Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Creación de una aplicación para la coordinación de actividades de proveedores

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Ethan Samuel Martínez Escamilla

Tutor: José Onofre Montesa Andrés

Tutor Practicas: José Francisco Cebrian Pedreira

Curso 2015 / 2016

Resumen

La coordinación de actividades es una tarea fundamental para las empresas de hoy en día. Debido a la legislación vigente es imperativo para las empresas asegurarse que sus proveedores de servicios cumplen con todas las normas, certificaciones y requisitos necesarios para realizar su actividad con garantías. Este Trabajo de Fin de Grado ilustra el desarrollo de una aplicación informática cuyo objetivo es facilitar esta tarea a la empresa Aguas de Valencia. Los usuarios podrán crear perfiles de empresas contratantes, almacenar información, guardar documentación y realizar validaciones para certificar que sus proveedores de servicios cumplen con sus obligaciones legales.

Palabras clave: Aplicación, ASP.NET, MySQL, gestión, proveedores.

Abstract

Coordination of activities is a key task for businesses today. Because the law to ensure that their service providers comply with all standards is imperative for companies, certifications and requirements needed to be conducted in order to provide business with guarantees. This Final Degree Work illustrates the development of a software system designed to facilitate this task to Aguas de Valencia company. Users can create profiles of contractors, store information, keep records and validations to certify that their service providers comply with their legal obligations.

Keywords : Aplication, ASP.NET, MySQL, management, providers.

Tabla de contenidos

Tabla de Ilustraciones

1. INTRODUCCIÓN

El propósito del presente Trabajo de Fin de Grado es el diseño y desarrollo de una aplicación web para el control de documentación a proveedores. Se trata de un desarrollo con aplicación real enmarcado en un convenio de colaboración con la empresa EMIVASA del grupo Aguas de Valencia y cumpliendo los requisitos definidos por el departamento de Prevención de Riesgos Laborales.

1.1. Objetivos

El grupo Aguas de Valencia es un grupo empresarial formado por en torno a 40 empresas. Es común que las empresas pertenecientes al grupo recurran a los servicios de contratistas externos para realizar diversas tareas. Para ello antes deben comprobar que esas empresas que subcontratan cumplan con una serie de requisitos legales para llevar a cabo su actividad. En este proceso, el grupo Aguas de Valencia demanda gran cantidad de documentación a estas empresas; desde certificados médicos de trabajadores a certificados de pagos a hacienda entre otros.

Además, puede darse el caso de que las empresas contratistas subcontraten a su vez a otras empresas y estas a otras y así sucesivamente. En cualquier caso, el grupo Aguas de Valencia tiene la obligación de comprobar la documentación de todas estas empresas subcontratadas y que realizan los trabajos.

Actualmente toda esta tramitacion de documentación no está automatizada por lo que coordinar todo este intercambio de información y posterior evaluación de las empresas contratistas supone un quebradero de cabeza para el departamento de Prevención de Riesgos Laborales.

El objetivo de este Trabajo de Fin de Grado es la creación de una aplicación web que permita simplificar y agilizar esta tramitacion de documentación entre empresas contratistas y el grupo Aguas de Valencia, facilitando además a los trabajadores del departamento de Prevención de Riesgos Laborales la evaluación y clasificación las empresas subcontratadas.

La aplicación debe ser una vía de comunicación directa entre las empresas contratistas y el grupo Aguas de Valencia. De modo que esta aplicación deberá permitir a las empresas contratistas acceder a un entorno en el que cargar y mantener documentación de forma organizada y ordenada mientras que el grupo Aguas de Valencia dispondrá de una herramienta que permitirá llevar un control de toda esa información, revisar la documentación proporcionada por los proveedores, acceder a un sistema de validación interno que comprobará si se cumplen ciertas condiciones de obligado cumplimiento y disponer de un sistema de alertas y notificaciones para en última instancia liberar de carga de trabajo y facilitar el cumplimiento de sus tareas a los trabajadores del departamento de Prevención de Riesgos Laborales de Aguas de Valencia.

1.2. Descripción Marco de Prácticas en Empresa

Este proyecto nace como consecuencia de la finalización de un convenio de prácticas de tipo curricular realizado con la empresa EMIVASA perteneciente al grupo Aguas de Valencia (AVSA). En vista de la positiva experiencia y los buenos resultados de este primer convenio se propuso realizar un segundo convenio de colaboración, esta vez de tipo extra curricular, que se centraría en el desarrollo del presente Trabajo Fin de Grado.

Las prácticas fueron realizadas dentro del departamento de sistemas informáticos de AVSA. Este departamento está divido en dos secciones; sistemas y desarrollo.

El alumno realizo la aplicación dentro de la sección de desarrollo que se divide a su vez en equipos de programadores que se forman en función de las tecnologías empleadas.

1.3. Estructura de la Memoria

La memoria realizada consta de una serie de bloques que engloban todo el trabajo realizado y que son descritos brevemente a continuación.

En primer lugar, metodologías empleadas donde se justificará la metodología empleada en el desarrollo del proyecto.

En segundo lugar, tecnología, donde se describen y justifican las distintas tecnologías que se emplearan en el desarrollo de la aplicación.

En tercer lugar, la especificación de requisitos, que consiste en describir las necesidades de la aplicación a desarrollar.

En cuarto lugar, herramientas, listado de las herramientas software empleadas para el desarrollo de la aplicación.

En quinto lugar, análisis y diseño, donde se describe el proceso de diseño y definición final de la aplicación, previo a su desarrollo.

En sexto lugar, implementación, en este apartado se describe el proceso de desarrollo de la aplicación, detallando el uso de tecnologías, herramientas y componentes concretos empleados en el desarrollo.

En séptimo lugar, evaluación, alberga el resultado final de la aplicación y se compara con los requisitos especificados por el cliente.

En octavo lugar, conclusión, donde se plantea el recorrido futuro de la aplicación dentro de la empresa, se discuten futuras ampliaciones y se incluyen impresiones personales del autor.

Por último, el anexo, donde se incluye el manual y el código completo de la aplicación

2. METODOLOGÍAS EMPLEADAS

En el presente capítulo se describen las metodologías empleadas en el desarrollo de la aplicación. Debe tenerse en cuenta que las metodologías aquí descritas fueron una restricción impuesta por la empresa en la que tuvo lugar el convenio de prácticas, teniendo que ser adaptadas por parte del alumno para encajar con las necesidades y dimensiones del proyecto.

2.1. Agile

Los equipos (1) de desarrollos informáticos de la empresa Aguas de Valencia utilizan ampliamente el desarrollo ágil de software en sus proyectos.

El desarrollo ágil de software envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Así el trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo.

Cada iteración del ciclo de vida incluye:

- Planificación
- Análisis de requisitos
- Diseño
- Codificación
- Pruebas
- Documentación

El objetivo del desarrollo ágil es que cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea potencialmente entregable, de manera que cuando el cliente lo solicite sólo sea necesario un esfuerzo mínimo para que el producto esté disponible para ser utilizado.

2.2. SCRUM

SCRUM (2) es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son:

- El Scrum Master; que procura facilitar la aplicación de scrum y gestionar cambios
- El Product Owner; que representa a los stakeholders (interesados externos o internos)
- El Team (equipo) que ejecuta el desarrollo

Durante cada sprint, (de entre una y cuatro semanas), el equipo avanza en el desarrollo de software que debe ser potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de requisitos priorizados que definen el trabajo a realizar. Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de

Sprint Planning. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo conversa con el Product Owner para analizar el trabajo a realizar. Durante el sprint, nadie puede cambiar el Sprint Backlog, los requisitos se mantienen inalterados durante el sprint.

Un principio clave de Scrum, es que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan, asi como que los desafíos impredecibles no pueden ser fácilmente abordados de forma predictiva y planificada. Por lo tanto, Scrum adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes.

Las principales ventajas de Scrum son:

- Flexibilidad a cambios. Gran capacidad de reacción ante los cambios planteados por las necesidades del cliente.

- Mayor calidad del software. El trabajo metódico y la necesidad de obtener una versión de trabajo funcional después de cada iteración, ayuda a la obtención de un software de alta calidad.

- Mayor productividad. gracias a la eliminación de la burocracia y la agilidad del equipo debida a que pueden estructurarse de manera autónoma.

- Predicciones de tiempos. A través de este marco de trabajo se conoce la velocidad media del equipo por sprint.

2.3. Aplicación de metodologías en el proyecto

Las metodologías anteriormente descritas fueron utilizadas en el desarrollo del presente trabajo. Para ello debieron adaptarse levemente para encajar con el marco de las prácticas. El alumno realizo el trabajo dentro de un equipo de desarrollo pero fue autónomo en la toma de decisiones y desarrollo.

Se siguió el desarrollo ágil al establecer iteraciones en desarrollo de dos semanas de duración aproximadamente. Así el desarrollo se organizó de la siguiente forma (etapas por iteración):

- Planificación: el alumno planificaba con su supervisor las jornadas de cada sprint.
- Análisis de requisitos: el alumno analizaba los requisitos con el cliente (departamento de Prevención de Riesgos Laborales)
- Diseño: el alumno afrontaba la fase de diseño que luego discutía con su supervisor.
- Codificación; el alumno comenzaba la programación del código.
- Pruebas; el alumno testeaba la aplicación junto con el supervisor.

- Documentación; el alumno documentaba los cambios en el producto y se enviaban al cliente, junto con la publicación de la nueva versión del software.

Una vez concluidos todos los pasos de una iteración, comenzaba la siguiente iteración en el desarrollo siguiendo el mismo esquema.

Para adaptar el modelo de referencia Scrum se asignaron los siguientes roles

- Scrum Master; este rol recayó sobre el supervisor y jefe del equipo de desarrollo, encargado de guiar al alumno en el desarrollo
- Product Owner; este rol recayó sobre el jefe del Departamento de Prevención de Riesgos Laborales, representa al cliente. El alumno mantuvo diversas reuniones con él para la especificación de requisitos
- Team; este rol recayó enteramente sobre el alumno que se encargó en solitario del desarrollo.

3. TECNOLOGÍA

La elección de las tecnologías empleadas viene condicionada por la imposición de algunas de ellas por parte de la empresa. Se especificó que la aplicación debería ser desarrollada mediante el framework ASP .NET de Microsoft (3) utilizando C# como lenguaje de programación principal y que la base de datos empleada debía ser MySql. Además, se requirió el uso del paquete de software DevExpress (que proporciona controles predefinidos de interfaz de usuario) del que se verán más detalles en el apartado de Herramientas.

A continuación, se describen brevemente las tecnologías empleadas

3.1. .NET

Se trata de un framework para desarrollo de aplicaciones creado por Microsoft. Está compuesto por diferentes componentes; el Common Language Runtime (CLR) y la biblioteca de clases de .NET Framework, que incluye clases, interfaces y tipos de valor que son compatibles con una amplia gama de tecnologías. Además, integra multitud de lenguajes de programación como C++ o VisualBasic.

- **Sobre el CLR**: Common Language Runtime administra la memoria, ejecución de subprocesos, ejecución de código, comprobación de la seguridad del código, compilación y demás servicios del sistema. Estas características son intrínsecas del código administrado que se ejecuta en Common Language Runtime. Permite al programador acceder a las características del Framework .NET con independencia del lenguaje de programación empleado

- **Sobre la biblioteca de clases:** La biblioteca de clases de .NET Framework es una colección de tipos reutilizables que se integran estrechamente con Common Language Runtime. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de .NET Framework. Además, los componentes de terceros se pueden integrar sin dificultades con las clases de .NET Framework.

3.2. ASP .NET

Dentro de la plataforma .NET, ASP .NET (4) es el framework de Microsoft para desarrollo de aplicaciones Web. Tiene acceso a las clases del framework .NET y el código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR) además de los propios de la programación web como son HTML5, CSS o JavaScript.

Actualmente, ASP.NET soporta tres modelos de programación: ASP.NET Web Forms, ASP.NET MVC y ASP.NET Web Pages. Aunque los tres modelos de programación se ejecutan sobre la misma base de ASP.NET, cada uno de ellos estructura la aplicación de maneras completamente distintas, promueve metodologías de desarrollo diferentes y se adapta a perfiles de desarrolladores distintos.

- **ASP.NET WebForms**: Fue el primero de los tres modelos de programación en existir, y proporciona un gran nivel de abstracción con un modelo de programación familiar basado en eventos y controles que favorece la productividad mediante la programación declarativa reduciendo la cantidad de código necesaria para implementar una determinada funcionalidad.
- **ASP.NET MVC**: Se concibió como alternativa a Web Forms y proporciona un modelo de programación basado en el popular patrón de arquitectura MVC. Entre sus principales características destacan su completa integración con pruebas unitarias y su separación más clara entre la lógica de presentación, la lógica de negocio y la lógica de acceso a datos.
- **ASP.NET Web Pages**: Es el más reciente de los tres modelos de programación, y fue creado como respuesta a una creciente demanda de desarrolladores web sin experiencia previa con ASP.NET, cuya iniciación en ASP.NET Web Forms o MVC les suponía una inversión inicial de tiempo demasiado grande. Web Pages proporciona un modelo de programación más simple y rápido de aprender, sin renunciar a toda la funcionalidad y flexibilidad de ASP.NET.

Finalmente se decidió realizar la aplicación en ASP.NET Webforms debido a que permitía un desarrollo más rápido y ágil que WebPages y MVA, una curva de aprendizaje asequible y, especialmente, a que los controles de UI (user interface) de DevExpress han demostrado funcionar mejor con Webforms que con MVC.

3.3. ASP.NET Webforms

Los formularios Web Forms (5) son páginas que los usuarios solicitan utilizando su navegador. Estas páginas se pueden escribir usando una combinación de HTML, script de cliente, los controles de servidor y código de servidor.

Cuando los usuarios solicitan una página, el código es compilado y ejecutado en el servidor por el framework, y entonces el framework genera código HTML interpretable por los navegadores web.

Las características principales de ASP .NET Webforms son:

- Está basado ASP.NET de Microsoft; el código se ejecuta en el servidor y de forma dinámica genera la salida de la Página Web al dispositivo cliente o navegador.
- Es compatible con cualquier navegador o dispositivo móvil.
- Es compatible con cualquier lenguaje soportado por el .NET Common Language Runtime, como Microsoft Visual Basic y Microsoft Visual C#.
- Permite agregar controles creados por el usuario y terceros (como es el caso de DevExpress).

Además ofrece las siguientes ventajas:

- Separación de HTML y otros códigos de interfaz de usuario de la lógica de la aplicación.
- Un conjunto amplio de controles de servidor como acceso a datos.
- Soporte para scripting de lado del cliente que se ejecuta en el navegador.

3.4. C#

Lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET.

La elección de C# viene condicionada por ser el lenguaje de programación predeterminado que se usa en el departamento de sistemas de AVSA para aplicaciones web.

3.5. html5

Lenguaje de marcado utilizado para presentar y estructurar contenido Web. Se trata de la 5 revisión del estándar HTML realizado por el World Wide Web Consortium (W3C).

3.6. MySql

MySQL (6) es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. Su elección viene motivada por requerimiento de la empresa de prácticas que tiene por norma trabajar con MySQL.

4. HERRAMIENTAS

4.1. Visual Studio 2013 Community

Microsoft Visual Studio (7) es un entorno de desarrollo para sistemas operativos Windows. Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP; al igual que entornos de desarrollo web como ASP.NET MVC, Django, etc.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET.

La versión 2013 Community hace referencia a la edición gratuita para fines académicos que distribuye Microsoft. Si bien no es la más actual se decidió utilizar la versión 2013 por problemas de compatibilidad con la base de datos MySQL.

- **Aplicación en el proyecto**: Visual Studio ha sido la herramienta principal de este proyecto. Su uso ha sido necesario debido al modelo y plataforma de programación impuesto por la empresa (Plataforma .NET de Microsoft).

4.2. Microsoft Team Foundation Server

Team Foundation Server (comúnmente abreviado como TFS) es un producto de Microsoft que proporciona administración de código fuente y control de versiones.

- **Aplicación en el proyecto:** Team Fundation Server se ha utilizado para realizar el control de versiones de la aplicación, así como el control de copias de seguridad.

4.3. MySql Workbench

MySql Workbench es una herramienta de gestión de bases de datos MySQL desarrollado por Oracle. Proporciona además utilidades para el diseño de bases de datos, permitiendo definir esquemas UML mediante un editor gráfico. La aplicación genera scripts en función de esos esquemas. También es capaz de realizar la operación inversa y generar esquemas UML en función de una base de datos existente.

- **Aplicación en el proyecto**: MySQL workbench ha sido una de las herramientas principales de este proyecto. Como se verá más adelante se ha utilizado para realizar esquemas previos de la base de datos, diseño UML, generación de scripts, creación de la Base de Datos y mantenimiento de la Base de Datos

4.4. Entity Framework

Entity Framework (EF) (8) es un asignador objeto-relacional que permite a los desarrolladores de .NET trabajar con datos relacionales usando objetos específicos del dominio. Elimina la necesidad de la mayor parte del código de acceso a datos que los desarrolladores suelen tener que escribir.

Permite crear un modelo en función de las necesidades del programador mediante la selección de un flujo de trabajo. Los flujos de trabajo que soporta Entity Framework son:

-**Code First**: permite definir el modelo de datos mediante código para después generar una base de datos.

-**Model First**: permite definir el modelo mediante un entorno gráfico para después generar una base de datos y el código del modelo.

-**Database First**: permite crear el modelo de datos a partir de una base de datos existente.

- **Aplicación en el proyecto**: Entity Framework se ha usado como Data Access Layer. Su elección viene motivada por la liberación de carga de trabajo y por su buena integración con Web Forms y DevExpress. La implementación se verá más adelante. Se ha utilizado el flujo de trabajo Database First.

4.5. DevExpress

DevExpress (9) es una empresa de software dedicada al desarrollo de controles de UI (User Interface o interfaz de usuario) para diversas plataformas. Su uso requiere una licencia de pago que da acceso a la suite informática. En nuestro caso se instala en forma de plug-in del IDE VisualStudio, añadiendo las librerías y herramientas necesarias para su uso con Web Forms y .NET. El uso de DevExpress fue un requisito de la empresa que basa muchos de sus proyectos en este conjunto de herramientas. Proporciona una gran variedad de controles que permiten liberar carga al programador. Más adelante se detallan los controladores empleados.

- Aplicación en el proyecto: DevExpress se ha utilizado en toda la capa de presentación e interfaz de usuario.

4.6. NuGet

NuGet es un gestor de paquetes gratuito y opensource para la plataforma de desarrollo de Microsoft. Se distribuye como extensión de Visual Studio y permite la instalación de paquetes y plugins tanto de organismos y empresas oficiales como de desarrolladores particulares.

- Aplicación en el proyecto: NuGet se ha utilizado para la instalación de diversos paquetes necesarios para la realización del proyecto como Entity Framework, Microsoft Source Control o los conectores necesarios de MySQL.

4.7. Otras Herramientas

Además de las herramientas definidas se han utilizado otras herramientas y componentes propios de la empresa. Estas son controles o web services ya definidos desarrollados por y para el departamento de sistemas de la información de AVSA.

Un servicio web (en inglés, Web Service o Web services) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores.

Componentes propios de la empresa:

- **Permisator**: es un sistema de desarrollo propio que gestiona los permisos de todas las aplicaciones del grupo AVSA desarrolladas en .NET. Se divide en dos partes principales; una web donde pueden registrarse las aplicaciones, asociarlas a usuarios y asignarles permisos y un Web Service que proporciona al programador una serie de métodos que permiten acceder a esa información.
- **Gestor de Notificaciones**: se trata de un Web Service que permite gestionar el envío de notificaciones por email o mensajes de texto. Proporciona una serie de métodos que a partir de una serie de parámetros permiten programar el envío de mensajes automáticamente.
- **Gestor de Usuario**: se trata de un Web Service que permite acceder a información de todos los trabajadores del grupo que tengan usuario del dominio interno. Entra la información que ofrece puede consultarse el correo electrónico o el teléfono de los usuarios.
- **Log de Operaciones**: se trata de una clase desarrollada por AVSA que automatiza tareas de registro de operaciones en las aplicaciones del grupo.

También se ha utilizado el control de desarrollo interno File Manager:

- **File Manager**: se trata de un control desarrollado en c# para gestionar la carga y almacenamiento de ficheros. Proporciona una interfaz que el programador puede integrar en sus proyectos con poca configuración e incluye todos los componentes necesarios para cargar ficheros, organizarlos en el servidor y acceder a ellos posteriormente.

5. ESPECIFICACIÓN DE REQUISITOS

En el presente apartado se especifican los requisitos de la aplicación definidos por el departamento de Prevención de Riesgos Laborales de la empresa Aguas de Valencia.

Los requisitos a continuación expuestos son resultado de una serie de reuniones con el responsable del área de Prevención de Riesgos Laborales.

5.1. Descripción General

Se desea la creación de una aplicación que permita llevar un control de la documentación de los contratistas y proveedores de servicios que trabajan para el Grupo Aguas de Valencia (de aquí en adelante AVSA). Esta documentación deberá ser revisada por los trabajadores de AVSA para comprobar que las empresas contratistas cumplen con la legalidad vigente y que pueden ejercer su actividad.

Se plantea la necesidad de un entorno en el que las empresas contratistas puedan acceder y cumplimentar formularios para hacer llegar cierta información a AVSA. Deberán proporcionar información referente a los siguientes apartados:

- Información general de la Empresa y sus documentos asociados.
- Información de Trabajos Especiales y sus documentos asociados.
- Información de Equipos y Vehículos y sus documentos asociados.
- Información de Trabajadores y sus documentos asociados.
- Información de Obras y sus documentos asociados.

La estructura de la aplicación deberá ser como se describe a continuación. Cada empresa contratista podrá tener uno o más documentos. También se podrán definir trabajos especiales, equipos, vehículos, trabajadores y obras. Todas estas entidades dependerán de la empresa y todas ellas tendrán a su vez documentos asociados.

Los puntos anteriores se definen en detalle en el siguiente apartado.

Se plantea que la aplicación tenga dos tipos de usuarios básicos;

Los contratistas accederán a la aplicación y tendrán acceso a la información referente a su empresa (nunca tendrán accesibles el resto de empresas). De ahora en adelante se hará referencia a estos usuarios como Usuarios Contratistas.

Por otro lado, los Administradores de Empresas Contratistas de AVSA (a partir de aquí Usuarios Administradores) tendrán acceso a una lista de empresas donde podrán consultar la información descrita y cargar o consultar documentos.

Complementariamente a las especificaciones anteriores se solicita la creación de un mecanismo que permita a los usuarios Administradores validar la documentación presentada por los contratistas de forma que estos últimos conozcan si un documento es aceptado o no por AVSA. Además, se pide un sistema de validación de documentación automático que compruebe la fecha de caducidad de los documentos y los marque como no válidos (estas comprobaciones son descritas en detalle en el punto siguiente).

También se pide la creación de un sistema de semáforos que mediante un código de colores permita diferenciar a las empresas aptas de las no aptas.

La aplicación deberá además tener un sistema de avisos por email que notifique a los usuarios contratistas y a los usuarios Administradores cuando un documento tenga próxima su fecha de caducidad.

Por último, se solicita que dada la importancia legal y operativa de la información almacenada por la aplicación se implemente algún registro de acciones llevadas a cabo por los usuarios. La aplicación deberá almacenar todas las transacciones que se lleven a cabo en la base de datos para ser consultadas en caso necesario. Además, se solicita expresamente que en el caso de los documentos se implemente un indicador que muestre por pantalla el último usuario que edito algún campo del documento.

A modo de resumen de las especificaciones generales de la aplicación se concluye que:

- La aplicación almacenará información de empresas contratistas.
- Cada empresa tendrá documentos asociados.
- Cada empresa tendrá trabajadores, trabajos especiales, obras, equipos y vehículos asociados que a su vez tendrán su propia documentación.
- Los usuarios de AVSA deben poder validar documentos.
- La aplicación debe realizar validaciones automáticas.
- La aplicación debe ser capaz de identificar empresas válidas y no válidas mediante un sistema visual (sistema de semáforos).
- La aplicación debe disponer de un sistema de notificaciones por email.
- La aplicación debe guardar un registro de las acciones que realizan sus usuarios.

5.2. Requisitos Específicos

En el presente apartado se definen al detalle las especificaciones planteadas en el apartado anterior.

5.2.1. Tipos de Usuarios

Según las especificaciones establecidas por el departamento de Prevención de Riesgos Laborales la aplicación deberá tener como mínimo dos tipos de usuarios; los usuarios de tipo contratista y los usuarios de tipo administrador.

- Usuarios Contratistas: Este tipo de usuario representa a las empresas. Por cada empresa se deberá crear un usuario de este tipo. Estos usuarios solo tendrán acceso a la información de la empresa a la que pertenecen y sólo podrán editar ciertos menús. Tampoco podrán validar documentos.
- Usuarios Administradores: Este tipo de usuarios representan al personal de AVSA. Tienen acceso completo a la aplicación y deben poder crear, actualizar y eliminar entidades, así como dar de alta, modificar y validar documentos.

Más adelante en esta memoria se especificarán las funciones concretas y al detalle de cada tipo de usuario.

5.2.2. Campos de Entidades

A continuación, se detallan los campos solicitados para cada una de las entidades de la aplicación, entendiendo como entidad los apartados principales definidos por AVSA.

5.2.2.1. Campos de Empresa

Para cada empresa se dese almacenar la siguiente información.

1. Nombre de Empresa; es un campo de texto definido por el usuario.
2. Estado; es un campo que define si la empresa es válida. Se desea que sea algún tipo de indicación visual.
3. Actividad; representa el tipo de actividad al que se dedica la empresa. Debe ser una lista de valores. Las opciones son las siguientes:
    a. Aplicación de Fangos
    b. Descarga de Cubas
    c. Empresa de Control
    d. Mantenimiento Eléctrico
    e. OCAS
    f. Mantenimiento de Equipos
    g. Mantenimiento de Sistemas de Detección
    h. Mantenimiento de equipos anticaidas
    i. Descarga de Productos Químicos
    j. Obras
    k. Mantenimiento de Colectores
4. CIF; el CIF de la empresa, editable por el usuario.
5. Dirección; campo de texto editable por el usuario.
6. C.P; código postal, campo de texto o numérico editable por el usuario.
7. Municipio; campo que representa el municipio de la empresa. Debe ser una lista de valores
8. Responsable de Prevención; campo de texto con el nombre de la persona de contacto de la empresa.
9. Teléfono Prevención; campo numérico que representa un teléfono de contacto del contratista.
10. Correo; campo de texto que representa el email de contacto de la empresa. Este correo es que deberá usarse en las notificaciones por email.
11. Servicio de prevención; este campo debe ser una lista de valores con los siguientes valores:
    a. Asumido por Empresa
    b. SPA
    c. SP Mancomunado
    d. SPP
12. Histórico; este campo será un checkbox editable por el usuario y servirá para marcar una empresa como dada de baja. En ningún caso se eliminarán empresas del sistema.

Se desea que estos campos sean editables sólo por los usuarios administradores. Esto implica que sólo los usuarios administradores podrán crear empresas y los usuarios contratistas verán estos campos, pero no deben ser capaces de editarlos.

5.2.2.2. Campos de Documentos de Empresa

Cada empresa deberá tener una serie de documentos asociados.

Estos documentos necesitarán los siguientes campos:

1. Tipo de Documentos; debe ser una lista de valores con las siguientes opciones:
    a. Certificado de Pagos a Hacienda
    b. Certificado Seguridad Social
    c. Contrato Servicio Prevención
    d. Comunicación Apertura Centro de Trabajo
    e. Contrato Servicio Prevención
    f. Convenio Colectivo
    g. Evaluación de Riesgos
    h. IC / ITA
    i. Normas Generales Salud
    j. Libro de Subcontratación
    k. Libro de Visitas
    l. Registro inscripción en el REA
    m. Registro inscripción en el RERA
    n. Seguro Responsabilidad Civil
    o. Escrito Subcontratistas
    p. Pago de la Mutua
    q. Carta Compromiso GAVSA
    r. Otros
2. Fecha de Carga; debe almacenar la fecha de carga del documento en el sistema. Debe autocompletarse.
3. Fecha de Emisión; es un campo de tipo fecha definido por el usuario.
4. Fecha de Caducidad; es un campo de tipo fecha definido por el usuario.
5. Campo de Validación; debe ser un campo de tipo checkbox que represente si un documento es válido o no. Es definido por el usuario.
6. Descripción; es un campo de texto largo definido por el usuario.
7. Documento; este campo debe dar acceso al documento en sí.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.2.3. Campos de Trabajos Especiales

Los trabajos especiales son aquellas actividades específicas y concretas que una empresa contratista puede realizar (por ejemplo, trabajos en altura). Debe tener los siguientes campos:

1. Tipo de Trabajo; debe ser una lista de valores con los tipos de trabajos que puede realizar:
    a. Amianto
    b. Desmontaje de Bombas

   c. Espacios Confinados
   d. Manejo de Cargas
   e. Productos Químicos
   f. Riesgo Eléctrico
   g. Trabajos en Altura
   h. Otros
2. Descripción; campo de texto largo editable por el usuario.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.2.4. Campos de Documentos de Trabajos Especiales

Para cada trabajo especial deben almacenarse distintos tipos de documentos. Se desea que cada documento tenga los siguientes campos:

1. Tipo de Documento; este campo debe ser una lista de valores con las siguientes opciones:
   a. Acta Coordinación Actividades Empresariales
   b. Certificado Trabajo Especial
   c. Procedimiento de Trabajo
   d. Recurso Preventivo
   e. Otros
2. Fecha de Carga; debe almacenar la fecha de carga del documento en el sistema. Debe autocompletarse.
3. Fecha de Emisión; es un campo de tipo fecha definido por el usuario.
4. Fecha de Caducidad; es un campo de tipo fecha definido por el usuario.
5. Campo de Validación; debe ser un campo de tipo checkbox que represente si un documento es válido o no. Es definido por el usuario.
6. Descripción; es un campo de texto largo definido por el usuario.
7. Documento; este campo debe dar acceso al documento en sí.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.2.5. Campos de Obras

Cada empresa podrá ligarse a una o más obras. Estas obras requerirán completar los siguientes campos.

1. Nombre Obra; campo de texto editable por el usuario.
2. Dirección; campo de texto editable por el usuario.
3. Fecha inicio; campo de tipo fecha que representa el inicio de la obra.
4. Fecha fin; campo de tipo fecha que representa el fin de la obra.

5.2.2.6. Campos de Documentos Obras

Para cada obra deben almacenarse distintos tipos de documentos. Se desea que cada documento tenga los siguientes campos:

1. Tipo de Documento; este campo debe ser una lista de valores con las siguientes opciones:
   a) Acta Coordinación de Actividades Empresariales
   b) Doc. Nombramiento Coordinador
   c) Plan de Seguridad
   d) Recurso Preventivo
   e) Otros
2. Fecha de Carga; debe almacenar la fecha de carga del documento en el sistema. Debe autocompletarse.
3. Fecha de Emisión; es un campo de tipo fecha definido por el usuario.
4. Fecha de Caducidad; es un campo de tipo fecha definido por el usuario.
5. Campo de Validación; debe ser un campo de tipo checkbox que represente si un documento es válido o no. Es definido por el usuario.
6. Descripción; es un campo de texto largo definido por el usuario.
7. Documento; este campo debe dar acceso al documento en sí.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.2.7. Campos de Equipos

Cada empresa puede tener distintos tipos de equipos dados de alta. Estos equipos deberán tener los siguientes campos:

1. Descripción; es un campo de texto largo definido por el usuario.
2. Manual; es un campo de texto largo definido por el usuario.
3. Información; es un campo de texto largo definido por el usuario.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.2.8. Campos de Documentos Equipos

Para cada equipo deben almacenarse distintos tipos de documentos. Se desea que cada documento tenga los siguientes campos:

1. Tipo de Documento; este campo debe un texto editable por el usuario.
2. Fecha de Carga; debe almacenar la fecha de carga del documento en el sistema. Debe autocompletarse.
3. Fecha de Emisión; es un campo de tipo fecha definido por el usuario.
4. Fecha de Caducidad; es un campo de tipo fecha definido por el usuario.
5. Campo de Validación; debe ser un campo de tipo checkbox que represente si un documento es válido o no. Es definido por el usuario.
6. Descripción; es un campo de texto largo definido por el usuario.
7. Documento; este campo debe dar acceso al documento en sí.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.2.9. Campos de Vehículos

Los datos que deben almacenarse para los vehículos son los siguientes.

1. Descripción; es un campo de texto largo definido por el usuario.
2. Manual; es un campo de texto largo definido por el usuario.
3. Información; es un campo de texto largo definido por el usuario.
4. Matricula; es un campo de texto largo definido por el usuario.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.2.10. Campos de Documentos Vehículos

Para cada Vehículo deben almacenarse distintos tipos de documentos. Se desea que cada documento tenga los siguientes campos:

1. Tipo de Documento; este campo debe ser una lista de valores con las siguientes opciones:
   a. Ficha Técnica
   b. ITV
   c. Permiso de Circulación
   d. Seguro del Vehículo
   e. Tarjeta de Transporte
   f. Otros
2. Fecha de Carga; debe almacenar la fecha de carga del documento en el sistema. Debe autocompletarse.
3. Fecha de Emisión; es un campo de tipo fecha definido por el usuario.
4. Fecha de Caducidad; es un campo de tipo fecha definido por el usuario.
5. Campo de Validación; debe ser un campo de tipo checkbox que represente si un documento es válido o no. Es definido por el usuario.
6. Descripción; es un campo de texto largo definido por el usuario.
7. Documento; este campo debe dar acceso al documento en sí.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.2.11. Campos de Trabajadores

Cada empresa puede tener varios trabajadores. Los trabajadores deben tener los siguientes campos

1. Nombre; campo de texto editable por el usuario.
2. Estado; es un campo que define si el trabajador es válido. Se desea que sea algún tipo de indicación visual.
3. Histórico; este campo será un checkbox editable por el usuario y servirá para marcar un trabajador como dado de baja.
4. DNI; campo de texto editable por el usuario.
5. Puesto de Trabajo; campo de texto editable por el usuario
6. Información; es un campo de texto largo definido por el usuario.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.2.12. Campos de Documentos de Trabajadores

Para cada Trabajador deben almacenarse distintos tipos de documentos. Se desea que cada documento tenga los siguientes campos:

1. Tipo de Documento; este campo debe ser una lista de valores con las siguientes opciones:
    a. Entrega Información
    b. Formación
    c. Hoja Entrega de EPIS
    d. Nombramiento Recurso Preventivo
    e. Nombramiento Trabajador autorizado/cualificado
    f. Reconocimiento Médico
    g. DNI
    h. Otros
2. Fecha de Carga; debe almacenar la fecha de carga del documento en el sistema. Debe autocompletarse.
3. Fecha de Emisión; es un campo de tipo fecha definido por el usuario.
4. Fecha de Caducidad; es un campo de tipo fecha definido por el usuario.
5. Campo de Validación; debe ser un campo de tipo checkbox que represente si un documento es válido o no. Es definido por el usuario.
6. Descripción; es un campo de texto largo definido por el usuario.
7. Documento; este campo debe dar acceso al documento en sí.

Estos campos deben ser editables por los usuarios contratistas y administradores.

5.2.3. Sistema de Validaciones

La aplicación deberá permitir a los usuarios administradores aprobar documentos. Los documentos aprobados por los usuarios serán considerados como válidos siempre que no estén caducados.

A su vez las empresas y trabajadores tendrán un indicador de validez. Este indicador será automático y para su cálculo se empleará la validez de los documentos que dependan de esa entidad.

Se definen las siguientes reglas:

- Para documentos: un documento perteneciente a cualquier tipo de entidad será válido cuando un usuario administrador lo especifique como tal y cuando su fecha de caducidad no sea anterior a la fecha actual. La comprobación de la fecha de caducidad la llevará a cabo la aplicación automáticamente. Un documento caducado no podrá ser marcado como válido.
- Para trabajadores: un trabajador será válido cuando todos sus documentos sean válidos. La comprobación la llevará a cabo la aplicación de forma autónoma.

- Para las empresas: una empresa será válida cuando todos sus documentos sean válidos y cuando todos los trabajadores asociados a ella sean válidos.

5.2.4. Sistema de Semáforos

El sistema de semáforos es una representación visual del sistema de validaciones. Se deberá implementar un indicador que se marque en rojo o verde en función de que una empresa (o trabajador) sea no válido o válido respectivamente.

5.2.5. Sistema de Validación automática de documentos

Se desea que la aplicación implemente una función de validación automática de documentos. La aplicación deberá ser capaz de manera periódica de comprobar si un documento es válido o no. Para ello se deberá implementar una rutina que de forma periódica compruebe la fecha de caducidad de los documentos y en los casos en que se detecte que la fecha ya no es apta deberá marcarse automáticamente ese documento como no válido.

Deberá tenerse en cuenta aquellos casos en los que el cambio de validez de un documento pueda afectar a la validez de otra entidad como ocurre en el caso de empresas y trabajadores. La aplicación deberá ser capaz de recalcular de forma dinámica la validez de estas entidades.

5.2.6. Sistema de Notificaciones por email

Otro de los requisitos definidos es un sistema de aviso de caducidad de documentos. La aplicación deberá ser capaz de comprobar diariamente que documentos van a caducar próximamente. Calculará cuáles de ellos tienen fecha de caducidad 30 días después de la fecha actual y enviará un correo electrónico automáticamente al responsable de la empresa contratista y a los administradores de la aplicación de AVSA para que revisen la documentación.

5.2.7. Detalle de Obras

La aplicación deberá registrar las obras por empresa y su documentación de forma similar a lo que ocurre con el resto de entidades, pero además de esto deberá tenerse en cuenta el registro de contratistas por obra.

Cada obra puede tener un número indeterminado de contratistas. AVSA siempre trabajará con un contratista principal, pero este a su vez puede subcontratar a otro contratista y así sucesivamente. Así pues, debe implementarse algún tipo de estructura que permita definir estas relaciones.

A modo de resumen:

- La aplicación debe ser capaz de registrar obras por empresa.
- Cada obra puede tener un número indeterminado de contratistas.
- La aplicación debe registrar la estructura de subcontratistas y poder representarla.

5.3. Mockups Propuestos por el Cliente

Un mockup es un esquema gráfico que pretende ilustrar el aspecto y funcionamiento de una aplicación software.

Para facilitar el proceso de especificación de requisitos se pidió al cliente que definiera una serie de mockups en los que describiera a grandes rasgos las funcionalidades deseadas y el aspecto que esperaba que tuviera la aplicación. A continuación, se adjuntan estos esquemas. Más adelante en este trabajo se verán los mockups propuestos por el alumno y se compararán con la primera versión enviada por el cliente.

Estos mockups son de una versión muy temprana por lo que no recogen muchos requisitos que serían añadidos posteriormente por el cliente.

5.3.1. Esquema de Empresa

En el esquema de empresa el cliente define algunos de los campos básicos que desea almacenar y define también una propuesta de organización de documentos.

Aquí el cliente propone tener campos fijos para cada documento y un acceso en forma de botón al documento en sí. Esta estructura tal y como la define el cliente no es implementable en un desarrollo real por lo que necesitará de un estudio de adaptación.

