Grado en Ingeniería en Sistemas Audiovisuales y Multimedia

Escuela Técnica Superior de Ingeniería de Telecomunicación

Curso académico 2016-2017

Trabajo Fin de Grado

FRONTEND DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE PARTES DE OBRA

Autor: Carlos Rodríguez García
Tutor: José María Cañas Plaza

Madrid 2017

A mi familia,
en especial a mis padres y mi hermana,
por darme esta oportunidad y todo su apoyo.

A mis amigos Alba y Nacho
por acompañarme todos estos años de aventura.

Y a Cristina
por todo su cariño y estar siempre a mi lado.

Gracias.

Índice general

Índice de figuras

Índice de tablas

Índice de fragmentos

Capítulo 1

Introducción

En este capítulo se introducen los conceptos de aplicaciones web y sus características así como la motivación principal que ha impulsado el desarrollo de este trabajo de fin de grado. Este TFG se ha desarrollado dentro de un proyecto industrial que aborda el desarrollo de una aplicación web para la gestión y generación de reportes o '*Partes de Obra*' para una empresa constructora de túneles.

1.1. Aplicaciones Web

El desarrollo de aplicaciones web ha evolucionado enormemente en la última década, tanto desde el punto de vista del desarrollo de software (numerosos entornos de desarrollo, bibliotecas, aplicaciones configurables, plugins, etc) y de la administración de sistemas (servicios de alojamiento, técnicas de estabilidad, monitorización, etc) como en funcionalidad.

En los últimos años se ha incrementado el uso de tecnologías web en muchos ámbitos. Se han ido adaptando y transformado para pasar de ser una página web de consulta de información, denominadas páginas web de contenido *estático*, a tener una fuerte interacción con el usuario, personalizándose la información e interactuando con el servidor, mostrando resultados en función de esa interacción del historial, etc, (webs *dinámicas*). En este camino es donde se ha encontrado un punto en común entre las páginas web y los programas, denominándose '*Aplicaciones web*'.

En la ingeniería de software se denomina '`Aplicación web`' a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador, utilizando como protocolo de comunicación `HTTP`. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores en la que se confía la ejecución al navegador y a un servidor web.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, a la independencia del sistema operativo, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales.

Las aplicaciones web quedan estructuradas en dos partes:

- *Cliente:* Permite crear interfaces de usuario atractivos e intuitivos para facilitar la introducción de datos por parte del usuario y permite la comunicación con el servidor. Las aplicaciones web que se basan en una mucha interactividad con el usuario suelen hacer especial énfasis en esta parte, ya que al ejecutarse dentro del navegador, hacen esta interactuacción muy rápida y fluida.

- *Servidor:* Es el encargado de recibir las peticiones, principalmente `HTTP` aunque también puede soportar otras tecnologías como `AJAX` o `WebSockets`, y responder de manera adecuada a estas peticiones. Permite implementar el comportamiento y la lógica de la aplicación, recibir peticiones, realizar búsquedas, etc, son algunos de los ejemplos que suelen estar ligados a una base de datos que nutre a la aplicación de todo el contenido.

El concepto de aplicación web se remonta al primer lenguaje de programación dedicado para este ámbito, `Perl`, inventado por Larry Wall en 1987. Desde entonces hasta hoy en día, los navegadores han ido ganando en rapidez, capacidad de cómputo y funcionalidades. Los estándares y lenguajes de programación dedicados a este sector han alcanzado un alto grado de optimización y madurez que permite ejecutar aplicaciones en cualquier dispositivo como si de una aplicación de escritorio se tratara usando únicamente el navegador. Las figuras 1.1, 1.2 1.3 y 1.4 son un ejemplo de aplicación web.

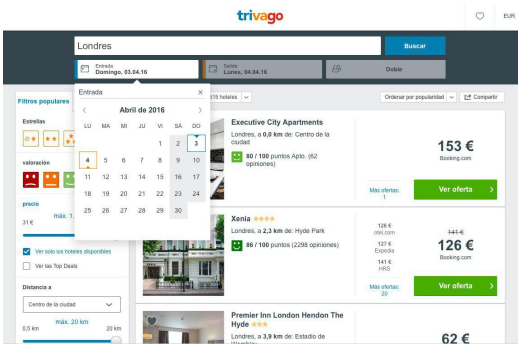


Figura 1.1: Trivago

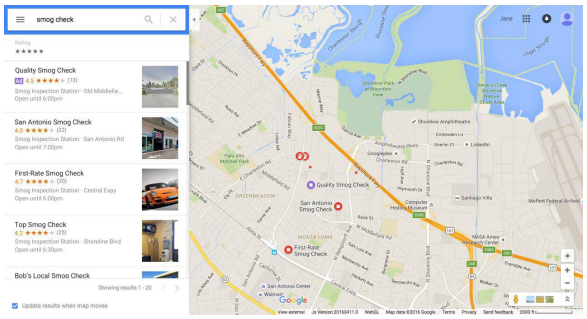


Figura 1.2: Google Maps

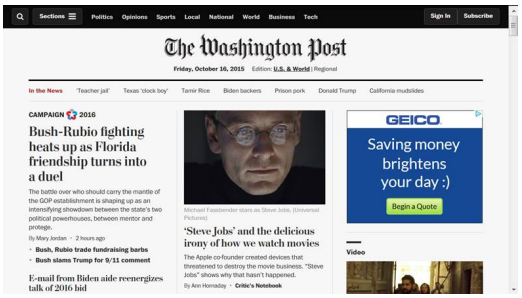


Figura 1.3: Washigton Post

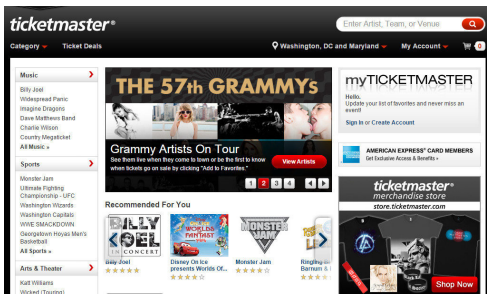


Figura 1.4: TicketMaster

Gran parte del crecimiento del uso de las aplicaciones web reside en que `JavaScript` se ha convertido en el lenguaje estandar para dotar al cliente de la lógica necesaria. Se puede observar en la figura 1.5 cómo el rendimiento del intérprete de `JavaScript` ha crecido exponencialmente con los años. Esto permite que las aplicaciones web aumenten la carga computacional en el cliente, siendo capaces de resolverlo de un modo suficientemente ágil. Esta evolución deja clara la apuesta que se está haciendo por el uso de aplicaciones web en el sector.

Hoy por hoy las aplicaciones web son usadas constantemente para todo tipo de entornos, ya sea comprar por internet, consultar prensa personalizada, multitud de herramientas de trabajo, acceso a contenido multimedia, llamadas de voz, video conferencias, etc.

Un ejemplo claro de aplicación web puede ser la página `Amazon`[1] (figura 1.6) que se basa en una de las características más importantes de las aplicaciones web, el contenido dinámico. Esta

[1] `https:www.amazon.es`

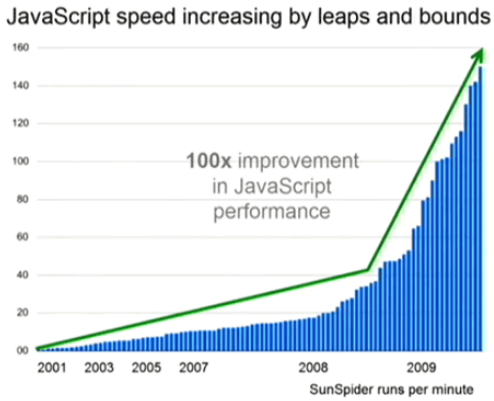


Figura 1.5: Evolución del rendimiento de JavaScript

característica nutre a la página de un contenido personalizado para cada usuario, ofreciéndole productos en función de compras anteriores o lugares que ha visitado.

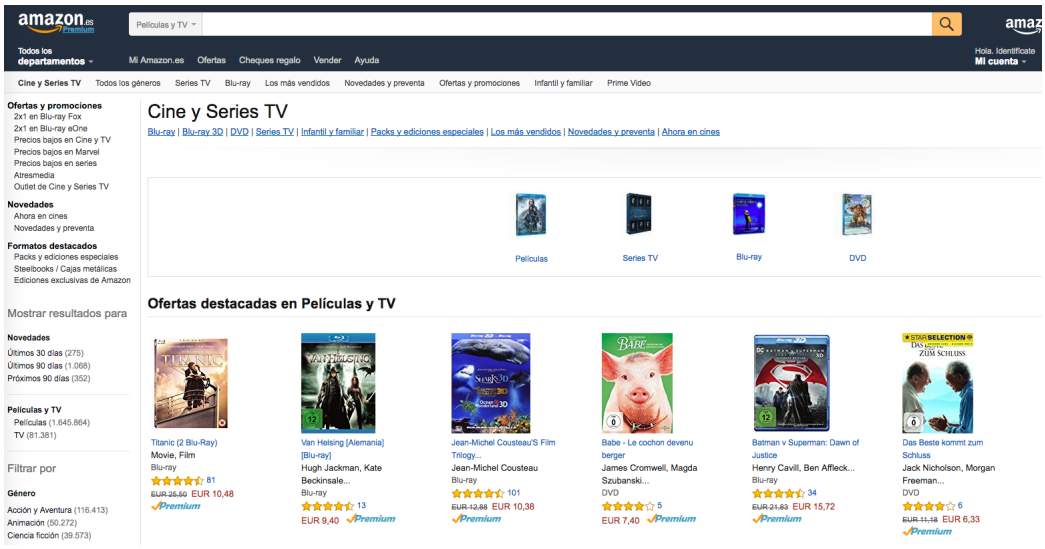


Figura 1.6: Ejemplo de Amazon como página dinámica

Otro ejemplo muy popular es `Netflix` (figura 1.7), una de las plataformas de vídeo en streaming más popular que existe en la actualidad. Es otra buena muestra de aplicación web basada en el contenido dinámico, recomendando series o películas relacionadas con las que ya han sido reproducidas, todas ellas bajo demanda, y con posibilidad de descarga para su visualización sin acceso a internet. `Netflix` y otras plataformas similares son la demostración del crecimiento en los últimos años de las aplicaciones web en el ámbito del contenido multimedia.
El uso de aplicaciones web frente a aplicaciones tradicionales de escritorio ofrece unas ventajas e inconvenientes que se detallan en los puntos 1.1.1 y 1.1.2.

1.1.1. Ventajas

Los aspectos positivos del uso de una aplicación web podemos desglosarlos en los siguientes puntos:

- El cliente sólo necesita tener acceso a un navegador web con conexión a internet para usar la aplicación, por lo que no necesita descargar, instalar ni configurar software específico.

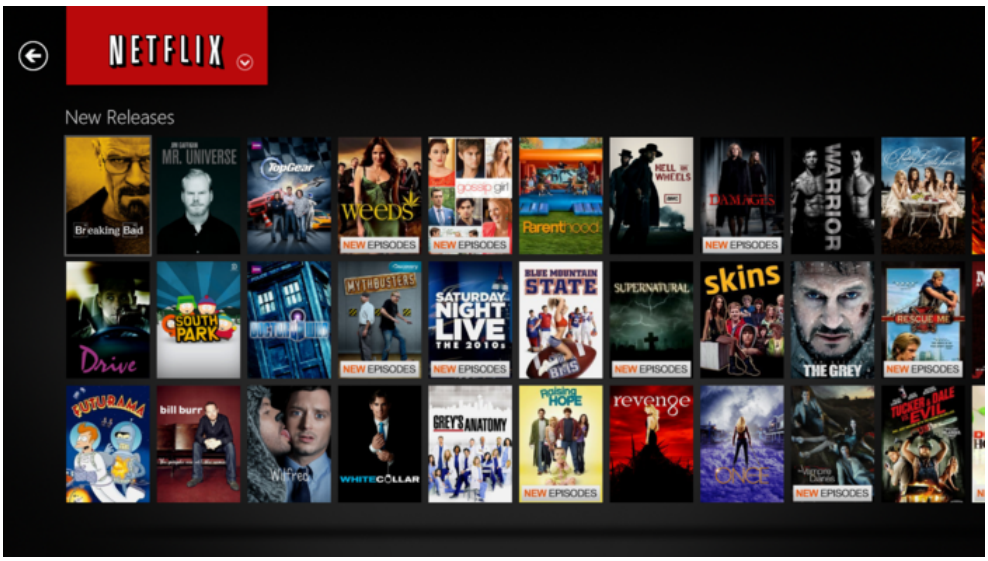


Figura 1.7: Ejemplo de Netflix como aplicación web dinámica

- Son multiplataforma. Puesto que el único software necesario en el lado del cliente es un navegador web, podemos usar una aplicación web desde cualquier dispositivo que disponga de uno. Además, desde el punto de vista de la programación, simplifica mucho el proceso ya que sólo es necesario desarrollar un único software para todas las plataformas (`Linux`, `MS-WindowsMacOS`) y no uno para cada una, traduciéndose en un menor coste de desarrollo y, por tanto, económico.
- Hacen uso de la responsividad. El acceso a la aplicación desde el navegador web permite que el contenido visual de la página se adapte a la pantalla del dispositivo que está accediendo. Igual que en el punto anterior, con un único desarrollo se estructura cada uno de los elementos de los que se compone la página web. Puede verse en la figura 1.8 un ejemplo.

Figura 1.8: Adaptación de la estructura mediante responsividad

- No requieren actualizaciones por parte del cliente. Una actualización por parte del servidor sobre la aplicación provoca que el acceso de un cliente descargue el nuevo contenido. Este proceso es ajeno al cliente ya que no tiene que realizar ningún procedimiento para disponer de la última versión.
- No hay problemas de compatibilidad entre versiones pues todos los usuarios trabajan con la última versión lanzada en el servidor.
- Menores requisitos de memoria. Las aplicaciones web suelen tener menor demanda de recursos *hardware* que el software instalado localmente.

1.1.2. Inconvenientes

Como contrapartida, las aplicaciones web presentan las siguientes desventajas:

- Requieren conexión a la red. Aunque cada vez se está popularizando más el uso de las aplicaciones web de manera *offline*, sí se requiere, al menos, un primer acceso para una descarga previa de los documentos necesarios para una navegación sin conexión. Por tanto, son dependientes del tipo de conexión a la red así como posibles problemas de acceso al servidor (p.e. saturación). Un fallo en uno o varios servidores podría dejar completamente sin servicio a las aplicaciones.
- Suelen tener menor rendimiento que las aplicaciónes de escritorio, ya que no son tan eficientes utilizando los recursos *hardware* del lado cliente.
- El usuario no puede elegir la versión de la aplicación que desea utilizar, ya que sólo puede ser usada aquella alojada en el servidor. Esta desventaja no suele ser demasiado significativa excepto en casos muy concretos.
- Son menos seguras que las aplicaciones de escritorio puesto que muchas operaciones necesitan de transacción con el servidor.
- El acceso a los datos es más lento debido a que el contenido es descargado cuando se accede al servidor. En el caso de las aplicaciones de escritorio, al estar alojadas en la máquina, el acceso es inmediato.

Actualmente las aplicaciones web están viviendo un auge en su uso promovido por todas las nuevas novedades que se han ido introduciendo en las tecnologías web.

1.2. Tecnologías Web

En este apartado se repasan las tecnologías web más comunes actualmente cuando se desarrollan aplicaciones web, tanto en el lado cliente como en el lado servidor, además de la comunicación entre cada una de las partes.

1.2.1. HTTP

Hypertext Transfer Protocol o HTTP (protocolo de transferencia de hipertexto) es el protocolo de comunicación, basado en texto, que permite las transferencias de información en la World Wide Web. `HTTP` fue desarrollado por el *World Wide Web Consortium* y la *Internet Engineering Task Force*, colaboración que culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el RFC 2616 que especifica la versión 1.1 de este protocolo. `HTTP` define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Trabaja sobre el protocolo TCP de la capa de transporte.

Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador web) se lo conoce como agente del usuario. A la información transmitida se le llama recurso y se identifica mediante un localizador uniforme de recursos (URL). El recurso es el resultado de la ejecución de un programa en el lado servidor como puede ser: una consulta a una base de datos, solicitar un documento, etc.

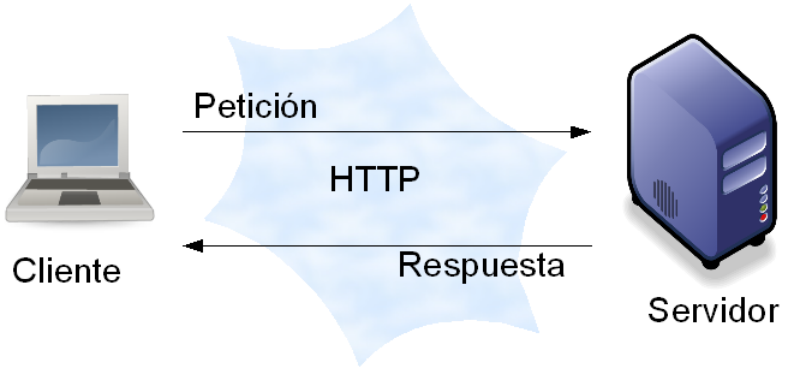


Figura 1.9: Modelo Cliente-Servidor

Además `HTTP` es un protocolo sin estados, por lo que no guarda información sobre conexiones anteriores. Actualmente el desarrollo de aplicaciones web requiere frecuentemente guardar información sobre estado. Para este cometido se utilizan las *Cookies* que es información de un servidor que puede almacenar en el navegador del cliente durante un tiempo limitado. Esta herramienta se utiliza con múltiples propósitos como, por ejemplo, implementar la noción de sesión en aplicaciones web.

1.2.2. Tecnologías Cliente

Hasta hace unos años, las aplicaciones web no podían competir con las aplicaciones de escritorio por los motivos vistos en la sub-sección 1.1.2, pero en últimos tiempos las innovaciones en las tecnologías web han permitido a las aplicaciones web igualarlas en muchos aspectos. Algunas de estas innovaciones son: el almacenamiento local, los "`Web Workers`", la introducción de gráficos web acelerados por hardware GPU en navegadores web, funcionalidad *offline*, `AJAX`, etc.

El modelo más usado para la construcción de páginas web en el lado cliente es `HTML`. `HTML`, en su versión 5, es un lenguaje de marcado estandarizado para la elaboración de páginas web, que permite definir su estructura y contenido.

A lo largo de sus diferentes versiones, se han incorporado y suprimido diversas características, con el fin de hacerlo más eficiente y facilitar el desarrollo de páginas web compatibles con distintos navegadores y plataformas (PC de escritorio, portátiles, teléfonos inteligentes, tabletas, etc.). No obstante, para interpretar correctamente una nueva versión de `HTML`, los desarrolladores de navegadores web deben incorporar estos cambios. Por estas razones, aún existen diferencias entre distintos navegadores y versiones al interpretar una misma página web.

El fuerte uso del estándar está haciendo que, claramente, se apueste por `HTML5`, reduciendo estas diferencias entre navegadores y dejando atrás otros entornos muy conocidos y no estandarizados como `Adobe Flash Player`, perteneciente a `Adobe`.

`HTML5` gestiona mejor los recursos del dispositivo donde se está renderizando la página por lo que disminuye los tiempos de carga, haciéndo más fluida la navegación. Además permite el acceso a elementos del equipo que hasta ahora requerían de *plugins* externos. Estos pueden ser: acceso a la cámara o micrófono del ordenador para permitir efectuar llamadas o video

conferencias, renderización de gráficos en tiempo real, como se ve en la figura 1.10, usando el hardware del equipo. Esto permite que aplicaciones con gran carga de procesamiento gráfico aproveche los recursos *hardware* locales y ofrezca resultados equiparables a aplicaciones que requieren de instalación.

Figura 1.10: Gráficos generados mediante JavaScript

Dado que `HTML5` aporta únicamente la estructura, son necesarios otros elementos que le aporten el estilo y presentación visual que se busca en la construcción de páginas web. Es por ello que, al igual que ha ocurrido con `HTML` y su evolución, se opte por `CSS`, en su versión 3, para añadir los estilos.

Estos dos lenguajes están estandarizados por el *World Wide Web Consortium* o `W3C`. Es un consorcio internacional que genera recomendaciones y estándares en el ámbito de las tecnologías web. Los estándares han ayudado a que se cree una uniformidad en el uso de estas tecnologías que ha permitido el crecimiento.

Por último, en el lado cliente se utiliza el lenguaje `JavaScript` para la interacción con el usuario y añadir lógica y dinamismo a las páginas web. El uso más común de `JavaScript` es escribir funciones incluidas en páginas `HTML`. Algunos ejemplos sencillos de este uso son:

- Cargar nuevo contenido para la página o enviar datos al servidor a través de una comunicación asíncrona con el servidor (`AJAX`) sin necesidad de recargar la página completa.
- Animación de los elementos de página, hacerlos desaparecer, cambiar su tamaño, moverlos, etc.
- Contenido interactivo como juegos y reproducción de audio y vídeo, etc.
- Validación de los valores de entrada de un formulario web para asegurarse de que son aceptables antes de ser enviado al servidor.

Estos tres lenguajes estandarizados hacen que la tendencia al uso de aplicaciones web se vea incrementada notablemente. Los resultados finales cada vez son menos distantes con una aplicación de escritorio en cuanto recursos, tiempos de carga y usabilidad.

Figura 1.11: Estándar Web

1.2.3. Tecnologías Servidor

En el otro lado de las aplicaciones web se encuentra el servidor. Es el encargado de recibir las solicitudes `HTTP` que llegan desde el cliente y responder a ellas de manera adecuada. A diferencia del cliente, las tecnologías usadas en el servidor son más heterogéneas. El lado servidor no cuenta con una estandarización que provoque una tendencia de uso de un tipo u otro de tecnología debido a que existe más de un paradigma, multitud de lenguajes de programación, bases de datos, sistemas operativos, etc. Es en este contexto donde han ido apareciendo distintas formas de desarrollar un servidor web, denominándose *entornos de desarrollo*.

Actualmente existen varios entornos del lado del servidor muy populares entre la comunidad de desarrolladores. Entre ellos se encuentra `Node.js`[2], un entorno en tiempo de ejecución multiplataforma, de código abierto, basado en el lenguaje de programación interpretado `ECMAScript` o (`JavaScript`), asíncrono y con una arquitectura orientada a eventos.

Figura 1.12: Entorno de desarrollo Node.js

Otro de los entornos de desarrollo en el lado servidor más populares basado en el lenguaje de programación `Python` es `Django`[3]. Es un entorno web de código abierto que respeta el patrón de diseño conocido como Modelo–Vista–Controlador. La meta fundamental de `Django` es facilitar la creación de sitios web complejos. Pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio "No te repitas"(DRY, del inglés *Don't Repeat Yourself*). `Python` es usado en todas las partes del entorno, incluso en configuraciones, archivos y en los modelos de datos.

Otro ejemplo, con una filosofía de diseño muy parecida a la de `Django` es `Ruby on Rails`[4]. También conocido como RoR, es un entorno de código abierto escrito en el lenguaje de programación `Ruby`, siguiendo también el paradigma del patrón Modelo-Vista-Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros entornos y con un mínimo de configuración.

[2] `https://nodejs.org/es/`
[3] `https://www.djangoproject.com/`
[4] `http://rubyonrails.org.es//`

Figura 1.13: Entorno de desarrollo Django

El lenguaje de programación Ruby hace uso de una sintaxis que muchos de sus usuarios encuentran muy legible.

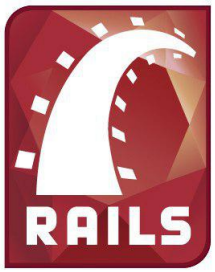


Figura 1.14: Entorno de desarrollo Ruby on Rails

Cada uno de estos entornos tendrá con toda probabilidad una base de datos asociada a los modelos definidos. Una de las grandes ventajas de estos entornos de desarrollo descritos es la facilidad con la que se conectan a las bases de datos, ya sean relacionales o no relacionales. La diferencia entre estas dos configuraciones es la manera en la que se estructuran los datos. Las bases de datos relacionales (como por ejemplo `MySQL`, `PostgreSQL` o `Microsoft SQL Server`) siguen un esquema de tablas donde el número de columnas es fijo para cada usuario. Por el contrario, las no relacionales (`MongoDB` como la más popular) aportan mayor dinamismo y pueden contener diversidad de campos para un mismo usuario.

La existencia de una base de datos para las aplicaciones web tiene una gran importancia. Para los ejemplos de aplicaciones presentadas en las sección 1.1 es necesaria una base de datos que almacene todo tipo de características del usuario para poder ofrecerle después recomendaciones basándose en búsquedas o contenido visitado, almacenar pedidos acerca de compras realizadas, etc.

Los entornos aquí presentados son sólo un ejemplo ilustrativo, no exhaustivo, cada uno con unas características y particularidades que determinarán la elección de uno u otro a la hora de desarrollar una aplicación web. Otros entornos con otros lenguajes de programación como pueden ser `Java`, y su entorno `Spring`, o uno de los lenguajes orientado a servidor más popular, `PHP`, resuelven también el desarrollo de una aplicación web.

1.2.4. Servidores en producción

Los entornos presentados en la sección 1.2.3 suelen ir acompañados de otros servidores web utilizados para desplegar y alojar los sitios en vivo o aplicaciones web desarrolladas, denominados *Servidores en producción*. Estos se encargan de gestionar las peticiones `HTTP`

que el cliente solicita y encaminarlas hacia la aplicación web desarrollada. Los servidores en producción más populares son `Apache` y `Nginx`. El primero de ellos abarca más de la mitad de los sitios web activos en la red ya que es un servidor web robusto y flexible. Por otro lado, `Nginx` está ganando popularidad dentro del mercado pese a llevar menos años activo.

1.3. Aplicación web de partes de obra para una empresa.

Este TFG se ha desarrollado en el contexto de un proyecto para una empresa que tiene el objetivo de digitalizar un proceso de introducción de *Partes de Obra*. El procedimiento para la creación de estos partes se basa en la introducción de datos ofreciendo un resumen diario de la jornada con una granularidad de introducción por hora. Hasta hace poco, se creaba un documento `Excel` con varias columnas que era impreso en papel y se distribuía a los operarios. Una vez relleno en papel pasaba por un proceso de escaneado y envío a la oficina central donde volvía a ser pasado, de nuevo, al documento de tipo `Excel`. Este proceso lleva consigo un alto coste en tiempo y no da la posibilidad de acceder a los datos inmediatamente ni permite hacerlo desde cualquier ubicación geográfica.

La principal motivación de la empresa es la de digitalizar este procedimiento buscando que sea más rápido y sencillo. Además, al guardar la información de cada parte de obra en una base de datos remota, permite la consulta y edición de estos de manera instantánea y muy sencilla desde prácticamente cualquier dispositivo electrónico que cuente con un navegador web.

Para llevar a cabo este digitalización se ha optado por una solución con tecnologías web. Una de las principales ventajas de estas tecnologías radica en la simplicidad pero, a su vez, mayor eficiencia en la recolección de información en un procedimiento que hasta ahora no tenía carácter inmediato ni acceso a los datos remotamente. La inclusión en este proyecto de una base de datos que refleja el material y personal desplegado en una obra, aporta mayor control acerca de la distribución de cada elemento. Para la introducción de datos en esa base de datos es donde entra en juego la aplicación web de este TFG. De esta manera, un operario puede introducir desde el navegador web de su dispositivo electrónico (teléfono inteligente, tablet, ordenador portátil, etc) los datos del formulario, incluso aunque no haya conexión a Internet y cuando exista, éstos se enviarán al servidor y serán tratados. Todas las transacciones se almacenarán en la base de datos de manera instantánea y su visualización en otra parte geográfica (y con cualquier dispositivo) será posible dado que accederán al mismo portal al ser multiplataforma.

En el capítulo 2 se verán los objetivos concretos tanto de la aplicación como de este TFG así como la metodología seguida y plan de trabajo. En el capítulo 3, se mostrará un resumen de las tecnologías usadas. A continuación, se procederá a explicar en detalle el diseño y funcionamiento del lado cliente de la aplicación en el capitulo 4. En el capitulo 5 se mostrará detalladamente el proceso de desarrollo tanto del interfaz de usuario como de sus principales funcionalidades. Después, se mostrarán los experimentos realizados en el capítulo 6 con el objetivo de comprobar el buen funcionamiento de la aplicación así como las opiniones y comentarios de los usuarios de esta. Por último, un capítulo de conclusiones 7 donde se expondrán los objetivos cumplidos así como los trabajos futuros que se plantean.

Capítulo 2

Objetivos

Tras presentar el contexto actual de las tecnologías web y el proyecto de empresa en el que se encuadra este Trabajo de Fin de Grado se presentan los distintos objetivos de este proyecto, así como la metodología de trabajo que se ha seguido en su desarrollo.

2.1. Objetivos

Este Trabajo de Fin de Grado está englobado dentro de un proyecto cuyo objetivo es crear una aplicación web de gestión de los partes de obra de una empresa de construcción. Hasta el momento estos partes eran creados a mano sobre una plantilla de Excel y posteriormente eran escaneados y enviados vía correo electrónico a las oficinas de la empresa. De esta manera quedaba almacenados un documento pdf sobre cada parte sin más análisis posterior que la inspección visual o la inserción a mano de sus datos en otro fichero Excel por parte de una segunda persona. Este sistema de creación de partes usa tecnologías anticuadas y poco eficientes y suponían un cierto esfuerzo y coste de tiempo a los usuarios.

