2016

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Desarrollo de una web de consulta para una planta industrial cerámica.

Grado en Ingeniería de Telecomunicaciones, Sonido e Imagen.

Trabajo Fin de Grado presentado en la Universidad Politécnica de Valencia en el campus de Gandía, para la obtención del título Graduado en Ingeniería de Telecomunicaciones, Sonido e Imagen.

José Salvador Gascón
Escuela Politécnica Superior de Valencia
23/08/2016

RESUMEN

A menudo las empresas necesitan disponer de diferentes bases de datos para almacenar infinidad de datos en ellas, desde datos de clientes hasta registros generados automáticamente por máquinas de una cadena de montaje para su control.

El objetivo principal de este proyecto consta en la planificación y elaboración de un sitio web en el cual los encargados y operarios de la empresa VIDRES S.A. puedan consultar determinados parámetros de funcionamiento y estado de las máquinas de la propia fábrica, las cuales almacenan dichos parámetros en una base de datos. Todo esto sin necesidad de utilizar software específico de tratamiento de base de datos y facilitando la consulta desde cualquier sitio de forma inalámbrica con cualquier dispositivo que disponga de un navegador web, desde un Smartphone hasta un equipo de sobremesa.

RESUM

Sovint les empreses necessiten disposar de diferents bases de dades per a emmagatzemar infinitat de dades en elles, des de dades de clients fins a registres generats automàticament per màquines d'una cadena de muntatge per al seu control.

L'objectiu principal d'aquest projecte consta en la planificació i elaboració d'una web en la qual els encarregats i operaris de l'empresa VIDRES S.A. puguen consultar determinats paràmetres de funcionament i estat de les màquines de la pròpia fàbrica, les quals emmagatzemen aquests paràmetres en una base de dades. Tot açò sense necessitat d'utilitzar programari específic de tractament de base de dades i facilitant la consulta des de qualsevol lloc de forma sense fil amb qualsevol dispositiu que dispose d'un navegador web, des d'un smartphone fins a un equip de sobretaula.

ABSTRACT

Often companies need to have different databases to store plenty of data on them. Customer data, data automatically generated by machines.

The main objective of this project consists in planning and developing a website in which managers and workers of the company VIDRES S.A. they can see certain operating parameters and status of the factory machines, which store these parameters in a database. All this without using specific processing software database and facilitating consultation from anywhere wirelessly with any device that has a web browser from a Smartphone to a desktop computer.

CONTENIDO

Este proyecto surge en la empresa Vidres S.A de la necesidad de obtener cierta información (sobre todo en la planta de fabricación) de una manera rápida y de fácil acceso desde cualquier punto de la fábrica.

El objetivo inicial de la plataforma era el disponer de los datos de los molinos cerámicos que hay en planta y tener el control de los sacos almacenados sin la necesidad de ejecutar el software antes explicado (MBASE), dado que este necesita de un PC para su ejecución y no es algo práctico disponer de muchos PCs para poder consultar los datos.

En base a este problema surge la idea de crear una plataforma web con la que consultar la base de datos en la que tenemos esta información y vía web mostrar los registros necesarios, pudiendo acceder a estos desde un Smartphone, una Tablet o cualquier dispositivo que pueda llevar un operario encima con un navegador web y sea más práctico que un PC.

En todo momento se intenta simular lo máximo posible el aspecto y funcionamiento de MBASE para que la gente que tenga que utilizar la plataforma lo haga de una manera a la que ya están acostumbrados, haciendo así que no haya pérdida de tiempo para adaptarse a un nuevo entorno ni a nuevas funcionalidades.

1. INTRODUCCIÓN

1.1 VIDRES

Vidres es una empresa la cual se dedica a la fabricación de fritas, esmaltes y colores cerámicos, productos utilizados en la industria de la baldosa cerámica. La empresa fue fundada en 1975 en Villarreal, creada con el fin de ofrecer respuestas a las necesidades del sector cerámico mediante la innovación, el diseño, la tecnología, la calidad y el servicio. Desde Vidres se exportan sus productos a todos los continentes con el objetivo de acercar la tecnología cerámica a todos los rincones del mundo.

La empresa la podríamos dividir en 2 grandes partes, una seria la planta de producción donde se llevan a cabo la fabricación de los materiales y la elaboración del producto final y la otra gran parte seria la zona de oficinas, donde podemos encontrar los diferentes departamentos esenciales para el correcto funcionamiento de todo, tales como, diseño, administración, ventas, informática, laboratorio de investigación, etc.

1.2 M-BASE

M-Base es un programa desarrollado por y para Vidres por el equipo informático de la propia empresa. M-Base se lleva desarrollando desde hace 10 años y se ha ido adaptando a las necesidades que han ido surgiendo durante todo este tiempo.

El programa se utiliza para gestionar la mayoría de la empresa, desde las fichas y el control de personal hasta la consulta de parámetros de la planta de producción como por ejemplo el estado de los hornos, el control y situación de los sacos de material etc.

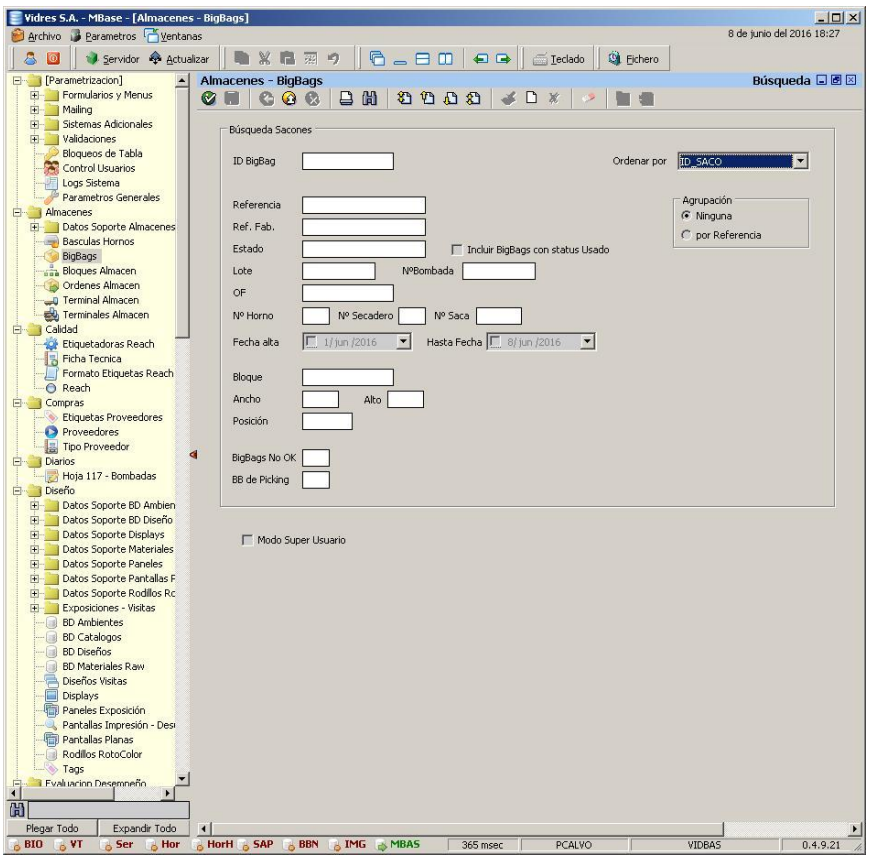


ILUSTRACIÓN 1. FORMULARIO BUSQUEDA DE SACOS EN MBASE

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVO PRINCIPAL

Como objetivo principal de este proyecto tenemos la planificación y el desarrollo de un sitio web en el plazo de 3 meses, utilizando los lenguajes de programación web HTML5, PHP, CSS3 y Javascript.

La función de este sitio es facilitar la consulta y el control de ciertos parámetros de la empresa desde cualquier dispositivo con un navegador web, ya sea un Smartphone, una Tablet, un ordenador, etc. sin la necesidad de tener un equipo con un software específico para la consulta de bases de datos. Para lograr esto se programará un sitio web desde cero, sin utilizar plantillas y totalmente personalizado a gusto de la empresa.

2.2 OBJETIVO SECUNDARIO

Dado que esta plataforma está enfocada al uso de cualquier persona de la empresa es necesario que esta tenga una interfaz intuitiva, sencilla y sobretodo funcional.

3. ESTADO DEL ARTE

3.1 AUTOMATIZACIÓN INDUSTRIAL

La Automatización Industrial es la aplicación de diferentes tecnologías para controlar y monitorear un proceso, maquina o dispositivo que por lo regular cumple funciones o tareas repetitivas, haciendo que opere automáticamente, reduciendo al mínimo la intervención humana.

Lo que se busca con la Automatización industrial es generar la mayor cantidad de producto, en el menor tiempo posible, con el fin de reducir los costos y garantizar una uniformidad en la calidad.

La Automatización Industrial la encontramos en muchos sectores de la economía, como en la Fabricación de Alimentos, Productos Farmacéuticos, Productos Químicos, en la Industria Gráfica, Petrolera, Automotriz, Plásticos, Telecomunicaciones entre otros, sectores en los cuales generan grandes beneficios. No solo se aplica a maquinas o fabricación de productos, también se aplica la gestión de procesos, de servicios, a manejo de la información, a mejorar cualquier proceso que con lleven a un desempeño más eficiente, desde la instalación, mantenimiento, diseño, contratación e incluso la comercialización.

En la automatización industrial existe un concepto fundamental llamado DCS (sistemas de control distribuido). Un DCS está formado por varios niveles de automatización:

-Nivel de campo: Donde se encuentran los sensores y actuadores.

-Nivel de control: Donde se encuentras los PLCs o las estaciones de automatización.

-Nivel de supervisión: Donde se encuentras las Estaciones de Operación y los Servidores de Proceso.

-Nivel MES: Donde se encuentran PCs con software especializados para la distribución de toda la información de planta así como la generación de reportes.

-Nivel ERP: Donde se encuentran igualmente PCs con software especializado para planificación y administración de la producción de toda la industria o empresa.

3.2 ¿APLICACIONES WEB O APLICACIÓN TRADICIONAL DE ESCRITORIO?

Últimamente las aplicaciones web están cogiendo mucha fuerza en el entorno empresarial (y no tan empresarial), dado a que en los últimos años han evolucionado mucho ofreciendo nuevas posibilidades interesantes. Esto tampoco quiere decir que las aplicaciones web sean la mejor solución en todos los casos frente a las aplicaciones de escritorio. Responder a la pregunta de si una aplicación web es mejor que una aplicación tradicional cliente/servidor no es posible hasta analizar la situación en la que se requiere y valorar los pros y contras de cada opción.

3.2.1 CARACTERÍSTICAS DE LA APLICACIÓN WEB:

-No requiere instalación de software específico en los clientes

Pro: Para acceder a una aplicación web simplemente es necesario disponer de un navegador (Opera, Firefox, Chrome, etc.) en el cual se muestra la interfaz de usuario con el área de trabajo, menús y demás, mientras que toda la compleja lógica se lleva a cabo del servidor.

-Bajo coste de actualización de los equipos

Pro: Dado que el servidor web suministra de forma centralizada las páginas a los navegadores que se conectan a él, cualquier actualización solo hay que aplicarla en el servidor para que todos los usuarios lo vean automáticamente.

-Todos con la misma versión y la más actualizada

Pro: Como consecuencia del punto anterior, los usuarios siempre están trabajando sobre la última versión que se le aplica al servidor, y puesto que todos trabajan sobre el mismo servidor no se da el caso de que 2 personas estén recibiendo una versión diferente del programa.

-Información centralizada

Pro: En las aplicaciones web no solo la lógica del negocio se encuentra en el servidor, sino también los datos que se ubican en una base de datos centralizada (ya sea en el mismo servidor o en uno diferente. La centralización tiene la ventaja de facilitar el acceso a la misma.

Contra: Tener la información centralizada supone un punto crítico en el sistema ya que si no se hacen copias de seguridad regularmente y se almacenan en un sitio diferente, una incidencia en la base de datos ocasionaría la perdida de toda la información de la empresa.

-Seguridad y copias de seguridad

Pro: Como consecuencia de tener los datos centralizados es más fácil establecer y llevar el control de una política de copias de seguridad centralizada. En el caso de que un puesto de trabajo que inutilizado por cualquier motivo la empresa no ha perdido información y puede desplegar rápidamente un nuevo puesto.

Contra: Si no se dispone de una buena seguridad se está ofreciendo todos los datos de la empresa en un mismo punto. Si alguien con malas intenciones consiguiera el acceso a este podría obtener toda la información de una misma tacada.

-Movilidad

Pro: Dependiendo de la implantación, si el software está ubicado en un servidor web en internet o bien disponemos de una intranet externalizada (extranet) cualquier usuario con un dispositivo conectado a internet podría acceder a la aplicación desde cualquier lugar a través de un navegador web (ordenador, Tablet, Smartphone, etc.).

Contra: Es verdad que hoy en día tener una conexión permanente a internet no es difícil y está a la orden del día (cable, wifi, 3g...) pero depender el 100% de una línea de datos no siempre es garantía de confianza ya que en un momento dado si nos quedamos sin conexión no podríamos trabajar.

-Reducción de costes en los puestos cliente

Pro: Debido a que las páginas se ofrecen desde el servidor web, este es quien suele ejecutar la mayoría de los procesos y la lógica del negocio, por lo tanto el equipo cliente no

necesita de un hardware potente, lo que se traduce en una reducción de costes y una mayor longevidad en el uso de los mismos puesto que estos no se quedan desfasados porque en un futuro se requieran operaciones más complejas.

Contra: Por otro lado ya que el servidor es el que se encarga de procesar prácticamente todo es necesario que este tenga un hardware lo suficientemente potente para soportar la carga de todo. También es conveniente tener equipos de reserva ya que al igual que la información, si este equipo queda inutilizado por cualquier motivo, en caso de no tener uno de reserva, todo el sistema quedaría paralizado.

3.2.2 CARACTERÍSTICAS DE APLICACIÓN TRADICIONAL DE ESCRITORIO

-Permite un mejor aprovechamiento del hardware equipo

Pro: Se aprovechan al máximo las prestaciones del equipo.

Contra: Si la aplicación exige un alto rendimiento del PC todos los equipos de la empresa deben estar a la altura con el gasto que eso conlleva.

-Menores tiempos de desarrollo

Pro: A menor tiempo de desarrollo, menor coste, lo que puede ser potencialmente más barato.