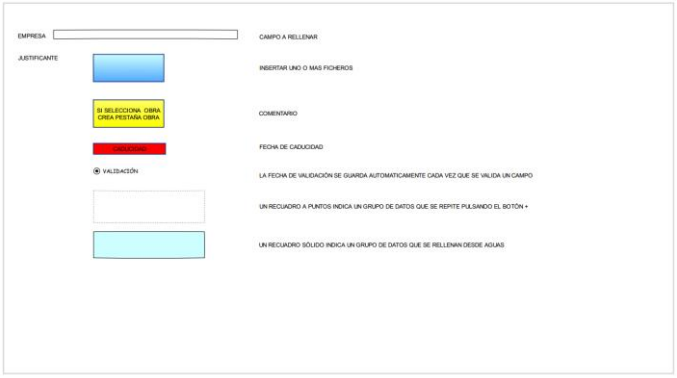


Ilustración 1 Esquema Empresa

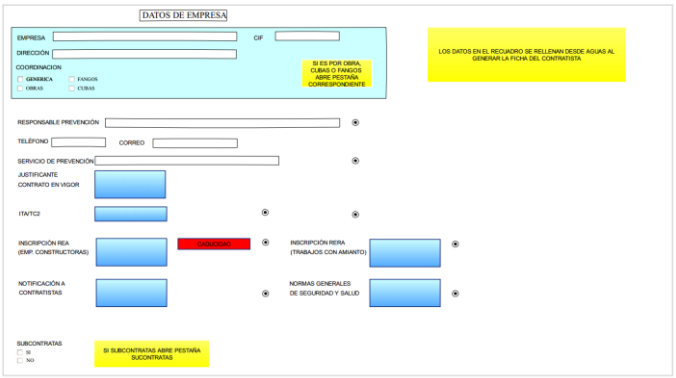


Ilustración 2 Esquema Empresa 2

5.3.2. Esquema Trabajos Especiales

El esquema de Trabajos Especiales define una estructura en la que el cliente desea poder adicionar nuevos Trabajos (mediante el botón +). Los botones azules representan el acceso al documento (archivo PDF o Word por ejemplo) y el botón rojo la fecha de caducidad.

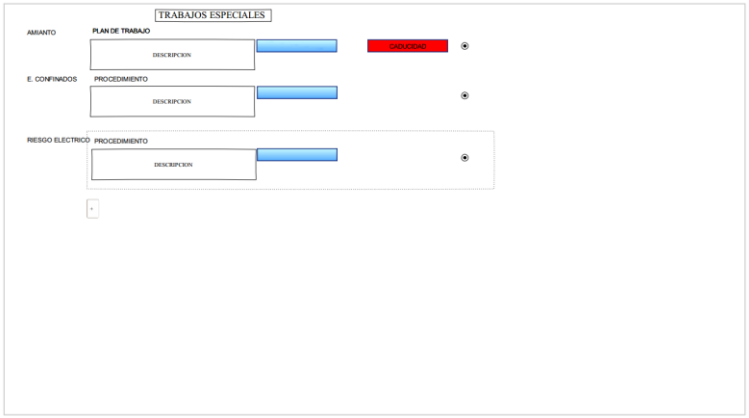


Ilustración 3 Esquema Trabajos Especiales

5.3.3. Esquema de Trabajadores

El esquema de trabajadores propuesto por el cliente continúa la tónica de los esquemas anteriores y presenta campos generales acompañados de campos con los documentos. Nótese el botón de + que permite adicionar nuevos trabajadores. Esta estructura tal y como la define el cliente no es implementable en un desarrollo real por lo que necesitará de un estudio de adaptación.

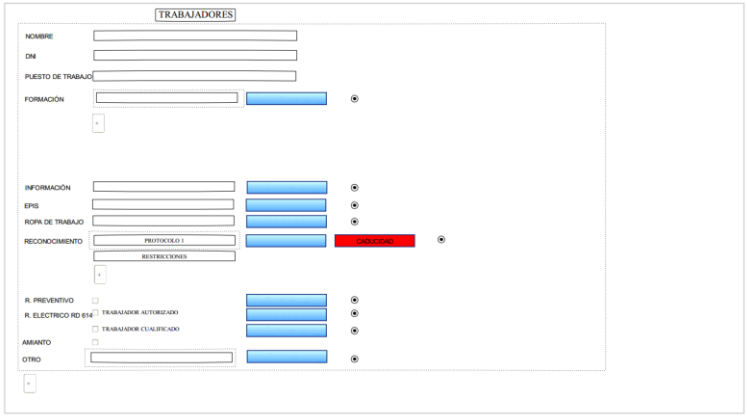


Ilustración 4 Esquema de Trabajadores

5.3.4. Esquema de Equipos y Vehículos

El esquema de Equipos y Vehículos presenta una estructura similar a los anteriores esquemas. Tiene un botón que permite adicionar nuevos equipos / vehículos y tiene botones para cargar los archivos. Al igual que el resto de esquemas no es implementable directamente tal y como lo define el cliente.

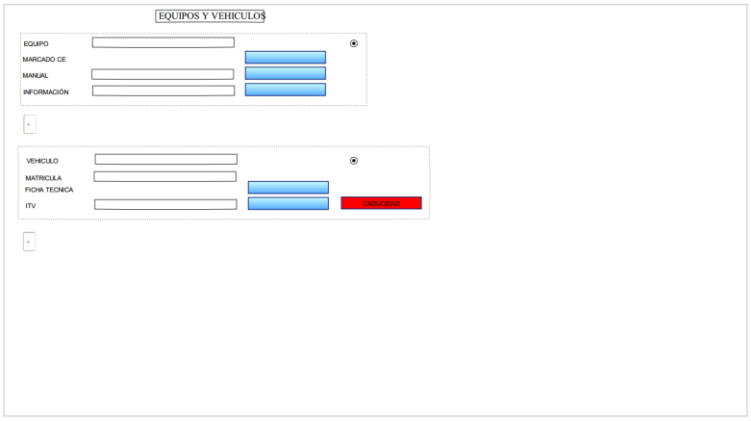


Ilustración 5 Esquema Equipos-Vehículos

5.3.5. Esquema de Obras

El esquema de obras propuesto por el cliente es similar a los anteriores. Tiene la particularidad que el cliente define claramente que hay campos que el contratista no debe poder editar. Tiene un botón + para adicionar obras y un botón para adjuntar documentos. Una vez más tal y como está definido no es implementable por el cliente.

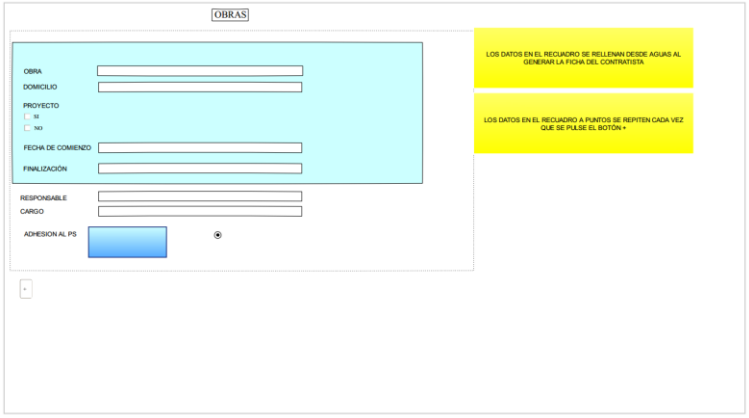


Ilustración 6 Esquema Obras

6. ANÁLISIS Y DISEÑO

6.1. Mockups Aplicación Propuestos

La primera tarea en la fase de diseño consistió en el estudio de los mockups propuestos por el cliente. Se analizaron y se buscó una aproximación que permitiera una implementación real.

El cliente presentaba en sus mockups diversas pantallas de entidades relacionadas entre sí. La entidad principal sería la empresa y de ella dependerían el resto de entidades. El problema del planteamiento del cliente es que la aplicación debe mostrar de forma simultánea una lista de empresas y a su vez poder acceder a todas las entidades dependientes.

La propuesta de diseño pasó por sintetizar todas las entidades en una misma ventana. Se presentaría un selector en la parte superior que permitiría seleccionar la empresa. Una vez seleccionada se presentaría la información de la empresa junto con una serie de desplegables de tipo acordeón que permitirían acceder a la información de las entidades dependientes de empresa (trabajadores, equipos, etc).

Se creó una primera propuesta que se muestra en la imagen siguiente:

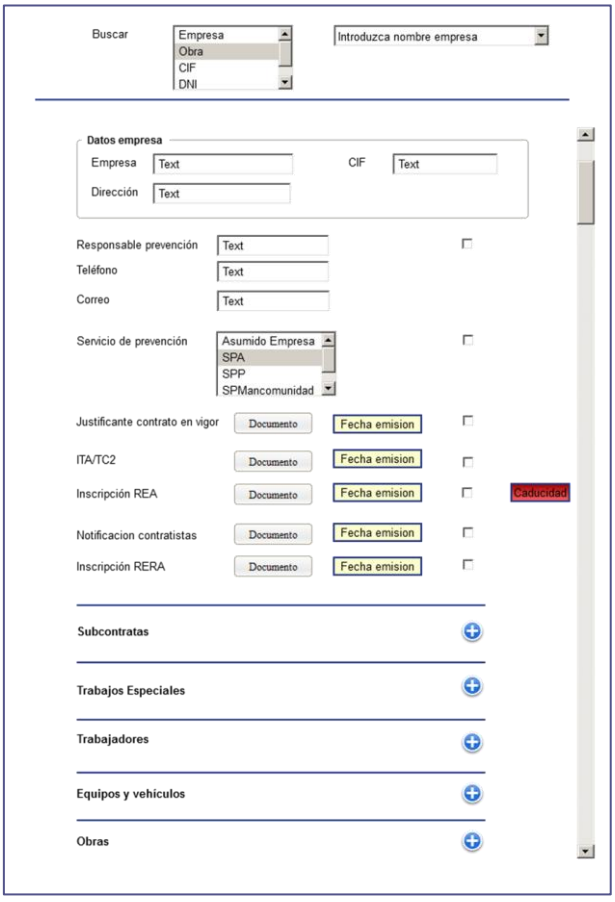


Ilustración 7 Mockup. Primera versión

Posteriormente se inició una fase de investigación en la que el alumno estudió la herramienta devexpress que proporciona controles de interfaz de usuario para asp.net, buscando aquellos controles que pudieran encajar con los mockups propuestos.

Un análisis en profundidad de los mockups junto con los controles de DevExpress reveló que existían opciones más óptimas que permitían mantener la aproximación original del cliente y que a su vez proporcionarían herramientas avanzadas de búsqueda y filtrado de datos.

En ese momento se crearon nuevos mockups que encajaran con este enfoque. En esta última versión se presenta una nueva organización basada en tablas anidadas. A continuación, se presentan los mockups propuestos finales:

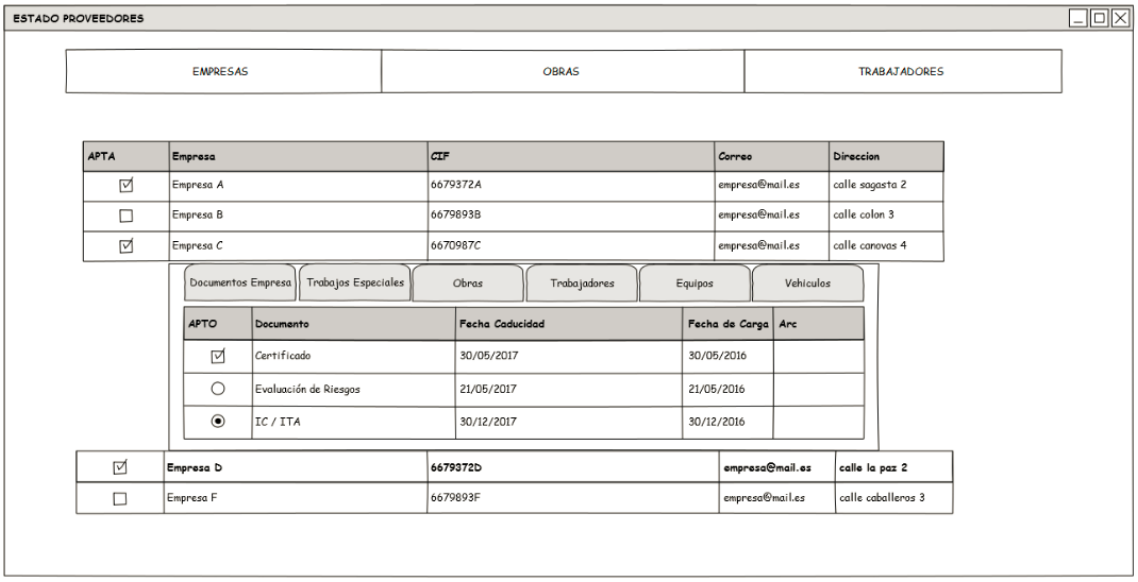


Ilustración 8 Mockup final Empresa

La aplicación se organizará en tres pantallas. La pantalla principal mostrada en la ilustración 8 ofrecerá una lista de empresas en forma de tabla. Tendrá un botón que hará que se desplegué una estructura de pestañas. Tendrá una pestaña por cada tipo de entidad que puede depender de empresa. Dentro de cada pestaña existirá una tabla que almacenará todas las entidades de ese tipo relacionadas con la empresa.

Además, tendrá dos páginas más; una permitirá explorar todos los trabajadores que existen en el sistema con independencia de la empresa a la que pertenezcan y la otra hará lo propio con las obras.

La página de trabajadores viene motivada por la necesidad de tener un explorador de trabajadores donde los usuarios puedan buscar trabajadores cuando desconocen a que empresa pertenecen.

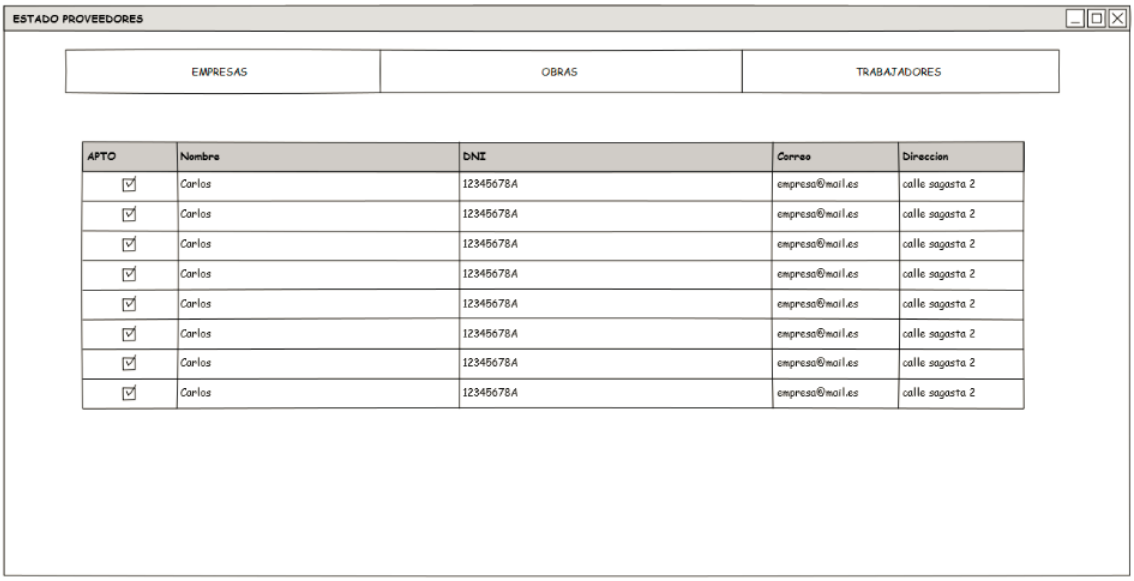


Ilustración 9 Mockup final Trabajadores

La página de Obras tiene una funcionalidad parecida a la de Trabajadores. Debe permitir buscar obras entre todas las existentes con independencia de la empresa a la que pertenecen. Se propone un sistema de pestañas similar al de empresa para mostrar sus documentos y los subcontratistas.

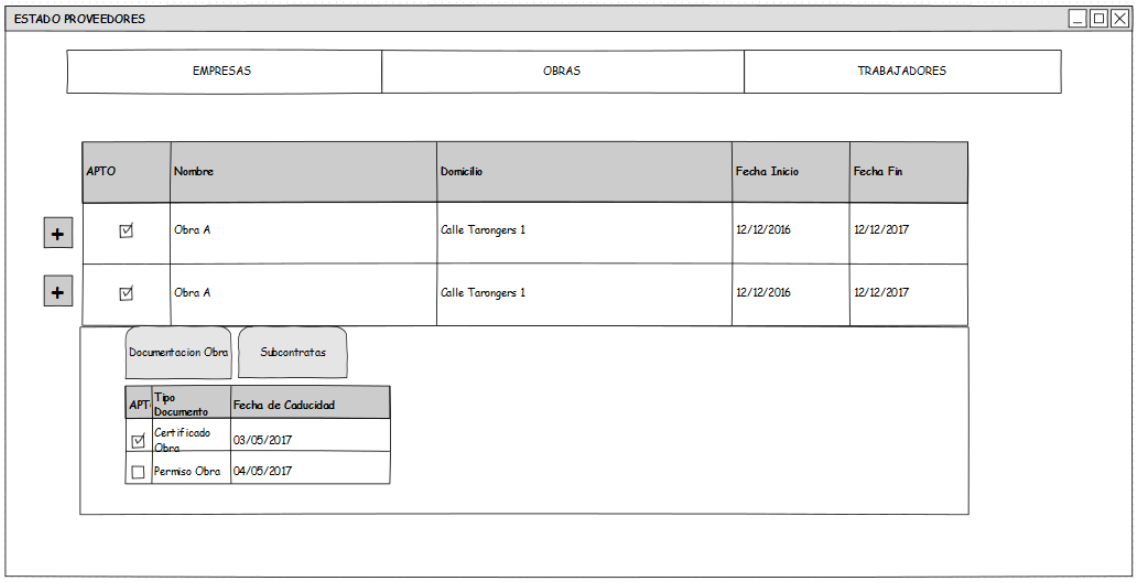


Ilustración 10 Mockup final Obras

Una vez preparados los mockups fueron presentados al cliente para su aprobación.

Todos los mockups han sido realizados con la aplicación de diseño gráfico Pencil.

6.2. Casos de Uso

Los diagramas de casos de uso (10) documentan el comportamiento de un sistema desde un punto de vista de usuario, es decir determinan los requisitos funcionales representando las funciones que un sistema puede ejecutar de forma sencilla. Un componente principal del diagrama de casos de uso es el papel del actor, que representa a un tipo de usuario en el sistema.

Los diagramas de casos de permiten establecer las siguientes relaciones

- **Asociación**: Hay una asociación entre un actor y un caso de uso si el actor interactúa con el sistema para llevar a cabo el caso de uso.
- **Include**: Se puede incluir una relación entre dos casos de uso de tipo “include” si se desea especificar comportamiento común en uno o más caso de uso.
- **Extend**: Se puede incluir una relación entre dos casos de uso de tipo “Extend” si se desea especificar diferentes variantes del mismo caso de uso es decir, la relación “extend” implica que el comportamiento de un caso de uso es diferente dependiendo de ciertas circunstancias.

El sistema tendrá dos tipos de usuario:

- **Usuario Administrador**: representa al personal de AVSA que debe valorar si los documentos, empresas y entidades son aptos. Deben tener acceso a todas las empresas y entidades y deben tener el nivel máximo de permisos. Entre las acciones que pueden realizar se incluye crear, editar y dar de baja documentos y entidades.
- **Usuario Contratista**: representa al contratista. Este usuario sólo tiene acceso a su empresa y a las entidades asociadas. Puede crear, editar y dar de baja documentos.

A continuación, se muestra el diagrama de casos de uso. Este diagrama se ha realizado utilizando la aplicación Microsoft Visio.

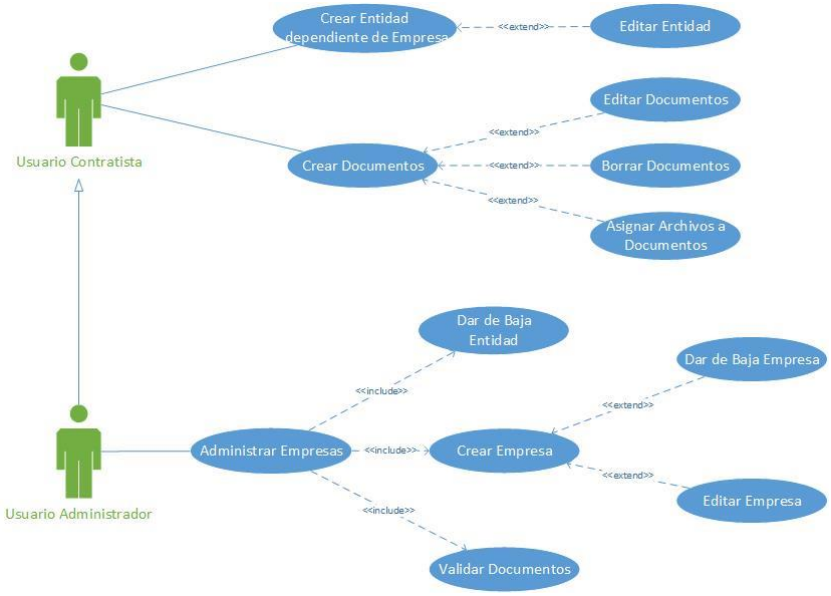


Ilustración 11 Diagrama de Casos de Uso

6.3. Diseño de la Base de Datos

Una base de datos es un “almacén” donde se guarda una colección o conjunto de informaciones (texto, imagen, sonido, video…) las cuales se encuentran relacionadas entre sí y pueden ser accesibles y consultadas en cualquier momento. Entre las principales características de los sistemas de bases de datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación
- Acceso a través de lenguajes de programación estándar.

El diseño de la base de datos es una parte de importancia capital en el desarrollo de cualquier aplicación informática ya que las decisiones tomadas en esta etapa del desarrollo tendrán repercusiones en etapas más avanzadas del mismo. Así una base de datos bien estructurada y con relaciones claras permitirá un desarrollo más cómodo a largo plazo.

En el caso de la aplicación que nos ocupa, la base de datos tenía especial importancia debido al uso del flujo de trabajo database first de entity framework elegido para el desarrollo (ver apartado herramientas). Recordar que esta aproximación permite generar automáticamente el código necesario para el acceso a la base de datos de forma transparente para el usuario.

La base de datos de la aplicación se organiza en torno a una tabla principal que almacena los datos de empresa. Esta tabla se relaciona a su vez con otras tablas, correspondientes a las tablas de entidades. A su vez cada tabla de entidades se relaciona con otra tabla que almacena los datos referentes a los documentos de esa entidad. Para más información acerca de las tablas y el diseño de la base de datos consultar los apartados “Diagrama UML de la Base de Datos” y “Tablas de la Base de Datos”.

6.3.1. Diagrama UML de Base de Datos

El UML (Unified Model Languaje) (11) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación o esquemas de bases de datos.

A continuación, se presenta el diagrama UML de la base de datos. El diagrama ha sido realizado utilizando la herramienta MySQL Workbench (ver apartado “Herramientas”).

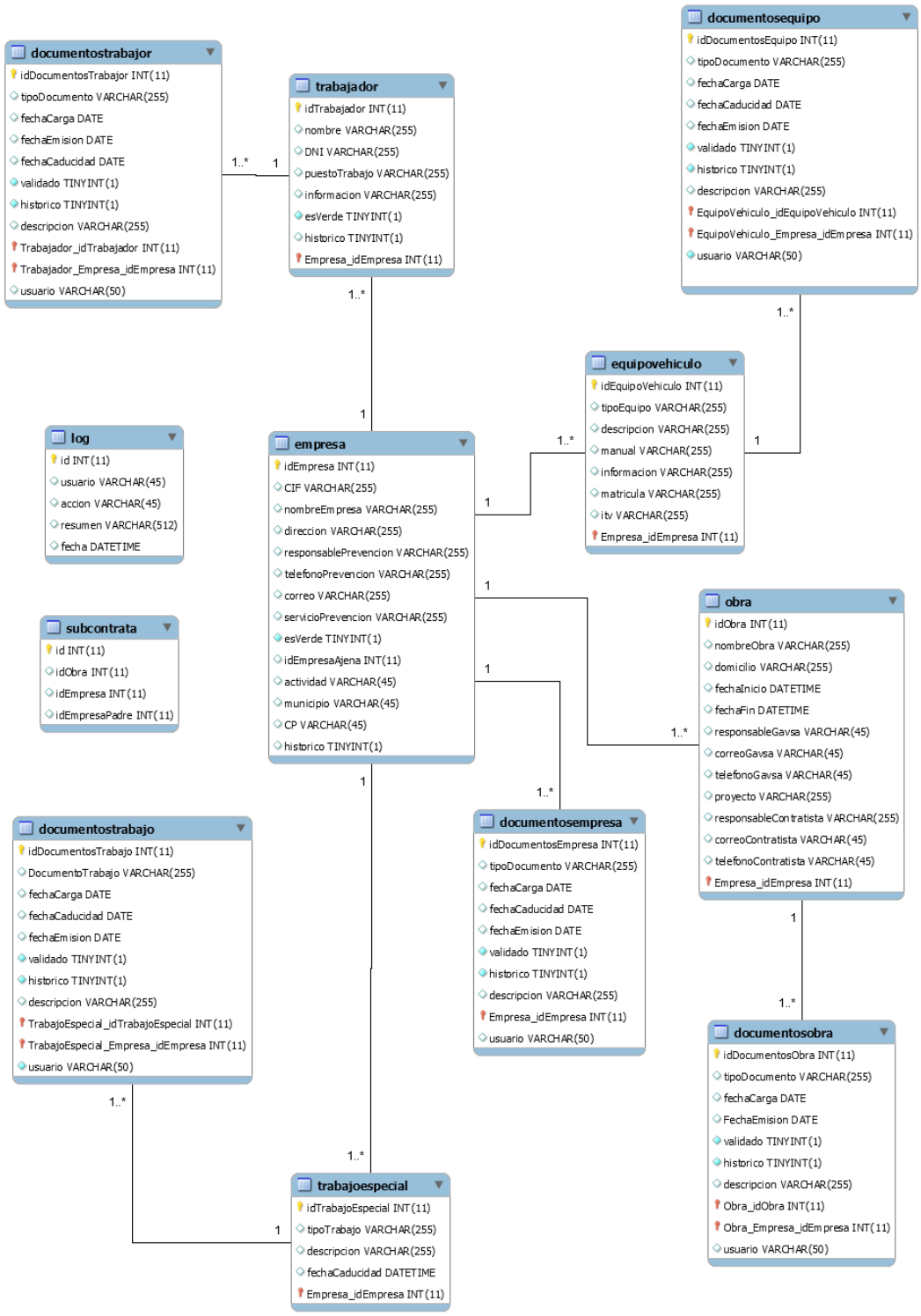


Ilustración 12 Diagrama UML de Base de Datos

6.3.2. Tablas de la Base de Datos

A continuación, se presentan las distintas tablas de la base de datos. Podrían dividirse en tres grupos. El primero contendría a la tabla principal (empresa). El segundo a las tablas de entidades (Trabajador, Obra, Equipo-Vehículo y Trabajo Especial) y el tercer grupo sería las tablas de documento.

En una primera versión de la base de datos se consideró crear una única tabla de documentos y con un identificador categorizar los documentos por tipo pero esta idea tuvo que desecharse debido a que en función de la entidad a la que perteneciera el documento, éste debía tener unos campos u otros, por lo que se optó por crear una tabla para cada tipo de documento. Además, la separación en varias tablas facilitó la programación del código destinado a la validación de documentos que simplificaba en gran medida su implementación. Debe remarcarse que en las tablas no se almacenan los documentos en sí sino campos de información que deben mostrarse en la aplicación. El almacenaje de documentos se realiza en una estructura de carpetas en el servidor (más información en el apartado de implementación).

Además de las tablas anteriores la base de datos contiene dos tablas más; subcontrata y log. Estas tablas tienen como objetivo el almacenamiento de datos auxiliares necesarios para la implementación de ciertas funciones (más información en el apartado de implementación).

A continuación, se presentan al detalle todas las tablas de la base de datos.

6.3.2.1. Tabla Empresa

La tabla de empresas es la tabla principal. Será esta tabla la que funcione de nexo entre todas las demás tablas de la base de datos.

Los campos de empresa son

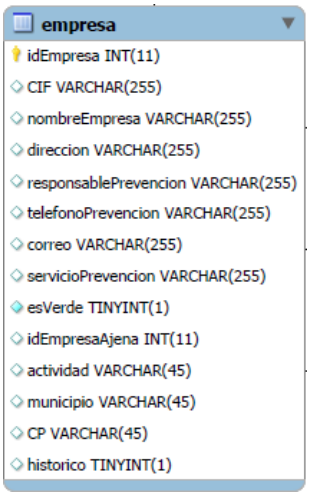


Ilustración 13 Tabla Empresa

- **idEmpresa**: Identificador de empresa de tipo INTEGER. Es la clave primaria
- **CIF**: Código para almacenar el CIF de la empresa, de tipo VARCHAR
- **nombreEmpresa**: almacena el nombre de la empresa. De tipo VARCHAR
- **direccion**: Campo de texto que almacena la dirección de la empresa. De tipo VARCHAR
- **responsablePrevencion**: Campo de texto que almacena el nombre del responsable. De tipo VARCHAR
- **correo**: campo de texto que almacena el correo electrónico de la empresa. De tipo VARCHAR.
- **servicioPrevencion**: aporta información de la empresa. De tipo VARCHAR
- **esVerde**: valor de tipo booleano que identifica a las empresas “verdes”; que son aptas para dar servicio. Ya que MySQL no tiene tipo de datos bolleano el tipo es TINYINT.
- **idEmpresaAjena**: campo auxiliar para relacionar empresas entre ellas. De tipo INTEGER.

- **actividad**: campo de texto que identifica la actividad a la que se dedica la empresa. De tipo VARCHAR
- **municipio**: campo de texto que identifica el municipio al que pertenece la empresa. De tipo VARCHAR.
- **CP**: campo de texto que identifica el código postal de la empresa. De tipo VARCHAR.
- **historico**: campo de tipo booleano que identifica aquellas empresas dadas de baja. De tipo TINYINT.

6.3.2.2. Tabla Trabajador

Esta tabla almacena los datos de los trabajadores. Estos trabajadores se asocian a empresas.

Los campos de la tabla trabajador son:

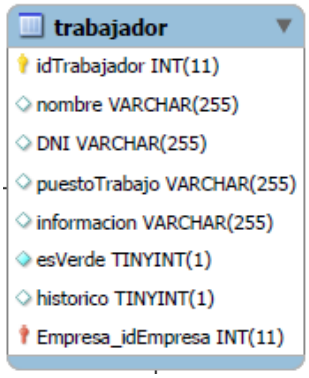


Ilustración 14 Tabla Trabajador

- **idTrabajador**: identificador de empresa de tipo INTEGER. Es la clave primaria.
- **Nombre**: campo de texto que almacena el nombre del trabajador. De tipo VARCHAR
- **DNI**: campo de texto que identifica el DNI del trabajador. De tipo VARCHAR
- **puestoTrabajo**: campo de texto que identifica el puesto de trabajo del trabajador. De tipo VARCHAR.
- **Información**: campo de texto de información. De tipo VARCHAR.
- **esVerde**: campo booleano que identifica a los trabajadores válidos. De tipo TINYINT.
- **historico**: campo booleano que identifica a los trabajadores dados de baja del sistema.
- **Empresa_idEmpresa**: identificador que asocia al trabajador con su empresa. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.

6.3.2.3. Tabla Obra

Esta tabla almacena los datos de las obras. Las obras se asocian a empresas.

Tiene los siguientes campos:

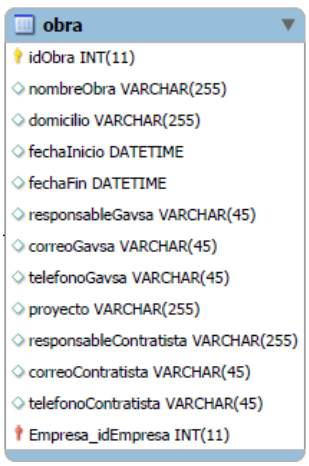


Ilustración 15 Tabla Obra

- **idObra**: identificador de Obra de tipo INTEGER. Es clave primaria.
- **nombreObra**: campo de texto que identifica la obra. De tipo VARCHAR
- **domicilio**: campo de texto que identifica la dirección de la obra. De tipo VARCHAR.
- **fechaInicio**: campo que representa el inicio de la obra. De tipo DATETIME
- **fechaFin**: campo que representa la fecha de fin de la obra. De tipo

DATETIME.

- **responsableGavsa**: nombre del encargado en GAVSA. De tipo VARCHAR.
- **telefonoGavsa**: teléfono del encargado GAVSA. De tipo VARCHAR.
- **proyecto**: campo de información. De tipo VARCHAR.
- **responsableContratista**: campo que representa al responsable de la obra por parte del contratista. De tipo VARCHAR.
- **correoContratista**: campo que representa el correo del responsable de la obra por parte del contratista. De tipo VARCHAR.
- **telefonoContratista**: campo que representa el teléfono del responsable de la obra por parte del contratista. De tipo VARCHAR.
- **Empresa_idEmpresa**: identificador que asocia la obra con su empresa. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.

6.3.2.4. Tabla Equipo Vehiculo

Esta tabla almacena datos de dos entidades, equipos y vehículos. Estas entidades se relacionan con empresas.

Tiene los siguientes campos:

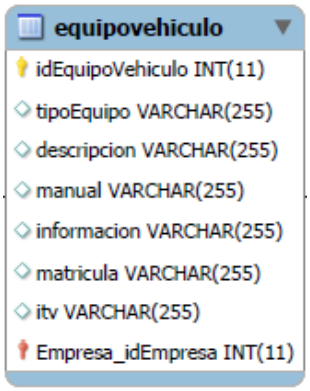


Ilustración 16 Tabla Equipo-Vehículo

- **idEquipoVehiculo**: identificador de Equipo o Vehículo de tipo INTEGER. Es clave primaria.
- **tipoEquipo**: este campo sirve para diferenciar equipos y vehículos. De tipo VARCHAR.
- **descripcion**: campo de información general. De tipo VARCHAR.
- **manual**: campo de información general. De tipo VARCHAR.
- **información**: campo de información general. De tipo VARCHAR.
- **matricula**: campo de información general. De tipo VARCHAR.
- **itv**: campo de información general. De tipo VARCHAR.
- **Empresa_idEmpresa**: identificador que asocia el equipo / vehículo con su empresa. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.

6.3.2.5. Tabla Trabajo Especial

Esta tabla contiene las entidades de tipo trabajo especial. Los trabajos especiales se asocian con empresas.

Tiene los siguientes campos:

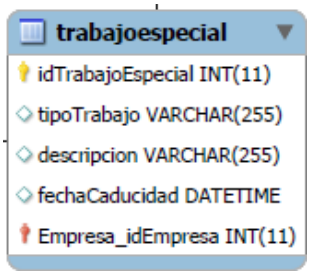


Ilustración 17 Tabla Trabajo Especial

- **idTrabajoEspecial**: identificador Trabajo Especial de tipo INTEGER. Es clave primaria.
- **tipoTrabajo**: este campo sirve para diferenciar equipos y vehículos. De tipo VARCHAR.
- **descripción**: este campo sirve para diferenciar equipos y vehículos. De tipo VARCHAR.
- **fechaCaducidad**: campo que representa la fecha de caducidad del trabajo. De tipo DATETIME.

- **Empresa_idEmpresa**: identificador que asocia el trabajo especial con su empresa. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.

6.3.2.6. Tabla Documentos Empresa

Esta tabla almacena los documentos pertenecientes a empresas. Se asocian directamente con empresas.

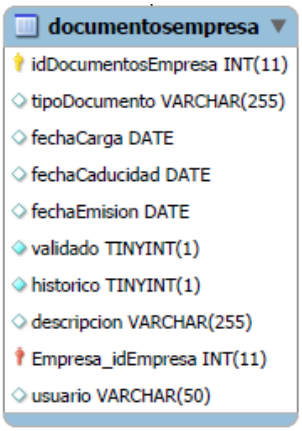


Ilustración 18 Tabla Documentos Empresa

Tiene los siguientes campos:

- **idDocumentosEmpresa**: identificador de documento de tipo INTEGER. Es clave primaria.
- **tipoDocumento**: campo de texto que permite especificar el tipo de documento. De tipo VARCHAR.
- **fechaCarga**: campo que almacena la fecha de carga en el sistema. De tipo DATE.
- **fechaCaducidad**: campo que almacena la fecha de caducidad del documento. De tipo DATE.
- **fechaEmision:** campo que almacena la fecha de emisión del documento. De tipo DATE.
- **validado:** campo booleano que almacena si un documento es válido. De tipo TINYINT.
- **histórico:** campo booleano que se emplea para identificar documentos dados de baja. De tipo TINYINT.
- **descripción:** campo de texto de información general. De tipo VARCHAR.
- **Empreasa_idEmpresa:** identificador que asocia el documento con su empresa. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.
- **usuario**: campo de texto que almacena el último usuario que accedió al registro. De tipo VARCHAR.

6.3.2.7. Tabla Documentos Trabajador

Esta tabla contiene documentos de trabajadores. Se asocian con trabajadores.

Tiene los siguientes campos:

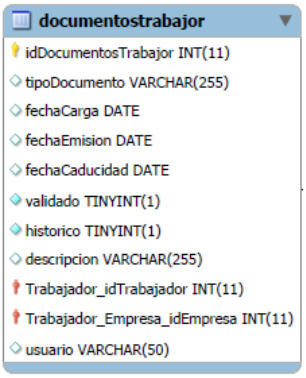


Ilustración 19 Tabla Documentos Trabajador

- **idDocumentoTrabajor**: identificador de documento de tipo INTEGER. Es clave primaria.
- **tipoDocumento**: campo de texto que permite especificar el tipo de documento. De tipo VARCHAR.
- **fechaCarga:** campo que almacena la fecha de carga en el sistema. De tipo DATE.
- **fechaEmision:** campo que almacena la fecha de emisión del documento. De tipo DATE.
- **fechaCaducidad:** campo que almacena la fecha de caducidad del documento. De tipo DATE.

- **validado:** campo booleano que almacena si un documento es válido. De tipo TINYINT.
- **histórico:** campo booleano que se emplea para identificar documentos dados de baja. De tipo TINYINT.
- **descripcion:** campo de texto de información general. De tipo VARCHAR.
- **Trabajador_idTrabajador:** identificador que asocia el documento con su trabajador. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.
- **Trabajador_Empresa_idEmpresa:** identificador que asocia el documento con su empresa. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.
- **usuario:** campo de texto que almacena el último usuario que accedió al registro. De tipo VARCHAR.

6.3.2.8. Tabla Documentos Obra

Esta tabla contiene documentos de obras. Se asocian con obras.

Tiene los siguientes campos:

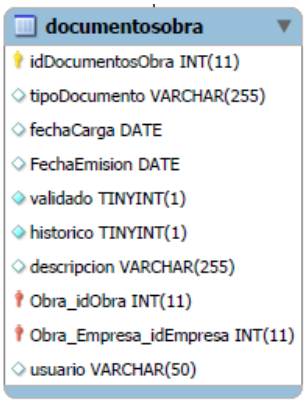


Ilustración 20 Tabla Documentos Obra

- **idDocumentosObra**: identificador de documento de tipo INTEGER. Es clave primaria.
- **tipoDocumento**: campo de texto que permite especificar el tipo de documento. De tipo VARCHAR.
- **fechadeCarga:** campo que almacena la fecha de carga en el sistema. De tipo DATE.
- **fechaEmision:** campo que almacena la fecha de emisión del documento. De tipo DATE.
- **validado:** campo booleano que almacena si un documento es válido. De tipo TINYINT.
- **histórico:** campo booleano que se emplea para identificar documentos dados de baja. De tipo TINYINT.
- **descripción:** campo de texto de información general. De tipo VARCHAR.
- **Obra_idObra:** identificador que asocia el documento con su obra. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.
- **Obra_Empresa_idEmpresa:** identificador que asocia el documento con su empresa. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.
- **usuario:** campo de texto que almacena el último usuario que accedió al registro. De tipo VARCHAR.