El proyecto industrial en el que se encuadra este Trabajo de Fin de Grado consiste en desarrollar una aplicación web la cual permita la introducción a través de dispositivos electrónicos de estos partes de obra, cuya información quedará guardada directamente en una base de datos. Una vez guardados estos datos, se generará automáticamente el parte de obra, el cual será accesible desde la aplicación, permitiendo a los usuarios su consulta, descarga y modificación. De esta manera, se facilita en gran medida la tarea de la generación de los partes de obras, así como su envío, su consulta y tratamiento.

Para la realización del proyecto, este ha sido dividido en dos Trabajos de Fin de Grado, quedando separadas la parte cliente y la parte servidora en cada uno de ellos. En concreto, esta memoria se centra la parte cliente web de proyecto industrial. El objetivo principal es crear una interfaz web basada en una serie de formularios HTML que permitan la introducción desde un navegador web de la información que posteriormente será enviada al servidor para su almacenamiento. Una vez creados, a través de esta interfaz se podrá acceder a ellos para su consulta, descarga o edición. Además, mediante la interfaz también se debe tener acceso a las bases de datos de Proyectos, Maquinaria y Empleados para añadir, consultar, editar y eliminar sus elementos.

Este objetivo principal a su vez ha articulado los siguientes subobjetivos que la aplicación ha de cumplir:

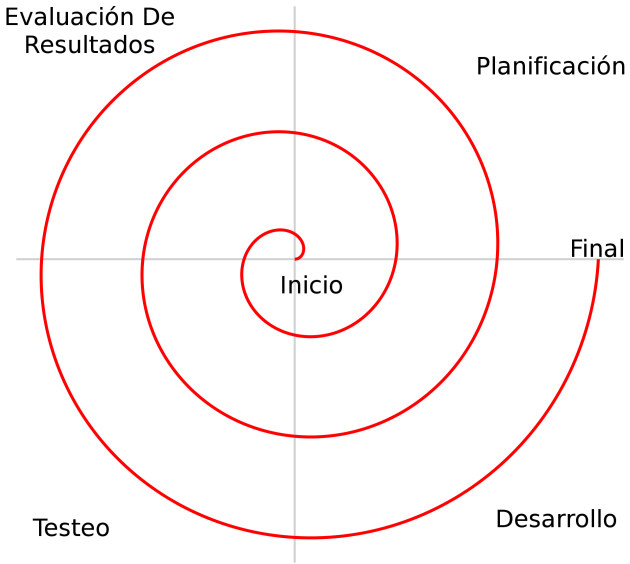
- Debe ser multiplataforma y responsiva, es decir, que se ajuste correctamente a los distintos tipos de dispositivos y de tamaños de éstos.

- Que se adapte lo mejor posible al modelo estándar de los partes de obra que se generaban anteriormente, ya que los datos introducidos a través de la interfaz deben generar partes con esta plantilla.
- Debe tener un sistema de notificaciones que permita al servidor alertar al usuario web de información importante.
- Ha de contar con un sistema de autoguardado durante la creación de partes de obra de modo que se puedan generar en varias sesiones, no necesariamente completar cada parte en una sola sesión.

Además, ha de cumplir con una serie de requisitos que permitan al usuario tener una buena experiencia de uso al trabajar con la aplicación. Debe de tener una interfaz de usuario lo más simple y usable posible, que permita que su manejo sea lo más intuitivo posible. Además, su rendimiento temporal debe ser bueno en distintos tipos de dispositivos, poniendo especial hincapié en dispositivos móviles.

2.2. Metodología

Durante el desarrollo de este Trabajo de Fin de Grado se han mantenido con reuniones periódicas, normalmente semanales, con el director del proyecto siguiendo la metodología de trabajo denominada en "Espiral". Este método se caracteriza por estar compuesto por una serie de iteraciones en forma de espiral las cuales comienzan con la creación de un primer prototipo del proyecto y al cual se le van añadiendo nuevas funcionalidades en cada interacción hasta abarcar el proyecto completo.



Cada una de estas interacciones se compone de cuatro fases:

- En la reunión con el director del proyecto se fijaban los objetivos a alcanzar durante la iteración estableciendo un orden de prioridad para estos. Además se determinaba un plan para abordar estos objetivos.
- Fase de desarrollo de dichos objetivos, con un primer periodo exploración del problema y sus posibles soluciones y un segundo periodo de desarrollo propiamente dicho de la solución elegida.

- Pruebas por parte de los desarrolladores de las nuevas funcionalidades añadidas durante la iteración.
- En la reunión siguiente con el tutor se evalúa si se ha cumplido con la planificación de la iteración.

Una vez completadas estas cuatro fases, comienza la siguiente iteración con la primera fase y así sucesivamente hasta alcanzar los todos los objetivos del proyecto.

Además de la interacción constante con el tutor, durante el desarrollo del proyecto se ha mantenido de manera constante comunicación con la empresa, tanto de forma presencial, con varias reuniones que han tenido lugar en la sede de ésta, como de forma telemática mediante correos electrónicos. Esta intercomunicación ha permitido que el desarrollo de la aplicación haya ido directamente encaminado a los deseos de la empresa, permitiéndonos a los desarrolladores solventar con ellos dudas que iban surgiendo sobre las necesidades y requisitos concretos que debía cumplir la aplicación.

En un primer momento nos proporcionaron un dossier con las primeras directrices y especificaciones iniciales que permitió dar forma al prototipo inicial, el cual fue presentado en una de estas reuniones presenciales.

Posteriormente, tras un tiempo de desarrollo, y una vez avanzadas la mayoría de las funcionalidades principales, fue lanzado un prototipo ya en plataforma de producción y se les proporcionó una serie de usuarios de prueba para que un reducido número de empleados de la empresa comenzaran a usar el aplicación y obtener de ellos la realimentación de uso. Gracias a esta realimentación se han ido puliendo distintos aspectos de la aplicación y solucionando problemas o errores que fueron surgiendo. Este prototipo en producción ha sido constantemente actualizado a la última versión de software con las últimas mejoras introducidas y la corrección de errores reportados.

En último lugar, y tras dar finalizado el desarrollo del proyecto, se pasó a una fase final en la que la empresa ha comenzado la implantación de la aplicación en todas sus obras, manteniendo la comunicación con ellos para solventar cualquier tipo de problema que pudiera surgir.

Se ha destacar que este Trabajo de Fin de Grado pertenece a un proyecto industrial dividido en dos TFGs. El desarrollo global del proyecto industrial se ha llevado a cabo de forma conjunta con el autor del otro trabajo, Nacho Arránz Águeda Para ello se trabajado en constante comunicación y de forma colaborativa, puesto que ambas partes del proyecto están íntimamente relacionadas y para su correcto funcionamiento es necesario conocer ampliamente el funcionamiento de la otra sección.

Se ha utilizado el software de control de versiones Git, en su version GitLab, que nos ha facilitado en gran medida tanto la gestión de versiones como la integración del código de ambos desarrolladores.

2.3. Plan de Trabajo

Con respecto al plan de trabajo seguido durante el desarrollo del proyecto, ha sido el siguiente:

- Exploración de las distintas tecnologías web actuales y realización de prototipos tempranos con ellas.
- Una vez elegidas las tecnologías web que se utilizarán definitivamente y basándose en las directrices iniciales de la empresa, creación de un primer prototipo de laboratorio para su presentación a la empresa con la funcionalidad básica.
- Desarrollo de las nuevas funcionalidades sobre el prototipo inicial, como las secciones de Proyectos, Maquinaria y Empleados.
- Lanzamiento de un segundo prototipo ya en desarrollo para que los trabajadores de la empresa comenzaran a interactuar con ella. Mientras se continuaba con el desarrollo de funcionalidades, se prestaba especial atención a reportes de errores generados por los usuarios.
- Una vez programadas las funcionalidades principales, se realizan labores de pulido flecos y corrección de errores finales.

Capítulo 3

Infraestructura

Para el desarrollo de este trabajo de fin de grado se ha seguido el estándar regulado por el W3C que es el más común a la hora de elaborar tanto páginas web como aplicaciones web. En este capítulo se explicará someramente la tecnología que se ha empleado en el desarrollo de la aplicación web en su lado cliente.

3.1. Lenguaje HTML

HTML, siglas en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), es un lenguaje de marcado utilizado para la construcción de páginas web. La primera versión de HTML fue lanzada en 1991 por Tim Bersers-Lee como un nuevo sistema de *hipertexto* para compartir documentos.

Este lenguaje de marcado se utiliza para definir la estructura y los elementos que forman una página web mediante el uso de una serie de etiquetas, identificadas por los caracteres < > anidadas entre sí. Cada elemento viene definido por una etiqueta de inicio (<nombre del elemento>) y por una etiqueta de cierre (<\nombre-del-elemento>), salvo algunos elementos que quedan definidos por una única etiqueta y no necesitan etiqueta de cierre.

Las etiquetas están compuestas por dos propiedades básicas: atributos y contenido. Cada una de estas propiedades tienen ciertas restricciones para se considere válido un documento HTML. Los atributos de cada elemento vienen definidos mediante una estructura clave-valor que se sitúan dentro de la etiqueta de inicio del elemento. El contenido se ubica entre las etiquetas de inicio y cierre del elemento. A modo de ejemplo, la estructura básica de un elemento HTML sería la siguiente: <nombre-del-elemento atributo="valor">Contenido<\nombre-del-elemento>.

Los elementos HTML pueden tener funciones muy variadas: (a) definir metadatos del propio documento HTML, como el elemento *head*, (b) contener líneas de código JavaScript para ser ejecutadas por el navegador web, como la etiqueta *script*, (c) elementos que permiten estructurar el contenido, como la etiqueta *div* o (d) definir contenido incrustado, por ejemplo contenido multimedia, como la etiqueta *img*.

En concreto en este trabajo de fin de grado se ha utilizado su quinta revisión, denominada HTML5, que fue publicada de forma definitiva en Octubre de 2014. Esta última versión del lenguaje establece una serie de nuevos elementos, atributos y funcionalidades mediante API's que permiten a las aplicaciones web adaptarse mejor a las nuevas funcionalidades y posibilidades tecnológicas. Algunas de estas novedades son las siguientes:

- Web Semántica con la introducción de etiquetas como *header*, *footer*, *nav* que, además de estructurar el contenido, permiten dar significado a su contenido. Esta característica es principalmente útil para los buscadores web, que les permite realizar una búsqueda más eficiente dentro de la página.

- Nuevas etiquetas para mejorar el soporte de contenido multimedia, como las nuevas etiquetas *audio* o *video* y añadido soporte para canvas, tanto 2D y 3D, que permite la creación de gráficos mediante scripting.

- Introducida una API para el trabajo *offline* que permite a las aplicaciones web descargar la información necesaria y funcionar sin conectividad continua con el servidor.

- Nueva API de Geolocalización para dispositivos que lo soporten.

- API Storage que engloba varias herramientas que facilitan el almacenamiento de información persistente de manera local en el navegador web. Algunas de estas herramientas son una base de datos basada en SQLite, o funciones como *localstorage* y *sessionstorage* que permiten guardar información en formato clave-valor ligados a un dominio web de manera más eficiente que mediante *Cookies*.

- Soporte para Web Sockets, una tecnología que permite la interacción bidireccional cliente-servidor a través de un canal *fullduplex*. De esta manera, el servidor puede tener la iniciativa en la comunicación, eliminando la restricción de que el servidor sólo podía comunicarse con el cliente como respuesta a una petición previa de éste.

- Introducción de Web Workers, hilos de ejecución en segundo plano que permiten realizar tareas con tiempos de ejecución largos sin bloquear la interfaz de usuario. Esta funcionalidad en concreto permite el desarrollo de aplicaciones web más complejas que las que se podían crear hasta ahora.

- Web RTC como API que permite la realización de llamadas de voz o videollamadas y compartir archivos P2P a través de la web.

Además de estas nuevas funcionalidades, HTML5 también ha introducido una serie de mejoras en las elementos ya existentes en el estándar, como en los formularios, añadiendo la API de validación de restricción o con nuevos valores para el campo *input*.

3.2. Hojas de estilo CSS

Hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Stylesheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado, típicamente HTML, aunque también otros lenguajes como XML o XHTML. Fue creado por Håkon Wium Lie en 1994 aunque hasta 1996 no fue estandarizado por el W3C mediante la recomendación *CSS1*. Actualmente W3C mantiene la tercera recomendación de CSS, denominado *CSS3*, la cuál ha sido utilizada en el desarrollo de este TFG.

CSS está diseñado para marcar la separación entre el contenido de un documento y la forma en la que se presentará éste (colocación, colores, etc de cada uno de los elementos). Esta característica provee al documento de una mejor accesibilidad y de una gran flexibilidad. Algunas de las ventajas de esta separación son:

- Permite que varios documentos HTML tenga un mismo estilo usando una única hoja de estilos CSS, reduciendo la complejidad y la repetición de código lo que ayuda en gran medida a facilitar el mantenimiento de éste.
- Es posible presentar un mismo documento HTML de manera distinta solo con la utilización de distintas hojas CSS. De esta manera se facilita la adaptación del documento HTML a distintos métodos de renderizado, tipos de dispositivos, tamaños de pantalla, etc.

CSS funciona a través de reglas CSS, típicamente definidas en un hoja de estilos .css que se asocian al documento HTML, aunque también pueden ir directamente dentro de este documento. Estas reglas están compuestas por un selector y un bloque de declaraciones. Los selectores determinan a qué etiquetas se aplicarán los estilos que son definidos, mediante declaraciones CSS, en los bloques de declaración. Estos selectores pueden aplicarse a todas las etiquetas de un mismo tipo, o solo a aquellas etiquetas con un atributo en particular, como por ejemplo un id o una clase HTML concretos. Además, estos selectores se pueden combinar entre sí de muchas maneras para obtener una gran flexibilidad y permitir una gran precisión a la hora de especificar el elemento HTML al que se quiere aplicar el estilo. Asociados a estos selectores hay una o más declaraciones CSS separadas entre sí por *;* . Estas declaraciones siguen una estructura de propiedad-valor, separadas mediante el carácter *:*, por la cual se definen la característica sobre la cual se aplica la declaración y el valor que tomará esta característica en el elemento HTML a la que son aplicados. En el fragmento 3.1 hay un ejemplo de una regla CSS.

```
p {

  background-color: red;
  background-style: none;

}
```

Fragmento 3.1: Ejemplo de una regla CSS

CSS establece un mecanismo para determinar qué regla se debe aplicar a un elemento cuando más de una regla intenta aplicar estilos a él. Este mecanismo se basa en la especificidad de cada regla CSS. De esta manera, al elemento HTML se aplicará aquellas de las reglas menos genérica y que por lo tanto, cuyo selector se más especifico para el elemento.

Otra de las características principales de CSS es la Herencia, por la cuál algunas de las propiedades no se aplican solamente a un elemento, si no que también se aplicarán a los descendientes de este. Esta relación de ancestro-descendiente se extrae de la propia estructura del documento HTML, permitiendo que los elementos descendientes pueden heredar los valores de las propiedades CSS de un elemento ancestro. Las propiedades que se heredan son aquellas relacionadas relacionadas con el texto (color, fuente, espaciado, el peso de la línea, propiedades de lista, alineación del texto, indentado, visibilidad, espaciado de espacios and y espaciado entre palabras) y aquellas que no pueden ser heredadas son aquellas relacionadas con la caja (fondo, bordes, visualización, *float*, tamaño, márgenes, tamaño mínimo y máximo, *outline*, desbordamiento, *padding*, posición, alineación vertical y z-index).

Esta característica de CSS permite evitar que algunas propiedades sean escritas múltiples veces, ayudando a la simplicidad y reduciendo la cantidad de código necesario. Además reduce los tiempos de carga por parte de los clientes al minimizar el tamaño de los archivos .css.

3.3. Lenguaje JavaScript (ECMAScript6)

Comúnmente abreviado como JS, es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Fue desarrollado originalmente por Brendan Eich de Netscape y su primera versión fue estandarizada en 1997.

Fue diseñado tomando características de lenguajes de programación ya existentes, como una sintaxis similar a C y convenciones del lenguaje del lenguaje Java, aunque en el fondo, es completamente distinto a ellos, como su semántica y su propósito.

Inicialmente fue concebido como lenguaje para el lado cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque con el paso de los años JavaScript se ha ido abriendo camino también en el lado servidor. Actualmente entornos de ejecución para la capa de servidor muy populares como NodeJS hacen uso de JS para su programación.

En la actualidad está en vigor el estándar ECMAScript 6 que fue publicado en junio de 2015 que ha dotado al lenguaje de de características avanzadas como la implementación de clases para la programación orientada a objetos, expresiones flecha, iteradores, promesas y etc... Actualmente JavaScript es uno de los lenguajes más populares y su crecimiento no deja de aumentar. Gran parte de este crecimiento se debe, además de las mejoras realizadas en el lenguaje que hemos descrito anteriormente, a la mejora exponencial de rendimiento que ha experimentado el interprete de JavaScript que permite a los navegadores ejecutar de manera mucho más rápida y eficiente el código JavaScript. Este crecimiento lo podemos apreciar en el gráfico 1.5.

Concretamente el uso de JavaScript en clientes web es el de escribir funciones embebidas en documentos HTML y que interactúan con el DOM (Document Object Model) de este. Algunos ejemplos de las funcionalidades que puede añadir al documento HTML son:

- Carga de nuevo contenido desde el servidor sin necesidad de recargar la página mediante AJAX, el cual veremos detalle en la sección 3.7.1
- Añadir animaciones a elementos documento HTML
- Validación del contenido de formularios
- Inclusión de contenido interactivo como juegos o contenido multimedia

En definitiva, lo que permite JavaScript es dotar a la web de la capacidad de interactuar de manera rápida y efectiva con el usuario haciéndola mucho más atractiva para él.

3.4. Document Object Model

Document Objetc Model ("Modelo de Objetos del Documento"), también conocido por sus siglas DOM, es un conjunto de utilidades específicamente diseñadas para manipular documentos XML y por extensión documentos HTML y XHTML. Técnicamente, DOM es una API de funciones que se pueden utilizar para manipular estos documentos, pudiendo modificar su contenido, estructura y estilo, de una manera rápida y eficiente. Más concretamente DOM proporciona tres elementos: (a) un conjunto estándar de objetos para representar documentos

HTML, XHTML y XML, (b) un modelo estándar sobre cómo pueden combinarse dichos objetos y (c) una interfaz estándar para acceder a ellos y manipularlos. En definitiva el DOM permite el acceso dinámico a través de programación al contenido de una página web pudiendo modificarlo fácilmente. Es responsabilidad del World Wid Web Consortium (W3C) que publicó la primera recomendación *DOM Level 1* en 1998. Actualmente existen cuatro revisiones sobre este estándar.

El funcionamiento de DOM se basa en transformar el documento XML original en una estructura más fácil de manejar formada por una jerarquía de nodos interconectados entre sí. Esta estructura se denomina *Árbol DOM* debido a su topología. Este árbol generado no sólo representa el contenido del documento si no también la relaciones existentes entre los elementos que lo forman mediante las ramas"del árbol.

Una vez formado este árbol DOM se puede acceder a él desde prácticamente cualquier lenguaje de programación ya que la API de DOM esta disponible para la mayoría de lenguajes de programación actuales. Aunque se pueda acceder a él mediante casi cualquier lenguaje de programación, el lenguaje por excelencia para la manipulación del árbol DOM es JavaScript ya que está prácticamente estandarizado como lenguaje de programación del lado cliente dentro de las tecnologías web. Esta relación es tan íntima hasta el punto en el que sin DOM JavaScripit no tiene ningún modelo o noción de la estructura ni del contenido de las páginas web. Esto se debe a que inicialmente ambas partes se concibieron como una única entidad, pero desarrollos posteriores han separado ambas partes y han hecho al DOM independiente de cualquier lenguaje de programación. En cualquier caso, en este Trabajo de Fin de Grado solamente se ha utilizado JavaScript (o librerías de este lenguaje como jQuery) para el acceso a árbol DOM.

DOM clasifica los nodos que componen el árbol DOM en distintos tipos. Uno de los más importantes el tipo *Document* al cual pertenece siempre el nodo raíz del documento. Para los nodos que forman el árbol DOM y que representan los elementos que componen el documento XML se utilizan diversas funciones que proporciona DOM. Estas funciones, como *getElementsByTagName()* o *getElementById()*, permiten el acceso a uno o más elementos que cumplan determinados parámetros, como pertenecer a una determinada etiqueta o tener un identificador concreto. Una vez elegido el elemento adecuado, DOM proporciona multitud de métodos que permiten modificar o eliminar estos nodos o crear nuevos. De esta manera, haciendo uso de la multitud de funciones que ofrece DOM se puede modificar, de una manera eficiente y sencilla, el árbol DOM y por extensión, la página web en cuestión.

3.5. Entorno Bootstrap:

Bootstrap [1]es un entorno de desarrollo para el diseño de sitios y aplicaciones web que contiene multitud de elementos de diseño basados en HTML y CSS así como extensiones opcionales basadas en JavaScript. Este entorno fue desarrollado por Mark Otto y Jacbod Thornton de Twitter inicalmente como una herramienta de trabajo interna. En 2011 la empresa Twitter libera la herramienta como código abierto y en apenas unos meses se convierte en el proyecto de desarrollo más popular en GitHub. Actualmente es compatible con la gran mayoría de navegadores web modernos. En este proyecto se ha utilizado en su versión número 3.

Boostrap es modular y se basa principalmente en hojas de estilo LESS (lenguaje de estilo que añade dinamismo a CSS) que implementa una gran cantidad de componentes, los cuales pueden

[1] http://getbootstrap.com/

ser personalizados por los desarrolladores adaptando el mismo archivo que utiliza Bootstrap. Esta personalización se puede llevar a cabo desde la propia documentación de Bootstrap que permite generar el paquete consecuentemente generado con la hoja de estilos CSS ya pre-compilada.

Algunos de los elementos que contiene Bootstrap son: tipografías, formularios, botones, cuadros, menús de navegación, iconos, *breadcrums*, mensajes de alerta, menús desplegables, diálogos modales, menús de pestañas y un largo etcétera.

Una de las características fundamentales de Bootstrap es su enfoque *"Mobile First"*, por la cuál está orientado a la creación de interfaces de usuario responsivas que se adapten adecuadamente a distintos tipos de dispositivos y de tamaños de pantalla. Para ello, Bootstrap cuenta con un sistema de rejilla dividida en doce columnas en las que se divide el ancho total de la pantalla y que permite al desarrollador establecer el número de columnas, y por tanto, el tamaño que ocupará cada elemento en función del tamaño de la pantalla. Se definen cuatro de tamaños de pantalla (xs, sm, md y lg) por lo que el desarrollador podrá definir hasta cuatro configuraciones distintas de tamaño para cada elemento en función del tamaño de la pantalla. De esta manera se consigue una gran flexibilidad para adaptar el sitio web la gran variedad de dispositivos que existe actualmente como móviles y tabletas de distintos formatos, ordenadores, etc...

Otra característica importante que añade este entorno son los *plugins* de JavaScript que incluye. Estos elementos están basados en la librería de JS jQuery y que ofrecen al desarrollador herramientas muy interesantes como diálogos, *tooltips* y carruseles. Además de estos elementos propios, también introduce mejoras en las funcionalidades de algunos elementos ya existentes como la función de autocompletar de formularios.

En concreto en este proyecto Bootstrap se ha utilizado como entorno para el diseño de la interfaz de usuario. Se ha utilizado como base del aspecto gráfico de la aplicación web ya que se han utilizado muchos de los componentes que ofrece este entorno como los botones, paneles, menús, etc... que posteriormente se han adaptado al diseño final de la aplicación. Además, se ha utilizado sus sistema de rejilla para que la interfaz de usuario se adapte correctamente a distintos tamaños de pantalla y distintos tipos de dispositivos.

3.6. Biblioteca jQuery

jQuery [2] es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, cuyo su principal objetivo es el de simplificar la interacción con los documentos HTML, manipular su árbol DOM y otros cometidos. Fue presentado en 2006 y es la librería JS más utilizada. Es software libre y código abierto.

jQuery ofrece una gran cantidad de características mediante funciones propias, lo que permite simplificar y facilitar la programación de estas funcionalidades. Algunas de estas características son el manejo de eventos, efectos y animaciones, manipulación de hojas de estilo CSS, la interacción con el servidor a través de AJAX, etc.

En concreto, para este TFG la librería jQuery se ha utilizado en su versión 3.1.0 y su principal fin ha sido el de facilitar la interacción con el árbol DOM como parte del código JavaScript de la aplicación web.

[2] https://jquery.com/

3.7. Interacción

3.7.1. AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente y mantienen comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas completamente, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

En la figura 3.1 vemos por un lado como existe una comunicación clásica entre cliente y servidor siguiendo el eje horizontal de tiempos. Estas peticiones tienen un tiempo de solicitud y respuesta donde no puede haber consultas entre medias. Por otro lado, en la parte inferior, vemos cómo, además de la petición estándar HTTP existen otras peticiones asíncronas con las iniciales y que puede haber respuesta antes de que llegue la del servidor.

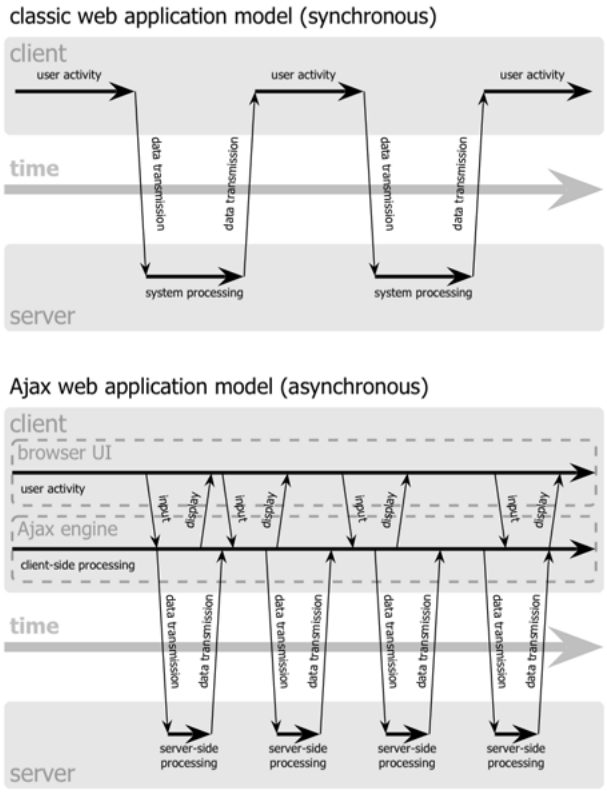


Figura 3.1: Diagrama de AJAX

Para el caso del TFG en particular, se emplea AJAX como validación de campos en los formularios que se van a rellenar mediante una consulta con la base de datos a través del servidor, informando al usuario si el campo relleno es o no válido antes de que se produzca el envío.

3.7.2. WebSockets

WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor. Debido a que

las conexiones TCP comunes sobre puertos diferentes al 80 son habitualmente bloqueadas por los administradores de redes, el uso de esta tecnología proporcionaría una solución a este tipo de limitaciones proveyendo una funcionalidad similar a la apertura de varias conexiones en distintos puertos. Este protocolo fue normalizado por la IETF (Grupo de Trabajo de Ingeniería de Internet) en 2011 y el W3C está actualmente en proceso de crear un estándar de la API de Web Socket.

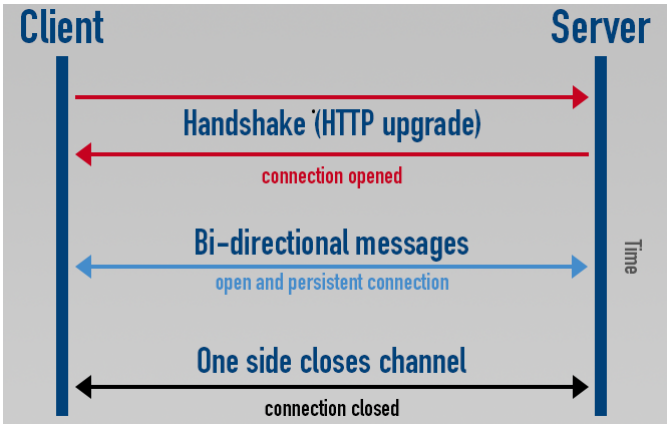


Figura 3.2: Diagrama WebSockets

En primer lugar el protocolo hace un estrechamiento de manos para abrir la conexión para dejar paso, a continuación, de la comunicación bi-direccional dejando una conexión persistente hasta que uno de los dos lados de la comunicación cierre la sesión. La especificación del protocolo WebSocket define dos nuevos esquemas de URI, ws: y wss: para conexiones no cifradas y cifradas.

Para este TFG, el uso de WebSocket se ha empleado para gestionar por un puerto paralelo al del servidor el uso de notificaciones para el cliente. Esta consulta con la base de datos es asíncrona y establecida en intervalos de tiempo desde el servidor. Si la condición se cumple, el servidor tomará la iniciativa y enviará al cliente una alerta que será reflejada por el usuario en forma de alerta o '`pop up`'.

Capítulo 4

Diseño y uso del sitio web

En este capítulo se dará una visión global del funcionamiento de la aplicación. Se analizará la estructura de secciones y el manejo del interfaz de usuario así como de las principales funcionalidades de esta. Posteriormente en el capítulo 5 explicarán los detalles de la implementación de estos elementos.

4.1. Diseño del sitio web

La estructura de la aplicación se puede dividir en cuatro grandes secciones atendiendo a su contenido:

- Proyectos
- Maquinaria
- Personal
- Administración

Cada una de estas secciones tendrá detrás su correspondiente base de datos en el lado servidor de la aplicación web. Estas partes contienen las distintas acciones básicas que se pueden realizar sobre la base de datos, como son añadir, consultar, editar o eliminar un elemento de cada base de datos. La figura 4.1 contiene un diagrama donde se puede ver las páginas más importantes que conforman la aplicación.

