
- Mejora usabilidad. Barra horizontal en los filtros ListBox es siempre visible
    - (TODOS)
- Se añade tabla CPSC.
- Se deshabilita Documentos de la pestaña DOC
    - SHORTAGE, PART PEDEGREE

IMS versión 1.1.0.1

- Ver versión 1.1.0.2

IMS versión 1.1.0.0

- Comprobación versión software.
- Se corrige bug al cancelar proceso.
    - GET DATA
- Mejora de usabilidad en los filtros. Se pueden expandir y contraer grupos de filtros.
    - CONCERNS, ALERTS, EVENTS, AIMS
- Añadido filtro Engineer Itera y PMT Itera
    - CONCERNS, ALERTS, EVENTS, AIMS
- Mejora de usabilidad. Cambio de la interfaz a un estilo Metro.
- Mejora de usabilidad. El contenido se adaptará al tamaño de la ventana.
- Nuevo formato comentarios Itera. Formato fecha-usuario-comentario. No se podrá borrar una vez introducido. En un campo se introduce el comentario nuevo y en otro se visualizará.
    - (TODOS)
- Mejora de usabilidad. Se estandariza todo lo posible filtros.

    - (TODOS)
- Añadido filtro Running Change
    - CONCERNS
- Se añade campo para, introduciendo un ítem, ir de Event a Concern, ir de Concern a Event, ir de AIMS a Concern.
    - CONCERNS, EVENTS, AIMS
- Se añade la fecha de la última actualización en la página GET DATA.
- Se corrige bug en páginas que muestran tablas
    - OTHERS
- Mejora de usabilidad. Las selecciones de los filtros se quedan marcadas.
    - (TODOS)
- Se inhabilita botón QuickUpdate en esta versión
    - CONCERNS, ALERTS
- Añadida tabla Historial Acceso
    - OTHERS

IMS versión 1.0.0.29

- Ver versión 1.1.0.0

IMS versión 1.0.0.28

- Si la contraseña es la contraseña por defecto (CDSID), se solicita una nueva contraseña personal.
- Añadida funcionalidad "Contraseña olvidada" en la ventana login. Crea un correo completo para enviar al administrador (utilizando un campo interno nuevo) y que resetee la contraseña.
- Se añade filtro Vehicle Line
    - EVENT
- Se añade filtro Activity
    - EVENT
- Se añade filtro ítem Active or Deleted (Filtro Record_Status)
    - (TODOS)
- Mejora usabilidad al guardar datos Itera. Puntero ratón muestra estado.
    - (TODOS)
- Mejora usabilidad en listado de ítems. Se añade información Vehicle Line.
    - EVENT

- Mejora en la información de actualización. Se añade información de la fecha de la última actualización
    - (TODOS)
- Se corrige bug en filtros
    - EVENT
- Mejora en la seguridad. Se guarda un historial de acceso.
- Se corrige bug en sección botones principales
    - (TODOS)
- Mejora en la usabilidad. Se actualizan campos automáticamente (concern)
    - EVENT

IMS versión 1.0.0.27

- En barra de menú superior se añade USERS -> REPORT A PROBLEM. Abrirá un formulario donde se pueden mandar problemas, sugerencias, errores que se tienen al utilizar la aplicación
    - (TODOS)
- Al introducir un comentario Itera se guardará la información del ingeniero activo y la fecha de modificación al final de este
    - (TODOS)
- Mejora del tiempo en actualización (Get Data)
    - CONCERNS, ALERTS
- Mejora usabilidad pestañas
    - AIMS
- Opción de introducir un número de Concern e ir a su información
    - AIMS (Pestaña Summary)
- Se corrige bug de actualización en el campo COUNT (Datos fijos)
    - (TODOS)
- Se corrige bug al realizar una búsqueda de un ítem mediante texto (campo búsqueda en datos fijos)
    - (TODOS)
- Se corrige bug con el formato de fechas (afectaba al campo RECORD_STATUS(ACTIVE/ DELETE) al actualizar)
    - (TODOS)
- Se corrige bug al guardar datos de Itera (aparecía un error después de guardar)
    - (TODOS)
- Se corrige bug al actualizar AIMS, Shortage, Part Pedegree. Reseteaba campos Ingeniero Itera y PMT Itera en algunas ocasiones.