6.3.2.9. Tabla Documentos Equipo

Esta tabla contiene documentos de equipos y vehiculos. Se asocian con equipos y vehículos.

Tiene los siguientes campos:

- **idDocumentosEquipo**: identificador de documento de tipo INTEGER. Es clave primaria.
- **tipoDocumento**: campo de texto que permite especificar el tipo de documento. De tipo VARCHAR.
- **fechaCarga:** campo que almacena la fecha de carga en el sistema. De tipo DATE.
- **fechaCaducidad:** campo que almacena la fecha de caducidad del documento. De tipo DATE.
- **fechaEmision:** campo que almacena la fecha de emisión del documento. De tipo DATE.
- **validado:** campo booleano que almacena si un documento es válido. De tipo TINYINT.
- **histórico:** campo booleano que se emplea para identificar documentos dados de baja. De tipo TINYINT.
- **descripcion:** campo de texto de información general. De tipo VARCHAR.
- **EquipoVehiculo_idEquipoVehiculo:** identificador que asocia el documento con su equipo / vehículo. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.
- **EquipoVehiculo_Empresa_idEmpresa:** identificador que asocia el documento con su empresa. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.
- **usuario:** campo de texto que almacena el último usuario que accedió al registro. De tipo VARCHAR.

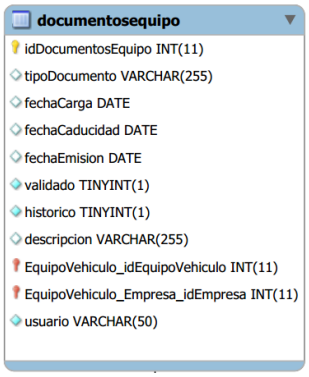


Ilustración 21 Tabla Documentos Equipo

6.3.2.10. Tabla Documentos Trabajo Especial

Esta tabla contiene documentos de trabajos especiales. Se asocian con trabajos especiales:

Tiene los siguientes campos:

- **idDocumentosTrabajo**: identificador de documento de tipo INTEGER. Es clave primaria.
- **DocumentoTrabajo**: campo de texto que permite especificar el tipo de documento. De tipo VARCHAR.
- **fechaCarga:** campo que almacena la fecha de carga en el sistema. De tipo DATE.
- **fechaCaducidad**: campo que almacena la fecha de caducidad del documento. De tipo DATE.
- **fechaEmision:** campo que almacena la fecha de emisión del documento. De tipo DATE.
- **validado:** campo booleano que almacena si un documento es válido. De tipo TINYINT.
- **historico:** campo booleano que se emplea para identificar documentos dados de baja. De tipo TINYINT.
- **descripción:** campo de texto de información general. De tipo VARCHAR.

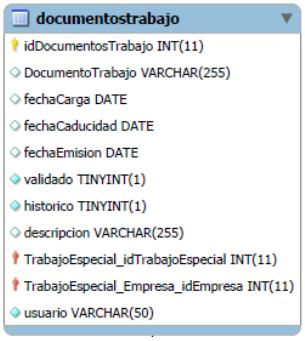


Ilustración 22 Tabla Documentos Trabajo

- **TrabajoEspecial_idTrabajoEspecial:** identificador que asocia el documento con su trabajo especial. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.
- **TrabajoEspecial_Empresa_idEmpresa:** identificador que asocia el documento con su empresa. Almacena el identificador de la empresa a la que pertenece. Es clave ajena.
- **usuario:** campo de texto que almacena el último usuario que accedió al registro. De tipo VARCHAR.

6.3.2.11. Tabla Subcontrata

Esta tabla almacena identificadores que definen relaciones entre empresas subcontratistas.

Tiene los siguientes campos:

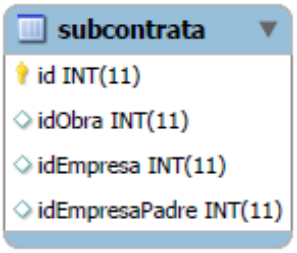


Ilustración 23 Tabla Subcontrata

- **id:** identificador. Es clave primaria. De tipo INTEGER.
- **idObra:** identificador de la obra a la que pertenece la empresa. De tipo INTEGER.
- **idEmpresa:** identificador de la empresa. De tipo INTEGER.
- **idEmpresaPadre:** identificador de Empresa. Representa al contratista principal de la obra.

6.3.2.12. Tabla Log

Esta es una tabla auxiliar que registra todas las transacciones de la base de datos.

Tiene los siguientes campos:

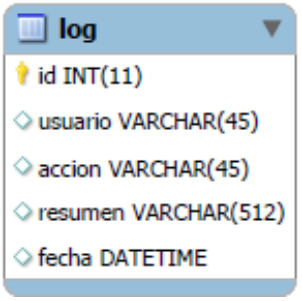


Ilustración 24 Tabla Log

- **id:** identificador. Es clave primaria. De tipo INTEGER.
- **usuario:** campo de texto que identifica al usuario que realiza la acción. De tipo VARCHAR.
- **acción:** campo de texto que almacena la acción que realiza el usuario. De tipo VARCHAR.
- **resumen:** campo de texto que almacena una serie de datos con información general del contexto de la aplicación en el momento de la acción. De tipo VARCHAR.
- **fecha:** campo que almacena la fecha y hora de la acción. De tipo DATETIME.

6.4. Diagrama de Clases

Visual Studio proporciona una herramienta que permite, a partir de un proyecto existente obtener un diagrama de clases. El siguiente esquema fue obtenido utilizando esta característica una vez el proyecto hubo finalizado. En él se aprecian las clases principales y todos sus métodos (se han incluido exclusivamente las clases creadas por el alumno, las clases obtenidas a partir de Web Services o Entity Framework no están incluidas).

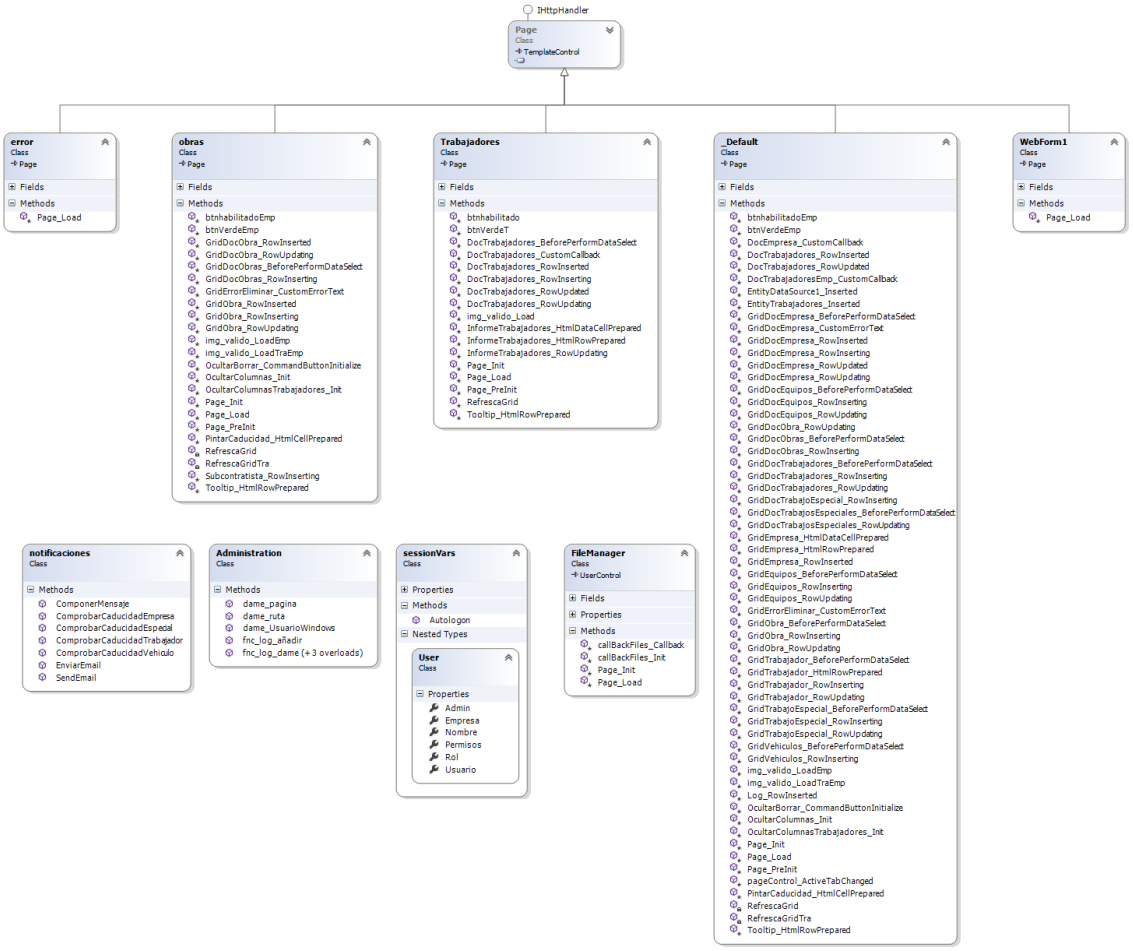


Ilustración 25 Diagrama de Clases

7. IMPLEMENTACIÓN

El presenta apartodo describe el proceso de implementación de la aplicación.

Nota: Algunas capturas de pantalla mostradas a continuación contienen datos reales de empresas y personas. Para proteger su privacidad las capturas han sido modificadas para ocultar los datos sensibles.

7.1. Descripción del proceso de desarrollo y estructura de Webforms.

Antes de comenzar con el proceso de implementación debe describirse la estructura y forma de programación de ASP.NET Webforms.

Webforms permite separación entre el código HTML y de UI (user interface) que se ejecuta en el cliente (el navegador web) y el código de la lógica de la aplicación que se ejecuta en el servidor.

Así el código de la aplicación se divide entre los archivos con extensión .aspx que contienen los controles de la interfaz de usuario y el código html (también llamados páginas) y el llamado code behind, archivos con extensión tipo .cs y que contiene la lógica de la aplicación (clases).

A continuación, se muestra la barra de navegación (llamada Solution Explorer) del IDE Visual Studio para ilustrar la organización de los archivos del proyecto.

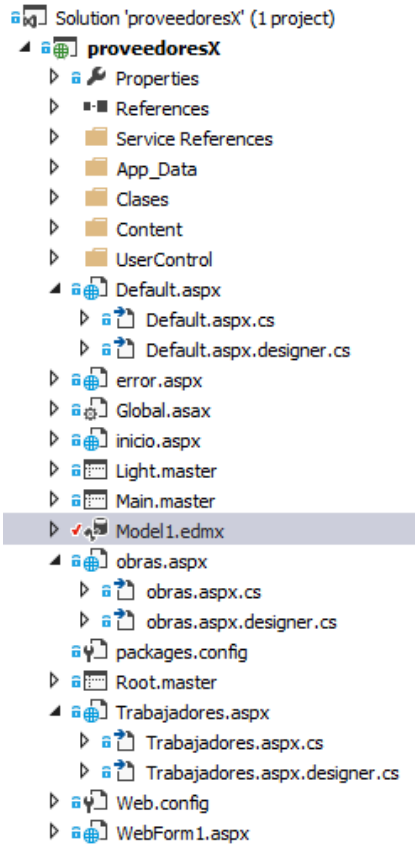


Ilustración 26 Solution Explorer

El otro rasgo identificativo de los Webforms es la programación conducida por eventos. Los eventos son acciones que ocurren en la parte cliente o servidor y que pueden ser manejados por el programador. Esto permite, por ejemplo, crear métodos que “esperen” a que el usuario realice cierta acción sobre un control en el navegador.

Para acceder a la base de datos se utiliza el O/RM (Object-Relational mapping o mapeo objeto relacional) Entity Framework. Los O/RM crean una base de datos orientada a objetos virtual, sobre la base de datos relacional posibilitando el uso de las características propias de la orientación a objetos y reduciendo el código que debe escribir el programador para el acceso a datos. Además tiene ventajas derivadas como que los cambios que se hagan en la base de datos no afectan al código escrito por el programador.

7.2. Implementación de la base de datos

El primer paso en el proceso de implementación es la creación de la base de datos. Para este proceso se partió del diseño UML de la base de datos visto en el apartado de Análisis y Diseño. Este diseño se creó utilizando la herramienta MySQL Workbench mediante una interfaz gráfica.

MySQL Workbench ofrece una opción llamada fordward engineering que permite, a partir del diagrama UML diseñado, crear un script de MySQL que puede ejecutarse desde el mismo MySQL Workbench (este script puede consultarse en el anexo). Una vez ejecutado la base de datos queda constituida y esta lista para usarse.

El siguiente paso es conectar la base de datos al proyecto. Para ello se utilizó el asignador objeto-relacional Entity Framework que proporciona el acceso a datos de forma ágil y rápida, ahorrando al programador la mayoría de código de acceso a datos.

Para realizar la conexión se utilizó el asistente de Visual Studio, una vez añadido el modelo puede consultarse desde el mismo IDE. La siguiente imagen muestra la vista del modelo de datos desde Visual Studio

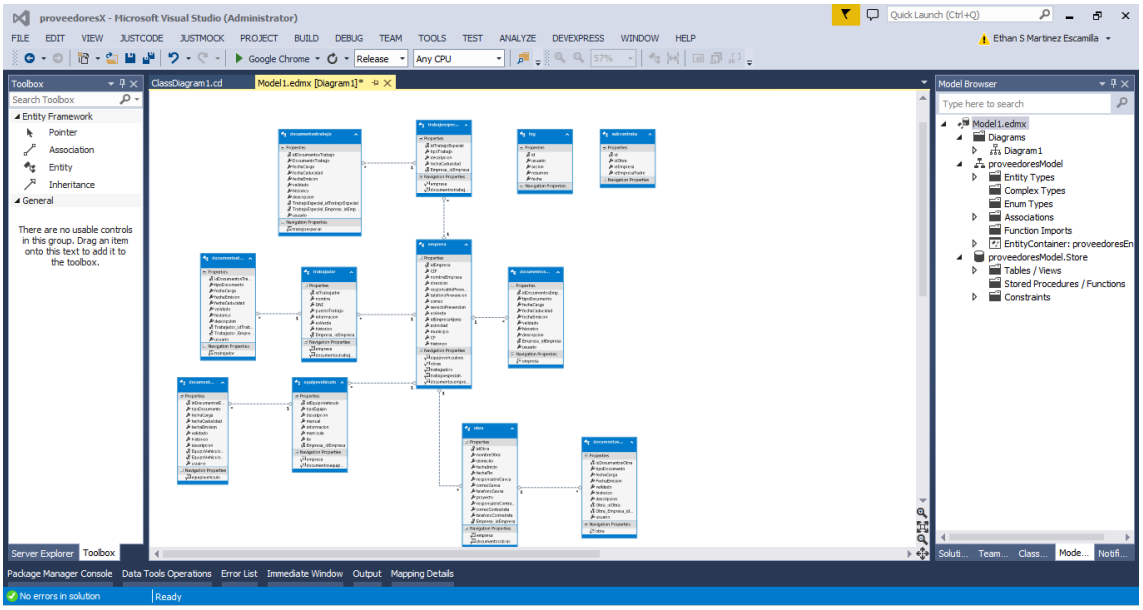


Ilustración 27 Importación de Modelo de Datos

7.3. User Interface (DevExpress)

La parte de interfaz de usuario se realizó utilizando controles de DevExpress.

DevExpress ofrece gran cantidad de controles de interfaz de usuario para múltiples plataformas y tecnologías. En el caso de Webforms ofrece potentes herramientas que agilizan mucho el proceso de programación y diseño de formularios.

El control principal utilizado en la aplicación ha sido el ASPxGridView

Los controles de tipo GridView permiten presentar datos en forma de tabla a partir de un origen de datos o datasource. Este origen de datos se configura para acceder a una tabla concreta de la base de datos. Una vez ligado el GridView a la base de datos es posible realizar operaciones de creación, edición y borrado sobre ella.

Además proporcionan operaciones de filtrado, ordenación y búsquedade datos.

A continuación se adjunta parte del código de la aplicación para ilustrar el uso de los controles de DevExpress (en html).

```
<dx:ASPxGridView ID="DocTrabajadores" ClientInstanceName="DocTrabajadores" runat="server"
DataSourceID="EntityDocTrabajadores"
OnBeforePerformDataSelect="DocTrabajadores_BeforePerformDataSelect">

<Columns>

<dx:GridViewDataTextColumn FieldName="idDocumentosTrabajor" ReadOnly="True" VisibleIndex="0"
Visible="false"></dx:GridViewDataTextColumn>

<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc" VisibleIndex="9">
</dx:GridViewCommandColumn>

<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="10" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>

</Columns>

<SettingsCommandButton>

<NewButton Image-Url="Content/Images/add.png" Text="Nuevo</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" "></EditButton>

</SettingsCommandButton>


</dx:ASPxGridView>
```

El código anterior muestra un ejemplo de GridView. Tiene una cabecera donde se declara el grid y puede asociarse a eventos como `OnBeforePerformDataSelect="DocTrabajadores_BeforePerformDataSelect">`. Este evento está llamando a un método que está en el code behind. El evento del ejemplo, OnBeforePerformDataSelect, ocurre antes de que se busque en la base de datos la información que debe mostrar el GridView También es en la cabecera donde se declara el origen de datos o datasource; `DataSourceID="EntityDocTrabajadores"`

Después entre las etiquetas `<Columns>` entre las etiquetas se declaran las distintas columnas que debe tener el grid. Estas columnas pueden ser de distintos tipos en

función de la información que queramos mostrar. Pueden ser textos, checkboxes, o columnas de control o command columns.

También en cualquier lugar del grid podremos declarar etiquetas de tipo settings que permiten ajustar parámetros de la configuración del grid. `<SettingsCommandButton>`

Una vez descrito el funcionamiento básico de los controles de UI se procede a identificar los archivos principales que definen las páginas de la aplicación:

- **Default.aspx**: es la página principal de la aplicación. Contiene el GridView de empresa y los GridViews anidados de las entidades Trabajadores, Equipos, Vehículos, Obras y Trabajos especiales además de todos los GridViewa de documentos.
- **Error.aspx**: Es una página auxiliar a la que se dirige la aplicación en caso de error, por ejemplo, en el caso de que intente acceder un usuario sin permisos.
- **Obras.aspx**: contiene el explorador de obras. Es un GridView que muestra todas las obras y en GridViews anidados las empresas que participan en ellas.
- **Trabajadores.aspx**: contiene el explorador de trabajadores. Es un GridView que muestra todos los trabajadores con independencia de la empresa a la que pertenecen y en GridViews anidados sus documentos asociados.

Todo el código de la aplicación puede consultarse en el anexo correspondiente.

7.3.1. Página Empresas (Default.aspx)

La página de empresa (Default.aspx) es la página principal de la aplicación.

Debe mostrar una tabla con todas las empresas. Para conseguirlo se recurre al control de DevExpress ASPxGridView.

Primero se debe configurar el origen de datos del GridView, para ellos utilizamos el código siguiente:

```
<asp:EntityDataSource runat="server" ID="EntityDataSource1"
ConnectionString="name=proveedoresEntities"
DefaultContainerName="proveedoresEntities" EnableDelete="True"
EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="empresas" OnInserted="EntityDataSource1_Inserted">

      <InsertParameters>
         <asp:Parameter Direction="ReturnValue" Name="idEmpresa" Type="Int32" />
      </InsertParameters>

</asp:EntityDataSource>
```

El código anterior define que conector (connection string) debe emplear para conectar con la base de datos (este conector está definido en el archivo Webconfig.aspx), define la tabla de la que debe obtener datos (empresa) y mediantes las distintas propiedades define las acciones sobre los datos que el grid puede realizar (`EnableDelete="True", EnableInsert="True",EnableUpdate="True"`). En este caso puede eliminar, actualizar y crear (insertar).

Una vez definido el origen de datos debe definirse el grid como se explica en el apartado implementación.

Una vez definido el GridView de empresa hay que preparar la estructura necesaria para acceder a las entidades dependientes de cada empresa. Es decir, cada línea o row del GridView tiene asociadas las entidades definidas en apartados anteriores, a saber: Documentos Empresa, Trabajadores, Trabajos Especiales, Obras, Equipos y Vehículos.

Además, las entidades Trabajadores, Trabajos Especiales, Obras, Equipos y Vehículos deberán tener a su vez documentos asociados.

Ilustración 28 Ejemplo ASPxGridView

Para ello se empleará la estructura Detail Row.

Un Detail Row es un desplegable que permite adicionar controles de cualquier tipo debajo de una fila o row, de forma anidada.

En el caso de empresa ese desplegable mostrará una serie de pestañas. Para definir las pestañas se utiliza el control de pestañas TabPages. Existirá una pestaña por cada tipo de entidad dependiente de Empresa.

Dentro de cada una de las pestañas existe un GridView que muestra las entidades de ese tipo relacionadas con la empresa. A su vez cada uno de estos GridViews deberá tener su propio Detail Row para almacenar los documentos dependientes de él.

Es decir, la aplicación basa su comportamiento en una estructura de GridViews anidados que muestran las entidades relacionadas entre sí. Esta estructura de GridViews anidados tendrá tres niveles como muestra el siguiente gráfico:

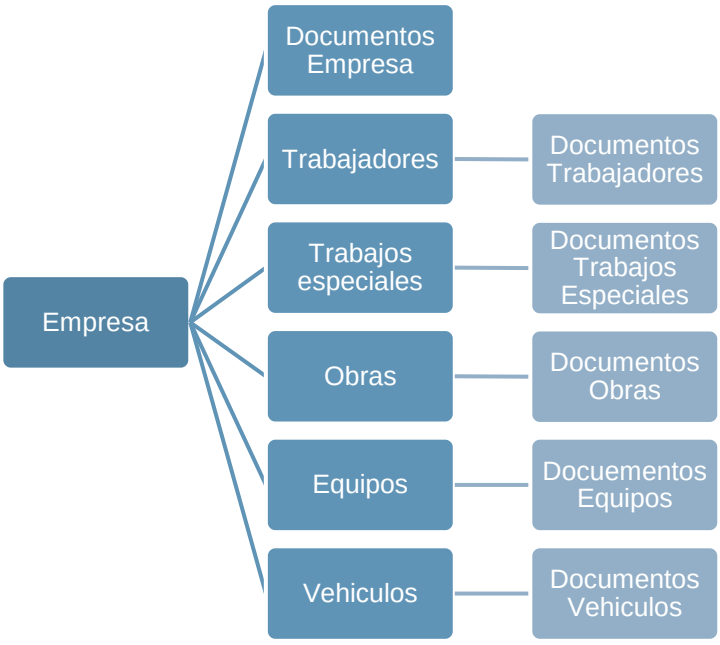


Ilustración 29 Jerarqui GridViews

Por ejemplo, supongamos que el usuario de la aplicación quiere consultar la ITV de un vehículo perteneciente a la empresa X.

Para ello primero deberá buscar la empresa X en el GridView principal. Después desplegará el Detail Row de esa empresa y accederá a la estructura de pestañas. Deberá navegar hasta la pestaña ”Vehículos” que mostrará un GridView con todos los vehículos pertenecientes a la empresa X.

Una vez localizado el vehículo accederá a su Detail Row, lo que mostrará otro GridView que presentará todos los documentos relacionados con ese vehículo. Sólo deberá navegar hasta el documento ITV y consultar la información deseada.

La definición de las pestañas y de los GridViews anidados se realiza dentro del mismo GridView de empresa. Cada uno de estos GridViews anidados deberá tener su propio origen de datos o Datasource. Debe tenerse en cuenta que si se configura el origen de datos por defecto se mostrarán todas las entidades que existan en las tablas. Para corregir eso debe modificarse el origen de datos de forma que sólo muestre los datos que interese en cada momento. Para ello se recurre a la clave ajena que se especificó en la definición de la base de datos.

Por ejemplo para mostrar los vehículos pertenecientes a una empresa en el datasource se define un where parameters que compara la clave ajena de Vehiculos (Empresa_idEmpresa) con la clave primaria de la tabla empresa (id_Empresa). Si estos campos coinciden significa que el vehículo pertenece a esa empresa.

El código para implementar esa funcionalidad es el siguiente:

```
<asp:EntityDataSource ID="EntityVehiculos" runat="server"
ConnectionString="name=proveedoresEntities"
DefaultContainerName="proveedoresEntities" EnableDelete="True"
EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="equipovehiculoes" Where="it.[Empresa_idEmpresa] = @paridEmpresa
AND it.[tipoEquipo]= @tipo">
 <WhereParameters>
 <asp:SessionParameter DbType="Int32" Name="paridEmpresa"SessionField="idEmpresa"
/>
 <asp:SessionParameter DbType="String" Name="tipo" SessionField="tipo" />
 </WhereParameters>
</asp:EntityDataSource>
```

De esta forma el DataSource hace la comprobación antes de recuperar los datos de la base de datos.

Este procedimiento se realiza para cada uno de los GridViews anidados.

El resultado final se muestra en la siguiente captura:

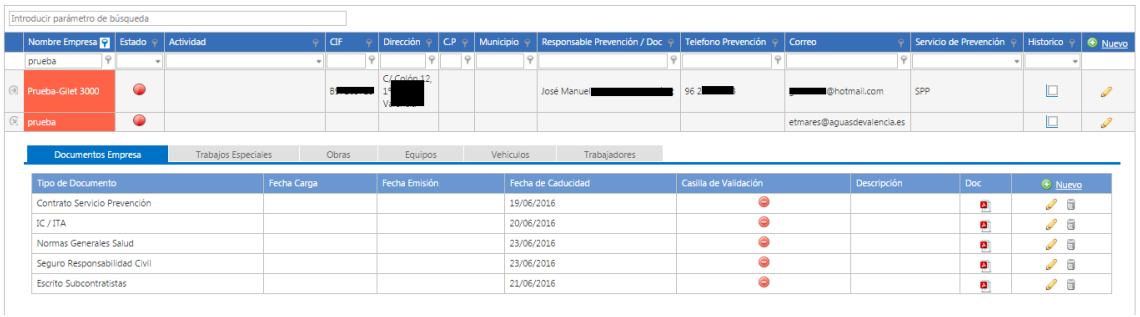


Ilustración 30 Detalle Detail Row

La siguiente captura muestra el detalle del botón que permite acceder al Detail Row

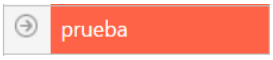


Ilustración 31 Botón Detail Row

7.3.2. Página Trabajadores (Trabajadores.aspx)

La página de Trabajadores es más sencilla que la página de empresa. Deberá mostrar un GridView que muestre todos los trabajadores con independencia de la empresa. Es decir, su datasource deberá seleccionar todas las filas de la tabla Trabajadores de la base de datos.

Esta página también deberá permitir visualizar los documentos asociados a los trabajadores. Para ello se empleará un Detail Row de forma similar a lo visto en el apartado anterior. Este Detail Row contendrá un GridView de Documentos de Trabajadores con un origen de datos o datasource modificado para mostrar sólo aquellos documentos que pertenezcan al trabajador.

A continuación, se muestra una captura de pantalla del resultado final:

Ilustración 32 GridView Trabajadores

7.3.3. Página Obras (Obras.aspx)

La página de obras tiene una organización similar a los apartados anteriores.

Cuenta con un GridView principal que muestra todas las obras, con independencia de la empresa a la que pertenecen. Tiene también un Detail Row con dos pestañas. Una para documentos de Obras y otra llama Subcontratistas. Esta última pestaña permite relacionar empresas entre sí. Recordando el capítulo de especificación de requisitos las obras pueden tener un contratista principal y este a su vez subcontratar más contratistas. La pestaña Subcontratistas permite plasmar esta relación. Cuenta con dos desplegables, uno para elegir el contratista que participa y otra para seleccionar que contratista lo ha contratado.

Para implementar la página obras se ha seguido los procedimientos descritos en el apartado Página Empresa.

Ilustración 33 GridView Obras

7.3.4. Funcionalidades auxiliares y configuración de estilos.

Este apartado tiene como objetivo describir otras funcionalidades de la UI (user interface) como el control de semáforos o el resaltado de documentos caducados. Además, se especificará todo lo relacionado con los estilos y apariencia visual de los GridViews.

7.3.4.1. Control de Semáforos

Los semáforos son una ayuda visual que ayuda a los usuarios de la aplicación a discriminar entre las entidades válidas y las que no lo son. Los semáforos están presentes en Empresa y Trabajador.

Ilustración 34 Control de Semáforos

Tienen dos posibles posiciones, rojo y verde. Los criterios de color se definen de la siguiente forma:

- Para empresa: Una empresa es verde si todos sus documentos y sus trabajadores son válidos. Es roja en caso contrario.
- Para trabajadores: Un trabajador es verde si todos sus documentos son válidos. Es rojo en caso contrario.

Para que un documento sea válido éste debe cumplir las siguientes condiciones:

- Si tiene fecha de caducidad esta debe ser mayor que la fecha actual.
- Debe haber sido validado por un usuario administrador (de forma manual).

En la siguiente imagen se aprecia el detalle de documentos de trabajadores y la casilla de validación que debe cumplimentar el usuario.

Tipo de Documento	Fecha Carga	Fecha Emisión	Fecha de Caducidad	Casilla de Validación	Descripción	Doc	Nuevo
Reconocimiento Médico	16/02/2016	16/03/2015	16/03/2016		Apto médico		
Hoja Entrega de EPIS	16/02/2016	05/03/2015			Epis		
Formación	16/02/2016	18/12/2012			Formación 20 h		
Entrega Información	16/02/2016	05/03/2015			Información PRL		

Ilustración 35 Validación de Documentos

7.3.4.2. Resaltar entidades históricas

Es posible que aparezcan situaciones en las que una empresa o trabajador deba darse de baja porque ya no van a ser requeridos sus servicios. Sin embargo, los datos y documentos pertenecientes a estas entidades deben permanecer en el sistema ya que por la legislación vigente estos datos deben ser almacenados. Por eso la aplicación no permite eliminar de forma definitiva ciertas entidades. En su lugar se definió el campo histórico que permite marcar aquellas entidades que no deben usarse. Este campo permite además discriminar a los usuarios de la aplicación entre entidades vigentes y no vigentes. Para ello pueden filtrar las búsquedas en el GridView de forma que vean o no a las entidades históricas.

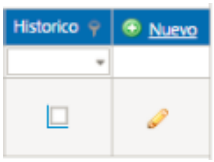


Ilustración 36 Detalle campo Histórico

Este campo también permite al programador discriminar entidades para, por ejemplo, tenerlas en cuenta para el control de semáforos.

Además, se implementó un sistema visual que resalta aquellas entidades históricas de forma que el usuario pudiera reconocerlas rápidamente. Para ello bastó con modificar el color del fondo de la fila que contiene la entidad en el GridView.

En la siguiente imagen se aprecian dos entidades históricas:

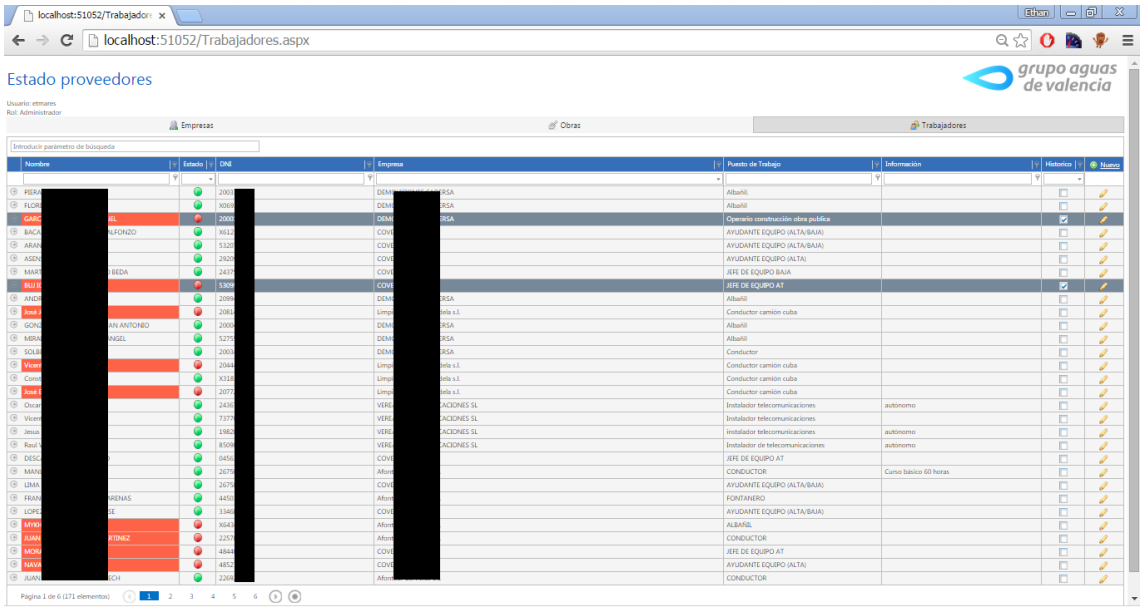


Ilustración 37 Ejemplo Entidades Históricas

7.3.4.3. Habilitar Filtros y Barra de Búsqueda

Los controles GridView de DevExpress permiten habilitar opciones de filtrado avanzadas que permiten al usuario localizar datos de forma rápida y certera. Para lograr esto basta con modificar los parámetros necesarios en la etiqueta Settings del GridView:

```
<SettingsDetail ShowDetailRow="True" />
        <SettingsSearchPanel Visible="True"></SettingsSearchPanel>
        <SettingsBehavior ConfirmDelete="True"></SettingsBehavior>
        <SettingsEditing EditFormColumnCount="1" Mode="Inline" />
        <Settings ShowFilterBar="Visible" ShowFilterRow="true"
             ShowFilterRowMenu="true" ShowFilterRowMenuLikeItem="true"
             ShowHeaderFilterButton="true" />
        <SettingsText    SearchPanelEditorNullText="Introducir    parámetro    de
búsqueda" />
```

7.3.4.4. Resaltar Fechas de documentos caducadas

De forma similar al resaltado de entidades históricas la aplicación es capaza de detectar si un documento está caducado. En caso de que así sea la celda con la fecha de caducidad cambia de color para facilitar su identificación al usuario de la aplicación.

En la siguiente imagen puede apreciarse esta funcionalidad:

Tipo de Documento	Fecha Carga	Fecha Emisión	Fecha de Caducidad	Casilla de Validación	Descripción	Doc	Nuevo
Contrato Servicio Prevención	11/11/2015		30/09/2016		Especialidades técnicas MC Prevención.		
Contrato Servicio Prevención	11/11/2015		13/03/2016		Vigilancia de la salud. MC Prevención		
Normas Generales Salud	11/11/2015				Normas de seguridad Grupo Aguas		
IC / ITA		06/11/2015			ITA, TC2 actualizado noviembre.		
Evaluación de Riesgos					Evaluación 2014, evaluación embarazada		
Otros					Escrito a subcontratistas firmado		
Otros			01/01/2016		Seguro convenio, recibo		
Seguro Responsabilidad Civil			01/01/2016		Seguro RC		
Contrato Servicio Prevención	15/02/2016						
Certificado Pagos a Hacienda	20/04/2016	15/03/2016			prueba de certificado		

Ilustración 38 Detalle Caducidad Documentos

7.3.4.5. Modificaciones Estéticas de los GridViews

Los controles GridView de DevExpress ofrecen avanzadas opciones de personalización de forma que el aspecto y apariencia de los controles son totalmente personalizables por el programador. Para ello se utilizan sentencias como la que se muestra a continuación:

```
<Styles>
   <Header BackColor="#376EB8" ForeColor="White"></Header>
   <Row BackColor="WhiteSmoke"></Row>
</Styles>
```

7.4. Code Behind

El Code Behind es el código que se ejecuta en la parte del servidor. Se refiere a aquellas clases que acompañan a las páginas. Está programado en C# y es donde pueden depositarse los métodos de la aplicación. Estos métodos pueden ser llamados desde el cliente mediante los eventos. El código se organiza en clases (archivos con extensión .cs) y están ligadas a una página (archivo .aspx).

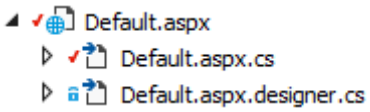


Ilustración 39 Detalle Code Behind

A continuación, se adjunta un método de ejemplo (se corresponde con el ejemplo del apartado anterior)

```
protected void GridDocTrabajadores_BeforePerformDataSelect(object sender,
EventArgs e)
{

Session["idTrabajadores"] = (sender as ASPxGridView).GetMasterRowKeyValue();

}
```

El código anterior es llamado desde el cliente en el evento `OnBeforePerformDataSelect="DocTrabajadores_BeforePerformDataSelect">` y se encarga simplemente de asignar un valor a una variable. Este evento ocurre antes de que el GridView recupere los datos a mostrar del DataSource. Así existen multitud de eventos

diferentes que podemos definir en la cabecera del GridView y luego ligar con código C# almacenado en la clase de la página del Code Behind.

En los siguientes sub-apartados se describen brevemente los métodos empleados en el code behind. Se encuentran agrupados según a qué página pertenecen. Además en el apartado Otros Métodos se enmarcan aquellos generales que se encuentran en clases convencionales.

7.4.1. Métodos principales de Página Empresa

Este apartado describe aquellos métodos más importantes de la Página Empresa. Muchos de ellos no se describen aquí ya que deben crearse por cada uno de los GridViews de la página y su comportamiento y definición es similar entre ellos. El código de todos estos métodos y de la aplicación completa puede consultarse en el Anexo.

Recordar que todos estos métodos se llaman desde eventos que ocurren en la página.

- **GridEmpresa_HtmlDataCellPrepared**: Este método lo llama el evento HtmlDataCellPrepared y cambia el color de las empresas inválidas (de la celda con el nombre) a rojo.

- **GridEmpresa_HtmlRowPrepared**: Este método lo llama el evento HtmlRowPrepared y cambia el color de la fila cuando la empresa está marcada como histórica.

- **GridEmpresa_RowInserted**: Este método se ejecuta cuando ocurre el evento RowInserted. Al insertar una fila se llama al método fnc_log_añadir y registra la operación de inserción en el log de operaciones.

- **GridTrabajoEspecial_BeforePerformDataSelect:** recupera el valor del id padre (La empresa a la que pertenece el trabajador) y lo asigna a una variable de sesión para poder usarse en el DataSource del GridView. Se llama en el evento BeforePerformDataSelect.

- **GridTrabajoEspecial_RowInserting**: este método lo llama el evento RowInserting y en él se asigna la clave ajena (id de empresa) a la entidad que se está insertando en la tabla.

- **OcultarColumnas_Init**: este método, llamado en el evento PageInit permite ocultar columnas de los GridView de la página (como la columna de edición) en caso de que el usuario no sea administrador.

- **RefrescaGridTra**: este método fuerza el refresco de los datos del grid. Se utiliza por ejemplo cuando el usuario edita un documento y se desea forzar la recarga de datos para efectuar validaciones en las entidades padre.

- **PintarCaducidad_HtmlCellPrepared:** este método, que se ejecuta en el evento HtmlCellPrepared pinta la celda de fecha de caducidad en los grid de documentos de un color resaltado cuando están caducados.

- **Tooltip_HtmlRowPrepared:** esté método, que se ejecuta en el evento HtmlRowPrepared muestra una etiqueta contextual que aparece cuando el usuario desplaza el cursor sobre una fila del GridView. En nuestro caso se muestra por pantalla el nombre del último usuario que editó la fila.

7.4.2. Métodos principales de Página Trabajadores

A continuación, se detallan los métodos principales de la página Trabajadores.

- **InformeTrabajadores_RowUpdating:** este método, ejecutado en el evento RowUpdating tiene como objetivo corregir un pequeño bug que ocasionaba la pérdida de información al actualizar una fila. Para corregirlo se fuerza la asignación de valores para los campos problemáticos.