-Mejores tiempos de respuesta

Pro: Que todo se procese de manera local evita que se tenga que depender de una conexión entre el equipo y un servidor y la velocidad de la que esta disponga.

Contra: Si se trabaja contra una base de datos que está en un servidor al que se accede a través de la red tenemos el mismo inconveniente que la aplicación web.

-No se depende de una conexión a internet

Pro: No depender de una conexión a la red, que en ciertas ocasiones puede ser inexistente, no supone un problema a la hora de seguir trabajando.

Contra: Como en el punto anterior, si se trabaja sobre una base de datos ajena al equipo local sí que se dependería de una conexión a internet o por contrapartida montar una base de datos local para ir trabajando con la consecuencia de tener que volcar después los datos a la principal cuando se disponga de conexión.

-Si el software lo permite, puede trabajar también a través de internet

Pro: La ventaja de trabajar online no es exclusiva del software web. En caso de que el software esté preparado también ofrecería la posibilidad de trabajar a través de internet.

3.3 LAS BASES DE DATOS

Una base de datos es una herramienta para recopilar y organizar información. En las bases de datos, se puede almacenar información sobre personas, productos, pedidos o cualquier otra cosa. Muchas bases de datos empiezan siendo una lista en un programa de procesamiento de texto o en una hoja de cálculo. A medida que crece la lista, empiezan a aparecer repeticiones e inconsistencias en los datos. Cada vez resulta más complicado comprender los datos presentados en la lista y existen pocos métodos para buscar o recuperar subconjuntos de datos para revisarlos. Cuando empiezan a observarse estos problemas, es aconsejable transferir la información a una base de datos creada mediante un sistema de administración de bases de datos (DBMS).

Una base de datos computarizada es un contenedor de objetos. Una base de datos puede contener más de una tabla. Por ejemplo, un sistema de seguimiento de inventario que usa tres tablas no son tres bases de datos, sino una base de datos que contiene tres tablas.

3.3.1 LAS TABLAS EN LAS BASES DE DATOS

Una tabla de una base de datos es similar en apariencia a una hoja de cálculo, en cuanto a que los datos se almacenan en filas y columnas. Como consecuencia, normalmente es bastante fácil importar una hoja de cálculo en una tabla de una base de datos. La principal diferencia entre almacenar los datos en una hoja de cálculo y hacerlo en una base de datos es la forma de organizarse los datos.

Para lograr la máxima flexibilidad para una base de datos, la información tiene que estar organizada en tablas, para que no haya redundancias. Por ejemplo, si se almacena información sobre empleados, cada empleado se insertará una sola vez en una tabla que se configurará para contener únicamente datos de los empleados. Los datos sobre productos se almacenarán en su propia tabla, y los datos sobre sucursales también tendrán su tabla aparte. Este proceso se conoce como normalización.

Cada fila de una tabla se denomina registro. En los registros es donde se almacena cada información individual. Cada registro consta de campos (al menos uno). Los campos corresponden a las columnas de la tabla. Por ejemplo, puede trabajar con una tabla denominada "Empleados", en la que cada registro (fila) contiene información sobre un empleado distinto y cada campo (columna) contiene un tipo de información diferente, como el nombre, los apellidos, la dirección, o similares. Los campos se deben configurar con un determinado tipo de datos, ya sea texto, fecha, hora, numérico, o cualquier otro tipo.

Otra forma de describir registros y campos es imaginando un catálogo de fichas tradicional de una biblioteca. Cada ficha del armario corresponde a un registro de la base de datos. Cada información contenida en una ficha (autor, título, etc.) corresponde a un campo de la base de datos.

4. ANÁLISIS DE REQUERIMIENTOS

En este capítulo se va a hablar de la fase de análisis, una parte vital en todo proyecto de desarrollo de software ya que es donde se definen los requisitos mínimos necesarios que debe cumplir el sistema y en el cual podemos obtener una vista general de la arquitectura de este. Es en este punto donde se deben asentar las bases del proyecto desde las cuales se construirá todo lo demás.

El proceso a seguir empieza por realizar una buena definición de requisitos y en la elección de una metodología de desarrollo acorde con los lenguajes de programación que se van a utilizar y las necesidades finales del proyecto.

4.1 REQUISITOS FUNCIONALES

En este apartado se van a definir las interacciones que tendrán los usuarios finales con el software desarrollado.

4.1.1 CONTROL DE ACCESOS

Gestión de usuarios:

Esta plataforma no tendrá gestión de usuarios, por lo tanto las altas o bajas de los empleados las gestiona el encargado de recursos humanos o el jefe de informática desde el programa MBASE.

4.1.2 IDENTIFICACIÓN

1. La plataforma mostrará una pantalla de acceso en la cual el trabajador debe rellenar con su nombre y clave personal para acceder.
2. El sistema validará los datos contra una base de datos.
3. El sistema mostrará un mensaje de error en caso de que alguno de los datos no sea correcto. Tanto como si el nombre de usuario no existe como si la contraseña no es la correcta para un determinado usuario, se mostrara en ambos casos el mismo mensaje de error.
4. En caso de que los datos de acceso sean correctos, se iniciara sesión y el usuario accederá a la pantalla principal.

4.1.3 CIERRE DE SESIÓN

1. Si pasa media hora y no ha habido actividad con la plataforma, la sesión se cerrará automáticamente.
2. El usuario tendrá un botón siempre visible para cerrar la sesión en cualquier momento.
3. Una vez la sesión cerrada será redirigido a la página de identificación y sin volver a iniciar sesión no habrá forma de acceder al interior de la plataforma.

4.2 PLATAFORMA

4.2.1 PANTALLA DE IDENTIFICACIÓN

1. La página de inicio de la plataforma constará de un formulario con 2 campos para introducir el nombre de usuario y la contraseña y 2 botones, uno para validar los datos y otro para borrarlos.
2. En la parte inferior del formulario de acceso habrá un cuadro de búsqueda de Google para poder acceder al buscador de manera rápida sin tener que navegar a la página de este.
3. Debajo de la barra e búsqueda se colocaran unos accesos directos a otras webs de utilidad para los trabajadores, accesos los cuales serán comunes para todos ya que se encuentran antes de la identificación personal.

4.2.2 CABECERA

La cabecera la encontraremos en la parte superior, ocupando el 100% del ancho de la web y constara de:

1. Un logo de Vidres S.A.
2. Un botón de inicio para volver a la página principal.
3. Una foto de la cara del usuario conectado.
4. El nombre del usuario conectado.
5. Un botón para cerrar la sesión activa en cualquier momento.

4.2.3 MENÚ LATERAL

1. Estará situado en la parte izquierda.
2. Este se podrá ocultar en cualquier momento.
3. Habrá un menú de tipo árbol para acceder a las distintas zonas de la plataforma.
4. El menú de tipo árbol tendrá diferentes entradas según los permisos de cada uno asignados en la gestión de usuarios.
5. En el caso de que el trabajador no tenga acceso a ninguna entrada del menú lateral este no aparecerá.

4.2.4 ZONA DE CONTENIDOS

1. La zona central será donde se vaya mostrando toda la información.
2. En caso de que no se solicite ninguna información o se pulse el botón de inicio, aquí se mostraran accesos directos a otras plataformas útiles para el usuario (cada usuario tendrá sus accesos de forma personal controlados por el jefe informático).

4.3 REQUISITOS NO FUNCIONALES

4.3.1 INTERFAZ Y ESTILO

1. Lo fundamental que se va a buscar en la aplicación web es mostrar la máxima información posible de un vistazo.

2. Debe tener un aspecto minimalista, sencillo y ágil. Se debe tener presente que es una plataforma para uso interno de la empresa en el que prima la eficacia para sacar el máximo rendimiento de los trabajadores y se le resta prioridad al aspecto.

3. Se intentara simular lo máximo posible el funcionamiento de MBASE para que los trabajadores se encuentren con un entorno familiar en el que no haya pérdidas de tiempo por adaptación.

4. La web debe estar montada para que todos los contenidos se puedan variar de forma dinámica desde MBASE por un administrador informático.

5. La web tiene que ser totalmente funcional tanto en un PC de sobremesa con un monitor de 24" como en un Smartphone con una pantalla de 5"

4.3.2 SEGURIDAD

1. Para acceder a la plataforma se necesitara autenticación personal.

2. Solo será necesario autentificarse una vez, al no ser que por inactividad se cierre la sesión o la cierre el usuario por sí mismo.

3. La contraseña se codificará en md5 para enviarse a través de la red.

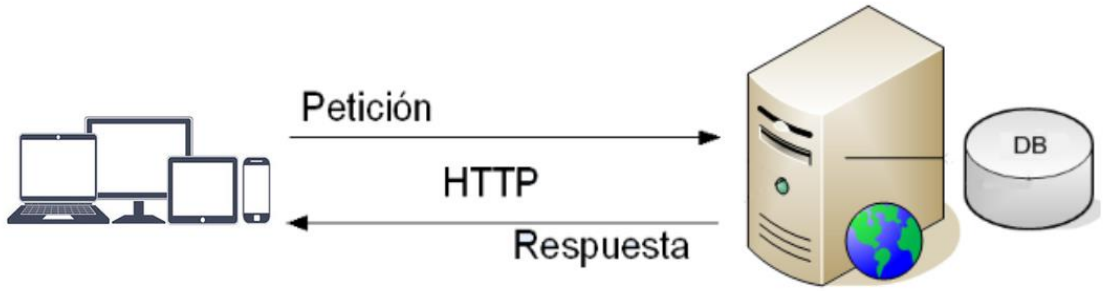
4. Se limitara el número de caracteres en los campos de texto.

5. DISEÑO

En este capítulo se va a describir todo el proceso de diseño de la plataforma. El diseño ofrece una idea completa sobre el software desarrollado en este proyecto.

5.1 ARQUITECTURA DE LA APLICACIÓN

La arquitectura de la plataforma se basa en el modelo cliente servidor. En este caso, y como puede verse en la siguiente figura, la plataforma web (consultada a través de cualquier cliente web) constituirá la parte cliente, y la parte servidor constara de la base de datos la cual estará alojada en un servidor al que se accederá mediante llamadas http.



El cliente es el que inicia las solicitudes y espera a que el servidor le responda con la información solicitada. En este caso (y como viene siendo normal) el usuario y el cliente interactúan a través de una interfaz gráfica, sencilla e intuitiva.

Por otro lado tenemos el servidor, el cual está esperando las solicitudes de los clientes, desempeñando un papel pasivo en la comunicación. Conforme este recibe las peticiones, procesa los datos y manda la información solicitada al cliente.

5.2 TECNOLOGÍAS UTILIZADAS

En este apartado se nombrarán las herramientas y tecnologías utilizadas en el proceso de desarrollo.

5.2.1 FIREBIRD

Firebird es un sistema de administración de base de datos relacional (o RDBMS) (Lenguaje consultas: SQL) de código abierto, basado en la versión 6 de Interbase.

En este proyecto no entra la gestión de la base de datos, sí que utilizaremos consultas SQL a través de PHP para recuperar los datos necesarios de esta.

5.2.2 NOTEPAD++

Notepad++ es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación. De soporte nativo a Microsoft Windows.

Se parece al Bloc de notas en cuanto al hecho de que puede editar texto sin formato y de forma simple. No obstante, incluye opciones más avanzadas que pueden ser útiles para usuarios avanzados como desarrolladores y programadores.

Se distribuye bajo los términos de la Licencia Pública General de GNU.

5.2.3 MOZILLA FIREFOX

Mozilla Firefox (llamado simplemente **Firefox**) es un navegador web libre y de código abierto desarrollado para Microsoft Windows, Android, OS X y GNU/Linux coordinado por la Corporación Mozilla y la Fundación Mozilla. Usa el motor Gecko para renderizar páginas web, el cual implementa actuales y futuros estándares web.

La herramienta de desarrollador "Inspeccionar elemento" que incorpora este navegador permite identificar con precisión el código HTML de todos los elementos que ves en una página web. El HTML y las hojas de estilo CSS que lo acompañan son completamente editables una vez que has abierto estas herramientas.

Con esta herramienta se puede experimentar haciendo los cambios que se quiera y luego actualizar la página para visualizar la página web con el aspecto deseado. Una herramienta muy útil que todo programador de páginas web utiliza.

5.2.4 ADOBE PHOTOSHOP CS6

Adobe Photoshop es un editor de gráficos rasterizados desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos, su nombre en español significa literalmente "taller de fotos". Es líder mundial del mercado de las aplicaciones de edición de imágenes y domina este sector de tal manera que su nombre es ampliamente empleado como sinónimo para la edición de imágenes en general.

5.2.5 HTML

HTML es el lenguaje que se emplea para el desarrollo de páginas de internet. Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. HTML dispone de etiquetas para imágenes, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc.

Se podría decir que HTML sirve para crear páginas web, darles estructura y contenido.

5.2.6 CSS3

CSS es un lenguaje de estilo que define la presentación de los documentos HTML (formatear el contenido previamente estructurado). Por ejemplo, CSS abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros temas.

5.2.7 PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Utilizado en servidores para gestionar llamadas http.

5.2.8 JAVASCRIPT

JavaScript es un lenguaje de programación, al igual que PHP, si bien tiene diferencias importantes con éste. JavaScript se utiliza principalmente del lado del cliente (es decir, se ejecuta en nuestro ordenador, no en el servidor) permitiendo crear efectos atractivos y dinámicos en las páginas web. Los navegadores modernos interpretan el código JavaScript integrado en las páginas web.

5.2.9 JQUERY

jQuery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Es la librería más utilizada

5.2.10 JQUERY DATATABLES

JQUERY DataTables es un plug-in para la biblioteca jQuery de JavaScript. Es una herramienta muy flexible que permite añadir controles avanzados de interacción a cualquier tabla realizada en HTML.

DataTables permite de forma muy cómoda la creación de tablas con paginación, búsqueda instantánea por columnas o por tabla, es compatible con cualquier fuente de datos (Ajax, procesamiento del lado del servidor, etc.), cuenta con una amplia variedad de extensiones para personalizar cada tabla y sacar el máximo rendimiento y un largo etcétera.

5.2.11 AJAX

Hasta ahora nos hemos centrado en JavaScript del lado del cliente. Pero las aplicaciones web o Apps para Smartphone tienen que conectar con servidores (donde normalmente residen los datos en bases de datos). Las peticiones a servidores se suelen hacer con lenguajes como PHP y el inconveniente que tienen son que ralentizan las páginas web.