- AIMS, SHORTAGE, PART PEDEGREE

IMS versión 1.0.0.26

- Se añade la posibilidad de filtrar por Program – Description
  - AIMS
- Opción para seleccionar los ítems asignados al usuario activo o los ítems no asignados a ningún usuario (My Stuff/Not Assigned)
  - (TODOS)
- Se modifican la imagen de los iconos de reset de cada filtro para una mejor usabilidad
  - (TODOS)
- Guardar un comentario de Itera no depende del resto de datos de Itera (el resto de datos pueden estar vacíos, no modificarse,…)
  - (TODOS)
- Al iniciar el programa se realizan conexiones con la base de datos para mejorar la velocidad de uso posterior.
  - (TODOS)
- Se añade RECORD STATUS que establece si el ítem está ACTIVE o DELETED en la zona de datos fijos.
  - (TODOS)
- Un ítem pasará a DELETED si tiene una antigüedad superior a 7 días y no ha sido actualizado en la última actualización.
  - (TODOS)
- Los diálogos para seleccionar archivos se abrirán en la última ruta utilizada por el usuario en vez de en la ruta por defecto del PC.
  - AIMS, SHORTAGE, PART PEDEGREE

IMS versión 1.0.0.24

- Se añade un botón para mostrar un Concern/Alert seleccionado en la ventana original de WERS.
  - DECK, ALERT
- Aparece una pantalla de selección de MY-PROGRAM y ACTIVITY al realizar una actualización de DECK o ALERT
  - DECK, ALERT
- Los layouts aparecerán centrados
  - (TODOS)

- En la pestaña ITERA aparecerá una tabla con un listado de asignaciones entre Ingeniero, PMT. IMP! ->SE DEBE DEFINIR COMO SE QUIERE TRABAJAR CON ESTA TABLA
    - (TODOS)
- En SETTINGS se podrá modificar los datos personales para acceder a WERS (ims6 y ims7).

IMS versión 1.0.0.23

- Se pueden modificar tablas MY-CODE y ACTIVITY desde app (se necesitan permisos)
- Se puede filtrar por ítems asignados al usuario activo o por No asignados
    - DECK, ALERT
- Guarda lista MY-CODE para cada ítem y se muestra en pestaña summary en la app
    - DECK, ALERT
- Guarda lista ACTIVITY para cada ítem y se muestra en pestaña summary en la app
    - DECK, ALERT
- Filtro por PROGRAM
    - DECK, ALERT
- Los comentarios Itera puede ser Multiline + CheckSpeeling + etc.
    - (TODOS)
- Se muestra el número total de items obtenidos con los filtros en la parte de info fija
    - (TODOS)
- El control de la ventana PCOMM es automática no necesita abrirla el usuario.
    - DECK, ALERT, EVENT
- Se mejora entrada de datos iniciales en ventana PCOMM para evitar problemas de sincronismo
- Los usuarios para WERS, etc., estarán inicialmente en blanco y deberán ser añadidos por cada usuario en la ventana settings.
- Se añade la opción de volver a la configuración inicial en el menú USER de la barra superior
- Se modifican partes de datos fijos y pestañas para mejorar el uso.
    - DECK, ALERT, AIMS
- Se añade la opción de ver todos los registros que no están asignados
    - DECK, ALERT
- El campo ID_ENGINEER_ITERA = 100 significa que no está asignado
- El campo ID_PMT_ITERA = 1 significa que no está asignado
- Al generar un Excel de los datos podrá contener varias hojas y se mostrará la que corresponda a los datos principales.
    - (TODOS)

- Se añade un campo RECORD_STATUS que indicará ACTIVE si está en la última actualización y DELETED si al realizar una actualización no se encuentra y ha pasado más de un día desde que se actiualizó.
    - (TODOS)
- Al guardar los datos de Itera y actualizar la base de datos se mantendrá en el dato seleccionado previamente de forma automática
    - (TODOS)

10. Conclusiones

Durante el desarrollo de esta aplicación me he encontrado con varios problemas que me gustaría comentar a modo de conclusión.

El desarrollo de aplicaciones implica el conocimiento de muchas áreas de la ingeniería software. Hoy en día es altamente improbable encontrarte con un desarrollo que no implique tener una base de datos disponible. Durante la realización de la ingeniería superior en informática y durante la realización del máster para el cual es este proyecto se han visitado por separado los distintos puntos que pueden abarcar un desarrollo software (creación de un aplicación, creación de bases de datos, creación de algoritmos para resolver problemas) pero en ningún momento se ha visto todo esto como un conjunto, ¿qué opciones tengo a la hora de desarrollar un software?, conociendo las diferentes tecnologías de desarrollo software ¿cómo puedo elegir la más correcta para cada caso?, teniendo la base de datos diseñada ¿cómo la uno y la utilizo desde mi aplicación?, … Estas son algunas preguntas que para mí son esenciales a la hora de desarrollar software y considero, siempre desde un punto de vista constructivo, que son preguntas que se deberían contestar antes de que los actuales alumnos se conviertan en trabajadores.

El primer punto de estas conclusiones y que más atención me llama es la negativa de los usuarios a cambiar el modo de trabajo al que están acostumbrados, incluso habiendo probado que las nuevas soluciones implican una mejora en su trabajo del día a día. La negativa es tal que me he encontrado ante la situación surrealista de ver que usuarios que tenían que utilizar la aplicación, no lo hayan hecho y hayan usado su tiempo, y por ende el de la empresa, durante semanas o meses para crear alternativas a parte de las funciones que realiza la aplicación, utilizando por ejemplo hojas de cálculo de Microsoft Excel con macros, en vez de colaborar y dar un feedback positivo para que la aplicación pudiera ir mejorándose mucho más rápido.

