UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

DISEÑO E IMPLEMENTACIÓN DE UNA HERRAMIENTA WEB PARA LA GESTIÓN Y CONTROL DE REPASADO MEDIANTE EL USO DE WEB SERVICE Y UN SOFTWARE DE VISUALIZACIÓN DE DATOS

MEMORIA PRESENTADA POR:

Alejandro Gisbert Alonso

GRADO DE *INGENIERÍA INFORMÁTICA*

Convocatoria de defensa: *Mayo de 2016*

Tabla de contenido

1 RESUMEN

El siguiente proyecto es la realización de una aplicación para la gestión de una serie de datos, surgido por la necesidad de cambio en una empresa durante el periodo de prácticas del alumno. En cuanto al aspecto informático de dicha empresa, cuenta con un sistema de planificación de recursos el cual utiliza demasiadas sesiones de usuario en algunas de sus secciones; en concreto en la de repasado de tejido. Por lo tanto, el objetivo principal es la eliminación de estas cuentas sobrantes para, de ese modo, reducir costes a la empresa y además, al utilizar un navegador web como método de introducción de datos, eliminar el uso de papel y bolígrafo.

Para ello, se decide crear una aplicación web ejecutada bajo cualquier navegador que, gracias a los servicios web que proporciona el sistema, posibilite la comunicación entre aplicaciones sin importar el código utilizado. En este caso, las aplicaciones que se comunican son tanto la nueva aplicación web como el sistema de planificación, y un controlador automático utilizado como contametros de la tela. Finalmente, los datos recogidos por la aplicación se verían sometidos a su posterior estudio por un software dedicado a ello.

Palabras clave: Servicio Web, ERP, Navision, SQL, EMS, PLC, Formación, QLIKVIEW, VPN, Remoto, Control, Barcode, Identificación, Contametros, Codeunit, Visual Studio, ASP.NET, VB

ABSTRACT

The following project is the development of an application for the management of a series of data, which came up as a need for change in a business during a student's work experience period. Regarding the IT aspect of this company, it includes an enterprise resource planning which uses too many user sessions in some of its sections; more specifically in the material revision. Therefore, the main objective is the removal of these spare accounts to reduce some of the company's costs and also, by using the web browser as a data inputting method, to avoid the use of pen and paper.

To do so, we decided to create a web application that can be run under any browser which, because of the web services provided by the system, enable the communication between apps without worrying about the code that is used. In this case, the communicating apps are the new web application, the enterprise resource planning, and an automatic controller used as a meter counting device. In the end, the data collected by the application would be subject to further study using a dedicated software.

Keywords: *Web Service, ERP, Navision, SQL, EMS, PLC, Training, QLIKVIEW, VPN, Remote, Control, Barcode, Identification, Meter Counting Device, Codeunit, Visual Studio, ASP.NET, VB*

2 INTRODUCCIÓN

El diseño e implementación de una herramienta web se deriva en una aplicación con el objetivo de facilitar la gestión de los datos recogidos y su posterior estudio dentro de una empresa. En el caso de esta empresa de textil, se centra en la sección de repasado, donde se quiere tener un control de incidencias por tipo de tela, producto utilizado, usuario, etc. De esta manera, los encargados tendrán una visión más amplia sobre las consecuencias de utilizar un determinado producto en una tela o incluso el tiempo medio que se emplea en el repasado de la misma.

Por otro lado, el personal que vaya a utilizar la aplicación será el encargado de registrar tanto las incidencias que se vayan encontrando a lo largo del repasado como el lugar donde se encuentren, y automáticamente se calculará el tiempo dedicado al mismo, los metros repasados totales, etc. Toda esta información se verá reflejada en la base de datos de la empresa para así poder conducirla a un control más exacto.

La finalidad de este proyecto consistirá en el trayecto recorrido durante su realización para así poder ver los resultados en una empresa real.

2.1 OBJETIVOS PRINCIPALES

Los principales objetivos que se han seguido a lo largo del desarrollo del proyecto han sido los siguientes:

- ✓ Sustitución del papel en la introducción de incidencias localizadas en una pieza de tela, por una aplicación web donde se inserten directamente a la base de datos.
- ✓ Eliminación de usuarios conectados simultáneamente al sistema de gestión de la empresa (ERP) con el fin de reducir costes y facilitar la introducción de datos en la base de datos.
- ✓ Visualizar el resultado final mediante una aplicación para mostrar estadísticas filtrando por el campo deseado (operario, metros realizados, tiempo de repasado, etc.)

2.1.1 Objetivos específicos

- ✓ Mediante una autentificación de usuario, controlar el acceso a la aplicación tanto por usuario como por máquina en la que se está trabajando.
- ✓ Poder elegir la pieza a repasar tanto por número de carro como número de la pieza en concreto.
- ✓ Visualizar las incidencias totales encontradas en una pieza o carro para su posterior impresión si fuera necesario.
- ✓ Imprimir el albarán de todas las piezas repasadas en un mismo carro, junto con los datos necesarios para su control durante el envío a otra empresa.
- ✓ Controlar el tiempo que ha durado el repasado, desde que se inicia hasta que se hace un parón o se finaliza el repasado de la pieza.
- ✓ Controlar los metros repasados de cada pieza mediante el uso de un Controlador Lógico Programable (PLC).
- ✓ Facilitar el cambio de una pieza a un carro nuevo sin salir del repasado de la misma.
- ✓ Desechar una pieza inutilizable a un carro especial.
- ✓ Introducir incidencias indicando el tipo de incidencia, la cantidad de las mismas, el metro o metros donde se ha/n encontrado y si ha/n sido arreglada/s o no.
- ✓ Visualizar las incidencias detectadas en la pieza seleccionada durante el repasado.

Muchos de estos objetivos específicos fueron apareciendo mediante el desarrollo de la aplicación ya que las pruebas realizadas con los operarios ofrecían nuevos propósitos y a su vez era necesaria la modificación de algunos otros. Con esto, se iban puliendo tanto la apariencia final como pequeñas funciones de la propia aplicación.

2.2 MOTIVACIÓN

El motivo principal por el cual quería realizar este proyecto vino a causa de un contrato de prácticas en una empresa textil en Alcoy, la cual se dedica a la industria de tejidos de fibras duras y mezclas. Además, en una de sus secciones se encarga del repasado de imperfecciones o incidencias de las diferentes piezas de tela de las que dispone. En esta parte en concreto se quería informatizar, como ya he dicho, la recolección de datos para su posterior estudio. Por ello, una forma de hacerlo, era sustituir el papel por un ordenador a la hora de introducir los datos en el sistema. Esto dio como idea el diseño de una aplicación web fácil y cómoda de utilizar, para que cualquier empleado y desde cualquier dispositivo se pudiera realizar dicha introducción.

Me pareció interesante la idea de diseñar una web ya que siempre había sido algo que me había atraído, y de ese modo podría mejorar mis conocimientos orientados en este campo con el fin de obtener mayor competitividad en un futuro. Además, el tema de aprender un poco más en cuanto a las bases de datos me pareció algo muy importante. Por otra parte, la empresa utilizaba el ERP (sistema de planificación de recursos empresariales) Microsoft Dynamics Navision con el que nunca había trabajado anteriormente. Así que gracias a eso podría tener la facilidad de aprender nuevas herramientas útiles que en una empresa real se utilizan diariamente.

En cuanto a la programación, era la primera vez que tocaba el lenguaje de programación de ASP.NET y por supuesto era la primera aplicación real en la que lo utilizaba. Sí que había trabajado anteriormente con el lenguaje PHP en algunas tareas por lo que no fue tan difícil adaptarme. Además tuve la oportunidad de utilizar algunas plantillas de CSS con las que pude optimizar la web y dejar un estilo visualmente más atractivo.

A lo largo de todo el proyecto la motivación fue aumentando a causa de querer seguir aprendiendo cosas nuevas y al mismo tiempo saber que era algo que un futuro iba a ser, de un modo u otro, útil para la empresa.

2.3 DESCRIPCIÓN DEL PROYECTO

Como ya he dicho en otros apartados, el proyecto tenía como objetivo la implantación de una aplicación web con la que, mediante el uso de servicios web, se pudieran comunicar los usuarios con los datos de la base de datos del Navision (actualmente utilizado en la empresa) en un navegador web cualquiera dentro de la empresa.

Anteriormente en cada máquina se instalaba el ERP para la introducción de incidencias, por eso era necesaria una cuenta distinta para cada una de ellas. Esto con el tiempo se fue eliminando y sustituyendo en algunas secciones del repasado con el uso de papel y bolígrafo como apaño hasta que surgiera una idea mejor. Con el uso de una aplicación capaz de comunicarse con el servidor de la base de datos, sería posible eliminar numerosas cuentas en esa sección y, con un simple navegador web y una sesión de usuario sería posible desarrollar el mismo trabajo de una manera más dinámica y visualmente más fácil de entender para los usuarios.

2.4 ACTIVIDAD PRINCIPAL DE LA EMPRESA

La empresa actualmente está enfocada en el sector textil, centrándose en la industria de tejidos de fibras duras y mezclas. Consta de diferentes secciones: compra de hilo, tejeduría, repasado y logística

La sección de tejeduría se encarga de la fabricación de la tela mediante el uso de distintas máquinas utilizando el hilo previamente comprado a los proveedores por su sección de comprado correspondiente a dicho hilo.

La sección de repasado gestiona las incidencias encontradas en cada pieza de tela. Cada pieza consta de ciertos metros de tela, la cual ha sido rollada dependiendo de su modelo y procedencia. Por otro lado, las incidencias encontradas durante el repasado de cada pieza se apuntan vía papel, para posteriormente informatizarlas e imprimirlas para el envío a la empresa correspondiente (albaranaje).

La sección de logística se encarga de almacenar y gestionar el envío de las piezas acabadas a las empresas correspondientes.

2.5 PLAZOS

El proyecto no estaba pensado para ser realizado en unos plazos determinados, pero sí que se utilizaron algunas pautas a seguir en cada punto de la realización del mismo. A continuación se procederá a una división de objetivos según unos plazos estimados de unas cinco semanas:

2.5.1 Comunicación con el contametros (1ª semana)

En cuanto se supo qué se iba a realizar, había que familiarizarse con el hardware que se estaba utilizando en esos momentos en la empresa. Una de las herramientas iba a ser un controlador lógico (PLC) Omron diseñado para contar y mostrar el metraje que iba recorriendo la tela durante el repasado. Este PLC trabaja mediante señales de transmisión eléctrica que aprovecha para convertirlas en líneas digitales de transmisión de datos. Estos datos no son más que líneas de bits enviadas por un puerto serie (COM), los cuales hay que traducir al lenguaje normal mediante un código específico dependiendo del controlador.

Una vez comprendido el funcionamiento del PLC haciendo diferentes pruebas con él, había que hacer posible la comunicación mediante la web. Gracias al uso de servicios web con el IDE de Microsoft Visual Studio 2015, se pudo crear una pequeña aplicación, la cual una vez instalada en cada máquina era capaz de recoger el metraje que había realizado cada repasado.

2.5.2 Navision y Servidor Web (2ª semana)

A continuación, había que hacer posible una comunicación cliente – servidor con la base de datos del ERP de Navision y la aplicación web. Como Navision disponía de herramientas para servicios web, con la que mediante “codeunits” podía comunicarse con servicios externos, se propuso utilizar esa tecnología para controlar la inserción de datos en la base de datos. Era necesaria la apertura de una serie de puertos donde se alojaba Navision y, a pesar de tener algunos problemas al respecto, se pudo proceder a la comunicación con la aplicación web mediante algunas pruebas haciendo uso de un pequeño código en Navision.

Además, se proporcionó una máquina virtual con el objetivo de instalar el servidor web en ella. El servidor web instalado fue el Windows Server con rol de servidor IIS.

2.5.3 Microsoft Visual Studio 2015 (3ª y 4ª semana)

Durante unos días únicamente se procedió a hacer pruebas con las diferentes tecnologías. Para empezar, había que familiarizarse con el software de Microsft Visual Studio, así como con el framework .NET y el lenguaje de programación de Visual Basic. Además, para poder ver que funcionaba la comunicación con el contametros a través de la aplicación, había que referenciar el “Web Service” de cada máquina con ésta. Por otra parte, mediante consultas SQL, se iban mostrando algunos datos recogidos en la base de datos de Navision y así comprobar que las funciones eran correctas.

Una vez terminado ese periodo de pruebas, se propuso a comenzar el diseño y montaje de la web (autentificación de usuario, búsqueda de piezas, repasado, impresión, etc.)

2.5.4 Fase de pruebas (5ª semana)

En esta última parte se lanzó la aplicación al servidor web para que fuera utilizada ya por operarios y así se pudieran hallar problemas que durante la programación no se tuvieron en cuenta. De esa manera sería posible su optimización y mejora respecto a los usuarios.

Por otro lado, también se procedió a mostrar ciertas tablas donde aparecían los datos previamente recogidos y de ese modo, mediante gráficos y estadísticas, se podía ver que todo estaba surtiendo según lo previsto. Finalmente, el gerente sería el encargado de estudiar dichos datos.

2.6 EXPECTATIVAS PREVIAS

Previamente se esperaba proveer una forma fácil y rápida con la que se pudiera gestionar toda la información relevante y así ayudar en una futura mejora a la propia empresa. Es decir, podría determinarse si un tipo de tela se consideraría rentable su repasado en función del tiempo dedicado o de la cantidad de incidencias encontradas; o la eficacia de repasado en función de los operarios. Además, se agilizaría el proceso de repasado y así se facilitaría el estudio del mismo; quitando al mismo tiempo diferentes sesiones del sistema de gestión de la empresa.

Todo esto sería posible utilizando alguna aplicación compatible con cualquier tipo de hardware y que a su vez proporcionase información como: tiempo de repasado, metros repasados, cantidad de incidencias detectadas, lugar en que se encuentran dichas incidencias, operario que ha desarrollado el repasado, horas de inicio y fin del repasado, máquina en la que se ha realizado el repasado, pieza de tela que se está repasando, carro en el cual se han almacenado dichas piezas, producto que se ha utilizado o que contiene cada tela, etc.

Ya que hasta el momento estos datos habían sido recogidos mediante papel y bolígrafo, el trabajo se veía en cierto modo ralentizado, poco preciso y no se tenían en cuenta ese estudio estadístico posterior. A causa de todo eso, el gerente de la empresa propuso hacer una herramienta con la que se pudieran gestionar todos los datos recogidos vía web.

2.7 PERSPECTIVAS Y FUNCIONES DEL PRODUCTO

Después de un periodo de discusión, se decidió desarrollar una aplicación multiplataforma, fácil de utilizar y que se adaptara a dichos puestos de trabajo. Por ello, se decidió crear una web dinámica con la que se pudiera comunicar cada máquina con el servidor donde se alojaba la base de datos de Navision. Dado que era necesario un lenguaje de programación el cual permitiera dicha comunicación, se escogió actuar bajo el framework .NET diseñado por Microsoft y que daba una mayor facilidad e independencia en cuanto al hardware utilizado y permitía un desarrollo más rápido de la aplicación. Microsoft Dynamics Navision es un software ERP (“Enterprise Resource Planning”) el cual forma parte también de la familia de productos de Microsoft. Por otra parte, el sistema de bases de datos que contendría toda la información necesaria para ser mostrada en la página web era SQL.

Una vez decidido todo esto, se necesitaría un servidor web donde poder publicar la página web y que permitiera ejecutar el código ASP. El servidor utilizado fue IIS (“Internet Information Services”) ya que adquiere un conjunto de servicios (FTP, HTTP…) que se creían adecuados para el sistema operativo Microsoft Windows, que por otro lado era el que se iba a utilizar en cada máquina y durante la programación de la aplicación.

Finalmente fueron necesarios una serie de ensayos de prueba y error durante la programación de la aplicación para corregir pequeños fallos e incluso añadir nuevas funcionalidades que iban siendo requeridas conforme se iba avanzando en el desarrollo. Dicha web fue diseñada de una forma un tanto homogénea para que independientemente del navegador utilizado ésta fuera capaz de presentar una estética aceptable. Aunque principalmente el navegador que estaba siendo utilizando hasta el momento era Internet Explorer. Además, la resolución utilizada iba a ser de 1024x768 ya que era la más útil en los monitores de los que disponíamos.

Para acabar, los protocolos principales utilizados iban a ser TCP/IP para la conexión de las redes de ordenadores y así facilitar el envío de datos; y el protocolo HTTP, como ya he dicho anteriormente, el cual permitiría el acceso a la web.

2.8 USUARIOS

Dentro de la sección de repasado en la empresa, existen una serie de usuarios los cuales utilizarían de algún modo la aplicación. Cada uno de ellos dispondría de una clave de usuario única que utilizaría para acceder a la web y desarrollar la tarea prevista. Los posibles usuarios serían:

2.8.1 Operarios

Los operarios son los encargados del repasado de las piezas de tela, por lo tanto aquellos que van a utilizar la aplicación en función de la máquina en la que se encuentren. Además, como ya he dicho anteriormente, cada uno dispondrá de su código de usuario para así controlar quien es el usuario que está efectuando el repasado. Estos usuarios serán los que utilicen la aplicación casi por completo.

2.8.2 Encargados

Los encargados, en cuanto a lo que la aplicación se refiere, son aquellos que tienen en cuenta quien está utilizándola de modo adecuado, sobre todo durante el periodo de prueba de la misma. No la utilizarán con el objetivo de repasar pero si para explicar a los operarios como funciona correctamente. Además, posteriormente serán capaces de ver el resultado de cada repasado una vez todos los datos hayan sido recogidos.