- **DocTrabajadores_BeforePerformDataSelect:** este método permite filtrar los documentos pertenecientes a la entidad padre (trabajador) para mostrar en el GridView anidado aquellos que pertenecen a ese trabajador. Para ello se compara la id del Trabajador con la clave ajena del documento (deben ser iguales).

- **InformeTrabajadores_HtmlDataCellPrepared:** este método permite pintar de rojo la celda del trabajador no válido. Se ejecuta en el evento HtmlDataCellPrepared.

- **DocTrabajadores_CustomCallback:** este método se ejecuta en el evento OnCustomCallBack y permite llamar a los métodos que validan automáticamente la fecha de caducidad de los documentos y la validez de las entidades padre (en este caso Trabajador.)

7.4.3. Métodos principales de Obras

- **GridObra_RowUpdating:** este método, ejecutado en el evento RowUpdating tiene como objetivo corregir un pequeño bug que ocasionaba la pérdida de información al actualizar una fila. Para corregirlo se fuerza la asignación de valores para los campos problemáticos

- **GridDocObras_BeforePerformDataSelect**: este método permite filtrar los documentos pertenecientes a la entidad padre (obra) para mostrar en el GridView anidado aquellos que pertenecen a esa obra. Para ello se compara la id de la Obra con la clave ajena del documento (deben ser iguales).

7.4.4. Otros Métodos

En la aplicación se utilizan otros métodos en clases independientes de carácter auxiliar. Tienen como objetivo realizar operaciones comunes dentro de otros métodos de la aplicación.

- **trabajadorValido:** este método comprueba si un trabajador es válido revisando que todos los documentos que dependan de él hayan sido validados y no estén caducados.

- **CambiarValidezDocumento:** este documento revisa las fechas de caducidad de los documentos, de forma que si se hubiera sobrepasado los convierte marca como no válidos.

- **empresaValido:** este método revisa si las empresas son válidas. Para ello comprueba que todos sus documentos son hayan sido validados y no estén caducados, por un lado, y que todos los trabajadores que dependen de la empresa sean válidos.

- **DeshabilitarEdicionGrid:** este método permite deshabilitar la edición, inserción y borrado en el GridView que se desee. Se utiliza en los casos en que el usuario no es administrador y no debe poder realizar ese tipo de operaciones.

7.5. Integración de los web services

La aplicación recurre al uso de diversos Web Services. El presente apartado tiene como objetivo definir los Web Services empleados y especificar las funcionalidades que los utilizan.

Un servicio web (en inglés, Web Service) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores.

Estos web services fueron desarrollados por AVSA para su uso interno y son herramientas de uso común en todas las aplicaciones informáticas del grupo.

Los web services utilizados han sido tres:

- **Permisator**
- **Gestor de Notificaciones**
- **Usuarios Service**

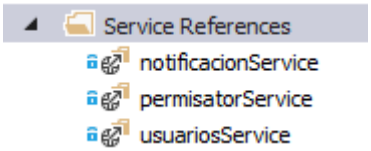


Ilustración 40 Detalle Web Services

7.5.1. Permisator

Se trata de una aplicación de desarrollo propio que permite un control de permisos de usuario centralizado para todas las aplicaciones de AVSA. Permite definir para cada aplicación Niveles de acceso, qué usuarios tienen acceso y a qué nivel y definir para cada usuario “roles”. Los roles son variables que deciden los programadores para ser

utilizadas más adelante. Por ejemplo, podríamos definir un rol llamado “Notificaciones” para un usuario. Otra aplicación del grupo utilizaría el Web Service de Permisator para identificar los roles asignados a ese usuario. Al encontrar el rol “Notificaciones” podría interpretarlo, por ejemplo, como que ese usuario está suscrito al servicio de avisos por email de la aplicación.

En la aplicación objeto del presente trabajo el servicio de Permisator se utiliza para el control de acceso a la aplicación.

La aplicación detecta el usuario de Windows que intenta acceder a la aplicación. Después se utiliza el servicio web para buscar a ese usuario entre los usuarios dados de alta en la aplicación. En caso de no existir se redirige al usuario a una página de error en la que se le informa que no tiene permisos de acceso.

En caso de sí existir se consulta mediante el Web Service que permisos tiene el usuario. Puede detectar dos tipos conforme se especificó en el apartado XX:

- **Administrador:** en caso de ser administrador se presentan en las páginas todos los elementos.
- **Usuario:** en caso de no ser administrador se restringe el DataSource del grid de empresas. Sólo tendría acceso a la empresa que tenga asignada el usuario en Permisator. Además, se restringen otros parámetros como ocultar algunas columnas del grid y eliminar la columna de control. Lo mismo ocurre con las páginas de trabajadores y obras, teniendo accesible sólo los trabajadores y obras de su empresa.

Para realizar estas comprobaciones se debe primero identificar al usuario que intenta acceder a la página. Esto puede hacerse en el Code Behind llamando al método autologon(). Este método utiliza el Web Service de Permisator para obtener nombre y permisos del usuario.

El método se ubica dentro del evento Page_PreInit para identificar el antes de cargar la página.

```
protected void Page_PreInit(object sender, EventArgs e)
{
 sessionVars.Autologon();
}
```

Una vez identificado al usuario debe comprobarse sus permisos para ajustar la configuración de los controles de la página antes de cargarlos. Para ello utilizamos el código siguiente en el evento Page_Init:

```
protected void Page_Init(object sender, EventArgs e)
{
    if (!sessionVars._UserLogged.Admin)
    {
```

```
        EntityInformeTrabajadores.Where = "it.[Empresa_idEmpresa] = " +
        sessionVars._UserLogged.Empresa;

        cls_proveedores.DeshabilitarEdicionGrid(InformeTrabajadores);
      }
}
```

El código anterior primero comprueba si el usuario autentificado en la aplicación es administrador. En caso de no serlo se entra en el if y se modifica el Datasource del grid de la página (en el caso del ejemplo EntityInformeTrabajadores) para que muestre sólo la empresa asociada al usuario. También se utiliza el método DeshabilitarEdicionGrid que elimina la columna de control del grid principal de la página, eliminando los controles de creación, edición y eliminación.

7.5.2. Gestor de Notificaciones

El gestor de notificaciones es un Web Service que aporta una serie de métodos que permiten al programador enviar emails a los usuarios de la aplicación.

La aplicación del presente trabajo debe permitir notificar con cierta antelación a los usuarios administradores cuando un documento va a caducar. En caso de detectar un documento que vaya a caducar se enviará un email de forma automática a estos usuarios.

Para ello se utilizan tres tipos de métodos diferentes todos ellos contenidos en la clase Notificaciones.cs

El primer método comprueba la caducidad de los documentos. Existe un método por cada tipo de documento que queremos comprobar. El código es el siguiente:

```
public static void ComprobarCaducidadEmpresa()
{
 using (proveedoresEntities context = new proveedoresEntities())
 {
     try
     {

        DateTime caducidad = DateTime.Now.AddMonths(+1);
        DateTime caducidad2 = caducidad.AddDays(-1);

        List<int> lDoc = (from contact in context.documentosempresas
                          where contact.fechaCaducidad < caducidad &&
                          contact.fechaCaducidad >   caducidad2
                          select contact.idDocumentosEmpresa).ToList();

        foreach (int doc in lDoc)
        {
         string correo = (from contact in context.empresas
                          where contact.idEmpresa == (from contact2 in
                          context.documentosempresas
                         where contact2.idDocumentosEmpresa == doc
                         select contact2.Empresa_idEmpresa).FirstOrDefault()
                         select contact.correo).FirstOrDefault();

                        EnviarEmail(correo,"Empresa", doc);
        }
      }
```

```
                catch (Exception e) { Console.CapsLock.ToString(); }
    }
  }
```

El código anterior obtiene una lista de los documentos de empresa que caducan dentro de un mes. Después recorre esa lista y almacena en la variable correo el email de la empresa. Seguidamente llama al método EnviarEmail que se especifica a continuación:

```
public static void EnviarEmail(string mail, string tipo, int document
{
       string mensaje = ComponerMensaje(document, tipo);
       var usuarios = permisator.dame_usuarios("Estado Proveedores");
       var listaAdmins = usuarios.Where(a => a.Rol == "Administrador").ToList();

       var listaDAP = permisator.dame_usuariosLDAP();

       foreach (permisatorService.Permiso userAdmin in listaAdmins
       {
        var user = listaDAP.Where(a => a.Codigo ==
       userAdmin.Usuario.ToLower()).FirstOrDefault();
              if (user != null)
              {
               string email = user.Email;
               Console.CapsLock.ToString();
               SendEmail(email, mensaje);
              }
       }
}
```

El método anterior llama a su vez al método ComponerMensaje que en función de distintas variables compone un string con el texto que se debe enviar por email y también al método SendEmail que es el método que utiliza el Web Service notificaciones para finalmente enviar el email. Este último método se especifica a continuación:

```
public static void SendEmail(string email, string mensaje)
{
 try
 {
   string emailRemitente = "coordinacionActividades@aguasdevalencia.es";
   string nombreRemitente = "[Coordinacion de Actividades]";
   string asunto = "Aviso de Caducidad de Documentos";
   string[] toList = new[] { email };
   string[] ccList = null;
   Dictionary<string, byte[]> attachements = new Dictionary<string, byte[]>();
   NotificacionServiceClient notificatorClient = new NotificacionServiceClient();
notificatorClient.SendEmailAsync(emailRemitente, nombreRemitente, toList, ccList,
asunto, mensaje, attachements

 }
 catch (Exception e){ Console.CapsLock.ToString();}
}
```

Todo el proceso descrito deberá realizarse de forma diaria para comprobar al menos una vez al día la validez de todos los documentos. Para conseguir esto se ha creado una página vacía, llamada WebForm1 que en el evento Load contiene la llamada a los

métodos necesarios para iniciar el proceso de comprobación de caducidad. Esta página es consultada automáticamente por una aplicación existente en AVSA que permite programar el acceso a páginas web.

El código es el siguiente:

```
protected void Page_Load(object sender, EventArgs e)
{
       notificaciones.ComprobarCaducidadEmpresa();
       notificaciones.ComprobarCaducidadEspecial();
       notificaciones.ComprobarCaducidadTrabajador();
       notificaciones.ComprobarCaducidadVehiculo();
}
```

7.5.3. Usuarios Service

El Web Service Usuarios Service proporciona acceso a una base de datos de usuarios del dominio de AVSA y que se utiliza para obtener los emails de los usuarios definidos en Permisator. Este Web Service se utiliza en una clase auxiliar para obtener la lista de usuarios total y luego compararla con los usuarios de la aplicación en permisator (y recuperar el correo de los últimos).

El método se llama DameUsuariosLDAP y se llama desde el método Enviar Email (ver apartado Gestor de Notificaciones).

El código del método es el siguiente:

```
public static usuariosService.Usuario[] dame_usuariosLDAP()
{
 var usuarios = new usuariosService.UsuariosServiceClient();
 return usuarios.dameUsuarios(false);
}
```

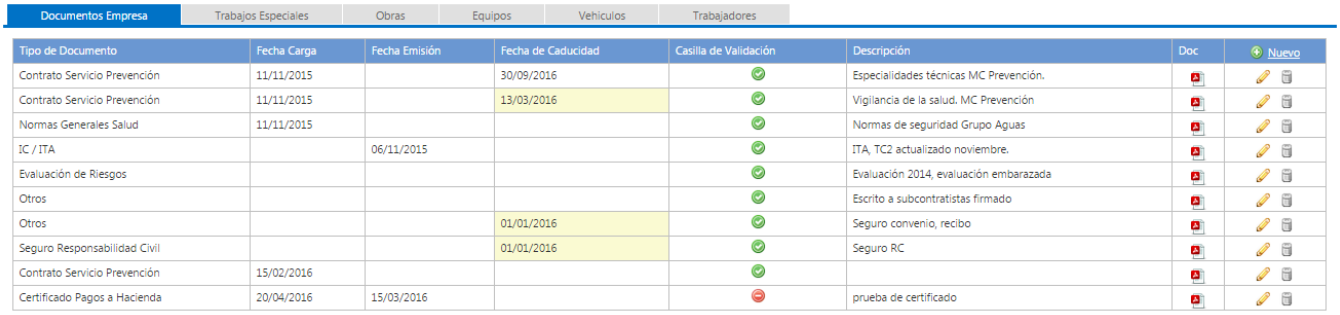
7.6. Control de gestión de ficheros

Una parte fundamental de la aplicación es el control de documentación. El propósito final de la aplicación es almacenar documentos que luego pueden ser consultados.

Para hacer esto se ha recurrido a un control de desarrollo interno del grupo AVSA.

Este control proporciona una interfaz para cargar documentos en una estructura de carpetas definida en el servidor.

A continuación, se ilustra el funcionamiento del control:



Documentos Empresa | Trabajos Especiales | Obras | Equipos | Vehículos | Trabajadores

Tipo de Documento	Fecha Carga	Fecha Emisión	Fecha de Caducidad	Casilla de Validación	Descripción	Doc	Nuevo
Contrato Servicio Prevención	11/11/2015		30/09/2016		Especialidades técnicas MC Prevención.		
Contrato Servicio Prevención	11/11/2015		13/03/2016		Vigilancia de la salud. MC Prevención		
Normas Generales Salud	11/11/2015				Normas de seguridad Grupo Aguas		
IC / ITA		06/11/2015			ITA, TC2 actualizado noviembre.		
Evaluación de Riesgos					Evaluación 2014, evaluación embarazada		
Otros					Escrito a subcontratistas firmado		
Otros			01/01/2016		Seguro convenio, recibo		
Seguro Responsabilidad Civil			01/01/2016		Seguro RC		
Contrato Servicio Prevención	15/02/2016						
Certificado Pagos a Hacienda	20/04/2016	15/03/2016			prueba de certificado		

Ilustración 41 Gestión de Ficheros

La imagen siguiente muestra el detalle del icono.

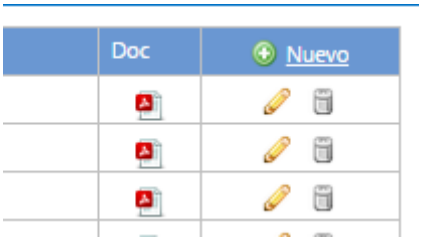


Ilustración 42 Botón Gestión de Ficheros

El icono se añade al GridView añadiendo una columna de control que contiene un botón personalizable o custom button al que le añadimos el icono que deseamos mostrar, en nuestro caso un icono de documento PDF. El código que lo permite es el siguiente:

```
<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc"
VisibleIndex="10">
<CustomButtons>

        <dx:GridViewCommandColumnCustomButton ID="showDoc" Image-
        Url="Content/Images/document-pdf-text.png">

                <Image Url="Content/Images/document-pdf-text.png"></Image>


        </dx:GridViewCommandColumnCustomButton>

</CustomButtons>
```

Al pulsar sobre el botón aparece la siguiente ventana en forma de pop-up:

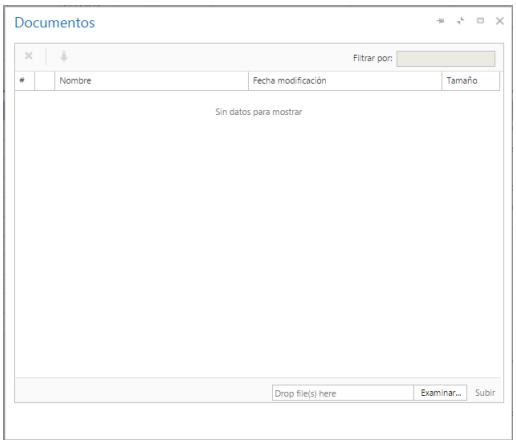


Ilustración 43 Pop Up Control de Ficheros

Para conseguir esto hay que modificar el comportamiento del botón de acuerdo a lo que interese al programador (en este caso, abrir el control file manager).

Para ello se añade las siguientes líneas en la declaración del grid:

```
CustomButtonClick="function(s,e){ if(e.buttonID=='showDoc'){
var rowKey = s.GetRowKey(e.visibleIndex);
OpenPopUpFileManager('Empresa\\'+rowKey);  } }" />
```

Continuando con la descripción del comportamiento, en esta ventana se accede a una lista documentos. Tiene un botón examinar que abre un explorador para seleccionar el fichero que desea el usuario en su ordenador. Su comportamiento se ilustra en la siguiente pantalla.

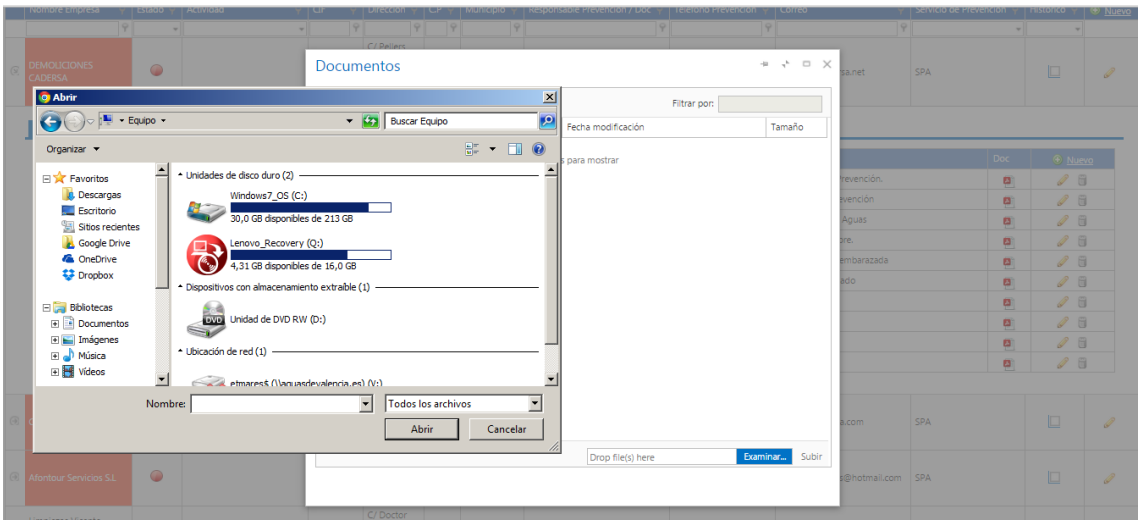


Ilustración 44 Explorador de Ficheros

Los documentos se almacenan en una estructura de carpetas en el servidor. La ruta de acceso se define en el webconfig de la aplicación.

El webconfig es un archivo que define distintos parámetros de configuración de la aplicación (como las credenciales de uso de la base de datos) o la versión del conector de MySQL. La ruta de la carpeta se define de la siguiente forma:

```
<add key="pathDocumentos" value="C:\docsProveedoresX" />
```

Así, cada vez que se crea un documento nuevo la aplicación accede a un directorio que contiene las siguientes carpetas:

- Empresa
- Obras
- Trabajadores
- TrabajosEspeciales
- EquiposVehiculos

Dentro de cada carpeta la aplicación crea una subcarpeta donde almacena el documento correspondiente. Esa subcarpeta tiene el nombre de la id del documento.

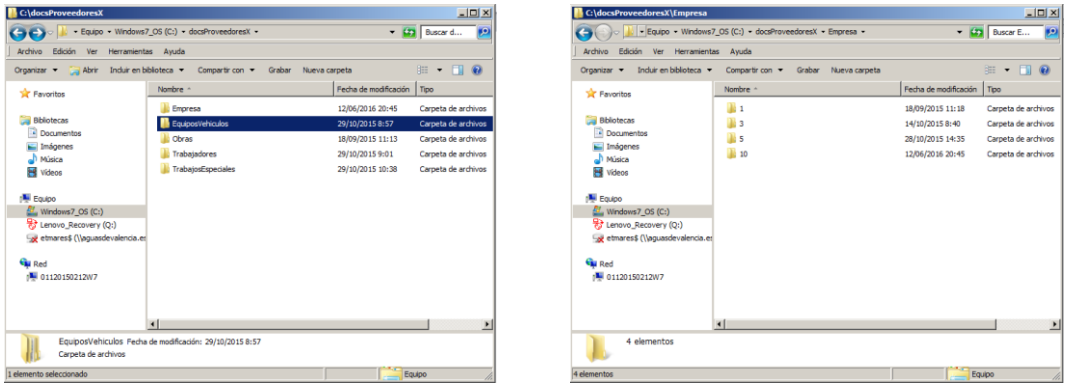


Ilustración 45 Estructura de Carpetas Control de Ficheros

Por ejemplo, si el usuario crea un nuevo documento de Empresa la aplicación accede a la carpeta empresa y dentro de esta crea una nueva carpeta cuyo nombre es la id de ese nuevo documento.

El extracto de código que lo permite es el siguiente:

```
protected void callBackFiles_Callback(object sender,
DevExpress.Web.CallbackEventArgsBase e)
{
var idROW = e.Parameter;
var pathFolder = url + "\\" + NamePath + "\\" + idROW;
if (!Directory.Exists(pathFolder))
Directory.CreateDirectory(pathFolder);
 Session["idRowSelected"] = pathFolder;
fileManager.Settings.RootFolder = pathFolder;
}
```

7.7. Log de Operaciones

Uno de los requisitos de la aplicación es mantener un log o registro de operaciones. Debido a la naturaleza y sensibilidad de los datos que almacena la aplicación es imprescindible mantener un registro de las operaciones que se realizan en la aplicación y qué usuario las lleva a cabo.

Para ello se creó la tabla Log que permite almacenar estos datos. Los métodos encargados de realizar las tareas de registro de datos están contenidos en la clase Administración.

Esta clase contiene una serie de métodos que son capaces de obtener información directamente de los GridViews de DevExpress.

Para utilizar estos métodos debemos añadir la siguiente línea de código a los eventos del GridView que queremos registrar:

```
Administration.fnc_log_añadir("InsertarEmpresa", Administration.fnc_log_dame(e));
```

El primer parámetro del método es un string que representa el tipo de operación que se está realizando. El segundo es un método que obtiene información del GridView.

El método fnc_log_dame devuelve una cadena de texto con la siguiente información:

```
[tipoDocumento]=Evaluación de Riesgos,[fechaCarga]=18/01/2016
0:00:00,[fechaEmision]=13/10/2014
0:00:00,[fechaCaducidad]=,[validado]=True,[descripcion]=jefe equipo
AT, ayudante equipo (alta/baja),
técnico.,[Empresa_idEmpresa]=8,[idDocumentosEmpresa]=44,[Id]=44','defa
ult.aspx','::1
```

Y la tabla log almacena información como muestra la siguiente imagen:

log (estadoproveedo....aguasdevalencia.es) | WebLoggerHelper.cs | cls_proveedores.cs

id	usuario	accion	resumen	fecha
1	etmares	Actualizar Emple...	[DocumentoTrab...	17/05/2016 10:...
2	etmares	Actualizar Emple...	[DocumentoTrab...	17/05/2016 10:...
3	etmares	Actualizar Emple...	[DocumentoTrab...	17/05/2016 11:...
4	etmares	Actualizar Docu...	','default.aspx','::1	17/05/2016 12:...
5	etmares	Actualizar Docu...	[DocumentoTrab...	17/05/2016 12:...
6	etmares	Actualizar Docu...	[DocumentoTrab...	17/05/2016 16:...

Ilustración 46 Detalle Tabla Log

Se almacena el usuario, la acción realizada, el resumen (el string obtenido con el método fnc_log_dame) y la fecha de la operación, de forma que en caso necesario esta información puede proporcionarse a los usuarios de la aplicación para que sea analizada por los responsables del departamento de Prevención de Riesgos Laborales.

Las acciones recopiladas por el Log de Operaciones son:

- **Actualización de Datos**: mediante el evento del GridView OnRowUpdating.
- **Inserción de Datos**: mediante el evento del GridView OnRowInserting.
- **Eliminación de Datos**: mediante el evento del GridView OnRowDeleting.

8. EVALUACIÓN

En el siguiente apartado se presenta el resultado final del desarrollo proporcionando capturas de pantalla de la aplicación en su estado actual, ya en funcionamiento y con información real.

Nota: Todas las capturas mostradas a continuación contienen datos reales de empresas y personas. Para proteger su privacidad las capturas han sido modificadas para ocultar los datos sensibles.

8.1. Página Empresa

La página de empresa es la página principal de la aplicación.

8.1.1. Vista General Empresa

En esta vista pueden consultarse todas las empresas almacenadas en el sistema. Cada empresa tiene un semáforo asociado que indica si es válida. En caso de no serlo, además, la celda correspondiente a su nombre se marca como roja.

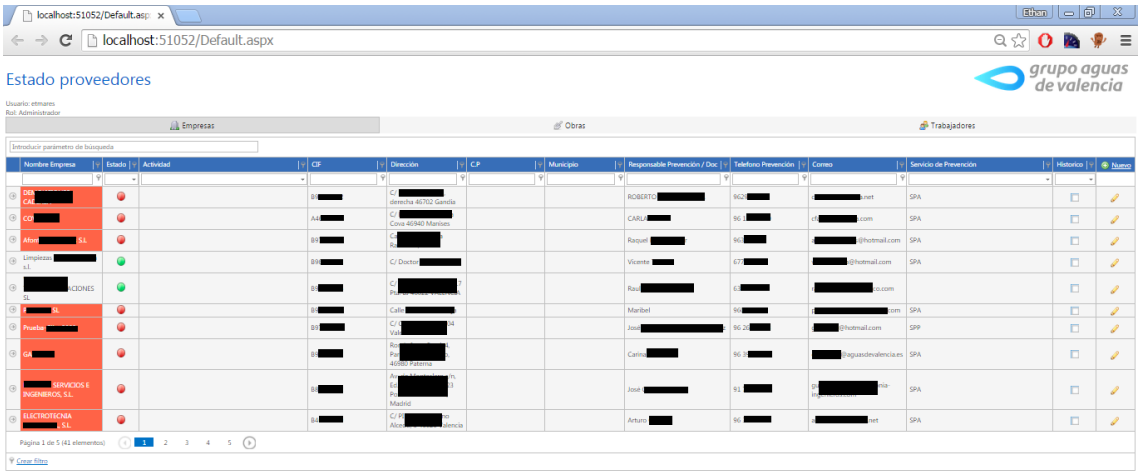


Ilustración 47 Página Empresa

8.1.2. Detail Row Documentos Empresa

Al hacer click sobre el botón de la flecha se muestra un desplegable que muestra distintas pestañas. Esto ocurre para cada fila de la tabla (para cada empresa almacenada en el sistema). Cada pestaña contiene las entidades relacionadas con esa empresa.

En el caso de los trabajadores estos también disponen de un semáforo que indica si este es válido o no.

Las entidades anidadas disponibles son:

- Documentos Empresa
- Trabajos Especiales
- Obras
- Equipos

- Vehículos
- Trabajadores

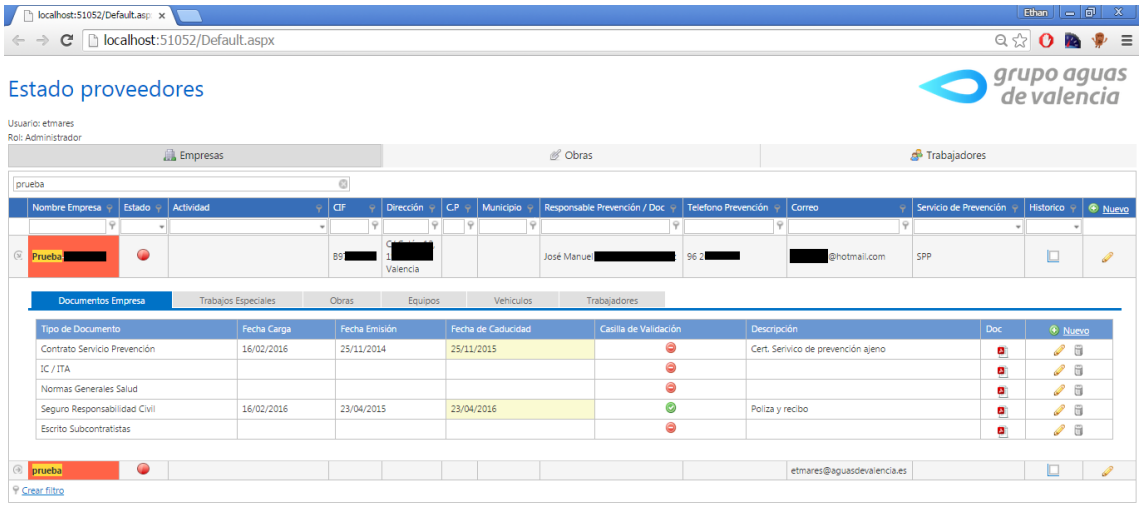


Ilustración 48 Detail Row Empresa

8.1.3. Detail Row de Entidades de Empresa

En el caso de Trabajos Especiales, Obras, Equipos, Vehículos y Trabajadores se almacenan documentos en otro GridView anidado, uno para cada entidad. En estos casos lo que se almacena son los documentos asociados a esa entidad. En la siguiente captura se muestran los documentos de un trabajador.

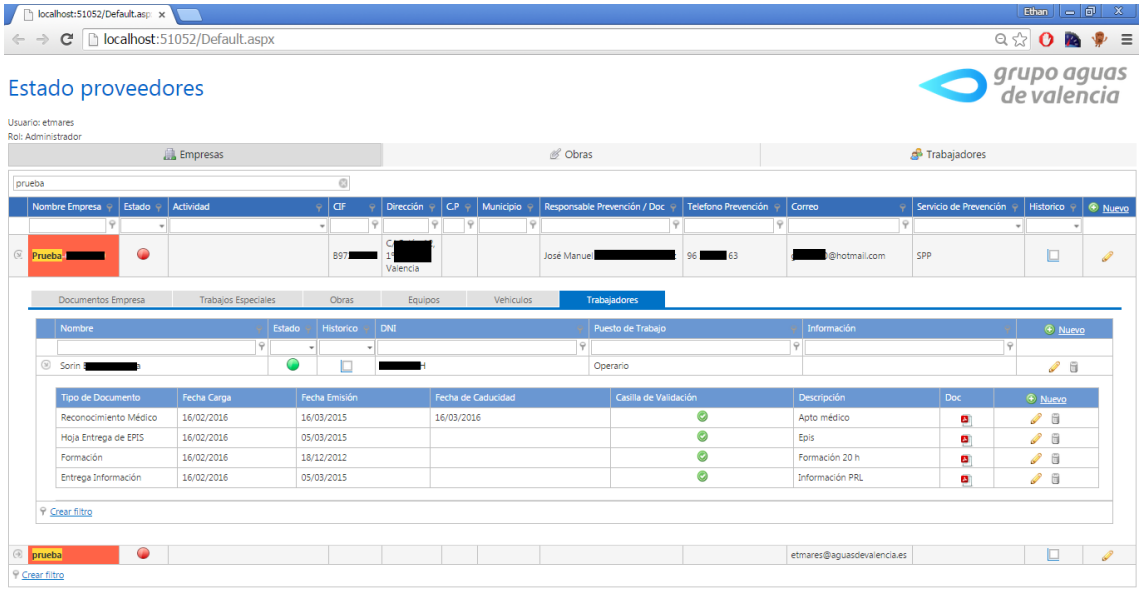


Ilustración 49 Detail Row Entidades Empresa

Esta vista muestra fechas diversas de los documentos, su casilla de validación y un botón que permite cargar documentos en el sistema (con el icono de Documento PDF).

8.1.4. Modo de edición

La siguiente pantalla muestra la vista de edición de un documento. Puede apreciersa que la edición se realiza en la misma fila directamente. Aparecen dos iconos nuevos, uno para guardar cambios y otro para cancelarlos, en la última columna.

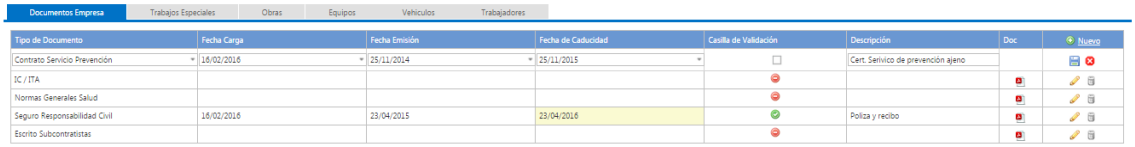


Ilustración 50 Detalle Modo Edición

8.1.5. Detalle Lista de Valores

La siguiente pantalla muestra un ejemplo de edición de una lista de valores, en este caso de un tipo de documento de empresa.

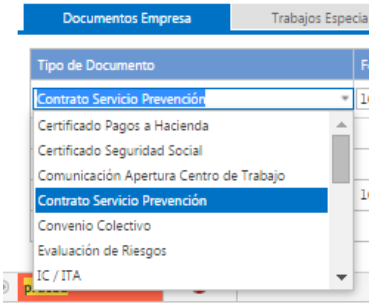


Ilustración 51 Detalle Lista de Valores

8.1.6. Detalle de selector de Fecha

La siguiente pantalla muestra un detalle de la edición de una entidad, en concreto el selector de fecha para los campos de tipo date.

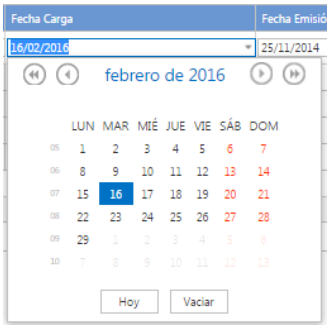


Ilustración 52 Detalle Selector de Fecha

8.1.7. Detalle Editor de Filtros

El editor de filtros es un control que permite a los usuarios realizar consultas avanzadas sobre las tablas. Su comportamiento se ilustra en la imagen siguiente.

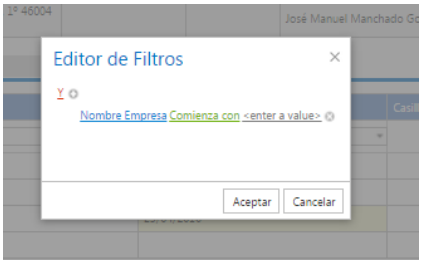


Ilustración 53 Editor de Filtros

8.1.8. Detalle de Semáforos y Validaciones

Las siguientes capturas muestran detalles relacionados con la validación de documentos.

En la siguiente captura se aprecia una tabla de documentos. Pueden apreciarse las celdas con información de fechas, que cambian de color en caso de estar caducadas, la casilla de validación, que muestra un icono verde o rojo en función de si el usuario administrador lo da o no como bueno y la casilla Doc que permite adjuntar documentos a cada fila.

Tipo de Documento	Fecha Carga	Fecha Emisión	Fecha de Caducidad	Casilla de Validación	Descripción	Doc	Nuevo
Contrato Servicio Prevención	11/11/2015		30/09/2016		Especialidades técnicas MC Prevención.		
Contrato Servicio Prevención	11/11/2015		13/03/2016		Vigilancia de la salud. MC Prevención		
Normas Generales Salud	11/11/2015				Normas de seguridad Grupo Aguas		
IC / ITA		06/11/2015			ITA, TC2 actualizado noviembre.		
Evaluación de Riesgos					Evaluación 2014, evaluación embarazada		
Otros					Escrito a subcontratistas firmado		
Otros			01/01/2016		Seguro convenio, recibo		
Seguro Responsabilidad Civil			01/01/2016		Seguro RC		
Contrato Servicio Prevención	15/02/2016						
Certificado Pagos a Hacienda	20/04/2016	15/03/2016			prueba de certificado		

Ilustración 54 Validación de Documentos

A continuación, se muestra una pantalla con el detalle de los semáforos.

Ilustración 55 Detalle de Semáforos

La siguiente pantalla muestra el detalle de la casilla de validación.

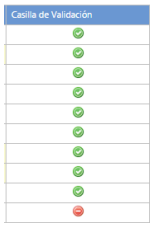


Ilustración 56 Detalle de Iconos de Validaciones

8.1.9. Detalle de File Manager

La siguiente pantalla muestra la pantalla de carga de documentos, a la que se accede al pulsar sobre el icono correspondiente las filas de los GridView de documentos.

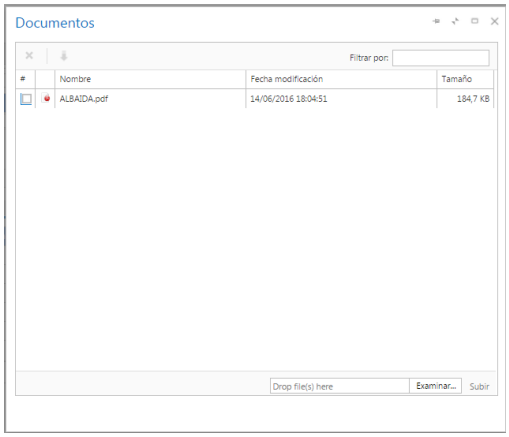


Ilustración 57 Detalle de File Manager

8.1.10. Detalle Filtros y Barra de Búsqueda

La siguiente pantalla muestra el detalle de los filtros y barras de búsqueda de los GridView.

En la parte superior se aprecia una barra de búsqueda en la que el usuario puede introducir caracteres. Se realiza una búsqueda en todos los campos del grid y se limita las filas mostradas a aquellas que coinciden con el criterio de búsqueda introducido por el usuario.

También puede definirse una búsqueda similar restringiéndose a cada fila, mediante la caja de texto que aparece justo debajo del nombre de la columna. De esta forma el usuario puede combinar criterios de búsqueda para realizar pesquisas más precisas. En caso de que el campo sea una lista de valores se muestra una lista con los valores posibles.

Pulsando sobre las columnas se ordenan los datos de forma alfabética, si es de tipo texto y de menor a mayor si son números o fechas. Volviendo a pulsar sobre la columna se invierte el criterio de alteración.

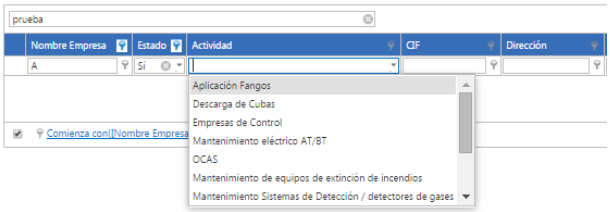


Ilustración 58 Detalle Filtros

8.2. Página Obras

La página de obras muestra todas las obras almacenadas en la aplicación, con independencia de la empresa a la que pertenecen.

8.2.1. Vista General de Obras

La vista de obras muestra una tabla con las obras y con distintos campos de información.

Ilustración 59 Vista General Obras

8.2.2. Vista Detail Row Subcontratas y Documentos Obra

Pulsando sobre el botón de la flecha se abre un desplegable con dos pestañas. Una para almacenar documentos y otro para llevar el registro de las subcontratas.

La siguiente pantalla muestra una obra desplegada, concretamente la pantalla de subcontratas.

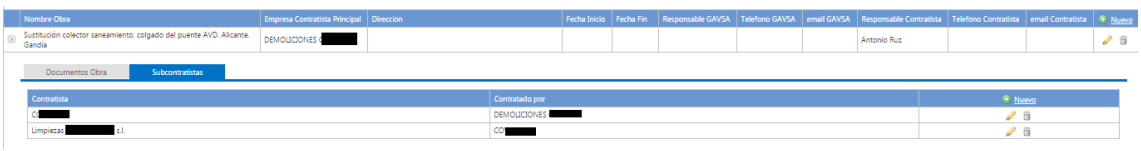


Ilustración 60 Detalle Subcontratas

La siguiente pantalla muestra un detalle de la edición de los documentos de obras.