Los requisitos por parte de los stackholders suelen ser muy genéricas e incluyen todas las funciones que sean posible incluso si estás muchas veces son contrarias y aun profundizando en las reuniones para especificar requisitos, estos cambian con el paso de tiempo por cambios de opinión por parte de los stackholders por lo que no se consigue un diseño robusto.

El no elegir las tecnologías adecuadas implicará un gran retraso en el desarrollo, desde mi punto de vista existe una relación directa entre el número de recursos que se posee y las tecnologías de desarrollo software a utilizar. A menos recursos más importante es utilizar tecnologías generalizadas que tengan mucha documentación y ayuda. Por lo que si se quiere innovar y utilizar nuevos lenguajes se debe tener muy presente que se debe disponer de grandes recursos para contrarrestar la falta de documentación y ayuda externa.

Las grandes compañías son muy reacias a proporcionar acceso a su información, incluso dentro de la misma compañía. Esto si bien es comprensible desde un punto de vista de seguridad actualmente ha dejado de ser una buena solución ya que una buena gestión de esa información puede dar un valor añadido a la empresa muy importante dándole la posibilidad de identificar los problemas y resolverlos con mucha más eficacia.

Al hilo del punto anterior, las grandes compañías también son muy reacias a cambiar el modo de trabajo que lleva arrastrando en ocasiones décadas, este solo se puede entender al miedo que existe a dejar de utilizar un sistema robusto que no crea prácticamente problemas pero a su vez un sistema arcaico como ocurre en Ford Motor Company con sus sistemas de acceso a la información a través de un emulador de un terminal Mainframe. Esta situación también la podemos ver en el sector bancario donde utilizan sistemas arcaicos pero a su vez altamente contrastados.

La etapa, a mi entender, más complicada de un desarrollo software es la correcta captación de los requisitos por parte del stackholder y plasmar estos en un diseño correcto. No entender, o mal interpretar, una serie de requisitos provocará cambios en el diseño y la implementación. Esta si bien digo que me parece la etapa más complicada también es la que más me apasiona a la hora de trabajar en un desarrollo software porque es donde estas construyendo la base, los cimientos de la aplicación que implicará tener una aplicación robusta y de calidad.

Por otro lado, hay que reconocer que la etapa de captación de requisitos y el diseño de una aplicación suele ser la parte más olvidada, sobretodo en proyectos con pocos recursos, donde se tiende a ir directamente a la implementación y realizar un desarrollo de prueba y

error con lo que el resultado final suele estar lleno de parches y suele implicar mucho más tiempo del previsto inicialmente.

11. Bibliografía

Título:	Gestión de la información y digitalización de procesos en la Oficina de Calidad - FORD VO Valencia
Autor:	LAGUNA HERNÁNDEZ, JORGE
Director(es):	Simó Ten, José Enrique
Entidad UPV:	Universitat Politècnica de València. Escola Tècnica Superior d'Enginyeria Informàtica
Titulación:	Ingeniería Informática-Enginyeria Informàtica
Tipo:	Proyecto/Trabajo fin de carrera/grado

- C#
    - http://en.wikipedia.org/wiki/C_Sharp_(programming_language)
    - http://en.wikipedia.org/wiki/Comparison_of_C_Sharp_and_Java
    - Fundamentals of Computer Programming with C# Author(s): Svetlin Nakov, Veselin Kolev and team. Publisher: Telerik Software Academy Published: 2013 - http://www.onlineprogrammingbooks.com/fundamentals-computer-programming-c/#sthash.1DAzErPt.dpuf
    - * Andrew Stellman (2008). Head First C#. Sebastopol, CA: O'Really Media.
    - * Matthew MacDonald (2007). Pro WPF: Windows Presentation Foundation in .NET 3.0. Berkeley, CA: Apress
- Preliminares wpf

  - http://en.wikipedia.org/wiki/Windows_Presentation_Foundation
  - http://en.wikipedia.org/wiki/Extensible_Application_Markup_Language
  - https://msdn.microsoft.com/es-es/library/cc295302.aspx
  - https://msdn.microsoft.com/es-es/library/aa970268(v=vs.110).aspx

- Mahapps
  - http://mahapps.com/
  - https://github.com/MahApps
  - https://es.wikipedia.org/wiki/Modern_UI

- Capas de negocio y desarrollo software
  - https://en.wikipedia.org/wiki/Multitier_architecture
  - http://lml.ls.fi.upm.es/ep/entornos.html

12. Otros Documentos

Junto a este proyecto se adjuntan una serie de documentos divididos en dos grupos.

* Documentos que contienen código fuente creado para las aplicaciones implementadas en las soluciones de los apartados 7,8,9,10,11.

* Guias de referencia de librerías externas utilizadas en este proyecto

* Presentaciones realizadas para la presentación y uso de la aplicación desarrollada en este proyecto.