2.9 REQUERIMIENTOS

Para completar con éxito los plazos estimados en la realización de la aplicación, harían falta una serie de requerimientos los cuales harían posible la realización de la misma:

- **Consulta de metraje**

Este tipo de información consiste en consultas en la que mediante un PLC conectado a la máquina procede a contar los metros repasados de cada tela. Esta consulta se realiza en tiempo real mediante el uso de un servicio web. Una vez instalado en cada ordenador de cada máquina el software requerido (Servidor IIS, Microsoft Visual Studio en caso de modificación de código, navegador que se iba a utilizar…), se instalaría el servicio web en cada máquina con el objetivo de poder hacer llamadas remotamente a esta y así hacer la consulta requerida.

- **Web Service Navision**

La versión que la empresa estaba utilizando en estos momento era Microsoft Dynamics NAV 2009, el cual proporcionaba “Web Services” y hacía más fácil la integración de otros sistemas. La integración de los “Web Services” con Navision está soportada gracias a los “codeunits”. Dichos “codeunits” hacían el código más fácil de mantener y de esta manera se podía omitir la duplicación de código para todos los servicios web requeridos. Además, con las autorizaciones y autentificaciones correspondientes, un sistema externo, en este caso Microsoft Visual Studio 2015, era capaz de leer la información llamando a este “codeunit”.

- **Familiarización con el framework ASP.NET**

ASP.NET es un framework para aplicaciones web muy útil para el diseño de sitios web dinámicos, el cual se iba a necesitar para generar la información deseada en cuanto a la petición del usuario. Estas páginas de ASP.NET son conocidas oficialmente como “Web Forms” y la extensión utilizada es ASPX. Estos archivos son capaces de contener etiquetas HTML así como controles procesados a través del servidor o el usuario donde se coloca el código dinámico requerido. La aplicación ASP.NET iba a estar alojada en el servidor web accediendo a ella mediante el protocolo HTTP como he dicho en apartados anteriores.

Por otro lado el lenguaje utilizado fue Visual Basic. Aunque no había trabajado anteriormente con VB ni C#, elegí VB por su mayor sencillez en cuanto a su curva de aprendizaje, su buena adaptación con las plataformas de los sistemas Windows, sus librerías y su facilidad de encontrar documentación útil para la realización del proyecto.

Las consultas SQL a la base de datos iban a tener un papel muy importante, ya que gracias a ellas era posible esa comunicación. No tenía mucha experiencia real en base de datos más allá de lo aprendido en clase, pero no fue especialmente difícil adaptarme a su uso ya que se trataban de consultas básicas.

- **Consulta de carro y pieza**

Este tipo de consulta consiste en una consulta SQL que devuelve, en el caso de los carros, la lista de piezas que contiene la base de datos filtrando por: piezas repasadas y sin repasar, y según el carro introducido en el cuadro de texto donde se introduce el mismo. Del mismo modo, en el caso de las piezas, la consulta SQL devuelve la pieza introducida en el cuadro de texto correspondiente. Al final el usuario podrá seleccionar la pieza que quiera repasar de cualquier manera, independientemente de que disponga de carro o no.

- **Consulta de incidencias**

Este tipo de consulta es un poco diferente a la anterior ya que en este caso se muestran todos los tipos de incidencias que se pueden dar durante el repasado de una pieza. En caso de aparecer una incidencia y querer comenzar la gestión de la misma, se deberá buscar en una lista desplegable y continuar con el método de inserción de incidencias.

- **Consulta albarán**

En cuanto al albarán se ejecutan varias consultas a la vez, ya que se deben mostrar más datos para posteriormente ser imprimidos, en función del carro seleccionado. El albarán consta de varios datos como: número de carro, piezas que los constituyen, cantidad de metros totales de tela, kilos totales, etc. Todos estos datos vienen dados por diferentes consultas SQL que, también vendrán definidas por varios filtros aplicados tales como: mostrar únicamente piezas repasadas, carro introducido en el cuadro de texto, etc. Además, para el cálculo de longitud de tela y pesaje total incluirá una operación matemática realizada en tiempo de ejecución. El usuario de almacenado podrá ver que todos los datos son correctos y posteriormente proceder con el albaranado del carro para su posterior envío.

2.10 RESTRICCIONES GENERALES

En cuanto a las restricciones, la más notable seria que la modificación estructural de la base de datos no sería posible por los usuarios. Estas modificaciones únicamente serían posibles por el administrador de la base de datos aunque, sí que podrían modificar los datos de algunas piezas como: cambiar una pieza a otro carro distinto o eliminar incidencias de una pieza en caso de equivocación.

Por otro lado, no será posible ninguna modificación estructural en el diseño de la web por parte de los usuarios, únicamente por parte del programador de la web o el informático que disponga del código. Aunque existe una autentificación de usuario, no existe una diferenciación entre ellos ya que la aplicación simplemente se centra en el repasado.

Ya que no se dará una gran afluencia de usuarios utilizando al mismo tiempo la aplicación, el problema de saturación del servidor no se dará, o por lo menos no debería, aun siendo uno de los problemas más comunes en la mayoría de sitios web. Aunque, en contraste, en caso de que fuera así por ampliación del personal que pudiera utilizarla, podría solucionarse aumentando la potencia del servidor con nuevo hardware; o si el problema se hallara en la velocidad de la que dispone el servidor también habría que tener en cuenta un posible cuello de botella de la red.

Por último, la modificación del software de visualización final será llevada a cabo únicamente por el informático o probablemente por el jefe, ya que éste será el usuario final que verá e interpretará esos datos.

2.11 CARÁCTER INNOVADOR

La innovación afecta a todas las áreas de la empresa: diseño y desarrollo, marketing, recursos humanos o la propia gestión. En cuanto a lo que este proyecto se refiere, la innovación está ligada en un contexto tecnológico. Sin embargo, gracias a la optimización de recursos y tiempo durante el repasado, también podría basarse en una mayor cuota de mercado posible que podría provocar ventajas competitivas en el sector. La necesidad de adecuarse mejor y más rápido al avance tecnológico es necesario para poder alcanzar un nivel de competitividad. En concreto, en la sección de repasado de la empresa, era necesario este cambio ya que, por problemas de tiempo y personal, aún se seguía utilizando el papel como método de introducción de algunos datos. Aunque sí que posteriormente se introducía en la base de datos de la empresa, no era del todo óptimo ya que se tenía que realizar el trabajo por duplicado.

En cuanto a la tecnología innovadora utilizada, sobre todo se ha centrado en el uso de intercambio de datos entre distintos tipos de software mediante el uso de servicios web. Dicha tecnología utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones software desarrolladas en lenguajes de programación distintos, y ejecutadas sobre cualquier plataforma. Esta tecnología era la adecuada para comunicar el software ERP Navision con la aplicación web, de ese modo únicamente con una web fácil de utilizar sería posible eliminar numerosas sesiones de Navision en cada puesto de trabajo.

2.12 RETOS TECNOLÓGICOS

Por otro lado, se disponía de cierto hardware que no se estaba utilizando (PLC del contametros), así que gracias a estos servicios web se pudieron comunicar estos controladores lógicos con las máquinas y de ese modo pasó a ser un punto importante dentro del proyecto.

Más allá de la aplicación en sí, hay una capa de tecnología que ni el usuario ni el consumidor final ven y que facilita el acceso a la información y mejora la experiencia del usuario.

También, en cierto modo, este tipo de tecnología de gestión podría permitir a las empresas optimizar sus estrategias de marketing y gestionar el tipo de estrategias a seguir en un futuro en caso de querer crecer dentro del sector. Este avance mejoraría los procesos e incrementaría la eficiencia, tanto en el aspecto tecnológico como en la relación de confianza entre los usuarios. Por último, gracias a estas mejoras podría posibilitar el encuentro de nuevas oportunidades de negocio.

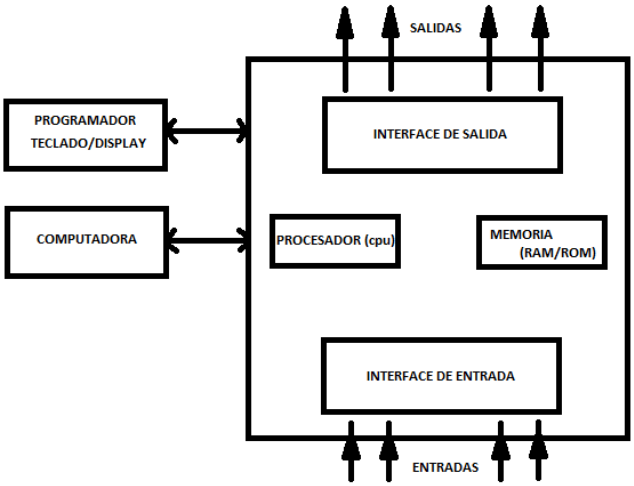
3 DESCRIPCIÓN DE LAS HERRAMIENTAS USADAS EN EL DESARROLLO

Durante la realización del proyecto fueron necesarios una serie de materiales y herramientas. En mi caso, estas herramientas se centraron básicamente en el uso de las máquinas (portátil donde se iba desarrollando el proyecto, máquina virtual donde se implantaba la web…) y las diferentes herramientas de programación (lenguajes de programación, entornos de desarrollo, etc.) También fue necesario el uso de determinado hardware de la empresa y un conocimiento básico del funcionamiento del mismo (PLC, máquina de repasado, etc.)

3.1 CONTROLADOR LÓGICO PROGRAMABLE

Actualmente casi todo tiende a automatizarse para facilitar las vidas de la gente. Para ello se suelen emplear computadores personales (PC), circuitos lógicos (CI), temporizadores, contadores, controladores lógicos programables (PLC), etc. El análisis y diseño de circuitos secuenciales está relacionado con el control secuencial o lógico. En estos sistemas de control las entradas y las salidas son de tipo binario. Las entradas suelen ser: pulsadores, interruptores, detectores, etc. Y las salidas pueden ser: dispositivos de señalización, alarmas, monitores, etc. En el caso de este PLC se trata de un detector de movimiento programado para contar los metros recorridos y de ese modo mostrarlo en una pantalla.

- **Controlador Lógico Programable (PLC):** es un equipo electrónico programable diseñado para controlar en tiempo real y en ambiente industrial un proceso secuencial. Este dispositivo electrónico nos permite utilizar funciones específicas (lógicas, de temporización, de conteo y aritméticas) para implementar un circuito de control. Estos controladores lógicos están constituidos por un conjunto de tarjetas o circuitos impresos sobre los que están ubicados los componentes electrónicos.



Estructura Interna de un PLC (Fig. 1)

El PLC empleado para el repasado del tejido y así mismo el utilizado durante la realización del proyecto era un PLC Omron, cuya estructura externa es de la siguiente manera:

Estructura externa de un PLC Omron (Fig. 2)

- Fuente de alimentación: Se requiere de una fuente de voltaje para la operación de todos los componentes. Ésta, en el caso de una interrupción del suministro eléctrico, sirve para mantener la información en la memoria borrable de tipo RAM, como es la hora y la fecha, los registros de contadores, etc. En el caso de los PLC modulares es necesario adicionar una batería externa.
- Módulo CPU: El corazón de un PLC es la Unidad Lógica Aritmética basada en un microprocesador, ya que ejecuta las instrucciones programadas en memoria.
- Unidades de memoria: la memoria almacena el código de mensajes o instrucciones que ejecuta la Unidad Lógica. La memoria está dividida en PROM, RAM y ROM.
    - RAM: Memoria de acceso aleatorio, volátil que puede ser leída y escrita según sea la aplicación. Cualquier posición de memoria puede ser accedida en cualquier momento.
    - ROM: Memoria de solo lectura, no volátil que puede ser leída pero no escrita. Es utilizada para almacenar programas y datos necesarios para la operación de un sistema.
    - PROM: Almacena los programas permanentes que coordinan y administrar los recursos del equipo.

- Módulos de entradas digitales: Proporciona el aislamiento eléctrico necesario y realiza el acondicionamiento de las señales eléctricas del voltaje. Las señales se adecúan a los niveles lógicos de voltaje de la Unidad Lógica.
- Módulos de salidas digitales: Acepta las señales lógicas provenientes de la Unidad Lógica, en los rangos de voltaje que le son propios y proporciona el aislamiento eléctrico a los dispositivos que se conectan con el exterior.

3.2 HERRAMIENTAS

En cuanto a la herramienta web, se trata de una aplicación software codificada en un lenguaje soportado por el navegador web utilizado, y que utiliza servicios web con un lenguaje de programación de ASP.NET y una integración de la base de datos. Dicha base de datos se encuentra en una máquina virtual donde se aloja el servidor de Navision, mientras que la web se aloja en un servidor web IIS en otra máquina virtual.

3.2.1 Lenguajes y frameworks

Los lenguajes utilizados, como ya he dicho en apartados anteriores, han sido seleccionados por su adaptación e integración en cuanto a las tecnologías impuestas en un principio… entre ellos, un lenguaje el cual tuve la oportunidad de familiarizarme brevemente, con ayuda de mi tutor dentro de la empresa, fue el propio de Navision. Esto fue gracias a que la empresa trabajaba casi por completo bajo este ERP:

- **C/AL (Navision):** C/AL es un lenguaje de programación específico de base de datos, y se utiliza principalmente para recuperar, insertar y modificar registros en una base de datos de Dynamics NAV. C/AL se asemeja al lenguaje Pascal ya que se basa en él. Por otro lado, recientemente se incorporó la posibilidad de que determinados tipos de objetos puedan ser publicados como servicios webs, con lo que facilitaba la integración desde otras aplicaciones; en nuestro caso nuestra aplicación web. Así mismo como el consumo y acceso controlado de datos, todos ellos mediante estándares.

Por otro lado, los lenguajes utilizados durante la programación de la aplicación fueron los siguientes:

- **HTML:** este lenguaje es un lenguaje de marcado utilizado en las páginas web para definir su contenido y estructura (texto, imágenes, videos, etc.) Dicho lenguaje se ha estandarizado completamente y es el que todos los navegadores han adoptado para la visualización de páginas web.
- **CSS:** CSS es un lenguaje que sirve para dotar de presentación y aspecto, de “estilo”, a páginas web (documentos HTML). CSS no es un lenguaje de programación en sí mismo, pero se podría decir que es un lenguaje que suele aparecer relacionado o próximo a un lenguaje de programación o que suele colaborar con un lenguaje de programación. Además, para el diseño de la aplicación web se utilizaron diferentes plantillas “Bootstrap” las cuales daban ese aspecto atractivo y a su vez proporcionaba esa posibilidad de adaptación a cualquier dispositivo.
- **VB.NET:** VB es un lenguaje de programación orientado a objetos implementado sobre el framework .NET que, además de facilitar el desarrollo de aplicaciones, mantiene una eficacia en el desarrollo de las mismas. En el caso de esta aplicación se utilizó el entorno de desarrollo de Microsoft Visual Studio 2015 para desarrollarla.

En cuanto al framework destinado para aplicaciones web desarrollado por Microsoft y utilizado durante el desarrollo del proyecto fue el ASP.NET, que además de ser usado sobre todo para el diseño de sitios webs dinámicos, aplicaciones web y servicios web XML, permite escribir código ASP.NET utilizando cualquier lenguaje admitido por el framework .NET. Las páginas de ASP.NET son conocidas oficialmente como formularios web y son el principal medio para el desarrollo de esta aplicación web.

Otro lenguaje que tuve la oportunidad de familiarizarme ligeramente fue el lenguaje de acceso a bases de datos relacionales SQL, el cual permite especificar diversos tipos de operaciones con ellas. El uso de una serie de sentencias (SELECT, INSERT, DELETE…) y distintas clausulas, permitía efectuar consultas con el fin de recuperar información de las distintas piezas a repasar en las tablas de la base de datos en cuestión, o hacer cambios en ella.

3.2.2 Entornos de desarrollo (IDE)

En cuanto a los entornos de desarrollo, los cuales se componen por ser programas informáticos y contener herramientas de programación, suelen centrarse en un lenguaje de programación en concreto o bien, pueden utilizarse para varios. En mi caso, el IDE utilizado para el desarrollo de la programación de la aplicación fue el Microsoft Visual Studio 2015, el cual se puede utilizar para la creación tanto de aplicaciones para dispositivos o aplicaciones de escritorio, ya sea para la web o en la nube. Además facilita el código para IOS, Android y Windows. Gracias a ciertas herramientas, el IDE facilita la implementación de aplicaciones web y el uso de lenguajes como HTML5, CSS3, JavaScript o ASP.NET así como una compatibilidad avanzada con ciertos marcos webs utilizados como “Bootstrap”.

Aunque el uso de este IDE junto al del framework ASP.NET está centrado en la compilación de servicios y aplicaciones web en la nube, en el caso de esta aplicación web se utiliza un servidor web local dentro de la empresa. También, la biblioteca de clases de .NET maneja la mayoría de las operaciones básicas que se encuentran involucradas en el desarrollo de aplicaciones: interacción entre dispositivos periféricos, manejo de datos (ADO.NET), cifrado de datos, administración de componentes web que corren tanto en el servidor como en el cliente (ASP.NET), herramientas de seguridad e integración con la seguridad del SO, etc.

Por otro lado, aunque no se trata de un entorno de desarrollo sino más bien una herramienta, se utilizó el SQL Manager for SQL Server con el objetivo de administrar las tablas de la base de datos y así utilizar sentencias durante el desarrollo del código. Esta herramienta es bastante útil ya que trabaja con casi cualquier versión de SQL Server y soporta las características más nuevas incluyendo sentencias, tablas, bases de datos entre otras. Además, permite desarrollar y administrar bases de datos en una interfaz de usuario.