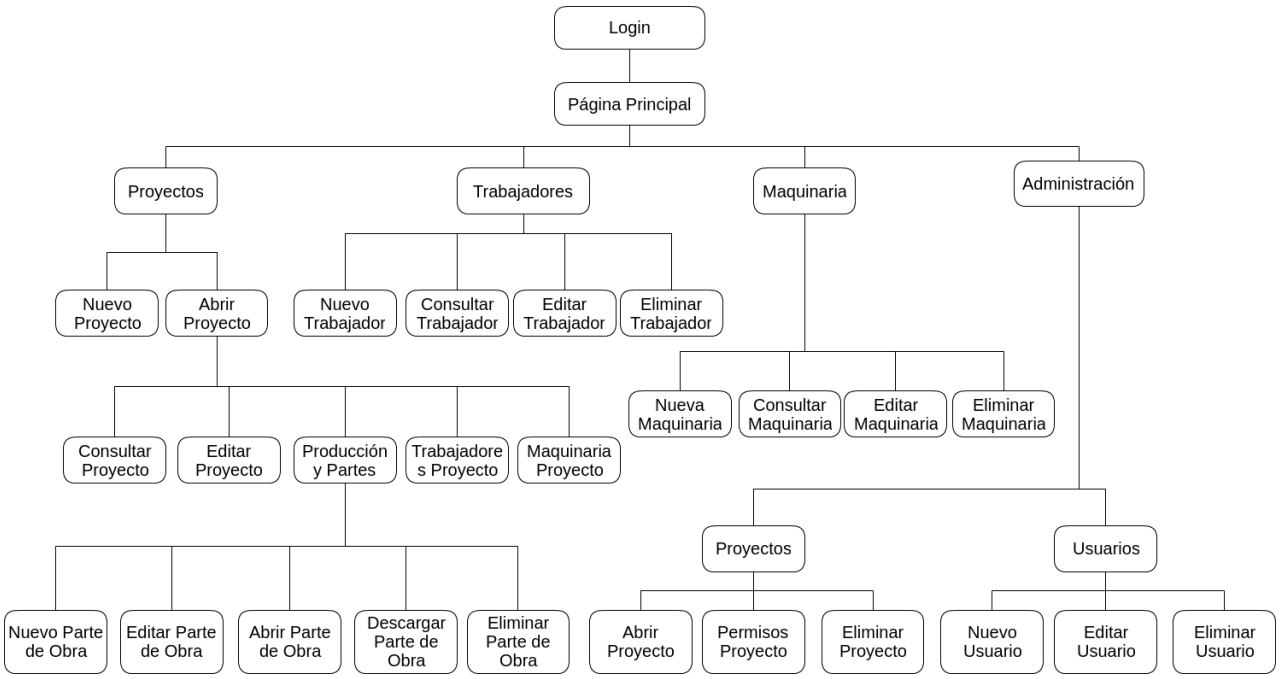


Figura 4.1: Estructura de páginas de la aplicación web

Para la creación de las páginas que conforman estas secciones se ha utilizado HTML5 para la estructura, CSS y en concreto el entorno Bootstrap para su diseño gráfico, el lenguaje de programación JavaScript para añadir diversas funcionalidades además de diversas librerías menores para usos concretos.

Para el diseño visual de la aplicación se ha tenido muy en cuenta el propósito de la creación de una interfaz simple cuyo uso sea lo más sencillo y cómodo posible.

Una de las principales características de la interfaz de usuarios es que esta se adapta a la clasificación por roles de los usuarios. La aplicación implementa un sistema de roles de usuarios con el fin de controlar el acceso al contenido de la aplicación. Este sistema consta de tres roles: Control, Jefe de Obra y Piloto.

Los usuarios de tipo control tienen acceso total a la aplicación y tiene la capacidad crear, consultar, editar o eliminar todos los elementos de todas las bases de datos. El rol Jefe de Obra solamente tiene acceso a los proyectos a los cuales esta autorizado y a consultar las bases de datos de maquinaria y de empleados. Por último los usuarios con rol de Pilotos solo tiene acceso a la creación de Partes de Obra para proyectos en los cuales esta asignado. El interfaz de usuario se adapta a estas restricciones ajustándose a cada uno de los roles, ocultando aquel contenido al que el usuario no tiene acceso y mostrando únicamente aquellas secciones para las que si tiene acceso. Este sistema simplifica la interfaz facilita su manejo a los usuarios.

A continuación se realizará un análisis exhaustivo de las todas las páginas que componen la aplicación analizando tanto su estructura como sus funcionalidades más importantes.

4.2. Plantilla de las páginas

Todas las páginas de la aplicación siguen una estructura básica compuesta por cuatro elementos principales.

- Barra de Navegación o Navbar:

Su principal cometido es además de servir de encabezado de la página es el de permitir al usuario navegar fácilmente entre las distintas secciones de aplicación. Tiene dos versiones, una versión *no-collapse* para dispositivos con un tamaño de pantalla grande que se puede ver en la figura 4.2.

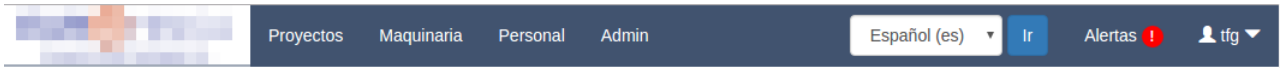


Figura 4.2: Barra de navegación de la aplicación

Y una versión *collapse* para dispositivos con pantallas medianas y pequeñas (principalmente dispositivos móviles) en la cual las opciones quedan "escondidas" y que son mostradas mediante un desplegable al pulsar el botón de la derecha (Figura 4.3).

Figura 4.3: Barra de navegación de la aplicación en formato *collapse*

El desplegable se muestra en la figura 4.4.

Figura 4.4: Desplegable de la barra de navegación en formato *collapse*

Ambas versiones se componen de:

- Accesos a las cuatros secciones principales de la aplicación, Proyectos, Maquinaria, Personal y Administración.
- Selector de idioma que permite al usuario elegir el idioma en el que se mostrará el interfaz y el botón para confirmar la elección.
- Indicador sobre si existen alertas actualmente y acceso a las de notificaciones.
- Sección con el nombre del actual Usuario y con un botón que le da la posibilidad al usuario de hacer Logout y salir de la aplicación.

Esta barra de navegación da la posibilidad al usuario de moverse por las grandes secciones de la aplicación de una manera rápida y moverse directamente a ellas desde cualquiera otra

página.

- Breadcrumbs:

Este elemento, ampliamente utilizado en las páginas web actuales, permite al usuario navegar de forma rápida por las páginas "padres"de la página actual. Consiste en una serie enlaces a las páginas superiores a la página actual en orden descendente, de esa manera el usuario puede "saltar"hacia arriba uno o más niveles rápidamente facilitando la navegación dentro de la aplicación. Cabe destacar que este elemento no está en todas las páginas por razones de diseño. En la figura 4.5 se puede ver un ejemplo de este elemento.

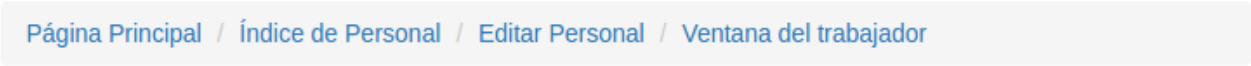


Figura 4.5: Breadcrums de la aplicación web

- Contenido de la página:

Es la parte principal de la página y en ella se encuentra el contenido de cada una de ellas. Esta sección cambiará su estructura en función de cada página concreta y se ajustará a su contenido. Más adelante se mostrarán algunos ejemplos.

- Footer:

Sirve como pie de página en el cual figura la versión actual de la aplicación y un botón que da acceso al formulario que permite el envío de comentarios para el desarrollo de la aplicación (Figura 4.6).

Figura 4.6: Pie de página de la aplicación web

4.3. Página Principal

Al entrar por primera vez en la aplicación nos entramos con la página de Login, la cual nos pide nuestro Nombre de Usuario y Contraseña como credenciales para acceder. Esta página de Inicio de Sesión la podemos ver en la figura 4.7.

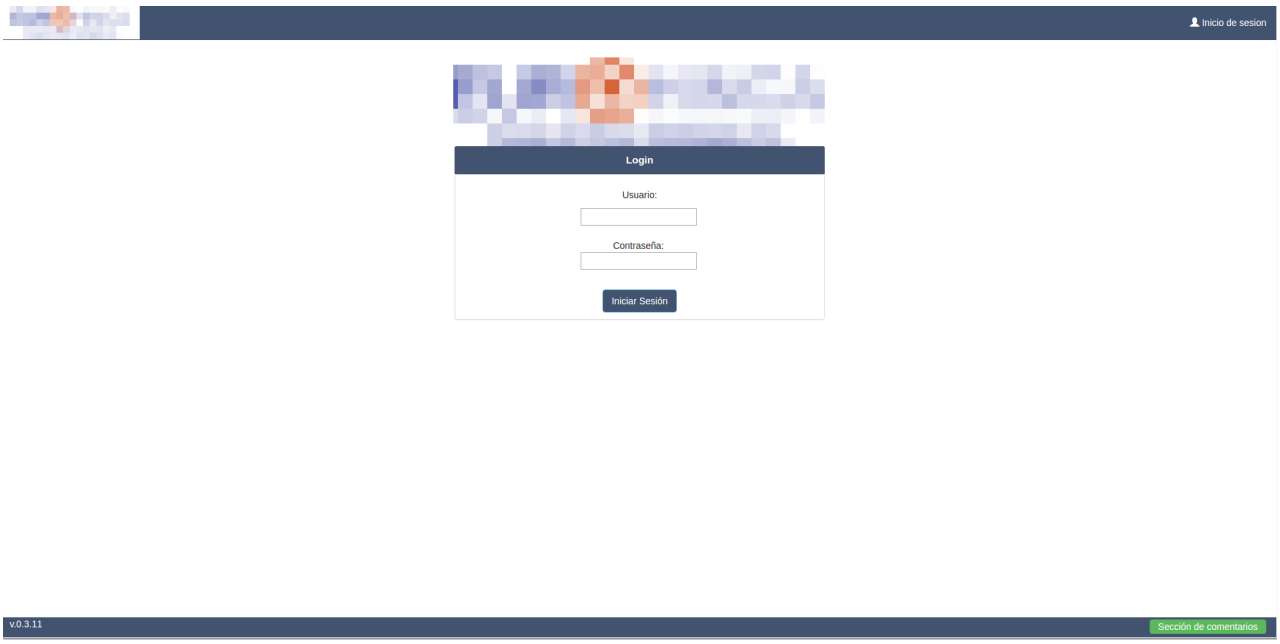


Figura 4.7: Página de Inicio de Sesión

Una vez identificados correctamente, entraremos en la página principal de la aplicación (Figura 4.8). Esta página tiene como contenido principal, una parrilla de imágenes que sirven de acceso a las secciones de la aplicación.

Figura 4.8: Página Principal de la aplicación

A la página de notificaciones se accede a través del indicador *Alertas* situado en la barra de navegaciones. Este sistema de notificaciones se ha desarrollado como petición expresa de algunos de los usuarios de la aplicación y tiene como cometido avisar a los usuarios con el rol *Control* que existen uno o más trabajadores cuyo certificado médico está cerca de caducar.

El usuario es avisado mediante el indicador de alertas citado anteriormente y mediante un sistema de *pop-ups* emergentes que son lanzados de manera periódica. Estos *pop-ups* son reemplazados por el lanzamiento de una notificación en sistemas móviles. En las figuras 4.9

y 4.10 podemos ver ejemplos de estos elementos.

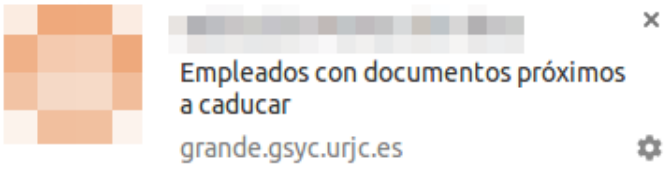


Figura 4.9: Notificación Pop-Up en versión de Escritorio

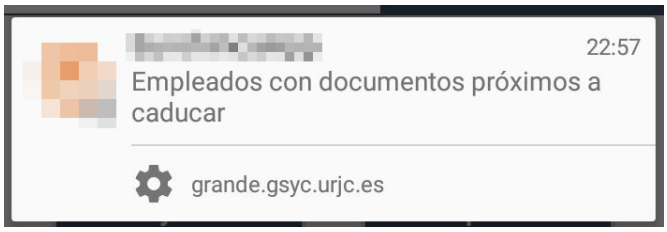


Figura 4.10: Notificación Pop-Up para Dispositivos Móviles

Además si el usuario pulsa en el cualquiera de estos elementos, será redirigido a un listado con todos los empleados cuyo certificado médico esté próximo a caducar (fecha de caducidad inferior a 15 días). Esta página se muestra en la figura 4.11.

Lista de Trabajadores con apto médico caducado

ID	Nombre	Estado	Oficio	Acciones
55	José Rodríguez Alonso	Caducado	Piloto	Editar
56	Miguel Díaz López	Caducado	Capataz	Editar
57	Daniel Gómez Rodríguez	Caducado	Electricista	Editar
58	Ignacio Fernández López	Caducado	Oficial de 1ª	Editar
59	Antonio Sánchez Torres	Caducado	Mecánico	Editar
60	Manuel Ruíz Ramírez	Caducado	Operario de Planta	Editar
61	Carlos Suárez Herrera	Caducado	Soldador	Editar
62	Javier Castro Silva	Caducado	Electricista	Editar

Figura 4.11: Página de Notificaciones

4.4. Sección Proyectos

Es la parte central de la aplicación, contiene toda la información relativa a cada uno de los proyectos de la empresa y, sobre todo, permite crear, consultar, editar o eliminar los Partes de Obra asociados a estos proyectos, el cual es el objetivo básico del trabajo de fin de grado. Al entrar en esta sección nos encontraremos con la página de la figura 4.12.

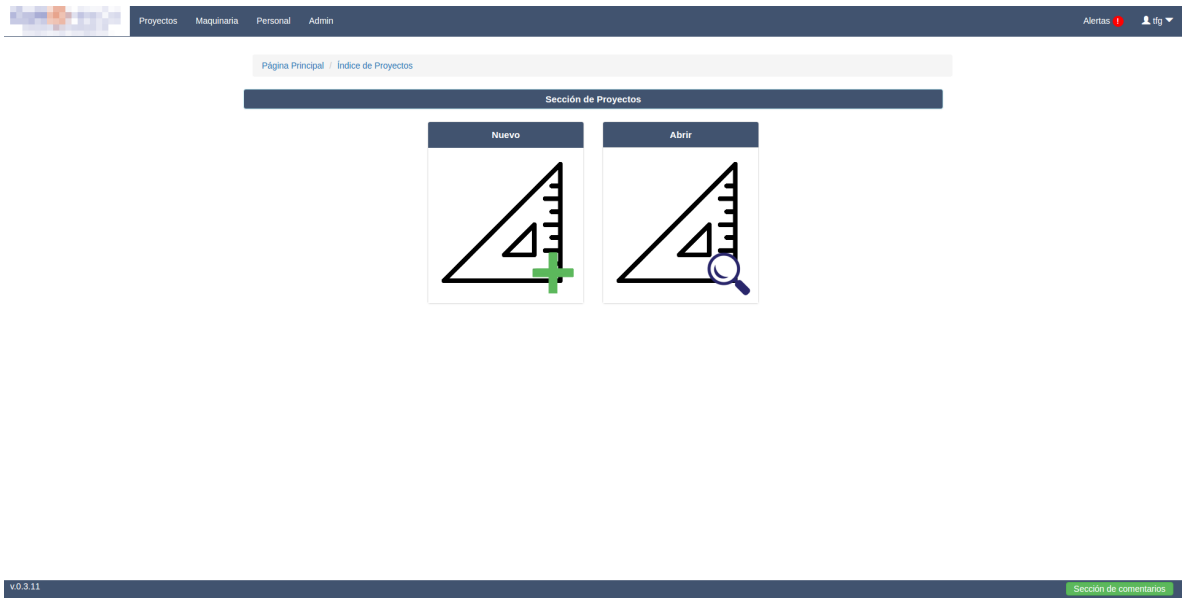


Figura 4.12: Página Principal de la sección Proyectos

Podemos ver que se ofrece el acceso a dos secciones, Nuevo Proyecto y Abrir Proyectos. Puesto que todos los partes de obra deben estar asociados al proyecto al que pertenecen, comenzaremos con la ventana que nos permite la creación de un nuevo Proyecto.

4.4.1. Nuevo Proyecto

Esta página está compuesta por un formulario con los datos necesarios. Lo podemos dividir en tres partes (Figuras 4.13, 4.15 y 4.16):

-Información básica del proyecto: (Figura 4.13)

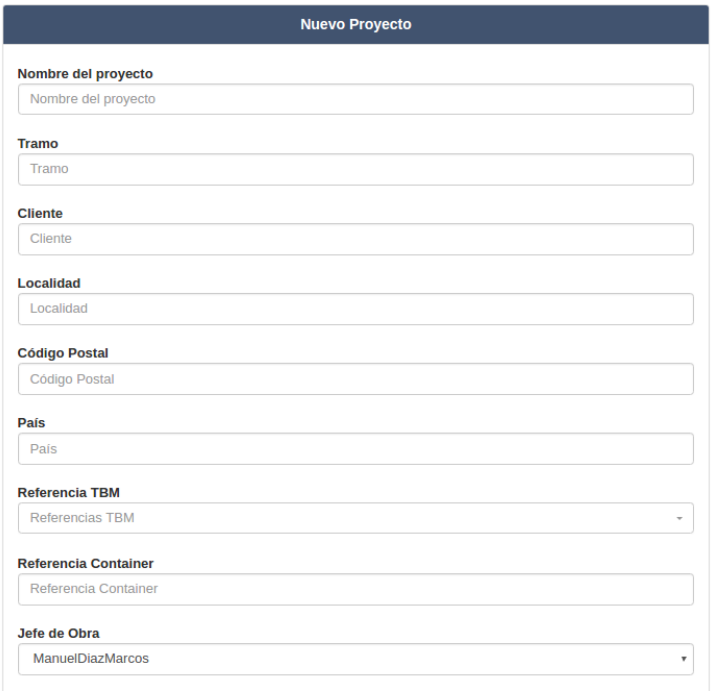


Figura 4.13: Formulario de la Información Básica para crear un nuevo proyecto

De esta sección cabe destacar tres campos:

- Referencia TBM: Hace referencia a la (o las tuneladora/s) que será asignada a este proyecto. Es la máquina central del proyecto, por lo que se ha diseñado un campo separado del resto de maquinaria auxiliar para su selección. Este campo debe completarse con, al menos, la referencia de una máquina incluida en la base de datos de Maquinaria. Para facilitar su introducción, al seleccionar el campo, se despliega una lista con una relación de las referencias de todas las máquinas identificadas con el tipo TBM (*Tunnel Boring Machine*) de la base de datos. Además, tiene un buscador que permite realizar un filtrado entre estas opciones.

También cabe destacar, que por necesidades de diseño es posible realizar una selección múltiple de estas opciones. Para ello se ha utilizado la librería Bootstrap-Select. En la figura 4.14 podemos ver un ejemplo.

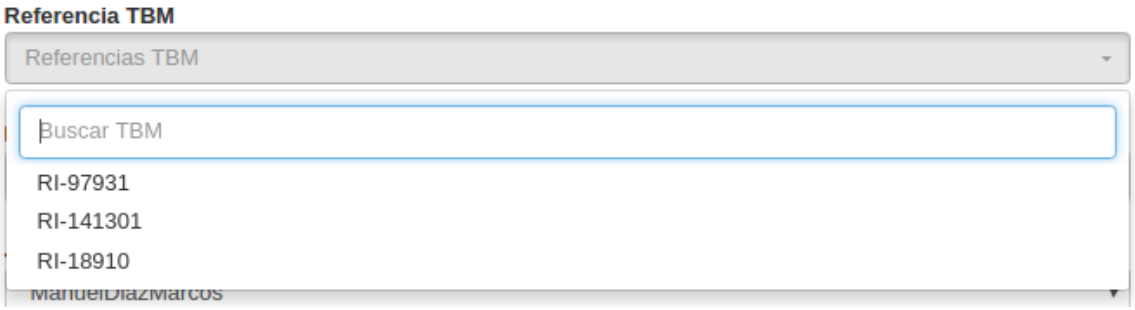


Figura 4.14: Campo Referencia TBM del formulario Proyecto Nuevo

-Trabajadores del Proyecto: En esta sección se escogerán de entre todos los trabajadores que constan en la aplicación aquellos que vayan a trabajar en este proyecto. Para ello se ha utilizado la librería Bootstrap Dual ListBox que genera el campo que podemos ver en la figura 4.15.

Figura 4.15: Campo Trabajadores del Proyecto del formulario Proyecto Nuevo

En el cuadro izquierdo están todos los trabajadores que constan en la base de datos de Personal junto con su oficio. En el cuadro de la derecha se colocarán aquellos empleados que se asignarán al proyecto nuevo. Además para facilitar la elección de los empleados, se dispone de sendos buscadores que permiten filtrar, tanto por oficio, como por nombre del trabajador las opciones disponibles. Además está presente el los botones de Seleccionar Todos y Deseleccionar Todos para un mayor comodidad.

-Maquinaria Auxiliar: En este campo se asignará al proyecto toda aquella maquinaria auxiliar, es decir, toda aquella maquinaria que será utilizada en el proyecto excluyendo las tuneladoras (TBM), las cuales serán asignadas en el campo Referencia TBM visto anteriormente. Este campo de Maquinaria Auxiliar sigue un esquema similar al visto en el campo anterior, ya que también ha sido utilizada la librería Bootstrap Dual ListBox. El resultado se puede observar en la figura 4.16.

Figura 4.16: Campo Maquinaria del Proyecto del formulario Proyecto Nuevo

Análogamente al campo anterior, en el cuadro de la izquierda quedarán toda la maquinaria de la aplicación, con su tipo y su referencia, y a la derecha aquellas maquinarias que serán asignadas al proyecto nuevo, también con su tipo y su referencia. También se dispone de sendos buscadores para facilitar el filtrado entre las opciones y los botones de Seleccionar Todos y Deseleccionar Todos

El resto de campos de esta parte pasan por un proceso de validación que comprueba ciertos requisitos específicos de cada campo como que no esté usado ya el Nombre del Proyecto, o que el Código Postal siga el esquema necesario. Además, este proceso comprueba que estos campos no queden vacíos. Si alguna de estas condiciones no se cumpliera para alguno de los campos, el botón de Crear Proyecto quedaría deshabilitado por lo que no se podría mandar el formulario.

Una vez finalizado el rellenado de todos los campos, pulsaremos el botón de Crear Proyecto y se procederá al envío de datos al servidor. Como respuesta, seremos redirigidos a la página principal del proyecto donde encontraremos las distintas acciones y secciones de éste.

4.4.2. Abrir Proyecto

Si elegimos la opción de Abrir Proyecto se nos dará acceso a un índice con un listado de todos los proyectos de la aplicación a los que tengamos acceso gracias a nuestro Rol dentro de la aplicación o a los permisos específicos que se tengan concedidos. De esta manera, los usuarios con el Rol de Control tendrán acceso a todos los proyectos, pero los usuarios con el Rol de Jefe de Obra y Piloto sólo tendrán acceso a aquellos Proyectos a los que hayan sido asignados. En esta página consta de los datos básicos como son el nombre, el tramo y la localidad, y un botón que nos permite el acceso a la sección propia de cada uno de ellos. Esta página esta presente en la figura 4.17.

Figura 4.17: Listado de Proyectos de la aplicación

- Opciones del proyecto:

Al pulsar en el botón Opciones de un proyecto, entraremos a la página en la que se listan las distintas secciones y acciones de cada uno de ellos (Figura 4.18). Desde esta página podremos acceder al área de Producción y Partes, en el que se encuentran todas las opciones referentes a los Partes de Obra de este proyecto; a Consultar Proyecto, donde se realiza un resumen de toda la información de este; Editar Proyecto, que nos permitirán realizar cambios sobre ellos; y las secciones Trabajadores y Maquinaria en la que se encuentra un listado sobre cada uno de estos apartados. Además, está el botón Salir del Proyecto que permite volver a la ventana anterior.

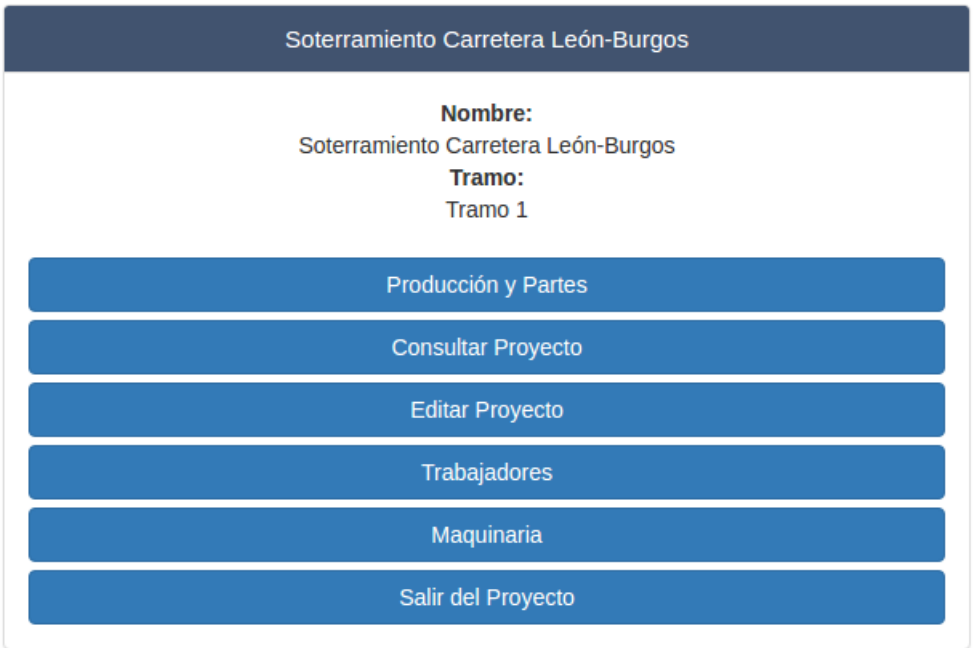
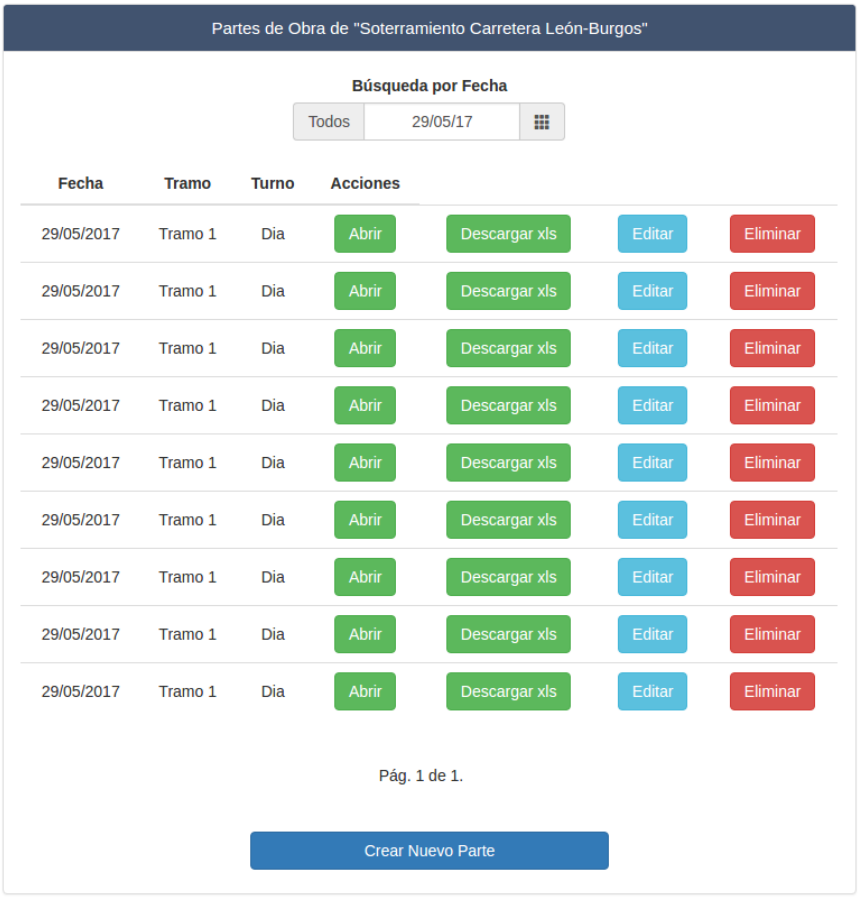


Figura 4.18: Página de Opciones del Proyecto

4.4.3. Producción y Partes de Obra

Esta es la sección central de la aplicación, ya que es la que nos permite crear, consultar, editar, descargar y eliminar los partes de obra de cada proyecto.

Esta se compone de un listado con todos los partes de obra pertenecientes al proyecto, ordenados mediante un sistema de paginación, que va ofreciendo los partes en grupos de seis elementos ordenados por su fecha, desde los más recientes a los mas antiguos. Además dispone de un buscador que permite filtrar los partes de obra por la fecha seleccionada y de un botón permite la creación de un nuevo parte de obra. Este listado se muestra en la figura 4.19.