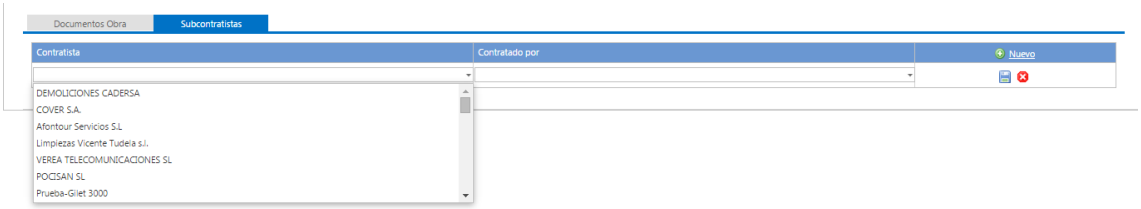


Ilustración 61 Detalle Documentos Obra

8.3. Página Trabajadores

La página de trabajadores muestra información referente a los trabajadores registrados en el sistema, con independencia de la empresa a la que pertenecen.

8.3.1. Vista general Trabajadores

La vista general de trabajadores muestra los datos de los trabajadores, en forma de tabla. En la imagen se aprecia la columna estado que contiene el semáforo que indica si un trabajador es válido o no. Se observa que en caso de no ser válido se resalta el nombre del trabajador en rojo. También se muestran varias filas de un color oscuro, que representan aquellos trabajadores marcados como históricos.

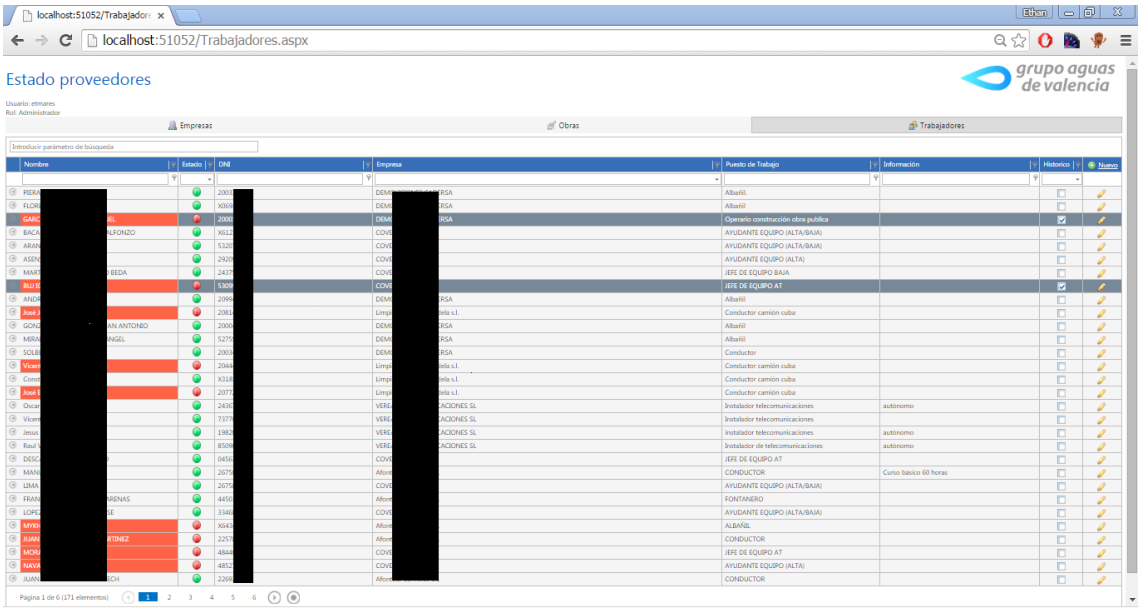


Ilustración 62 Página Trabajadores

8.3.2. Detalle Detail Row Documentos Trabajadores

Pulsando sobre el botón de la flecha se accede a un menú desplegable que muestra los documentos relacionados con el trabajador. Cada fila representa un documento, puede accederse a su información, así como ver su estado. Mediante la columna Doc puede accederse a los documentos almacenados.

Ilustración 63 Detalle Detail Row Trabajadores

9. CONCLUSIÓN

El desarrollo de este trabajo y de la aplicación de la que es objeto ha supuesto un profundo estudio de una amplia variedad de herramientas de gran actualidad. Todo lo empleado en el desarrollo de la aplicación, desde el sistema de gestión de bases de datos MySQL a la plataforma .NET de Microsoft pasando por el paquete de controladores de UI DevExpress o herramientas más específicas como el O/RM Entity Framework eran desconocidos por el alumno autor de este trabajo. El estudio de esta y otras herramientas ha supuesto todo un reto para el autor, que ha visto recompensado su esfuerzo con la obtención de conocimientos que suponen un punto diferencial en el mercado laboral a día de hoy.

Además, ha proporcionado una oportunidad para aplicar los conocimientos adquiridos a lo largo de los estudios del grado en ingeniería informática, pudiendo combinar conocimientos de distintas asignaturas en un proyecto real. Entre otros se han aprovechado conocimientos de Bases de Datos, Programación, Ingeniería del Software o Gestión de Proyectos.

A lo largo del desarrollo se han encontrado numerosas dificultades derivadas del desconocimiento del alumno de estas tecnologías. Afortunadamente y debido al amplio ámbito de uso de las tecnologías empleadas es posible acceder a gran cantidad de documentación que ha permitido superar estas dificultades. También ha sido de gran ayuda el apoyo y consejo de los trabajadores de AVSA cuya experiencia ha ayudado al alumno a superar los retos que se iban planteando.

La aplicación desarrollada por el alumno se encuentra actualmente implantada en la empresa Aguas de Valencia y forma ya parte del día a día del trabajo del Departamento de Gestión de Riesgos Laborales.

En el futuro próximo se espera realizar una serie de mejoras sobre la aplicación en función de las sugerencias propuestas por los trabajadores una vez han podido trabajarla en profundidad. Se espera, además, que la aplicación pase a tener un ámbito transversal entre diferentes departamentos y no sólo dentro del departamento de Prevención de Riesgos Laborales.

10. BIBLIOGRAFIA

1. **Wikipedia.** Desarrollo ágil de software. [En línea] 2016. https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software.

2. **Schwaber, K, Sutherland, J.** La Guía Definitiva de Scrum: Las Reglas del Juego. *scrumguides.org.* [En línea] Julio de 2013. http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf.

3. **Microsoft.** ASP.NET. [En línea] http://www.asp.net/.

4. **Gaylord, Jason N., y otros, y otros.** *Professional ASP.NET 4.5 in C# and VB.* s.l. : United States: Wrox Press Ltd, 2013 .

5. **Microsoft.** WebForms. [En línea] 2016. http://www.asp.net/web-forms.

6. **Zawodny, Jeremy D, Balling, Derek J.** *MySQL Avanzado.* s.l. : Anaya Multimedia, 2008.

7. **Wikipedia.** Visual Studio. [En línea] 2016. https://en.wikipedia.org/wiki/Microsoft_Visual_Studio.

8. **Microsoft.** Entity Framework. [En línea] 2016. http://www.asp.net/entity-framework.

9. **DevExpress.** DevExpress Documentation. [En línea] 2016. https://documentation.devexpress.com/#AspNet/CustomDocument7873.

10. **Pressman, RS.** *Ingenieria del Software un enfoque aplicado 7 ed.* s.l. : McGrawHill, 2010.

11. **Orallo, Enrique Hernández.** disca.upv.es. [En línea] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

11. ANEXOS

11.1. Manual de la aplicación

11.1.1. Cómo consultar una empresa

La pantalla principal de la aplicación muestra una lista de empresas. Las empresas se muestran por defecto en orden de creación, de más antiguas a más nuevas. El color de la casilla del nombre de la empresa indica si esta es válida o no válida. Esto también puede identificarse por medio del indicador de color de la segunda columna.

Si el usuario que accede a la aplicación es administrador podrá editar la información de la empresa pulsando sobre el icono del lápiz de la última columna de la tabla.

Para crear nuevas empresas se debe pulsar sobre el bótón nuevo, en la cabecera de la última columna. Se añadirá una fila en la tabla con campos que el usuario puede editar.

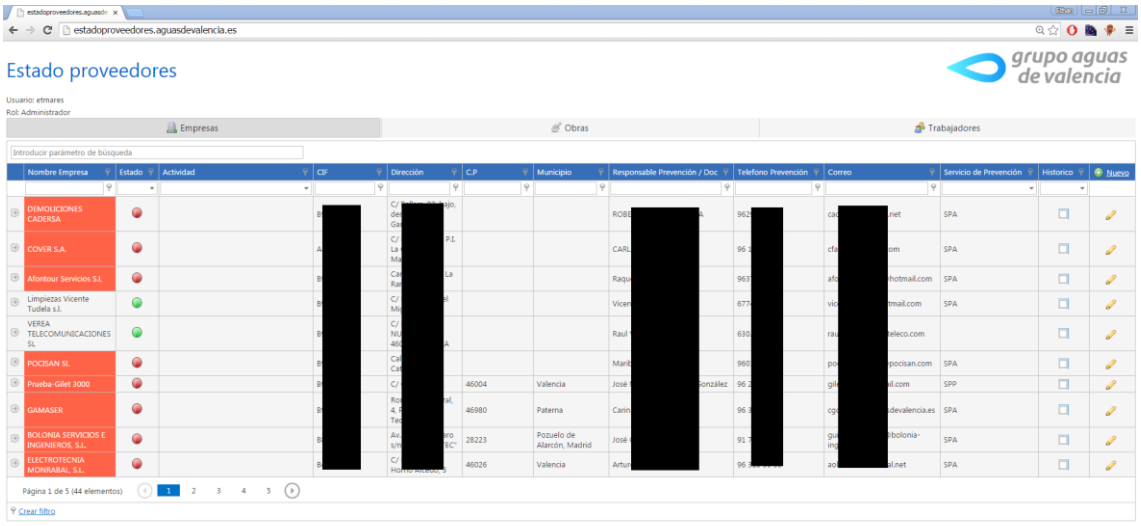


Ilustración 64 Manual-Empresa

11.1.2. Cómo buscar una empresa

Se pueden distintos tipos de búsqueda de empresas. Las opciones posibles son:

- **Mediante la barra de búsqueda general**: es la barra de búsqueda que se encuentra sobre la tabla. Aquí el usuario puede introducir un texto y la aplicación buscará contenido coincidente en todas las columnas y filas de la tabla, resaltando en amarillo los resultados.
- **Mediante la barra de búsqueda de columna**: se trata de un campo de texto presente en la cabecera de todas las columnas. Desde aquí puede introducirse un texto que se buscará en la columna seleccionada. Puede convinarse con búsquedas de otras columnas.
- **Sobre el botón filtro**: pulsando el botón filtro (junto al nombre de la columna) se muestra un desplegable con todos los valores existentes en la columna.

- **Ordenando alfabéticamente**: pulsando una vez sobre el nombre de la columna es posible ordenar las filas en orden alfabético.
- **Con el botón "crear filtro"**: en la parte inferior de la tabla existe un botón que proporciona una interfaz al usuario donde puede crear filtros personalizados y combinar consultas sobre la tabla

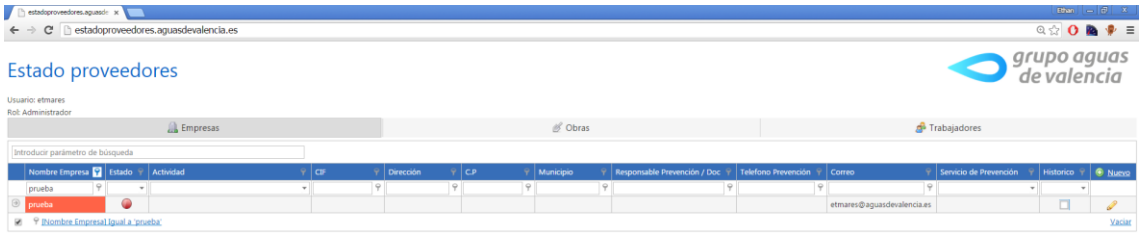


Ilustración 65 Manual - Buscar Empresa

11.1.3. Como consultar el detalle de una empresa

Es posible consultar el detalle de una empresa pulsando sobre el botón de la izquierda de la fila con forma de flecha. Esto abre un menú que da acceso distintas pestañas. Cada tabla contiene entidades relacionadas con la empresa. Pulsando en la pestaña correspondiente pueden consultarse dichas entidades.

Todas las operaciones de búsqueda y ordenación de la tabla empresa también se aplican a estas tablas.

Pulsando sobre el botón nuevo pueden crearse nuevas entidades. Las entidades existentes pueden modificarse pediante el botón con forma de lápiz o eliminarse con el botón de la papelera (sólo en caso de que el usuario sea administrador).

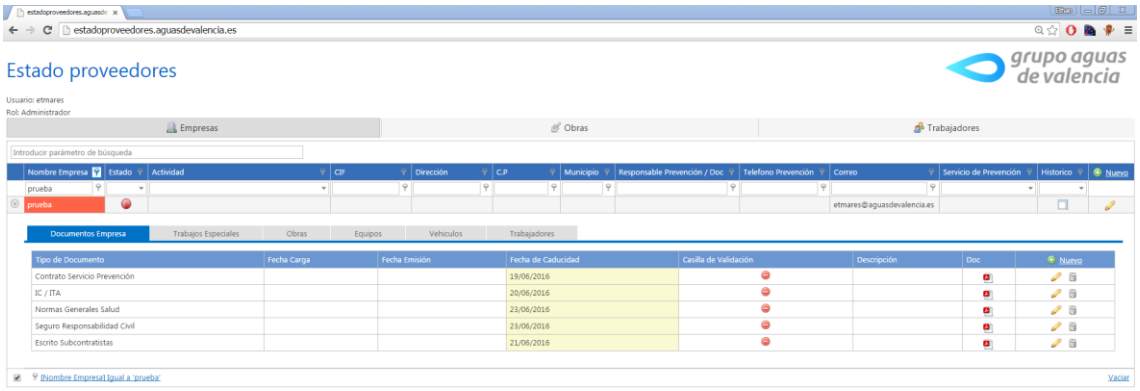


Ilustración 66 Manual-Detalle Empresa

11.1.4. Como consultar documentos de entidades.

A su vez las entidades relacionadas con empresa pueden tener documentos asociados. Para acceder a ellos debe pulsarse sobre el botón con forma de flecha presente al comienzo de la ficha de la entidad. Esto mostrará una tabla con los documentos relacionados con dicha entidad.

Pueden adicionarse nuevos documentos con el botón nuevo, editarse existentes con el botón del lápiz y eliminarse con el botón de la papelera.

El usuario puede adjuntar documentos pulsando sobre el icono del pdf de la columna Doc.

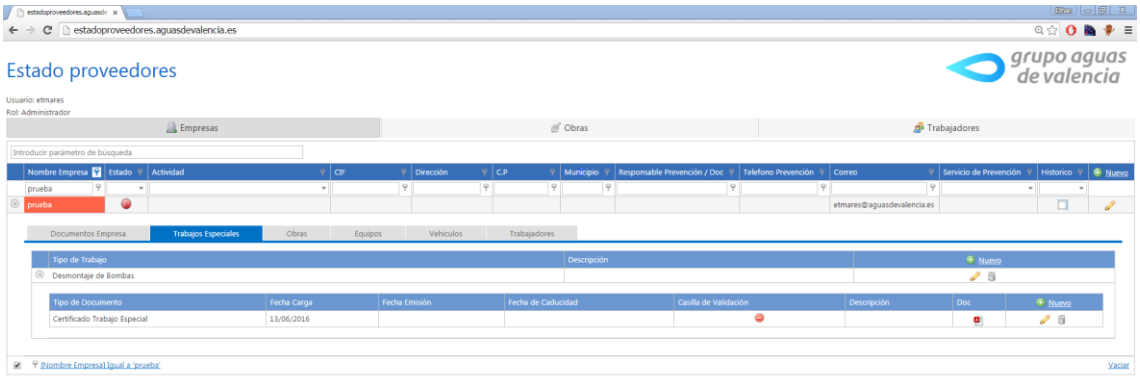


Ilustración 67 Manual-Documentos entidades

11.1.5. Cómo consultar obras

Para consultar obras debe pulsarse sobre la pestaña correspondiente en la parte superior.

Se accederá a una página que muestra una tabla con todas las obras en el sistema.

Pueden crearse nuevas obras pulsando sobre el botón nuevo o editarse existentes pulsando sobre el botón editar con forma de lápiz.

Ilustración 68 Manual-Obras

11.1.6. Cómo consultar documentos de obras

Pulsando sobre el botón con forma de flecha del inicio de la fila de obra se abre un desplegable con dos pestañas. Pulsando sobre la pestaña documentos puede consultarse la documentación asociada a la obra. Pueden crearse nuevos documentos, editar los existentes o eliminarlos mediante los botones de la última columna.

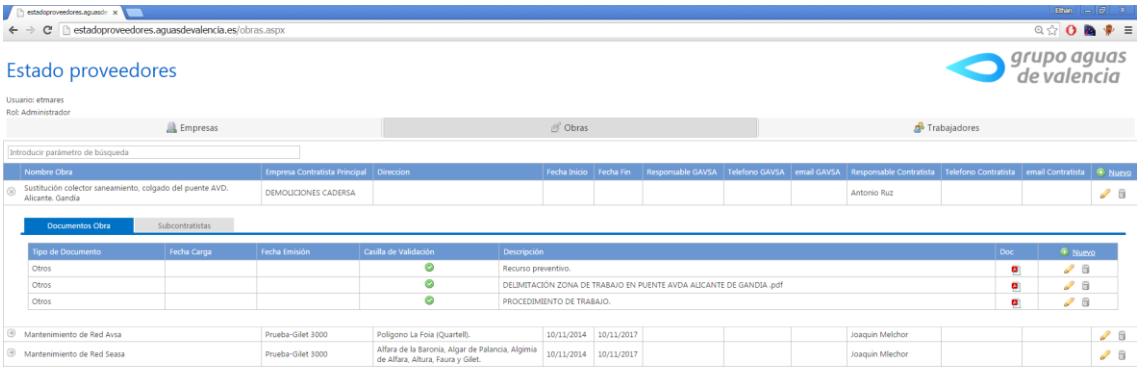


Ilustración 69 Manual - Documentos Obras

11.1.7. Cómo consultar subcontratistas por obras

Pulsando sobre la pestaña sibcontratistas en el detalle de obra se accede a una tabla que permite definir relaciones entre subcontratistas.

Pulsando sobre el botón nuevo se adiciona un contratista a la obra. Deberá seleccionarse de la lista desplegable que aparece. Se debe indicar también por quien esta contratado mediante el campo contratado por.

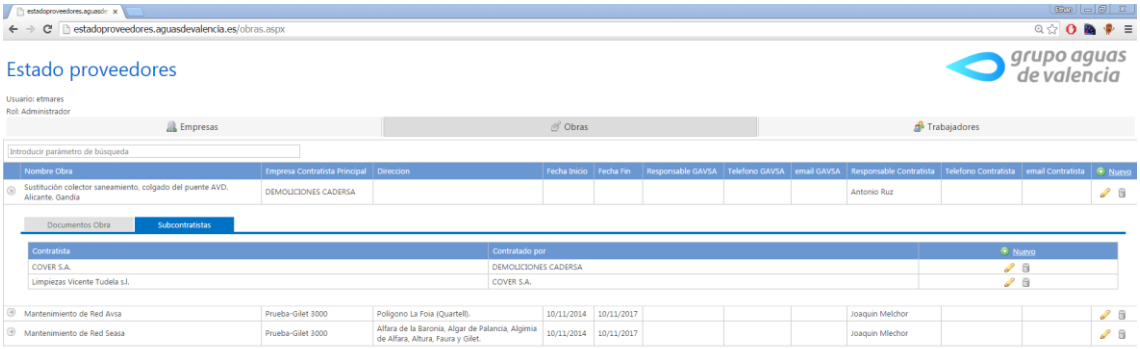


Ilustración 70 Manual - Contratistas Obras

11.1.8. Cómo consultar trabajadores

Para consultar trabajadores debe pulsarse sobre la pestaña trabajadores de la parte superior.

Se accede a una página que muestra una lista con todos lostrabajadores con independencia a la ampresa a la que pertenecen. Pueden crearse nuevos trabajadores con el botón nuevo o editarse los existentes con el botón editar con forma de lapiz.

Es posible realizar operaciones de búsqueda y ordenación de forma similar a la tabla empresa.

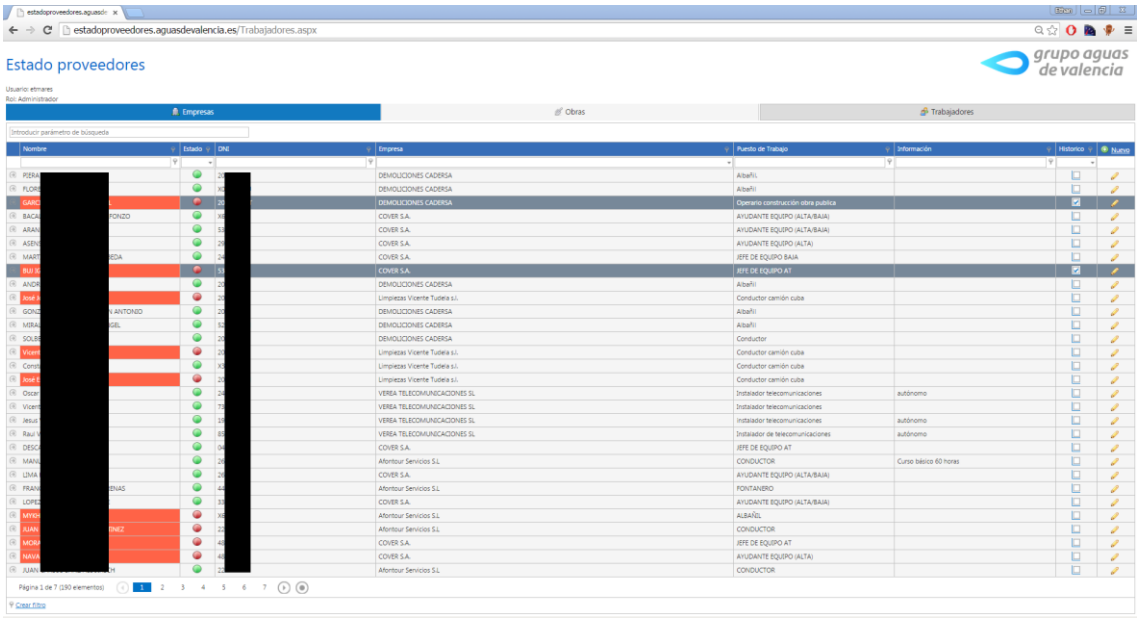


Ilustración 71 Manual - Trabajadores

11.1.9. Cómo consultar documentos de trabajadores

Pulsando sobre el botón con forma de flecha se accede al detalle de los trabajadores. Pueden consultarse, crearse o eliminarse documentos.

Pulsando sobre el botón de pdf puede añadirse o consultar los documentos. También es posible validar documentos pulsando sobre el icono de válido de columna “Casilla de Validación”.

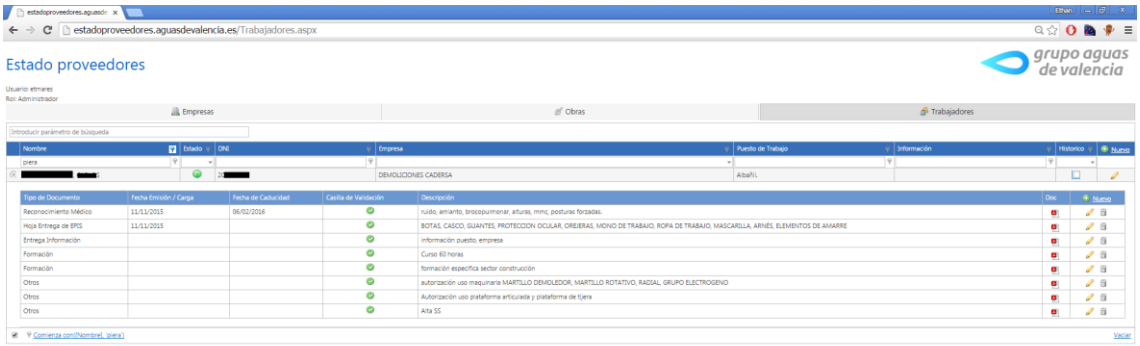


Ilustración 72 Manual - Documentos Trabajadores

11.1.10. Cómo consultar / cargar documentos

Al pulsar sobre el botón de pdf (presente en todas las filas de todas las tablas de documentos) se abre una ventana que muestra una lista de documentos. Pulsando sobre alguno de estos documentos el usuario puede descargarlos automáticamente.

Es posible añadir documentos pulsando sobre el botón examinar, que abre una ventana de exploración de Windows para que el usuario seleccione un documento en su ordendador que será cargado en el sistema.

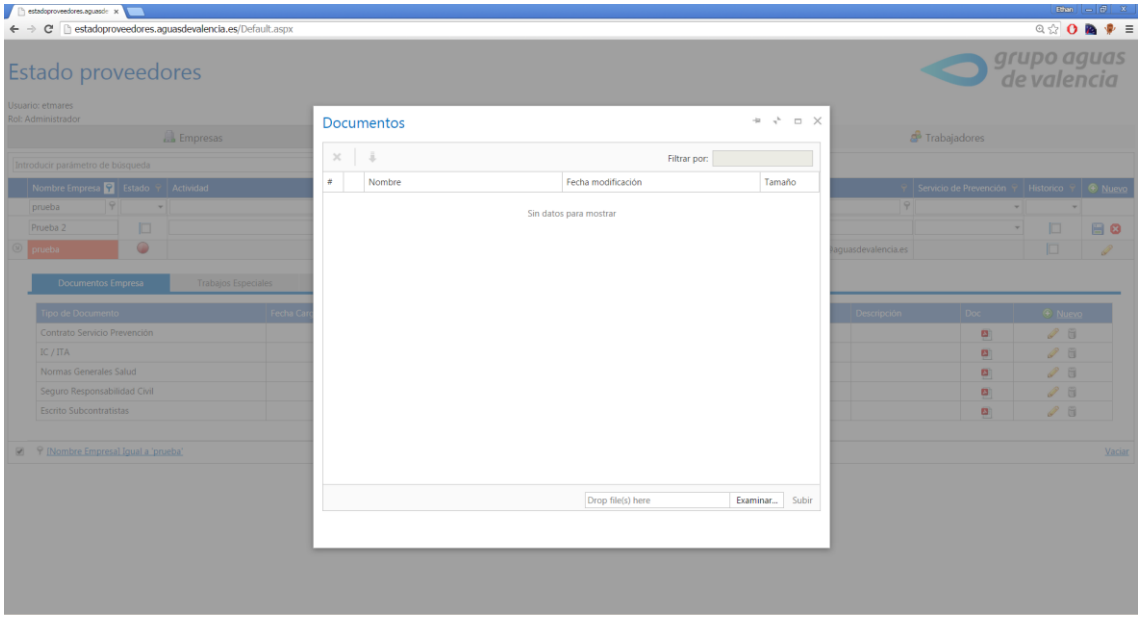


Ilustración 73 Manual - Cargar Documentos

11.2. Código de la aplicación

Este anexo contiene el código principal de la aplicación. Se presenta sólo el código definido por el alumno, omitiendo el código obtenido mediante generación automática, entity framework o web services.

11.2.1. Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" MasterPageFile="~/Main.master"
CodeBehind="Default.aspx.cs" Inherits="proveedoresX._Default" %>

<%@ Register Src="~/UserControl/FileManager.ascx" TagPrefix="uc1" TagName="FileManager" %>
<%-- Componente de gestión de FILE MANAGER--%>

<asp:Content ID="Content" ContentPlaceHolderID="MainContent" runat="server">

<script type="text/javascript">          <%-- Componente de gestión de FILE MANAGER--%>

function OpenPopUpFileManager(s, e) {
var id = s;

callBackFiles.PerformCallback(id);
pop_fileManager.Show();
}
</script>

<uc1:FileManager runat="server" ID="FileManager" />


<dx:ASPxGridView ID="GridEmpresa" ClientInstanceName="GridEmpresa" runat="server"
DataSourceID="EntityDataSource1" AutoGenerateColumns="False" EnableTheming="True"
KeyFieldName="idEmpresa"
OnCustomErrorText="GridErrorEliminar_CustomErrorText" SettingsBehavior-ConfirmDelete="true"
Width="100%" OnRowInserted="GridEmpresa_RowInserted"
OnHtmlDataCellPrepared="GridEmpresa_HtmlDataCellPrepared"
OnHtmlRowPrepared="GridEmpresa_HtmlRowPrepared">

<Columns>

<dx:GridViewDataTextColumn FieldName="idEmpresa" ReadOnly="True" VisibleIndex="1"
Visible="False">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="nombreEmpresa" VisibleIndex="2" Caption="Nombre Empresa">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="actividad" VisibleIndex="3" Caption="Actividad">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Aplicación Fangos" Value="Aplicación Fangos" />
<dx:ListEditItem Text="Descarga de Cubas" Value="Descarga Cubas" />
<dx:ListEditItem Text="Empresas de Control" Value="Empresa de Control" />
<dx:ListEditItem Text="Mantenimiento eléctrico AT/BT" Value="MTO Electrico AT/BT" />
<dx:ListEditItem Text="OCAS" Value="OCAS" />
<dx:ListEditItem Text="Mantenimiento de equipos de extinción de incendios" Value="MTO Extinción"
/>
<dx:ListEditItem Text="Mantenimiento Sistemas de Detección / detectores de gases" Value="MTO
Detectores" />
<dx:ListEditItem Text="Mantenimiento equipos anticaidas" Value="MTO anticaidas" />
<dx:ListEditItem Text="Descarga Productos Químicos" Value="Descarga Prod Quimicos" />
<dx:ListEditItem Text="Obras" Value="Obras" />
<dx:ListEditItem Text="Mantenimiento Colectores" Value="MTO Colectores" />
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataCheckColumn FieldName="esVerde" Caption="Estado" EditFormSettings-
Visible="False" ReadOnly="true" Width="30" VisibleIndex="2">
<EditFormSettings Visible="False"></EditFormSettings>
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarVerde" Cursor="crosshair" ImageUrl='<%#
btnVerdeEmp(Convert.ToBoolean(Eval("esVerde"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
```

```
</dx:GridViewDataCheckColumn>
<dx:GridViewDataTextColumn FieldName="CIF" VisibleIndex="4" Caption="CIF">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="direccion" VisibleIndex="5" Caption="Dirección">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="CP" VisibleIndex="5" Caption="C.P">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="municipio" VisibleIndex="5" Caption="Municipio">
</dx:GridViewDataTextColumn>

<dx:GridViewDataTextColumn FieldName="responsablePrevencion" VisibleIndex="6"
Caption="Responsable Prevención / Doc">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="telefonoPrevencion" VisibleIndex="7" Caption="Telefono
Prevención">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="correo" VisibleIndex="8" Caption="Correo">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="servicioPrevencion" VisibleIndex="9" Caption="Servicio
de Prevenci&#243;n">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Asumido Empresa" Value="Asumido Empresa" />
<dx:ListEditItem Text="SPA" Value="SPA" />
<dx:ListEditItem Text="SPMancomunado" Value="SPMancomunado" />
<dx:ListEditItem Text="SPP" Value="SPP" />
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataCheckColumn FieldName="historico" VisibleIndex="10" Caption="Historico"
Visible="true" Width="30">
</dx:GridViewDataCheckColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="11" ShowNewButtonInHeader="True"
ShowDeleteButton="False">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#376EB8" ForeColor="White"></Header>
<Row BackColor="WhiteSmoke"></Row>
</Styles>
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png"></Image>

<Styles>
<Style ForeColor="White"></Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png"></Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png"></Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png"></Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png"></Image>
</CancelButton>
</SettingsCommandButton>
<SettingsDetail ShowDetailRow="True" />
<SettingsSearchPanel Visible="True"></SettingsSearchPanel>
<SettingsBehavior ConfirmDelete="True"></SettingsBehavior>
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<Settings ShowFilterBar="Visible" ShowFilterRow="true" ShowFilterRowMenu="true"
ShowFilterRowMenuLikeItem="true" ShowHeaderFilterButton="true" />
<SettingsText SearchPanelEditorNullText="Introducir parámetro de búsqueda" />
<Templates>
<DetailRow>
<%-- ABRIMOS CONTROL DE PESTAÑAS--%>
<dx:ASPxPageControl runat="server" ID="pageControl" Width="100%" EnableCallBacks="true"
ActiveTabIndex="2" OnActiveTabChanged="pageControl_ActiveTabChanged">
<TabPages>
```

```
<dx:TabPage Text="Documentos Empresa" Visible="true">
<%-- ABRIMOS Pestaña Documentos empresa--%>
<ContentCollection>
<dx:ContentControl runat="server">

<dx:ASPxGridView ID="GridDocEmpresa" ClientInstanceName="GridDocEmpresa" runat="server"
AutoGenerateColumns="False" DataSourceID="EntityDocEmpresa" Width="100%"
KeyFieldName="idDocumentosEmpresa"
OnBeforePerformDataSelect="GridDocEmpresa_BeforePerformDataSelect"
OnRowInserting="GridDocEmpresa_RowInserting" OnRowUpdating="GridDocEmpresa_RowUpdating"
OnRowInserted="Log_RowInserted" SettingsBehavior-ConfirmDelete="true"
OnRowUpdated="GridDocEmpresa_RowUpdated" OnCustomCallback="DocEmpresa_CustomCallback"
OnCustomErrorText="GridDocEmpresa_CustomErrorText"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize" OnInit="OcultarColumnas_Init"
OnHtmlDataCellPrepared="PintarCaducidad_HtmlCellPrepared"
OnHtmlRowPrepared="Tooltip_HtmlRowPrepared">
<ClientSideEvents
EndCallback="function(s,e) { if (s.cpUpdate) { if (s.cpUpdate == '1') {  GridEmpresa.Refresh();
delete s.cpUpdate; } } }"
RowDblClick="function(s,e) { s.StartEditRow(e.visibleIndex); }"
ContextMenuItemClick="function(s,e) { ContextMenu(s,e); }"
CustomButtonClick="function(s,e){ if(e.buttonID=='showDoc'){
var rowKey = s.GetRowKey(e.visibleIndex);
OpenPopUpFileManager('Empresa\\'+rowKey);  } }" />

<Columns>

<dx:GridViewDataTextColumn FieldName="idDocumentosEmpresa" ReadOnly="True"
ShowInCustomizationForm="True" VisibleIndex="1" Visible="false">
</dx:GridViewDataTextColumn>

<dx:GridViewDataComboBoxColumn FieldName="tipoDocumento" VisibleIndex="2" Caption="Tipo de
Documento">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Certificado Pagos a Hacienda" Value="Certificado Pagos a Hacienda" />
<dx:ListEditItem Text="Certificado Seguridad Social" Value="Certificado Seguridad Social" />
<dx:ListEditItem Text="Comunicación Apertura Centro de Trabajo" Value="Comunicación Apertura
Centro de Trabajo" />
<dx:ListEditItem Text="Contrato Servicio Prevención" Value="Contrato Servicio Prevención" />
<dx:ListEditItem Text="Convenio Colectivo" Value="Convenio Colectivo" />
<dx:ListEditItem Text="Evaluación de Riesgos" Value="Evaluación de Riesgos" />
<dx:ListEditItem Text="IC / ITA" Value="IC / ITA" />
<dx:ListEditItem Text="Normas Generales Salud" Value="Normas Generales Salud" />
<dx:ListEditItem Text="Libro de Subcontratación" Value="Libro de Subcontratación" />
<dx:ListEditItem Text="Libro de Visitas" Value="Libro de Visitas" />
<dx:ListEditItem Text="Registro inscripción en el REA" Value="Registro inscripción en el REA" />
<dx:ListEditItem Text="Registro inscripción en el RERA" Value="Registro inscripción en el RERA"
/>
<dx:ListEditItem Text="Seguro Responsabilidad Civil" Value="Seguro Responsabilidad Civil" />
<dx:ListEditItem Text="Escrito Subcontratistas" Value="Escrito Subcontratistas" />
<dx:ListEditItem Text="TC1" Value="TC1" />
<dx:ListEditItem Text="TC2" Value="TC2" />
<dx:ListEditItem Text="Pago de la Mutua" Value="Pago de la Mutua" />
<dx:ListEditItem Text="Carta Compromiso GAVSA" Value="Carta Compromiso GAVSA" />
<dx:ListEditItem Text="Otros" Value="Otros" />
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataDateColumn FieldName="fechaCarga" ShowInCustomizationForm="True"
VisibleIndex="3" Caption="Fecha Carga">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaEmision" ShowInCustomizationForm="True"
VisibleIndex="4" Caption="Fecha Emisión">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaCaducidad" ShowInCustomizationForm="True"
VisibleIndex="5" Caption="Fecha de Caducidad">
</dx:GridViewDataDateColumn>
<dx:GridViewDataCheckColumn FieldName="validado" VisibleIndex="6" Caption="Casilla de
Validación">
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarValidado" Cursor="crosshair" OnLoad="img_valido_LoadEmp"
ImageUrl='<%# btnhabilitadoEmp(Convert.ToBoolean(Eval("validado"))) %>'>
</dx:ASPxImage>
```

```
</DataItemTemplate>
</dx:GridViewDataCheckColumn>

<dx:GridViewDataCheckColumn FieldName="historico" ShowInCustomizationForm="True"
VisibleIndex="7" Caption="Histórico" Visible="false">
</dx:GridViewDataCheckColumn>


<dx:GridViewDataTextColumn FieldName="descripcion" ShowInCustomizationForm="True"
VisibleIndex="8" Caption="Descripción">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Empresa_idEmpresa" ShowInCustomizationForm="True"
VisibleIndex="9" Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc" VisibleIndex="10">
<CustomButtons>
<dx:GridViewCommandColumnCustomButton ID="showDoc" Image-Url="Content/Images/document-pdf-
text.png">
<Image Url="Content/Images/document-pdf-text.png">
</Image>
</dx:GridViewCommandColumnCustomButton>
</CustomButtons>
</dx:GridViewCommandColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="11" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsLoadingPanel Mode="Disabled" />
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsBehavior ConfirmDelete="True" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png">
</Image>
<Styles>
<Style ForeColor="White">
</Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png">
</Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png">
</Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png">
</Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png">
</Image>
</CancelButton>
</SettingsCommandButton>

</dx:ASPxGridView>

</dx:ContentControl>
</ContentCollection>
</dx:TabPage>
<dx:TabPage Text="Trabajos Especiales" Visible="true">
<%-- ABRIMOS Pestaña Trabajos Especiales--%>
<ContentCollection>
<dx:ContentControl runat="server">

<dx:ASPxGridView ID="GridTrabajoEspecial" runat="server" AutoGenerateColumns="False"
DataSourceID="EntityTrabajosEspeciales" KeyFieldName="idTrabajoEspecial"
OnCustomErrorText="GridErrorEliminar_CustomErrorText"
OnBeforePerformDataSelect="GridTrabajoEspecial_BeforePerformDataSelect"
```

```
OnRowInserting="GridTrabajoEspecial_RowInserting"
OnRowUpdating="GridTrabajoEspecial_RowUpdating" SettingsBehavior-ConfirmDelete="true"
Width="100%" OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize"
OnRowInserted="Log_RowInserted">

<Columns>
<dx:GridViewDataTextColumn FieldName="idTrabajoEspecial" ReadOnly="True" VisibleIndex="1"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="tipoTrabajo" VisibleIndex="2" Caption="Tipo de
Trabajo">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Amianto" Value="Amianto" />
<dx:ListEditItem Text="Desmontaje de Bombas" Value="Desmontaje de Bombas" />
<dx:ListEditItem Text="Espacios Confinados" Value="Espacios Confinados" />
<dx:ListEditItem Text="Manejo de Cargas" Value="Manejo de Cargas" />
<dx:ListEditItem Text="Productos Químicos" Value="Productos Químicos" />
<dx:ListEditItem Text="Riesgo Eléctrico" Value="Riesgo Eléctrico" />
<dx:ListEditItem Text="Trabajos en Altura" Value="Trabajos en Altura" />
<dx:ListEditItem Text="Otros" Value="Otros" />
</Items>
</PropertiesComboBox>