3.2.3 Software

El software utilizado, el cual no se profundizó en gran medida ya que únicamente se necesitarían ciertas funciones puntuales, fue el siguiente:

- **Microsoft Dynamics NAV:** Navision es un software ERP que forma parte de la familia de productos de Microsoft, el cual cubre algunas áreas funcionales dentro de una organización como: la gestión financiera (contabilidad, presupuestos, informes financieros, registros entre empresas…), ventas y marketing (clientes, pedidos, precios…), compras (proveedores, planificación de compras, costes…), almacén (inventario, enviar y recibir productos, gestión de distintos almacenes…), entre otras. Y que además, es posible añadir nuevas funcionalidades ya que el código fuente es accesible según la licencia obtenida.

  Por otra parte, el lenguaje de programación que utiliza se puede encontrar en cualquier objeto de la aplicación como: tablas, formularios, informes, menús, etc., cuyo código viene organizado por triggers o desencadenantes (de documentación, eventos o funciones). Principalmente, tanto la presentación de los datos, su adquisición o almacenamiento se realizan a través de formularios e informes organizándolos utilizando su sistema gestor de base de datos.
- **Qlikview:** Qlikview es una plataforma software que, gracias a una conexión SQL a una base de datos, es posible realizar un estudio de éstos utilizando diversas funcionalidades como: tablas, listas, gráficas, etc. Y que además, al proporcionar una conexión directa con la base de datos y combinando consultas y filtros, concede una posibilidad de contemplar en tiempo real la información y de ese modo tener una visión clara y rápida de la misma.

En el caso de los datos recogidos durante el repasado era interesante utilizar distintas "hojas", las cuales se utilizan para filtrar la información que se quiere mostrar, como: año en el que quieres buscar la información, mes, semana, día, máquina de repasado, operario, producto, piezas repasadas, etc.

3.2.4 Arquitectura Cliente – Servidor

Este tipo de arquitectura utilizado, es un modelo que distribuye las tareas entre servidores (recursos o servicios) y clientes (peticiones). El cliente realiza ciertas peticiones al servidor el cual éste da respuesta. En el caso de esta aplicación, se trata de un servidor web ejecutado en una máquina virtual que ejecuta todas las gestiones y que restringe el uso únicamente a usuarios registrados.

Por lo general, el que remite una solicitud se le llama cliente y es el que inicia las peticiones al servidor esperando sus respuestas mediante una interfaz gráfica de usuario. El receptor de la solicitud se conoce como servidor y por lo tanto es el que espera a que lleguen esas peticiones para posteriormente procesarlas y enviar la respuesta al cliente. Además, acepta un número determinado de clientes en función del hardware utilizado y el tipo de red.

Como en la mayoría de los sitios web se requiere de una arquitectura cliente – servidor ya que el servidor sirve las páginas web al navegador (cliente). En el caso de esta aplicación web programada utilizando formularios web contenidos en archivos con una extensión aspx, el navegador web solicita una información, el servidor la recopila en la base de datos, la articula en la página web y la envía al navegador de nuevo.

En cuanto a las ventajas que tiene el uso de este tipo de arquitectura, podrían ser las siguientes:

- Control de accesos e integridad de datos para detectar accesos no identificados.
- Escalabilidad, en cuanto al aumento de capacidad tanto de servidores como de clientes, por separado y en cualquier momento.
- Fácil mantenimiento al estar las funciones distribuidas entre varios ordenadores independientes. Esta independencia se conoce como encapsulación.

Aunque también existen una serie de desventajas como por ejemplo:

- La congestión de tráfico enviando peticiones simultáneas al servidor podría ralentizarlo o incluso colapsarlo. Aunque en este caso no se daría esa afluencia de clientes al mismo tiempo.
- Si el servidor cae, las peticiones de los clientes no pueden ser satisfechas por lo que el trabajo se vería afectado. En cambio, para las redes de pares no se da este problema, ya que los recursos están generalmente distribuidos por varios nodos en la red.
- El coste puede ser elevado en algunas ocasiones ya que se necesita hardware y software específico por parte del servidor para satisfacer el trabajo.
- El cliente no dispone de ciertos recursos como en el servidor por lo que, por ejemplo en el caso de imprimir un albarán, es necesaria una ventana previa de impresión en el navegador que lo permita.

3.3 RESTRICCIONES TECNOLÓGICAS

Ya que se buscaba una manera de informatizar la información obtenida durante el repasado, se iba a necesitar una nueva manera de introducir los datos. Se disponía de máquinas, las cuales eran utilizadas con el objetivo de que las piezas de tela fueran recorridas en busca de imperfecciones. Esto, en parte, provocaba la necesidad de un tiempo de familiarización con este tipo de tecnología utilizada, ya que se trataba de instrumentos que únicamente se encontrarían dentro de la empresa.

En cuanto al cálculo de metraje, ya se disponía de unos PLC que se iban a utilizar para contar los metros de tela, así como la cantidad de defectos en franjas específicas de tela, etc. Dichos PLC necesitaban ser conectados al PC de cada máquina de repasado para que éste enviara, por medio de pulsos eléctricos, una

serie de bits de datos que, posteriormente iban a ser interpretados por medio de una aplicación. Previamente se propuso instalar, en lugar del PLC, algún tipo de procesador el cual permitiera controlar dicho metraje. Esto podría haber hecho reducir el elevado coste de comprar nuevos PLC en las máquinas que no disponían de ellos, pero finalmente se decidió dar uso de esta tecnología.

Por otro lado, el sistema operativo impuesto, y por tanto, que se iba a utilizar durante todo el desarrollo de la aplicación iba a ser Microsoft Windows, tanto en los ordenadores de las máquinas donde se iba a destinar la aplicación como durante la programación de la misma. Por ello se intentó utilizar toda la tecnología propuesta de Microsoft como: Visual Studio, el framework de desarrollo ASP.NET, servidor IIS, etc.

4 ANÁLISIS Y DISEÑO DE LA APLICACIÓN

4.1 ANÁLISIS DE REQUERIMIENTOS

Cada sistema basado en algún tipo de software tiene un propósito el cual debe cumplir, normalmente expresado con algo que el sistema, en este caso la aplicación, debe llevar a cabo. El requerimiento o característica principal del sistema era la informatización de una serie de datos recogidos con los que posteriormente se someterían a estudio. Durante este análisis se debería tener en cuenta como descripción especifica de la implementación el uso de una base de datos previamente proporcionada con la que se iba a trabajar durante todo el proyecto.

Al llegar a la conclusión de que se iba a proceder a la creación de una aplicación web, se tendría que describir la funcionalidad que se esperaba que el sistema pudiera proveer. En este caso, la aplicación iba a proveer una serie de requisitos hacia los usuarios como: el uso de una autentificación personalizada tanto por parte del usuario y la máquina utilizada; y una serie de requisitos del sistema como: capacidad de interacción con la base de datos para la búsqueda tanto de piezas como de carros, envío de tiempos de repasado, cantidad de incidencias, metros repasados, metros donde se encuentran las incidencias, posibilidad de saneamiento de las piezas, posibilidad de impresión de incidencias detectadas totales, etc.

Aunque por otro lado, también había que tener en cuenta una serie de requisitos no funcionales, que en caso de incumplimiento podrían inutilizar el correcto funcionamiento de la aplicación, así como: la fiabilidad del correcto uso de la aplicación por parte de los usuarios, la velocidad de respuesta y rapidez de ejecución, la portabilidad y usabilidad, etc.

Estos requisitos fueron surgiendo sobre todo durante el tiempo de prueba de la aplicación, ya que los usuarios la utilizaban, en algunos casos, de forma diferente a la que se espera durante la programación de la misma:

- Refresco de la página cada cierto tiempo para que los metros recorridos aparecieran en pantalla de forma real.
- Inutilizar el botón de borrado del teclado para evitar salirse de la aplicación sin pulsar el botón de finalizar repasado o pausar repasado.
- Diseño establecido acorde a las resoluciones establecidas para que los usuarios se adaptaran mejor a la usabilidad de la aplicación.

Muchas veces estos requisitos eran más importantes que los propios requisitos de funcionabilidad establecidos por el cliente.

Al final el usuario era el que iba a ver reflejado si el sistema estaba correctamente adaptado a sus necesidades, y por ello era importante que el dominio de la aplicación fuera satisfactorio: utilizando un lenguaje técnico conocido por los usuarios así como una disposición de las opciones de la aplicación fáciles de comprender.

4.2 DIAGRAMA DE CLASES

Para entender mejor las funciones que se realizan con esta aplicación se ha realizado este diagrama UML con el software ArgoUML. Además, se procederá a explicar brevemente la situación y las características de la aplicación en base al diagrama.

El principal objetivo de la aplicación es intentar agrupar la mayoría posible de datos para su posterior estudio. El usuario entra en la aplicación autentificándose y posteriormente procede a la introducción de piezas para su repasado. Las características serían las siguientes:

- De cada usuario se almacena:
  - IDUsuario
  - Nombre
  - Apellidos
  - Código
- Un usuario puede repasar en varias máquinas a la vez, pero cada máquina solo puede desempeñar la función de repasado únicamente por un usuario.
- Durante el repasado es importante registrar:
  - Fecha y hora de inicio y fin de repasado
  - Número de pieza
  - Producto usado en cada pieza
  - IDUsuario
  - IDMáquina
  - Metros realizados
  - Número de carro
- Durante la introducción de incidencias es importante registrar:
  - Número de pieza
  - Producto usado en cada pieza
  - IDUsuario
  - IDMáquina
  - Metro de inicio y fin de incidencia
  - IDIncidencia
  - Si ha sido reparada la incidencia o no
  - Cantidad de incidencias del mismo tipo
- Los usuarios además podrán eliminar incidencias en caso de equivocación.
- Los usuarios podrán hacer un cambio de pieza a un carro distinto en caso de desecharla por completo o incluso crear un carro diferente.

El Usuario va a ser la clase principal que interactuará con la aplicación a través del Stock, el cual, es una clase que contiene los elementos, tanto piezas como carros, de los que se abastece la base de datos a la que accede la Aplicación. Por ello, todas las consultas realizadas por el Usuario se harán a esa clase. El Usuario será el que procederá a las búsquedas del Stock para proceder a comenzar el repasado de la pieza.

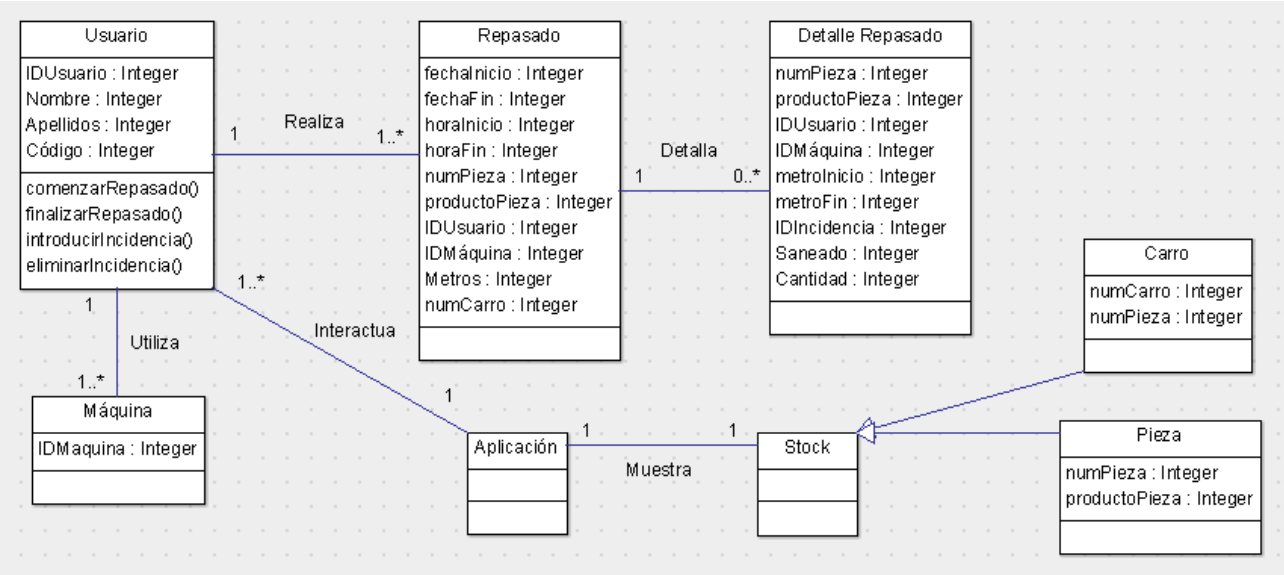


Diagrama de clases (Fig. 3)

4.3 DIAGRAMA DE CASOS DE USO

Para que se entiendan más intuitivamente las acciones que se pueden realizar, o al menos algunas de las más importantes, se adjunta la siguiente figura:

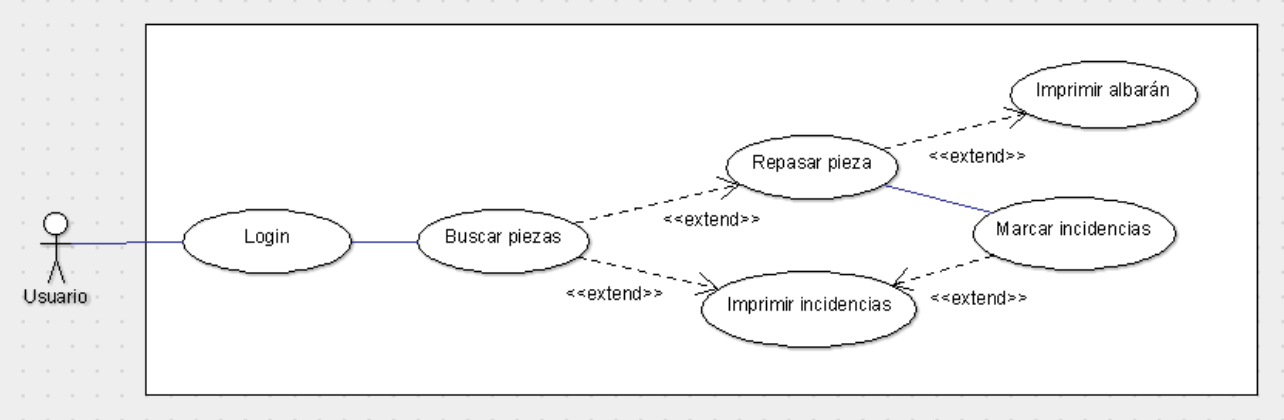


Diagrama de casos de uso (Fig. 4)

4.4 DIAGRAMA DE SECUENCIA

A continuación, mediante un diagrama de secuencia, se reflejará el proceso que se sigue para realizar la acción de repasado a nivel de ejecución.

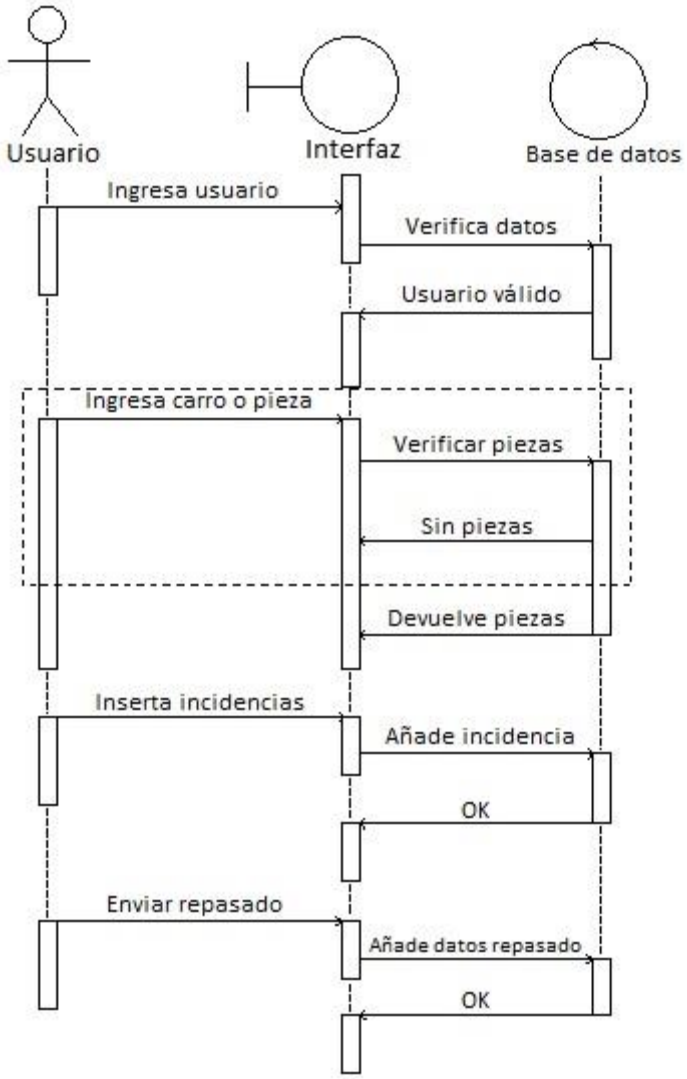


Diagrama de secuencia (Fig.5)

5 DISEÑO

El diseño de la aplicación está basado en una programación por capas, en el que el objetivo principal es separar la parte de presentación con la lógica. Podemos diferenciar varias capas:

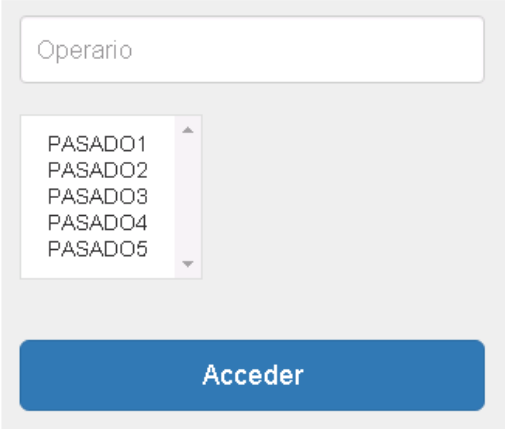
- Capa de presentación: interfaz gráfica que ve el usuario (fácil de usar) que se comunica con la capa lógica.
- Capa lógica: se reciben las peticiones del usuario y se envían las respuestas según las reglas establecidas. Además, se comunica con la capa de datos donde solicita al gestor de la base de datos almacenar o recuperar datos y con la capa de presentación donde recibe las solicitudes y presenta los resultados.
- Capa de datos: se acceden a los datos por medio del gestor de bases de datos con el objetivo de almacenarlos o recuperarlos.