Partes de Obra de "Soterramiento Carretera León-Burgos"

Búsqueda por Fecha

Fecha	Tramo	Turno	Acciones			
29/05/2017	Tramo 1	Dia	Abrir	Descargar xls	Editar	Eliminar
29/05/2017	Tramo 1	Dia	Abrir	Descargar xls	Editar	Eliminar
29/05/2017	Tramo 1	Dia	Abrir	Descargar xls	Editar	Eliminar
29/05/2017	Tramo 1	Dia	Abrir	Descargar xls	Editar	Eliminar
29/05/2017	Tramo 1	Dia	Abrir	Descargar xls	Editar	Eliminar
29/05/2017	Tramo 1	Dia	Abrir	Descargar xls	Editar	Eliminar
29/05/2017	Tramo 1	Dia	Abrir	Descargar xls	Editar	Eliminar
29/05/2017	Tramo 1	Dia	Abrir	Descargar xls	Editar	Eliminar
29/05/2017	Tramo 1	Dia	Abrir	Descargar xls	Editar	Eliminar

Pág. 1 de 1.

Figura 4.19: Página de Producción y Partes del Proyecto

El buscador, el cual funciona mediante una conexión Ajax con el servidor, permite que se le muestren al usuario sólo aquellos partes del día que elegido. Para facilitar la elección del día se despliega el calendario de la figura 4.20. También se da la opción de, al pulsar en el botón Todos, se vuelven a mostrar todos los partes del proyecto. Para realizar este buscador, además de la conexión Ajax, se ha utilizado la librería Bootstrap-Datepicker para el calendario desplegable:

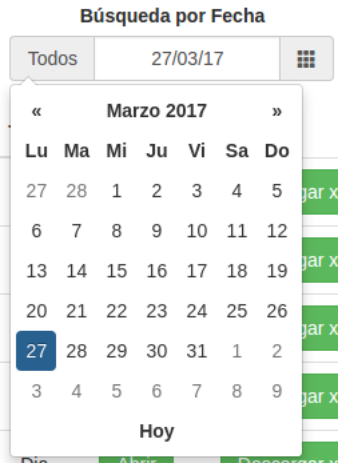


Figura 4.20: Buscador de Partes de Obra por fecha

El listado de Partes de Obra se compone de la información básica de cada uno de los elementos y de botones que dan acceso a las distintas acciones que se puede realizar sobre ellos. Estas acciones son: Abrir, Descargar xls, Editar y Eliminar.

En primer lugar analizaremos las ventana de Parte Nuevo, la cual nos permite la creación de un nuevo parte asociado al proyecto actual.

- Introducción de Parte Nuevo:

Este formulario, al estar compuesto por un gran número de campos, se ha dividido en secciones mediante un sistema de pestañas, que sólo muestra los campos de la sección sobre la cual el usuario esta interactuando, mientras el resto quedan ocultos. Esto nos permite realizar un mejor uso del espacio disponible en la pantalla del dispositivo. Esta división en pestañas se muestra en la figura 4.21.

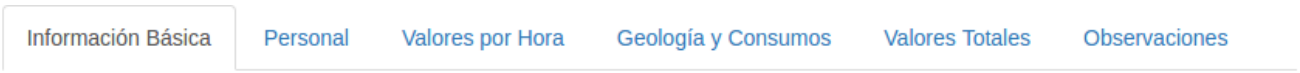


Figura 4.21: Sistema de Pestañas de la página Parte Nuevo

La pestaña de Información Básica (Figura 4.22) contiene los campos del Parte de Obra que lo relacionan con el proyecto al que pertenece, como el nombre de este y el tramo, el piloto de la tuneladora durante el turno de trabajo, la fecha del parte, si pertenece al turno de día o de noche, la tuneladora (TBM) utilizada y la referencia del container usado.

Figura 4.22: Campos de Información Básica de Parte Nuevo

Todos los campos de esta sección vienen rellenados automáticamente con la información del proyecto al que pertenece el parte para facilitar la creación, ya que esta información suele ser común a todos los partes del proyecto. Cabe destacar que el campo Piloto tiene como peculiaridad que estará relleno y bloqueado con el nombre del usuario cuando este tenga el rol Piloto y estará sin rellenar si tiene los roles de Control o Jefe de Obra a los cuales se les ofrecerá un desplegable con los pilotos para que elijan el adecuado.

La pestaña de Personal (Figura 4.23) sirve para seleccionar, de aquellos trabajadores asignados al proyecto, aquellos que han trabajado durante el turno del parte nuevo.

Figura 4.23: Campo Trabajadores de la página Parte Nuevo

Tiene una estructura similar a la parte de Personal de la ventana Proyecto Nuevo visto anteriormente. En la parte izquierda están todos los trabajadores asignados al proyecto y la parte derecha aquellos trabajadores asignados al turno del parte.

La pestaña de Valores por Hora está dedicada a la creación de los distintos tipo intervalos de tiempo que compondrán el Parte de Obra. Estos intervalos se definen como periodos de tiempo durante los cuales, en la obra se han realizado trabajos relacionados con alguno de los cuatro tipos establecidos, los cuales son: Perforación, Bajada de Tubo, Avería y Parada. Además en esta pestaña se definirá la duración del turno al que se refiere el parte de obra.

Para estos cometidos se ha creado un visor que representa la duración de tiempo de cada uno de los intervalos dentro de la duración total del turno. Este elemento se presenta en la figura 4.24.

Figura 4.24: Visor de Intervalos del Parte de Obra

La parte superior consta de dos campos que permiten la selección de la hora inicial y la hora final del turno. Una vez elegidas, se pulsará el botón de Definir Turno lo que reconstruirá el visor del turno actualizándolo a la nueva duración. Por defecto, la duración es de doce horas en turno de día, es decir de 8:00 a 20:00, pero la extensión puede ser menor.

En la figura 4.25 se muestra un ejemplo de como se adapta el visor a una duración del turno distinta:

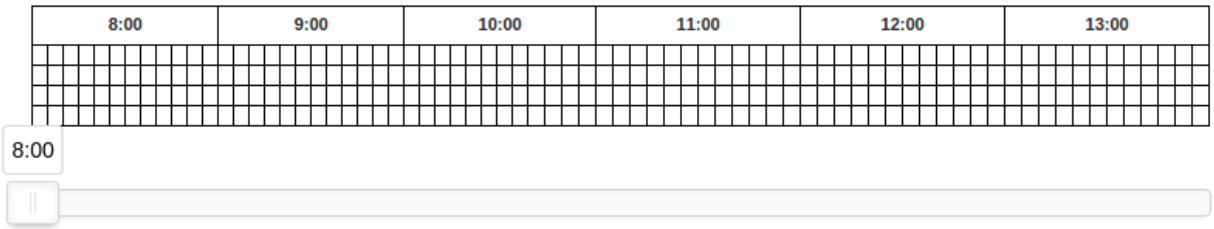


Figura 4.25: Visor de Intervalos con un turno de seis horas

Al intentar definir una nueva duración del turno tras haber creado algún intervalo de tiempo, generará un dialogo de confirmación (Figura 4.26), ya que al realizar esta acción, todos los intervalos de tiempo ya creados serán eliminados.

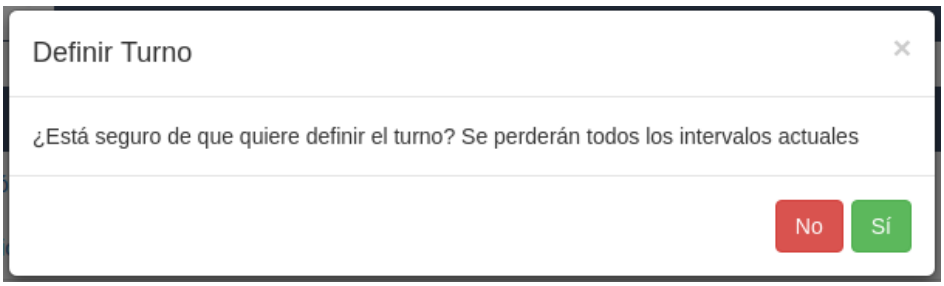


Figura 4.26: Ventana de confirmación para definir un nuevo turno

Para la creación de los intervalos de tiempo se debe definir el tipo de intervalos, y la duración del mismo.

Para el tipo, existe un desplegable que permite elegir entre los cuatro tipos de intervalos: Perforación, Bajada de Tubo, Avería y Parada.La duración se puede establecer mediante dos formas, con el deslizador o con los campos Inicio Intervalo y Final Intervalo.

Mediante el deslizador, podemos deslizar los puntos de inicio y final hasta la horas deseadas, definiendo asía la duración. Para facilitar la elección de la hora, al pulsar sobre el elemento deslizante correspondiente aparecerá un dialogo con la hora que corresponda. Además, el color del deslizador sobre el periodo seleccionado será el que corresponda al tipo de intervalo elegido. Un ejemplo de su funcionamiento se muestra en la figura 4.27.

Figura 4.27: Deslizador del visor de intervalos

La segunda opción es introducción en la hora inicial y final de intervalo en los campos definidos para ello (Figura 4.28).

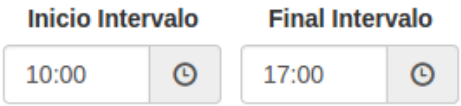


Figura 4.28: Campos Inicio de Intervalo y Final de Intervalo

Una vez elegido el tipo y la duración del intervalo pulsaremos el botón Añadir. Esto creará el nuevo intervalo de tiempo con el tipo y sus campos correspondientes. También está presente el botón Reiniciar que elimina todos los intervalos de tiempo creados hasta el momento.

Al crear un intervalo nuevo se rellenará en el visor el periodo de tiempo correspondiente al intervalo con el color de su tipo y además se genera un botón, con el título y el color correspondiente al tipo de intervalo y la hora de inicio y la hora final de este. Un ejemplo del resultado final se puede ver en las figuras 4.29 y 4.30. Este botón permite mostrar y ocultar los campos asociados a cada intervalo. Cada tipo de intervalo tiene unos campos propios asociados, los cuales analizaremos a continuación.

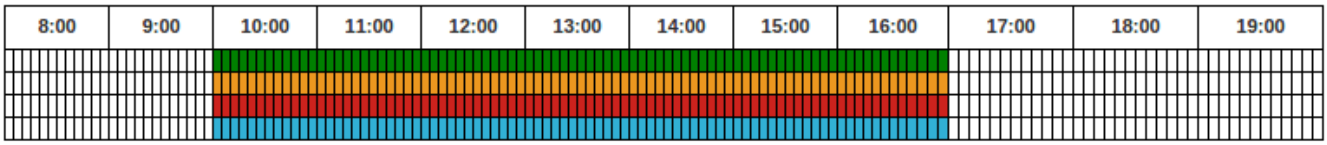


Figura 4.29: Visor de Intervalos con un intervalo de cada tipo

Los distintos botones son:

Figura 4.30: Botones de los distintos tipos de intervalos

Además, todos los intervalos disponen de dos botones comunes (Figura 4.31), uno para editar la hora de inicio y la hora final y otro para eliminarlo.

Figura 4.31: Botones Editar Tiempos de Intervalo y Eliminar Intervalo

Al pulsar el primero de ellos aparecerá un desplegable que indica cómo realizar esta actualización y el botón para llevarla a cabo. Para elegir los nuevos puntos de inicio y final se utilizarán los mismos métodos que a la hora de crear un nuevo intervalo, los cuales hemos visto anteriormente, y se pulsará el botón.

Para eliminar el intervalo, basta con pulsar el botón correspondiente.

Cada intervalo lleva asociados una serie de campos con la información relativa a cada actividad.

- Perforación: Los campos asociados a este tipo principalmente están relacionados con parámetros de control asociados a la maquinaria y se puede dividir en tres grupos: Valores de Producción, Alineación y Cilindros de Mando. En la figura 4.32 se muestran todos los campos asociados a este tipo de intervalo.

Valores de Producción

Presion Bastidor

Bar

Estación Uso

#

Presion Estacion

Bar

Velocidad

mm/m

Presion Corte

Bar

Empuje Bastidor

Ton

Velocidad Bentonita

s

Alineacion

Vertical

mm

Horizontal

mm

Pitch

mm

Yaw

mm

Roll

mm

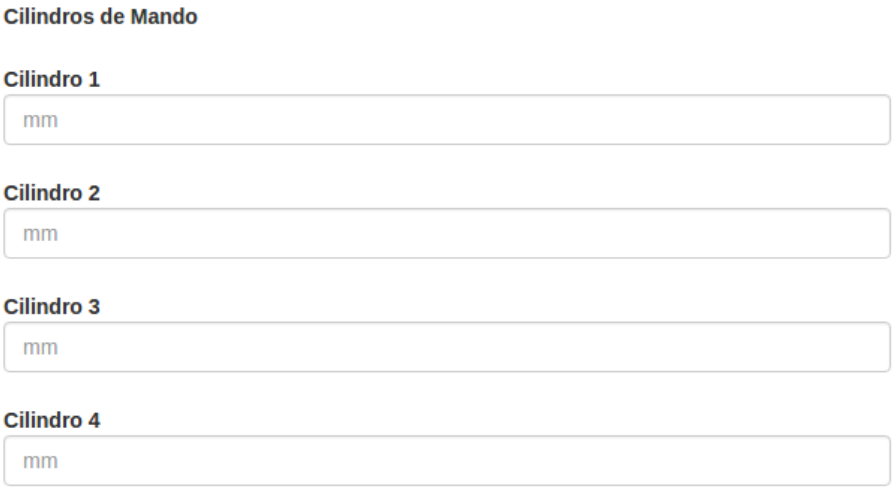


Figura 4.32: Formulario para introducir un Intervalo de Perforación nuevo

- Avería: Principalmente se adjunta información sobre la localización de la avería, el tipo de ésta y las acciones que se han llevado a cabo para subsanarla, así como un campo de texto para añadir información adicional. Los campos asociados a este tipo de intervalo se muestran en la figura 4.33.

Figura 4.33: Formulario para introducir una nueva Avería

La mayoría de los campos se han definido como seleccionables, excepto el de Descripción Detallada el cual permite la introducción de varias líneas de texto. En los seleccionables, las opciones de estos se adaptarán a la tecnología de la maquinaria principal para ofrecer al usuario las alternativas adecuadas. Además, al elegir cada una de las opciones presentes del campo Localización de la Avería, aparecerá uno o más campos específico de la opción elegida, que permite concretar más aún la localización (Figura 4.34).

Figura 4.34: Formulario para introducir una Avería con los campos desplegados

- Bajada de Tubo: Este intervalo de tiempo está destinado a aquellos trabajos destinados al aprovisionamiento y montaje del revestimiento del túnel. La información que contiene principalmente es la identificación de estos revestimientos y un campo para adjuntar observaciones. Todos campos del tipo de intervalo Bajada de Tubo están presentes en la figura 4.35.

Figura 4.35: Formulario para introducir un nuevo Intervalo de Bajada de Tubo

- Parada: Este intervalo sirve para definir todas aquellas detenciones que se pueden producir durante los trabajos normales de excavación debidas a muy distintas causas, las cuales están definidas en el desplegable que permite la elección de ellas. Además, se dispone de un campo de texto para adjuntar información adicional sobre dicha parada. Los campos asociados a los intervalos de parada se muestran en la figura 4.36.

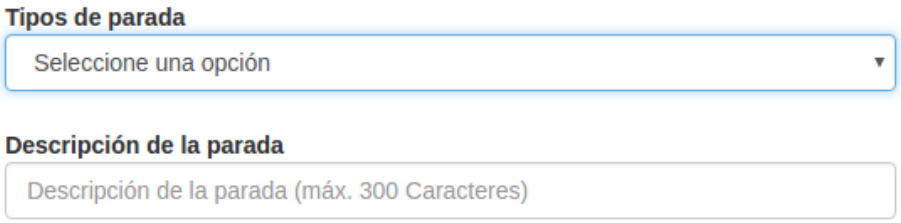


Figura 4.36: Formulario para introducir una nueva Parada

La pestaña Geología y Consumos (Figura 4.37) contiene los campos en los que se reflejarán dos aspectos del trabajo realizado durante el turno: La tipología del terreno sobre el que se ha estado trabajando, para la cual se ha diseñado un selector para elegir entre las distintas opciones posibles, y gasto de diversos recursos que se hayan realizado. Estos recursos son: Gasóleo, Agua, Bentonita, Polímero y Discos/Picas. Para reflejar estos consumos, se dispone de campos numéricos específicos para cada uno de ellos.

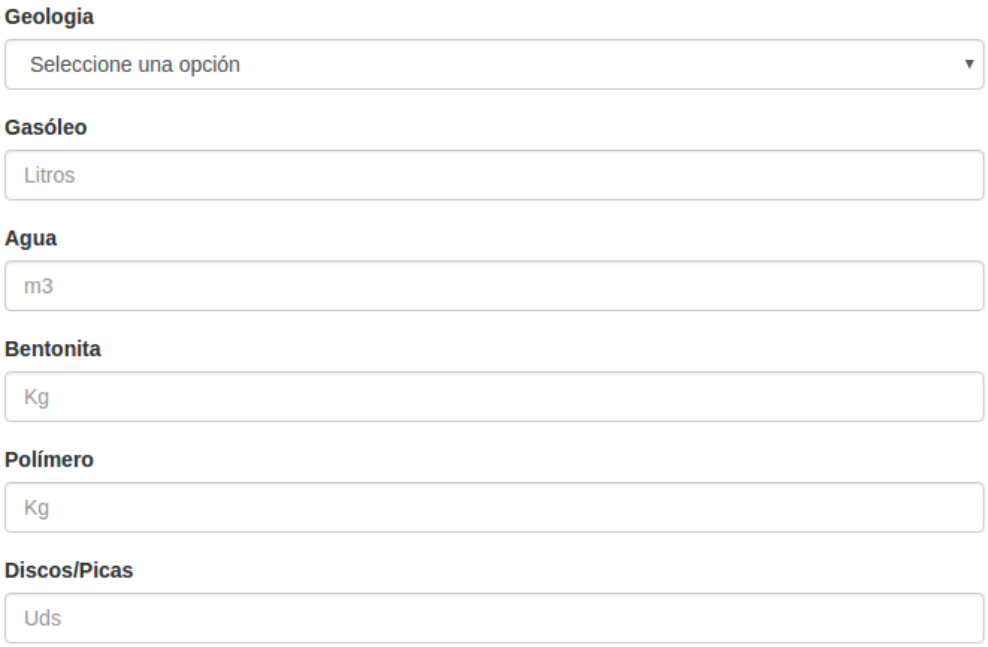


Figura 4.37: Campos de la pestaña Geología y Consumos

En la pestaña de Valores Totales (Figura 4.38) se encuentran una gran cantidad de campos numéricos, excepto el campo de Inyección Bentonita Frente, destinados a caracterizar la situación de la excavación a la finalización del turno, como por ejemplo el punto inicial y final de la perforación realizada durante el turno, el número de revestimiento inicial y final, valores asociados a la maquinaria utilizada como los campos QA, QE o HW.

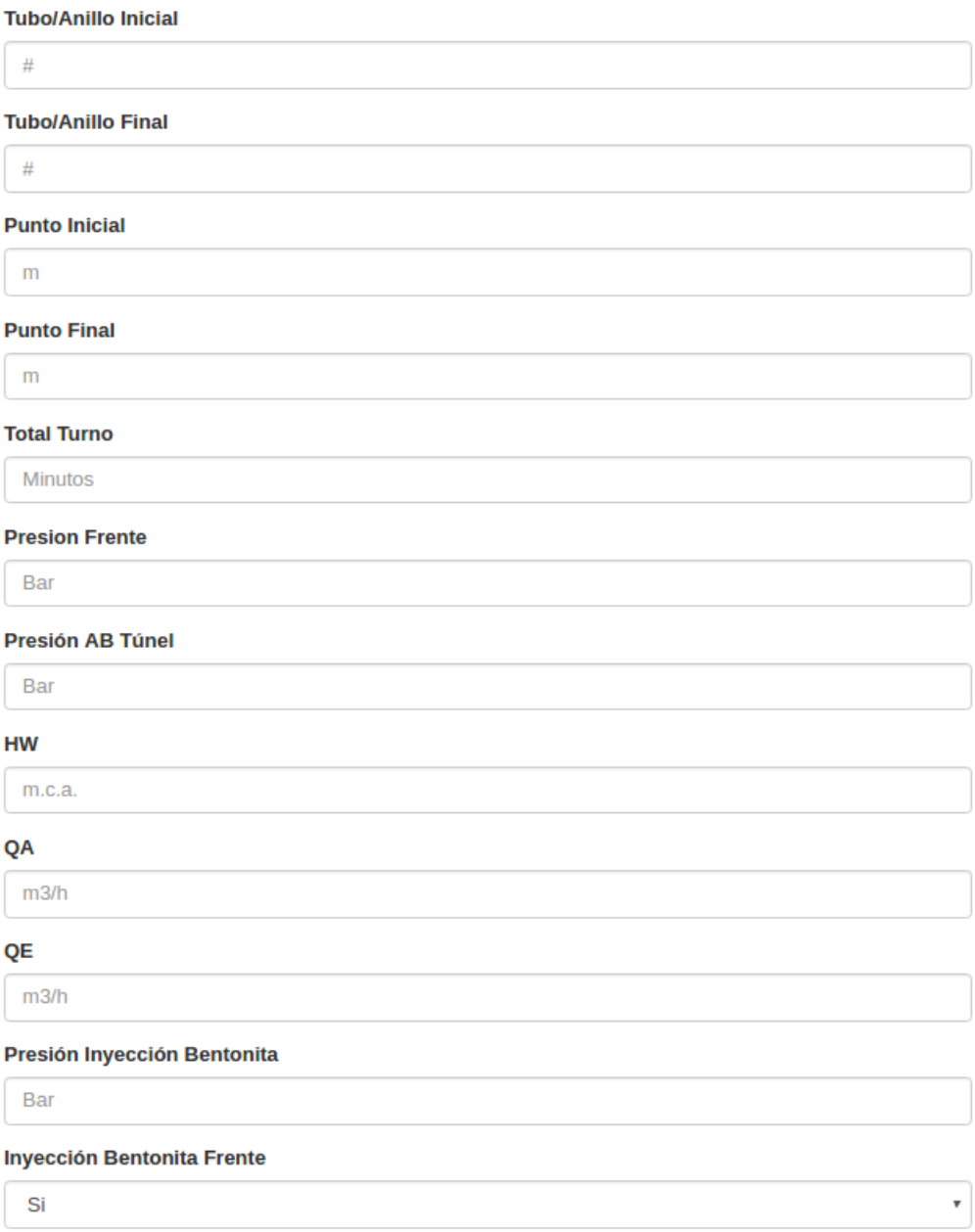


Figura 4.38: Campos de la pestaña Valores Totales

La última pestaña se denomina Observaciones y esta destinada a recoger notas en formato texto sobre tres principales aspectos del turno: Trabajos Realizados, Averías y Observaciones. De esta manera se le da la posibilidad al usuario de completar el parte de obra con aclaraciones que son imprescindibles en muchas ocasiones. Los campos que conforman esta pestaña están presentes en la figura 4.39.

Figura 4.39: Campos de la pestaña Observaciones

Para finalizar, fuera del sistema de pestañas y en la parte inferior de la página nos encontramos con dos botones como los de la figura 4.40

Figura 4.40: Botones Crear Parte de Obra y Descartar Parte de Obra

El botón *Crear Parte de Obra* nos permite confirmar la creación del parte de obra y el envío de este al servidor que procederá a su guardado en la base de datos. Tras esto se nos redirigirá a la ventana de Producción y Partes del proyecto donde ya estará presente el nuevo parte generado disponible para su consulta tanto en formado Excel como en formado PDF o para su edición o eliminación.

El botón rojo con un aspa blanca, permitirá al usuario, tras una ventana de confirmación, descartar el parte actual de tal manera que la información se borrará también del sistema de Autoguardado.

- Autoguardado

Para facilitar al usuario la creación de un nuevo parte de obra, se ha desarrollado un sistema de guardado automático. De esta manera el usuario puede pausar el rellenado de la información del parte y retomarlo cuando desee sin necesidad de enviar la información al servidor. Su funcionamiento consiste en que el navegador guarda periódicamente toda la información que ha ido introduciendo el usuario. Esta información queda guardada localmente y queda disponible para que el usuario retome el parte en cualquier momento. Cuando existe un parte de obra guardado localmente, se genera en la ventana de Producción y se genera una sección como la presente en la figura 4.41.

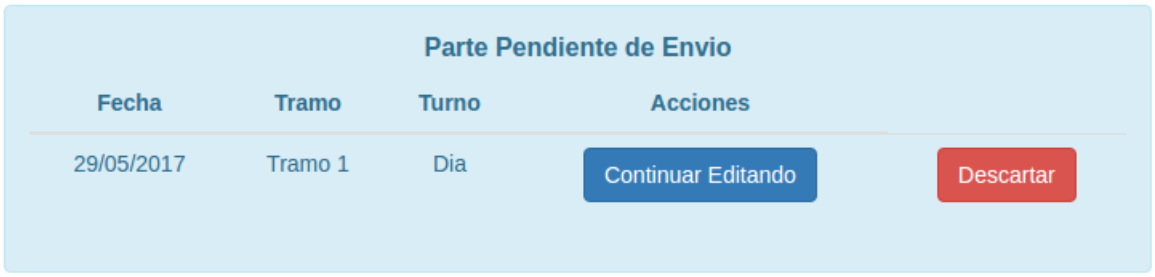


Figura 4.41: Sección Parte Pendiente de Envío en la página Producción y Partes

Esta sección tiene dos botones que permiten al usuario continuar con la creación del parte, lo nos redirigirá a una ventana como la de *Parte Nuevo* pero en la cual, la información que ya haya sido por el usuario, ya habrá sido rellenada automáticamente. El botón *Descartar* permite, tras una ventana de confirmación, eliminar definitivamente toda la información guardada localmente.

Cabe destacar que el sistema está diseñado para que sólo se pueda guardar localmente un único parte de obra, por lo que se si se quiere comenzar la creación de un nuevo parte de obra, la información de este nuevo parte de obra, sustituirá a la del parte que estaba guardado anteriormente. Para informar al usuario de ello, al intentar crear un nuevo parte de obra, si existe un parte de obra guardado localmente, se generará una ventana de confirmación para que el usuario tome la decisión que considere adecuada.

4.4.4. Manejo de partes creados

Una vez creado el nuevo parte de obra, éste se añadirá al listado de partes de la ventana *Producción y Partes* del proyecto correspondiente. Sobre cada uno de estos partes se pueden realizar cuatro acciones, *Abrir, Descargar xls, Editar y Eliminar* (Figura 4.42), las cuales procederemos a detallar:

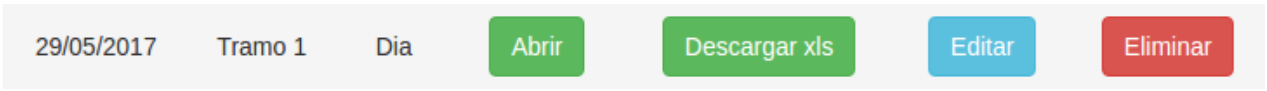


Figura 4.42: Botones del Parte de Obra en el listado de la página Producción y Partes

- **Abrir:** Al pulsar esta opción, la aplicación nos redirigirá hacia una página en la que se mostrará el parte de obra seleccionado en formato PDF a través del visor nativo del navegador que estemos utilizando. Desde esta ventana se da la opción de guardar el documento o incluso de imprimirlo directamente. Un ejemplo del parte de obra en formato .PDF se presenta en la figura 4.43.
- **Descargar xls:** Esta opción permite la descarga del parte de obra en formato xls.
- **Editar:** Esta opción permite al usuario la edición de un parte de obra ya creado. Para ello, la aplicación redirige al usuario a un formulario muy similar al visto anteriormente en la sección de *Parte Nuevo* con la particularidad todos los campos se le proporcionarán al usuario ya rellenados con la información guardada en la base de datos del parte de obra a editar. De esta manera, el usuario sólo debe actualizar aquellos campos que necesite y los demás campos mantendrán el mismo valor. Por lo demás, la estructura y el funcionamiento

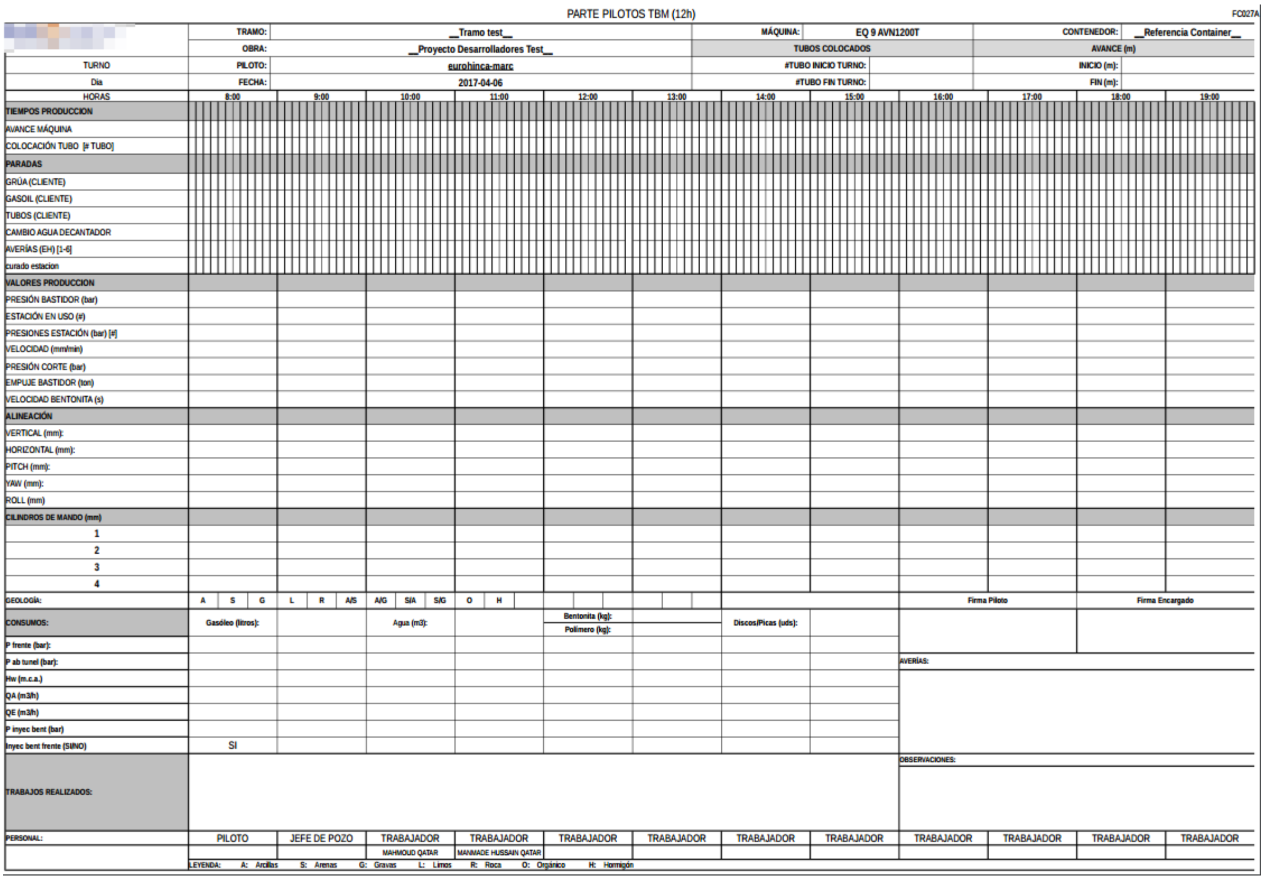


Figura 4.43: Parte de Obra en formato .pdf

son idénticos a los de la ventana de *Parte Nuevo*.