Realizar peticiones al servidor y esperar respuesta puede consumir tiempo (el tiempo necesario para recargar una página completa). Para agilizar los desarrollos web surgió Ajax (inicialmente Asynchronous JavaScript And XML, aunque hoy día ya no es una tecnología ligada a XML con lo cual no pueden asociarse las siglas a estos términos), una tecnología que busca evitar las demoras propias de las peticiones y respuestas del servidor mediante la transmisión de datos en segundo plano usando un protocolo específicamente diseñado para la transmisión rápida de pequeños paquetes de datos.

Con Ajax, se hace posible realizar peticiones al servidor y obtener respuesta de este en segundo plano (sin necesidad de recargar la página web completa) y usar esos datos para, a través de JavaScript, modificar los contenidos de la página creando efectos dinámicos y rápidos.

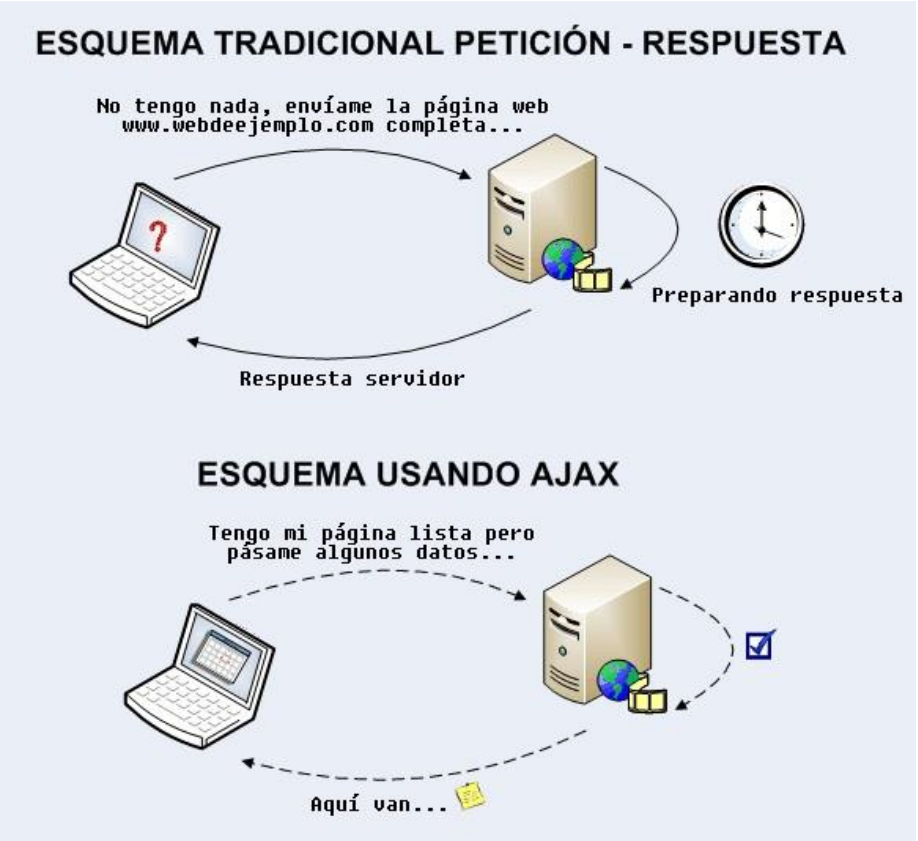


ILUSTRACIÓN 2. ESQUEMA FUNCIONAMIENTO AJAX

Las ventajas que proporciona Ajax son varias:

a) No es necesario recargar y redibujar la página web completa, con lo que todo es más rápido.

b) El usuario no percibe que haya demoras: está trabajando y al ser las comunicaciones en segundo plano no hay interrupciones.

c) Los pasos que antes podía ser necesario dar cargando varias páginas web pueden quedar condensados en una sola página que va cambiando gracias a Ajax y a la información recibida del servidor.

5.3 ORGANIZACIÓN DE PANTALLAS

Con este diagrama se pretende dar una visión de la estructura de la plataforma:

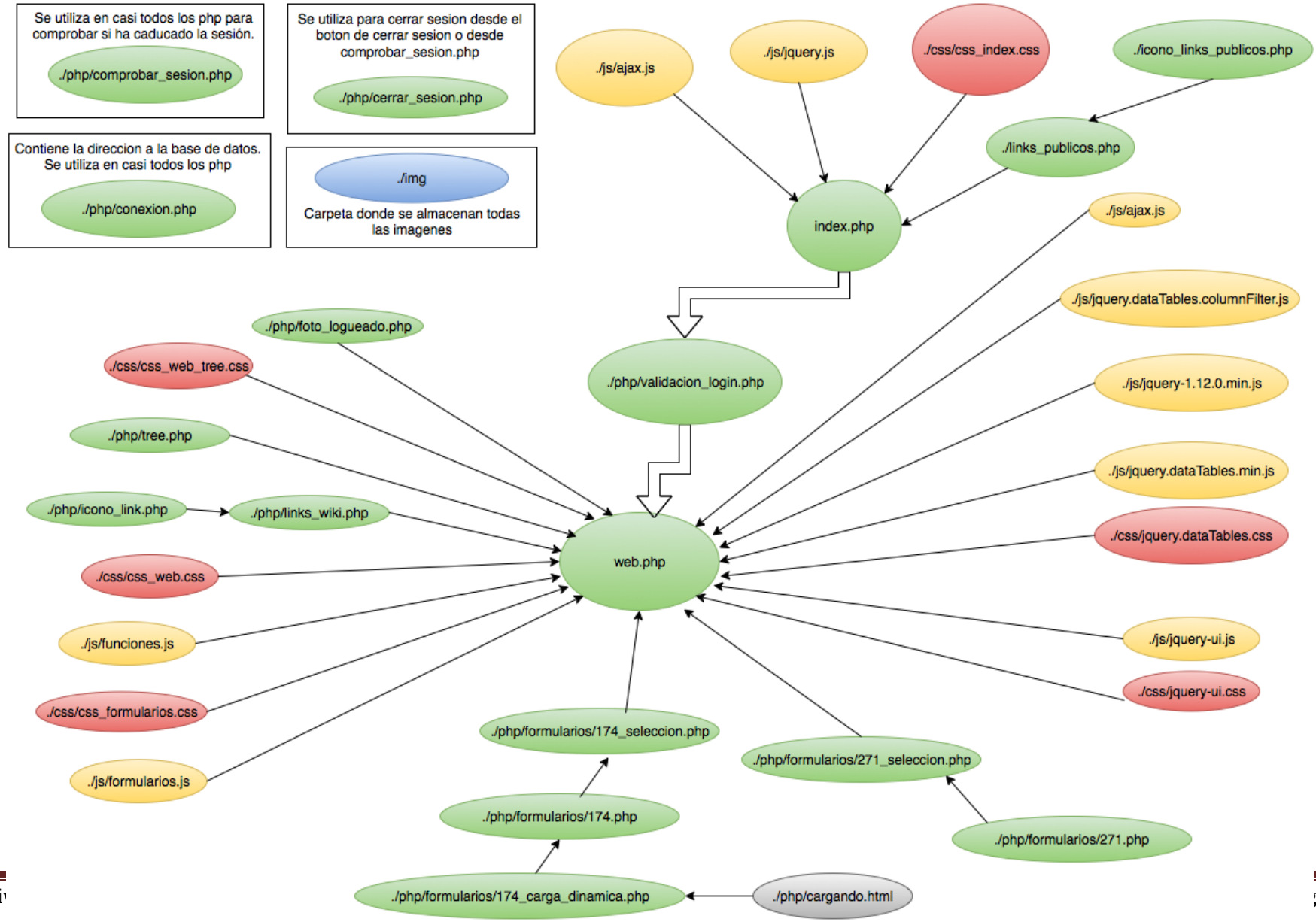


ILUSTRACIÓN 3. ESTRUCTURA DE LA CARPETA DEL SITIO WEB

En el esquema anterior podemos encontrar diferentes objetos:

-Ficheros ".php" los cuales contienen código PHP y HTML y se encargan de mostrar contenido generalmente de la base de datos.

-Ficheros ".css" los cuales contienen código CSS3. Son los ficheros encargados de contener los estilos y el diseño de las diferentes páginas.

-Ficheros ".js" los cuales contienen código de JavaScript. En estos ficheros podemos encontrar las funciones de JavaScript que se utilizan para cargar contenido dinámicamente o para realizar diferentes acciones y efectos en la web.

-La carpeta de imágenes que es donde se almacenan las imágenes que se utilizan en la plataforma (para tener una mayor organización en el servidor donde está alojada la web).

-Fichero ".html en el cual solo tenemos código HTML.

Como se ha comentado en el apartado de tecnologías utilizadas, para la realización de este sitio web se utiliza la tecnología AJAX, la cual permite cargar contenido dinámicamente sin tener que recargar la web completa, por lo tanto podríamos decir que tenemos 2 paginas diferentes sobre las que se va a cargar el resto de contenido. La primera es la de "index.php" y la segunda es "web.php".

5.4 DISEÑO DE LA INTERFAZ

Dado que esta plataforma se convertirá en la página de inicio de los empleados, en esta primera pantalla aparte del formulario de acceso a la plataforma, se decidió incluir en esta un acceso a los sitios que más se suelen utilizar así como una barra de búsqueda de un buscador típico para realizar búsquedas directamente sin tener que acudir a la página de este:

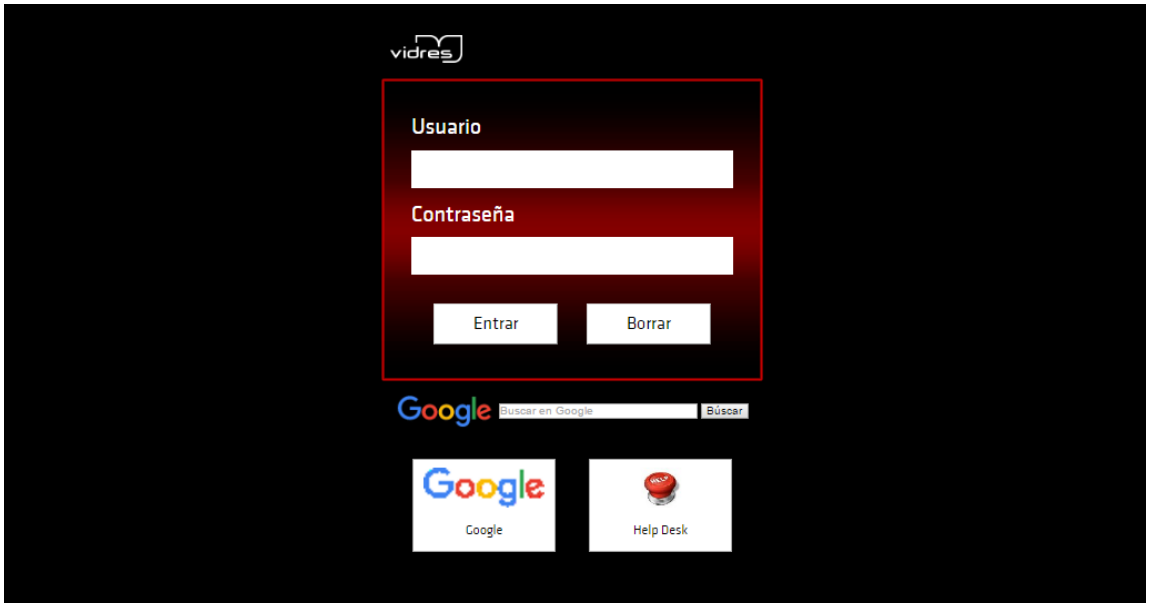


ILUSTRACIÓN 4. PANTALLA DE IDENTIFICACION

Habitualmente, en este tipo de plataformas, suele estar la opción de registrarse en el sistema creándose una cuenta personal para acceder, pero dado a que esto se desarrolla para un entorno industrial, no tenemos esta opción ya que las altas de los usuarios las hace el jefe de informática desde una aplicación ajena a la web (M-BASE).

Una vez realizada la identificación accedemos al interior de la plataforma, donde podemos encontrar un menú de tipo árbol a la izquierda y una lista de accesos a diferentes sitios en el centro de la pantalla:

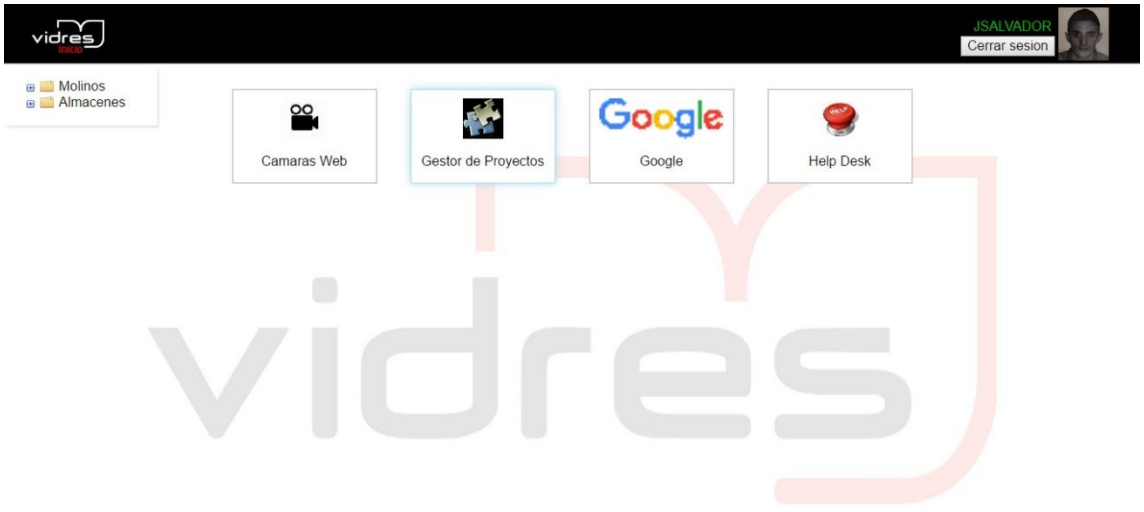


ILUSTRACIÓN 5. PANTALLA PRINCIPAL

Los contenidos de la pantalla principal son personales de cada usuario y es el jefe informático el que controla la visibilidad de estos. Estos accesos son links los cuales llevan a otras plataformas ajenas las cuales se abren en una pestaña aparte.