En el caso de esta aplicación, la interacción viene a través de formularios, es decir peticiones que se hacen al servidor para que sean procesadas. Esto hace que la información solicitada aparezca en el navegador y permita interactuar con ella. Como ya he dicho, el usuario será el que utiliza dicha interfaz y con ella se comunique con los datos.

Por otra parte, la parte de los datos se gestiona mediante la base de datos y el sistema gestor para acceder y almacenar la información de forma segura.

5.1 INTERFAZ

La interfaz de usuario, en cuanto al acceso identificado a la aplicación, está diseñada de forma intuitiva y fácil de utilizar. Está compuesta por una caja de texto donde se debe introducir el identificador del operario (proporcionado previamente), una lista de las máquinas de repasado disponibles (vienen preseleccionadas según la máquina desde donde se esté repasando en función de la dirección IP del ordenador de la misma) y un botón para acceder a la aplicación.



Panel de acceso de la aplicación (Fig. 6)

Por otro lado, la imagen general donde se realiza el repasado de la pieza una vez seleccionada es la siguiente:

Pantalla mostrada durante el repasado (Fig. 7)

Todas las páginas de la aplicación vienen dadas por dos etiquetas que se distinguen fácilmente:

- Cabecera: se muestra un texto meramente informativo y, en el caso de estar repasando, la máquina y operario seleccionado.
- Cuerpo: se muestra toda la información y, por medio de formularios, se recogen los datos que el usuario introduce para posteriormente ser gestionados.

5.1.1 Navegabilidad

La estructura se detalla mediante una serie de páginas, las cuales se utilizan para acceder a sus distintos puntos. Gracias a un diagrama fácilmente representado se explicará la navegación de la aplicación por parte de los usuarios. Ya que para poder acceder a la aplicación se necesita estar registrado previamente y que todos los usuarios tienen los mismos privilegios, no es necesaria una distinción entre ellos al no existir una diferenciación durante la navegación:

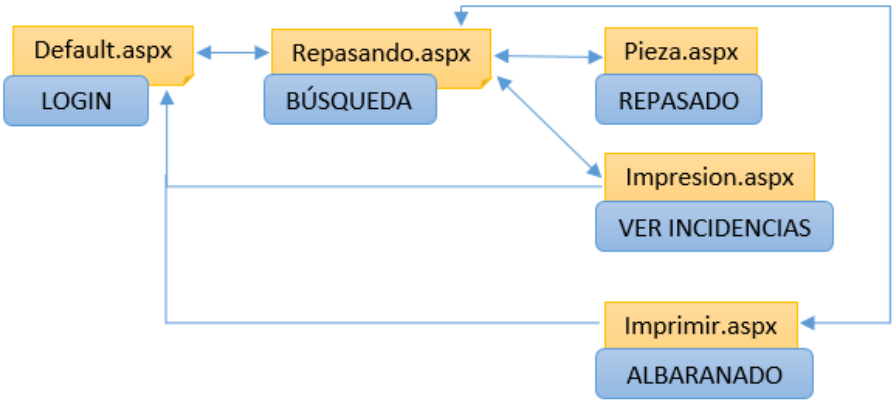


Diagrama de navegabilidad de la aplicación (Fig. 8)

5.2 Nivel lógico

En este nivel se ven afectadas algunas funciones, tales como: las peticiones que se efectúan a la base de datos, ya sea durante el acceso de la aplicación o la búsqueda de piezas; las operaciones que calcula la aplicación en tiempo real antes y después de comunicarse con la base de datos (tiempo total repasado, metros repasados, kilos totales de un carro…); la comprobación de acceso a la aplicación por parte de los usuarios; las transformaciones que se efectúan en la base de datos, ya sea por un cambio de estado en una pieza o incluso en cuanto a la modificación de las incidencias, etc.

6 Implementación de la solución

En cuanto a cómo se iba a proceder a la solución del proyecto, no se establecieron unos plazos previstos para cada punto del desarrollo, si no que más bien fueron surgiendo una serie de necesidades conforme se iba avanzando, teniendo más o menos claro el objetivo final. Aun así, como ya he dicho en otros apartados, sí que se tuvieron en cuenta algunos objetivos menores para tener una mayor constancia y así que la motivación no decayera en caso de atasco.

6.1 Desarrollo

Para una mayor claridad, se procederá a clasificar estos pequeños objetivos por plazos o puntos intentando profundizar en cada uno de ellos:

6.1.1 Comunicación con el contametros

Para empezar, la zona de repasado de la empresa disponía de una máquina de extendido de la tela, la cual utilizaba una máquina que expandía los rollos de tela en carros para su posterior repasado. Además contaba de cinco máquinas de repasado, siendo una de ellas de doble repasado (aunque se utilizaba más bien para desplegar la tela rollada en carros si era necesario). Cada máquina de repasado la utilizaba un operario y disponía de un PC (“ThinClient”) que, anteriormente, se utilizaba para controlar las incidencias en tiempo real por medio del ERP Navision. También cada máquina disponía de un PLC que mostraba el metraje durante el repasado y, aunque no estaba siendo utilizado en esos momentos, disponía de todo el cableado y de una conexión mediante un conversor a la red.

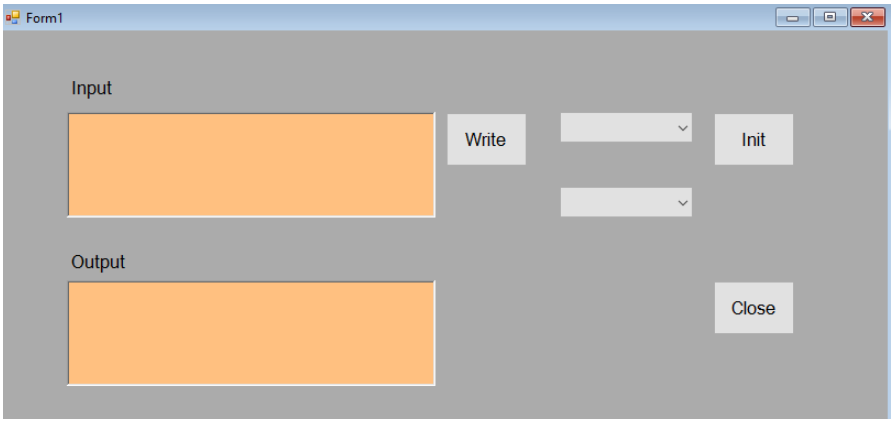
Una vez conocida la zona donde se efectuaba el repasado y cómo funcionaba, (el rollo de tela se enganchaba a la máquina de repasado y conforme se iba repasando se iba extendiendo en un carro, el cual contaba de varios rollos diferentes pero siempre de un mismo tipo de tela), había que entender el funcionamiento del PLC ya que iba a ser uno de los principales pilares de información, puesto que daría a conocer: la cantidad de tela repasada, el metro donde se recogieron las incidencias, si una tipo de tela habría

que desecharla por su cantidad de incidencias en ciertos metros, etc. La familiarización con el funcionamiento del contametros fue un tanto complicada al principio, ya que era la primera vez que tenía la oportunidad de enfrentarme a un PLC Omron de este estilo. Previamente, aunque sí que existía una pequeña aplicación de usuario que, mediante un conversor de COM ("Component Object Model") a red estos PLC podían comunicarse de manera local con el administrador, no estaba siendo de gran utilidad. Por ello, se decidió aprovecharla y por medio de servicios web, cada contametros se pudiera comunicar con cualquier usuario dentro de la red que quisiera utilizar la aplicación. Esto era especialmente útil ya que Navision disponía de una utilidad con estos servicios y además todo venía bajo Windows lo que lo hacía más sencillo en mi caso.

Para poder comunicar el PLC con el PC y realizar las primeras pruebas, se iba a necesitar un conversor de PLC a USB que, con la ayuda de un cable y la instalación de los drivers necesarios, iba a permitir la conexión. En un principio se hicieron las pruebas en el portátil para mayor comodidad y no molestar demasiado a los operarios que estaban repasando el tejido en ese momento. Con la ayuda del hombre que se encargó de programar los PLC, fui capaz de aprender que dichos PLC contestaban líneas de datos en función de la línea previamente enviada a éste, es decir, según un código enviado que fuera capaz de entender, respondería uno nuevo el cual habría que traducir posteriormente. Además, según el tipo de PLC y como estaba configurado habían tanto bits desechables, como un sentido diferente de rotación, entre otras cosas.

A continuación, sabiendo por encima la constitución del PLC, había que configurar el puerto serie con el PC al que se había conectado para que utilizara los valores programados y de ese modo no hubiera problemas de concurrencia (bits por segundo, bits de datos, bits de paridad y bits de parada). Además, disponía de varias líneas de transmisión de datos diferentes según el caso: poner a cero el contador, cambiar el sentido en el que gira el contametros o leer los pulsos del mismo. Por último, era necesario transformar el número de pulsos en metros mediante una sencilla operación usando una pequeña aplicación de escritorio de prueba programada también con Visual Basic. En dicha aplicación era necesario seleccionar el puerto COM conectado y, mediante un cuadro de texto, se le enviaría la línea de datos que el PLC traduciría y así devolvería una contestación. A la línea de datos devuelta se le debía realizar la operación para traducir los metros que tenía guardados en ese momento.

En cuanto a la programación de dicha aplicación, disponía de un botón de escritura el cual enviaba la línea de datos escrita en el cuadro de diálogo y que solo estaría activo una vez se detectara un puerto serie conectado e inicializado. Para poder seleccionar un puerto serie e inicializarlo, era necesario utilizar la herramienta de "SerialPort" del Visual Studio; el cual, utilizando los constructores necesarios, inicializando la instancia de la clase y utilizando el puerto que haya conectado, era posible manejar sus propiedades (velocidad en baudios, búfer de envío, nombre del puerto, etc.) En un principio solo se probaba la transmisión de lectura del contametros ya que lo único que importaba en ese momento era que el PLC respondiera una línea de datos sin importar cuál, ya que eso quería decir que la comunicación se había realizado con éxito y se había configurado correctamente. Más tarde sí que era necesario probar las demás líneas de transmisión.



Ventana de la aplicación de prueba (Fig. 9)

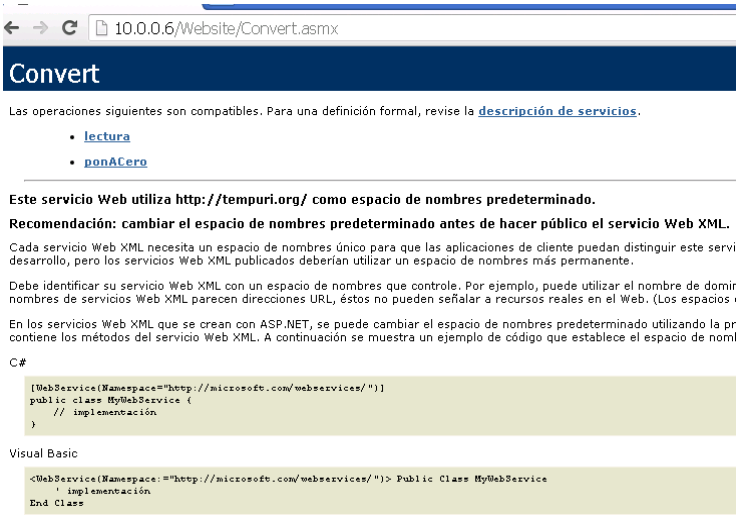
Una vez comprobado el funcionamiento, había que buscar formas de comunicar el PLC en la red local, pero esta vez conectando el puerto serie al "ThinClient" de la máquina de repasado. Para ello, era necesaria una nueva aplicación web que, mediante servicios web y bajo el Visual Studio de nuevo, iba a poder enviar la cadena de datos al PLC y que éste devolviera (en caso de lectura) el metraje. Según cierta información encontrada por internet, era posible realizarlo mediante determinadas librerías, pero resultó más intuitivo crear una pequeña aplicación propia que pudiera poner a cero el contador y leer dicho metraje.

Esta aplicación era el servicio web que iba a estar instalado en cada "ThinClient" y que la aplicación final llamaría para poder recibir los datos de los contametros. La programación tenía algunas similitudes con la de prueba en cuanto al código ya que utilizaría las mismas propiedades (velocidad, número de bytes de lectura, búfer), pero se le añadiría algún método nuevo (apertura y cierre del puerto, especificación de la cadena enviada por el puerto serie). Después de varía pruebas, resultó que era necesario cerrar y abrir el puerto COM cada vez que ejecutara una operación ya que si no, podían surgir problemas si la acción no se había realizado completamente; y además, en el caso de estos PLC, era necesario enviar la cadena de código concreta y un carácter de retorno de carro (vbCr) al final. Por último, según la acción, se debían despreciar una serie de bits y con los restantes hacer la operación de cálculo de metraje en el caso de lectura, cerrando finalmente, el puerto COM de nuevo. Todos estos datos a configurar se verán mejor en un anexo 8.1 donde se encuentra la hoja de configuración del PLC.

Para continuar con la preparación de cada máquina, simplemente quedaría activar las funcionalidades de "Internet Information Services" (IIS) en cada PC y de ese modo dejar instalado el servidor web. En la carpeta "inetpub" sería donde habría que subir la aplicación del servicio web y así ésta vendría predeterminada por el puerto 80. Esto haría que posteriormente desde la aplicación final de repasado se pudiera referenciar sin problemas este servicio y de ese modo podría leer el metraje del contametros en caso de lectura.

Por otra parte, no todos los "ThinClients" venían con Windows instalado por lo que en alguno de ellos era necesaria su previa instalación y configuración acorde a las necesidades de cada operario (cierto tamaño de letra, quitar opciones de administrador en la cuenta de usuario para solo poder acceder a la aplicación web…), además de un comando en las opciones de red que haría su salida por un proxy específico que haría mucho más rápida la navegación.

Antes de hacer la prueba final desde la aplicación, ya que no estaba programada aún, se hicieron las pruebas localmente desde los propios PC de las máquinas. Esto era posible entrando al localhost y así, según la función deseada, enviaría la información al contametros. En caso de lectura abriría una nueva ventana con el metraje del contametros, y en caso de puesta a cero abriría una con un booleano true en caso de haberse puesto a cero correctamente, o false en caso de no haber funcionado.



Ventana de pruebas para el servicio web (Fig. 10)

Para finalizar, al utilizar la aplicación de repasado, los "ThinClients" deberían estar encendidos para poder leer por el puerto serie. Éstos siempre estarían activos durante el repasado, y además no habría problema en cuanto a los recursos utilizados de la aplicación ni en fallos de lectura por no reconocer el puerto. Una vez hecho, ya sería posible leer los contametros desde cualquier PC dentro de la red indicando la máquina previamente.

6.1.2 Servidor web

Después de tener claro el funcionamiento de los PLC y la comunicación mediante la red gracias a los servicios web, era hora de empezar con el desarrollo de la propia aplicación que iba a utilizarse en las máquinas. Para ello se necesitaría un servidor donde alojar la web y que la administrara pudiendo acceder a ella desde dentro de la empresa.

El servidor proporcionado contaba con una versión de Windows Server que, además, se le iba a tener que activar la función de rol de servidor web IIS que incluía "Internet Information Services 7". Este rol no es más que una plataforma web unificada que integra IIS, ASP.NET y otras funcionalidades de Windows y que además permite compartir información con usuarios, en este caso en una red interna. Dicho servidor dispondría del direccionamiento IP del mismo (127.0.0.1 correspondiente al localhost), puerto por el que escucharía (por defecto el 80) y el documento predeterminado que lanzaría el servidor al conectarse.

Igualmente, se tendrían que administrar ciertas funciones del servidor como por ejemplo, dentro del grupo de aplicaciones – la versión de .NET Framework 4.0 como predeterminado, ya que el proyecto utilizaría dicha versión. El protocolo de transferencia de archivos que se utilizaría en el servidor era simplemente agregando el sitio web como una aplicación desde el administrador de IIS y de ese modo dando una ruta compartida que se pudiera acceder desde dentro de la red, editando ciertos permisos de acceso, y así habilitando la transferencia de los archivos requeridos. La carpeta donde introducir los archivos sería la de inetpub propia del IIS, al igual que en los "ThinClients".

6.1.3 Navision

En mi caso, la utilización de este sistema de planificación llamado Navision era una novedad y por lo tanto no tenía claro cómo hacerle frente. Gracias a mi tutor en la empresa fue posible una primera toma de contacto, aprendiendo discretamente su principal uso dentro del área del proyecto: inventario de los productos y gestión de los almacenes.

A continuación, se creó un "codeunit" (contenedor usado para el código propio de Navision usado en muchos objetos de aplicación) que iba a ser utilizado en varios informes y de ese modo eliminaría la necesidad de duplicar código y facilitaría su mantenimiento. En dicho "codeunit" llamado "WebService", se crearían las funciones que posteriormente se utilizarían para insertar o eliminar incidencias, enviar los datos de repasado (usuario, máquina, metros, tiempo…) y cambiar piezas entre carros dentro de la base de datos. Sin embargo, antes de comenzar con la programación en las piezas, se hicieron distintas pruebas únicamente insertando piezas en una de las tablas para comprobar que la programación funcionaba.