- **Eliminar:** Este botón permite al usuario eliminar de la base de datos de la aplicación el parte de obra seleccionado. Tras pulsarlo, se redirigirá hacia una ventana de confirmación como la de la figura 4.44.

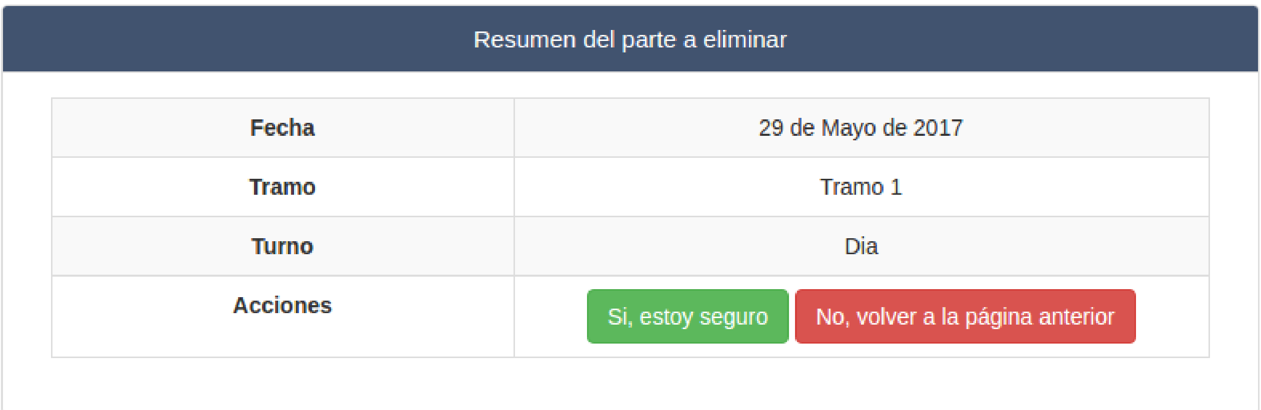


Figura 4.44: Ventana de confirmación para eliminar un Parte de Obra

La ventana de *Producción y Partes* de la sección de Proyectos es una de la partes esenciales de aplicación puesto que a través de ella se realizan todas aquellas acciones relacionadas el parte de obra.

4.4.5. Consultar Proyecto

En esta página encontraremos una tabla con toda la información de este proyecto, como la información básica, las tuneladoras (TBM), la maquinaria auxiliar y los trabajadores asignados a él. Esta página se muestra en la figura 4.45.

Soterramiento Carretera León-Burgos

Nombre	Soterramiento Carretera León-Burgos
Tramo	Tramo 1
Cliente	Carreteras RA S.A
Localidad	León
Código Postal	28941
País	España
Referencia TBM	RI-97931
Referencia Container	W-1586971
Jefe de Obra	Manuel Diaz Marcos
Trabajador asignado 1	Francisco Pérez Martínez
Trabajador asignado 2	José Rodríguez Alonso
Trabajador asignado 3	Miguel Díaz López
Trabajador asignado 4	Daniel Gómez Rodríguez
Trabajador asignado 5	Ignacio Fernández López
Trabajador asignado 6	Antonio Sánchez Torres
Otras maquinas 1	Grua | RI-1981W
Otras maquinas 2	Planta_bentonita | RI-901
Otras maquinas 3	Planta_bentonita | RI-991
Otras maquinas 4	Ventilador | RI-891
Otras maquinas 5	Torre_refrigeracion | RI-190I

Figura 4.45: Página Consultar Proyecto

- Editar Proyecto:

Este formulario (Figura 4.46) permite actualizar toda la información referente al proyecto, así como los empleados y las maquinarias asignadas a él. Tiene una estructura muy similar al de la ventana de Proyecto Nuevo con la peculiaridad de que todos los campos vienen rellenados con la información actual para una mayor comodidad al editarlo.

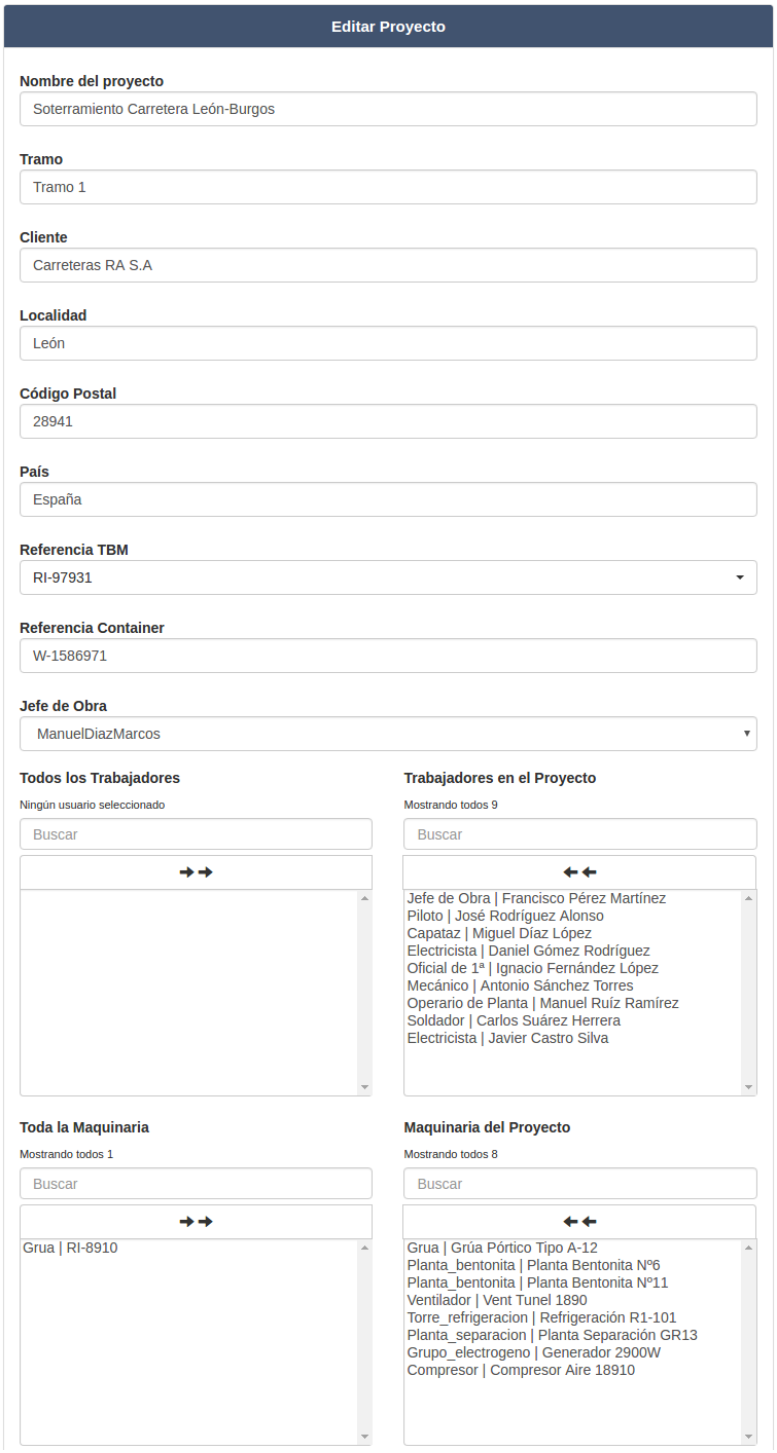


Figura 4.46: Página Consultar Proyecto

- Trabajadores:

Si pulsamos en el botón Trabajadores se ofrecerá un listado resumen con todos los empleados asignados al proyecto, donde consta la información básica de cada trabajador, el Nombre, y Oficio y un botón que da acceso a la ventana de consultar trabajador, la cual veremos más adelante en la sección de Personal. Esta ventana se muestra en la figura 4.47.

Trabajadores del Proyecto

ID	Nombre	Oficio	Acciones
54	Francisco Pérez Martínez	Jefe de Obra	Ver
55	José Rodríguez Alonso	Piloto	Ver
56	Miguel Díaz López	Capataz	Ver
57	Daniel Gómez Rodríguez	Electricista	Ver
58	Ignacio Fernández López	Oficial de 1ª	Ver
59	Antonio Sánchez Torres	Mecánico	Ver
60	Manuel Ruíz Ramírez	Operario de Planta	Ver
61	Carlos Suárez Herrera	Soldador	Ver
62	Javier Castro Silva	Electricista	Ver

Figura 4.47: Página Trabajadores del Proyecto

- Maquinaria:

En esta ventana (Figura 4.48) se puede ver un resumen de la maquinaria asignada al proyecto actual y distinguir dos partes. La parte superior la cual está dedicada a las tuneladoras, las cuales son las máquinas principales del proyecto. Sobre cada una de estas máquinas, tenemos disponible información como su Descripción, su Tipo, su Modelo, su Referencia Interna y su Referencia del Proveedor. La parte inferior es similar a la superior, pero ésta es referida a la maquinaria auxiliar. Sobre estas máquinas, tenemos información sobre su Descripción, su Tipo y su Referencia Interna únicamente.

Maquinaria del proyecto

Tuneladoras

Descripción	Tuneladora EPB AV-1981
Tipo	TBM
Modelo	RUV-1081
Referencia EH	RI-97931
Referencia Proveedor	QP-1891

Maquinaria del proyecto

	Descripción	Tipo	Referencia EH
Maquinaria 1	Grúa Pórtico Tipo A-12	Grua	RI-1981W
Maquinaria 2	Planta Bentonita Nº6	Planta Bentonita	RI-901
Maquinaria 3	Planta Bentonita Nº11	Planta Bentonita	RI-991
Maquinaria 4	Vent Tunel 1890	Ventilador	RI-891
Maquinaria 5	Refrigeración R1-101	Torre Refrigeración	RI-190I
Maquinaria 6	Planta Separación GR13	Planta Separación	RI-10901
Maquinaria 7	Generador 2900W	Grupo Electrógeno	RI-90181
Maquinaria 8	Compresor Aire 18910	Compresor	RI-7781

Figura 4.48: Página Maquinaria del Proyecto

Para finalizar tenemos el botón de *Salir del Proyecto*, que permite volver a la ventana con el listado de proyectos.

4.5. Sección Maquinaria:

Dentro de esta sección se encuentran las páginas que permiten realizar todas las acciones sobre la Maquinaria presente en la base de datos de la aplicación web. Desde la página inicial de la sección tendremos acceso a las cuatro acciones básicas que podremos realizar: Consultar, Crear, Editar y Eliminar. A continuación detallaremos cada una de estas páginas. La página principal de esta sección se puede ver en la figura 4.49.

Figura 4.49: Página inicial de la sección Maquinaria

- Consultar Maquinaria:

Esta parte se dividida en dos ventanas. Una primera ventana estructurada en forma de tabla con los campos más representativos de la base de datos, cuyo su cometido es facilitar al usuario la búsqueda de la máquina que quiere consultar. Para ello la tabla dispone de filtros para cada uno de los campos que permiten al usuario visualizar únicamente los elementos que coincidan con los parámetros introducidos.

Los filtros son de dos tipos en función del tipo de campo: seleccionables, en los que se muestran como opciones todos los valores del campo correspondiente presentes en la base de datos, y de texto, para aquellos campos en los que la mejor opción es realizar un búsqueda concreta. De esta manera, se permite al usuario tanto realizar búsquedas de una única maquinaria, como de grupos de ellas.

Además, cabe destacar que la tabla permite elegir el número de elementos que se muestran por página con el campo situado a en la esquina superior izquierda, y que, cada entrada de la tabla dispone, en la parte derecha de un botón azul con la leyenda *Ver* que da acceso a la segunda pagina de esta sección, la cual analizaremos más adelante. Por último, en la parte inferior, tenemos el botón *Volver al menú de maquinaria*, que permite volver a la página anterior. Un ejemplo de esta página se presenta en la figura 4.50.

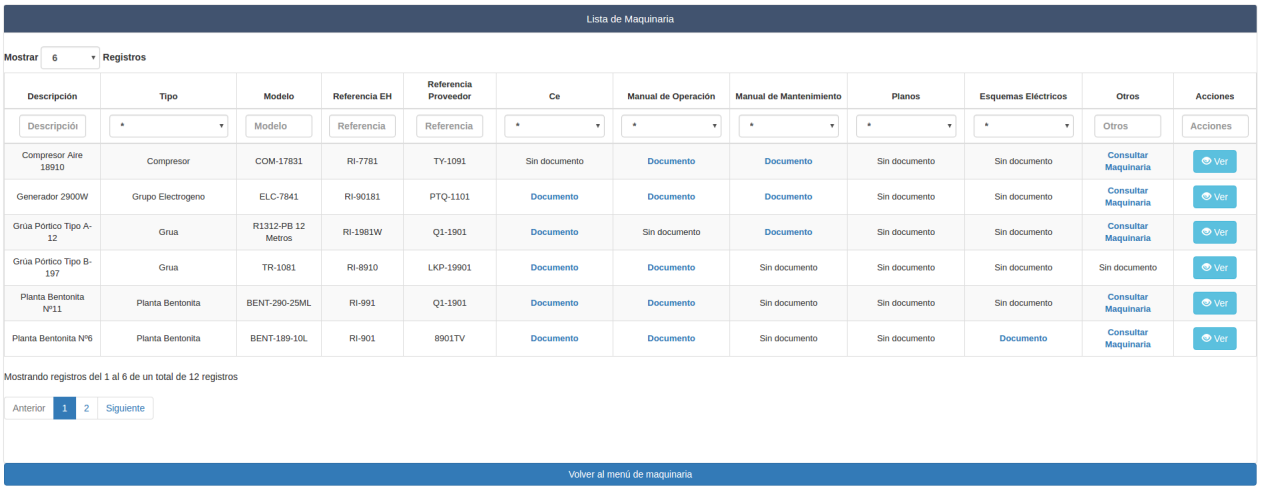


Figura 4.50: Página Consultar Maquinaria

Una vez elegida la máquina que se quiere consultar, pulsamos el botón correspondiente, y se mostrará una ventana con toda la información disponible sobre ella (Figura 4.51), como su Descripción, Tipo, Modelo, Referencia Interna y Referencia Proveedor. Además desde esta ventana, tendremos acceso a todos los documentos adjuntos a la maquinaria, normalmente documentos relacionados con el funcionamiento, mantenimiento, etc, de cada una de ellas. Para ello basta con pulsar en *Abrir*, lo que nos redirigirá al archivo en cuestión.

Figura 4.51: Página Consultar Máquina

4.5.1. Crear Maquinaria

Esta ventana permite introducir nuevas maquinarias en la base de datos. Consta principalmente de campos de texto y seleccionables para la información básica de la maquinaria, *Descripción, Tipo, Modelo y Referencia Interna y de Proveedor*, y de campos que permiten

subir al servidor documentos pertenecientes a la maquinaria, que, una vez añadida a la base de datos, estarán disponibles para su consulta. Existen cinco documentos comunes, *CE, Manual de Operación, Manual de Mantenimiento, Planos y Esquemas Eléctricos* pero además se le da la opción al usuario de añadir tantos documentos extras como desee utilizando el campo Otros Documentos y el símbolo *"+"* que esta situado a su lado. Pulsando este botón, generará un nuevo campo que permite adjuntar otro archivo.

Una vez introducidos todos los datos, pulsaremos el botón *Crear Máquina* y será guardada en la base de datos de la aplicación. Cabe destacar que todos los campos de información básica son requeridos, por lo que, si al pulsar el botón de Crear, alguno de estos campos está vacío, la aplicación informará al usuario que debe de rellenar estos campos antes de crearla. El formulario completo para crear una nueva máquina lo podemos ver en la figura 4.52.

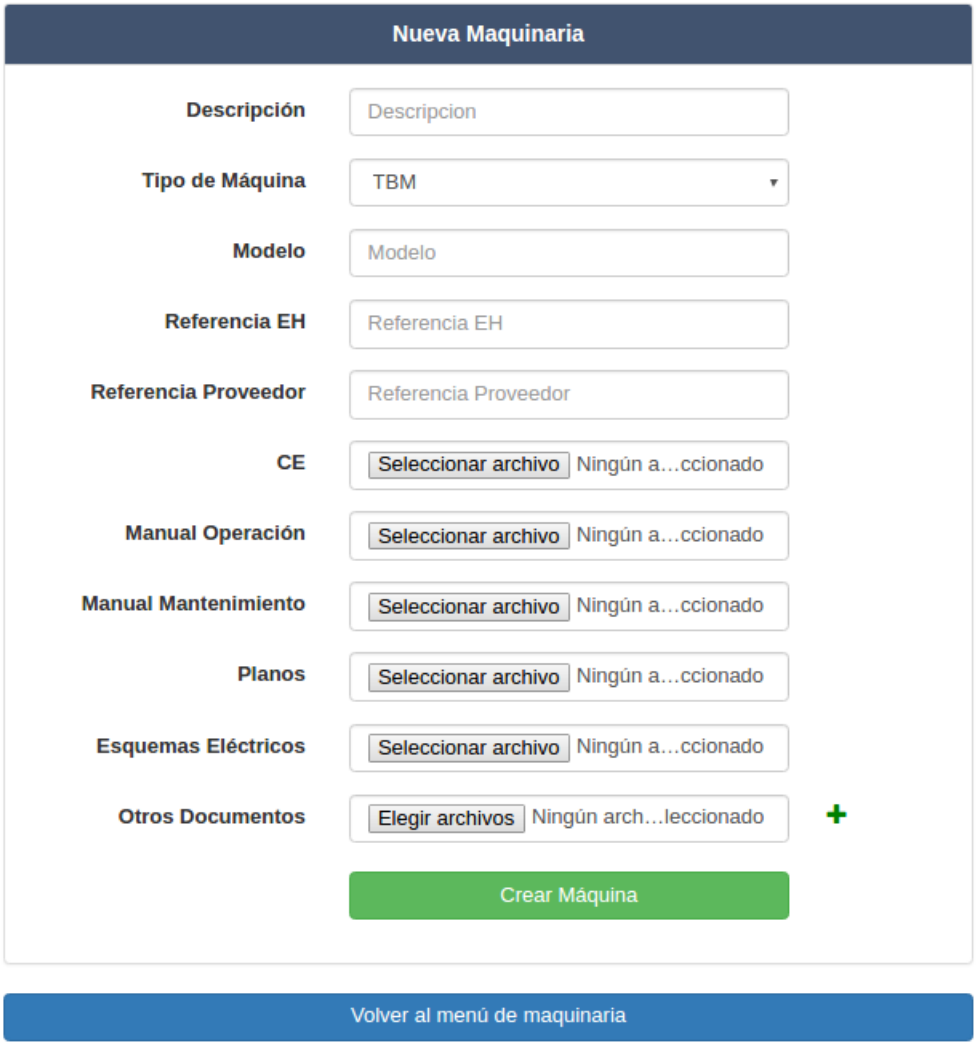


Figura 4.52: Formulario Nueva Maquinaria

4.5.2. Editar Maquinaria

Esta sección permite actualizar la información sobre maquinaria guardada en la base de datos. Como la anterior, se divide en dos páginas, la primera (Figura 4.53) es un listado con todos los registros presentes en la aplicación, en la que se puede ver la información principal de cada una de ellos, *Descripción, Modelo, Tipo, Referencia Interna y Referencia de Proveedor*. Cada uno de estos registros tiene su correspondiente botón *Editar* que da acceso al formulario donde introducir los datos actualizados.

Lista de Máquinas

Descripción	Modelo	Tipo	Ref.EH	Ref.Proveedor	Acciones
Compresor Aire 18910	COM-17831	Compresor	RI-7781	TY-1091	Editar
Generador 2900W	ELC-7841	Grupo_electrogeno	RI-90181	PTQ-1101	Editar
Grúa Pórtico Tipo A-12	R1312-PB 12 Metros	Grua	RI-1981W	Q1-1901	Editar
Grúa Pórtico Tipo B-197	TR-1081	Grua	RI-8910	LKP-19901	Editar
Planta Bentonita Nº11	BENT-290-25ML	Planta_bentonita	RI-991	Q1-1901	Editar
Planta Bentonita Nº6	BENT-189-10L	Planta_bentonita	RI-901	8901TV	Editar
Planta Separación GR13	PR-190145	Planta_separacion	RI-10901	PQ-1021	Editar
Refrigeración R1-101	REF- 190P	Torre_refrigeracion	RI-190I	PR-11901	Editar
Tuneladora EPB AV-1421	P1231-B	TBM	RI-141301	JKL1301	Editar
Tuneladora EPB AV-1981	RUV-1081	TBM	RI-97931	QP-1891	Editar
Tuneladora EPB UV-47517	Q1-1350 Plus	TBM	RI-18910	OP-189	Editar
Vent Tunel 1890	VENT-189-01	Ventilador	RI-891	9101-Q	Editar

Figura 4.53: Página Editar Maquinaria

Una vez pulsado este botón, se redirigirá a una página muy similar a la vista en *Crear Maquinaria* con la particularidad que los campos sobre los que hay información ya en la base de datos, se proporcionarán ya rellenados.

Con respecto a los campos referidos a documentos adjuntos, en aquellos en los que ya haya un archivo subido, se puede apreciar que se da la opción de visualizar este documento o de eliminarlo. Si se elige la segunda opción, se genera la entrada correspondiente vacía, lo que permitirá al usuario adjuntar un nuevo archivo para ese campo si lo desea. Una vez realizados

los cambios necesarios, se pulsará el botón *Actualizar* para grabarlos en la base de datos y se redirigirá a una ventana similar a la vista en *Consultar Maquinaria* donde se muestran la información actualizada.

4.5.3. Eliminar Maquinaria

Desde esta parte se permite al usuario eliminar una máquina de la base de datos de la aplicación. Su estructura es muy similar a la anterior, con un listado con todos los registros que da acceso a una ventana (Figura 4.54) en la cual se muestra toda la información de la maquinaria. En la parte inferior tenemos dos botones, uno para confirmar que el usuario desea eliminar la maquinaria seleccionada y otro que permite volver a la ventana anterior.

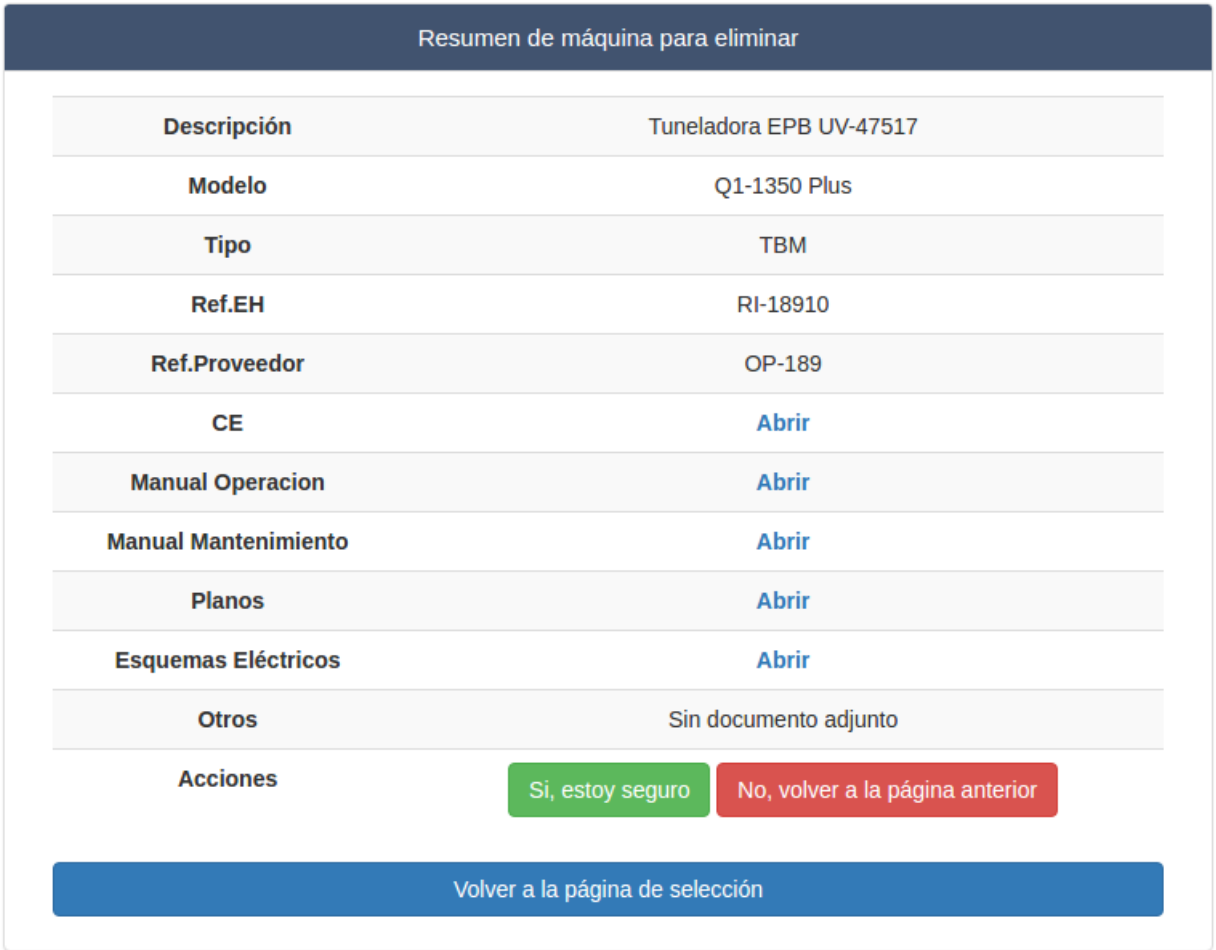


Figura 4.54: Ventana de confirmación para Eliminar Máquina

4.6. Sección Personal

Esta sección esta dedicada a la gestión de los empleados de la empresa que realizan distintos trabajos en las obras llevadas a cabo por la empresa. Tiene una estructura muy similar a la sección anterior puesto que sus funciones también lo son, pero en esta ocasión trabajando sobre la base de datos de trabajadores.

En primer lugar encontramos una página con acceso a las cuatro acciones básicas que podemos realizar (Figura 4.55). A continuación analizaremos cada una de ellas.

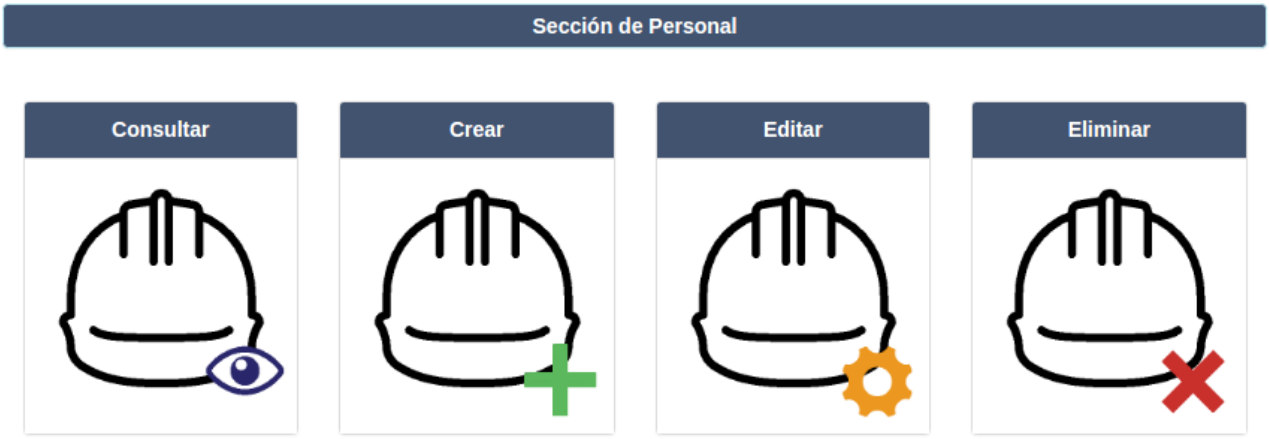


Figura 4.55: Página inicial de la sección Personal

4.6.1. Consultar Personal

Al igual que su equivalente de Maquinaria, se dispone en dos páginas, una primera, con estructura de tabla y una segunda en la que ya se muestran los datos del trabajador seleccionado.