</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataTextColumn FieldName="descripcion" VisibleIndex="3" Caption="Descripción">
</dx:GridViewDataTextColumn>
<dx:GridViewDataDateColumn FieldName="fechaCaducidad" VisibleIndex="4" Caption="Fecha de
Caducidad del Certificado" Visible="false">
</dx:GridViewDataDateColumn>
<dx:GridViewDataTextColumn FieldName="Empresa_idEmpresa" ReadOnly="False" VisibleIndex="5"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="10" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsBehavior ConfirmDelete="True" />
<SettingsEditing EditFormColumnCount="1" Mode="PopupEditForm" />
<SettingsDetail ShowDetailRow="True" />
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png">
</Image>
<Styles>
<Style ForeColor="White">
</Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png">
</Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png">
</Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png">
</Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png">
</Image>
</CancelButton>
</SettingsCommandButton>

<Templates>

<DetailRow>
```

```
<%-- Documentos Trabajos Especiales--%>
<dx:ASPxGridView ID="GridDocTrabajosEspeciales" runat="server" AutoGenerateColumns="False"
DataSourceID="EntityDocTrabajosEspeciales" KeyFieldName="idDocumentosTrabajo"
OnBeforePerformDataSelect="GridDocTrabajosEspeciales_BeforePerformDataSelect"
OnRowInserting="GridDocTrabajoEspecial_RowInserting"
OnRowUpdating="GridDocTrabajosEspeciales_RowUpdating" SettingsBehavior-ConfirmDelete="true"
Width="100%" OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize"
OnInit="OcultarColumnas_Init" OnHtmlRowPrepared="Tooltip_HtmlRowPrepared"
OnRowInserted="Log_RowInserted">

<ClientSideEvents
RowDblClick="function(s,e) { s.StartEditRow(e.visibleIndex); }"
ContextMenuItemClick="function(s,e) { ContextMenu(s,e); }"
CustomButtonClick="function(s,e){ if(e.buttonID=='showDoc'){
var rowKey = s.GetRowKey(e.visibleIndex);
OpenPopUpFileManager('TrabajosEspeciales\\'+rowKey);  } }" />

<Columns>
<dx:GridViewDataTextColumn FieldName="idDocumentosTrabajo" ReadOnly="True" Visible="False">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="DocumentoTrabajo" Caption="Tipo de Documento">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Acta Coordinación Actividades Empresariales" Value="Acta Coordinación
Actividades Empresariales" />
<dx:ListEditItem Text="Certificado Trabajo Especial" Value="Certificado Trabajo Especial" />
<dx:ListEditItem Text="Procedimiento de Trabajo" Value="Procedimiento de Trabajo" />
<dx:ListEditItem Text="Recurso Preventivo" Value="Recurso Preventivo" />
<dx:ListEditItem Text="Otros" Value="Otros" />
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataDateColumn FieldName="fechaCarga" ShowInCustomizationForm="True" Caption="Fecha
Carga">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaEmision" ShowInCustomizationForm="True"
Caption="Fecha Emisión">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaCaducidad" ShowInCustomizationForm="True"
Caption="Fecha de Caducidad">
</dx:GridViewDataDateColumn>
<dx:GridViewDataCheckColumn FieldName="validado" Caption="Casilla de Validación">
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarValidado" Cursor="crosshair"
ImageUrl='<%#btnhabilitadoEmp(Convert.ToBoolean(Eval("validado"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>
<dx:GridViewDataCheckColumn FieldName="historico" Caption="Histórico" Visible="false">
</dx:GridViewDataCheckColumn>
<dx:GridViewDataTextColumn FieldName="descripcion" Caption="Descripción">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="TrabajoEspecial_idTrabajoEspecial" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="TrabajoEspecial_Empresa_idEmpresa" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc">
<CustomButtons>
<dx:GridViewCommandColumnCustomButton ID="showDoc" Image-Url="Content/Images/document-pdf-
text.png">
</dx:GridViewCommandColumnCustomButton>
</CustomButtons>
</dx:GridViewCommandColumn>
<dx:GridViewCommandColumn ShowEditButton="True" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsCommandButton>
```

```
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-
ForeColor="White"></NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" "></EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" "></DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" "></UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" "></CancelButton>
</SettingsCommandButton>

</dx:ASPxGridView>
</DetailRow>

</Templates>
</dx:ASPxGridView>
</dx:ContentControl>
</ContentCollection>
</dx:TabPage>

<%-- ABRIMOS Pestaña Obras--%>
<dx:TabPage Text="Obras" Visible="true">

<ContentCollection>
<dx:ContentControl runat="server">

<dx:ASPxGridView ID="GridObra" runat="server" AutoGenerateColumns="False" Width="100%"
DataSourceID="EntityObras" KeyFieldName="idObra"
OnBeforePerformDataSelect="GridObra_BeforePerformDataSelect"
OnRowInserting="GridObra_RowInserting" OnRowUpdating="GridObra_RowUpdating"
OnCustomErrorText="GridErrorEliminar_CustomErrorText" SettingsBehavior-ConfirmDelete="true"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize"
OnRowInserted="Log_RowInserted">
<Columns>
<dx:GridViewDataTextColumn FieldName="idObra" ReadOnly="True" VisibleIndex="1" Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="nombreObra" VisibleIndex="2" Caption="Nombre Obra">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="domicilio" VisibleIndex="3" Caption="Direccion">
</dx:GridViewDataTextColumn>
<dx:GridViewDataDateColumn FieldName="fechaInicio" VisibleIndex="4" Caption="Fecha Inicio">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaFin" VisibleIndex="5" Caption="Fecha Fin">
</dx:GridViewDataDateColumn>
<dx:GridViewDataTextColumn FieldName="responsableGavsa" VisibleIndex="7" Caption="Responsable
GAVSA">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="telefonoGavsa" VisibleIndex="7" Caption="Telefono GAVSA">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="correoGavsa" VisibleIndex="7" Caption="email GAVSA">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="responsableContratista" VisibleIndex="7"
Caption="Responsable Contratista">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="telefonoContratista" VisibleIndex="7" Caption="Telefono
Contratista">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="correoContratista" VisibleIndex="7" Caption="email
Contratista">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Empresa_idEmpresa" VisibleIndex="8" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="9" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsBehavior ConfirmDelete="True" />
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsDetail ShowDetailRow="True" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png">
</Image>
```

```
<Styles>
<Style ForeColor="White">
</Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png">
</Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png">
</Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png">
</Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png">
</Image>
</CancelButton>
</SettingsCommandButton>
<Templates>
<DetailRow>
<%--Documentos Obra--%>
<dx:ASPxGridView ID="GridDocObra" runat="server" AutoGenerateColumns="False"
DataSourceID="EntityDocObra" Width="100%" EnableTheming="True" KeyFieldName="idDocumentosObra"
OnBeforePerformDataSelect="GridDocObras_BeforePerformDataSelect"
OnRowInserting="GridDocObras_RowInserting" OnRowUpdating="GridDocObra_RowUpdating"
SettingsBehavior-ConfirmDelete="true"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize" OnInit="OcultarColumnas_Init"
OnHtmlRowPrepared="Tooltip_HtmlRowPrepared" OnRowInserted="Log_RowInserted">
<ClientSideEvents
RowDblClick="function(s,e) { s.StartEditRow(e.visibleIndex); }"
ContextMenuItemClick="function(s,e) { ContextMenu(s,e); }"
CustomButtonClick="function(s,e){ if(e.buttonID=='showDoc'){
var rowKey = s.GetRowKey(e.visibleIndex);
OpenPopUpFileManager('Obras\\'+rowKey);  } }" />

<Columns>

<dx:GridViewDataTextColumn FieldName="idDocumentosObra" ReadOnly="True" VisibleIndex="1"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="tipoDocumento" VisibleIndex="2" Caption="Tipo de
Documento">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Acta Coordinación de Actividades Empresariales" Value="Acta Coordinación
de Actividades Empresariales" />
<dx:ListEditItem Text="Doc. Nombramiento Coordinador" Value="Doc. Nombramiento Coordinador" />
<dx:ListEditItem Text="Plan de Seguridad" Value="Plan de Seguridad" />
<dx:ListEditItem Text="Recurso Preventivo" Value="Recurso Preventivo" />
<dx:ListEditItem Text="Otros" Value="Otros" />
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataDateColumn FieldName="fechaCarga" ShowInCustomizationForm="True"
VisibleIndex="3" Caption="Fecha Carga">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="FechaEmision" ShowInCustomizationForm="True"
VisibleIndex="4" Caption="Fecha Emisión">
</dx:GridViewDataDateColumn>
<dx:GridViewDataCheckColumn FieldName="validado" VisibleIndex="5" Caption="Casilla de
Validación">
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarValidado" Cursor="crosshair"
ImageUrl='<%#btnhabilitadoEmp(Convert.ToBoolean(Eval("validado"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>
<dx:GridViewDataCheckColumn FieldName="historico" VisibleIndex="6" Caption="Historico"
Visible="false">
</dx:GridViewDataCheckColumn>
```

```
<dx:GridViewDataTextColumn FieldName="descripcion" VisibleIndex="7" Caption="Descripción">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Obra_idObra" ReadOnly="False" VisibleIndex="8"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Obra_Empresa_idEmpresa" ReadOnly="False" VisibleIndex="9"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc" VisibleIndex="10">
<CustomButtons>
<dx:GridViewCommandColumnCustomButton ID="showDoc" Image-Url="Content/Images/document-pdf-
text.png">
</dx:GridViewCommandColumnCustomButton>
</CustomButtons>
</dx:GridViewCommandColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="11" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>

</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-
ForeColor="White"></NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" "></EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" "></DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" "></UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" "></CancelButton>
</SettingsCommandButton>

</dx:ASPxGridView>
</DetailRow>

</Templates>
</dx:ASPxGridView>

</dx:ContentControl>
</ContentCollection>
</dx:TabPage>

<%-- ABRIMOS Pestaña Equipos--%>
<dx:TabPage Text="Equipos" Visible="true">

<ContentCollection>
<dx:ContentControl runat="server">

<dx:ASPxGridView ID="GridEquipos" runat="server" AutoGenerateColumns="False"
DataSourceID="EntityEquipos" Width="100%" EnableTheming="True" KeyFieldName="idEquipoVehiculo"
OnBeforePerformDataSelect="GridEquipos_BeforePerformDataSelect"
OnRowInserting="GridEquipos_RowInserting" OnRowUpdating="GridEquipos_RowUpdating"
OnCustomErrorText="GridErrorEliminar_CustomErrorText" SettingsBehavior-ConfirmDelete="true"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize"
OnRowInserted="Log_RowInserted">
<Columns>

<dx:GridViewDataTextColumn FieldName="idEquipoVehiculo" ReadOnly="True" VisibleIndex="1"
Visible="false"></dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="tipoEquipo" VisibleIndex="2" Caption="Tipo de Equipo"
Visible="false">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Equipo" Value="Equipo" />
<dx:ListEditItem Text="Vehículo" Value="Vehiculo" />
</Items>
</PropertiesComboBox>

</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataTextColumn FieldName="descripcion" VisibleIndex="3" Caption="Descripción">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="manual" VisibleIndex="4" Caption="Manual">
</dx:GridViewDataTextColumn>
```

```
<dx:GridViewDataTextColumn FieldName="informacion" VisibleIndex="5" Caption="Información">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="matricula" VisibleIndex="6" Caption="Matrícula (si
tiene)">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="itv" VisibleIndex="7" Caption="ITV" Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Empresa_idEmpresa" VisibleIndex="8" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="9" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsBehavior ConfirmDelete="True" />
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsDetail ShowDetailRow="True" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png">
</Image>
<Styles>
<Style ForeColor="White">
</Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png">
</Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png">
</Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png">
</Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png">
</Image>
</CancelButton>
</SettingsCommandButton>
<Templates>
<DetailRow>
<%--Documentos Equipos--%>
<dx:ASPxGridView ID="GridDocEquipos" runat="server" AutoGenerateColumns="False"
DataSourceID="EntityDocEquipos" Width="100%" KeyFieldName="idDocumentosEquipo"
OnBeforePerformDataSelect="GridDocEquipos_BeforePerformDataSelect"
OnRowInserting="GridDocEquipos_RowInserting" OnRowUpdating="GridDocEquipos_RowUpdating"
SettingsBehavior-ConfirmDelete="true"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize" OnInit="OcultarColumnas_Init"
SettingsDetail-AllowOnlyOneMasterRowExpanded="False" OnHtmlRowPrepared="Tooltip_HtmlRowPrepared"
OnRowInserted="Log_RowInserted">
<ClientSideEvents
RowDblClick="function(s,e) { s.StartEditRow(e.visibleIndex); }"
ContextMenuItemClick="function(s,e) { ContextMenu(s,e); }"
CustomButtonClick="function(s,e){ if(e.buttonID=='showDoc'){
var rowKey = s.GetRowKey(e.visibleIndex);
OpenPopUpFileManager('EquiposVehiculos\\'+rowKey);  } }" />

<Columns>

<dx:GridViewDataTextColumn FieldName="idDocumentosEquipo" ReadOnly="True" Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="tipoDocumento" Caption="Tipo de Documento">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Ficha Técnica" Value="Ficha Técnica" />
<dx:ListEditItem Text="ITV" Value="ITV" />
<dx:ListEditItem Text="Permiso de Circulación" Value="Permiso de Circulación" />
```

```
<dx:ListEditItem Text="Seguro del Vehículo" Value="Seguro del Vehículo" />
<dx:ListEditItem Text="Tarjeta de Transporte" Value="Tarjeta de Transporte" />
<dx:ListEditItem Text="Otros" Value="Otros" />
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataDateColumn FieldName="fechaCarga" ShowInCustomizationForm="True" Caption="Fecha
Carga">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaEmision" ShowInCustomizationForm="True"
Caption="Fecha Emisión">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaCaducidad" ShowInCustomizationForm="True"
Caption="Fecha de Caducidad">
</dx:GridViewDataDateColumn>
<dx:GridViewDataCheckColumn FieldName="validado" Caption="Casilla de Validación">
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarValidado" Cursor="crosshair"
ImageUrl='<%#btnhabilitadoEmp(Convert.ToBoolean(Eval("validado"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>
<dx:GridViewDataCheckColumn FieldName="historico" Caption="Historico" Visible="false">
</dx:GridViewDataCheckColumn>
<dx:GridViewDataTextColumn FieldName="descripcion" Caption="Descripción">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="EquipoVehiculo_idEquipoVehiculo" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="EquipoVehiculo_Empresa_idEmpresa" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc">
<CustomButtons>
<dx:GridViewCommandColumnCustomButton ID="showDoc" Image-Url="Content/Images/document-pdf-
text.png">
</dx:GridViewCommandColumnCustomButton>
</CustomButtons>
</dx:GridViewCommandColumn>
<dx:GridViewCommandColumn ShowEditButton="True" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>

</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-
ForeColor="White"></NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" "></EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" "></DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" "></UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" "></CancelButton>
</SettingsCommandButton>
</dx:ASPxGridView>
</DetailRow>

</Templates>
</dx:ASPxGridView>

</dx:ContentControl>
</ContentCollection>
</dx:TabPage>

<%-- ABRIMOS Pestaña Vehiculos--%>
<dx:TabPage Text="Vehiculos" Visible="true">
<ContentCollection>
<dx:ContentControl runat="server">

<dx:ASPxGridView ID="GridVehiculos" runat="server" AutoGenerateColumns="False"
DataSourceID="EntityVehiculos" Width="100%" EnableTheming="True" KeyFieldName="idEquipoVehiculo"
OnBeforePerformDataSelect="GridVehiculos_BeforePerformDataSelect"
```

```
OnRowInserting="GridVehiculos_RowInserting" OnRowUpdating="GridEquipos_RowUpdating"
OnCustomErrorText="GridErrorEliminar_CustomErrorText" SettingsBehavior-ConfirmDelete="true"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize"
OnRowInserted="Log_RowInserted">
<Columns>

<dx:GridViewDataTextColumn FieldName="idEquipoVehiculo" ReadOnly="True" VisibleIndex="1"
Visible="false"></dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="tipoEquipo" VisibleIndex="2" Caption="Tipo de Equipo"
Visible="false">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Equipo" Value="Equipo" />
<dx:ListEditItem Text="Vehículo" Value="Vehiculo" />
</Items>
</PropertiesComboBox>

</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataTextColumn FieldName="descripcion" VisibleIndex="3" Caption="Descripción">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="manual" VisibleIndex="4" Caption="Manual">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="informacion" VisibleIndex="5" Caption="Información">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="matricula" VisibleIndex="6" Caption="Matrícula (si
tiene)">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="itv" VisibleIndex="7" Caption="ITV" Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Empresa_idEmpresa" VisibleIndex="8" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="9" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsBehavior ConfirmDelete="True" />
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsDetail ShowDetailRow="True" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png">
</Image>
<Styles>
<Style ForeColor="White">
</Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png">
</Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png">
</Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png">
</Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png">
</Image>
</CancelButton>
</SettingsCommandButton>
<Templates>
<DetailRow>
<%--Documentos Equipos--%>
<dx:ASPxGridView ID="GridDocEquipos" runat="server" AutoGenerateColumns="False"
DataSourceID="EntityDocEquipos" Width="100%" KeyFieldName="idDocumentosEquipo"
OnBeforePerformDataSelect="GridDocEquipos_BeforePerformDataSelect"
```

```
OnRowInserting="GridDocEquipos_RowInserting" OnRowUpdating="GridDocEquipos_RowUpdating"
SettingsBehavior-ConfirmDelete="true"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize" OnInit="OcultarColumnas_Init"
SettingsDetail-AllowOnlyOneMasterRowExpanded="False" OnHtmlRowPrepared="Tooltip_HtmlRowPrepared"
OnRowInserted="Log_RowInserted">
<ClientSideEvents
RowDblClick="function(s,e) { s.StartEditRow(e.visibleIndex); }"
ContextMenuItemClick="function(s,e) { ContextMenu(s,e); }"
CustomButtonClick="function(s,e){ if(e.buttonID=='showDoc'){
var rowKey = s.GetRowKey(e.visibleIndex);
OpenPopUpFileManager('EquiposVehiculos\\'+rowKey);  } }" />

<Columns>

<dx:GridViewDataTextColumn FieldName="idDocumentosEquipo" ReadOnly="True" Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="tipoDocumento" Caption="Tipo de Documento">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Ficha Técnica" Value="Ficha Técnica" />
<dx:ListEditItem Text="ITV" Value="ITV" />
<dx:ListEditItem Text="Permiso de Circulación" Value="Permiso de Circulación" />
<dx:ListEditItem Text="Seguro del Vehículo" Value="Seguro del Vehículo" />
<dx:ListEditItem Text="Tarjeta de Transporte" Value="Tarjeta de Transporte" />
<dx:ListEditItem Text="Otros" Value="Otros" />
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataDateColumn FieldName="fechaCarga" ShowInCustomizationForm="True" Caption="Fecha
Carga">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaEmision" ShowInCustomizationForm="True"
Caption="Fecha Emisión">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaCaducidad" ShowInCustomizationForm="True"
Caption="Fecha de Caducidad">
</dx:GridViewDataDateColumn>
<dx:GridViewDataCheckColumn FieldName="validado" Caption="Casilla de Validación">
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarValidado" Cursor="crosshair"
ImageUrl='<%#btnhabilitadoEmp(Convert.ToBoolean(Eval("validado"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>
<dx:GridViewDataCheckColumn FieldName="historico" Caption="Historico" Visible="false">
</dx:GridViewDataCheckColumn>
<dx:GridViewDataTextColumn FieldName="descripcion" Caption="Descripción">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="EquipoVehiculo_idEquipoVehiculo" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="EquipoVehiculo_Empresa_idEmpresa" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc">
<CustomButtons>
<dx:GridViewCommandColumnCustomButton ID="showDoc" Image-Url="Content/Images/document-pdf-
text.png">
</dx:GridViewCommandColumnCustomButton>
</CustomButtons>
</dx:GridViewCommandColumn>
<dx:GridViewCommandColumn ShowEditButton="True" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>

</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-
ForeColor="White"></NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" "></EditButton>
```

```
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" "></DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" "></UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" "></CancelButton>
</SettingsCommandButton>
</dx:ASPxGridView>
</DetailRow>
<%--Documentos Equipos--%>
</Templates>
</dx:ASPxGridView>

</dx:ContentControl>
</ContentCollection>
</dx:TabPage>

<%-- ABRIMOS PestañaTrabajadores--%>

<dx:TabPage Text="Trabajadores" Visible="true">
<ContentCollection>
<dx:ContentControl runat="server">

<dx:ASPxGridView ID="GridTrabajador" ClientInstanceName="GridTrabajador" runat="server"
AutoGenerateColumns="False" DataSourceID="EntityTrabajadores" Width="100%" EnableTheming="True"
KeyFieldName="idTrabajador" OnBeforePerformDataSelect="GridTrabajador_BeforePerformDataSelect"
OnRowInserting="GridTrabajador_RowInserting" OnRowUpdating="GridTrabajador_RowUpdating"
OnHtmlRowPrepared="GridTrabajador_HtmlRowPrepared"
OnCustomErrorText="GridErrorEliminar_CustomErrorText" SettingsBehavior-ConfirmDelete="true"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize"
OnInit="OcultarColumnasTrabajadores_Init" SettingsDetail-ShowDetailRow="False"
OnRowInserted="Log_RowInserted">
<Columns>
<dx:GridViewDataTextColumn FieldName="idTrabajador" ReadOnly="True" VisibleIndex="1"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="nombre" VisibleIndex="2" Caption="Nombre">
</dx:GridViewDataTextColumn>
<dx:GridViewDataCheckColumn FieldName="esVerde" Caption="Estado" EditFormSettings-
Visible="False" ReadOnly="true" Width="30" VisibleIndex="3">
<EditFormSettings Visible="False" />
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarVerde" Cursor="crosshair" ImageUrl='<%#
btnVerdeEmp(Convert.ToBoolean(Eval("esVerde"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>
<dx:GridViewDataCheckColumn FieldName="historico" Caption="Historico" EditFormSettings-
Visible="False" Width="30" VisibleIndex="3">

<EditFormSettings Visible="False" />

</dx:GridViewDataCheckColumn>
<dx:GridViewDataTextColumn FieldName="DNI" VisibleIndex="4" Caption="DNI">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="puestoTrabajo" VisibleIndex="5" Caption="Puesto de
Trabajo">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="informacion" VisibleIndex="6" Caption="Información">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Empresa_idEmpresa" VisibleIndex="7" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="8" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<Settings ShowFilterBar="Visible" ShowFilterRow="true" ShowFilterRowMenu="true"
ShowFilterRowMenuLikeItem="true" ShowHeaderFilterButton="true" />
<SettingsBehavior ConfirmDelete="True" />
<SettingsDetail />
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsDetail ShowDetailRow="True" />
<SettingsCommandButton>
```

```
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png">
</Image>
<Styles>
<Style ForeColor="White">
</Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png">
</Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png">
</Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png">
</Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png">
</Image>
</CancelButton>
</SettingsCommandButton>
<Templates>
<DetailRow>
<%--Documentos Trabajadores--%>
<dx:ASPxGridView ID="GridDocTrabajadores" ClientInstanceName="DocTrabajadores" runat="server"
AutoGenerateColumns="False" DataSourceID="EntityDocTrabajadores" Width="100%"
KeyFieldName="idDocumentosTrabajor"
OnBeforePerformDataSelect="GridDocTrabajadores_BeforePerformDataSelect"
OnRowInserting="GridDocTrabajadores_RowInserting"
OnRowUpdating="GridDocTrabajadores_RowUpdating" OnRowUpdated="DocTrabajadores_RowUpdated"
OnRowInserted="DocTrabajadores_RowInserted" SettingsBehavior-ConfirmDelete="true"
OnCustomCallback="DocTrabajadoresEmp_CustomCallback"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize" OnInit="OcultarColumnas_Init"
OnHtmlRowPrepared="Tooltip_HtmlRowPrepared">
<ClientSideEvents
EndCallback="function(s,e) { if (s.cpUpdate) { if (s.cpUpdate == '1') {
GridTrabajador.Refresh(); GridEmpresa.Refresh();  delete s.cpUpdate; } } }"
RowDblClick="function(s,e) { s.StartEditRow(e.visibleIndex); }"
ContextMenuItemClick="function(s,e) { ContextMenu(s,e); }"
CustomButtonClick="function(s,e){ if(e.buttonID=='showDoc'){
var rowKey = s.GetRowKey(e.visibleIndex);
OpenPopUpFileManager('Trabajadores\\'+rowKey);  } }" />

<Columns>

<dx:GridViewDataTextColumn FieldName="idDocumentosTrabajor" ReadOnly="True" Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="tipoDocumento" Caption="Tipo de Documento"
Width="150px">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Entrega Información" Value="Entrega Información" />
<dx:ListEditItem Text="Formación" Value="Formación" />
<dx:ListEditItem Text="Hoja Entrega de EPIS" Value="Hoja Entrega de EPIS" />
<dx:ListEditItem Text="Nombramiento Recurso Preventivo" Value="Nombramiento Recurso Preventivo"
/>
<dx:ListEditItem Text="Nombramiento Trabajador autorizado/cualificado" Value="Nombramiento
Trabajador autorizado/cualificado" />
<dx:ListEditItem Text="Reconocimiento Médico" Value="Reconocimiento Médico" />
<dx:ListEditItem Text="DNI" Value="DNI" />
<dx:ListEditItem Text="Otros" Value="Otros" />
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataDateColumn FieldName="fechaCarga" ShowInCustomizationForm="True" Caption="Fecha
Carga">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaEmision" ShowInCustomizationForm="True"
Caption="Fecha Emisión">
</dx:GridViewDataDateColumn>
```

```
<dx:GridViewDataDateColumn FieldName="fechaCaducidad" ShowInCustomizationForm="True"
Caption="Fecha de Caducidad">
</dx:GridViewDataDateColumn>
<%--<dx:GridViewDataCheckColumn FieldName="validado" VisibleIndex="5" Caption="Casilla de
Validación">
</dx:GridViewDataCheckColumn>--%>

<dx:GridViewDataCheckColumn FieldName="validado" Caption="Casilla de Validación">
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarValidado" Cursor="crosshair"
OnLoad="img_valido_LoadTraEmp"
ImageUrl='<%#btnhabilitadoEmp(Convert.ToBoolean(Eval("validado"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>

<dx:GridViewDataCheckColumn FieldName="historico" Caption="Histórico" Visible="false">
</dx:GridViewDataCheckColumn>
<dx:GridViewDataTextColumn FieldName="descripcion"
Caption="Descripción"></dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Trabajador_idTrabajador" ReadOnly="False"
Visible="false"></dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Trabajador_Empresa_idEmpresa" ReadOnly="False"
Visible="false"></dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc">
<CustomButtons>
<dx:GridViewCommandColumnCustomButton ID="showDoc" Image-Url="Content/Images/document-pdf-
text.png">
</dx:GridViewCommandColumnCustomButton>
</CustomButtons>
</dx:GridViewCommandColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="9" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsEditing Mode="Inline"></SettingsEditing>
<SettingsLoadingPanel Mode="Disabled" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-
ForeColor="White"></NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" "></EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" "></DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" "></UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" "></CancelButton>
</SettingsCommandButton>

</dx:ASPxGridView>
</DetailRow>
<%--Documentos Trabajadores--%>
</Templates>
</dx:ASPxGridView>

</dx:ContentControl>
</ContentCollection>
</dx:TabPage>

</TabPages>
</dx:ASPxPageControl>
<%-- CERRAMOS CONTROL DE PESTAÑAS--%>
</DetailRow>
</Templates>
</dx:ASPxGridView>


<%--DATASOURCES--%>

<asp:EntityDataSource runat="server" ID="EntityTrabajosEspeciales"
DefaultContainerName="proveedoresEntities" ConnectionString="name=proveedoresEntities"
EnableFlattening="False" EntitySetName="trabajoespecials" EnableDelete="True"
EnableInsert="True" EnableUpdate="True" Where="it.[Empresa_idEmpresa] = @paridEmpresa">
<WhereParameters>
```

```
<asp:SessionParameter DbType="Int32" Name="paridEmpresa" SessionField="idEmpresa" />
</WhereParameters>
</asp:EntityDataSource>

<asp:EntityDataSource ID="EntityDocTrabajosEspeciales" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="documentostrabajoes" Where="it.[TrabajoEspecial_idTrabajoEspecial] =
@paridTrabajosEspeciales">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridTrabajosEspeciales"
SessionField="idTrabajosEspeciales" />
</WhereParameters>
</asp:EntityDataSource>


<asp:EntityDataSource ID="EntityDocEmpresa" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="documentosempresas" Where="it.[Empresa_idEmpresa] = @paridEmpresa">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridEmpresa" SessionField="idEmpresa" />
</WhereParameters>
</asp:EntityDataSource>


<asp:EntityDataSource ID="EntityEquipos" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="equipovehiculoes" Where="it.[Empresa_idEmpresa] = @paridEmpresa AND
it.[tipoEquipo]= @tipo">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridEmpresa" SessionField="idEmpresa" />
<asp:SessionParameter DbType="String" Name="tipo" SessionField="tipo" />
</WhereParameters>
</asp:EntityDataSource>

<asp:EntityDataSource ID="EntityVehiculos" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="equipovehiculoes" Where="it.[Empresa_idEmpresa] = @paridEmpresa AND
it.[tipoEquipo]= @tipo">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridEmpresa" SessionField="idEmpresa" />
<asp:SessionParameter DbType="String" Name="tipo" SessionField="tipo" />
</WhereParameters>
</asp:EntityDataSource>


<asp:EntityDataSource ID="EntityDocEquipos" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="documentosequipoes" Where="it.[EquipoVehiculo_idEquipoVehiculo] =
@paridEquipoVehiculo">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridEquipoVehiculo" SessionField="idEquipos" />
</WhereParameters>
</asp:EntityDataSource>


<asp:EntityDataSource ID="EntityObras" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="obras" Where="it.[Empresa_idEmpresa] = @paridEmpresa">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridEmpresa" SessionField="idEmpresa" />
</WhereParameters>
</asp:EntityDataSource>

<asp:EntityDataSource ID="EntityDocObra" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="documentosobras" Where="it.[Obra_idObra] = @paridObra">
<WhereParameters>
```

```
<asp:SessionParameter DbType="Int32" Name="paridObra" SessionField="idObra" />
</WhereParameters>
</asp:EntityDataSource>


<asp:EntityDataSource ID="EntityTrabajadores" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="trabajadors" OnInserted="EntityTrabajadores_Inserted"
Where="it.[Empresa_idEmpresa] = @paridEmpresa">
<InsertParameters>
<asp:Parameter Direction="ReturnValue" Name="idTrabajador" Type="Int32" />
</InsertParameters>
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridEmpresa" SessionField="idEmpresa" />
</WhereParameters>
</asp:EntityDataSource>

<asp:EntityDataSource ID="EntityDocTrabajadores" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="documentostrabajors" Where="it.[Trabajador_idTrabajador] = @paridTrabajadores">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridTrabajadores" SessionField="idTrabajadores" />
</WhereParameters>
</asp:EntityDataSource>


<asp:EntityDataSource runat="server" ID="EntityDataSource1"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="empresas" OnInserted="EntityDataSource1_Inserted">
<InsertParameters>
<asp:Parameter Direction="ReturnValue" Name="idEmpresa" Type="Int32" />
</InsertParameters>
</asp:EntityDataSource>


<asp:EntityDataSource ID="EntitySubcontrata" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EntitySetName="subcontratas" EnableFlattening="False" EnableDelete="True" EnableInsert="True"
EnableUpdate="True" Where="it.[idContratante] = @paridEmpresa" Include="empresa,
empresa.subcontratas">

<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridEmpresa" SessionField="idEmpresa" />
</WhereParameters>