Las entradas que aparecen en el menú de tipo árbol de la izquierda son personales de cada usuario dependiendo de los accesos a los que este tenga permisos.

ILUSTRACIÓN 6. MENÚ LATERAL DESPLEGADO

Cada entrada del menú lateral lleva a uno o varios formularios los cuales se utilizan para obtener la información que se desea.

En el caso de "OPC Molinos" nos lleva a un formulario en el cual seleccionamos el molino del que deseamos obtener información:

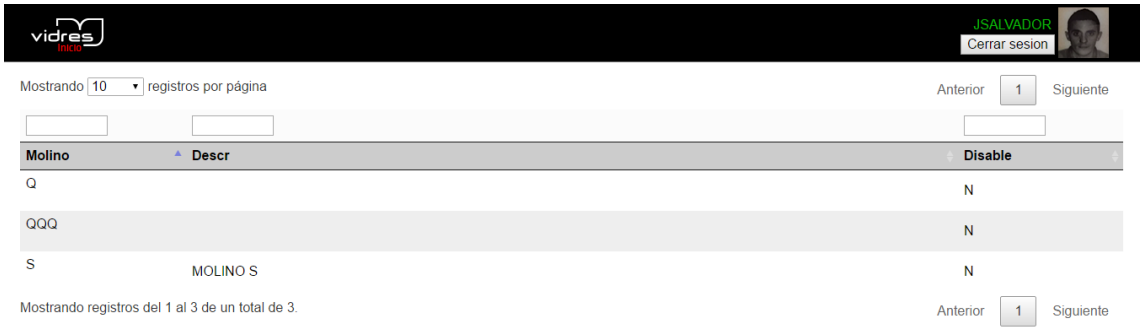


ILUSTRACIÓN 7. OPC MOLINOS: SELECCION DE MOLINO

Una vez seleccionado el molino se mostraran los cambios de eventos que han ocurrido en este:

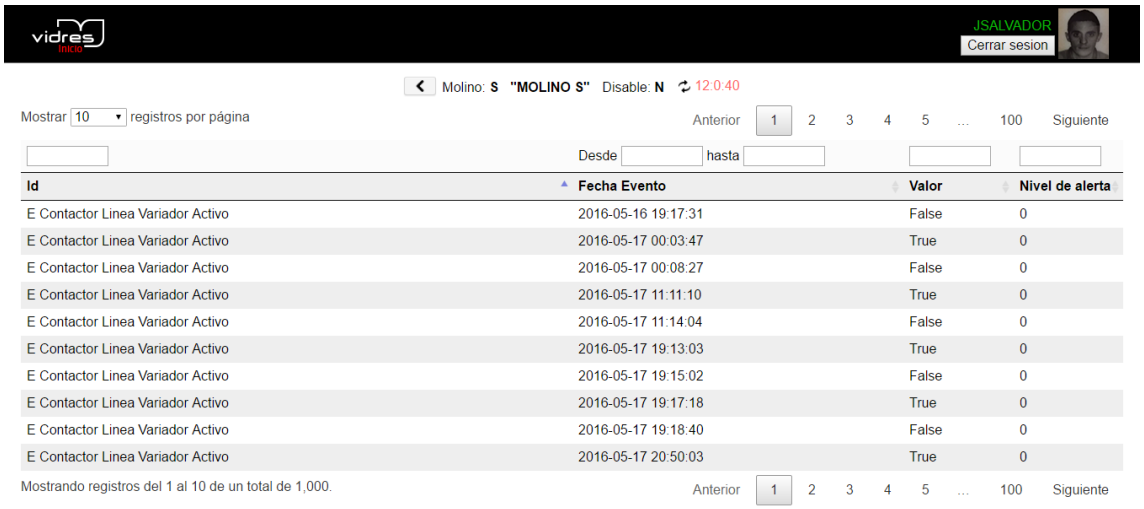


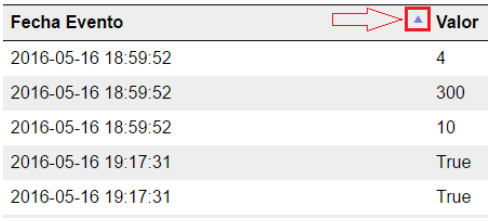
ILUSTRACIÓN 8. INFORMACIÓN DE MOLINO S

En las 2 ilustraciones anteriores podemos observar que las tablas con las que se trabajan disponen de la posibilidad de:

1. Repartir la información en páginas, con la posibilidad de moverte entre ellas pulsando el número de la página directamente, para que no aparezcan todos los datos de golpe saturando la pantalla con demasiada información:

ILUSTRACIÓN 9. PAGINACIÓN DE TABLA

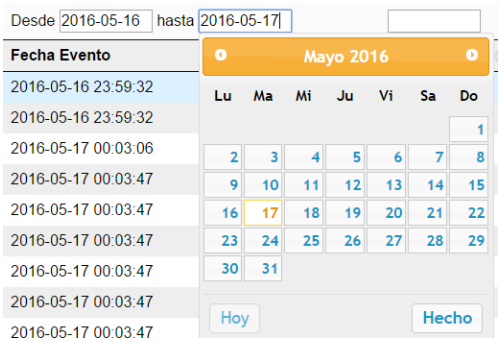
2. Ordenar los datos ascendente o descendentemente, por columna haciendo 'click' en la cabecera de esta:



Fecha Evento	Valor
2016-05-16 18:59:52	4
2016-05-16 18:59:52	300
2016-05-16 18:59:52	10
2016-05-16 19:17:31	True
2016-05-16 19:17:31	True

ILUSTRACIÓN 10. COLUMNAORDENADA POR FECHA DE EVENTO

3. Filtrar campo por texto o fecha:



4. Posibilidad de cambiar el número de registros por página:

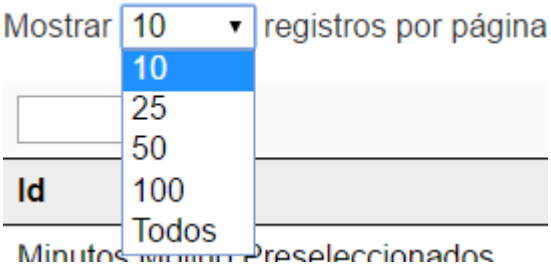


ILUSTRACIÓN 13. NÚMERO DE REGISTROS POR PÁGINA

Por otro lado tenemos la entrada 'BigBags', la cual nos lleva a un formulario donde podemos filtrar la búsqueda para visualizar el estado de los sacos donde se almacena el material fabricado:

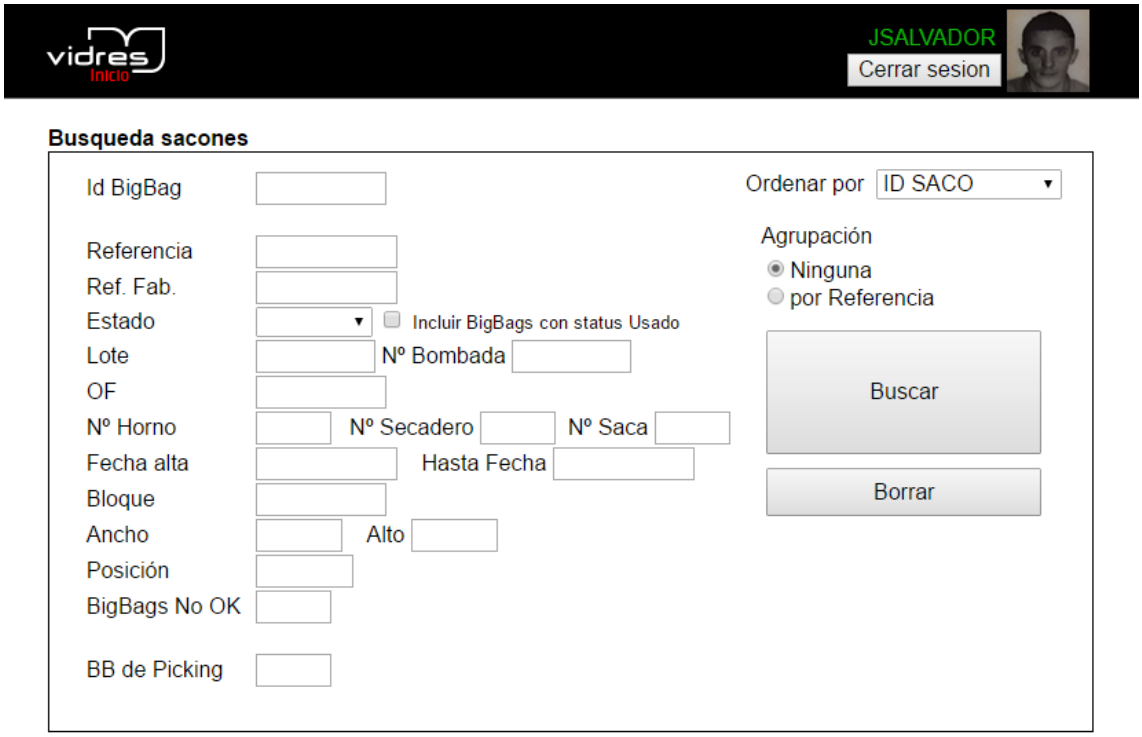
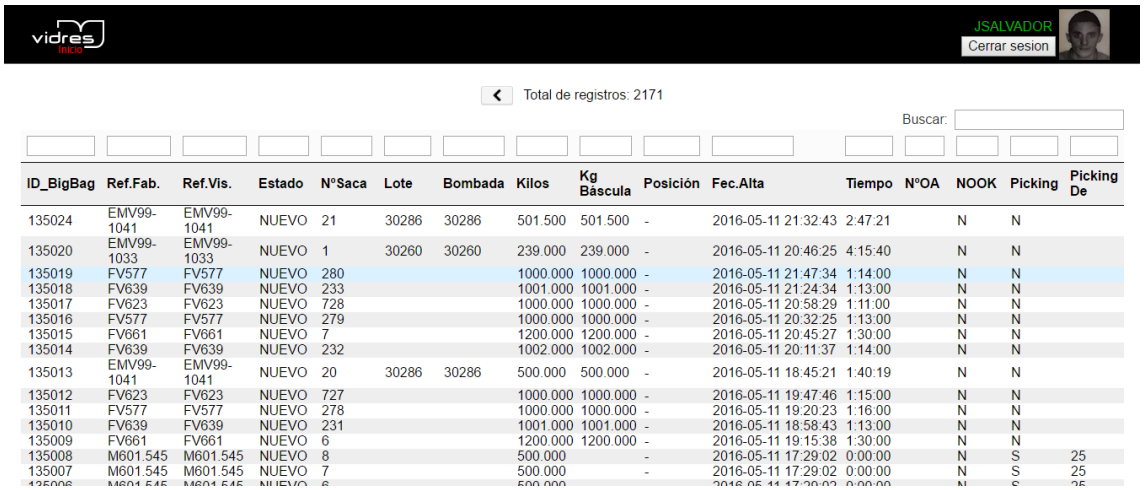


ILUSTRACIÓN 14. FORMULARIO BUSQUEDA DE SACONES

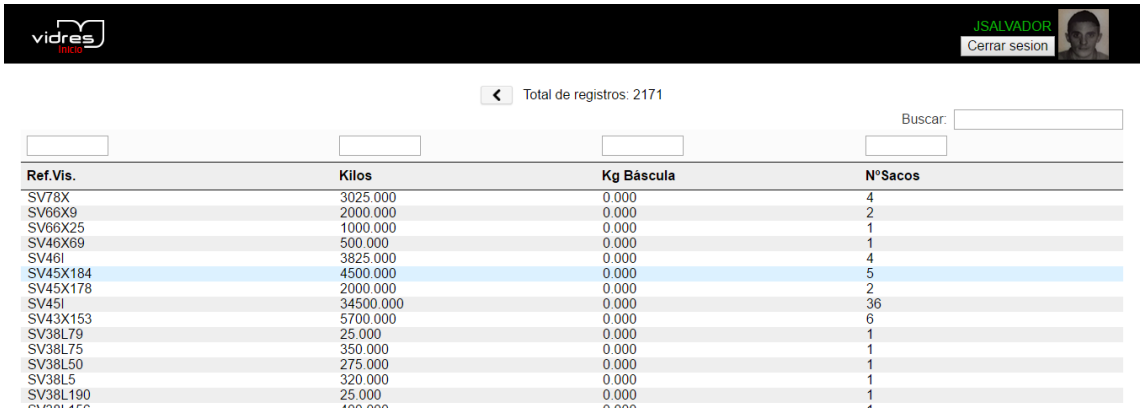
En este punto podemos filtrar rellenando los campos por los que se desee filtrar o se pueden dejar en blanco para mostrar todos los datos sin ningún tipo de filtro. Aquí los operarios pueden filtrar por alguno de los campos escribiendo texto, pueden seleccionar por qué campo se van a ordenar los sacos y/o si los quieren ver agrupados por referencia.