En la primera de estas páginas, presente en la figura 4.56, se muestran la información más importante de cada uno de los trabajadores presentes en la base de datos. Existen filtros para cada uno de los campos que permiten al usuario realizar búsquedas complejas sobre uno o más campos, lo que facilita la tarea. También destaca el botón *Exportar datos en formato Excel* que permite al usuario descargar un archivo .xls actualizado con todos los trabajadores presentes en la base de datos y toda la información sobre ellos.

Una vez seleccionado el empleado que se quiere consultar, se pulsa el botón correspondiente situado en la parte derecha de la tabla y se dará acceso a la segunda página, que podemos observar en la figura 4.57.

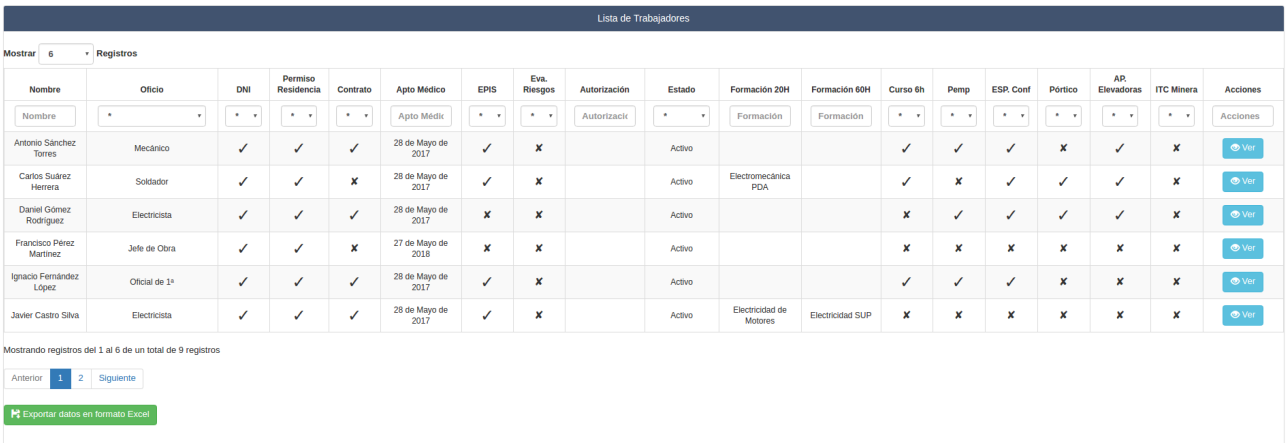


Figura 4.56: Página Consultar Personal

En esta segunda página (Figura 4.57) podemos ver un listado con toda la información del trabajador seleccionado.

Trabajador

Nombre	Antonio Sánchez Torres
Oficio	Mecánico
DNI	✔
Permiso de residencia	✔
Contrato	✔
Apto Médico	28 de Mayo de 2017
EPIS	✔
Ficha de evaluación de riesgos	✖
Equipos de Trabajo	✖
Autorización	
Estado	✔
Formación 20 Horas	
Formación 60 Horas	
Curso de 6 horas	✔
Pemp	✔
Esp Conf	✔
Pórtico	✖
AP Elevadoras	✔
ITC Minería	✖
Observaciones	✖

Figura 4.57: Página de Consultar un Trabajador

4.6.2. Editar Personal

Al igual que el apartado anterior, se divide en dos páginas. La primera de ellas (Figura 4.58) se compone de un listado con todos los trabajadores presentes en la base de datos de la aplicación. Este listado está compuesto con algunos de los datos más relevantes de cada trabajador como Nombre y Apellidos, Oficio, datos relacionados con la formación de cada uno de ellos y sobre su documentación disponible. Además de estos datos se dispone de el botón /textitEditar que dará acceso a la segunda página y de un icono que da información sobre el *Estado* de cada uno de los trabajadores: Activo e Inactivo.

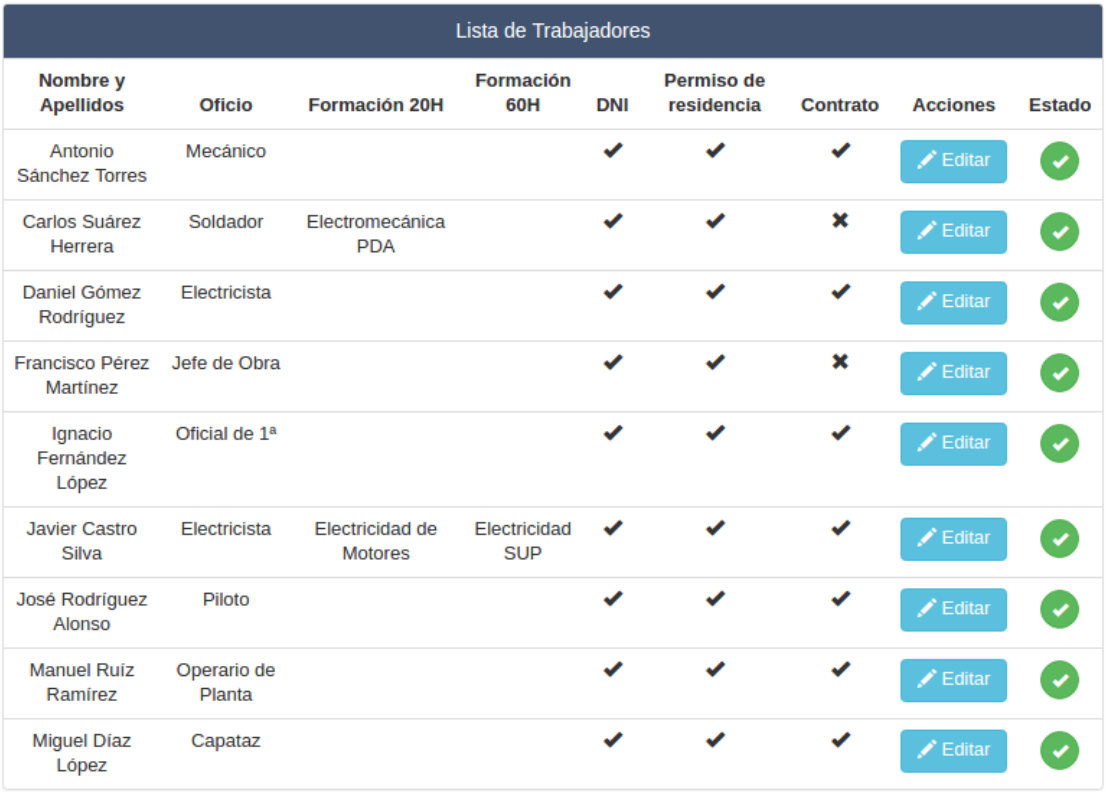


Figura 4.58: Página de Editar Personal

Una vez seleccionado el trabajador el cual queremos actualizar, pulsaremos el botón *Editar* y tendremos acceso a la segunda página, en la figura 4.59.

Esta página está formada por un formulario similar al utilizado al crear un nuevo trabajador, con la particularidad que los campos se le proporcionan al usuario rellenados con la información actual del trabajador. De esta manera bastará con el usuario actualice la información que desee y pulse el botón *Actualizar Trabajador* y la información quedará almacenada en la base de datos.

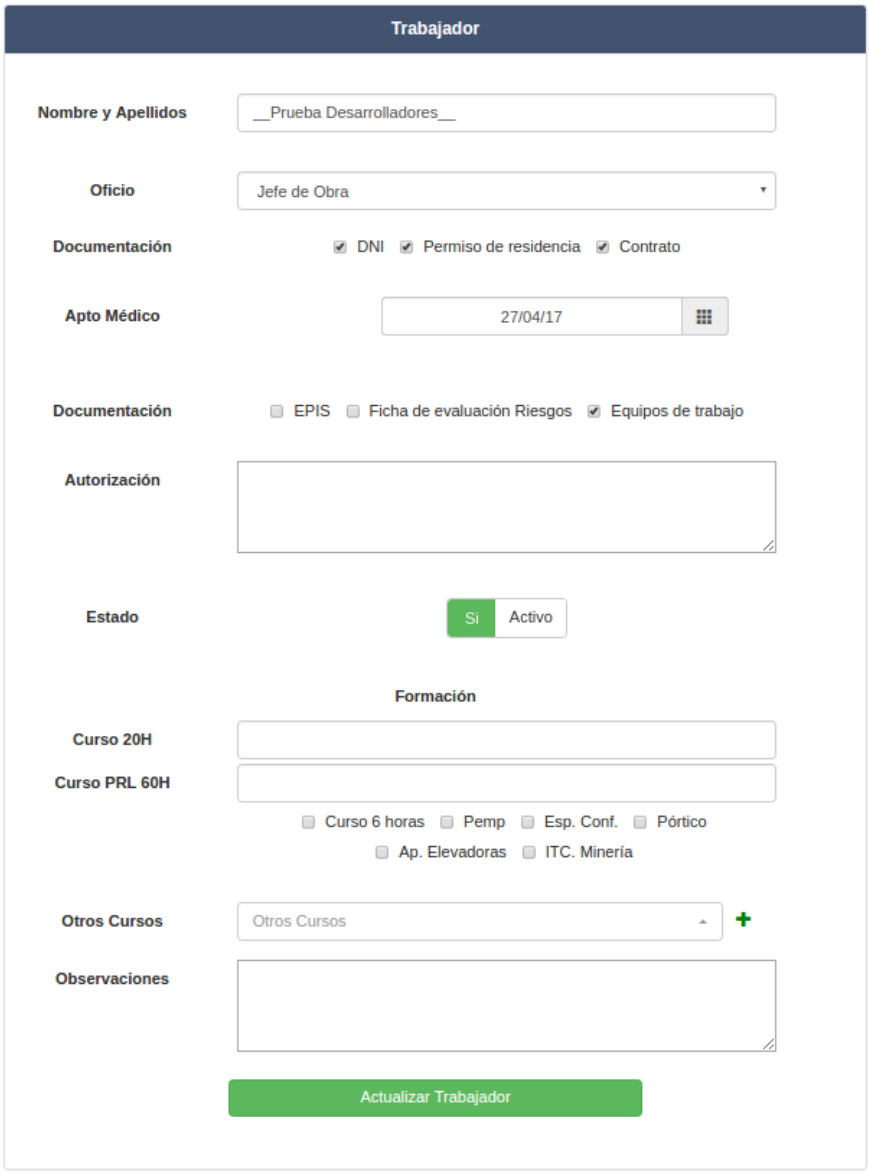


Figura 4.59: Formulario para Editar Trabajador

4.6.3. Eliminar Personal

Análogamente al anterior, el apartado que permite eliminar de la base de datos un trabajador, se compone de una primera ventana en forma de listado que permite seleccionar el trabajador a eliminar y una segunda ventana con los datos más importantes del trabajador y que tiene como propósito servir como confirmación al usuario.

4.7. Sección de Administración

Esta sección, a la cual sólo tienen acceso aquellos usuarios con el rol de Control, tiene dos propósitos principales: gestionar los permisos de los *Jefes de Obra* y *Pilotos* para acceder a un proyecto y crear nuevos usuarios así como actualizar los atributos de los usuarios ya presentes en la aplicación.

Al entrar en la sección encontramos una página que permite elegir entre la subsección de Proyectos y de Usuarios.

4.7.1. Proyectos

En la subsección de *Proyectos* encontramos una ventana con un listado, mostrado en la figura 4.60, con todos los proyectos presentes en la aplicación. Para cada uno de ellos, además de datos sobre estos, tenemos dos botones: *Abrir, Permisos* y *Eliminar*.

Figura 4.60: Página de Proyectos de la sección Administración

El primero de ellos redirigirá a la ventana *Opciones del Proyecto* correspondiente, la cual ya ha sido analizada anteriormente.

El botón *Permisos* da acceso a una nueva ventana en la que, en la parte superior encontramos una tabla con la información más relevante del proyecto. En la parte superior encontramos un sistema de dos listas, similar a las utilizadas anteriormente en la aplicación. En la lista situada a la izquierda se disponen todos los usuarios que no tienen acceso al proyecto seleccionado y en la lista de la derecha aquellos con acceso. Por lo tanto para editar el permiso de acceso de un usuario, sólo debemos seleccionarlo y colocarlo en la lista adecuada. Usaremos el botón *Editar Permisos* para confirmar los cambios. Esta página se puede ver en la figura 4.61.

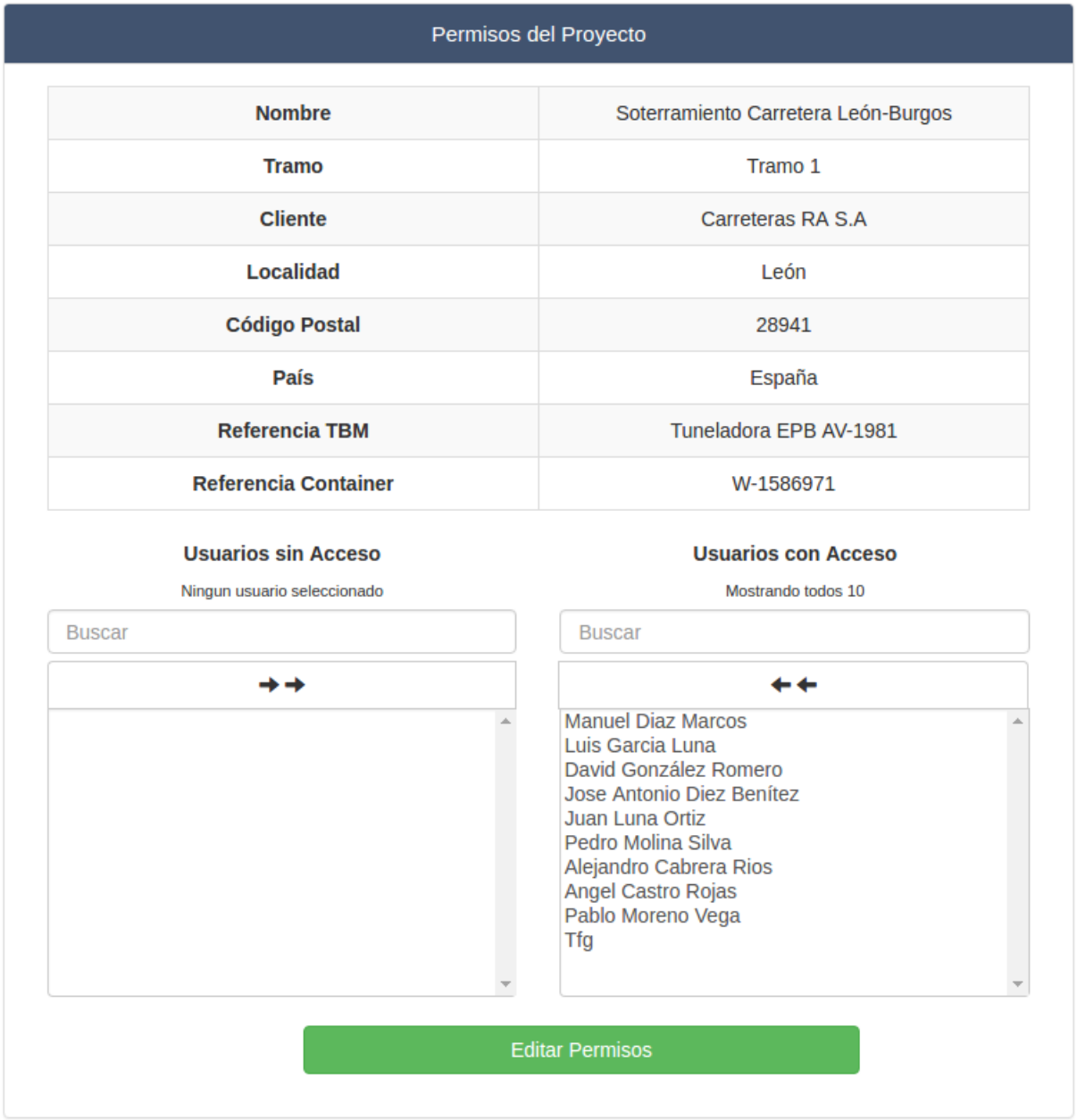


Figura 4.61: Página Permisos del Proyecto

Por último, con el botón *Eliminar* redirigirá a una ventana de confirmación, mostrada en la figura 4.62, con los datos del proyecto seleccionado y el botón para confirmar el borrado del proyecto.

Soterramiento Carretera León-Burgos

Nombre	Soterramiento Carretera León-Burgos
Tramo	Tramo 1
Cliente	Carreteras RA S.A
Localidad	León
Codigo Postal	28941
Pais	España
Referencia TBM	Tuneladora EPB AV-1981
Referencia Container	W-1586971
Francisco Pérez Martínez	**Trabajador asignado 1**
José Rodríguez Alonso	**Trabajador asignado 2**
Miguel Díaz López	**Trabajador asignado 3**
Daniel Gómez Rodríguez	**Trabajador asignado 4**
Ignacio Fernández López	**Trabajador asignado 5**
Antonio Sánchez Torres	**Trabajador asignado 6**
Manuel Ruíz Ramírez	**Trabajador asignado 7**
Carlos Suárez Herrera	**Trabajador asignado 8**
Javier Castro Silva	**Trabajador asignado 9**
Acción	Eliminar proyecto

Figura 4.62: Ventana de confirmación para Eliminar Proyecto

4.7.2. Usuarios

La segunda subsección tiene como finalidad crear, editar y eliminar los usuarios que tendrán acceso a la aplicación. Inicialmente tenemos un listado con todos los usuarios dados de alta en la aplicación. En este listado está el Nombre de Usuario, el Nombre y el Rol de cada uno de ellos así como los botones que permiten editarlo y eliminarlo. El listado de usuarios que da acceso a sus acciones correspondiente se muestra en la figura 4.63

Lista Usuarios

ID	Username	Nombre	Rol	Acciones	
2	ManuelDiazMarcos	Manuel Diaz Marcos	Jefe de Obra	Editar	Eliminar
3	LuisGarciaLuna	Luis Garcia Luna	Piloto	Editar	Eliminar
4	DavidGonzalezRomero	David González Romero	Piloto	Editar	Eliminar
5	JoseAntonioDiezBenitez	Jose Antonio Diez Benítez	Jefe de Obra	Editar	Eliminar
6	JuanLunaOrtiz	Juan Luna Ortiz	Control	Editar	Eliminar
7	PedroMolinaSilva	Pedro Molina Silva	Piloto	Editar	Eliminar
8	AlejandroCabreraRios	Alejandro Cabrera Rios	Jefe de Obra	Editar	Eliminar
9	AngelCastroRojas	Angel Castro Rojas	Control	Editar	Eliminar
10	PabloMorenoVega	Pablo Moreno Vega	Control	Editar	Eliminar

Figura 4.63: Listado de Usuarios de la sección Administración

Al final del listado está el botón *Crear Nuevo Usuario* que permite la creación de nuevos usuarios. Para ello se debe rellenar un sencillo formulario como el que ve en la figura 4.64

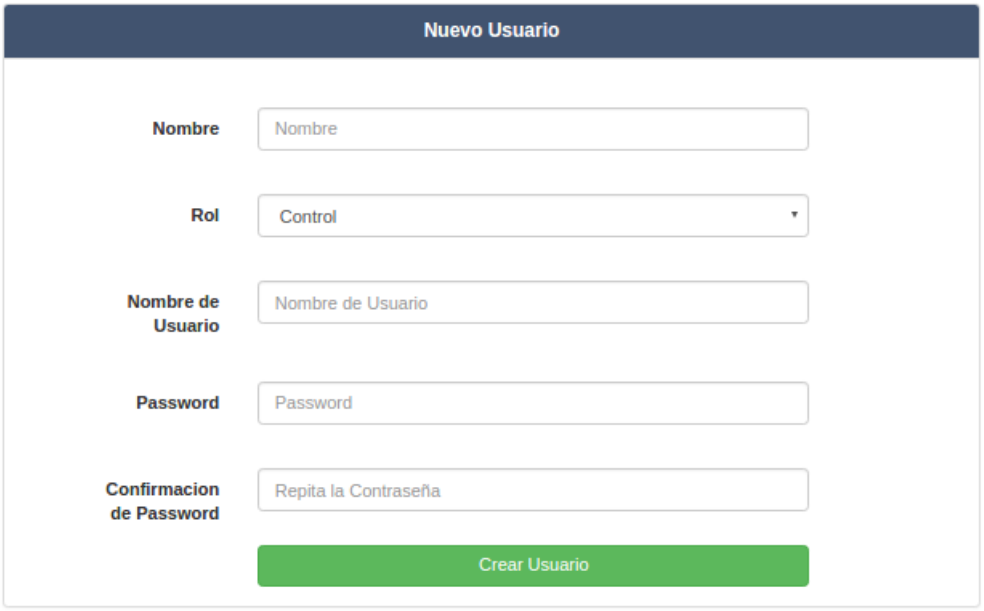


Figura 4.64: Formulario para crea un Nuevo Usuario

La acción de *Editar* se utiliza un formulario, similar al visto anteriormente, previamente rellenado con los datos actuales del usuario. Para la de *Eliminar*, al igual que en el resto de la aplicación, el usuario accede a una página de confirmación (Figura 4.65) para evitar posibles errores.

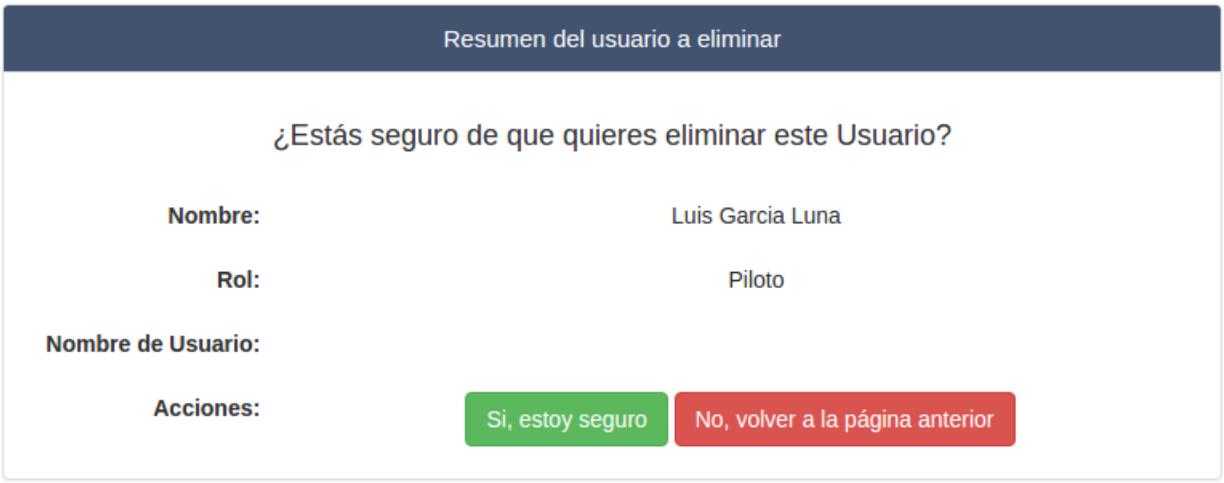


Figura 4.65: Ventana de confirmación para Eliminar Usuario

Capítulo 5

Desarrollo y Programación

En este capítulo analizaremos cómo se han desarrollado las principales funcionalidades de la aplicación. Para se profundizará en la implementación concreta de su características principales en las siguientes secciones.

5.1. Formularios

Los formularios son la columna vertebral de la aplicación, ya que mediante ellos, le damos la oportunidad al usuario de introducir datos en la aplicación, desde los pertenecientes a Trabajadores y Maquinarias a los que conforman los Partes de Obra.

Los formularios son componentes pertenecientes a HTML y que se definen mediante las etiquetas *<form>* e *<input>*. La primera sirve para definir el inicio y el final de un formulario que estará compuesto por uno o más campos de distinto tipo definidos por la segunda. En el fragmento 5.1 podemos ver un extracto de código de uno de los múltiples formularios presentes en la aplicación.

```
<form class="form-horizontal" method="POST" action="" enctype="multipart/form-
    data" data-toggle="validator">
{% csrf_token %}
        <div class="form-group">
            <label class="control-label col-sm-2 col-sm-offset-1 col-xs-3 col-xs
                -offset-1">Nombre</label>
            <div class="col-xs-7 col-sm-7">
                <input id="Nombre" name="nombre" placeholder="Nombre" class="
                    form-control input-md" type="text" required value="{{user.
                    first_name}}">
            </div>
        </div>
        <br>
        <div class="form-group">
            <label class="control-label col-sm-2 col-sm-offset-1 col-xs-3 col-xs
                -offset-1">Rol</label>
            <div class="col-xs-7 col-sm-7">
                <select id="Rol" name="rol" class="form-control">
                        <option value="{{rol_actual}}">{{rol_actual}}</option>
                        {% for rol in usuario_roles %}
                            {% if rol != rol_actual %}
                                <option value="{{rol}}">{{rol}}</option>
                            {%endif%}
                        {% endfor %}
                </select>
```

```
            </div>
        </div>
        <br>
        <div class="form-group">
            <label class="control-label col-sm-2 col-sm-offset-1 col-xs-3 col-xs
                -offset-1">Nombre de Usuario</label>
            <div class="col-xs-7 col-sm-7">
                <input id="Usuario" name="username" placeholder="Nombre de
                    Usuario" class="form-control input-md" type="text" required
                    value="{{user.username}}">
            </div>
        </div>
        <br>

        <div class="form-group">
            <label class="control-label col-sm-2 col-sm-offset-1 col-xs-3 col-xs
                -offset-1"></label>
            <div class="col-xs-7 col-sm-7">
                <button id="submit" name="Submit" class="btn btn-success btn-
                    block">Editar Usuario</button>
            </div>
        </div>
    </div>
</form>
```

Fragmento 5.1: Código del formulario para crear un nuevo usuario

Como se puede observar, para definir un formulario, sólo es necesario utilizar código HTML, pero para darle un aspecto visual adecuado y para añadir funcionalidades asociadas a ellos, es necesario el uso de CSS y JS. Para el diseño gráfico hemos utilizado el definido en la librería Bootstrap, sin añadir ningún cambio significativo. Con respecto a las funcionalidades asociadas que han sido desarrolladas, serán analizadas a continuación.

5.2. Validación de Formularios

Es un proceso de comprobación de algunos campos para confirmar que la información que contiene es válida antes de ser enviada al servidor, y en caso contrario, avisar al usuario para que corrija estos datos. Desde el punto de vista del desarrollo existen dos tipos validación: Validación llevada a cabo únicamente desde el cliente y la validación en la cual interviene el servidor.

La validación desde el cliente es un código JS que comprueba que los datos introducidos en un campo de un formulario cumplen con unas determinadas reglas, por ejemplo que no está vacío o que tiene una morfología correcta. Un ejemplo claro de esto es la validación sobre el campo Código Postal del formulario Proyecto Nuevo. En el fragmento 5.2 el código desarrollado de esta funcionalidad:

```
function validacionDNI(input){
    if (input.value.length == 9) {
        digitos_DNI = input.value.slice(0, 8)
        letras_DNI = input.value.slice(8, 9)
        digitos_NIE = input.value.slice(1, 9)
        letras_NIE = input.value.slice(0, 1)
        if (isNaN(letras_DNI) && !(isNaN(digitos_DNI))) {
            formgroup = document.getElementById(("form-group " + input.id))
            formgroup.className = "form-group"
```

```
            submit = document.getElementById("submit")
            submit.disabled = false
        } else if (isNaN(letras_NIE) && !(isNaN(digitos_NIE))) {
            formgroup = document.getElementById(("form-group " + input.id))
            formgroup.className = "form-group"
            submit = document.getElementById("submit")
            submit.disabled = false
        } else {
            formgroup = document.getElementById(("form-group " + input.id))
            formgroup.className += " has-error"
            submit = document.getElementById("submit")
            submit.disabled = true
        }
    } else {
        formgroup = document.getElementById(("form-group " + input.id))
        formgroup.className += " has-error"
        submit = document.getElementById("submit")
        submit.disabled = true
    }
}
```

Fragmento 5.2: Código para la validación de formularios desde el cliente

En este caso, si el usuario introduce un DIN o NIE que se considera no válido, el código, utilizando la librería Bootstrap, remarca el campo en color rojo, y actualiza los atributos HTML del formulario para mostrar un mensaje de error y desactivar el botón de *Crear Nuevo Empleado* para impedir al usuario que envíe la información errónea. Una vez el usuario introduzca un número de documento válido, el código revierte los cambios y permite al usuario el envío de los datos.

En el segundo tipo de validación, ésta se lleva a cabo tras realizar un la interacción con el servidor. Principalmente se utiliza para evitar campos repetidos, por ejemplo evitar que dos Proyectos tengan el mismo nombre y para comprobar que la información introducida en algunos campos hace referencia a Proyectos, Maquinarias, Empleados y Usuarios existentes en la base de datos de la aplicación.

Esta interacción se lleva a cabo mediante AJAX (Asynchronous JavaScript And XML), técnica que permite mantener una comunicación asíncrona con el servidor. Esta interacción se realiza para pedir al servidor que le envíe la información adecuada al cliente que, tras realizar las comprobaciones adecuadas, confirmará si los datos introducidos son válidos.