</asp:EntityDataSource>



</asp:Content>
```

11.2.2. Default.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using DevExpress.Web;
using DevExpress.Web.Data;
using System.Data;
using System.Data.Entity.Core;
using proveedoresX.Clases;

namespace proveedoresX
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_PreInit(object sender, EventArgs e)
        {
            sessionVars.Autologon();
        }

        protected void Page_Init(object sender, EventArgs e)
        {

            if (!sessionVars._UserLogged.Admin)//este es para los clientes(empresas)
            {
                EntityDataSource1.Where = "it.[idEmpresa] = "+ sessionVars._UserLogged.Empresa;
//+1
                //EntityTrabajadores.Where = "it.[historico] == False";


                cls_proveedores.DeshabilitarEdicionGrid(GridEmpresa); //impide columna edicion a
usuario no administrador


            }

            //var lal = new ASPxGridView();
            //Form.Controls.Add(lal);
        }

        protected void Page_Load(object sender, EventArgs e)
        {

            var lolo = sessionVars._UserLogged;


        }

        #region METODOS EMPRESA


        protected void GridEmpresa_HtmlDataCellPrepared(object sender,
ASPxGridViewTableDataCellEventArgs e)
        {

            if (e.DataColumn.FieldName == "nombreEmpresa")
            {
                if (e.GetValue("esVerde") != null && e.GetValue("esVerde").Equals(false))
                {
                    e.Cell.BackColor = System.Drawing.Color.Tomato;
                    e.Cell.ForeColor = System.Drawing.Color.White;
                }
            }
        }

        protected void GridEmpresa_HtmlRowPrepared(object sender, ASPxGridViewTableRowEventArgs
e)//pintar row trabajadores historicos
        {
            if (e.GetValue("historico") != null && e.GetValue("historico").Equals(true))
            {
                e.Row.BackColor = System.Drawing.Color.LightSlateGray;
```

```
                e.Row.ForeColor = System.Drawing.Color.White;
            }
        }


        protected void GridEmpresa_RowInserted(object sender, ASPxDataInsertedEventArgs e)
        {
            ASPxGridView row = (sender as ASPxGridView);
            Administration.fnc_log_añadir("Insertar Empresa", Administration.fnc_log_dame(e));
        }

        //protected void GridEmpresa_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        //{
        //    Administration.fnc_log_añadir("Actualizar Empresa",
Administration.fnc_log_dame(e));
        //}

        #endregion

        #region METODOS TRABAJAO ESPECIAL

        //Métodos Grid Trabajo Especial
        protected void GridTrabajoEspecial_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
        }

        protected void GridTrabajoEspecial_RowInserting(object sender,
ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();

        }

        protected void GridTrabajoEspecial_RowUpdating(object sender, ASPxDataUpdatingEventArgs
e)
        {
            e.OldValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["idTrabajoEspecial"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idTrabajoEspecial" });

            e.NewValues["Empresa_idEmpresa"] = e.OldValues["Empresa_idEmpresa"];
            e.NewValues["idTrabajoEspecial"] = e.OldValues["idTrabajoEspecial"];
            Administration.fnc_log_añadir("Actualizar Trab. Especial",
Administration.fnc_log_dame(e));

        }



        #endregion

        #region METODOS OBRA
        //Métodos Grid OBRA
        protected void GridObra_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
        }

        protected void GridObra_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();


        }

        protected void GridObra_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["idObra"] = ((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new
string[] { "idObra" });

            e.NewValues["Empresa_idEmpresa"] = e.OldValues["Empresa_idEmpresa"];
            e.NewValues["idObra"] = e.OldValues["idObra"];
```

```
            Administration.fnc_log_añadir("Actualizar Trab. Especial",
Administration.fnc_log_dame(e));

        }

        #endregion

        #region METODOS EQUIPOS
        //Métodos Grid EQUIPOS
        protected void GridEquipos_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            Session["tipo"] = "Equipo";
        }

        protected void GridVehiculos_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            Session["tipo"] = "Vehiculo";
        }

        protected void GridEquipos_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.NewValues["tipoEquipo"] = "Equipo";

        }

        protected void GridVehiculos_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.NewValues["tipoEquipo"] = "Vehiculo";

        }

        protected void GridEquipos_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["idEquipoVehiculo"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idEquipoVehiculo" });

            e.NewValues["Empresa_idEmpresa"] = e.OldValues["Empresa_idEmpresa"];
            e.NewValues["idEquipoVehiculo"] = e.OldValues["idEquipoVehiculo"];

            Administration.fnc_log_añadir("Actualizar Equipos", Administration.fnc_log_dame(e));

        }

        #endregion

        #region METODOS TRABAJADOR
        //Métodos Grid TRABAJADOR
        protected void GridTrabajador_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
        }

        protected void GridTrabajador_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();

        }

        protected void GridTrabajador_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["idTrabajador"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idTrabajador" });

            e.NewValues["Empresa_idEmpresa"] = e.OldValues["Empresa_idEmpresa"];
            e.NewValues["idTrabajador"] = e.OldValues["idTrabajador"];
            Administration.fnc_log_añadir("Actualizar Trabajador",
Administration.fnc_log_dame(e));
```

```
        }

        protected void GridTrabajador_HtmlRowPrepared(object sender,
ASPxGridViewTableRowEventArgs e)//pintar row trabajadores historicos
        {
            if (e.GetValue("historico") != null && e.GetValue("historico").Equals(true))
            {
                e.Row.BackColor = System.Drawing.Color.LightSlateGray;
                e.Row.ForeColor = System.Drawing.Color.White;

            }
        }

        #endregion

        #region DOC TRABAJOS ESPECIALES

        //Métodos Grid DOCUMENTOS TRABAJOS ESPECIALES
        protected void GridDocTrabajosEspeciales_BeforePerformDataSelect(object sender,
EventArgs e)
        {
            Session["idTrabajosEspeciales"] = (sender as ASPxGridView).GetMasterRowKeyValue();

        }

        protected void GridDocTrabajoEspecial_RowInserting(object sender,
ASPxDataInsertingEventArgs e)
        {
            e.NewValues["TrabajoEspecial_idTrabajoEspecial"] = (sender as
ASPxGridView).GetMasterRowKeyValue();//obtiene KEY padre
            e.NewValues["TrabajoEspecial_Empresa_idEmpresa"] = (sender as
ASPxGridView).GetMasterRowFieldValues("Empresa_idEmpresa");//obtiene un valor determinado del
padre
            e.NewValues["fechaCarga"] = DateTime.Now;
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;



        }

        protected void GridDocTrabajosEspeciales_RowUpdating(object sender,
ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["TrabajoEspecial_idTrabajoEspecial"] = (sender as
ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["TrabajoEspecial_Empresa_idEmpresa"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] {
"TrabajoEspecial_Empresa_idEmpresa" });
            e.OldValues["idDocumentosTrabajo"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idDocumentosTrabajo"
});


            e.NewValues["TrabajoEspecial_idTrabajoEspecial"] =
e.OldValues["TrabajoEspecial_idTrabajoEspecial"];
            e.NewValues["TrabajoEspecial_Empresa_idEmpresa"] =
e.OldValues["TrabajoEspecial_Empresa_idEmpresa"];
            e.NewValues["idDocumentosTrabajo"] = e.OldValues["idDocumentosTrabajo"];
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;


            Administration.fnc_log_añadir("Actualizar Documento Trabajo Especial",
Administration.fnc_log_dame(e));
        }

        #endregion

        #region DOC OBRAS
        //Métodos Grid DOCUMENTOS OBRAS
        protected void GridDocObras_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idObra"] = (sender as ASPxGridView).GetMasterRowKeyValue();
        }
```

```
        protected void GridDocObras_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Obra_idObra"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.NewValues["Obra_Empresa_idEmpresa"] = (sender as
ASPxGridView).GetMasterRowFieldValues("Empresa_idEmpresa");
            e.NewValues["fechaCarga"] = DateTime.Now;
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;


        }

        protected void GridDocObra_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["Obra_idObra"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["Obra_Empresa_idEmpresa"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "Obra_Empresa_idEmpresa"
});
            e.OldValues["idDocumentosObra"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idDocumentosObra" });


            e.NewValues["Obra_idObra"] = e.OldValues["Obra_idObra"];
            e.NewValues["Obra_Empresa_idEmpresa"] = e.OldValues["Obra_Empresa_idEmpresa"];
            e.NewValues["idDocumentosObra"] = e.OldValues["idDocumentosTrabajo"];
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;


            Administration.fnc_log_añadir("Actualizar Documento Obra",
Administration.fnc_log_dame(e));
        }

        #endregion

        #region DOC EQUIPOS
        //Métodos Grid DOCUMENTOS EQUIPOS
        protected void GridDocEquipos_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idEquipos"] = (sender as ASPxGridView).GetMasterRowKeyValue();
        }
        protected void GridDocEquipos_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["EquipoVehiculo_idEquipoVehiculo"] = (sender as
ASPxGridView).GetMasterRowKeyValue();
            e.NewValues["EquipoVehiculo_Empresa_idEmpresa"] = (sender as
ASPxGridView).GetMasterRowFieldValues("Empresa_idEmpresa");
            e.NewValues["fechaCarga"] = DateTime.Now;
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;
        }
        protected void GridDocEquipos_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["EquipoVehiculo_idEquipoVehiculo"] = (sender as
ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["EquipoVehiculo_Empresa_idEmpresa"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] {
"EquipoVehiculo_Empresa_idEmpresa" });
            e.OldValues["idDocumentosEquipo"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idDocumentosEquipo" });


            e.NewValues["EquipoVehiculo_idEquipoVehiculo"] =
e.OldValues["EquipoVehiculo_idEquipoVehiculo"];
            e.NewValues["EquipoVehiculo_Empresa_idEmpresa"] =
e.OldValues["EquipoVehiculo_Empresa_idEmpresa"];
            e.NewValues["idDocumentosEquipo"] = e.OldValues["idDocumentosEquipo"];
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;


            Administration.fnc_log_añadir("Actualizar Documento Equipos",
Administration.fnc_log_dame(e));
        }

        #endregion

        #region DOC TRABAJADORES
```

```
        //Métodos Grid DOCUMENTOS TRABAJADORES
        protected void GridDocTrabajadores_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idTrabajadores"] = (sender as ASPxGridView).GetMasterRowKeyValue();
        }

        protected void GridDocTrabajadores_RowInserting(object sender,
ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Trabajador_idTrabajador"] = (sender as
ASPxGridView).GetMasterRowKeyValue();
            e.NewValues["Trabajador_Empresa_idEmpresa"] = (sender as
ASPxGridView).GetMasterRowFieldValues("Empresa_idEmpresa");
            e.NewValues["fechaCarga"] = DateTime.Now;
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;

        }

        protected void GridDocTrabajadores_RowUpdating(object sender, ASPxDataUpdatingEventArgs
e)
        {

            e.OldValues["Trabajador_idTrabajador"] = (sender as
ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["Trabajador_Empresa_idEmpresa"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] {
"Trabajador_Empresa_idEmpresa" });
            e.OldValues["idDocumentosTrabajor"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idDocumentosTrabajor"
});


            e.NewValues["Trabajador_idTrabajador"] = e.OldValues["Trabajador_idTrabajador"];
            e.NewValues["Trabajador_Empresa_idEmpresa"] =
e.OldValues["Trabajador_Empresa_idEmpresa"];
            e.NewValues["idDocumentosTrabajor"] = e.OldValues["idDocumentosTrabajor"];
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;


            Administration.fnc_log_añadir("Actualizar Documento Trabajador",
Administration.fnc_log_dame(e));

         }

        protected void DocTrabajadores_RowInserted(object sender, ASPxDataInsertedEventArgs e)
        {
            RefrescaGridTra(sender);
            Administration.fnc_log_añadir("Insertar Trabajador",
Administration.fnc_log_dame(e));

        }

        protected void DocTrabajadores_RowUpdated(object sender, ASPxDataUpdatedEventArgs e)
        {

            RefrescaGridTra(sender);
            // RefrescaGrid(GridEmpresa); // creo  que va aqui

        }

        protected void DocTrabajadoresEmp_CustomCallback(object sender,
ASPxGridViewCustomCallbackEventArgs e)
        {
            ASPxGridView grid = (sender as ASPxGridView);

            if (!string.IsNullOrEmpty(e.Parameters))
            {
                var datos = e.Parameters.Split('|');

                if (datos[0] == "CambiarValidado")
                {
                    int visibleindex = Convert.ToInt32(datos[1]);
```

```
                    bool esValido = Convert.ToBoolean(grid.GetRowValues(visibleindex,
"validado"));
                    int idDocumentosTrabajor = Convert.ToInt32(grid.GetRowValues(visibleindex,
"idDocumentosTrabajor"));

                    cls_proveedores.CambiarValidezDocumento(idDocumentosTrabajor, !esValido);

                    grid.JSProperties["cpUpdate"] = "1";


                }
                //else if (datos[0] == "CambiarHistorico")
                //{
                //    int visibleindex = Convert.ToInt32(datos[1]);
                //    bool esValido = Convert.ToBoolean(grid.GetRowValues(visibleindex,
"historico"));
                //    int idDocumentosTrabajor = Convert.ToInt32(grid.GetRowValues(visibleindex,
"idDocumentosTrabajor"));

                //    cls_proveedores.CambiarHistoricoDocumento(idDocumentosTrabajor,
!esValido);

                //    //grid.JSProperties["cpUpdate"] = "1";
                //    grid.DataBind();
                //}

            }
        }

        #endregion

        #region DOC EMPRESA
        //Métodos Grid DOCUMENTOS EMPRESA
        protected void GridDocEmpresa_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();

        }

        protected void GridDocEmpresa_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.NewValues["fechaCarga"] = DateTime.Now;
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;
        }

        protected void GridDocEmpresa_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["Empresa_idEmpresa"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["idDocumentosEmpresa"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idDocumentosEmpresa"
});

            e.NewValues["Empresa_idEmpresa"] = e.OldValues["Empresa_idEmpresa"];
            e.NewValues["idDocumentosEmpresa"] = e.OldValues["idDocumentosEmpresa"];
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;


            Administration.fnc_log_añadir("Actualizar Documento Empresa",
Administration.fnc_log_dame(e));


        }

        protected void GridDocEmpresa_RowUpdated(object sender, ASPxDataUpdatedEventArgs e)
        {
            RefrescaGrid(sender);

        }

        protected void GridDocEmpresa_RowInserted(object sender, ASPxDataInsertedEventArgs e)
        {
            RefrescaGrid(sender);
```

```
            Administration.fnc_log_añadir("Insertar Doc. Empresa",
Administration.fnc_log_dame(e));
        }

        protected void GridDocEmpresa_CustomErrorText(object sender,
ASPxGridViewCustomErrorTextEventArgs e)
        {
            WebLoggerHelper.i().Error(e.Exception);
            e.ErrorText = e.Exception.InnerException.Message + e.Exception.StackTrace;
        }

        protected void DocEmpresa_CustomCallback(object sender,
ASPxGridViewCustomCallbackEventArgs e)
        {
            ASPxGridView grid = (sender as ASPxGridView);

            if (!string.IsNullOrEmpty(e.Parameters))
            {
                var datos = e.Parameters.Split('|');

                if (datos[0] == "CambiarValidado")
                {
                    int visibleindex = Convert.ToInt32(datos[1]);
                    bool esValido = Convert.ToBoolean(grid.GetRowValues(visibleindex,
"validado"));
                    int idDocumentosEmpresa = Convert.ToInt32(grid.GetRowValues(visibleindex,
"idDocumentosEmpresa"));

                    cls_proveedores.CambiarValidezDocumentoEmpresa(idDocumentosEmpresa,
!esValido);

                    grid.JSProperties["cpUpdate"] = "1";
                }



            }
        }

        #endregion

        #region CONTROL SEMAFOROS Y EDICION VALIDO

        private void RefrescaGridTra(object sender)
        {
            ASPxGridView grid = (sender as ASPxGridView);
            int padre = Convert.ToInt32(grid.GetMasterRowKeyValue());
            // int empresaPadre = Convert.ToInt32(grid.); revisar

            cls_proveedores.trabajadorValido(padre);
            //cls_proveedores.empresaValido(empresaPadre); // añadido para comprobar tambien sus
trabajadores

            grid.JSProperties["cpUpdate"] = "1";

        }

        private void RefrescaGrid(object sender)
        {
            ASPxGridView grid = (sender as ASPxGridView);
            int padre = Convert.ToInt32(grid.GetMasterRowKeyValue());

            cls_proveedores.empresaValido(padre);

            grid.JSProperties["cpUpdate"] = "1";
        }


        protected string btnVerdeEmp(bool esVerde)
        {
            return (esVerde) ? "~/Content/Images/green.png" : "~/Content/Images/red.png";
        }
```

```
        protected string btnhabilitadoEmp(bool habilitado)
        {
            return (habilitado) ? "~/Content/Images/accept.png" : "~/Content/Images/delete.png";

        }


        protected void img_valido_LoadEmp(object sender, EventArgs e)
        {

            ASPxImage image = (sender as ASPxImage);
            GridViewDataItemTemplateContainer container = (image.NamingContainer as
GridViewDataItemTemplateContainer);

            image.ClientSideEvents.Click = String.Format(@"function (s, e) {{
GridDocEmpresa.PerformCallback(""{1}|"" + {0}); }}", container.VisibleIndex, image.ID);

        }


        protected void img_valido_LoadTraEmp(object sender, EventArgs e)
        {

            ASPxImage image = (sender as ASPxImage);
            GridViewDataItemTemplateContainer container = (image.NamingContainer as
GridViewDataItemTemplateContainer);

            image.ClientSideEvents.Click = String.Format(@"function (s, e) {{
DocTrabajadores.PerformCallback(""{1}|"" + {0}); }}", container.VisibleIndex, image.ID);

        }



        #endregion

        #region AÑADE ROWS AUTOMATICAMENTE A LOS DETAIL HIJOS DOCEMPRESA Y DOCTRABAJADORES

        protected void EntityDataSource1_Inserted(object sender,
EntityDataSourceChangedEventArgs e)
        {
            empresa newlyAdded = (empresa)e.Entity;
            int newId = newlyAdded.idEmpresa;

            var doc1 = new documentosempresa();
            var doc2 = new documentosempresa();
            var doc3 = new documentosempresa();
            var doc4 = new documentosempresa();
            var doc5 = new documentosempresa();

            doc1.Empresa_idEmpresa = newId;
            doc1.tipoDocumento = "Contrato Servicio Prevención";

            doc2.Empresa_idEmpresa = newId;
            doc2.tipoDocumento = "IC / ITA";

            doc3.Empresa_idEmpresa = newId;
            doc3.tipoDocumento = "Normas Generales Salud";

            doc4.Empresa_idEmpresa = newId;
            doc4.tipoDocumento = "Seguro Responsabilidad Civil";

            doc5.Empresa_idEmpresa = newId;
            doc5.tipoDocumento = "Escrito Subcontratistas";

            using (proveedoresEntities context = new proveedoresEntities())
            {
                context.documentosempresas.Add(doc1);
                context.documentosempresas.Add(doc2);
                context.documentosempresas.Add(doc3);
                context.documentosempresas.Add(doc4);
                context.documentosempresas.Add(doc5);
                context.SaveChanges();
            }
```

```
        }

        protected void EntityTrabajadores_Inserted(object sender,
EntityDataSourceChangedEventArgs e)
        {
            trabajador newlyAdded = (trabajador)e.Entity;
            int newId = newlyAdded.idTrabajador;
            int padre = newlyAdded.Empresa_idEmpresa;

            var doc1 = new documentostrabajor();
            var doc2 = new documentostrabajor();
            var doc3 = new documentostrabajor();
            var doc4 = new documentostrabajor();


            doc1.Trabajador_idTrabajador = newId;
            doc1.Trabajador_Empresa_idEmpresa = padre;
            doc1.tipoDocumento = "Reconocimiento Médico";

            doc2.Trabajador_idTrabajador = newId;
            doc2.Trabajador_Empresa_idEmpresa = padre;
            doc2.tipoDocumento = "Hoja Entrega de EPIS";

            doc3.Trabajador_idTrabajador = newId;
            doc3.Trabajador_Empresa_idEmpresa = padre;
            doc3.tipoDocumento = "Formación";

            doc4.Trabajador_idTrabajador = newId;
            doc4.Trabajador_Empresa_idEmpresa = padre;
            doc4.tipoDocumento = "Entrega Información";

            using (proveedoresEntities context = new proveedoresEntities())
            {
                context.documentostrabajors.Add(doc1);
                context.documentostrabajors.Add(doc2);
                context.documentostrabajors.Add(doc3);
                context.documentostrabajors.Add(doc4);
                context.SaveChanges();
            }
        }

        #endregion

        #region MODIFICACION GRID PARA USUARIOS NO ADMINISTRADORES

        protected void OcultarBorrar_CommandButtonInitialize(object sender,
ASPxGridViewCommandButtonEventArgs e) //oculta el boton eliminar
        {
            if (!sessionVars._UserLogged.Admin)
            {
                if (e.ButtonType == ColumnCommandButtonType.Delete)
                {
                    e.Visible = false;
                }
            }
        }

        protected void OcultarColumnas_Init(object sender, EventArgs e)
        {
            ASPxGridView grd = (sender as ASPxGridView);

            if (!sessionVars._UserLogged.Admin)
            {
                var oculta = grd.Columns["validado"] as GridViewDataCheckColumn;
                oculta.Visible = false;
            }
        }



        protected void OcultarColumnasTrabajadores_Init(object sender, EventArgs e)
        {
            ASPxGridView grd = (sender as ASPxGridView);
```

```
            if (!sessionVars._UserLogged.Admin)
            {
                var histoculta = grd.Columns["historico"] as GridViewDataCheckColumn;
                histoculta.Visible = false;
            }
        }

        //protected void ComprobarCaducidad_Init(object sender, EventArgs e)
        //{
        //    ASPxGridView grd = (sender as ASPxGridView);
        //    DateTime fecha = Convert.ToDateTime(grd.Columns["fechaCaducidad"]);

        //    if (fecha != null)
        //    {
        //        if (fecha <= DateTime.Now)
        //        {
        //             = "false";

        //        }



        //    }

        //}



#endregion

        protected void PintarCaducidad_HtmlCellPrepared(object sender,
ASPxGridViewTableDataCellEventArgs e)
        {

            if (e.DataColumn.FieldName == "fechaCaducidad")
            {
                DateTime fecha = Convert.ToDateTime(e.GetValue("fechaCaducidad"));
                if (fecha != DateTime.MinValue && fecha <= DateTime.Now)
                {
                    e.Cell.BackColor = System.Drawing.Color.LightGoldenrodYellow;
                }
            }
        }

        protected void Tooltip_HtmlRowPrepared(object sender, ASPxGridViewTableRowEventArgs e)
        {
            if (e.GetValue("usuario") != null)
            {
                e.Row.ToolTip = e.GetValue("usuario").ToString();
            }
        }

        protected void Log_RowInserted(object sender, ASPxDataInsertedEventArgs e)
        {
            Administration.fnc_log_añadir("Insertar Entidad", Administration.fnc_log_dame(e));
        }

        //Método Texto para error en borrado en cascada
        protected void GridErrorEliminar_CustomErrorText(object sender,
ASPxGridViewCustomErrorTextEventArgs e)
        {
            if (e.Exception is System.Data.UpdateException)
            {
                e.ErrorText = "Ha ocurrido un error al eliminar.\nRevise que no existen
registros dependientes.";
            }
        }

        protected void pageControl_ActiveTabChanged(object source, TabControlEventArgs e)
        {

        }
```

```
    }
}
```

11.2.3. Obras.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Root.master" AutoEventWireup="true"
CodeBehind="obras.aspx.cs" Inherits="proveedoresX.obras" %>

<%@ Register Src="~/UserControl/FileManager.ascx" TagPrefix="uc1" TagName="FileManager" %>

<asp:Content ID="Content1" ContentPlaceHolderID="Content" runat="server">
<%--Documentos Obra--%>
<script type="text/javascript">
function OpenPopUpFileManager(s, e) {
var id = s;

callBackFiles.PerformCallback(id);
pop_fileManager.Show();
}
</script>
<uc1:FileManager runat="server" ID="FileManager" />


<dx:ASPxGridView ID="GridObra" runat="server" AutoGenerateColumns="False" Width="100%"
DataSourceID="EntityObras" KeyFieldName="idObra" OnRowInserting="GridObra_RowInserting"
OnRowUpdating="GridObra_RowUpdating" OnCustomErrorText="GridErrorEliminar_CustomErrorText"
SettingsBehavior-ConfirmDelete="true"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize"
OnRowInserted="GridObra_RowInserted">
<SettingsSearchPanel Visible="True"></SettingsSearchPanel>
<SettingsText SearchPanelEditorNullText="Introducir parámetro de búsqueda" />
<Columns>
<dx:GridViewDataTextColumn FieldName="idObra" ReadOnly="True" VisibleIndex="1" Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="nombreObra" VisibleIndex="2" Caption="Nombre Obra">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn Caption="Empresa Contratista Principal" VisibleIndex="2"
FieldName="Empresa_idEmpresa" ReadOnly="false " Visible="true">
<PropertiesComboBox DataSourceID="Empresa" ValueField="idEmpresa"
TextField="nombreEmpresa"></PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataTextColumn FieldName="domicilio" VisibleIndex="3" Caption="Direccion">
</dx:GridViewDataTextColumn>
<dx:GridViewDataDateColumn FieldName="fechaInicio" VisibleIndex="4" Caption="Fecha Inicio">
</dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaFin" VisibleIndex="5" Caption="Fecha Fin">
</dx:GridViewDataDateColumn>
<dx:GridViewDataTextColumn FieldName="responsableGavsa" VisibleIndex="7" Caption="Responsable
GAVSA">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="telefonoGavsa" VisibleIndex="7" Caption="Telefono GAVSA">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="correoGavsa" VisibleIndex="7" Caption="email GAVSA">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="responsableContratista" VisibleIndex="7"
Caption="Responsable Contratista">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="telefonoContratista" VisibleIndex="7" Caption="Telefono
Contratista">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="correoContratista" VisibleIndex="7" Caption="email
Contratista">
</dx:GridViewDataTextColumn>

<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="9" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
```

```
</Styles>
<SettingsBehavior ConfirmDelete="True" />
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsDetail ShowDetailRow="True" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png">
</Image>
<Styles>
<Style ForeColor="White">
</Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png">
</Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png">
</Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png">
</Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png">
</Image>
</CancelButton>
</SettingsCommandButton>
<Templates>
<DetailRow>
<dx:ASPxPageControl runat="server" ID="pageControl" Width="100%" EnableCallBacks="true"
ActiveTabIndex="0">
<TabPages>
<dx:TabPage Text="Documentos Obra" Visible="true">
<ContentCollection>
<dx:ContentControl runat="server">

<dx:ASPxGridView ID="GridDocObra" runat="server" AutoGenerateColumns="False"
DataSourceID="EntityDocObra" Width="100%" EnableTheming="True" KeyFieldName="idDocumentosObra"
OnBeforePerformDataSelect="GridDocObras_BeforePerformDataSelect"
OnRowInserting="GridDocObras_RowInserting" OnRowUpdating="GridDocObra_RowUpdating"
SettingsBehavior-ConfirmDelete="true"
OnCommandButtonInitialize="OcultarBorrar_CommandButtonInitialize" OnInit="OcultarColumnas_Init"
OnHtmlRowPrepared="Tooltip_HtmlRowPrepared" OnRowInserted="GridDocObra_RowInserted">
<ClientSideEvents
RowDblClick="function(s,e) { s.StartEditRow(e.visibleIndex); }"
ContextMenuItemClick="function(s,e) { ContextMenu(s,e); }"
CustomButtonClick="function(s,e){ if(e.buttonID=='showDoc'){
var rowKey = s.GetRowKey(e.visibleIndex);
OpenPopUpFileManager('Obras\\'+rowKey);  } }" />

<Columns>

<dx:GridViewDataTextColumn FieldName="idDocumentosObra" ReadOnly="True" VisibleIndex="1"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="tipoDocumento" VisibleIndex="2" Caption="Tipo de
Documento">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Acta Coordinación de Actividades Empresariales" Value="Acta Coordinación
de Actividades Empresariales" />
<dx:ListEditItem Text="Doc. Nombramiento Coordinador" Value="Doc. Nombramiento Coordinador" />
<dx:ListEditItem Text="Plan de Seguridad" Value="Plan de Seguridad" />
<dx:ListEditItem Text="Recurso Preventivo" Value="Recurso Preventivo" />
<dx:ListEditItem Text="Otros" Value="Otros" />
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataDateColumn FieldName="fechaCarga" ShowInCustomizationForm="True"
VisibleIndex="3" Caption="Fecha Carga">
</dx:GridViewDataDateColumn>
```

```
<dx:GridViewDataDateColumn FieldName="FechaEmision" ShowInCustomizationForm="True"
VisibleIndex="4" Caption="Fecha Emisión">
</dx:GridViewDataDateColumn>
<dx:GridViewDataCheckColumn FieldName="validado" VisibleIndex="5" Caption="Casilla de
Validación">
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarValidado" Cursor="crosshair"
ImageUrl='<%#btnhabilitadoEmp(Convert.ToBoolean(Eval("validado"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>
<dx:GridViewDataCheckColumn FieldName="historico" VisibleIndex="6" Caption="Historico"
Visible="false">
</dx:GridViewDataCheckColumn>
<dx:GridViewDataTextColumn FieldName="descripcion" VisibleIndex="7" Caption="Descripción">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Obra_idObra" ReadOnly="False" VisibleIndex="8"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Obra_Empresa_idEmpresa" ReadOnly="False" VisibleIndex="9"
Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc" VisibleIndex="10">
<CustomButtons>
<dx:GridViewCommandColumnCustomButton ID="showDoc" Image-Url="Content/Images/document-pdf-
text.png">
<Image Url="Content/Images/document-pdf-text.png">
</Image>
</dx:GridViewCommandColumnCustomButton>
</CustomButtons>
</dx:GridViewCommandColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="11" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>

</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsBehavior ConfirmDelete="True" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png">
</Image>
<Styles>
<Style ForeColor="White">
</Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png">
</Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png">
</Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png">
</Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png">
</Image>
</CancelButton>
</SettingsCommandButton>
</dx:ASPxGridView>
</dx:ContentControl>
</ContentCollection>
</dx:TabPage>
<dx:TabPage Text="Subcontratistas" Visible="true">
<ContentCollection>
<dx:ContentControl>
```

```
<dx:ASPxGridView ID="Subcontratista" runat="server" AutoGenerateColumns="False"
DataSourceID="EntityDataSource1" KeyFieldName="id"
OnBeforePerformDataSelect="GridDocObras_BeforePerformDataSelect"
OnRowInserting="Subcontratista_RowInserting" Width="100%">
<Columns>

<dx:GridViewDataTextColumn FieldName="id" ReadOnly="True" VisibleIndex="1"
Visible="false"></dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="idObra" VisibleIndex="2"
Visible="false"></dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn Caption="Contratista" VisibleIndex="2" FieldName="idEmpresa"
ReadOnly="false " Visible="true">
<PropertiesComboBox DataSourceID="Empresa" ValueField="idEmpresa"
TextField="nombreEmpresa"></PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataComboBoxColumn Caption="Contratado por" VisibleIndex="2"
FieldName="idEmpresaPadre" ReadOnly="false " Visible="true">
<PropertiesComboBox DataSourceID="Empresa" ValueField="idEmpresa"
TextField="nombreEmpresa"></PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>

<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="3" ShowNewButtonInHeader="True"
ShowDeleteButton="True"></dx:GridViewCommandColumn>
</Columns>
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsEditing EditFormColumnCount="1" Mode="Inline" />
<SettingsBehavior ConfirmDelete="True" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png">
</Image>
<Styles>
<Style ForeColor="White">
</Style>
</Styles>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png">
</Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png">
</Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png">
</Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png">
</Image>
</CancelButton>
</SettingsCommandButton>
</dx:ASPxGridView>


</dx:ContentControl>
</ContentCollection>


</dx:TabPage>


</TabPages>
</dx:ASPxPageControl>
<%--Documentos Obra--%>
</DetailRow>
<%--Documentos Obra--%>
</Templates>
```

```
</dx:ASPxGridView>


<asp:EntityDataSource ID="EntityObras" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="obras">
<WhereParameters>
</WhereParameters>
</asp:EntityDataSource>

<asp:EntityDataSource ID="EntityDocObra" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="documentosobras" Where="it.[Obra_idObra] = @paridObra">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridObra" SessionField="idObra" />
</WhereParameters>
</asp:EntityDataSource>

<asp:EntityDataSource runat="server" ID="Empresa" DefaultContainerName="proveedoresEntities"
ConnectionString="name=proveedoresEntities" EnableFlattening="False" EntitySetName="empresas"
Select="it.[idEmpresa], it.[CIF], it.[nombreEmpresa]"></asp:EntityDataSource>

<asp:EntityDataSource ID="EntityDataSource1" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="subcontratas" Where="it.[idObra] = @paridObra">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridObra" SessionField="idObra" />
</WhereParameters>
</asp:EntityDataSource>
</asp:Content>
```

11.2.4. Obras.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using DevExpress.Web;
using DevExpress.Web.Data;
using System.Data;
using System.Data.Entity.Core;
using proveedoresX.Clases;

namespace proveedoresX
{
    public partial class obras : System.Web.UI.Page
    {
        protected void Page_PreInit(object sender, EventArgs e)
        {
            sessionVars.Autologon();
        }

        protected void Page_Init(object sender, EventArgs e)
        {

            if (!sessionVars._UserLogged.Admin)
            {

                EntityObras.Where = "it.[Empresa_idEmpresa] = " +
sessionVars._UserLogged.Empresa; //impide edicion a usuario no administrador

                cls_proveedores.DeshabilitarEdicionGrid(GridObra);
            }

            //var lal = new ASPxGridView();
            //Form.Controls.Add(lal);
        }
```

```
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        #region METODOS OBRA
        //Métodos Grid OBRA


        protected void GridObra_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            //e.NewValues["Empresa_idEmpresa"] = (sender as
ASPxGridView).GetMasterRowKeyValue();
        }

        protected void GridObra_RowInserted(object sender, ASPxDataInsertedEventArgs e)
        {
            Administration.fnc_log_añadir("Insertar Obra", Administration.fnc_log_dame(e));
        }

        protected void GridObra_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["Empresa_idEmpresa"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "Empresa_idEmpresa" });
            e.OldValues["idObra"] = ((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new
string[] { "idObra" });

            e.NewValues["Empresa_idEmpresa"] = e.OldValues["Empresa_idEmpresa"];
            e.NewValues["idObra"] = e.OldValues["idObra"];

            Administration.fnc_log_añadir("Insertar Documento Obra",
Administration.fnc_log_dame(e));
        }

        #endregion

        #region DOC OBRAS
        //Métodos Grid DOCUMENTOS OBRAS
        protected void GridDocObras_BeforePerformDataSelect(object sender, EventArgs e)
        {
            Session["idObra"] = (sender as ASPxGridView).GetMasterRowKeyValue();
        }

        protected void GridDocObra_RowInserted(object sender, ASPxDataInsertedEventArgs e)
        {
            Administration.fnc_log_añadir("Insertar Documento Obra",
Administration.fnc_log_dame(e));
        }

        protected void GridDocObras_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Obra_idObra"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.NewValues["Obra_Empresa_idEmpresa"] = (sender as
ASPxGridView).GetMasterRowFieldValues("Empresa_idEmpresa");
            e.NewValues["fechaCarga"] = DateTime.Now;
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;
        }

        protected void GridDocObra_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["Obra_idObra"] = (sender as ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["Obra_Empresa_idEmpresa"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "Obra_Empresa_idEmpresa"
});
            e.OldValues["idDocumentosObra"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idDocumentosObra" });


            e.NewValues["Obra_idObra"] = e.OldValues["Obra_idObra"];
            e.NewValues["Obra_Empresa_idEmpresa"] = e.OldValues["Obra_Empresa_idEmpresa"];
            e.NewValues["idDocumentosObra"] = e.OldValues["idDocumentosTrabajo"];
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;
```

```
            Administration.fnc_log_añadir("Actualizar Doc. Obra",
Administration.fnc_log_dame(e));
        }

        #endregion

        #region CONTROL SEMAFOROS Y EDICION VALIDO

        private void RefrescaGridTra(object sender)
        {
            ASPxGridView grid = (sender as ASPxGridView);
            int padre = Convert.ToInt32(grid.GetMasterRowKeyValue());
            // int empresaPadre = Convert.ToInt32(grid.); revisar

            cls_proveedores.trabajadorValido(padre);
            //cls_proveedores.empresaValido(empresaPadre); // añadido para comprobar tambien sus
trabajadores

            grid.JSProperties["cpUpdate"] = "1";

        }

        private void RefrescaGrid(object sender)
        {
            ASPxGridView grid = (sender as ASPxGridView);
            int padre = Convert.ToInt32(grid.GetMasterRowKeyValue());

            cls_proveedores.empresaValido(padre);

            grid.JSProperties["cpUpdate"] = "1";
        }


        protected string btnVerdeEmp(bool esVerde)
        {
            return (esVerde) ? "~/Content/Images/green.png" : "~/Content/Images/red.png";
        }


        protected string btnhabilitadoEmp(bool habilitado)
        {
            return (habilitado) ? "~/Content/Images/accept.png" : "~/Content/Images/delete.png";

        }


        protected void img_valido_LoadEmp(object sender, EventArgs e)
        {

            ASPxImage image = (sender as ASPxImage);
            GridViewDataItemTemplateContainer container = (image.NamingContainer as
GridViewDataItemTemplateContainer);

            image.ClientSideEvents.Click = String.Format(@"function (s, e) {{
GridDocEmpresa.PerformCallback(""{1}|"" + {0}); }}", container.VisibleIndex, image.ID);

        }

        protected void img_valido_LoadTraEmp(object sender, EventArgs e)
        {

            ASPxImage image = (sender as ASPxImage);
            GridViewDataItemTemplateContainer container = (image.NamingContainer as
GridViewDataItemTemplateContainer);

            image.ClientSideEvents.Click = String.Format(@"function (s, e) {{
DocTrabajadores.PerformCallback(""{1}|"" + {0}); }}", container.VisibleIndex, image.ID);

        }

        #endregion

        #region MODIFICACION GRID PARA USUARIOS NO ADMINISTRADORES
```

```
        protected void OcultarBorrar_CommandButtonInitialize(object sender,
ASPxGridViewCommandButtonEventArgs e) //oculta el boton eliminar
        {
            if (!sessionVars._UserLogged.Admin)
            {
                if (e.ButtonType == ColumnCommandButtonType.Delete)
                {
                    e.Visible = false;
                }
            }
        }

        protected void OcultarColumnas_Init(object sender, EventArgs e)
        {
            ASPxGridView grd = (sender as ASPxGridView);

            if (!sessionVars._UserLogged.Admin)
            {
                var oculta = grd.Columns["validado"] as GridViewDataCheckColumn;
                oculta.Visible = false;
            }
        }



        protected void OcultarColumnasTrabajadores_Init(object sender, EventArgs e)
        {
            ASPxGridView grd = (sender as ASPxGridView);

            if (!sessionVars._UserLogged.Admin)
            {
                var histoculta = grd.Columns["historico"] as GridViewDataCheckColumn;
                histoculta.Visible = false;
            }
        }

        //protected void ComprobarCaducidad_Init(object sender, EventArgs e)
        //{
        //    ASPxGridView grd = (sender as ASPxGridView);
        //    DateTime fecha = Convert.ToDateTime(grd.Columns["fechaCaducidad"]);

        //    if (fecha != null)
        //    {
        //        if (fecha <= DateTime.Now)
        //        {
        //             = "false";

        //        }



        //    }

        //}

        #endregion

        protected void PintarCaducidad_HtmlCellPrepared(object sender,
ASPxGridViewTableDataCellEventArgs e)
        {

            if (e.DataColumn.FieldName == "fechaCaducidad")
            {

                DateTime fecha = Convert.ToDateTime(e.GetValue("fechaCaducidad"));
                if (fecha != DateTime.MinValue && fecha <= DateTime.Now)
                {

                    e.Cell.BackColor = System.Drawing.Color.LightGoldenrodYellow;


                }

            }
```

```
        }


        //Método Texto para error en borrado en cascada
        protected void GridErrorEliminar_CustomErrorText(object sender,
ASPxGridViewCustomErrorTextEventArgs e)
        {
            if (e.Exception is System.Data.UpdateException)
            {
                e.ErrorText = "Ha ocurrido un error al eliminar.\nRevise que no existen
registros dependientes.";

            }
        }

        protected void Subcontratista_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["idObra"] = (sender as ASPxGridView).GetMasterRowKeyValue();
        }

        protected void Tooltip_HtmlRowPrepared(object sender, ASPxGridViewTableRowEventArgs e)
        {
            if (e.GetValue("usuario") != null)
            {
                e.Row.ToolTip = e.GetValue("usuario").ToString();
            }
        }



    }
}
```

11.2.5. Trabajadores.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Main.master" AutoEventWireup="true"
CodeBehind="Trabajadores.aspx.cs" Inherits="proveedoresX.Trabajadores" %>

<%@ Register Src="~/UserControl/FileManager.ascx" TagPrefix="uc1" TagName="FileManager" %>


<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">


<%-- Componente de gestión de FILE MANAGER--%>
<script type="text/javascript">
function OpenPopUpFileManager(s, e) {
var id = s;

callBackFiles.PerformCallback(id);
pop_fileManager.Show();
}
</script>

<uc1:FileManager runat="server" ID="FileManager" />


<dx:ASPxGridView ID="InformeTrabajadores" ClientInstanceName="InformeTrabajadores"
runat="server" AutoGenerateColumns="False" DataSourceID="EntityInformeTrabajadores"
KeyFieldName="idTrabajador" Theme="MetropolisBlue" EnableTheming="True" Width="100%"
OnRowUpdating="InformeTrabajadores_RowUpdating"
OnHtmlDataCellPrepared="InformeTrabajadores_HtmlDataCellPrepared"
OnHtmlRowPrepared="InformeTrabajadores_HtmlRowPrepared">
<Columns>
<dx:GridViewDataTextColumn FieldName="idTrabajador" ReadOnly="True" Visible="false">
</dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="nombre" Caption="Nombre"></dx:GridViewDataTextColumn>
```

```
<dx:GridViewDataCheckColumn FieldName="esVerde" Caption="Estado" EditFormSettings-
Visible="False" ReadOnly="true" Width="30">
<EditFormSettings Visible="False"></EditFormSettings>
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarVerde" Cursor="crosshair" ImageUrl='<%#
btnVerdeT(Convert.ToBoolean(Eval("esVerde"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>
<dx:GridViewDataTextColumn FieldName="DNI" Caption="DNI"></dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn Caption="Empresa" FieldName="Empresa_idEmpresa" ReadOnly="false
">
<PropertiesComboBox DataSourceID="Empresa" ValueField="idEmpresa"
TextField="nombreEmpresa"></PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataTextColumn FieldName="puestoTrabajo" Caption="Puesto de
Trabajo"></dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="informacion"
Caption="Información"></dx:GridViewDataTextColumn>
<%--<dx:GridViewDataTextColumn FieldName="Empresa_idEmpresa" VisibleIndex="6" ReadOnly="False"
Visible="false">
</dx:GridViewDataTextColumn>--%>
<dx:GridViewDataTextColumn HeaderStyle-BackColor="#376EB8" HeaderStyle-ForeColor="White"
FieldName="esVerde" Caption="verde" Visible="false">
<HeaderStyle BackColor="#376EB8" ForeColor="White"></HeaderStyle>
</dx:GridViewDataTextColumn>
<dx:GridViewDataCheckColumn FieldName="historico" Caption="Historico" Visible="true" Width="30"
>
</dx:GridViewDataCheckColumn>
<dx:GridViewCommandColumn HeaderStyle-BackColor="#376EB8" HeaderStyle-ForeColor="White"
ShowEditButton="True" ShowDeleteButton="False" Caption=" " Width="55"
ShowNewButtonInHeader="True">
<HeaderStyle BackColor="#376EB8" ForeColor="White"></HeaderStyle>
</dx:GridViewCommandColumn>

</Columns>
<SettingsPager Mode="ShowPager" PageSize="30" AllButton-Visible="true">
<AllButton Visible="True"></AllButton>
</SettingsPager>
<SettingsEditing Mode="Inline"></SettingsEditing>

<SettingsSearchPanel Visible="True"></SettingsSearchPanel>
<SettingsText SearchPanelEditorNullText="Introducir parámetro de búsqueda" />


<Styles>
<Row BackColor="WhiteSmoke"></Row>
<Header BackColor="#376EB8" ForeColor="White"></Header>
</Styles>
<SettingsBehavior ConfirmDelete="True"></SettingsBehavior>
<Settings ShowFilterBar="Visible" ShowFilterRow="true" ShowFilterRowMenu="true"
ShowFilterRowMenuLikeItem="true" ShowHeaderFilterButton="true" />
<SettingsDetail ShowDetailRow="True" />
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-ForeColor="White">
<Image Url="Content/Images/add.png"></Image>
</NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" ">
<Image Url="Content/Images/pencil.png"></Image>
</EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" ">
<Image Url="Content/Images/bin_closed.png"></Image>
</DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" ">
<Image Url="Content/Images/disk.png"></Image>
</UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" ">
<Image Url="Content/Images/cancel.png"></Image>
</CancelButton>
</SettingsCommandButton>