Una vez sabido que funcionaría, se comenzaron a programar las siguientes funciones:

- **Insertar incidencias:** sus atributos (pieza, producto, usuario, máquina, código de incidencia, cantidad, donde empieza y acaba, la fecha y hora en que se ha encontrado la incidencia utilizando el formato DATE del Navision, y si ha sido saneada la incidencia o no) se insertarían en una de las tablas. En este caso, en la programación de esta función, únicamente se inicializaría la tabla donde se insertarían los datos, se crearían las variables haciendo referencia a los atributos necesarios y finalmente se insertarían en la tabla.
- La función de **eliminar incidencias** era más sencilla ya que únicamente se necesitaría resetear la referencia a la tabla, seleccionar tres atributos (pieza, donde empieza, y la fecha y hora) y comenzar un bucle en el que cada vez que encontrase un registro de esa incidencia se eliminase.

Por otra parte, habrían dos funciones que enviarían los datos de repasado, una durante el comienzo del repasado y otra al finalizar el repasado:

- En el caso de **empezar el repasado**, ya que el número de la pieza estaría utilizándose en dos tablas diferentes, se insertaría en una de ellas y en la otra simplemente se modificaría. En cuanto a la programación, pasaba lo mismo que al insertar incidencias; es decir, se inicializaría la tabla, se crearían las variables necesarias haciendo referencia a sus atributos, y en el caso de la otra tabla se usaría la misma función que la de borrado de incidencias, pero en lugar de borrarla se modificaría. Además, como podían repasarse las piezas más de una vez, si el repasado comenzaba de nuevo, se agregarían ciertos datos (usuario en caso de ser diferente, y fecha y hora).
- En el caso de **finalizar el repasado** se insertarían los mismo atributos pero añadiendo: metros repasados, cambio de estado de la pieza a repasar, y la fecha fin de repasado.

Para finalizar, hizo falta una nueva función para el albaranado de los carros, el cual era un tanto más complejo ya que tenía referencias a varias tablas y por lo tanto necesitaría por ejemplo: cambiar de nuevo los estados de las piezas, el número de piezas dentro del carro, así como el usuario, la fecha y hora, los metros, el producto utilizado en las piezas, la empresa, el telar, etc. Sin embargo, la programación del propio albarán se realizaría más tarde durante el diseño de la web en el que aparecerían más datos.

No todas las funciones se crearon desde el principio ya que no todas eran necesarias. Por ello, no todo el código tiene un orden específico. Las funciones principales eran las de inicio y fin de repasado ya que eran las que enviarían los datos más útiles, así como las incidencias. A su vez que se iba programando, se irían referenciando dichas funciones en el proyecto para comprobar que funcionaban correctamente. Para ello habría que entrar al formulario de Navision donde se encontraban todos los datos de las piezas y comprobar que en caso de haber repasado una pieza éste saldría en último lugar.

6.1.4 Familiarización Visual Studio 2015

El siguiente paso en la realización del proyecto sería la instalación del IDE de desarrollo de software Visual Studio 2015 (adquirido desde la UPV) y a continuación, comenzaría un proceso de adaptación, el cual, después de varias pruebas con distintos sitios webs sencillos utilizando el IIS local del propio servidor, se llegó a la conclusión de que sería más sencillo programar las nuevas aplicaciones web desde el portátil de trabajo. De esa manera creando una aplicación web vacía y dándole una referencia de “WindowsForms”, sería más sencillo subirlo al servidor web por medio de la carpeta compartida anteriormente comentada.

Continuando con el desarrollo del proyecto, lo siguiente iba a ser comprobar el funcionamiento del servicio web del contametros creado antes. Para ello, se necesitaría crear una aplicación web de prueba que poseyera un único botón el cual iba a leer el metraje del contametros. Dicho botón constaba de un evento de acción que, al clicarlo, lanzaría la función de lectura, y así acabaría escribiendo los datos del metraje en un cuadro de texto. Para referenciar dicha aplicación con el servicio web, únicamente había que agregar una referencia de servicio al proyecto y seguidamente la referencia web, indicando la URL completa donde se alojaba. Como cada máquina de repasado disponía de un contametros distinto, habría que hacer una referencia diferente para cada una de ellas. Después de hacer la referencia, ya sería posible utilizar las funciones de lectura y puesta a cero del contametros en el proyecto.

También como última referencia a mencionar, la propia dispuesta en el “codeunit” de Navision llamado “WebService” el cual, fue creado con la intención de insertar o borrar incidencias, enviar los datos de repasado a la base de datos, y cambiar el carro a uno nuevo. Dicha referencia llamaba al propio servidor en el que se encontraba alojado Navision dentro de la carpeta del “Codeunit”; abriendo también varios puertos concretos para posibilitar la comunicación.

La última prueba iba a estar relacionada con las consultas a la base de datos de Navision donde se iban a encontrar todas las tablas con toda la información necesaria. Utilizando ASP.NET era sencillo ejecutar

consultas SQL ya que, únicamente se necesitarían crear los orígenes de datos configurándolos debidamente: la conexión de datos, el comando SELECT para permitir seleccionar las columnas que se mostrarían y su orden, y una condición de filtro de las filas devueltas en caso de querer que cumplieran ciertas condiciones.

6.1.5 Diseño y montaje de la web

El diseño de la web debía tener unos objetivos claros, sobre todo para el usuario que tenía que saber dónde encontrar cada funcionalidad en cualquier momento. La navegación de la web debía de ser a su vez fácil, más o menos intuitiva y consistente, se debían usar el menor número de saltos de página posible, y se debía tener en todo momento la mayor cantidad de información en pantalla posible. Además, la interfaz debía ser lo más uniforme posible a través de la web: sistemas de colores diferenciados para facilitar la navegación, un mismo uso de tipografía para simplificar la legibilidad en relación con los colores elegidos, la colocación de los elementos, etc. También era muy importante tener en cuenta la resolución en la que se iba a desplegar la web principalmente; ya que se iba a utilizar una resolución de 1024x768 en todos los equipos, era mejor adaptar la web a esa resolución aunque también poseyera un diseño adaptable en algunos casos.

Por otro lado, ya que la mayoría de acciones que se iban ejecutar producirían cambios de estado en la base de datos, era importante utilizar ventanas emergentes en ciertas ocasiones para informar de ello, pero sin abusar, ya que haría la navegación menos atractiva.

Al haber hecho las pruebas necesarias previamente con las consultas y referenciando los servicios web, quedaría comenzar con el diseño en sí. El primer punto iba a ser la página de inicio de sesión de usuario, ya que ésta sería la primera que verían los usuarios. Típicamente, una web tiene una página de inicio; y de ésta dependen una serie de páginas. En cuanto al código HTML, destacar que en la cabecera se iban a vincular las hojas de estilos CSS utilizados durante todo el proyecto (bootstrap), además de sus correspondientes scripts. Por otra parte, el cuerpo de esta página constaría de dos contenedores:

- La **barra de navegación**: aparecería en todas las páginas, sería visible en todo momento y constaría de ciertos botones dependiendo de la página en la que se encuentre el usuario.
- El **formulario** de entrada: en este formulario figurarían cuatro partes claves para la correcta autentificación de usuario y de la máquina:
    - La caja de texto en la que se introduciría el código de usuario.
    - Un listado en el que se seleccionaría la máquina en la que se encuentre el operario, aunque ésta vendría preseleccionada en función del PC en el que se encuentre. Al mismo tiempo se dejaría cambiar de máquina en caso de ser necesario. Además, constaría de una herramienta para hacer posible la selección del origen de los datos en la base de datos.
    - Una etiqueta que aparecería un texto de error en color rojo en caso de equivocación de usuario durante el inicio de sesión o de no haber seleccionado ninguna máquina.
    - Botón aceptar para enviar los datos del formulario a la base de datos y de ese modo corroborarlos.

En cuanto al código de visual basic, además de la función de autoselección de la máquina, se añadirían ciertas variables de sesión de máquina y usuario para guardarlas en la siguiente página.

La siguiente página sería la de introducción de carro o pieza para comenzar posteriormente con el repasado. En cuanto a la cabecera, como ya he dicho, funcionaría de la misma manera que en la página de inicio. En cambio, el cuerpo tendría funcionalidades distintas como:

- La **barra de navegación**: además de ser visible en todo momento como en la página anterior, se añadirían dos etiquetas en las que aparecerían el usuario y la máquina introducidos, asimismo un botón de salir para cerrar la sesión.
- El **formulario** principal: aparecerían dos contenedores en los que aparecería toda la información relativa a la búsqueda de carro y pieza:

- Dos cajas de texto, una para la introducción de carro y otra para introducir la pieza en caso de no disponer del carro; un botón para enviar el formulario (solo la parte de las cajas de texto); y dos botones que solo aparecerán activos una vez esté seleccionado el carro o la pieza y se podría ver las incidencias totales o proceder a imprimir el albarán.
- Dos tablas en las que aparecerían todas las piezas del carro o la pieza seleccionada (repasada y no repasada) en las que se mostrarían todos los datos de las piezas así como un botón para comenzar su repasado.

También aparecería un último contenedor en el que está montado el diseño del albarán pero que no se mostraría en la web, únicamente se vería en una nueva ventana al pulsar el botón de imprimir albarán. Estaría dividido por tablas junto a las consultas necesarias para que aparecieran los datos del carro seleccionado (número de piezas, pesaje total, metros totales, empresa, dirección de envío, etc.)

La siguiente página, y al mismo tiempo una de las más importantes ya que enviaría la mayoría de datos a la base de datos, iba a ser la propia del repasado. Como siempre, la cabecera dispondría de las mismas vinculaciones que las páginas anteriores, y la barra de navegación presentaría los datos de usuario y máquina seleccionado al igual que en la página anterior, pero excluyendo el botón de salir ya que para salir de esa página era necesario enviar los datos recogidos hasta el momento. También incluiría un script, el cual recargaría la página sin borrar ningún dato del formulario introducido para que, en caso de pulsar algún botón no deseado en el teclado (por ejemplo el botón de retroceder) éste no tuviera efecto al menos una vez cada dos minutos. Por otro lado, el cuerpo dispondría de los siguientes campos:

- Una **tabla** donde aparecen: el metro por el que va el repasado de esa pieza; los metros realizados durante la sesión de repasado; la fecha y hora de inicio de repasado de esa sesión; un botón de paro de repasado que envía los datos de repasado hasta el momento a la base de datos, no reinicia el contametros y sirve para ausentarse del repasado en caso de cambio de turno de operario; y un botón de finalizar repasado que envía todos los datos a la base de datos y pone el contador del metraje del contametros a cero para repasar una nueva pieza.
- Un **contenedor** en el que aparece una cuadrícula con los datos de la pieza a repasar (número de pieza, producto que contiene, metros de la pieza, carro al que pertenece).
- Una nueva **tabla** con un cuadro de texto para introducir un nuevo carro y un botón de enviar que permite cambiar la pieza a un carro distinto.
- Otra **tabla** en la que aparecen diez filas distintas habilitadas para introducir hasta diez tipos de incidencias distintas al mismo tiempo. En cada columna aparece un dato indispensable para ser enviado (máquina, usuario, un desplegable para seleccionar el código de la incidencia, un cuadro de texto para introducir la cantidad de incidencias del mismo tipo, otro cuadro de texto para introducir el metro donde empieza y acaba la/las incidencia/s, un botón para seleccionar si ha sido arreglada o no, y un botón para enviar todos esos datos de esa incidencia).
- Un último **contenedor** donde irían apareciendo todas las incidencias enviadas en tiempo real, y en caso de querer eliminar una por equivocación, un botón disponible para ello.

En cuanto al código de visual basic se verían todas las variables creadas, tanto de sesión como locales, el cálculo de los metros a enviar, las referencias a la base de datos de Navision, tanto de insertar y eliminar incidencias, como el comienzo de repasado, o incluso el cambio de pieza a un nuevo carro. Todas estas variables enviadas dependen según la máquina y el usuario seleccionados. De este modo se controlan una gran cantidad de datos, que posteriormente serán necesarios para su estudio.

A continuación, en cuanto a la impresión de incidencias, únicamente constaría de una tabla en la que aparecerían todas las incidencias de todas las piezas del carro seleccionado. De esta manera sería posible imprimirlas y colocarlas en dicho carro si los operarios lo deseasen. Ocurre lo mismo con el albaranado, dando

la opción de imprimir el número de piezas dentro del carro y cierta información necesaria para el camión que enviaría dicho carro a otra empresa.

Pese a que la aplicación principalmente solo abarcaría el repasado de piezas, conforme se iba avanzando se sabía que eran necesarias ciertas funcionalidades como: una autentificación para que cada operario tuviera su propio código y de ese modo se tuviera un mayor control en relación al usuario, búsqueda de carros donde aparecieran todas las piezas de dicho carro o bien búsqueda de piezas diferenciando aquellas que ya se hubieran repasado al menos una vez, eliminar incidencias en caso de equivocación, botón de paro de repasado en caso de necesidad sin reiniciar el metraje del contametros (ya que una vez se finalizase el repasado de una pieza el contador del metraje se pondría a cero automáticamente), impresión del total de incidencias de un carro si fuera imperioso, impresión de albarán de un carro con las piezas repasadas para el envío del mismo, etc.

Por último, se iniciaría la fase de pruebas con operarios para de ese modo saber si era necesario añadir nuevas funcionalidades, ya que éstos iban a ser los principales consumidores de la aplicación.

6.1.6 Qlikview

Durante el periodo de pruebas, gracias a que los operarios iban utilizando la aplicación, se disponía de una buena cantidad de datos en la base de datos y de ese modo también se podía ver si la aplicación se estaba utilizando del modo indicado. Con ello, se utilizó un software llamado Qlikview; el cual ofrecía una plataforma para el análisis visual que aportaba conocimiento, datos y claridad en la toma de decisiones, y a su vez garantizaba esa toma de decisiones con una mayor seguridad.

Con la ayuda de nuevo del tutor de la empresa, aprendí como utilizar las sentencias SQL necesarias de la base de datos en cuestión, y de ese modo utilizando ciertas gráficas y estadísticas, se podían examinar los datos que los operarios habían introducido previamente.

Para empezar es necesario editar un nuevo script con el objetivo de introducir las sentencias SQL necesarias. Entre ellas se encontraban:

- **Variables de interpretación numérica** del script: una serie de variables introducidas que ayudan en la interpretación de la sintaxis del script:
  - ThousandSep=’.’: el separador de miles definido reemplaza al símbolo de agrupación de dígitos del sistema operativo (configuración regional).
  - MoneyThousandSep=’.’: el separador de miles definido remplaza el símbolo de agrupación de dígitos para moneda del sistema operativo (configuración regional).
  - TimeFormat=’h:mm:ss’: el formato definido reemplaza el formato de hora del sistema operativo (configuración regional).
  - DateFormat=’DD/MM/YYYY’: el formato definido reemplaza el formato de fecha del sistema operativo (configuración regional).
  - MonthNames=’ene.;feb.;mar.;abr.;may.;jun.;jul.;ago.;sep.;oct.;nov.;dic.’: el formato que se ha definido reemplaza los nombre de los meses del sistema operativo (configuración regional).
  - Entre otros…
- La sentencia para la **conexión OLEDB** permitiendo enlazar los paquetes con la fuente de datos en cuestión. En este caso la conexión OLEDB utiliza el proveedor de Microsoft OLEDB para SQL Server. Sería necesario introducir dicha conexión OLEDB y a su vez incluir el origen de datos para archivos de SQL Server. Una vez se crea la conexión, indicando tanto el correspondiente usuario y contraseña como la fuente de datos, ésta recibe un nombre de origen de datos (DSN) que posteriormente se utiliza para identificar conexiones en controles de datos.
- Por último habría que añadir las **consultas** necesarias de los datos introducidos por los usuarios de la aplicación. En este caso dichos datos se encontraban en diferentes tablas y además había que

utilizar distintos filtros como: los datos introducidos a partir de 2015, que pertenecieran las piezas al grupo de repasadas, o incluso que el estado de la pieza fuera el adecuado.

6.2 DIFICULTADES Y SOLUCIONES

Durante la realización del proyecto fueron surgiendo ciertos contratiempos, que además de retrasar su elaboración, hacía que su desarrollo fuera más cautivador e interesante. En cuanto a las dificultades durante la composición del proyecto cabría destacar sobre todo las propias durante el periodo de programación en este lenguaje llamado ASP.NET, el cual nunca había tenido la oportunidad de probar. Sin embargo, gracias a su similitud con otros lenguajes de programación de webs dinámicas, el uso general de código del lado del servidor y su simpatía con Windows, hizo su desarrollo un tanto más sencillo de lo esperado. Aun así, las principales dificultades serían las siguientes:

6.2.1 Durante el proceso de comunicación con el contametros

- **Comodidad en las pruebas:** en la empresa se disponía de una sala cerca de las oficinas donde principalmente se desarrollaba el proyecto. Ya que el cuarto donde se encontraban las máquinas de repasado estaban algo lejos de esta sala, no era óptimo ir y venir en cada momento que se quería hacer una prueba con el PLC. Por ello se decidió hacer las pruebas en el cuarto de las máquinas utilizando el portátil y de ese modo se eliminaría esa pérdida de tiempo de desplazamiento.
- **Aplicación de prueba:** para saber el funcionamiento del PLC era necesario disponer del PDF de configuración del puerto serie, así como los códigos que había que enviar para recibir una respuesta correcta. Después de varias pruebas, se llegó a la conclusión de que, según estaba programado la función de poner a cero el contador, sólo leería hasta diez bytes recibidos por el búfer. En cambio, la función de lectura leería hasta diecinueve bytes del búfer. Éstos números se encontraron haciendo pruebas hasta que el código recibido, una vez traducido a metros, era el deseado, y con la ayuda del encargado de la programación del PLC.