Cuando se pulsa el botón de buscar llegamos a la siguiente ventana, donde encontramos el resultado de nuestra búsqueda:



ID_BigBag	Ref.Fab.	Ref.Vis.	Estado	NºSaca	Lote	Bombada	Kilos	Kg Báscula	Posición	Fec.Alta	Tiempo	NºOA	NOOK	Picking	Picking De
135024	EMV99-1041	EMV99-1041	NUEVO	21	30286	30286	501.500	501.500	-	2016-05-11 21:32:43	2:47:21		N	N	
135020	EMV99-1033	EMV99-1033	NUEVO	1	30260	30260	239.000	239.000	-	2016-05-11 20:46:25	4:15:40		N	N	
135019	FV577	FV577	NUEVO	280			1000.000	1000.000	-	2016-05-11 21:47:34	1:14:00		N	N	
135018	FV639	FV639	NUEVO	233			1001.000	1001.000	-	2016-05-11 21:24:34	1:13:00		N	N	
135017	FV623	FV623	NUEVO	728			1000.000	1000.000	-	2016-05-11 20:58:29	1:11:00		N	N	
135016	FV577	FV577	NUEVO	279			1000.000	1000.000	-	2016-05-11 20:32:25	1:13:00		N	N	
135015	FV661	FV661	NUEVO	7			1200.000	1200.000	-	2016-05-11 20:45:27	1:30:00		N	N	
135014	FV639	FV639	NUEVO	232			1002.000	1002.000	-	2016-05-11 20:11:37	1:14:00		N	N	
135013	EMV99-1041	EMV99-1041	NUEVO	20	30286	30286	500.000	500.000	-	2016-05-11 18:45:21	1:40:19		N	N	
135012	FV623	FV623	NUEVO	727			1000.000	1000.000	-	2016-05-11 19:47:46	1:15:00		N	N	
135011	FV577	FV577	NUEVO	278			1000.000	1000.000	-	2016-05-11 19:20:23	1:16:00		N	N	
135010	FV639	FV639	NUEVO	231			1001.000	1001.000	-	2016-05-11 18:58:43	1:13:00		N	N	
135009	FV661	FV661	NUEVO	6			1200.000	1200.000	-	2016-05-11 19:15:38	1:30:00		N	N	
135008	M601.545	M601.545	NUEVO	8			500.000		-	2016-05-11 17:29:02	0:00:00		N	S	25
135007	M601.545	M601.545	NUEVO	7			500.000		-	2016-05-11 17:29:02	0:00:00		N	S	25

ILUSTRACIÓN 15. RESULTADO BUSQUEDA DE SACOS SIN AGRUPAMIENTO



Ref.Vis.	Kilos	Kg Báscula	NºSacos
SV78X	3025.000	0.000	4
SV66X9	2000.000	0.000	2
SV66X25	1000.000	0.000	1
SV46X69	500.000	0.000	1
SV46I	3825.000	0.000	4
SV45X184	4500.000	0.000	5
SV45X178	2000.000	0.000	2
SV45I	34500.000	0.000	36
SV43X153	5700.000	0.000	6
SV38L79	25.000	0.000	1
SV38L75	350.000	0.000	1
SV38L50	275.000	0.000	1
SV38L5	320.000	0.000	1
SV38L190	25.000	0.000	1

ILUSTRACIÓN 16. RESULTADO BUSQUEDA DE SACOS AGRUPADOS POR REFERENCIA VISUAL

Dado que se fabrican tantos sacos en la fábrica a lo largo del año y que el resultado de la búsqueda puede ser de decenas de miles, los resultados se van mostrando en grupos de 50. En caso de tener la necesidad de ver más, conforme el usuario se desplaza hacia abajo y llega al final de la lista, automáticamente se cargan 50 sacos más a la página, haciendo más rápida la carga de estos, sin saturar a la base de datos leyendo una cantidad innecesaria de datos.

5.5 BASE DE DATOS

5.5.1 VISTAS

En teoría de bases de datos, una vista es una consulta que se presenta como una tabla (virtual) a partir de un conjunto de tablas en una base de datos relacional.

Las vistas tienen la misma estructura que una tabla: filas y columnas. La única diferencia es que sólo se almacena de ellas la definición, no los datos. Los datos que se recuperan mediante una consulta a una vista se presentarán igual que los de una tabla.

Dado que durante la realización de este proyecto se trabajaba con una base de datos de una empresa real, en la cual no se podía permitir borrados accidentales ni fallos que pudieran ocasionar perdidas de datos importantes para la fábrica, se crearon vistas a partir de las tablas con permisos de solo lectura, ya que la idea inicial de la plataforma es la de poder consultar datos sin necesidad de modificar o escribir en ninguna tabla.

Las vistas necesarias para la plataforma son (la fila naranja muestra la clave primaria de la vista la cual no puede estar vacía ni repetirse en varios registros):

WEB_VUSERS	
ID_USER	Integer (Not null)
NOMBRE	Varchar(50)
PERFIL	Varchar(30)
LOGIN	Varchar(16)
MD5	Varchar(32)
DISABLED	Char(1)
FOTO	BLOB

TABLA 1. VISTA WEB_VUSERS

WEB_VUSERS: Vista en la que se almacena los datos de los usuarios existentes.

WEB_VPERMISOS	
ID_MENU	Integer (Not null)
CAPTION	Varchar(50)
DESCR	Varchar(100)
PARENT	Integer
WINCLASSNAME	Varchar(64)
PERFIL	Varchar(30)
ICON	BLOB
LINKHTML	Varchar(250)

TABLA 2. VISTA WEB_VPERMISOS

WEB_VPERMISOS: Se almacena los links que están en la página de login, los links que están en la página principal y los links a los formularios que están en el menú de la izquierda (comparte clave primaria con WEB_VGETFOLDERS).

WEB_VGETFOLDERS	
ID_MENU	Integer (Not null)
CAPTION	Varchar(50)
DESCR	Varchar(100)
PARENT	Integer
WINCLASSNAME	Varchar(64)

TABLA 3. VISTA WEB_VGETFOLDERS

WEB_VGETFOLDERS: Se almacenan las carpetas del menú tipo árbol de la izquierda (comparte clave primaria con WEB_VPERMISOS).

WEB_VMOLINOS	
MOLINO	Varchar(3) (Not null)
DESCR	Varchar(80)
DISABLE	Char(1)

TABLA 4. VISTA WEB_VMOLINOS

WEB_VMOLINOS: Se almacenan los molinos.

WEB_VMOLINOS_HIST	
MOLINO	Varchar(3)
ID	Varchar(250)(Not null)
FWHEN	Timestamp
LVL	Smallint
VALOR	Varchar(10)

TABLA 5. VISTA WEB_VMOLINOS_HIST

WEB_VMOLINO_HIST: Almacena información de los eventos de molinos.

WEB_VALM_SACOS	
ID_SACO	Bigint (Not null)
TIPO	Varchar(4) (Not null)
ID_STATUS	Integer
REF_VISUAL	Varchar(20)
REF_FABRIC	Varchar(20)
KG	Decimal(15,3)
KG_BASCULA	Decimal(15,3)
LOTE	Integer
NOF	Integer
BLOQUE	Varchar(10)
ANCHO	Varchar(3)
ALTO	Varchar(3)
POSICION	Smallint
PC	Varchar(16)
FV	Timestamp
PUM	Varchar(16)
FUM	Timestamp
NBOMBADA	Integer
FINICIO	Timestamp
FFIN	Timestamp
NHORNO	Smallint
NSECADERO	Smallint
NSACA	Integer
IN_NOA	Bigint
NOOK	Char(1)
OBS_NOOK	Varchar(80)
HOMOGENEA	Char(1)
ESCAMBIO	Char(1)
NCAMBIO	Smallint
CAMBIODE	Varchar(20)
HASPICKING	Char(1)
PICKINGOF	Smallint

TABLA 6. VISTA WEB_VALM_SACOS

WEB_VALM_SACOS: Contiene información de los sacos almacenados en el almacén.

5.6 DISEÑO DE SCRIPTS Y PROCEDIMIENTOS ALMACENADOS

Como ya se ha mencionado en el capítulo de análisis, la arquitectura de este proyecto se basa en el modelo cliente servidor. Este apartado da una primera perspectiva sobre la comunicación de la web con el servidor mediante archivos php.

Un fichero .php no necesariamente tiene que tener programada una o varias consultas a la base de datos, hay veces que simplemente se utilizan para poder utilizar variables de sesión u otro motivo. En este caso los ficheros que leen de la base de datos son:

links_publicos.php: Lee de la base de datos los links que aparecen en la página de identificación.

icono_links_publicos.php: Para obtener el icono de los links de la página de identificación.

links_wiki.php: Obtiene los links de la página principal según el perfil de cada usuario.

icono_link.php: Obtiene el icono de cada link de la página principal.

tree.php: Recupera las entradas del menú de la izquierda para elaborar el menú de tipo árbol dependiendo del perfil de cada uno.

foto_logueado.php: Obtiene la imagen de perfil del usuario conectado.

validación_login.php: Valida contra la base de datos si el usuario existe para permitir el acceso o no.

174.php: Lee los sacos de la vista WEB_VALM_SACOS teniendo en cuenta los filtros introducidos en 174_seleccion.php.

174_carga_dinamica.php: El fichero que va cargando sacos de manera dinámica de 50 en 50 conforme el usuario se desplaza hasta el final de la página en la consulta de sacos.

271_seleccion.php: Consulta la base de datos para poder mostrar los molinos que hay y poder seleccionar del cual se desea ver la información.

271.php: Es el fichero que se encarga de leer los datos de los eventos del molino seleccionado en 271_seleccion.php.

Tanto los ficheros .php como la base de datos, están localizados en uno de los servidores web dentro de la empresa, al cual se accederá mediante peticiones http.

5.7 SEGURIDAD

A pesar de ser una plataforma para el uso de trabajadores y la cual no se va a publicitar para gente ajena a la empresa, esta puede alcanzarse a través de internet, lo que supone tener un mínimo de seguridad para evitar robos de información no deseados o accesos a la base de datos inadecuados.

Para reforzar la seguridad, a la hora de realizar la validación de usuario, desde el cliente se envía al servidor la contraseña cifrada en md5, la cual es contrastada con la contraseña del usuario también almacenada con esta encriptación, evitando así en todo momento trabajar con la contraseña real sin encriptar.

Otra buena costumbre a la hora de programar una página web es la de comprobar lo que se escribe en los diferentes campos de texto de la web evitando así ataques de "sql injection", por ejemplo, si en la página donde se filtra la búsqueda de sacos, si el ID del saco solo puede ser numérico se evitará que en este tipo de campos se pueda escribir letras o símbolos.

Las variables de sesión se destruirán pasado un tiempo o al cerrar el navegador para evitar que otra persona acceda desde un ordenador donde otra persona haya accedido al sitio y se haya olvidado de cerrar su sesión al finalizar sus labores.

6 IMPLEMENTACIÓN

En este capítulo se va a hablar de la fase de implementación del proyecto. Aquí se explicaran algunos detalles relevantes así como las dificultades que han ido surgiendo en el proceso de programación.

Para programar el proyecto se ha utilizado el software Notepad++, del que ya se ha hablado antes y el navegador web Firefox con su herramienta de inspección de código con el cual se puede previsualizar código sobre una página web sin tener que escribirlo en el fichero final (tremendamente útil en la maquetación de la página web).

Durante el desarrollo han surgido algunas dificultades cuyas soluciones pueden ser de interés en el futuro.

6.1 ORGANIZACIÓN DE ARCHIVOS

El primer paso antes de ponerse a desarrollar fue la creación de una estructura de carpetas para tener una mejor organización. Para el desarrollo del proyecto se van a utilizar ficheros de diferentes formatos (*.php, *.js, *.html, etc.) y tener una buena organización puede ser vital a la hora de navegar luego entre los ficheros en busca de alguna errata o intentando implementar alguna nueva función.

Partiendo de la carpeta raíz se crea la siguiente estructura:

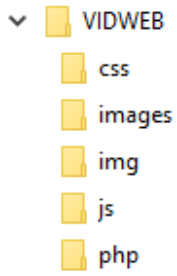


ILUSTRACIÓN 17. ESTRUCTURA INICIAL CARPETAS

Una vez la estructura hecha, se puede comenzar a crear los ficheros y las subcarpetas que se crean convenientes conforme avanza el proceso.

6.2 PROGRAMACIÓN

A partir de aquí se va a hablar de la programación de los puntos más importantes y de los problemas que surgieron en el proceso.

6.2.1 LOGIN - VALIDACIÓN

Lo primero que se programó fue la pantalla de control de accesos. Para realizar esta pantalla, inicialmente se crearon 2 campos de entrada para el usuario y la contraseña y 2 botones, uno para validar los datos y otro para limpiar los datos de ambos campos, hasta ahí sin mucha dificultad.

Una vez el usuario introduce su usuario y su contraseña, hay cierta información que se guarda en lo que se conocen como las variables de sesión. Una variable de sesión es una

variable normal con la virtud de que no va a ser destruida en el momento que se cambia de página, si no que mientras el usuario no cierre esa sesión (la sesión se destruya), en cualquier parte del sitio se podrá hacer uso de ella. Por ejemplo para tener en todo momento el nombre del usuario conectado en una variable, o la hora de cuando inició sesión, etc.

En este caso, si la validación es correcta se guardan todas estas variables de sesión:

```
$_SESSION['ID_USER']=$fila->ID_USER;
$_SESSION['NOMBRE']=$fila->NOMBRE;
$_SESSION['DPTO']=$fila->DPTO;
$_SESSION['FBAJA']=$fila->FBAJA;
$_SESSION['PERFIL']=$fila->PERFIL;
$_SESSION['LOGIN']=$fila->LOGIN;
$_SESSION['MD5']=$fila->MD5;
$_SESSION['DISABLED']=$fila->DISABLED;
$_SESSION['FOTO']=$fila->FOTO;
$_SESSION['GRAPHTYPE']=$fila->GRAPHTYPE;
$_SESSION['LASTLOGINOK']=$fila->LASTLOGINOK;
$_SESSION['LASTLOGINERROR']=$fila->LASTLOGINERROR;
$_SESSION['BADLOGINTRIES']=$fila->BADLOGINTRIES;
$_SESSION['PASWDATE']=$fila->PASWDATE;
```

ILUSTRACIÓN 18. VARIABLES DE SESIÓN DE UN USUARIO LOGUEADO

Por ejemplo, en la página principal, el nombre del usuario conectado se lee de la variable $_SESSION['NOMBRE'].

6.2.2 LOGIN - APARIENCIA

Siendo que el objetivo del proyecto es que fuera la página principal de todo trabajador de la empresa, se optó por poner una barra de búsqueda en la cual al realizar una búsqueda esta se haga a través del buscador de Google sin la necesidad de navegar hasta la página de este. El problema se solucionó con el siguiente código proporcionado por la misma Google (simplemente copiar, pegar y situar donde se desee):

```
<center>
    <FORM method=GET action="http://www.google.com/search">
    <TABLE bgcolor="#000000" style="margin-top: 10px;"><tr><td>
    <A HREF=" http://www.google.com/"><IMG SRC="./img/google.jpg" border="0" ALT="Google"
    align="absmiddle" style="height:40px;width:auto;"></A>
    <INPUT TYPE=text name=q size=31 maxlength=255 value="" placeholder="Buscar en Google"
    id="buscador_google" />
    <INPUT TYPE=hidden name=hl value=es />
    <INPUT type=submit name=btnG VALUE="Búscar" />
    </td></tr></TABLE>
    </FORM>
</center>
```

ILUSTRACIÓN 19. CÓDIGO PROPORCIONADO POR GOOGLE

Quedando así en la página:

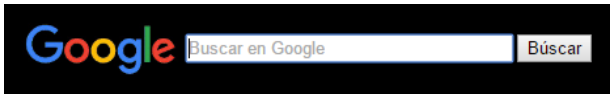


ILUSTRACIÓN 20. BARRA BUSQUEDA GOOGLE

Cualquier búsqueda realizada en ese recuadro es como si se hiciera en la misma página www.google.es y como la mayoría de veces los trabajadores accederán al navegador con la intención de realizar alguna búsqueda por internet, para evitar que una vez cargada la

página tenga que ir al recuadro y pinchar para escribir en él, se añadió una función que automáticamente pone el foco del teclado sobre el:

```
<script>
    function cursor_google(){
        document.getElementById('buscador_google').focus();
    }
</script>
```

ILUSTRACIÓN 21. FUNCION JAVASCRIPT

Esta función está escrita en Javascript y simplemente contiene una línea que apunta al elemento con id='buscador_google' y le pone el foco encima. En la primera imagen de código se puede observar una de las etiquetas <INPUT> la cual contiene "id='buscador_google'".