El código de la función JavaScript que realiza la comunicación con el servidor se muestra en el fragmento 5.3.

```
function sendAjax(text_form, tipo, campo, proyecto) {
    xmlHttp.open("POST", "/ajax", true);
    xmlHttp.setRequestHeader("X-CSRFToken", getCookie('csrftoken'));
    data = [tipo, campo, proyecto, text_form.value]
    xmlHttp.send(data);
}
```

Fragmento 5.3: Código para la validación de formularios mediante comunicación con el servidor

Una parte de la función que realiza las comprobaciones sobre la información es se muestra en el fragmento 5.4. Ésta en concreto es la validación del nombre de un proyecto para evitar la introducción de nombres ya presentes en la aplicación:

```
var xmlHttp;
```

```
function ajaxFunction() {
    xmlHttp = new XMLHttpRequest();
    xmlHttp.onreadystatechange = function() {
        if (xmlHttp.readyState == 4) {
         options = JSON.parse(xmlHttp.responseText)
                tipo = xmlHttp.getResponseHeader("tipo")
                campo = xmlHttp.getResponseHeader("campo")
                        if (tipo == "validacion_nombre_proyecto") {
                if (options.length != 0) {
                        error_message = document.getElementById("error_message_"
                            + campo)
                        error_message.style = "display:block"
                        console.log(campo)
                        campo = document.getElementById(("group_" + campo))
                        campo.className += " has-error"
                        campo = document.getElementById(("group_" + campo))
                        submit = document.getElementById("submit")
                        submit.disabled = true
                }else{
                        error_message = document.getElementById("error_message_"
                            + campo)
                        error_message.style = "display:none"
                        campo = document.getElementById(("group_" + campo))
                        campo.className = "form-group has-success"
                        submit = document.getElementById("submit")
                        submit.disabled = false
                    }
                }
        }
  }
}
```

Fragmento 5.4: Código de la función que realiza la comunicación AJAX con el servidor

En todos los casos en los que se usa la validación, si la información no cumple los requisitos adecuados se marca en rojo el campo correspondiente, mostrando un mensaje de error y desactivando el botón para enviar el formulario.

5.3. Listado Opciones

De forma similar al apartado anterior, otra de las funcionalidades auxiliares a los formularios es ofrecer al usuario un listado de las opciones disponibles en aquellos campos en los que debe elegir, por ejemplo un empleado o una máquina, que debe estar en la base de datos. De esta manera, el usuario dispone de una lista donde poder elegir la opción adecuada con la información actualizada de la base de datos. Además de facilitar la tarea al usuario, éste no se podrá equivocar al rellenar el campo. La función que implementa esta funcionalidad es muy similar a la del apartado anterior, con la diferencia que cuando recibe la información del servidor, el código JavaScript actualiza el atributo *Option* del campo correspondiente, y lo rellena con las opciones disponibles en la base de datos.

5.4. Campos Especiales

Los campos de formularios definidos en el estándar de HTML en ocasiones no cubrían completamente las necesidades requeridas por la aplicación. Para estas ocasiones se han utilizado librerías JavaScript específicas para cada caso. Estas librerías generan código HTML,

CSS y JavaScript que se inserta en la estructura de la página creando el campo deseado. A continuación analizaremos la implementación de estas librerías en la aplicación.

- Dual List Box:

Esta librería ha sido utilizada en aquellos campos en los que el usuario debe elegir una o más opciones de entre una lista con múltiples elementos. Esta biblioteca genera dos cajas. En la caja del lado izquierdo aparecen todas las opciones que ofrece al usuario la aplicación para elegir y en la del lado derecho aquellas opciones que ya han sido seleccionadas por él y que serán las que se envíen al servidor para su guardado en la base de datos. Además, la librería incluye otras ventajas como un buscador para cada una de las cajas, que permite filtrar entre las opciones para facilitar la tarea y dos botones sobre cada uno de los listados para *Seleccionar Todo* y *Deseleccionar Todo*. Esta biblioteca ha sido utilizada en varias páginas de la aplicación. En el fragmento 5.5 el código utilizado en una de ellas para su implementación:

```
<label class="col-md-12" for="resto-trabajadores"><u>{% trans "Trabajadores del
    proyecto" %}</u></label>
<br><br>
<select id="dual_list_trabajadores" multiple="multiple" name="trabajadores">
   {% for empleado in lista_empleados %}
       <option value="{{empleado.nombre}}">{{empleado.oficio}} | {{empleado.
           nombre}}</option>
   {% endfor %}
</select>

<script type="text/javascript">
        $('#dual_list_trabajadores').bootstrapDualListbox({
                filterTextClear: "Mostrar todos",
                filterPlaceHolder: "Buscar",
                selectedListLabel: "Trabajadores en el Proyecto",
                nonSelectedListLabel : "Todos los Trabajadores",
                selectorMinimalHeight: 200,
                infoTextEmpty: "Ningun usuario seleccionado"
        });
</script>
```

Fragmento 5.5: Código para usar la librería Dual List Box

Como se puede observar, el código para usarla se divide en dos partes, el código HTML que genera el campo del tipo *Select* sobre el cual actuará el código de la biblioteca modificándolo y el código JavaScript que realiza la llamada a la función, que ejecuta el código de la librería, que tiene como atributos algunas variables que permiten personalizar los textos que añade.

Con el uso de esta librería se consigue que el usuario pueda comprobar de una manera muy rápida e intuitiva, aquellas opciones que tiene seleccionadas y tenerlas fácilmente diferenciadas de aquellas que no lo está.

- Bootstrap Select Picker:

Esta biblioteca permite introducir distintas modificaciones y mejoras al campo del tipo *Select* estándar de HTML. Este tipo de campo está destinado a que el usuario elija, de entre múltiples opciones, una o más de ellas. Algunas de estas mejoras que implementa la librería son la posibilidad de añadir un buscador entre las opciones, el diseño gráfico del campo, botones de *Seleccionar Todos* y *Deseleccionar Todos*, etc.

En esta aplicación hemos utilizado esta biblioteca en diversos casos, utilizando principalmente la mejora de añadir el buscador, que resulta de gran utilidad cuando el número de opciones que es ofrecido al usuario es relativamente grande. En el fragmento 5.6 se puede observar un extracto del código utilizado.

```
<select id="piloto" name="piloto" class="form-control">
        <option value="">Ninguno</option>
        {% for piloto in pilotos %}
                {% if piloto.perms == pilot %}
                        <option value="{{ piloto.first_name }}">{{ piloto.
                            first_name | capfirst}}</option>
                {% endif %}
        {% endfor %}
</select>

<script type="text/javascript">
        $('#piloto').selectpicker({
                size: 4,
                liveSearch: true,
                liveSearchPlaceholder: "Buscar Piloto",
                noneSelectedText: "Piloto"
        });
        $('#piloto').selectpicker('refresh');
</script>
```

Fragmento 5.6: Código para usar la librería Bootstrap Select Picker

La implementación es muy similar a la vista en el apartado anterior, con una parte de código HTML donde se define el campo *Select* y una segunda parte con código JavaScript con la llamada a la biblioteca y diferentes parámetros de configuración.

Cabe destacar que tanto esta librería como la anterior hacen uso de *jQuery*.

- noUiSlider:

Otro tipo de campo que tiene definido el estándar HTML es el *Range*. Este tipo esta destinado a que el usuario, mediante un *slider* o deslizador, elija un número de los comprendidos dentro de un rango de valores. Este elemento tal y como viene definido en HTML tiene muchas limitaciones y es por esta razón por la ha sido utilizada esta librería JavaScript.

Dentro de la aplicación, esta biblioteca ha sido utilizada en los formularios que permiten crear y editar un parte de obra. En concreto se utiliza para definir la hora de inicio y la hora de finalización de los intervalos de tiempo durante los que se ha realizado una actividad concreta, como excavar, bajar revestimientos, etc.

Una de las principales ventajas que aporta el uso de esta librería es que permite en un mismo *slider* definir un intervalo de tiempo, pudiendo seleccionar dos horas que delimitarán los dos extremos del periodo de tiempo, la hora en la que comenzó ese intervalo y en la que se dio por finalizado.

Profundizando más en su funcionamiento, el deslizador se adapta a la duración del turno definido en el parte de obra, en primer lugar al turno por defecto definido en la aplicación, de 8:00 a 20:00, y posteriormente, si el usuario modifica este turno, se adaptará al que sea establecido. Por lo tanto, la hora mínima que se le permite al usuario elegir será la hora de inicio del turno sobre el que se asocia el parte de obra, y la hora máxima, será la hora de finalización

del dicho turno. De esta manera, el usuario puede elegir cualquier periodo de tiempo dentro del turno de trabajo definido.

Otro detalle es que el elemento tiene establecida una definición mínima de periodos de cinco minutos. Otro de los añadidos interesantes que nos aporta la biblioteca es la definición de *ToolTips*, que son los carteles que aparecen sobre cada uno de los deslizadores cuando el usuario está interactuando sobre ellos y que contienen la hora sobre la que se sitúa el elemento en ese momento y que permiten elegir la hora adecuada de una manera muy intuitiva.

En el fragmento 5.7 se muestra un extracto del código utilizado para el uso de la librería en la aplicación:

```
<div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
        <div id="Tiempos">
                <div id="slider_Tiempos"></div><br><br>
        </div>
</div>
<script type="text/javascript">
    /*Inicializacion del Slider */
    slider("Tiempos", 0, 0, 144);
    slider = document.getElementById("slider_Tiempos")
    document.getElementsByClassName("noUi-tooltip")[0].style="display:none"
    document.getElementsByClassName("noUi-tooltip")[1].style="display:none"

    /*Eventos para mostrar/ocultar los tooltips del slider*/
    /*Evento para relacionar la hora de inicio del periodo de tiempo con el
       silder*/
    slider.noUiSlider.on('slide', function(){
        slider_values = document.getElementById("slider_Tiempos").noUiSlider.get
            ()
        horas = Math.floor(slider_values[0]/60)
        minutos = Math.round(slider_values[0]%60)
        hora_inicio = document.getElementById("hora_inicio_turno").value.split("
            :")
        hora_act = (parseInt(hora_inicio[0]) + horas)
        if(hora_act >= 24){
            hora_act = hora_act - 24
        }
        minuto_act = (parseInt(hora_inicio[1]) + minutos)
        if((minuto_act == 0) || (minuto_act == 5)){
            minuto_act = "0" + minuto_act
        }
        hora_final = String(hora_act) + ":" + String(minuto_act)
        document.getElementById("timepicker_hora_inicio").value = hora_final
    });
    /*Evento para relacionar la hora de inicio del periodo de tiempo con el
       silder*/

</script>
```

Fragmento 5.7: Código para usar la librería noUiSlider

De manera similar a los dos apartados anteriores, el código se divide en una parte HTML y en otra parte JavaScript, con la particularidad que en este caso, el código JS se complica, ya que, además de la llamada a la función de la librería, es necesario definir eventos de interacción del usuario con el elemento para que se ejecuten diferentes sentencias, como mostrar y ocultar los *tooltips* o mostrar las horas seleccionadas en otros documentos HTML.

Dentro de este apartado es importante mencionar que también se ha desarrollado un visor que permite al usuario tener una visión general del turno al que pertenece el parte de obra que está introduciendo y de todos los intervalos que ha ido introduciendo.

Este visor está conformado por una tabla HTML en forma de cuadrícula en la que se representa la duración del turno. Cada vez que el usuario añada un nuevo intervalo de tiempo, el periodo de tiempo correspondiente a ese intervalo se coloreará en la fila y del color correspondiente a su tipo. De igual manera, al eliminar un intervalo, el periodo correspondiente cambiará su color a blanco.

5.5. Sistema de Autoguardado

El sistema de autoguardado es una funcionalidad implementada en la página que permite guardar automáticamente de forma local toda la información que está introduciendo el usuario sobre el nuevo parte de obra para que, si lo desea, pueda continuar rellenando el parte posteriormente. Los turnos que cubren normalmente los partes son muy extensos y contienen muchos datos, por lo que facilitar el rellenado progresivo de esta información es especialmente útil.

Para el desarrollo de esta funcionalidad se ha utilizado la API *Web Storage* de HTML5 y JavaScript. Esta API permite a los navegadores almacenar información del tipo clave-valor de una forma mucho mas intuitiva que mediante el sistema de *cookie*s. En concreto se ha utilizado el mecanismo *localstorage* con el que persiste la información guardada incluso tras cerrar el navegador.

El código que implementa el sistema de autoguardado se divide principalmente en dos funciones. La primera de ellas que se encarga de guardar localmente toda la información que el usuario va introduciendo. Para ello en primer lugar, se asegura de que el navegador soporta la API *Web Storage* y si es así recorre todos los campos del formulario uno a uno guardando su contenido en la API utilizando como clave el nombre del campo. En el fragmento 5.8 se muestra un trozo del código de esta parte.

```
function Autoguardado(){

    if (typeof(Storage) !== "undefined") {

        localStorage.clear()

        console.log("Autoguardando Parte de Obra")

        // ========================= INFORMACION BASICA ==========================

        localStorage.nombre_proyecto = document.getElementById("nombre_proyecto"
            ).value
        localStorage.tramo = document.getElementById("tramo").value
        localStorage.piloto = document.getElementById("piloto").value
        localStorage.fecha = document.getElementById("fecha").value
        localStorage.turno = document.getElementById("turno").value
        localStorage.referencia_tbm = document.getElementById("referencia_tbm").
            value
        //localStorage.tbm = document.getElementById("tbm").value
        localStorage.contenedor = document.getElementById("contenedor").value
```

Fragmento 5.8: Parte del código de la implementación de la funcionalidad de Autoguardado

Un caso especial es el referente a los campos que especifican los distintos tipos de intervalo (perforación, avería, bajada de tubo y parada). En este caso, habrá tantos campos de cada tipo como intervalos haya de cada clase. Para ello se recurre a bucles para cada tipo de intervalo que iteran tantas veces como el número de elementos que exista en el parte y utilizamos como clave el número de intervalo y el nombre del campo. El código que implementa el sistema de autoguardado para los intervalos del tipo *Parada* se muestra en el fragmento 5.9.

```
num_paradas = document.getElementById("num_paradas").value

localStorage.num_paradas = num_paradas

for (var i = 1; i <= num_paradas; i++) {
    try{

        value = document.getElementById("parada_" + i + "_hora_inicio").value
        localStorage.setItem("parada_" + i + "_hora_inicio" , value);

        value = document.getElementById("parada_" + i + "_hora_final").value
        localStorage.setItem("parada_" + i + "_hora_final" , value);

        value = document.getElementById("parada_" + i + "_tipo_parada").value
        localStorage.setItem("parada_" + i + "_tipo_parada" , value);

        value = document.getElementById("parada_" + i + "_descripcion_parada").
            value
        localStorage.setItem("parada_" + i + "_descripcion_parada" , value);

        console.log("Autoguardado Parada " + i)
    }catch(err){
        console.log("Error al autoguardar la Parada: " + i)
    }
}
```

Fragmento 5.9: Código de implementación de la funcionalidad Autoguardado para los intervalos de tipo Parada

En caso que el navegador no soporte la API *Web Storage* se genera una notificación para avisar al usuario de que esta funcionalidad no está disponible. Para generar esta notificación se utiliza la librería de JavaScript *Notify* (Fragmento 5.10).

```
        // Si el navegador no soporta el Local Storage, generamos una
            notificacion
        // para avisarle que no esta disponible esta funcion.
        $.notify({
            // options
            icon: 'glyphicon glyphicon-warning-sign',
            title: 'Autoguardado:',
            message: 'El sistema de autoguardado no esta disponible en este
                navegador',
            target: '_blank'
        },{
            // settings
            element: 'body',
            position: null,
            type: "danger",
            allow_dismiss: true,
            newest_on_top: false,
```

```
            placement: {
                from: "top",
                align: "center"
            },
            offset: 20,
            spacing: 10,
            z_index: 1031,
            delay: 0,
            timer: 1000,
            mouse_over: null,
            animate: {
                enter: 'animated fadeInDown',
                exit: 'animated fadeOutUp'
            },

            [ ... ]
        });
    }
```

Fragmento 5.10: Código de uso de la librería Notify

Para que este sistema de autoguardado se comporte correctamente la función que acabamos de analizar se debe ejecutar periódicamente. Para ello se ha utilizado la función *setInterval* de JavaScript (Fragmento 5.11) que permite ejecutar cada un determinado periodo de tiempo una función. En este caso, este intervalo de tiempo se ha establecido en un segundo.

```
var interval_autoguardado = setInterval(Autoguardado, 1000)
```

Fragmento 5.11: Código que permite ejecutar periódicamente la función Autoguardado

La segunda función del sistema de autoguardado realiza el proceso inverso. Cuando el usuario accede a un parte guardado localmente, la página que se carga es igual a la que se cargaría al crear un parte de obra nuevo, con la particularidad que todos los campos que ya habían sido previamente rellenados contienen la información guardada. La función que realiza este trabajo es muy similar a la vista anteriormente pero recuperando todos los campos guardados en la API de almacenamiento del navegador e introduciéndole en los campos del formulario. En el fragmento 5.12 un extracto del código de la función a modo de ejemplo, en concreto la parte que guarda la información básica del parte de obra:

```
function Recuperar_Parte(){

    document.getElementById("nombre_proyecto").value = localStorage.
        nombre_proyecto
    document.getElementById("tramo").value = localStorage.tramo
    document.getElementById("piloto").value = localStorage.piloto
    document.getElementById("fecha").value = localStorage.fecha
    document.getElementById("turno").value = localStorage.turno
    document.getElementById("referencia_tbm").value = localStorage.
        referencia_tbm
    //document.getElementById("tbm").value = localStorage.tbm
    document.getElementById("contenedor").value = localStorage.contenedor
```

Fragmento 5.12: Extracto del código de la función que recupera la información previamente guardada

Y también de forma análoga, en el fragmento 5.13 a la función de guardado se utilizan bucles para recuperar los campos de todos los intervalos presentes en el parte de obra:

```
num_paradas_autoguardadas = localStorage.num_paradas
```

```
for (var i = 1; i <= num_paradas_autoguardadas; i++) {
    console.log("Numero de Paradas :" + i)
    if( localStorage.getItem("parada_" + i + "_hora_inicio")){
        GenerarParada("Autoguardado")
    }else{
        ++ document.getElementById("num_paradas").value
        console.log("No existe la Parada:" + i)
    }
    //++ document.getElementById("num_intervalos").value
}
```

Fragmento 5.13: Extracto del código de la función que recupera la información de las Paradas

5.6. Sistema de Notificaciones

Otra de las funcionalidades desarrolladas en esta aplicación es un sistema de notificaciones basado en tecnologías Push. Las tecnologías Push son formas de comunicación entre cliente y servidor cuya principal característica es que la iniciativa en la comunicación la tiene el servidor. En concreto la tecnología Push que se ha utilizado para el desarrollo de esta funcionalidad son los *WebSockets*. Además, para generar la alerta en el interfaz de usuario se han utilizado dos API, la API estándar de HTML5 para crear alertas emergentes en dispositivos de escritorio y la API Web Push Notification para dispositivos móviles que aunque no es estándar es promovido por Google y es soportado por los dos principales navegadores en dispositivos móviles: Chrome y Mozilla Firefox.

En concreto, el sistema de notificaciones es usado en la aplicación para alertar a los usuarios con el rol *Control* de que hay empleados cuyo certificado médico está próximo a caducar y debe renovarse.

El funcionamiento es el siguiente. Cuando un usuario con el rol *Control* inicia sesión en la aplicación, el cliente abre una conexión WebSocket con el servidor. Para ello se ejecuta el código mostrado en el fragmento 5.14

```
<script type="text/javascript">
    socket = new WebSocket("wss://" + document.location.hostname + "/
        notificaciones/");
    socket.onmessage = function(e) {
        Notificaciones(e)
    }
    // Call onopen directly if socket is already open
    if (socket.readyState == WebSocket.OPEN) socket.onopen();
</script>
```

Fragmento 5.14: Código que inicia la conexión mediante Web Sockets con el servidor

De esta manera el servidor recibirá la petición y abrirá la sesión de WebSocket. El servidor guardará un listado con las sesiones abiertas y a todas ellas mandará periódicamente, si es necesario, un mensaje a través de la conexión WebSocket.

El cliente, cuando recibe uno de estos mensajes ejecuta la siguiente función cuyo código podemos ver a continuación y que analizaremos posteriormente. En primer lugar activa el *badge* de notificaciones situado en la barra de navegación de la aplicación, cambiando su color a rojo, y que indica al usuario permanentemente si existen notificaciones pendientes. Además de esto,

el cliente mantiene unas marcas de tiempo para que la periodicidad de las notificaciones de mínimo cinco minutos. El código de esta parte se muestra en el fragmento 5.15.

```
function Notificaciones(message) {
    badge = document.getElementById("notificacion_badge")

    badge.innerHTML = "!"
    badge.style = "background-color:red"

    mensajeParse = JSON.parse(message.data)
    storagemame = "notificacion_" + mensajeParse.id
    if (sessionStorage.getItem(storagemame) == "") {
        sessionStorage.setItem(storagemame, new Date.now())
    }
```

Fragmento 5.15: Activación del *badge* de notificaciones de la barra de navegación

Posteriormente se comprueban estos sellos de tiempo y si el tiempo desde la última notificación es mayor o igual a cinco minutos, se genera la notificación.

Para generar la notificación primero se comprueba que el navegador tiene soporte para la API de Notificaciones de HTML5. Si tienen este soporte, se comprueba si el usuario ha dado permiso a la aplicación para mostrar notificaciones. Si ambas condiciones se cumplen, se realiza la llamada a la API de HTML5 que generará la notificación emergente con las características determinadas como el título y el contenido de la notificación. Si esta llamada a la API falla, será indicativo que la aplicación está funcionando en un dispositivo móvil ya que en navegadores como Chrome, aunque tienen soporte para la API de HTML5 ésta está bloqueada. Por lo tanto, en caso de error, se ejecutará la función *showNotifactions* que contiene el código para lanzar la API *Notifications Push* mencionada anteriormente. El código detallado anteriormente se presenta en el fragmento 5.16.

```
var title = "EmpresaApp"
    var extra = {
        icon: "/static/img/favicon_eurohinca.png",
        body: mensajeParse.content,
        badge: "/static/img/favicon_eurohinca.png",
        requireInteraction: false,
    }

    if (sessionStorage.getItem(storagemame) < (Date.now() - 300000)) {
        sessionStorage.setItem(storagemame, Date.now())
        if (!("Notification" in window)) {
            console.log("Este navegador no tiene soporte para el sistema de
                notifiaciones")
            // Comprobamos si el permiso para emitir notificaciones ya ha sido
                concedido
        } else if (Notification.permission === "granted") {
            // Lanzamos la notificacion a traves de la API de HTML5
            try {
                Notificacion = new Notification(title, extra)
                Notificacion.onclick = function(event) {
                    window.location.pathname = "/notificaciones"
                };
            // En caso de error, significa que el navegador no soporta la api de
                HTML5 e intentamos
            // lanzar la Notificaciones a traves del Service Worker,
                principalmente para Chrome en // Android.
            } catch (error) {
                showNotifications()
```

```
            }
            //showNotifications()
            // Si el permiso aun no ha sido concedido, pedimos permiso al
               usuario
        } else if (Notification.permission !== 'denied') {
            Notification.requestPermission(function (permission) {
            // Si el usuario nos da permiso, lanzamos la notificacion
            if (permission === "granted") {
                Notificacion = new Notification( title, extra)
                Notificacion.onclick = function(event) {
                    window.location.pathname = "/notificaciones"
                };
            } else {
                console.log("Permiso para lanzar notificaciones denegado")
            }
        });
    } else {
        // Si el usuario no nos concede el permiso, no podemos lanzar la
           notificacion.
        console.log("Permiso para lanzar notificaciones denegado")
    }
  }
}
```

Fragmento 5.16: Código que genera las notificaciones emergentes

La función *showNotifactions* (Fragmento 5.17) es similar a la vista anteriormente, el código comprueba los permisos de la aplicación para mostrar notificaciones y en caso positivo, se utiliza la API textitNotifications Push para lanzar la notificación en dispositivos móviles con los atributos definidos.

```
function showNotifications() {
    Notification.requestPermission(function(result) {
        if (result === 'granted') {
            navigator.serviceWorker.ready.then(function(registration) {
                Notificacion = registration.showNotification('EurohincaApp', {
                    body: 'Empleados con documentos proximos a caducar',
                    requireInteraction: false,
                    tag: '/notificaciones',
                    icon: "/static/img/favicon_eurohinca.png",
                    badge: "/static/img/favicon_eurohinca168.png",
                    url: "/notifiaciones",
                });
            });
        }
    });
}
```

Fragmento 5.17: Codigo que genera las notificaciones emergentes en dispositivos móviles

5.7. Responsividad

Uno de los objetivos de este trabajo de fin de grado era que la aplicación fuera multidispositivo, y para ello es indispensable que la interfaz gráfica se adapte correctamente a los distintos tamaños y resoluciones de estos dispositivos. Para ello se ha utilizado el entorno front-end Bootstrap.

Esta librería CSS trabaja para gestionar cómo se deben adaptar los elementos HTML que

componen una página web a los distintos tamaños y resoluciones de los dispositivos, con una parrilla dividida en doce columnas. El programador decide cuántas de estas columnas debe ocupar cada uno de los elementos HTML en función del tamaño de la pantalla. Bootstrap define cuatro tipos de tamaño de pantalla en función de su resolución horizontal: Extra Small, Small, Medium y Large. Eso se implementa mediante clases CSS que se asignan a los elementos HTML. De esta manera se puede definir el lugar y el tamaño que deben ocupar los elementos dentro de una página en función de la pantalla en la que se muestra.

En el fragmento 5.18 se puede ver como ejemplo el código de la página principal de la aplicación donde se puede ver que los cuatro accesos de las secciones principales se les ha asignado su correspondiente clase de CSS para que ajusten su tamaño a los distintos tamaños de pantalla.

```
<div class="panel panel-info" id="mod_panel-heading" style="padding: 5px 0px 5px
    0px"><b>{% trans "Pagina Principal" %}</b></div>
    <div class="col-xs-6 col-sm-6 col-md-3 col-lg-3">
        <a href="/proyectos">
            <div class="panel panel-default" id="mod_panel-default">
                <div class="panel panel-heading" id="mod_panel-heading"><b>{%
                    trans "Proyectos" %}</b></div>
                    <img src='{% static "img/nuevo_proyecto.png" %}' class="img-
                        responsive" alt="Image" />
            </div>
        </a>
    </div>
    <div class="col-xs-6 col-sm-6 col-md-3 col-lg-3">
        <a href="/maquinaria_index">
            <div class="panel panel-default" id="mod_panel-default">
                <div class="panel panel-heading" id="mod_panel-heading"><b>{%
                    trans "Maquinaria" %}</b></div>
                <img src='{% static "img/maquinaria.png" %}' class="img-
                    responsive" style="width:100%; height:100%" alt="Image" />
            </div>
        </a>
    </div>
    <div class="col-xs-6 col-sm-6 col-md-3 col-lg-3">
        <a href="/personal_index">
            <div class="panel panel-default" id="mod_panel-default">
                <div class="panel panel-heading" id="mod_panel-heading"><b>{%
                    trans "Personal" %}</b></div>
                <img src='{% static "img/trabajadores.png" %}' class="img-
                    responsive" style="width:100%" alt="Image" />
            </div>
        </a>
    </div>
    <div class="col-xs-6 col-sm-6 col-md-3 col-lg-3">
        <a href="/administracion">
            <div class="panel panel-default" id="mod_panel-default">
                <div class="panel panel-heading" id="mod_panel-heading"><b>{%
                    trans "Administracion" %}</b></div>
                <img src='{% static "img/administracion.png" %}' class="img-
                    responsive" style="width:100%" alt="Image" />
            </div>
        </a>
</div>
```

Fragmento 5.18: Código con las clases CSS que configuran las responsividad de la página

El resultado final se puede apreciar en las figuras 5.1, 5.2, 5.3 y 5.4 de la página principal de la aplicación realizadas en dispositivos de distintos tamaños y resoluciones:

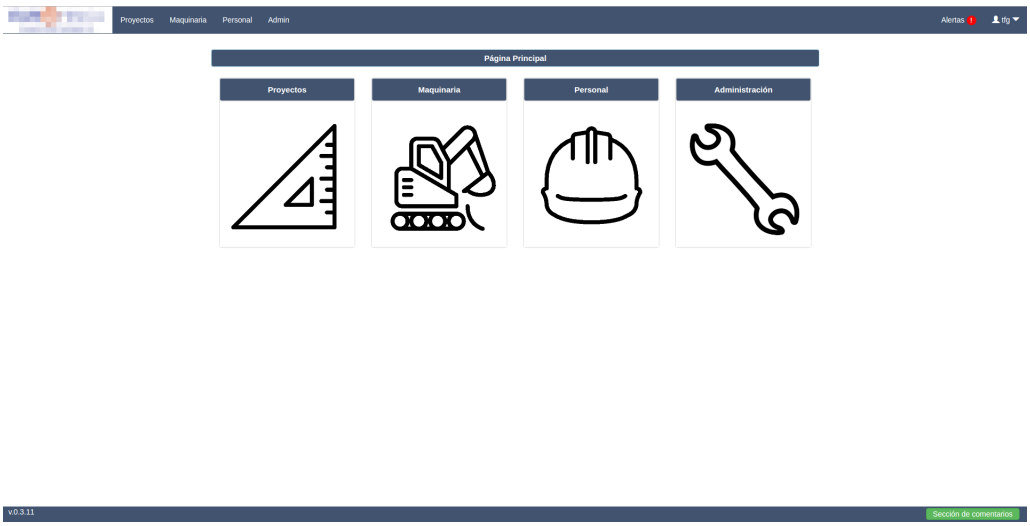


Figura 5.1: Ordenador Portátil, 15,6 pulgadas, Resolución 1920x1080

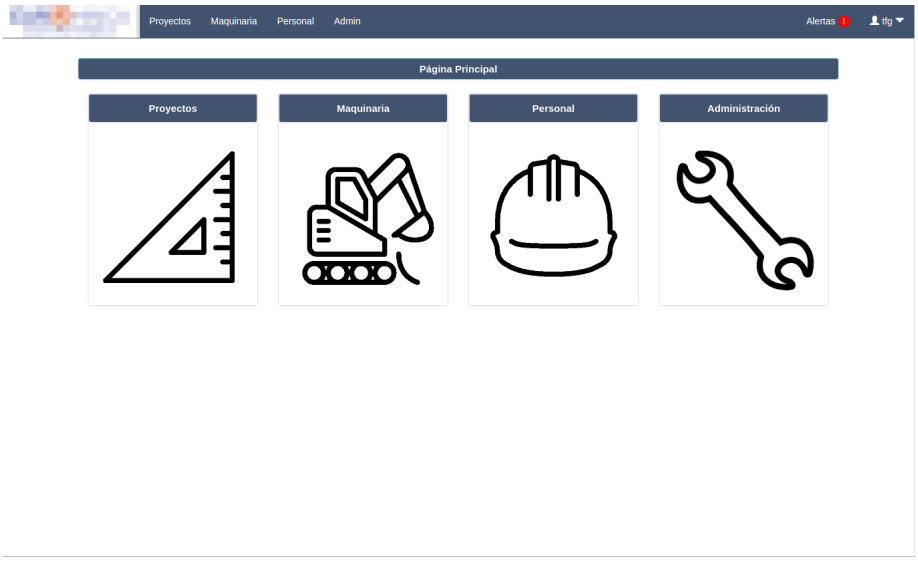


Figura 5.2: Ipad Pro, 12,9 pulgadas, Resolución 1366x1024

Como podemos ver, en aquellas pantallas con un tamaño mayor, se muestran los cuatro accesos en línea, mientras que en aquellas pantallas más pequeñas y con un número de píxeles horizontales inferior, principalmente orientado a dispositivos móviles, los accesos se disponen en forma de cuadrícula dos por dos.