6.2.2 Durante el proceso de adaptación a los servicios web

- **Administración de las funciones del servidor IIS:** al no tener práctica con los servidores web IIS, la única forma de obtener los resultados deseados era a base de ensayo y error. Entre algunos de los problemas se encontraban los relacionados con la configuración del administrador del IIS (utilización del framework correcto, agregar el proyecto al grupo de aplicaciones correcto para que la versión del .NET no diera problemas, edición de los permisos necesarios a la hora de subir los datos al servidor, etc.) Además, era necesario activar, dentro del archivo de configuración del proyecto, los mensajes de errores para que dieran información más concreta a la hora de buscar la fuente, en caso contrario habría que lanzar el proyecto directamente desde el servidor web para poder ver dichos errores.
- **Creación del servicio web:** después de varias pruebas con el Visual Studio y buscando información por internet, se aprendió cómo crear un servicio web que posteriormente iba a ser referenciado para la aplicación final. Para ello se debía crear un nuevo sitio web vacío que, además se le agregaría el elemento de Web Service (ASMX) programando las dos funciones necesarias (lectura y puesta a cero del contador).
- **Programación del servicio web:** el código de las dos funciones se asemeja en los siguientes puntos:
    - Ambas funciones comenzaban cerrando el puerto serie ya que en caso de no hacerse, si ocurría algún fallo durante el proceso de lectura o puesta a cero, éste se quedaría abierto y ya no se podría acceder a él, por lo que sería necesario desconectar y volver a conectar el cable COM físicamente.
    - Seguidamente era necesario programar la configuración del puerto serie para que cada vez que se lanzara la función se ejecutara dicha configuración automáticamente. Sobre todo se encontraron dificultades en el protocolo de handshaking, el cual se encarga de establecer los

parámetros por la transmisión de datos del puerto serie antes de que comience la comunicación. Este parámetro no se encontraba en la hoja de configuración y por lo tanto fue un tanto complicado configurarlo.

- A continuación habría que abrir el puerto serie de nuevo para enviar el código de lectura al PLC junto al carácter de retorno de carro (vbCr) para la función de impresión. A su vez, como ya he dicho antes, el programa iría leyendo byte a byte el búfer hasta llegar a la cantidad reconocida por la función en cuestión.
- En cuanto a la función de lectura, sería necesario hacer el cálculo del metraje utilizando la cadena recibida y finalmente devolviendo el resultado. Por otro lado, la función de puesta a cero únicamente devolvería un valor booleano.

6.2.3 Durante el proceso de utilización del ERP Navision

- **Usando C/AL de Navision:** Navision emplea un lenguaje parecido a pascal el cual, junto a algunas herramientas de desarrollo, puede diseñar ciertos objetos (tablas, informes, formularios…) y utilizarlos durante la programación en Navision. Para poder programar en este lenguaje fue necesario guiarse utilizando programas antiguos, cuyos objetos y funciones eran parecidas a las que se iban a necesitar para el servicio web. Gracias a que Navision avisaba si existían errores en el código imposibilitando su compilación, facilitó la comprensión de la estructura. En cuanto al código, utilizando el método de prueba y error y comprobando los resultados en las tablas correspondientes, fue posible ejecutar un código lo suficientemente estable y que hiciera lo requerido.
- **Hora de Navision:** surgieron algunos problemas con la función de tiempo porque la aplicación recogía la hora correcta pero Navision devolvía un formato de hora distinto. Por ello, finalmente se optó por hacer un pequeño cálculo después de recoger dicha hora.
- **No aparecía la referencia del "codeunit" del "web service" de Navision:** cuando se acabó toda la fase del contametros, era el momento de continuar con las pruebas de llamada del servicio web en el proyecto. Para ello, como ya he dicho en otros apartados, era necesario agregar una referencia en el proyecto y utilizar la dirección donde se encontraba el "codeunit" de Navision. El problema surgió cuando después de varios intentos utilizando distintas claves y direcciones no podía acceder al directorio. El problema finalmente era la afluencia dentro del servidor por lo que se decidió hacer un cambio de hardware aumentado la memoria RAM y de ese modo finalmente se pudo acceder.

6.2.4 Durante el proceso de familiarización con Visual Studio

- **Uso de Visual Studio:** en un principio el IDE que se iba a utilizar iba a ser la versión de 2010 ya que la mayoría de aplicaciones de ese tipo que existían en la empresa estaban realizadas con este. Sin embargo al saber que existía una nueva versión (2015) era interesante probarla para ver si ofrecía diferentes funcionalidades y si facilitaba su uso. Por ello habían ciertas funcionalidades diferentes en cuanto a dichas versiones y para conocer su funcionamiento era necesario hacer búsquedas en internet. Aun así, finalmente encontré el IDE más intuitivo que el anterior.
- **ASP.NET:** en cuanto al lenguaje de programación, no destacaría ninguna dificultad en concreto. Aunque no había trabajado con él anteriormente, cuando era necesario añadir una nueva funcionalidad únicamente con las nociones básicas en PHP y buscando por internet siempre se podía sacar una solución, o al menos parcialmente.
- **.NET Framework:** había que tener cuidado con la compatibilidad de versiones en .NET Framework ya que, de manera predeterminada, una aplicación se ejecuta en la versión de .NET Framework para la que se creó. Si esa versión no está presente y en el archivo de configuración no se han definido las versiones compatibles, puede producirse un error de inicialización de .NET y por lo tanto el intento de ejecutar la aplicación no se realiza correctamente. Al crear un nuevo proyecto en Visual Studio 2015 la versión por defecto es la 4.5, por lo que el archivo de configuración va a contener esa versión. Para que funcione correctamente, la versión configurada en el servidor web IIS en el que se alojaría la aplicación debería ser la misma o al menos una compatible. Para definir las versiones concretas en

las que se ejecutaría la aplicación, habría que agregar uno o varios elementos de <runtimeCompatible> al archivo de configuración de la aplicación. Además, era importante recordar que el primer elemento especificaba la versión de mayor preferencia y el último, la de menor preferencia. En una ocasión se tuvo que crear un nuevo proyecto cambiando el framework ya que hubo problemas con el archivo de configuración y al modificarlo saltaban distintos errores.

6.2.5 Durante el proceso de diseño de la web

- **Crystal reports (albarán):** el albarán en un principio no era algo necesario, pero el encargado de repasado insistió en que fuera posible imprimir las piezas que llevaría un carro al ser transportado junto a otro atributos como: peso, cantidad de metros, destino, etc. Para ello se realizó mediante un contenedor en HTML en el que aparecieran todos estos datos y que fuera invisible en la página para que no ocupara espacio. Seguidamente únicamente con una función para imprimir dicho div estaba resuelto. Igualmente surgieron distintos problemas sobre todo en cuanto a los códigos de barras ya que éstos no aparecían correctamente. Finalmente se entendió que era necesario instalar una fuente concreta en el servidor y reiniciarlo para que funcionara; además de añadir ciertos scripts al proyecto. Más tarde, mi tutor de la empresa me dijo si era posible utilizar una funcionalidad llamada Crystal Reports de Visual Studio que, además de ser una aplicación para diseñar y gestionar informes desde una fuente de datos organizándolos por filas y columnas según el formato necesario, haría los formularios más sencillos a la hora de imprimirlos. Pero finalmente esto se dejó como un trabajo futuro.
- **Botones retroceso e intro del teclado:** durante el periodo de pruebas algunos operarios tenían algunas quejas ya que, era posible que estuvieran introduciendo datos en la aplicación y pulsando el botón retroceso se salieran de la aplicación perdiendo así los datos acumulados que no se habían enviado. Del mismo modo apretando intuitivamente el botón intro se accionaba el primer botón que había en la web, por lo que podría significar perder, del mismo modo, el trabajo realizado. Para solucionar el problema del botón de retroceso se pensó en inhabilitar el botón, pero se supuso que no sería práctico, por ello se llegó a la conclusión de inhabilitar el botón en lugar de para siempre, cada cierto tiempo. De esta forma, se utilizó un script que cada dos minutos actualizara la página añadiendo un # al final de la URL para que si se pulsaba el botón de retroceso se borrara ese # y no se perdiera nada de la información recogida hasta el momento. Por otro lado, habían dos formas para solucionar el problema del botón intro: una era utilizando una funcionalidad de ASP.NET que, mediante extensiones AJAX, en función del cuadro de texto desde donde se pulsara intro accionaría un botón en concreto haciendo de ese modo lo deseado; o únicamente poniendo un botón invisible al comienzo de la página que no hiciera nada y así al pulsar intro no sucedería nada.
- **Acuerdos entre operarios y jefes:** muchas veces a la hora de añadir o cambiar funcionalidades de aplicación éstas no gustaban a todos. Por ello siempre había que tener en cuenta tanto a los jefes que dictaban como quería que fuera, así como a los operarios que iban a utilizar la aplicación. Después de muchas reuniones tanto con encargados de los operarios como con el jefe, se llegaron a conclusiones que más o menos satisfacían a ambos.

6.2.6 Durante el proceso de analizado de los datos

- **Datos enviados a Navision:** en algunos casos, habían algunas piezas llegaban sin número o el tiempo de repasado era excesivamente pequeño teniendo en cuenta el producto utilizado en esa pieza de tejido. Finalmente, se descubrió que ese operario estaba utilizando la aplicación erróneamente; ya que en algunas ocasiones el tiempo de espera de envío de incidencias o de finalización del repasado era mayor de lo habitual, se entendía que el operario salía de la aplicación demasiado pronto y de ese modo no dejaba tiempo al servidor de recoger los datos correctamente. Se solucionó añadiendo ventanas emergentes en caso de tardar un tiempo excesivo para dar el tiempo necesario tanto a la aplicación de enviar los resultados como al servidor de recogerlos.

7 CONCLUSIONES Y TRABAJO FUTURO

En este proyecto se ha conseguido el objetivo propuesto: se ha diseñado una herramienta para facilitar la recolección de datos durante el repasado de las piezas de tejido con los que posteriormente, dependiendo de los requisitos particulares del cliente, se facilitaría su estudio para una posible optimización de los recursos. En este capítulo se resaltarán tanto los resultados obtenidos como las líneas de trabajo futuro.

7.1 RESULTADOS OBTENIDOS

Los resultados más relevantes obtenidos durante el desarrollo y la ejecución del proyecto se pueden resumir en los siguientes puntos, presentados así mismo como conclusiones:

- **Los servicios web** son muy útiles en cuanto a la comunicación de diferentes aplicaciones. Como ya he dicho en otros puntos, se trata de una tecnología que utiliza una serie de protocolos y estándares que sirven para intercambiar datos entre distintas aplicaciones de software desarrolladas en distintos lenguajes de programación y ejecutadas sobre cualquier plataforma. Además estas aplicaciones pueden utilizar estos servicios web para intercambiar datos.

En este caso, Navision proporciona servicios web que facilitan la integración de otros sistemas y viene soportado a través de "codeunits". Añadir estas extensiones "codeunit" a una página es ventajoso si se quieren desarrollar operaciones más allá de las de lectura y escritura decidiendo de ese modo cuáles.

- El uso de un **servidor web** dentro de la empresa proporciona cierta flexibilidad en cuanto a la programación. Los servidores web procesan aplicaciones del lado del servidor; en el caso de esta aplicación se utiliza un servidor de la empresa el cual se utiliza exclusivamente para el mantenimiento de la web. El código recibido por el cliente se compila y se ejecuta por el navegador web utilizado dentro de la red de la empresa accediendo de ese modo a la aplicación.

En este caso, el servidor web se encuentra dentro de la empresa y por ello proporciona mayor facilidad a la hora de manejarlo así como de configurarlo. Para ello se utiliza el rol servidor web (IIS) el cual, proporciona una plataforma segura y fácil de administrar a la hora de hospedar sitios web, servicios y aplicaciones de manera fiable, y que además unifica e integra ASP.NET con los servicios de FTP. Algunas ventajas que se obtienen al utilizarlo serían:

  - La seguridad se refuerza gracias a un aislamiento automático de las aplicaciones conseguido al proporcionar a los procesos de trabajo una identidad única.
  - Aumenta la velocidad del sitio web mediante el almacenamiento en caché dinámico integrado.
  - Posibilidad de utilizar el protocolo de transferencia de archivos para permitir que los propietarios del sitio carguen y descarguen archivos.
  - Configuración de aplicaciones web que están escritas con varias tecnologías, como ASP, ASP.NET y PHP.
- **ASP.NET** es eficaz a la hora de realizar aplicaciones ya que, al formar parte de .NET Framework y al codificar las aplicaciones ASP.NET, tiene acceso a distintas clases en .NET Framework como:
  - El espacio de nombre **System** contiene clases fundamentales y clases base que definen tipos de valor de uso frecuente, eventos y controladores de eventos, interfaces, atributos y excepciones de procesamiento.
  - Los espacios de nombres **System.Data** contienen clases para tener acceso a datos y administrarlos desde distintos orígenes. Estos espacios de nombres forman conjuntamente la arquitectura ADO.NET junto a sus proveedores de datos (SQL Server, Oracle, ODBC y OleDB).
  - Los espacios de nombres **System.IO** contienen tipos que admiten entrada y salida, incluida la posibilidad de leer y escribir datos en flujos de forma síncrona o asíncrona, asignar archivos al

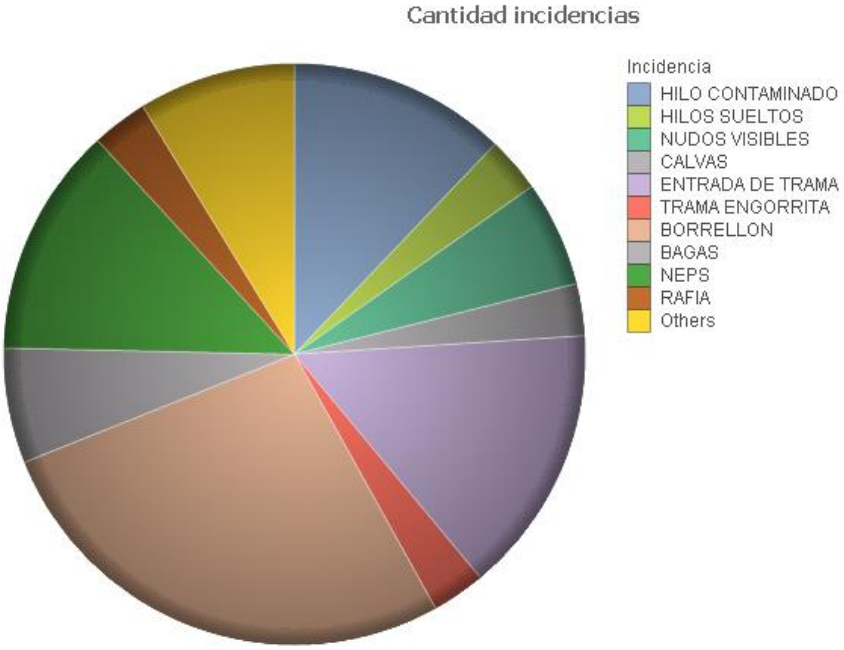
espacio de direcciones lógicas de una aplicación, etc. El aprovechado en esta aplicación sería el **System.IO.Ports** utilizado para administrar el flujo de datos hacia y desde puertos serie.

- El espacio de nombres **System.Web.Services.Protocols** está formado por las clases que definen los protocolos utilizados para transmitir datos a través del cable durante la comunicación entre los clientes de servicios Web XML y los servicios Web XML creados mediante ASP.NET.

Además, ASP.NET proporciona un amplio rango de herramientas que facilitan la comprensión en cuanto a la programación, sobre todo en cuanto a la integración con las bases de datos ya que permite la conexión con cualquier base de datos SQL compatible.

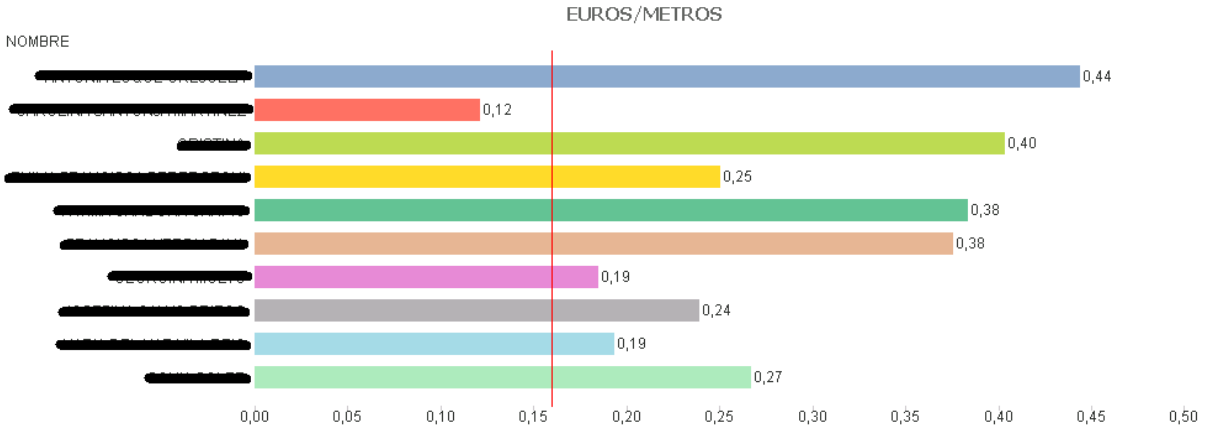
- El uso de **Bootstrap** para el diseño adaptable de la aplicación en diferentes dispositivos. Esta herramienta haría que, con la ayuda de ciertas librerías jQuery, el aspecto visual de la aplicación fuera mucho más satisfactoria visualmente. Esto sería de agradecer por los operarios ya que facilitaría su uso y además ayudaría a que éstos quisieran realmente sustituir el papel y bolígrafo por ella.
- Los datos del **qlikview** extraídos son importantes ya que son los que finalmente se estudiarán. Existen ciertas estadísticas importantes ya que por ellas se sabrá si vale la pena seguir utilizando un tipo de producto concreto o si el tiempo que utilizan los operarios durante el repasado es el correcto. Los siguiente datos expuestos no son datos que realmente tengan una gran relevancia, pero serán los mostrados para intentar reservar la privacidad de la empresa. Realmente lo interesante de este software es utilizar una serie de tablas en las que seleccionando el tipo de incidencia buscado, o la máquina, o incluso el momento de repasado, éste muestre en tiempo real toda la información requerida. Es decir, si se crean objetos de tipo tabla con los datos que se quieren mostrar, éstos irán cambiando en función de lo seleccionado.