Para poner más a mano las cosas, también se decidió poner bajo de la barra de búsqueda, accesos directos a sitios que los trabajadores usen de forma continuada. Puesto que este contenido debe ser dinámico para poder asignar nuevos links sin tener que editar el código fuente de la web, se optó por añadir los links a la base de datos (en la vista WEB_VPERMISOS) y leerlos cada vez que se actualiza la página:

```
echo '<a href="'.$fila->LINKHTML .'" target="_blank"><div class="box_link_wiki">';


    echo '<div class="icono_link_wiki"><img
    src="./icono_links_publicos.php?link_icono='.$fila->LINKHTML.'"
    height="48px"></img></div>';
    echo '<div class="link_link_wiki">'. $fila->CAPTION .'</div>';
echo '</div></a>';
```

ILUSTRACIÓN 22. CODIGO LINKS LOGIN

Con este código se lee el link del campo 'LINKHTML', se añade el nombre del link leído del campo 'CAPTION' y se dibuja el icono llamando al archivo 'icono_links_publicos.php' pasándole como parámetro el link del icono que se desea:

```
$query="select ICON from WEB_VPERMISOS where LINKHTML='$link_icono'";
$res=ibase_query($con,$query);
$image_icon=ibase_fetch_object($res);
ibase_blob_echo($image_icon->ICON);
```

ILUSTRACIÓN 23. CODIGO PARA LEER ICONO DEL LINK

Quedando finalmente con esta apariencia:

ILUSTRACIÓN 24. 2 LINKS EN PAGINA LOGIN

6.2.3 PÁGINA PRINCIPAL

Una vez validados en el sistema llegamos a la página principal. En este punto se busca realizar una interfaz simple, ágil y eficaz. Un requisito que se exigía era que la zona donde se va a cargar la información relevante fuera amplia y que las cosas como la foto de perfil, el botón de cerrar la sesión actual o el botón de inicio para volver atrás no ocuparan mucho espacio más del necesario.

La solución a esto fue:

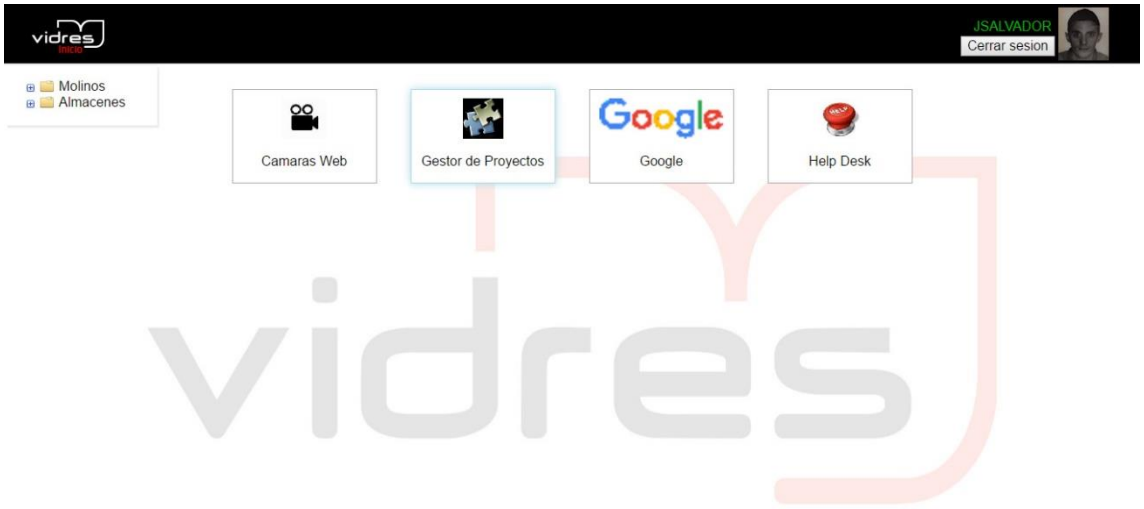


ILUSTRACIÓN 25. PÁGINA PRINCIPAL DE LA WEB

Una cabecera estrecha con la información justa y necesaria, una zona central amplia para cargar el máximo de información posible y un menú lateral desplegable que se oculta en cuanto se entra en alguna sección para dejar aún más hueco en pantalla.

En la parte superior, la cabecera, tenemos el logo de la empresa, en el cual se aprovechó para poner el botón de inicio y no añadir más botones por la página y a la parte derecha el nombre del usuario conectado, una foto personal y un botón para cerrar la sesión.

Aunque a simple vista no se aprecie, la cabecera está compuesta por una tabla con 4 columnas, la de la izquierda de tamaño fijo con el logo, la del centro de tamaño variable para adaptarse al tamaño de la página, y en el lado derecho las otras 2 columnas, una con el nombre de usuario y el botón de cerrar sesión y la otra con la foto de perfil.

```
<header id="cabecera">
    <table style="width:100%;">
        <tr style="height:68px;">
            <td style="width:120px">                                    Columna 1
                <div id="margen_logo"></div>
                <div id="logo" onClick="cargaren('./php/links_wiki.php',
                '#folio');"></div>
            </td>
            <td>                                                        Columna 2
                <div id="tip_of_the_day"><div style="text-align: center;position
                relative;top: 50%;-ms-transform:
                translateY(-50%);-webkit-transform: translateY(-50%);transform:
                translateY(-50%); height:100%"> <?php echo $tod ?></div>
            </td>
            <td style="height: 62px; width: 130px; overflow: auto; text-align:
            right;">
                    <div id="datos_perfil">
                        <div id="nombre_usuario_perfil"><?php echo $nom_usu; ?>
                        </div>
                        <div id="cerrar_sesion_perfil"><a href=
                        "./php/cerrar_sesion.php"><input type="button" value=
                        "Cerrar sesion"></input></a></div>
                    </div>                                              Columna 3
            </td>
            <td style="width:94px">
                <div id="imagen_perfil"><img src="php/foto_logueado.php" class=
                "perfil"></img></div>
            </td>                                                       Columna 4
        </tr>
    </table>
</header>
```

ILUSTRACIÓN 26. CODIGO DE LA CABECERA

Para crear una tabla en HTML primero hay que abrir la etiqueta <table>, esto indica que a partir de aquí va a hacer una tabla, cada fila de la tabla se crea con las etiquetas <tr></tr> y entre estas 2 se añaden las columnas que se deseen siendo cada etiqueta <td></td> una columna.

Una tabla de 2 filas y 2 columnas seria:

```
<table>
        <tr> //Esto indica una nueva fila
                <td>Información en columna 1 de la fila 1</td>
                <td> Información en columna 2 de la fila 1</td>
        </tr>
        <tr> //Inicio de la segunda fila
                <td> Información en columna 1 de la fila 2</td>
                <td> Información en columna 2 de la fila 2</td>
        </tr>
</table>
```

Hay una nueva moda de no utilizar tablas para la maquetación de una página web, pero aunque haya otras maneras de poder hacerlo, hay ocasiones en las que el uso de estas es lo más eficaz.

En el logo se puede apreciar el atributo onClick, el cual llama a la función 'cargaren()'.

```
<div id="logo" onClick="cargaren('./php/links_wiki.php',
```

ILUSTRACIÓN 27. LLAMADA A FUNCIÓN CARGAREN()

Esta función es con diferencia la más utilizada en todo el código:

```
function cargaren(URLfichero, id_div_destino){

    $.ajax({
        url: URLfichero,
        success:function(data){
            $(id_div_destino).html(data);

        }
    })

}
```

ILUSTRACIÓN 28. CODIGO DE LA FUNCION CARGAREN()

Como se ha comentado en puntos anteriores, en esta web se utiliza tecnología Ajax y gracias a la biblioteca de Javascript llamada JQUERY esto se convierte en una tarea muy sencilla, todo se reduce al código que tenemos en la imagen anterior. En este caso se medió el código en una función llamada 'cargaren' a la cual se le pasan 2 atributos, el fichero de PHP/HTML que se quiere cargar y el contenedor donde se quiere cargar el contenido, de esta manera pasándole diferentes parámetros a esa función se puede cargar todas las páginas sin tener que escribir cada vez el código de dentro de la función.

6.2.4 MENU LATERAL

En la página principal también disponemos de un menú lateral izquierdo de tipo árbol el cual nos da acceso a los diferentes formularios a los que tenemos acceso. En el desarrollo de esta sección se encontró el problema que más tiempo llevo resolver.

Dado que todo el contenido se lee a partir de una base de datos, el menú y las entradas de este se leen de la vista WEB_VPERMISOS y debemos tener en cuenta varios puntos:

1. Los links que se utilizan en la página principal y en la página de autentificación también se guardan en esta vista.
2. El usuario solo debe ver la entrada de los formularios a los que tenga acceso en base a su perfil.
3. Pueden haber usuarios que no tengan permiso de acceso a ningún formulario, en este caso no se debe crear el menú tipo árbol.
4. En la vista WEB_VGETFOLDERS se guardan las carpetas padres y en la vista WEB_VPERMISOS las entradas a los formularios, hay que relacionar las carpetas padres con las hijas y diferenciar los diferentes niveles para dibujarlos.
5. Las entradas a las que el usuario no tiene permisos no deben verse.
6. Las carpetas que no contengan ningún formulario dentro al que el usuario tenga acceso no deben verse.

Una vez conocidos los puntos importantes a tener en cuenta lo primero que se hace son las consultas para obtener los datos deseados cumpliendo los requisitos de los puntos 1, 2 y 3.

```
$query2="select * from WEB_VPERMISOS where PERFIL='$perfil_login'  and LINKHTML
is null";
$res2=ibase_query($con,$query2);

$query3="select * from WEB_VGETFOLDERS;";
$res3=ibase_query($con,$query3);
```

ILUSTRACIÓN 29. 2 CONSULTAS SQL PARA OBTENER DATOS

Con la primera consulta leemos los campos de la vista que LINKHTMK es 'null', esto quiere que no es un link y que es una entrada de formulario. También tenemos la comprobación del perfil, para que solo se muestren las entradas a las que este usuario tiene acceso.

Una vez hecho esto guardamos los campos que nos interesan en un vector, primero guardamos las entradas que nos ha devuelto la primera sentencia SQL de la imagen anterior y en la segunda línea almacenamos todas las carpetas en el mismo vector:

```
while($fila1 = ibase_fetch_object($res2)){

    $array2[$fila1->ID_MENU]= array("ID_MENU" => $fila1->ID_MENU, "CAPTION"
    => $fila1->CAPTION, "PARENT" => $fila1->PARENT, "WINCLASSNAME" => $fila1
    ->WINCLASSNAME);
    $_SESSION['crea_tree_view']=1;

}

while($fila2 = ibase_fetch_object($res3)){

    $array2[$fila2->ID_MENU] = array("ID_MENU" => $fila2->ID_MENU, "CAPTION"
     => $fila2->CAPTION, "PARENT" => $fila2->PARENT, "WINCLASSNAME" => "");

}
```

ILUSTRACIÓN 30. ALMACENAMIENTO DE INFORMACIÓN DE SQL EN ARRAYS

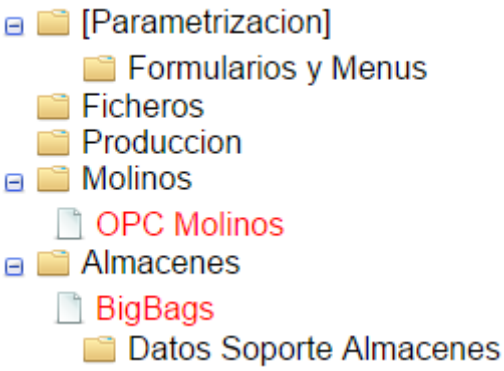
Con la variable $_SESSION['crea_tree_view'] después comprobaremos si hay que crear el menú lateral o no. Como se ha comentado antes, en caso de que la consulta no haya devuelto ningún valor es porque el usuario no tiene permisos y no debe de crearse el menú.

El siguiente paso es crear una función recursiva para dibujar el menú en la web (Se dice que una función es recursiva cuando el cuerpo de la función utiliza a la propia función.):

```
function createTreeView($array, $currentParent, $currLevel = 0, $prevLevel = -1) {
    foreach ($array as $categoryId => $category) {
        if ($currentParent == $category['PARENT']) {

            if ($currLevel > $prevLevel){
                echo " <ol> ";
            }
            if ($currLevel == $prevLevel){
                echo " </li> ";
            }

            //este if es para que pinte el formulario de una forma y las carpetas
            de otra

            if($category['WINCLASSNAME'] == ""){
                echo '<li class="visible" > <label>'.utf8_encode($category["CAPTION"
                ]).'</label> <input type="checkbox" name="subfolder2"/>';
            }
            else{
                echo '<li id="form'.$category["ID_MENU"].'"
                onClick="cargaren(\'./php/formularios/'.$category["ID_MENU"].
                '_seleccion.php\', \'#folio\');ocultar_menu();"> <a
                class="form_submenu" href="#" >'.$category["CAPTION"].'</a> <input
                type="checkbox" name="subfolder2"/>';
            }

            if ($currLevel > $prevLevel) {
                $prevLevel = $currLevel;
            }

            $currLevel++;
            createTreeView ($array, $categoryId, $currLevel, $prevLevel);
            $currLevel--;
        }
    }
        if ($currLevel == $prevLevel){
            echo " </li>  </ol> ";
        }
}
```

ILUSTRACIÓN 31. FUNCION RECURSIVA PARA CREAR EL MENÚ

Llegado a este punto el resultado es este:



Lo único que nos falta para para cumplir con los 6 puntos anteriores es que no se vean las carpetas que no contienen ninguna entrada a formulario, en el caso de mi usuario solo deberían verse las carpetas "MOLINOS" y "ALMACENES" ya que son las únicas que contienen dentro un formulario al menos. Después de probar muchas cosas y tras mucho fracaso al final la idea que tuvo éxito fue la de utilizar Javascript para conseguir el efecto deseado.