<Templates>
<DetailRow>
```

```
<dx:ASPxGridView ID="DocTrabajadores" ClientInstanceName="DocTrabajadores" runat="server"
DataSourceID="EntityDocTrabajadores"
OnBeforePerformDataSelect="DocTrabajadores_BeforePerformDataSelect" Theme="MetropolisBlue"
AutoGenerateColumns="False" OnRowInserting="DocTrabajadores_RowInserting"
OnRowUpdating="DocTrabajadores_RowUpdating" OnRowUpdated="DocTrabajadores_RowUpdated"
OnRowInserted="DocTrabajadores_RowInserted" KeyFieldName="idDocumentosTrabajor" Width="100%"
OnCustomCallback="DocTrabajadores_CustomCallback" OnHtmlRowPrepared="Tooltip_HtmlRowPrepared">
<ClientSideEvents
EndCallback="function(s,e) { if (s.cpUpdate) { if (s.cpUpdate == '1') {
InformeTrabajadores.Refresh();  delete s.cpUpdate; } } }"
RowDblClick="function(s,e) { s.StartEditRow(e.visibleIndex); }"
ContextMenuItemClick="function(s,e) { ContextMenu(s,e); }"
CustomButtonClick="function(s,e){ if(e.buttonID=='showDoc'){
var rowKey = s.GetRowKey(e.visibleIndex);
OpenPopUpFileManager('Trabajadores\\'+rowKey);  } }" />
<Columns>
<dx:GridViewDataTextColumn FieldName="idDocumentosTrabajor" ReadOnly="True" VisibleIndex="0"
Visible="false"></dx:GridViewDataTextColumn>
<dx:GridViewDataComboBoxColumn FieldName="tipoDocumento" VisibleIndex="1" Caption="Tipo de
Documento">
<PropertiesComboBox>
<Items>
<dx:ListEditItem Text="Formación" Value="Formación" />
<dx:ListEditItem Text="Reconocimiento Médico" Value="Reconocimiento Médico" />
<dx:ListEditItem Text="Entrega Información" Value="Entrega Información" />
<dx:ListEditItem Text="Nombramiento Recurso Preventivo" Value="Nombramiento Recurso Preventivo"
/>
<dx:ListEditItem Text="Nombramiento Trabajador autorizado/cualificado" Value="Nombramiento
Trabajador autorizado/cualificado" />
<dx:ListEditItem Text="Hoja Entrega de EPIS" Value="Hoja Entrega de EPIS" />
<dx:ListEditItem Text="DNI" Value="DNI"/>
<dx:ListEditItem Text="Otros" Value="Otros"/>
</Items>
</PropertiesComboBox>
</dx:GridViewDataComboBoxColumn>
<dx:GridViewDataDateColumn FieldName="fechaEmision" VisibleIndex="2" Caption="Fecha Emisión /
Carga"></dx:GridViewDataDateColumn>
<dx:GridViewDataDateColumn FieldName="fechaCaducidad" VisibleIndex="3" Caption="Fecha de
Caducidad"></dx:GridViewDataDateColumn>
<dx:GridViewDataCheckColumn FieldName="validado" VisibleIndex="4" Caption="Casilla de
Validación">
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarValidado" Cursor="crosshair" OnLoad="img_valido_Load"
ImageUrl='<%# btnhabilitado(Convert.ToBoolean(Eval("validado"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>
<dx:GridViewDataCheckColumn FieldName="historico" VisibleIndex="5" Caption="Histórico"
Visible="false">
<DataItemTemplate>
<dx:ASPxImage runat="server" ID="CambiarHistorico" Cursor="crosshair" OnLoad="img_valido_Load"
ImageUrl='<%# btnhabilitado(Convert.ToBoolean(Eval("historico"))) %>'>
</dx:ASPxImage>
</DataItemTemplate>
</dx:GridViewDataCheckColumn>
<dx:GridViewDataTextColumn FieldName="descripcion" VisibleIndex="6"
Caption="Descripción"></dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Trabajador_idTrabajador" ReadOnly="True" VisibleIndex="7"
Visible="false"></dx:GridViewDataTextColumn>
<dx:GridViewDataTextColumn FieldName="Trabajador_Empresa_idEmpresa" ReadOnly="True"
VisibleIndex="8" Visible="false"></dx:GridViewDataTextColumn>
<dx:GridViewCommandColumn Visible="true" ButtonType="Image" Caption="Doc" VisibleIndex="9">
<CustomButtons>
<dx:GridViewCommandColumnCustomButton ID="showDoc" Image-Url="Content/Images/document-pdf-
text.png">
</dx:GridViewCommandColumnCustomButton>
</CustomButtons>
</dx:GridViewCommandColumn>
<dx:GridViewCommandColumn ShowEditButton="True" VisibleIndex="10" ShowNewButtonInHeader="True"
ShowDeleteButton="True">
</dx:GridViewCommandColumn>
</Columns>
<SettingsEditing Mode="Inline"></SettingsEditing>
<SettingsBehavior ConfirmDelete="True" />
```

```
<SettingsLoadingPanel Mode="Disabled" />
<Styles>
<Header BackColor="#6A97D2" ForeColor="White"></Header>
</Styles>
<SettingsCommandButton>
<NewButton Image-Url="Content/Images/add.png" Text="Nuevo" Styles-Style-
ForeColor="White"></NewButton>
<EditButton Image-Url="Content/Images/pencil.png" Text=" "></EditButton>
<DeleteButton Image-Url="Content/Images/bin_closed.png" Text=" "></DeleteButton>
<UpdateButton Image-Url="Content/Images/disk.png" Text=" "></UpdateButton>
<CancelButton Image-Url="Content/Images/cancel.png" Text=" "></CancelButton>
</SettingsCommandButton>
</dx:ASPxGridView>
</DetailRow>
</Templates>
</dx:ASPxGridView>

<asp:EntityDataSource runat="server" ID="EntityInformeTrabajadores"
DefaultContainerName="proveedoresEntities" ConnectionString="name=proveedoresEntities"
EnableFlattening="False" EnableDelete="True" EnableInsert="True" EnableUpdate="True"
EntitySetName="trabajadors"></asp:EntityDataSource>

<asp:EntityDataSource ID="EntityDocTrabajadores" runat="server"
ConnectionString="name=proveedoresEntities" DefaultContainerName="proveedoresEntities"
EnableDelete="True" EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
EntitySetName="documentostrabajors" Where="it.[Trabajador_idTrabajador] = @paridTrabajadores">
<WhereParameters>
<asp:SessionParameter DbType="Int32" Name="paridTrabajadores" SessionField="idTrabajadores" />
</WhereParameters>
</asp:EntityDataSource>

<asp:EntityDataSource runat="server" ID="Empresa" DefaultContainerName="proveedoresEntities"
ConnectionString="name=proveedoresEntities" EnableFlattening="False" EntitySetName="empresas"
Select="it.[idEmpresa], it.[CIF], it.[nombreEmpresa]"></asp:EntityDataSource>

</asp:Content>
```

11.2.6. Trabajadores.aspx.cs

```
using DevExpress.Web;
using DevExpress.Web.Data;
using proveedoresX.Clases;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace proveedoresX
{
    public partial class Trabajadores : System.Web.UI.Page
    {
        protected void Page_PreInit(object sender, EventArgs e)
        {
            sessionVars.Autologon();
        }

        protected void Page_Init(object sender, EventArgs e)
        {

            if (!sessionVars._UserLogged.Admin)
            {

                EntityInformeTrabajadores.Where = "it.[Empresa_idEmpresa] = " +
sessionVars._UserLogged.Empresa; //impide edicion a usuario no administrador

                cls_proveedores.DeshabilitarEdicionGrid(InformeTrabajadores);
            }

            //var lal = new ASPxGridView();
            //Form.Controls.Add(lal);
        }
```

```
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void InformeTrabajadores_RowUpdating(object sender, ASPxDataUpdatingEventArgs
e)
        {
            e.OldValues["Empresa_idEmpresa"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "Empresa_idEmpresa" });
            e.OldValues["idTrabajador"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idTrabajador" });

            e.NewValues["Empresa_idEmpresa"] = e.OldValues["Empresa_idEmpresa"];
            e.NewValues["idTrabajador"] = e.OldValues["idTrabajador"];
            Administration.fnc_log_añadir("Actualizar Trabajador",
Administration.fnc_log_dame(e));
        }

        protected void DocTrabajadores_BeforePerformDataSelect(object sender, EventArgs e)
        {

            Session["idTrabajadores"] = (sender as ASPxGridView).GetMasterRowKeyValue();

        }

        protected void DocTrabajadores_RowInserting(object sender, ASPxDataInsertingEventArgs e)
        {
            e.NewValues["Trabajador_idTrabajador"] = (sender as
ASPxGridView).GetMasterRowKeyValue();
            e.NewValues["Trabajador_Empresa_idEmpresa"] = (sender as
ASPxGridView).GetMasterRowFieldValues("Empresa_idEmpresa");
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;


        }

        protected void DocTrabajadores_RowUpdating(object sender, ASPxDataUpdatingEventArgs e)
        {
            e.OldValues["Trabajador_idTrabajador"] = (sender as
ASPxGridView).GetMasterRowKeyValue();
            e.OldValues["Trabajador_Empresa_idEmpresa"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] {
"Trabajador_Empresa_idEmpresa" });
            e.OldValues["idDocumentosTrabajor"] =
((ASPxGridView)sender).GetRowValuesByKeyValue(e.Keys[0], new string[] { "idDocumentosTrabajor"
});

            e.NewValues["Trabajador_idTrabajador"] = e.OldValues["Trabajador_idTrabajador"];
            e.NewValues["Trabajador_Empresa_idEmpresa"] =
e.OldValues["Trabajador_Empresa_idEmpresa"];
            e.NewValues["idDocumentosTrabajor"] = e.OldValues["idDocumentosTrabajor"];
            e.NewValues["usuario"] = sessionVars._UserLogged.Usuario;

            Administration.fnc_log_añadir("Actualizar Doc. Trabajador",
Administration.fnc_log_dame(e));

        }

        protected void InformeTrabajadores_HtmlDataCellPrepared(object sender,
ASPxGridViewTableDataCellEventArgs e)
        {
            if (e.DataColumn.FieldName == "nombre")
            {

                if (e.GetValue("esVerde") != null && e.GetValue("esVerde").Equals(false))
                {
                    e.Cell.BackColor = System.Drawing.Color.Tomato;
                    e.Cell.ForeColor = System.Drawing.Color.White;

                }
```

```
            }
        }

        protected void InformeTrabajadores_HtmlRowPrepared(object sender,
ASPxGridViewTableRowEventArgs e)//pintar row trabajadores historicos
        {
            if (e.GetValue("historico") != null && e.GetValue("historico").Equals(true))
            {
                e.Row.BackColor = System.Drawing.Color.LightSlateGray;
                e.Row.ForeColor = System.Drawing.Color.White;


            }
        }

        protected void DocTrabajadores_RowUpdated(object sender, ASPxDataUpdatedEventArgs e)
        {
            RefrescaGrid(sender);
            Administration.fnc_log_añadir("Actualizar Trabajador",
Administration.fnc_log_dame(e));
        }

        protected void DocTrabajadores_RowInserted(object sender, ASPxDataInsertedEventArgs e)
        {
            RefrescaGrid(sender);
            Administration.fnc_log_añadir("Insertar Trabajador",
Administration.fnc_log_dame(e));
        }


        protected void RefrescaGrid(object sender)
        {
            ASPxGridView grid = (sender as ASPxGridView);
            int padre = Convert.ToInt32(grid.GetMasterRowKeyValue());

            cls_proveedores.trabajadorValido(padre);

            grid.JSProperties["cpUpdate"] = "1";
        }

        protected void img_valido_Load(object sender, EventArgs e)
        {

            ASPxImage image = (sender as ASPxImage);
            GridViewDataItemTemplateContainer container = (image.NamingContainer as
GridViewDataItemTemplateContainer);

            image.ClientSideEvents.Click = String.Format(@"function (s, e) {{
DocTrabajadores.PerformCallback(""{1}|"" + {0}); }}", container.VisibleIndex, image.ID);

        }

        protected string btnhabilitado(bool habilitado)
        {
            return (habilitado) ? "~/Content/Images/accept.png" : "~/Content/Images/delete.png";
        }



        protected string btnVerdeT(bool esVerde)
        {
            return (esVerde) ? "~/Content/Images/green.png" : "~/Content/Images/red.png";
        }

        protected void DocTrabajadores_CustomCallback(object sender,
ASPxGridViewCustomCallbackEventArgs e)
        {
            ASPxGridView grid = (sender as ASPxGridView);

            if (!string.IsNullOrEmpty(e.Parameters))
            {
                var datos = e.Parameters.Split('|');

                if (datos[0] == "CambiarValidado")
```

```
                {
                    int visibleindex = Convert.ToInt32(datos[1]);
                    bool esValido = Convert.ToBoolean(grid.GetRowValues(visibleindex,
"validado"));
                    int idDocumentosTrabajor = Convert.ToInt32(grid.GetRowValues(visibleindex,
"idDocumentosTrabajor"));

                    cls_proveedores.CambiarValidezDocumento(idDocumentosTrabajor, !esValido);

                    grid.JSProperties["cpUpdate"] = "1";
                }
                else if (datos[0] == "CambiarHistorico")
                {
                    int visibleindex = Convert.ToInt32(datos[1]);
                    bool esValido = Convert.ToBoolean(grid.GetRowValues(visibleindex,
"historico"));
                    int idDocumentosTrabajor = Convert.ToInt32(grid.GetRowValues(visibleindex,
"idDocumentosTrabajor"));

                    cls_proveedores.CambiarHistoricoDocumento(idDocumentosTrabajor, !esValido);

                    //grid.JSProperties["cpUpdate"] = "1";
                    grid.DataBind();
                }

            }
        }


        protected void Tooltip_HtmlRowPrepared(object sender, ASPxGridViewTableRowEventArgs e)
        {
            if (e.GetValue("usuario") != null)
            {
                e.Row.ToolTip = e.GetValue("usuario").ToString();
            }
        }


    }

}
```

11.2.7. WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="proveedoresX.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
,.-
</div>
</form>
</body>
</html>
```

11.2.8. WebForms.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Data.Objects;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using proveedoresX.Clases;

namespace proveedoresX
{
    public partial class WebForm1 : System.Web.UI.Page
    {

        protected void Page_Load(object sender, EventArgs e)
        {
            notificaciones.ComprobarCaducidadEmpresa();
            notificaciones.ComprobarCaducidadEspecial();
            notificaciones.ComprobarCaducidadTrabajador();
            notificaciones.ComprobarCaducidadVehiculo();
        }


    }


}
```

11.2.9. Notificaciones.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using proveedoresX;
using proveedoresX.permisatorService;
using proveedoresX.Clases;
using proveedoresX.notificacionService;
using proveedoresX.usuariosService;

namespace proveedoresX
{
    public class notificaciones
    {

        public static void ComprobarCaducidadEmpresa()
        {
            using (proveedoresEntities context = new proveedoresEntities())
            {
                try
                {

                    DateTime caducidad = DateTime.Now.AddMonths(+1);
                    DateTime caducidad2 = caducidad.AddDays(-1);

                    List<int> lDoc = (from contact in context.documentosempresas
                                      where contact.fechaCaducidad < caducidad &&
contact.fechaCaducidad > caducidad2
                                      select contact.idDocumentosEmpresa).ToList();

                    foreach (int doc in lDoc)
                    {
                        string correo = (from contact in context.empresas
                                         where contact.idEmpresa == (from contact2 in
context.documentosempresas
                                                                     where
contact2.idDocumentosEmpresa == doc
                                                                     select
contact2.Empresa_idEmpresa).FirstOrDefault()
                                         select contact.correo).FirstOrDefault();

                        EnviarEmail(correo,"Empresa", doc);
                    }

                }

                catch (Exception e) { Console.CapsLock.ToString(); }
            }
```

```
        }

        public static void ComprobarCaducidadEspecial()
        {
            using (proveedoresEntities context = new proveedoresEntities())
            {
                try
                {
                    DateTime caducidad = DateTime.Now.AddMonths(+1);
                    DateTime caducidad2 = caducidad.AddDays(-1);

                    List<int> lDoc = (from contact in context.documentostrabajoes
                                      where contact.fechaCaducidad < caducidad &&
contact.fechaCaducidad > caducidad2
                                      select contact.idDocumentosTrabajo).ToList();

                    foreach (int doc in lDoc)
                    {
                        string correo = (from contact in context.empresas
                                         where contact.idEmpresa == (from contact2 in
context.documentostrabajoes
                                                                     where
contact2.idDocumentosTrabajo == doc
                                                                     select
contact2.TrabajoEspecial_Empresa_idEmpresa).FirstOrDefault()
                                         select contact.correo).FirstOrDefault();

                        EnviarEmail(correo,"Trabajo especial", doc);
                    }



                }

                catch (Exception e) { Console.CapsLock.ToString(); }
            }


        }

        public static void ComprobarCaducidadTrabajador()
        {
            using (proveedoresEntities context = new proveedoresEntities())
            {
                try
                {
                    DateTime caducidad = DateTime.Now.AddMonths(+1);
                    DateTime caducidad2 = caducidad.AddDays(-1);

                    List<int> lDoc = (from contact in context.documentostrabajors
                                      where contact.fechaCaducidad < caducidad &&
contact.fechaCaducidad > caducidad2
                                      select contact.idDocumentosTrabajor).ToList();

                    foreach (int doc in lDoc)
                    {
                        string correo = (from contact in context.empresas
                                         where contact.idEmpresa == (from contact2 in
context.documentostrabajors
                                                                     where
contact2.idDocumentosTrabajor == doc
                                                                     select
contact2.Trabajador_Empresa_idEmpresa).FirstOrDefault()
                                         select contact.correo).FirstOrDefault();

                        EnviarEmail(correo,"Trabajador", doc);
                    }



                }
```

```
                catch (Exception e) { Console.CapsLock.ToString(); }
            }


        }

        public static void ComprobarCaducidadVehiculo()
        {
            using (proveedoresEntities context = new proveedoresEntities())
            {
                try
                {
                    DateTime caducidad = DateTime.Now.AddMonths(+1);
                    DateTime caducidad2 = caducidad.AddDays(-1);

                    List<int> lDoc = (from contact in context.documentosequipoes
                                      where contact.fechaCaducidad < caducidad &&
contact.fechaCaducidad > caducidad2
                                      select contact.idDocumentosEquipo).ToList();

                    foreach (int doc in lDoc)
                    {
                        string correo = (from contact in context.empresas
                                         where contact.idEmpresa == (from contact2 in
context.documentosequipoes
                                                                     where
contact2.idDocumentosEquipo == doc
                                                                     select
contact2.EquipoVehiculo_Empresa_idEmpresa).FirstOrDefault()
                                         select contact.correo).FirstOrDefault();

                        EnviarEmail(correo,"Equipo/Vehiculo", doc);
                    }



                }

                catch (Exception e) { Console.CapsLock.ToString(); }
            }


        }

        public static void EnviarEmail(string mail, string tipo, int document)// llamo al metodo
de roberto ¿? y envio el mail al string que recibo y a los emails de los usuarios de la
aplicación subscritos
        {

            string mensaje = ComponerMensaje(document, tipo);
            var usuarios = permisator.dame_usuarios("Estado Proveedores");

            //SendEmail(mail,mensaje); // Primero se manda el mail a la EMPRESA. güemes tendrá
que decidir si se lo mandamos ya a la empresa o no. DE MOMENTO NO

            var listaAdmins = usuarios.Where(a => a.Rol == "Administrador").ToList();

            var listaDAP = permisator.dame_usuariosLDAP();

            foreach (permisatorService.Permiso userAdmin in listaAdmins) //Aqui se obtiene el
correo del usuario Administrador y se envia
            {
                var user = listaDAP.Where(a => a.Codigo ==
userAdmin.Usuario.ToLower()).FirstOrDefault();
                if (user != null)
                {
                    string email = user.Email;
                    Console.CapsLock.ToString();
                    SendEmail(email, mensaje);
                }

            }

        }
```

```
        public static string ComponerMensaje(int iddocumento, string tipo)
        {
            using (proveedoresEntities context = new proveedoresEntities())
            {
                string empresa ="";
                string entidad = "";
                string documento="";

                if (tipo == "Empresa")
                {
                    empresa = (from contact in context.empresas
                               where contact.idEmpresa == (from contact2 in
context.documentosempresas
                                                           where contact2.idDocumentosEmpresa ==
iddocumento
                                                           select
contact2.Empresa_idEmpresa).FirstOrDefault()
                               select contact.nombreEmpresa).FirstOrDefault();

                    entidad = empresa;

                    documento = (from contact in context.documentosempresas
                                 where contact.idDocumentosEmpresa == iddocumento
                                 select contact.tipoDocumento).FirstOrDefault();
                }

                if (tipo == "Trabajo especial")
                {
                    empresa = (from contact in context.empresas
                               where contact.idEmpresa == (from contact2 in
context.documentostrabajoes
                                                           where contact2.idDocumentosTrabajo ==
iddocumento
                                                           select
contact2.TrabajoEspecial_Empresa_idEmpresa).FirstOrDefault()
                               select contact.nombreEmpresa).FirstOrDefault();

                    entidad = (from contact in context.trabajoespecials
                                   where contact.idTrabajoEspecial == (from contact2 in
context.documentostrabajoes
                                                                       where
contact2.idDocumentosTrabajo == iddocumento
                                                                       select
contact2.TrabajoEspecial_idTrabajoEspecial).FirstOrDefault()
                                   select contact.tipoTrabajo).FirstOrDefault();

                    documento = (from contact in context.documentostrabajoes
                                 where contact.idDocumentosTrabajo == iddocumento
                                 select contact.DocumentoTrabajo).FirstOrDefault();

                }

                 if (tipo == "Trabajador")
                {
                    empresa = (from contact in context.empresas
                               where contact.idEmpresa == (from contact2 in
context.documentostrabajors
                                                           where contact2.idDocumentosTrabajor
== iddocumento
                                                           select
contact2.Trabajador_Empresa_idEmpresa).FirstOrDefault()
                               select contact.nombreEmpresa).FirstOrDefault();

                    entidad = (from contact in context.trabajadors
                                   where contact.idTrabajador == (from contact2 in
context.documentostrabajors
                                                                  where
contact2.idDocumentosTrabajor == iddocumento
                                                                  select
contact2.Trabajador_idTrabajador).FirstOrDefault()
                                   select contact.nombre).FirstOrDefault();

                    documento = (from contact in context.documentostrabajors
```

```
                                    where contact.idDocumentosTrabajor == iddocumento
                                    select contact.tipoDocumento).FirstOrDefault();

                }

                 if (tipo == "Equipo/Vehiculo")
                 {
                     empresa = (from contact in context.empresas
                                where contact.idEmpresa == (from contact2 in
context.documentosequipoes
                                                            where contact2.idDocumentosEquipo ==
iddocumento
                                                            select
contact2.EquipoVehiculo_Empresa_idEmpresa).FirstOrDefault()
                                select contact.nombreEmpresa).FirstOrDefault();

                     entidad = (from contact in context.equipovehiculoes
                                where contact.idEquipoVehiculo == (from contact2 in
context.documentosequipoes
                                                                   where contact2.idDocumentosEquipo
== iddocumento
                                                                       select
contact2.EquipoVehiculo_idEquipoVehiculo).FirstOrDefault()
                                select contact.descripcion).FirstOrDefault();

                     documento = (from contact in context.documentosequipoes
                                  where contact.idDocumentosEquipo == iddocumento
                                  select contact.tipoDocumento).FirstOrDefault();

                 }

                string mensaje = "El documento "+ documento + ", perteneciente a la empresa " +
empresa + " y a la entidad " + entidad + " caducará en un mes, por favor revise la
documentación.";
                return mensaje;
            }

        }


        public static void SendEmail(string email, string mensaje)
        {
            try
            {
                string emailRemitente = "coordinacionActividades@aguasdevalencia.es";
                string nombreRemitente = "[Coordinacion de Actividades]";
                string asunto = "Aviso de Caducidad de Documentos";
                string[] toList = new[] { email };
                string[] ccList = null;
                Dictionary<string, byte[]> attachements = new Dictionary<string, byte[]>();

                NotificacionServiceClient notificatorClient = new NotificacionServiceClient();
                notificatorClient.SendEmailAsync(emailRemitente, nombreRemitente, toList,
ccList, asunto, mensaje, attachements);//con SenEmailAsync se manda asicronamente (no esperamos
a que el mensaje se envie para continuar)

            }
             catch (Exception e)
                {
                    Console.CapsLock.ToString();
                }
        }

    }
}
```

11.2.10. cls_proveedores.cs

```
using DevExpress.Web;
using proveedoresX.Clases;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace proveedoresX
{
    public class cls_proveedores
    {

        public static void trabajadorValido(int IdTrabajador)
        {

            using (proveedoresEntities context = new proveedoresEntities())
            {
                bool tiene_falso = (from contact in context.documentostrabajors
                                    where contact.Trabajador_idTrabajador == IdTrabajador
                                 && contact.validado == false
                                    select contact).Any();


                var trabajador = context.trabajadors.Where(w => w.idTrabajador ==
IdTrabajador).FirstOrDefault();
                if (trabajador != null)
                {
                    trabajador.esVerde = !tiene_falso;

                    context.Entry(trabajador).CurrentValues.SetValues(trabajador);
                    context.SaveChanges();
                }


            }
        }

        public static void CambiarValidezDocumento(int IdDocumento, bool esValido) //AQUI
        {

            using (proveedoresEntities context = new proveedoresEntities())
            {

                int idTrabajador = (from contact in context.documentostrabajors
                                    where contact.idDocumentosTrabajor == IdDocumento
                                    select contact.Trabajador_idTrabajador).FirstOrDefault();
                int idEmpresa = (from contact in context.documentostrabajors
                                 where contact.idDocumentosTrabajor == IdDocumento
                                 select contact.Trabajador_Empresa_idEmpresa).FirstOrDefault();

                var documento = context.documentostrabajors.Where(w => w.idDocumentosTrabajor ==
IdDocumento).FirstOrDefault();
                if (documento != null)
                {
                    documento.validado = esValido;

                    context.Entry(documento).CurrentValues.SetValues(documento);
                    context.SaveChanges();

                    trabajadorValido(idTrabajador);
                    empresaValido(idEmpresa); // para comprobación cruzada con trabajador
                }

            }
        }

        public static void CambiarHistoricoDocumento(int IdDocumento, bool esValido)
        {

            using (proveedoresEntities context = new proveedoresEntities())
            {
```

```
                int idTrabajador = (from contact in context.documentostrabajors
                                    where contact.idDocumentosTrabajor == IdDocumento
                                    select contact.Trabajador_idTrabajador).FirstOrDefault();

                var documento = context.documentostrabajors.Where(w => w.idDocumentosTrabajor ==
IdDocumento).FirstOrDefault();
                if (documento != null)
                {
                    documento.historico = esValido;

                    context.Entry(documento).CurrentValues.SetValues(documento);
                    context.SaveChanges();

                    //trabajadorValido(idTrabajador);
                }

            }
        }

        public static void empresaValido(int idEmpresa)
        {

            using (proveedoresEntities context = new proveedoresEntities())
            {
                bool tiene_falso = (from contact in context.documentosempresas
                                    where contact.Empresa_idEmpresa == idEmpresa
                                  && contact.validado == false
                                    select contact).Any();

                bool trabajadorMalo = (from cont in context.trabajadors //añadido para la
comprobacion de empresa-trabajadores
                                       where cont.Empresa_idEmpresa == idEmpresa &&
cont.historico == false
                                     && cont.esVerde == false
                                       select cont).Any();



                var empresa = context.empresas.Where(w => w.idEmpresa ==
idEmpresa).FirstOrDefault();
                if (empresa != null)
                {
                    empresa.esVerde = !tiene_falso && !trabajadorMalo;
                    // && !trabajadorMalo;

                    context.Entry(empresa).CurrentValues.SetValues(empresa);
                    context.SaveChanges();

                }

            }
        }

        public static void CambiarValidezDocumentoEmpresa(int IdDocumento, bool esValido)
        {

            using (proveedoresEntities context = new proveedoresEntities())
            {

                int idEmpresa = (from contact in context.documentosempresas
                                 where contact.idDocumentosEmpresa == IdDocumento
                                 select contact.Empresa_idEmpresa).FirstOrDefault();

                var documento = context.documentosempresas.Where(w => w.idDocumentosEmpresa ==
IdDocumento).FirstOrDefault();
                if (documento != null)
                {
                    documento.validado = esValido;

                    context.Entry(documento).CurrentValues.SetValues(documento);
                    context.SaveChanges();

                    empresaValido(idEmpresa);
```

```
                }

            }
        }

        public static void CambiarHistoricoDocumentoEmpresa(int IdDocumento, bool esValido)
        {

            using (proveedoresEntities context = new proveedoresEntities())
            {

                int idEmpresa = (from contact in context.documentosempresas
                                 where contact.idDocumentosEmpresa == IdDocumento
                                 select contact.Empresa_idEmpresa).FirstOrDefault();

                var documento = context.documentosempresas.Where(w => w.idDocumentosEmpresa ==
IdDocumento).FirstOrDefault();
                if (documento != null)
                {
                    documento.historico = esValido;

                    context.Entry(documento).CurrentValues.SetValues(documento);
                    context.SaveChanges();

                    //trabajadorValido(idTrabajador);
                }

            }
        }

        public static void DeshabilitarEdicionGrid(ASPxGridView grd)
        {
            //grd.SettingsContextMenu.Enabled = false;
            //grd.SettingsContextMenu.EnableColumnMenu = DevExpress.Utils.DefaultBoolean.False;
            //grd.SettingsContextMenu.EnableGroupPanelMenu =
DevExpress.Utils.DefaultBoolean.False;
            //grd.SettingsContextMenu.EnableRowMenu = DevExpress.Utils.DefaultBoolean.False;
            //grd.SettingsContextMenu.RowMenuItemVisibility.EditRow = false;
            //grd.SettingsContextMenu.RowMenuItemVisibility.DeleteRow = false;
            //grd.SettingsContextMenu.RowMenuItemVisibility.NewRow = false;
            //grd.ClientSideEvents.RowDblClick = "";

            foreach (GridViewColumn item in grd.Columns)
            {
                if (item.GetType() == typeof(GridViewCommandColumn))
                {
                    item.Visible = false;
                }
            }
        }


    }
}
```

11.3. Scritp Base de Datos

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema estadoproveedore
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema estadoproveedore
```

```
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `estadoproveedore` DEFAULT CHARACTER SET utf8 ;
USE `estadoproveedore` ;

-- -----------------------------------------------------
-- Table `estadoproveedore`.`empresa`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`empresa` (
  `idEmpresa` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `CIF` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `nombreEmpresa` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `direccion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `responsablePrevencion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `telefonoPrevencion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `correo` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `servicioPrevencion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `esVerde` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `idEmpresaAjena` INT(11) NULL DEFAULT NULL COMMENT '',
  `actividad` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `municipio` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `CP` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `historico` TINYINT(1) NULL DEFAULT '0' COMMENT '',
  PRIMARY KEY (`idEmpresa`)  COMMENT '')
ENGINE = InnoDB
AUTO_INCREMENT = 54
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`documentosempresa`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`documentosempresa` (
  `idDocumentosEmpresa` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `tipoDocumento` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `fechaCarga` DATE NULL DEFAULT NULL COMMENT '',
  `fechaCaducidad` DATE NULL DEFAULT NULL COMMENT '',
  `fechaEmision` DATE NULL DEFAULT NULL COMMENT '',
  `validado` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `historico` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `descripcion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `Empresa_idEmpresa` INT(11) NOT NULL COMMENT '',
  `usuario` VARCHAR(50) NULL DEFAULT NULL COMMENT '',
  PRIMARY KEY (`idDocumentosEmpresa`, `Empresa_idEmpresa`)  COMMENT '',
  INDEX `fk_DocumentosEmpresa_Empresa1_idx` (`Empresa_idEmpresa` ASC)  COMMENT '',
  CONSTRAINT `fk_DocumentosEmpresa_Empresa1`
    FOREIGN KEY (`Empresa_idEmpresa`)
    REFERENCES `estadoproveedore`.`empresa` (`idEmpresa`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 341
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`equipovehiculo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`equipovehiculo` (
  `idEquipoVehiculo` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `tipoEquipo` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `descripcion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `manual` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `informacion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `matricula` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `itv` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `Empresa_idEmpresa` INT(11) NOT NULL COMMENT '',
  PRIMARY KEY (`idEquipoVehiculo`, `Empresa_idEmpresa`)  COMMENT '',
  INDEX `fk_EquipoVehiculo_Empresa1_idx` (`Empresa_idEmpresa` ASC)  COMMENT '',
  CONSTRAINT `fk_EquipoVehiculo_Empresa1`
    FOREIGN KEY (`Empresa_idEmpresa`)
    REFERENCES `estadoproveedore`.`empresa` (`idEmpresa`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 17
DEFAULT CHARACTER SET = utf8;
```

```
-- -----------------------------------------------------
-- Table `estadoproveedore`.`documentosequipo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`documentosequipo` (
  `idDocumentosEquipo` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `tipoDocumento` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `fechaCarga` DATE NULL DEFAULT NULL COMMENT '',
  `fechaCaducidad` DATE NULL DEFAULT NULL COMMENT '',
  `fechaEmision` DATE NULL DEFAULT NULL COMMENT '',
  `validado` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `historico` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `descripcion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `EquipoVehiculo_idEquipoVehiculo` INT(11) NOT NULL COMMENT '',
  `EquipoVehiculo_Empresa_idEmpresa` INT(11) NOT NULL COMMENT '',
  `usuario` VARCHAR(50) NOT NULL COMMENT '',
  PRIMARY KEY (`idDocumentosEquipo`, `EquipoVehiculo_idEquipoVehiculo`,
`EquipoVehiculo_Empresa_idEmpresa`)  COMMENT '',
  INDEX `fk_DocumentosEquipo_EquipoVehiculo1_idx` (`EquipoVehiculo_idEquipoVehiculo`
ASC, `EquipoVehiculo_Empresa_idEmpresa` ASC)  COMMENT '',
  CONSTRAINT `fk_DocumentosEquipo_EquipoVehiculo1`
    FOREIGN KEY (`EquipoVehiculo_idEquipoVehiculo` , `EquipoVehiculo_Empresa_idEmpresa`)
    REFERENCES `estadoproveedore`.`equipovehiculo` (`idEquipoVehiculo` ,
`Empresa_idEmpresa`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 21
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`obra`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`obra` (
  `idObra` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `nombreObra` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `domicilio` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `fechaInicio` DATETIME NULL DEFAULT NULL COMMENT '',
  `fechaFin` DATETIME NULL DEFAULT NULL COMMENT '',
  `responsableGavsa` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `correoGavsa` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `telefonoGavsa` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `proyecto` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `responsableContratista` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `correoContratista` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `telefonoContratista` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `Empresa_idEmpresa` INT(11) NOT NULL COMMENT '',
  PRIMARY KEY (`idObra`, `Empresa_idEmpresa`)  COMMENT '',
  INDEX `fk_Obra_Empresa1_idx` (`Empresa_idEmpresa` ASC)  COMMENT '',
  CONSTRAINT `fk_Obra_Empresa1`
    FOREIGN KEY (`Empresa_idEmpresa`)
    REFERENCES `estadoproveedore`.`empresa` (`idEmpresa`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 11
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`documentosobra`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`documentosobra` (
  `idDocumentosObra` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `tipoDocumento` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `fechaCarga` DATE NULL DEFAULT NULL COMMENT '',
  `FechaEmision` DATE NULL DEFAULT NULL COMMENT '',
  `validado` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `historico` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `descripcion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `Obra_idObra` INT(11) NOT NULL COMMENT '',
  `Obra_Empresa_idEmpresa` INT(11) NOT NULL COMMENT '',
  `usuario` VARCHAR(50) NULL DEFAULT NULL COMMENT '',
  PRIMARY KEY (`idDocumentosObra`, `Obra_idObra`, `Obra_Empresa_idEmpresa`)  COMMENT '',
  INDEX `fk_DocumentosObra_Obra1_idx` (`Obra_idObra` ASC, `Obra_Empresa_idEmpresa` ASC)
COMMENT '',
```

```
  CONSTRAINT `fk_DocumentosObra_Obra1`
    FOREIGN KEY (`Obra_idObra` , `Obra_Empresa_idEmpresa`)
    REFERENCES `estadoproveedore`.`obra` (`idObra` , `Empresa_idEmpresa`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 14
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`trabajoespecial`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`trabajoespecial` (
  `idTrabajoEspecial` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `tipoTrabajo` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `descripcion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `fechaCaducidad` DATETIME NULL DEFAULT NULL COMMENT '',
  `Empresa_idEmpresa` INT(11) NOT NULL COMMENT '',
  PRIMARY KEY (`idTrabajoEspecial`, `Empresa_idEmpresa`)  COMMENT '',
  INDEX `fk_TrabajoEspecial_Empresa1_idx` (`Empresa_idEmpresa` ASC)  COMMENT '',
  CONSTRAINT `fk_TrabajoEspecial_Empresa1`
    FOREIGN KEY (`Empresa_idEmpresa`)
    REFERENCES `estadoproveedore`.`empresa` (`idEmpresa`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 19
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`documentostrabajo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`documentostrabajo` (
  `idDocumentosTrabajo` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `DocumentoTrabajo` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `fechaCarga` DATE NULL DEFAULT NULL COMMENT '',
  `fechaCaducidad` DATE NULL DEFAULT NULL COMMENT '',
  `fechaEmision` DATE NULL DEFAULT NULL COMMENT '',
  `validado` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `historico` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `descripcion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `TrabajoEspecial_idTrabajoEspecial` INT(11) NOT NULL COMMENT '',
  `TrabajoEspecial_Empresa_idEmpresa` INT(11) NOT NULL COMMENT '',
  `usuario` VARCHAR(50) NOT NULL COMMENT '',
  PRIMARY KEY (`idDocumentosTrabajo`, `TrabajoEspecial_idTrabajoEspecial`,
`TrabajoEspecial_Empresa_idEmpresa`)  COMMENT '',
  INDEX `fk_DocumentosTrabajo_TrabajoEspecial1_idx` (`TrabajoEspecial_idTrabajoEspecial`
ASC, `TrabajoEspecial_Empresa_idEmpresa` ASC)  COMMENT '',
  CONSTRAINT `fk_DocumentosTrabajo_TrabajoEspecial1`
    FOREIGN KEY (`TrabajoEspecial_idTrabajoEspecial` ,
`TrabajoEspecial_Empresa_idEmpresa`)
    REFERENCES `estadoproveedore`.`trabajoespecial` (`idTrabajoEspecial` ,
`Empresa_idEmpresa`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 26
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`trabajador`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`trabajador` (
  `idTrabajador` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `nombre` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `DNI` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `puestoTrabajo` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `informacion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `esVerde` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `historico` TINYINT(1) NULL DEFAULT '0' COMMENT '',
  `Empresa_idEmpresa` INT(11) NOT NULL COMMENT '',
  PRIMARY KEY (`idTrabajador`, `Empresa_idEmpresa`)  COMMENT '',
  INDEX `fk_Trabajador_Empresa1_idx` (`Empresa_idEmpresa` ASC)  COMMENT '',
  CONSTRAINT `fk_Trabajador_Empresa1`
```

```
    FOREIGN KEY (`Empresa_idEmpresa`)
    REFERENCES `estadoproveedore`.`empresa` (`idEmpresa`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 186
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`documentostrabajor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`documentostrabajor` (
  `idDocumentosTrabajor` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `tipoDocumento` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `fechaCarga` DATE NULL DEFAULT NULL COMMENT '',
  `fechaEmision` DATE NULL DEFAULT NULL COMMENT '',
  `fechaCaducidad` DATE NULL DEFAULT NULL COMMENT '',
  `validado` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `historico` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '',
  `descripcion` VARCHAR(255) NULL DEFAULT NULL COMMENT '',
  `Trabajador_idTrabajador` INT(11) NOT NULL COMMENT '',
  `Trabajador_Empresa_idEmpresa` INT(11) NOT NULL COMMENT '',
  `usuario` VARCHAR(50) NULL DEFAULT NULL COMMENT '',
  PRIMARY KEY (`idDocumentosTrabajor`, `Trabajador_idTrabajador`,
`Trabajador_Empresa_idEmpresa`)  COMMENT '',
  INDEX `fk_DocumentosTrabajor_Trabajador1_idx` (`Trabajador_idTrabajador` ASC,
`Trabajador_Empresa_idEmpresa` ASC)  COMMENT '',
  CONSTRAINT `fk_DocumentosTrabajor_Trabajador1`
    FOREIGN KEY (`Trabajador_idTrabajador` , `Trabajador_Empresa_idEmpresa`)
    REFERENCES `estadoproveedore`.`trabajador` (`idTrabajador` , `Empresa_idEmpresa`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 1237
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`log`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`log` (
  `id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `usuario` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `accion` VARCHAR(45) NULL DEFAULT NULL COMMENT '',
  `resumen` VARCHAR(512) NULL DEFAULT NULL COMMENT '',
  `fecha` DATETIME NULL DEFAULT NULL COMMENT '',
  PRIMARY KEY (`id`)  COMMENT '')
ENGINE = InnoDB
AUTO_INCREMENT = 394
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `estadoproveedore`.`subcontrata`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `estadoproveedore`.`subcontrata` (
  `id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `idObra` INT(11) NULL DEFAULT NULL COMMENT '',
  `idEmpresa` INT(11) NULL DEFAULT NULL COMMENT '',
  `idEmpresaPadre` INT(11) NULL DEFAULT NULL COMMENT '',
  PRIMARY KEY (`id`)  COMMENT '',
  UNIQUE INDEX `subcontrataUnico` (`idObra` ASC, `idEmpresa` ASC)  COMMENT '',
  INDEX `idContratante_idx` (`idObra` ASC)  COMMENT '',
  INDEX `idContratado_idx` (`idEmpresa` ASC)  COMMENT '')
ENGINE = InnoDB
AUTO_INCREMENT = 40
DEFAULT CHARACTER SET = utf8;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```