En la siguiente gráfica se pueden observar que durante el mes de marzo de 2016 el mayor número de incidencias encontrado durante el repasado serían los borrellones seguido de las entradas de trama. Ya que no es un resultado realmente relevante no se puede sacar mucha información al respecto pero sí que se puede decir que sin importar el producto los borrellones, al ser taras de pequeña importancia, son los más encontrados al repasar tejido.



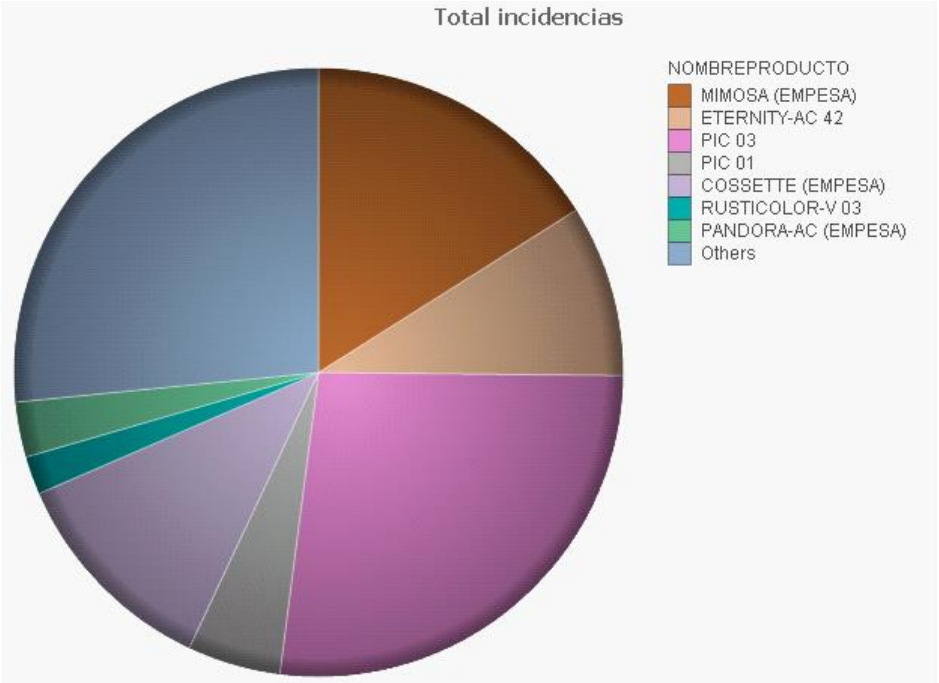
Gráfica de cantidad de incidencias por tipo de incidencia (Fig. 11)

A continuación, en la siguiente gráfica sí que se observa algunos datos algo más importantes en cuanto a los operarios y el consumo utilizado durante el repasado. Se puede decir que hay una gran diferencia en cuanto al primer y el segundo operario. Es posible que el primero utilice una cantidad mayor de dinero que el segundo.



Gráfica de euros/metro por operario (Fig. 12)

En la siguiente gráfica se puede observar como el producto llamado PIC 03 es el que más incidencias recoge durante el repasado. Utilizando este dato más el tiempo que se tarde en repasar y el gasto que esto suponga se puede decidir si vale la pena seguir repasando este tipo de producto o no.



Gráfica de total de incidencias por tipo de producto utilizado (Fig. 12)

7.2 SATISFACCIÓN DEL PRODUCTO FINAL Y SU UTILIZACIÓN EN LA EMPRESA

El producto final radica en la necesidad de la recolección de datos para poder ser analizarlos más tarde. Aunque sí que es verdad que el uso de la aplicación ha facilitado el estudio de los datos, sería quizá necesario simplificar su utilización porque, pese a que es cierto que la completitud de datos es ligeramente mayor que en la utilización únicamente de papel y bolígrafo, su manejo puede resultar un tanto tedioso si no se usa correctamente.

Por otra lado, la aplicación vendría a ser utilizada únicamente por los operarios, por lo que ellos serían los encargados de manifestar si el resultado obtenido era satisfactorio o no. Ya que hubo un proceso de prueba de varios días, fue posible añadir y editar funcionalidades en la aplicación para que su uso fuera lo más satisfactorio posible. En cambio, tanto el jefe como el encargado de repasado serían los que se ocuparían de ver si su utilización era necesaria dentro de la empresa, debido a que serían ellos los que verían los resultados estadísticos finales.

7.3 CONCLUSIONES DEL TRABAJO REALIZADO

Una vez llegado a este punto del proyecto, la simple idea de estar finalizándolo me aporta una gran satisfacción, por el simple hecho de que hace más de medio año me parecía algo muy lejano y en cierto modo inalcanzable. Ya que esta aplicación se ha usado en una empresa real (que continúa usándose en estos momentos), he aprendido a enfrentarme a un trabajo real del día a día de cualquier trabajador.

Además, la realización del proyecto me ha valido como referencia a la hora de tomar decisiones y afrontar distintas tareas: aquellas que parecían sencillas resultaban ser más complicadas por el hecho de no haber trabajado con ellas antes; y tareas que a simple vista parecían más complicadas, sacarlas adelante con mayor facilidad de lo esperado.

Para acabar, la elaboración de este trabajo me ha dado a conocer un nuevo mundo en cuanto a la programación web y me ha hecho crecer en el ámbito laboral aprendiendo, además de lo relacionado al proyecto, otras tareas adicionales que iban surgiendo dentro de la empresa. Igualmente, la idea de eliminar sesiones de usuario de Navision y utilizar únicamente un navegador para poder hacer las mismas funciones me parece algo muy interesante y me hace pensar que puede ser muy eficaz si se profundizase más en ello. Finalmente decir que me siento bastante realizado viendo los resultados obtenidos.

7.4 TRABAJO FUTURO

La utilización de la aplicación en un principio, aunque fuera adaptable en cuanto a su diseño web, estaba pensada para su uso en los ordenadores “ThinClient” de las máquinas. En algunos casos, dichos ordenadores estarían colocados en zonas de más difícil acceso para el operario por culpa de la disposición de las máquinas. Por ello, se pensó que en un futuro podría ser interesante utilizar tablets que facilitarían el uso de la aplicación. De esta forma se pretendería evitar problemas como: incomodidad, dificultad de acceso o manejo.

Por ejemplo, en el caso de las tablets, no necesitarían un lugar fijo donde ser colocadas y por ello el operario podría transportarla y utilizarla en cualquier lugar y cualquier momento. Además, ya que la aplicación se ejecuta únicamente con un navegador, no necesitaría ningún software adicional que pudiera ralentizar la tablet; únicamente con un navegador web instalado ya sería posible su uso. Esto podría dar una mayor fluidez y de ese modo la satisfacción del usuario sería mayor.

Por otra parte, existen empresas externas que se dedican únicamente al repasado de tejido. Estas empresas al estar fuera de la red de la propia empresa no serían capaces de utilizar la aplicación de modo normal. Por ello se podría utilizar algún servidor dedicado únicamente al exterior y que de ese modo estas empresas externas pudieran utilizar la aplicación vía internet. Al utilizar la misma aplicación alojada en un mismo servidor dentro de la empresa los datos se verían reflejados del mismo modo en la base de datos. Además, ya se fueron haciendo pruebas con un servidor de prueba detectando automáticamente si la dirección IP desde la que se ejecutaba la aplicación no era una de las conocidas, de ese modo automáticamente se trataría de una repasadora externa.

8 BIBLIOGRAFÍA

- Información general de un controlador lógico programable. Recuperado del sitio web de la Wikipedia: https://es.wikipedia.org/wiki/Controlador_lógico_programable
- Enrique, L. (2013). Supervisión y Automatización Industrial / Control por PLC. Recuperado del sitio web: http://es.slideshare.net/josse-anlo/control-por-plc
- Información general de Microsoft Dynamics NAV. Recuperado del sitio web de la Wikipedia: https://es.wikipedia.org/wiki/Microsoft_Dynamics_NAV
- *Introduction to C/AL / Using C/AL.* Recuperado del sitio web de Microsoft: https://msdn.microsoft.com/en-us/library/dd355277.aspx
- *Developing Codeunits.* Recuperado del sitio web de Microsoft: https://msdn.microsoft.com/en-us/library/dd355410.aspx
- Introducción al framework de desarrollo adaptable. Recuperado del sitio web de Bootstrap: http://getbootstrap.com/
- *Wutlu*. *Findcond Navbar / Bootsnipp.* Recuperado del sitio web de Bootsnipp: http://bootsnipp.com/snippets/featured/findcond-navbar
- Información general acerca de .NET Framework. Recuperado del sitio web de Microsoft y de la Wikipedia: https://msdn.microsoft.com/es-es/library/zw4w595w(v=vs.110).aspx https://es.wikipedia.org/wiki/Microsoft_.NET
- Información general del framework para aplicaciones ASP.NET. Recuperado del sitio web de la Wikipedia: https://es.wikipedia.org/wiki/ASP.NET
- Información general del lenguaje de programación Visual Basic. Recuperado del sitio web de la Wikipedia: https://es.wikipedia.org/wiki/Visual_Basic_.NET
- Información general sobre Qlik. Recuperado del sitio web de la compañía: http://global.qlik.com/es/company
- Introducción a la arquitectura cliente – servidor. Recuperado del sitio web de la Wikipedia: https://es.wikipedia.org/wiki/Cliente-servidor
- Introducción a los servicios web. Recuperado del sitio web de la Wikipedia: https://es.wikipedia.org/wiki/Servicio_web
- *Walkthrough: Creating and Using an ASP.NET Web Service in Visual Web Developer.* Recuperado del sitio web de Microsoft: https://msdn.microsoft.com/en-us/library/8wbhsy70(v=vs.90).aspx
- Kelvin Le (2012). *Visual Basic Serial COM Port Tutorial (Visual Studio).* Recuperado del sitio web de Youtube: https://www.youtube.com/watch?v=krN6pWm6s6o
- Trabajar con Qlikview / Sintaxis de script / Variables de interpretación. Recuperado del sitio web de Qlikview: https://help.qlik.com/es-ES/qlikview/12.0/Subsystems/Client/Content/Scripting/NumberInterpretationVariables/number-interpretation-variables.htm
- *Script for Browser backward inactivation.* Recuperado del sitio web de StackOverflow: http://stackoverflow.com/questions/29590041/how-can-i-prevent-the-browser-to-scroll-to-the-top-on-hash-change
- *Introduction UpdatePanel Control (AJAX) /* Corregir el refresco de la página al pulsar la tecla intro. Recuperado del sitio web de Microsoft: https://msdn.microsoft.com/en-us/library/bb399001.aspx

El resto de información ha sido suministrada tanto por el tutor de la empresa como a base de ensayo y error durante la realización del proyecto.

9 ANEXOS

A continuación aparecerán datos interesantes y que se han utilizado durante la realización del proyecto creyendo conveniente su presentación de forma separada.

9.1 HOJA DE CONFIGURACIÓN DEL PLC

CONFIGURACIÓN PUERTO SERIE:

- Bits por segundo: 9600
- Bits de datos: 7
- Bits de paridad: Par
- Bits de parada: 2

PUESTA A CERO DEL CONTADOR:

- Enviar al PLC → @00WD0100000153*{13}
- El PLC contesta → @00WD0053*{13}

CONTAJE: SENTIDO DIRECTO:

- Enviar al PLC → @00WD0500000056*{13}
- El PLC contesta → @00WD0053*{13}

CONTAJE: SENTIDO INVERSO:

- Enviar al PLC → @00WD0500000157*{13}
- El PLC contesta → @00WD0053*{13}

LEER PULSOS DEL PLC:

- Enviar al PLC → @00RD0160000253*{13}
- El PLC contesta → @00RD00XXXXYYYYZZ*{13}

Donde: XXXX → Parte baja del número de pulsos del metraje.
YYYY → Parte alta del número de pulsos del metraje.
ZZ → Checksum dependiendo de los datos

Pulsos actuales: YYYYXXXX Pulsos

CONVERSIÓN DE PULSOS A METROS:

Para transformar el número de pulsos en metros bastará con aplicar una regla de tres. Ejemplo:

Si **N** metros → **P** pulsos
M metros → **PT** pulsos leídos = YYYYXXXX

Con lo cual los metros totales:

$$\mathbf{M} = (\mathbf{PT} \times \mathbf{N}) / \mathbf{P}$$

Donde:

N metros es una distancia fácil de medir con el encoder (desarrollo de la rueda del encoder).
P es el número de pulsos leídos en esa distancia.
M es el resultado en las unidades deseadas obtenido en el cálculo.
PT el número de pulsos totales leído (YYYYXXXX).

9.2 CÓDIGO DE LA COMUNICACIÓN CON EL CONTAMETROS

```
Imports System
Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.Threading
Imports System.IO.Ports
Imports System.ComponentModel

<System.Web.Script.Services.ScriptService()>
<WebService(Namespace:="http://tempuri.org/")>
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)>
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()>
Public Class Convert
   Public SerialPort1 As New SerialPort("COM4")

   <WebMethod()>
   Public Function ponACero() As String
      SerialPort1.Close()

      Dim a As String
      a = "0"

      SerialPort1.BaudRate = 9600
      SerialPort1.StopBits = StopBits.Two
      SerialPort1.DataBits = 7
      SerialPort1.Handshake = Handshake.XOnXOff
      SerialPort1.Parity = Parity.Even

      SerialPort1.Open()

      SerialPort1.Write("@00WD0100000153*" & vbCr)

      Do While SerialPort1.BytesToRead < 10
      Loop

      a = SerialPort1.ReadExisting()

      SerialPort1.Close()

      Return True
   End Function

   <WebMethod()>
   Public Function lectura() As Double
      SerialPort1.Close()

      Dim a As String
      a = "0"

      SerialPort1.BaudRate = 9600
      SerialPort1.StopBits = StopBits.Two
      SerialPort1.DataBits = 7
      SerialPort1.Handshake = Handshake.XOnXOff
      SerialPort1.Parity = Parity.Even

      SerialPort1.Open()

      SerialPort1.Write("@00RD0160000253*" & vbCr)

      Do While SerialPort1.BytesToRead < 19
      Loop

      a = SerialPort1.ReadExisting()
```

```
        SerialPort1.Close()

        Dim Str1 As String = a.Substring(7, 4)
        Dim Str2 As String = a.Substring(11, 4)
        Dim suma As String = Str2 + Str1

        Dim Int1 As Integer = Conversion.Int(suma)
        Dim metraje As Double = Math.Round((suma) / 40) / 100

        Return metraje
    End Function
End Class
```

9.3 CÓDIGO DEL CODEUNIT DE NAVISION

InsInci(lapieza : Text[30];empieza : Decimal;producto : Text[30];UsuarioMaquina : Text[30];usuarioAlta : Text[30];cod_incidencia : Text

```
RecMP.INIT;
RecMP.Pieza:=lapieza;
RecMP.Empieza:=empieza;
RecMP.Producto:=producto;
RecMP."Fecha+hora":=COPYSTR(FORMAT(DT2DATE(FechaFinal))+'
'+FORMAT(DT2TIME(FechaFinal)),1,17);
RecMP."Fecha registro":=FechaFinal;
RecMP.Fecha:=DT2DATE(FechaFinal);
RecMP.hora:=DT2TIME(FechaFinal);
RecMP."usuario maquina":=UsuarioMaquina;
RecMP."usuario alta":=usuarioAlta;
RecMP."Cód. incidencia":=cod_incidencia;
RecMP.Cantidad:=cantidad;
RecMP.Acaba:=acaba;
RecMP.Saneado:=Saneado;
RecMP.Reoperado:=TRUE;
RecMP.INSERT;
```