El código del menú de árbol de la imagen anterior es el siguiente:

```
▼<div id="general_submenu">
  ▼<div id="content" class="general-style1">
    ▼<ol>
      ▶<li>…</li>
      ▶<li>…</li>
      ▶<li>…</li>
      ▼<li>
          <label>Molinos</label>
          <input type="checkbox" name="subfolder2">
        ▼<ol>
          ▼<li id="form271" onclick="cargaren('./php/
          formularios/271_seleccion.php', '#folio');
          ocultar_menu();">
              <a class="form_submenu" href="#">OPC
              Molinos</a>
              <input type="checkbox" name="subfolder2">
            </li>
          </ol>
        </li>
      ▼<li>
          <label>Almacenes</label>
          <input type="checkbox" name="subfolder2">
        ▼<ol>
          ▼<li id="form174" onclick="cargaren('./php/
          formularios/174_seleccion.php', '#folio');
          ocultar_menu();">
              <a class="form_submenu" href=
              "#">BigBags</a>
              <input type="checkbox" name="subfolder2">
            </li>
          ▶<li>…</li>
          </ol>
        </li>
      </ol>
    </div>
  </div>
```

ILUSTRACIÓN 32. CODIGO MENU ARBOL V.1

En HTML las etiquetas <ol> i <li> se utilizan para crear listas ordenadas. La etiqueta <ol> indica que eso es una lista y la etiqueta <li> indica que es un elemento dentro de la lista. Existe la posibilidad de anidar listas metiendo una etiqueta <ol> dentro de una etiqueta <li>, ejemplo:

```
<ol> //Inicio de una lista
        <li>Esto es un elemento</li>
        <li>
                <ol> //Esto sería una lista dentro de un elemento
                        <li> Esto es un elemento dentro de una lista anidada</li>
                </ol>
        </li>
</ol>
```

De esta manera conseguimos el efecto de tabulación que se ve en la imagen del menú de árbol. También podemos observar que toda la lista está dentro de un elemento , cosa que se aprovechara para solucionar el problema de las carpetas vacías.

Con javascript existe la posibilidad de ver cuál es la etiqueta padre de otra, partiendo de esta base podemos observar que el elemento donde tenemos el link al formulario tiene una etiqueta llamada class="form_submenu":

```
<a class="form_submenu" href="#">OPC Molinos</a>
<a class="form_submenu" href="#">BigBags</a>
```

Entonces la idea que surgió fue la de crear el menú entero pero oculto, añadiéndole a las etiquetas <li> una clase con el atributo "display:none", una vez el menú creado pero sin estar a la vista con las 2 funciones siguientes de javascript, buscamos el elemento que tenga el atributo class="form_submenu" (que solo son las entradas a los formularios que el usuario debería de poder ver) y desde esa etiqueta, ir subiendo niveles, hasta llegar a la primera etiqueta DIV que se encuentre, e ir quitando el atributo de invisible de estas:

```
function findUpTag(el, tag) {
  while (el.parentNode) {

        if(el.className != 'visible'){
            el.className = 'visible';
        }

        el = el.parentNode;
        if (el.tagName === tag)
            break;
    }
}


function limpia_menu(){

    selector = document.querySelectorAll('.form_submenu');

    for(elemento in selector) {

        var el = selector[elemento];
        findUpTag(el, "DIV");

    }
}
```

ILUSTRACIÓN 33. FUNCIÓN PARA MOSTRAR EL MENU LATERAL

Una vez cargada la página, en el cuerpo de esta llamamos la función limpia_menu, a la cual se le dice que la clase que se quiere buscar es '.form_submenu' y se hace un bucle para que

llame a la segunda función (la que hace que los elementos se vuelvan visibles) tantas veces como elementos con la clase deseada hayan.

Mientras no se encuentre que la etiqueta padre sea un <DIV>, la función ira subiendo niveles hasta que alcance la etiqueta padre <DIV>. Los resultados fueron:

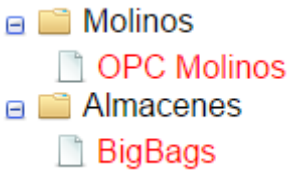


ILUSTRACIÓN 34. MENÚ FUNCIONANDO COMO SE DESEA

6.2.5 PLUG-IN DATATABLES

6.2.5.1 DATATABLES

Otro aspecto interesante para comentar es el de las tablas. Otro requisito que se puso conforme evolucionaba el proyecto fue la necesidad de enriquecer las tablas de datos. Recuperar los datos de una base de datos y mostrarlo en una tabla es una tarea relativamente sencilla, el problema es cuando con la respuesta de la base de datos se hace una tabla la cual contiene 4500 filas.

Llegados a este punto lo primero que se planteó fue la idea de añadir paginación a las tablas. Se estuvo investigando diferentes opciones para conseguir esto y después de algunas pruebas se llegó a la conclusión de que la mejor manera era utilizando DataTables, un plug-in para la biblioteca jQuery de Javascript el cual es una herramienta muy flexible que permite añadir controles avanzados de interacción a cualquier tabla HTML.

Con esta herramienta, enriquecer una tabla se hace mucho más sencillo. A groso modo se puede decir que para utilizar este plug-in se necesita que la tabla tenga un formato especial y una función en Javascript que será la que añada los controles avanzados a la tabla.

Antes ya se ha hablado del cómo se hace una tabla en HTML, partiendo de aquí la única diferencia es que hay que separar la cabeza de la tabla del cuerpo añadiendo las etiquetas <thead></thead> y <tbody></tbody>.

```
<table>
    <tr>
        <td>Cabecera 1</td>
        <td>Cabecera 2</td>
    </tr>
    <tr>
        <td>Elemento 1-1</td>
        <td>Elemento 1-2</td>
    </tr>
    <tr>
        <td>Elemento 2-1</td>
        <td>Elemento 2-2</td>
    </tr>
</table>
```

ILUSTRACIÓN 34. TABLA NORMAL

```
<table>
    <thead>
        <tr>
            <td>Cabecera 1</td>
            <td>Cabecera 2</td>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Elemento 1-1</td>
            <td>Elemento 1-2</td>
        </tr>
        <tr>
            <td>Elemento 2-1</td>
            <td>Elemento 2-2</td>
        </tr>
    </tbody>
</table>
```

ILUSTRACIÓN 35. TABLA FORMATO DATATABLES

Lo primero es construir la tabla leyendo los datos de la base de datos:

```
include '../conexion.php';
$query="select FIRST 1000 MOLINO, ID, FWHEN, VALOR, coalesce(LVL, 0) as LVL from WEB_VMOLINOS_HIST";
$res=ibase_query($con,$query);
```

ILUSTRACIÓN 35. CONSULTA A BASE DE DATOS

Guardamos los datos en un vector de datos:

```
while($fila_result = ibase_fetch_object($res)){

        $array_molinos[$cuenta]= array("MOLINO" => $fila_result->MOLINO, "ID" => $fila_result->ID, "FWHEN" =>
        $fila_result->FWHEN, "VALOR" => $fila_result->VALOR, "LVL" => $fila_result->LVL);
        $cuenta++;
}
```

ILUSTRACIÓN 36. GUARDANDO DATOS DE CONSULTA EN UN VECTOR

Una vez con los datos guardados construimos la tabla. Para ello creamos la cabecera de la tabla con los nombres de las columnas escritos a mano. Luego se hace un bucle para leer todos los datos del vector y por cada ciclo creamos una fila de la tabla nueva:

```
<table id="datos_formulario" align="center">
    <thead>
        <tr style="cursor:pointer;">
            <td></td>
            <td></td>
            <td></td>
            <th></th>
        </tr>
        <tr style="cursor:pointer;">
            <th><b>Id</b></th>
            <th><b>Fecha Evento</b></th>
            <th><b>Valor</b></th>
            <th><b>Nivel de alerta</b></th>
        </tr>

    </thead>
    <tbody>

    <?php

    foreach ($array_molinos as $molinos) {
            if($molinos["MOLINO"]==$letra_molino){
                if($molinos["LVL"]>=20 && $molinos["LVL"]<=29){
                        echo '<tr class="alerta_3_molinos">';
                    }else if ($molinos["LVL"]>=10 && $molinos["LVL"]<=19){
                        echo '<tr class="alerta_2_molinos">';
                    }else{
                        echo '<tr>';
                    }
                        echo '<td>'.$molinos["ID"].'</td>';
                        echo '<td>'.$molinos["FWHEN"].'</td>';
                        echo '<td>'.$molinos["VALOR"].'</td>';
                        echo '<td>'.$molinos["LVL"].'</td>';
                    echo '</tr>';
            }
        }
    ?>
    </tbody>
</table>
```

ILUSTRACIÓN 37. CONSTRUCCIÓN DE UNA TABLA

Lo que tenemos hasta este punto es una tabla, con 1000 datos seguidos y sin ninguna posibilidad de filtrado, búsqueda ni paginación:

Molino: S "MOLINO S" Disable: N 15:17:36

Id	Fecha Evento	Valor	Nivel de alerta
Minutos Molino Preseleccionados	2016-05-16 18:59:52	4	0
Preseleccion Agua Dosificada	2016-05-16 18:59:52	10	0
S Activacion Contactor Linea Molino	2016-05-16 18:59:52	300	0
E Contactor Linea Variador Activo	2016-05-16 19:17:31	False	0
E Fallo Presion Aceite Reductor Molino	2016-05-16 19:17:31	False	0
E Fallo Variador Molino	2016-05-16 19:17:31	True	0
E Puerta 1 Abajo Abierta	2016-05-16 19:17:31	True	0
E Puerta 1 Arriba Abierta	2016-05-16 19:17:31	True	0
E Puerta 2 Arriba Abierta	2016-05-16 19:17:31	True	0
E Puerta 2 Abajo Abierta	2016-05-16 19:17:31	True	0
E Pulsador Marcha Molino	2016-05-16 19:17:31	False	0
E Pulsador Molino Posicion de Carga	2016-05-16 19:17:31	False	0
E Pulsador Molino Posicion Descarga	2016-05-16 19:17:31	False	0
E Pulsador Paro Molino	2016-05-16 19:17:31	True	0
E Pulsador Rearme Fallos PLC	2016-05-16 19:17:31	False	0
E Pulsos de Agua	2016-05-16 19:17:31	False	0
E Reserva E19	2016-05-16 19:17:31	False	0
E Reserva E20	2016-05-16 19:17:31	False	0
E Reserva E21	2016-05-16 19:17:31	False	0
E Reserva E22	2016-05-16 19:17:31	False	0
E Reserva E23	2016-05-16 19:17:31	False	0
E Sensor Posicion Molino	2016-05-16 19:17:31	False	0
E Sensor Reduccion Velocidad Molino	2016-05-16 19:17:31	False	0
E Sistema Automatico	2016-05-16 19:17:31	True	0
E Sistema Manual	2016-05-16 19:17:31	False	0
E Termico Enfriamiento Aceite Molino	2016-05-16 19:17:31	False	0
E Termico Ventilador Molino	2016-05-16 19:17:31	False	0
Enclavamiento Alarma Contactor de Linea	2016-05-16 19:17:31	True	0
Enclavamiento Alarma Fallo Variador	2016-05-16 19:17:31	False	0
Enclavamiento Alarma No Cuenta Litros de Agua en la Dosificacion	2016-05-16 19:17:31	False	0
Enclavamiento Alarma No Funciona Dosificacion Agua Molino	2016-05-16 19:17:31	False	0
Enclavamiento Alarma No Gira Molino	2016-05-16 19:17:31	False	0
Enclavamiento Alarma Presion Aceite	2016-05-16 19:17:31	False	0
Enclavamiento Alarma Puerta 1 Carga	2016-05-16 19:17:31	False	0
Enclavamiento Alarma Puerta 1 Descarga	2016-05-16 19:17:31	False	0

ILUSTRACIÓN 38. TABLA SENCILLA

Llegados a este punto, se necesita una función en Javascript en la que se ejecuta el código que convierte esta tabla plana en una tabla enriquecida. El código necesario se puede encontrar en la página del desarrollador del plug-in con mucha variedad de ejemplos explicativos sobre cómo funciona. En este caso el código necesario ha sido el siguiente:

```
$('#datos_formulario').dataTable(
{
    "paging": true,
    "ordering": true,
    "autoWidth": false,
    "info": true,
    "searching": true,
    "columns": [
        { "width": "50%" },
        { "width": "30%" },
        { "width": "10%" },
        { "width": "10%" }
    ],
    "lengthMenu": [[10, 25, 50, 100, -1], [10, 25, 50, 100, "Todos"]],
    "pagingType": "simple_numbers",
    "responsive": true,
    "dom": '<"top"lp<"clear">>rt<"bottom"ip<"clear">>',
    "language": {
        "sSearch":         "Buscar:",
        "sLoadingRecords": "Cargando...",
        "sZeroRecords":    "No se encontraron resultados",
        "sEmptyTable":     "Ningún dato disponible en esta tabla",
        "lengthMenu": "Mostrar _MENU_ registros por página",
        "zeroRecords": "No hay datos",
        "oPaginate": {
            "sFirst":    "Primero",
            "sLast":     "Último",
            "sNext":     "Siguiente",
            "sPrevious": "Anterior"},
        "oAria": {
            "sSortAscending":  ": Activar para ordenar la columna de manera ascendente",
            "sSortDescending": ": Activar para ordenar la columna de manera descendente"},
        "info": "Página _PAGE_ de _PAGES_",
        "sInfo": "Mostrando registros del _START_ al _END_ de un total de _TOTAL_.",
        "infoEmpty": "No hay registros",
        "infoFiltered": "(filtrado de  un total de _MAX_ registros)."
    }
}).columnFilter({   sPlaceHolder: "head:before",
                    aoColumns:[ {type: "text"},
                                {type: "date-range"},
                                {type: "text"},
                                {type: "text"}
                    ]});    }
```

ILUSTRACIÓN 39. CÓDIGO DATATABLES PARA TABLA ENRIQUECIDA

Como se puede intuir en el código, se puede cambiar prácticamente todos los detalles de la tabla, esto es útil por ejemplo cuando el que lo ha desarrollado es inglés y se necesita que las cosas estén en español.