EmpezarRepasado(numPieza : Code[20];codUsuario : Code[20];usuarioOrdenador : Code[20];metros : Decimal;numItem : Code[20];Fechainicio :

```
RecHisN.INIT;
RecHisN.Pieza:=numPieza;
RecHisN."Usuario registro":=codUsuario;
RecHisN."Inicio Repasado":= Fechainicio;RecHisN.Fecha:=fecha;
RecHisN.Hora:=hora;
RecHisN."Usuario/ordenador":=usuarioOrdenador;
RecHisN.Metros:=metros;
RecHisN."Estado pieza":=4;
RecHisN."Item No.":=numItem;
RecHisN."Minutos Repasados":=MINUTOS;
RecHisN.INSERT(TRUE);

RecProduccion.RESET;
RecProduccion.SETRANGE(RecProduccion."Lot No.",numPieza);
```

```
    IF RecProduccion.FINDFIRST THEN BEGIN
        RecProduccion.Repasada:=TRUE;
        RecProduccion.MODIFY;
    END;

    RecHisN.RESET;
    RecHisN.SETRANGE(RecHisN.Pieza,numPieza);
    RecHisN.SETRANGE(RecHisN."Usuario registro",codUsuario);
    RecHisN.SETRANGE(RecHisN."Estado pieza",4);
    RecHisN.SETRANGE( RecHisN.Fecha,fecha);
    RecHisN.SETRANGE(RecHisN.Hora,hora);
    IF RecHisN.FINDLAST THEN BEGIN
        RecHisN."Fecha + hora":=FechaFin;
        RecHisN.MODIFY;
    END;

BorrarInci(Pieza : Code[10];Empieza : Decimal;"Fecha+Hora" : Code[17])

    RecMP.RESET;
    RecMP.SETRANGE(RecMP.Pieza,Pieza);
    RecMP.SETRANGE(RecMP.Empieza,Empieza);
    RecMP.SETRANGE(RecMP."Fecha+hora","Fecha+Hora");
    IF RecMP.FINDFIRST THEN BEGIN
        RecMP.DELETE;
    END;

CambiarCarro(nuevoCarro : Code[20];CarroActual : Code[20];pieza : Code[20]) ErrorTxt : Text[30]

    IF nuevoCarro<>'' THEN BEGIN
        RecP.RESET;
        RecP.SETRANGE(RecP."Nº carro",nuevoCarro);
            IF RecP.FINDFIRST THEN BEGIN
                varTipoCarro := RecP."Tipo carro.";
            END;
        lon:=STRLEN(nuevoCarro);
        RecP.RESET;
        RecP.SETRANGE(RecP."Lot No.",pieza);
        IF CarroActual<>'' THEN BEGIN
            RecP.SETRANGE(RecP."Nº carro",CarroActual);
        END;
        IF RecP.FINDFIRST THEN BEGIN
            IF (varTipoCarro = RecP."Tipo carro.") OR (varTipoCarro = '') OR (nuevoCarro = 'C001') OR
    (nuevoCarro = 'C002')  THEN BEGIN
                RecP2.RESET;
                RecP2.SETRANGE(RecP2."Item No.",RecP."Item No.");
                RecP2.SETRANGE(RecP2."Lot No.",RecP."Lot No.");
                IF RecP2.FINDFIRST THEN BEGIN
                    RecP2."Nº carro":=nuevoCarro;
                    RecP2."Nº Albaran Salida":=nuevoCarro+COPYSTR(RecP2."Nº Albaran Salida",lon+1);
                    RecP2.MODIFY;
                END;
```

```
            END
        ELSE ErrorTxt:='Este carro es de tipo '+ varTipoCarro;
        END;
    END;
```

FinRepasado(numPieza : Code[20];codUsuario : Code[20];usuarioOrdenador : Code[20];metros : Decimal;numItem : Code[20];Fechainicio : Dat

```
    RecHisN.INIT;
    RecHisN.Pieza:=numPieza;
    RecHisN."Usuario registro":=codUsuario;
    RecHisN."Inicio Repasado":= Fechainicio;
    RecHisN.Fecha:=fecha;
    RecHisN.Hora:=hora;
    MINUTOS:= ROUND ((FechaFin-Fechainicio) /60000,1);
    RecHisN."Usuario/ordenador":=usuarioOrdenador;
    RecHisN.Metros:=metros;
    RecHisN."Estado pieza":=4;
    RecHisN."Item No.":=numItem;
    RecHisN."Minutos Repasados":=MINUTOS;
    RecHisN.INSERT(TRUE);
    RecHisN.RESET;
    RecHisN.SETRANGE(RecHisN.Pieza,numPieza);
    RecHisN.SETRANGE(RecHisN."Usuario registro",codUsuario);
    RecHisN.SETRANGE(RecHisN."Estado pieza",4);
    RecHisN.SETRANGE( RecHisN.Fecha,fecha);
    RecHisN.SETRANGE(RecHisN.Hora,hora);
    IF RecHisN.FINDLAST THEN BEGIN
        RecHisN."Fecha + hora":=FechaFin;
        RecHisN.MODIFY;
    END;
```

Albaran(NCARRO : Code[10];npiezasencarro : Integer;USERID : Code[10];CodUsu : Code[10])

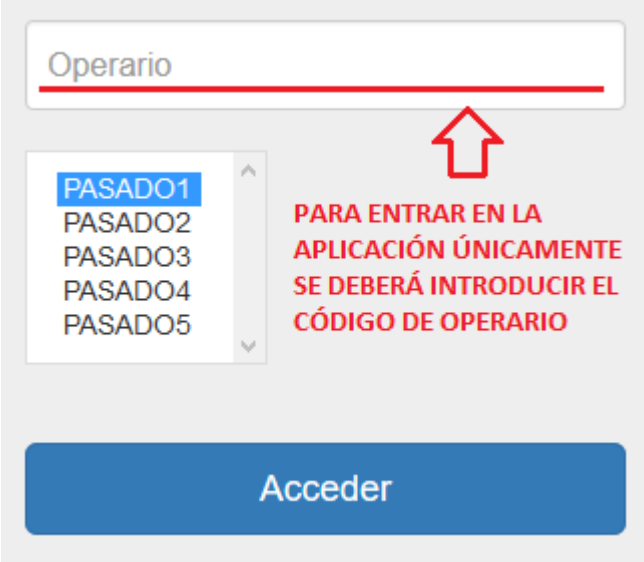
```
    IF NCARRO='' THEN BEGIN
        ERROR('Falta nº de carro.');
    END;
    IF npiezasencarro=0 THEN BEGIN
        ERROR('Este carro no tiene piezas.');
    END;
    IF npiezasencarro<>0 THEN BEGIN
        RecPro.RESET;
        RecPro.ASCENDING(FALSE);
        RecPro.SETCURRENTKEY("Nº carro",AcabExtPB,"Orden lectura en carro");
        RecPro.SETRANGE(RecPro."Nº carro",NCARRO);
        IF RecPro.FINDFIRST THEN REPEAT
            RecPro."Estado pieza":=5;
            RecPro."Nº Albaran Salida":=NCARRO+GUION+'01';
            RecPro."Carro finalizado":=TRUE;
            RecPro."Fecha cierre":=TODAY;
            RecPro."Hora cierre":=TIME;
```

```
        RecPro."Usuario cierre":=USERID;
        RecPro.MODIFY;

        RecHistorico.INIT;
        RecHistorico.Pieza:=RecPro."Lot No.";
        RecHistorico."Usuario registro":=CodUsu;
        RecHistorico.Fecha:=TODAY;
        RecHistorico.Hora:=TIME;
        RecHistorico."Usuario/ordenador":=USERID;
        RecHistorico.Metros:=RecPro.Metros;
        RecHistorico."Estado pieza":=5;
        RecHistorico."Item No.":=RecPro."Item No.";
        RecHistorico.Empresa:=RecPro.Empresa;
        RecHistorico.Telar:=RecPro.Telar;
        RecHistorico.INSERT(TRUE);

        RecHisNuevo.INIT;
        RecHisNuevo.Pieza:=RecPro."Lot No.";
        RecHisNuevo."Usuario registro":=CodUsu;
        RecHisNuevo.Fecha:=TODAY;
        RecHisNuevo.Hora:=TIME;
        RecHisNuevo."Usuario/ordenador":=USERID;
        RecHisNuevo.Metros:=RecPro.Metros;
        RecHisNuevo."Estado pieza":=5;
        RecHisNuevo."Item No.":=RecPro."Item No.";
        RecHisNuevo.Empresa:=RecPro.Empresa;
        RecHisNuevo.Telar:=RecPro.Telar;
        RecHisNuevo.INSERT(TRUE);
    UNTIL RecPro.NEXT = 0;
END;
```

9.4 MANUAL DE USUARIO REALIZADO PARA LOS OPERARIOS

CUALQUIER BOTÓN PULSADO UNA VEZ NO SE VOLVERÁ A PULSAR HASTA QUE LA ACCIÓN SE COMPLETE.

9.4.1 Usuario y máquina

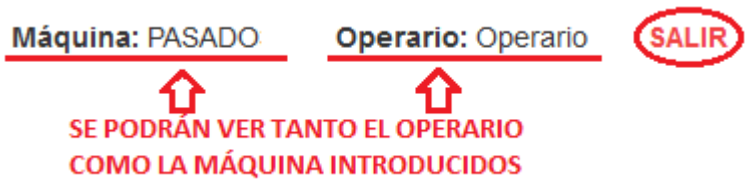


Una vez introducidos los datos, se debe presionar el botón de 'Acceder' o la tecla 'Intro' del teclado.

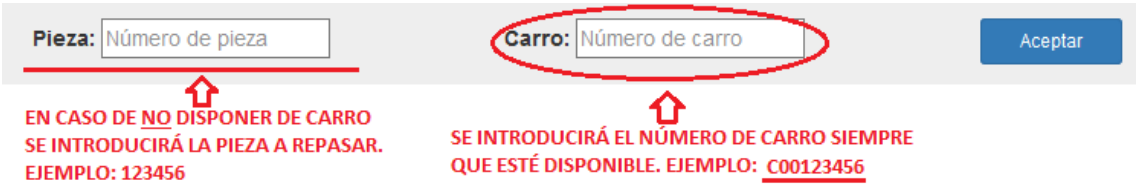
En caso de disponer de la tarjeta habrá que situarse sobre el cuadro de operario y pasar la tarjeta por el lector.

La máquina vendrá seleccionada de manera automática. Aunque se podrá modificar en caso de necesidad.

9.4.2 Pieza / Carro



Estarán siempre la máquina y el operario introducidos. En caso de querer cambiarlos habría que presionar el botón "SALIR".



- Los carros empiezan por "C" + "00" + seis números.
- Las piezas suelen tener seis números, aunque pueden contener al final tres números más.
- Una vez seleccionado el carro o la pieza se tendrá que pulsar el botón "Aceptar" o la tecla "Intro" del teclado.
- **NO RELLENAR LOS DOS CAMPOS "CARRO" Y "PIEZA", ÚNICAMENTE RELLENAR UNO DE ELLOS. EN EL CASO DE INTRODUCIRSE UN CARRO Y UNA PIEZA A LA VEZ PREDOMINARÁ EL CARRO.**

9.4.3 Empezar Repasado / Repasar

9.4.3.1 Carro

Al introducir un carro pueden salir piezas repasadas y por repasar:

Pieza por repasar, al pulsar el botón se iniciará el repasado desde el metro 0

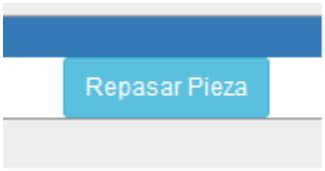
Botón azul: Pieza no repasada.

Pieza ya iniciada, el contador mantendra los metros anteriormente repasados

Botón verde: Pieza ya repasada al menos una vez.

9.4.3.2 Pieza

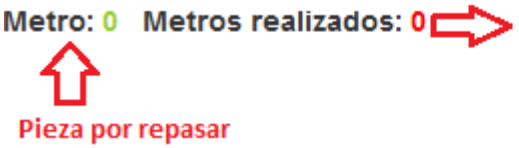
Al introducir una pieza únicamente aparecerá esa misma:



Al entrar por pieza siempre mantendrá el metraje anterior aunque este sea 0.

9.4.4 Repasado

9.4.4.1 CASO 1: Nuevo repasado

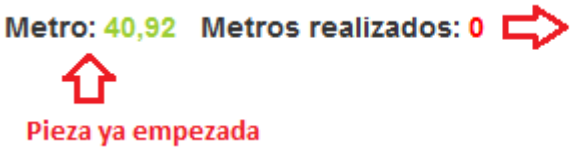


Aquí se irán sumando cada metro realizado en este repasado

En verde: Metros repasados totales de la pieza.

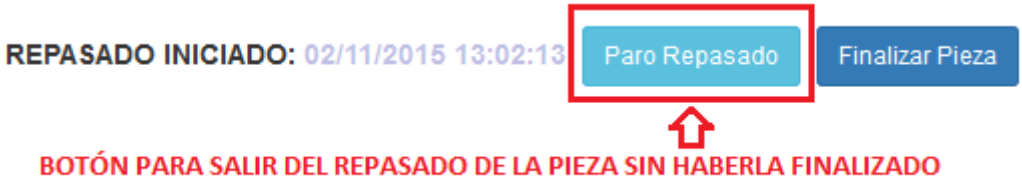
En rojo: Metros repasados actuales.

9.4.4.2 CASO 2: Continuar repasado



Aquí se irán sumando cada metro realizado en este repasado

9.4.4.3 Botón Paro Repasado

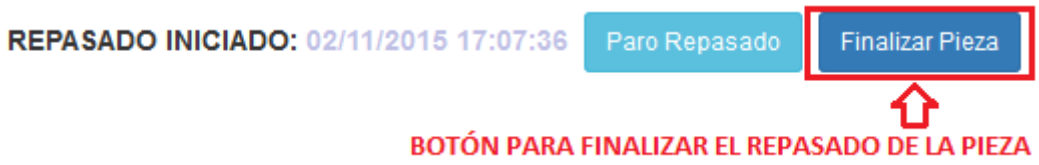


- 'Paro Repasado' únicamente deberá pulsarse en caso de que se quiera interrumpir el repasado de la pieza enviando así los datos pertinentes. Ejemplo: abandono momentáneo de la máquina

(necesidades, café, ayudar otro compañero…), acabar turno de trabajo sin haber finalizado la pieza.

- En caso de no haber enviado todas las incidencias pendientes de enviar no dejará parar o finalizar el repasado.

9.4.4.4 Botón Finalizar Pieza



- Pulsa únicamente en caso de haber finalizado la pieza. Al pulsarlo reiniciará el metraje.
- En caso de no haber enviado todas las incidencias pendientes de enviar no dejará parar o finalizar el repasado.

9.4.5 Añadir incidencias

9.4.5.1 Columnas

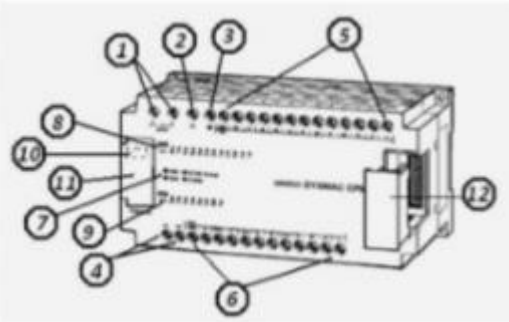
- **Máquina**: Máquina en la que se está realizando el repasado.
- **Usuario**: Código de usuario con el que se ha iniciado la aplicación.
- **Cód_inciencia**: Listado de incidencias posibles. Pulsar para seleccionar la incidencia deseada (por defecto: AGUAS (MOIRE)).
- **Cantidad**: Número de incidencias de un mismo tipo que se quieren enviar juntas. Casos:
  - Visualizando pocas: al localizar una incidencia enviarla (una tara por metro).
  - Visualizando muchas: al localizar más de una incidencia enviarlas al finalizar de verlas.

9.4.6 Ver incidencias / Imprimir albarán

- Una vez dentro de la ventana que muestra las incidencias se podrán imprimir pulsando el botón de ‘Imprimir Incidencias’ o volver a la página anterior pulsando el de ‘Volver’.
- El botón de ‘Imprimir Albarán’ únicamente imprimirá el albarán de las piezas ya repasadas que correspondan al carro. Esto hará un cambio de estado en todas las piezas repasadas.

9.5 CARACTERÍSTICAS PRINCIPALES DEL PLC CPM1 DE OMRON

Es importante conocer las características técnicas de los Controladores Lógicos Programables que publica el fabricante, pues de ello depende hacer una buena elección para una correcta aplicación, buscando siempre abatir costos. Descripción de los componentes:



1. Terminales de entrada de fuente de alimentación. Se conectan a la fuente de alimentación (100 a 240 Vca).
2. Terminales de tierra funcional. Se conecta a tierra para mejorar la inmunidad al ruido y reducir riesgo de descarga eléctrica.
3. Terminal de tierra de protección. Conectar a tierra para reducir riesgo de descarga eléctrica.
4. Terminales de salida de fuente de alimentación. Para alimentar los dispositivos de entrada.
5. Terminales de entrada. Se conectan los terminales de los dispositivos (interruptores o sensores) de entrada.
6. Terminales de salida. Se conectan los dispositivos de salida a controlar.
7. Indicadores de estado del PLC. Estos indicadores muestran el estado de operación del PLC:

Indicador	**Estado**	**Significado**
POWER (verde)	ON	Alimentación conectada al PLC
	OFF	Alimentación no conectada al PLC
RUN (verde)	ON	PLC en modo RUN o MONITOR
	OFF	PLC en modo PROGRAM o se ha producido un error fatal
ERROR/ALARM (rojo)	ON	Se ha producido un error fatal (el PLC para la operación)
	Destella	Se ha producido un error fatal (el PLC continúa la operación)
	OFF	Indica operación normal
COMM (naranja)	ON	Se están transfiriendo datos por el puerto de periféricos
	OFF	No se están transfiriendo datos por el puerto de periféricos

8. Indicadores de entrada. Estos indicadores se encienden cuando está en ON el correspondiente terminal de entrada.
9. Indicadores de salida. Estos indicadores se encienden cuando está en ON el correspondiente terminal de salida.
10. Controladores de potenciómetro analógico. Mediante ellos se seleccionan los contenidos de IR 250 y IR 251 a un valor entre 0 y 200.
11. Puerto de periféricos. Conecta el PLC a un periférico.
12. Conector de unidad de E/S de expansión. Conecta la CPU del PLC a una unidad de expansión de E/S para añadir 12 puertos de entrada y 8 puertos de salida.