Como se puede observar las opciones son de lo más variado. Desde controlar cual es el ancho de cada columna, hasta controlar todos los textos que aparecen en la tabla (como por ejemplo el texto que aparece si no hay datos: “No se encontraron resultados”).

El resultado sería el siguiente:

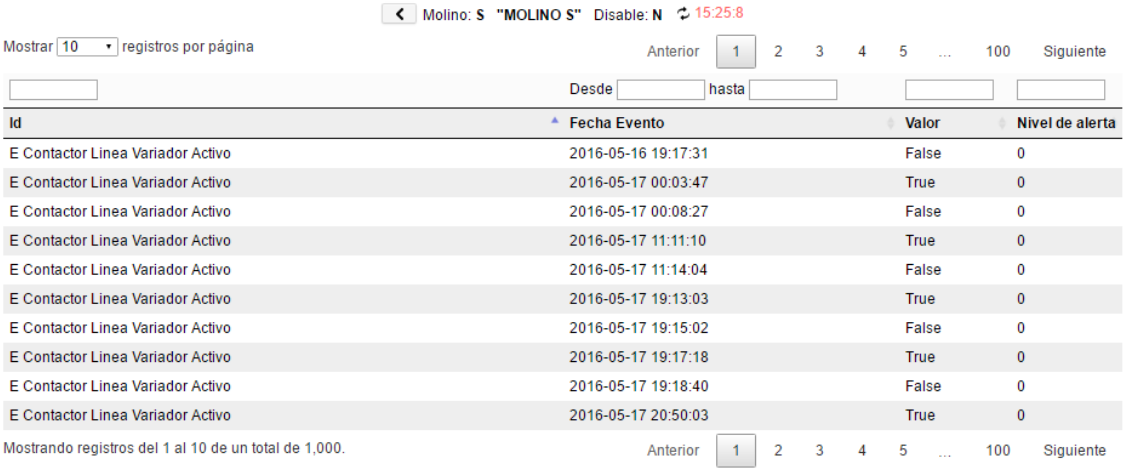


ILUSTRACIÓN 40. TABLA ENRIQUECIDA

5.2.5.2 COLUMNFILTER

Otro punto a favor y que no se ha comentado antes sobre el plug-in DataTables es la posibilidad que ofrece de que otros desarrolladores puedan añadir cosas muy interesantes como la que se ha utilizado en este caso. Podría decirse que es un plug-in de un plug-in, se trata de ColumnFilter. Por si DataTables no venía ya bastante completo existe este plug-in, que ha sido de gran ayuda, el cual permite añadir un filtro por cada columna para realizar búsquedas por columna.

DataTables tiene una opción similar (aunque en la opción no aparece ya que en esa tabla no se ha utilizado) y es la de poner un recuadro de texto en la parte superior o inferior de la tabla que sirve para buscar una cadena de texto en toda la tabla y mostrar solo aquellas filas de datos que contengan dicha cadena de texto.

Explicado esto ColumnFilter tiene la posibilidad de añadir un cuadro para filtrar por cada columna, para que la cadena de texto que se escriba en la caja de texto de una columna solo realice la búsqueda en esa columna y no en toda la tabla.

ILUSTRACIÓN 41. FILTROS POR COLUMNA DE COLUMNFILTER

Un inconveniente que se encontró es que cuando se utilizada este plug-in perdíamos el texto de la cabecera, por ejemplo, en el caso de la imagen anterior el texto "Id, Fecha Evento, Valor, Nivel de alerta" se sustituían por los recuadros de búsqueda. Sustituir esto fue tan fácil como añadir otra fila en la cabecera, aunque estuviera vacía para que se situaran los cuadros en esa primera:

```
<table id="datos_formulario" align="center">
    <thead>
        <tr style="cursor:pointer;">
            <td></td>
            <td></td>
            <td></td>
            <th></th>
        </tr>
        <tr style="cursor:pointer;">
            <th><b>Id</b></th>
            <th><b>Fecha Evento</b></th>
            <th><b>Valor</b></th>
            <th><b>Nivel de alerta</b></th>
        </tr>

    </thead>
    <tbody>
```

ILUSTRACIÓN 42. FILA CABECERA SIN DATOS

Una vez todo preparado, después del código de DataTables se añade el código siguiente y ya están listos nuestros filtros por columna:

```
        "infoFiltered": "(filtrado de  un total de _MAX_ registros)."
    }
}).columnFilter({    sPlaceHolder: "head:before",
                     aoColumns:[ {type: "text"},
                                 {type: "date-range"},
                                 {type: "text"},
                                 {type: "text"}
                     ]});     }
```

ILUSTRACIÓN 43. CÓDIGO COLUMNFILTER

La programación de este plug-in es muy sencilla, simplemente hay que meter una línea de texto con el tipo de datos que se va a filtrar por cada columna, en orden de izquierda a derecha. En este caso se tiene que la primera, tercera y cuarta columna de izquierda a derecha será de tipo texto y la segunda será de tipo date-range.

Para poner la guinda al pastel, este plug-in permite trabajar con otro plug-in de JQUERY llamado DatePicker. Este es un plug-in que permite seleccionar una fecha a golpe de ratón y de manera visual, sin tener que escribir la fecha. Por lo tanto con la combinación de estos dos se puede conseguir lo siguiente:

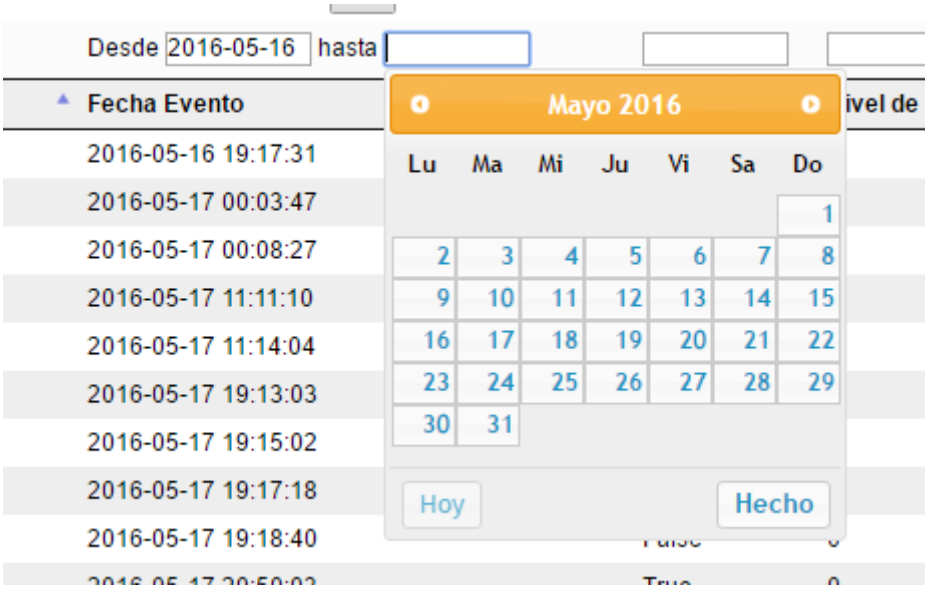


ILUSTRACIÓN 44. MUESTRA FUNCIONAMIENTO DE DATEPICKER

A parte de la comodidad que ofrece poder seleccionar la fecha con más nivel de detalle, la ventaja más potente es que el usuario que va a realizar la búsqueda no se equivoque en el formato de esta ya que no es lo mismo filtrar a partir del día 16/05/16 que 16-05-16, Según como este en base de datos una búsqueda podría devolver los datos deseados y la otra no devolver ninguno.

6.2.6 PISTOLA LECTORA DE CÓDIGOS DE BARRAS

Otra función interesante de la plataforma es la posibilidad de utilizar pistolas lectoras de códigos de barras. En la empresa una práctica habitual es la de comprobar stocks utilizando una pistola de código de barras leyendo las etiquetas que tienen los sacos pegadas, pudiendo así encontrar de manera instantánea un saco en la plataforma y poder conocer toda la información de una manera rápida.

Puesto que estas pistolas la mayoría van por USB, es algo muy práctico poder conectar una a un móvil o una Tablet (pequeños dispositivos que pueden acceder sin problemas a la web) y se pueden llevar en una mano, ya que antes esto se hacía con un ordenador instalado en un torito de carga, limitando el movimiento por la fábrica.

En la siguiente imagen tenemos un ejemplo de las etiquetas y las pistolas utilizadas en la empresa:

ILUSTRACIÓN 45. ETIQUETAS SACOS Y PISTOLA LECTORA DE CODIGOS DE BARRAS

El funcionamiento de una pistola lectora de códigos de barras es muy simple, cuando la pistola lee un código, esta se comporta como un teclado normal, es como si fuera un teclado numérico en el cual se presionan las teclas de los números que ha leído y al final aplica un enter.

El código que permite que esto funcione en la web es el siguiente:

```
function pistola_de_codigos(){

    var codigo_pistola ="";


    $(window).keydown(function (e) {
        //use e.which
        //ENTER es la tecla 13
        if(codigo_pistola.length>=13){
            codigo_pistola=codigo_pistola.substring(1)
        }

        var keyCode = e.which;

        console.log(e, keyCode, e.which)
        if (keyCode == 48 || keyCode == 96) {

            codigo_pistola = codigo_pistola + "0";

        }else if (keyCode == 49 || keyCode == 97) {

            codigo_pistola = codigo_pistola + "1";

        }else if (keyCode == 50 || keyCode == 98) {

            codigo_pistola = codigo_pistola + "2";

        }else if (keyCode == 51 || keyCode == 99) {

            codigo_pistola = codigo_pistola + "3";

        }else if (keyCode == 52 || keyCode == 100) {

            codigo_pistola = codigo_pistola + "4";

        }else if (keyCode == 53 || keyCode == 101) {

            codigo_pistola = codigo_pistola + "5";

        }else if (keyCode == 54 || keyCode == 102) {

            codigo_pistola = codigo_pistola + "6";

        }else if (keyCode == 53 || keyCode == 101) {

            codigo_pistola = codigo_pistola + "5";

        }else if (keyCode == 54 || keyCode == 102) {

            codigo_pistola = codigo_pistola + "6";

        }else if (keyCode == 55 || keyCode == 103) {

            codigo_pistola = codigo_pistola + "7";

        }else if (keyCode == 56 || keyCode == 104) {

            codigo_pistola = codigo_pistola + "8";

        }else if (keyCode == 57 || keyCode == 105) {

            codigo_pistola = codigo_pistola + "9";

        }else if (keyCode == 13) { //Enter


            if(codigo_pistola.length == 12 && codigo_pistola.substring(0,4)=="0001"){
                //alert("ES un codigo");
                document.getElementById("formu_busqueda_sacones").reset();
                document.getElementById("usado").checked=true;
                document.getElementById("id_bigbag").value=codigo_pistola.substring(5,12);
                codigo_pistola = "";
                cargar_datos_busqueda_sacones('#formu_busqueda_sacones', 0);

            }else{

                codigo_pistola = "";
                validar_formulario_sacones();
            }

        }

    });
}
```

ILUSTRACIÓN 46. CÓDIGO PISTOLA LECTORA CODIGOS DE BARRAS

En el código anterior tenemos una función de Javascript que va comprobando si las teclas que se pulsan corresponden al patrón de los códigos de las etiquetas. Una etiqueta se caracteriza porque los primeros 4 números son “0001” y los 8 siguientes son el número identificador del saco.

En esta función se comprueba que la tecla pulsada sea un número, en caso de que si lo almacena en una variable y espera a la siguiente tecla. En caso de que una de las teclas pulsadas no sea un número, borra el contenido de la variable y vuelve a empezar. En caso de que todo lo que se pulsen sean números, cuando llega a 12 comprueba si los 4 primeros números corresponden con “0001”, en caso de que si, utiliza los 8 números restantes como identificador de saco para realizar una búsqueda de saco en la base de datos.

7 CONCLUSIONES

Se han cumplido los siguientes puntos:

1. Se han cumplido los objetivos planteados en un principio
2. A día de hoy el proyecto está en marcha en una empresa real.
3. La plataforma ofrece el servicio deseado a la empresa.
4. Se ha adquirido mucha experiencia trabajando para un entorno real y teniendo que cumplir unos requisitos mínimos exigidos.

La experiencia de desarrollar un proyecto desde el principio es muy interesante, ya que uno se afronta a un problema el cual tiene que darle solución. Realizar un proyecto así enseña a que no siempre se es necesario tener los conocimientos previos para realizarlo, sino que es más importante sentarse frente al problema y plantarle cara buscándose uno la vida a decir que algo no se sabe hacer y que como no sabe eso no es para él.

Como ya se ha dicho, realizar un proyecto de principio a fin, pasando por todas sus fases es una experiencia enriquecedora y de la que uno puede estar orgulloso si el final es el deseado. También se ha aprendido que todas y cada una de las fases son igual de importantes que las demás y que no se puede dejar nada al azar.

Se han adquiridos nuevos conocimientos en PHP, HTML, Javascript, SQL y sobre todo conocimientos de trabajo en un entorno real para una empresa de verdad.

8 BIBLIOGRAFIA

https://es.wikipedia.org/wiki/Automatización_industrial

http://www.vidres.com/

http://www.prismasoftwaregestion.com/blog/software-web-vs-software-no-web-o-de-escritorio-12/

https://support.office.com/es-es/article/Conceptos-b%C3%A1sicos-sobre-bases-de-datos-a849ac16-07c7-4a31-9948-3c8c94a7c204

http://aprenderaprogramar.es/index.php?option=com_content&view=article&id=882:iqu e-es-y-para-que-sirve-ajax-ventajas-e-inconvenientes-javascript-asincrono-xml-y-json-cu01193e&catid=78:tutorial-basico-programador-web-javascript-desde-&Itemid=206

https://datatables.net/

https://datatables.net/forums/discussion/5033/datatables-column-filter-add-on-for-the-data-table

https://secure.php.net/manual/es/index.php

https://jquery.com/

http://firebirdsql.org/

http://www.w3schools.com/cssref/pr_pos_overflow.asp

https://es.wikipedia.org/

http://api.jquery.com/jquery.ajax/

http://blog.iweb.com/es/2014/02/seguridad-web-amenazas/2457.html

https://notepad-plus-plus.org/

https://www.mozilla.org/es-MX/