Para esta demostración se han utilizado a modo de ejemplo distintos tipos de dispositivos: un ordenador personal, una tablet de gran formato, una tablet pequeño formato y un móvil de tamaño estándar. Como podemos ver, el diseño de la web se adapta perfectamente a los distintos tamaños ajustándose a sus necesidades y siendo totalmente funcional.

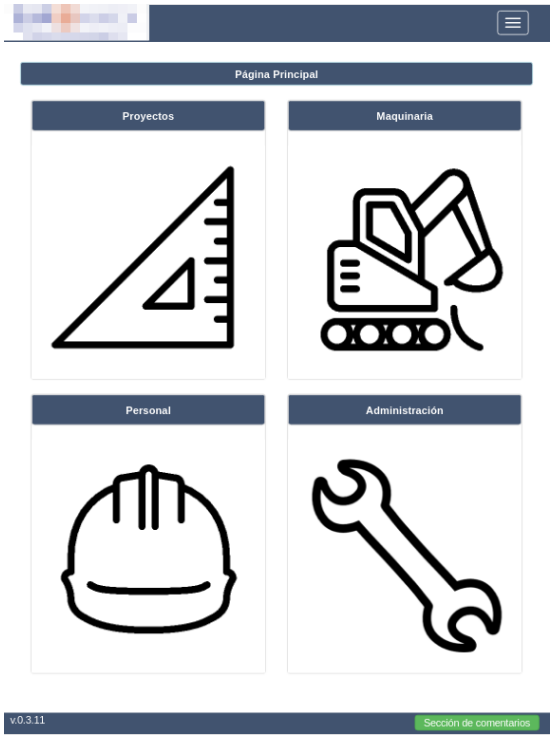


Figura 5.3: Ipad, 9,7 pulgadas, Resolución 768x1024

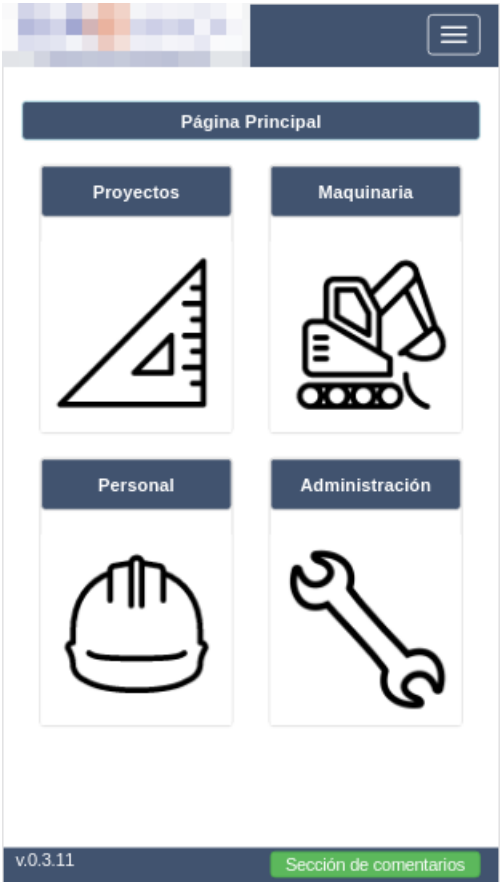


Figura 5.4: Nexus 6P, 6 pulgadas, Resolución 412x732

Capítulo 6

Experimentos

En este capítulo se describen los experimento realizados que validan la parte cliente de la aplicación web. Se pueden dividir en dos grupos: por un lado pruebas de rendimiento y por otro lado la experiencia y opiniones de los usuarios que han llevado a cabo las pruebas y tests de la aplicación.

6.1. Rendimiento temporal de la Aplicación

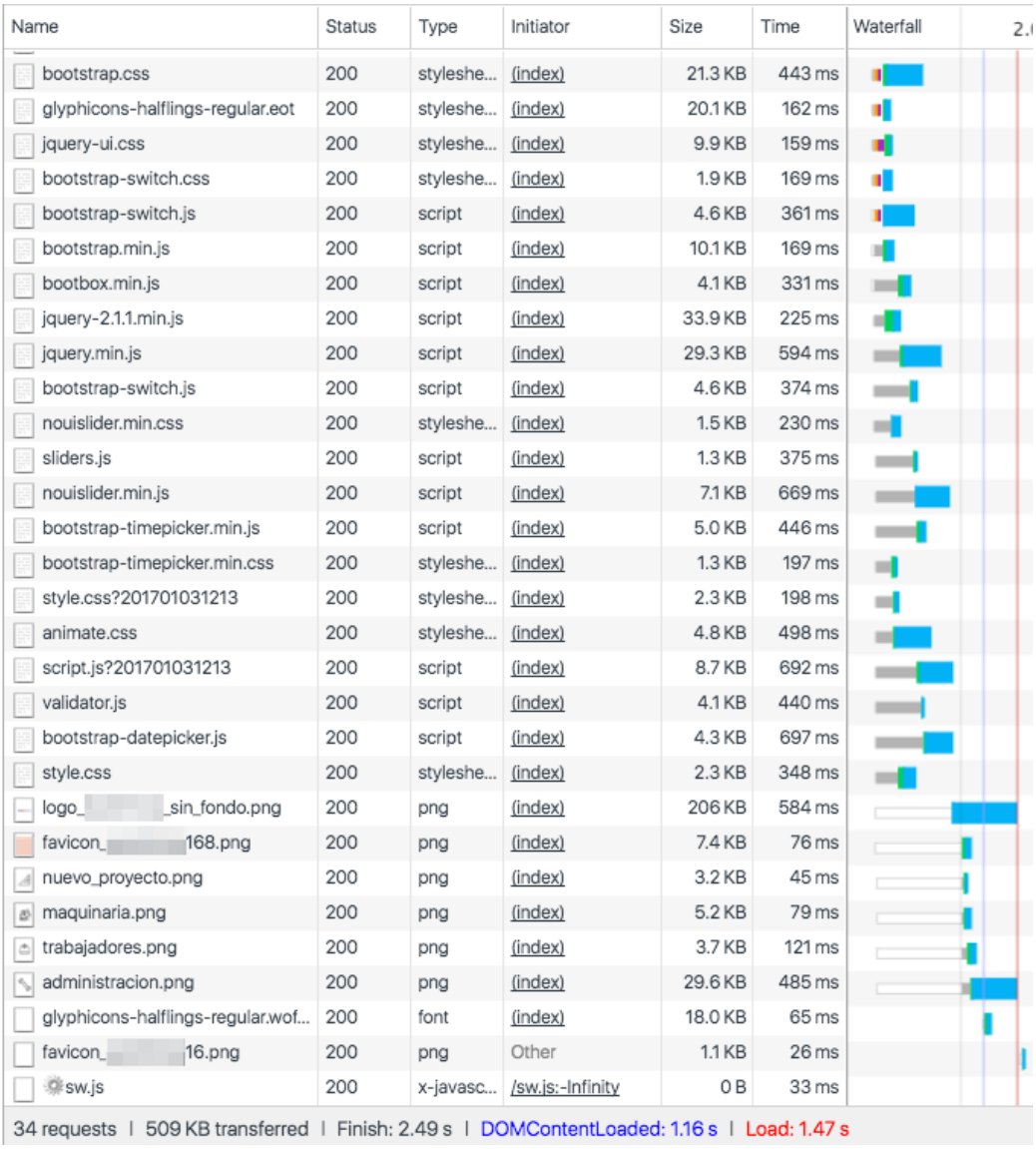
Se ha realizado un experimento para comprobar la velocidad y el rendimiento que es capaz de ofrecer la aplicación en cuatro de los principales navegadores actuales con distintas condiciones tanto de hardware (ordenador portátil, ordenador de sobremesa, *smartphone* y *tablet*) como de conexión a la red. Se han utilizado varios navegadores que cumplen con el estándar HTML5: `Google Chrome`, `Mozilla Firefox`, `Microsoft Edge` y `Opera`.

Figura 6.1: Navegadores Web con los que se ha realizado el experimento

El test se realiza desde tres ubicaciones distintas con diferentes anchos de banda, que han sido medidos con la herramienta en línea de comprobación velocidad de la conexión. Esta herramienta da los siguientes datos de conexión para las siguientes localizaciones:

- **Red Doméstica**: La red del domicilio particular ofrece 32 Mbps de Bajada y 6.55 Mbps de subida.
- **Red de la Universidad**: La red inalámbrica interna de la universidad proporciona 10Mbps de descarga y 20Mbps de subida. Esta red sirve como medida comparativa de referencia con respecto al resto. Usaremos la red inalámbrica para simular un usuario en la red cercana donde se encuentra el servidor
- **Red 4G**: Red móvil con arquitectura 4G con una velocidad de descarga de 28Mbps y una velocidad de subida de 8.27 Mbps.

Para evaluar este rendimiento nos centraremos principalmente en tres parámetros: Número de peticiones, cantidad de información transferida y tiempo total de renderización. Estos



Name	Status	Type	Initiator	Size	Time	Waterfall
bootstrap.css	200	styleshe...	(index)	21.3 KB	443 ms	
glyphicons-halflings-regular.eot	200	styleshe...	(index)	20.1 KB	162 ms	
jquery-ui.css	200	styleshe...	(index)	9.9 KB	159 ms	
bootstrap-switch.css	200	styleshe...	(index)	1.9 KB	169 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	361 ms	
bootstrap.min.js	200	script	(index)	10.1 KB	169 ms	
bootbox.min.js	200	script	(index)	4.1 KB	331 ms	
jquery-2.1.1.min.js	200	script	(index)	33.9 KB	225 ms	
jquery.min.js	200	script	(index)	29.3 KB	594 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	374 ms	
nouislider.min.css	200	styleshe...	(index)	1.5 KB	230 ms	
sliders.js	200	script	(index)	1.3 KB	375 ms	
nouislider.min.js	200	script	(index)	7.1 KB	669 ms	
bootstrap-timepicker.min.js	200	script	(index)	5.0 KB	446 ms	
bootstrap-timepicker.min.css	200	styleshe...	(index)	1.3 KB	197 ms	
style.css?201701031213	200	styleshe...	(index)	2.3 KB	198 ms	
animate.css	200	styleshe...	(index)	4.8 KB	498 ms	
script.js?201701031213	200	script	(index)	8.7 KB	692 ms	
validator.js	200	script	(index)	4.1 KB	440 ms	
bootstrap-datepicker.js	200	script	(index)	4.3 KB	697 ms	
style.css	200	styleshe...	(index)	2.3 KB	348 ms	
logo_ _sin_fondo.png	200	png	(index)	206 KB	584 ms	
favicon_ 168.png	200	png	(index)	7.4 KB	76 ms	
nuevo_proyecto.png	200	png	(index)	3.2 KB	45 ms	
maquinaria.png	200	png	(index)	5.2 KB	79 ms	
trabajadores.png	200	png	(index)	3.7 KB	121 ms	
administracion.png	200	png	(index)	29.6 KB	485 ms	
glyphicons-halflings-regular.wof...	200	font	(index)	18.0 KB	65 ms	
favicon_ 16.png	200	png	Other	1.1 KB	26 ms	
sw.js	200	x-javasc...	/sw.js:-Infinity	0 B	33 ms	

34 requests | 509 KB transferred | Finish: 2.49 s | DOMContentLoaded: 1.16 s | Load: 1.47 s

Figura 6.2: Cuadro de peticiones con el navegador Google Chrome en ordenador portátil mediante Red Doméstica

parámetros permiten hacernos una idea del tiempo que tardará la aplicación en poder ser utilizada por el usuario así como de lo pesada o ligera que es ésta. Ambos parámetros son esenciales en una experiencia de usuario óptima.

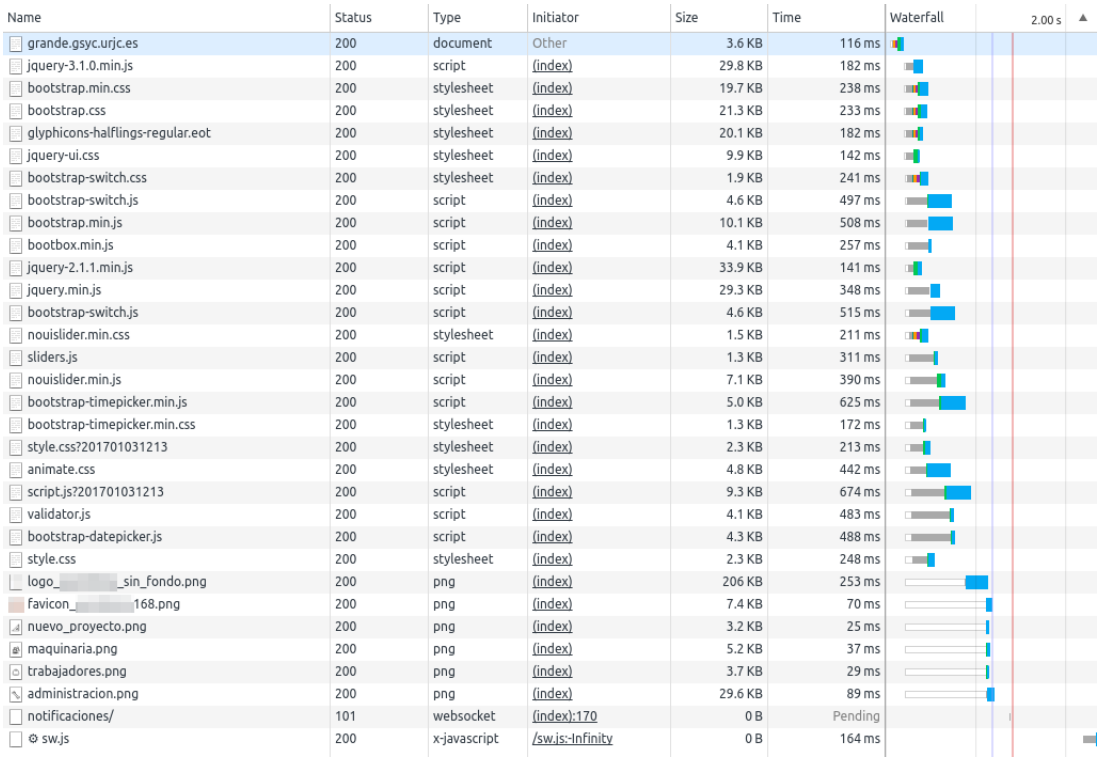
Desde la figura 6.2 hasta la figura 6.12 puede verse un desglose de los archivos solicitados, la cantidad de información transmitida en la petición y el tiempo total que ha tardado en cargarse la página principal de la aplicación para cada uno de los dispositivos a través la Red Doméstica.

- **Google Chrome**: En las figuras 6.2, 6.3, 6.4 y 6.5 el desglose de las peticiones realizadas desde el navegador Google Chrome en los distintos dispositivos.
  - Ordenador portátil (Figura 6.2)
  - Ordenador sobremesa (Figura 6.3)
  - Smartphone (Figura 6.4)
  - Tableta (Figura 6.5)

Name	Status	Type	Initiator	Size	Time	Waterfall
glyphicons-halflings-regular.eot	200	stylesheet	(index)	20.1 KB	76 ms	
jquery-ui.css	200	stylesheet	(index)	9.9 KB	144 ms	
bootstrap-switch.css	200	stylesheet	(index)	1.9 KB	106 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	144 ms	
bootstrap.min.js	200	script	(index)	10.1 KB	161 ms	
bootbox.min.js	200	script	(index)	4.1 KB	136 ms	
jquery-2.1.1.min.js	200	script	(index)	33.9 KB	202 ms	
jquery.min.js	200	script	(index)	29.3 KB	190 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	174 ms	
nouislider.min.css	200	stylesheet	(index)	1.5 KB	106 ms	
sliders.js	200	script	(index)	1.3 KB	173 ms	
nouislider.min.js	200	script	(index)	7.1 KB	213 ms	
bootstrap-timepicker.min.js	200	script	(index)	5.0 KB	185 ms	
bootstrap-timepicker.min.css	200	stylesheet	(index)	1.3 KB	79 ms	
style.css?201701031213	200	stylesheet	(index)	2.3 KB	127 ms	
animate.css	200	stylesheet	(index)	4.8 KB	116 ms	
script.js?201701031213	200	script	(index)	9.2 KB	198 ms	
validator.js	200	script	(index)	4.1 KB	206 ms	
bootstrap-datepicker.js	200	script	(index)	4.3 KB	208 ms	
style.css	200	stylesheet	(index)	2.3 KB	131 ms	
logo_ _sin_fondo.png	200	png	(index)	206 KB	348 ms	
favicon_ 168.png	200	png	(index)	7.4 KB	78 ms	
nuevo_proyecto.png	200	png	(index)	3.2 KB	451 ms	
maquinaria.png	200	png	(index)	5.2 KB	262 ms	
trabajadores.png	200	png	(index)	3.7 KB	30 ms	
administracion.png	200	png	(index)	29.6 KB	318 ms	
notificaciones/	101	websocket	VM928:35	0 B	Pending	
glyphicons-halflings-regular.woff2	200	font	(index)	18.0 KB	59 ms	
favicon_ 16.png	200	png	Other	1.1 KB	19 ms	
⚙ sw.js	200	x-javascri...	sw.js	0 B	22 ms	

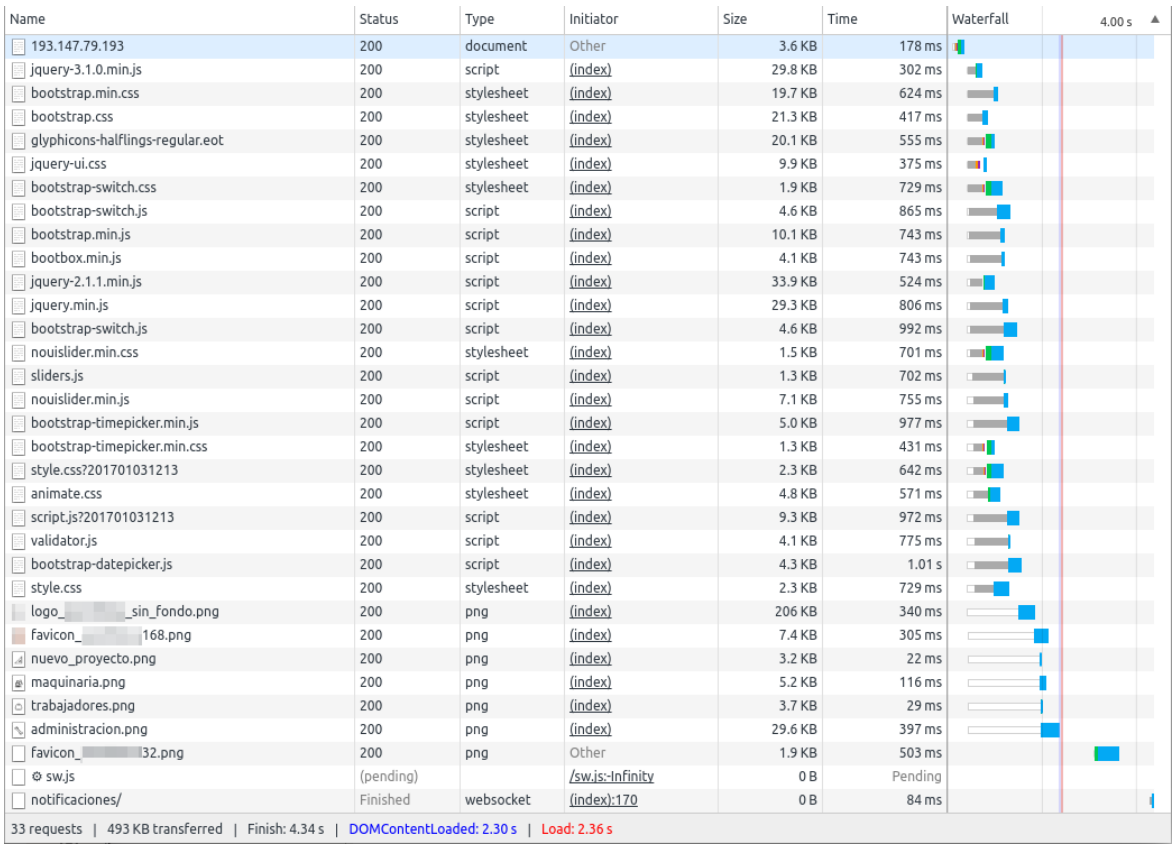
34 requests | 510 KB transferred | Finish: 2.09 s | DOMContentLoaded: 849 ms | Load: 1.08 s

Figura 6.3: Cuadro de peticiones con el navegador Google Chrome en ordenador de sobremesa mediante Red Doméstica



Name	Status	Type	Initiator	Size	Time	Waterfall 2.00 s
grande.gsyc.urjc.es	200	document	Other	3.6 KB	116 ms	
jquery-3.1.0.min.js	200	script	(index)	29.8 KB	182 ms	
bootstrap.min.css	200	stylesheet	(index)	19.7 KB	238 ms	
bootstrap.css	200	stylesheet	(index)	21.3 KB	233 ms	
glyphicons-halflings-regular.eot	200	stylesheet	(index)	20.1 KB	182 ms	
jquery-ui.css	200	stylesheet	(index)	9.9 KB	142 ms	
bootstrap-switch.css	200	stylesheet	(index)	1.9 KB	241 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	497 ms	
bootstrap.min.js	200	script	(index)	10.1 KB	508 ms	
bootbox.min.js	200	script	(index)	4.1 KB	257 ms	
jquery-2.1.1.min.js	200	script	(index)	33.9 KB	141 ms	
jquery.min.js	200	script	(index)	29.3 KB	348 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	515 ms	
nouislider.min.css	200	stylesheet	(index)	1.5 KB	211 ms	
sliders.js	200	script	(index)	1.3 KB	311 ms	
nouislider.min.js	200	script	(index)	7.1 KB	390 ms	
bootstrap-timepicker.min.js	200	script	(index)	5.0 KB	625 ms	
bootstrap-timepicker.min.css	200	stylesheet	(index)	1.3 KB	172 ms	
style.css?201701031213	200	stylesheet	(index)	2.3 KB	213 ms	
animate.css	200	stylesheet	(index)	4.8 KB	442 ms	
script.js?201701031213	200	script	(index)	9.3 KB	674 ms	
validator.js	200	script	(index)	4.1 KB	483 ms	
bootstrap-datepicker.js	200	script	(index)	4.3 KB	488 ms	
style.css	200	stylesheet	(index)	2.3 KB	248 ms	
logo_ _sin_fondo.png	200	png	(index)	206 KB	253 ms	
favicon_ 168.png	200	png	(index)	7.4 KB	70 ms	
nuevo_proyecto.png	200	png	(index)	3.2 KB	25 ms	
maquinaria.png	200	png	(index)	5.2 KB	37 ms	
trabajadores.png	200	png	(index)	3.7 KB	29 ms	
administracion.png	200	png	(index)	29.6 KB	89 ms	
notificaciones/	101	websocket	(index):170	0 B	Pending	
⚙ sw.js	200	x-javascript	/sw.js:-Infinity	0 B	164 ms	

Figura 6.4: Cuadro de peticiones con el navegador Google Chrome en teléfono móvil mediante Red Doméstica



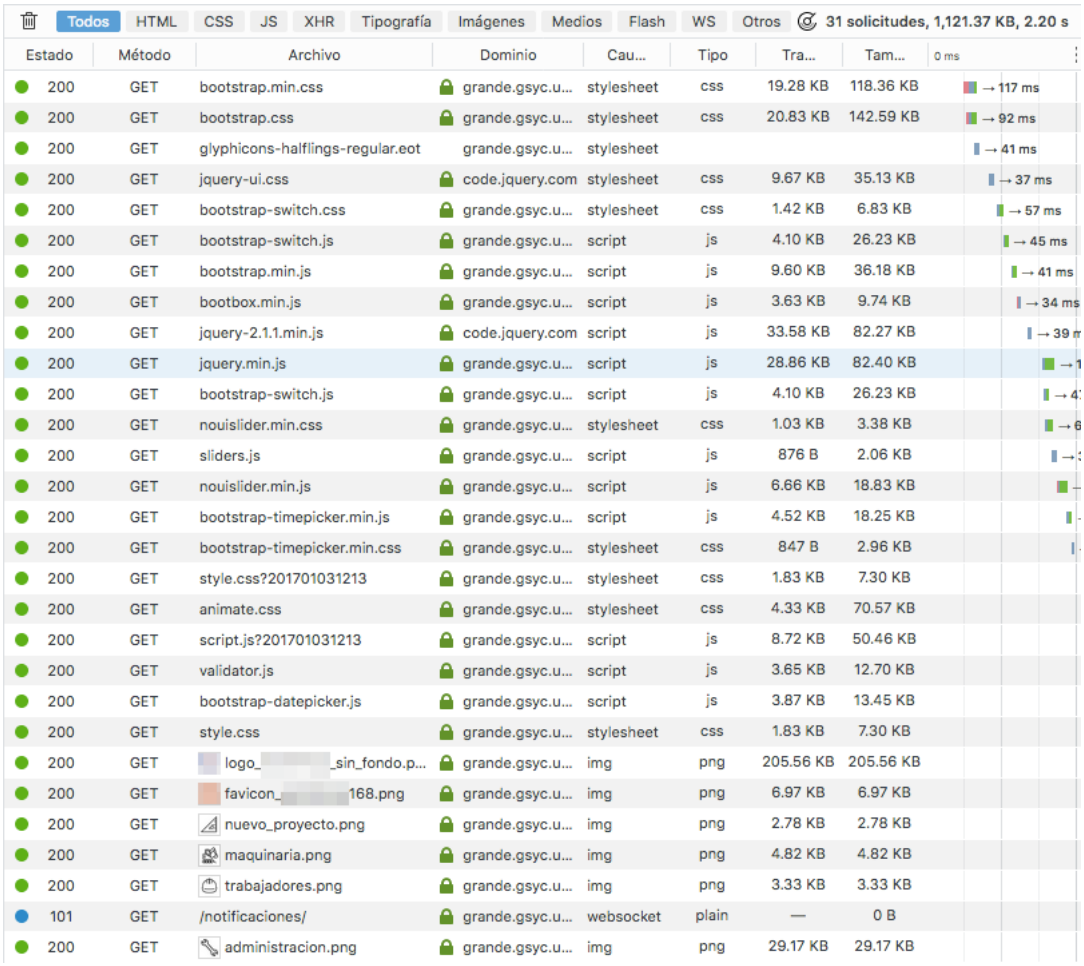
Name	Status	Type	Initiator	Size	Time	Waterfall
193.147.79.193	200	document	Other	3.6 KB	178 ms	
jquery-3.1.0.min.js	200	script	(index)	29.8 KB	302 ms	
bootstrap.min.css	200	stylesheet	(index)	19.7 KB	624 ms	
bootstrap.css	200	stylesheet	(index)	21.3 KB	417 ms	
glyphicons-halflings-regular.eot	200	stylesheet	(index)	20.1 KB	555 ms	
jquery-ui.css	200	stylesheet	(index)	9.9 KB	375 ms	
bootstrap-switch.css	200	stylesheet	(index)	1.9 KB	729 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	865 ms	
bootstrap.min.js	200	script	(index)	10.1 KB	743 ms	
bootbox.min.js	200	script	(index)	4.1 KB	743 ms	
jquery-2.1.1.min.js	200	script	(index)	33.9 KB	524 ms	
jquery.min.js	200	script	(index)	29.3 KB	806 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	992 ms	
nouislider.min.css	200	stylesheet	(index)	1.5 KB	701 ms	
sliders.js	200	script	(index)	1.3 KB	702 ms	
nouislider.min.js	200	script	(index)	7.1 KB	755 ms	
bootstrap-timepicker.min.js	200	script	(index)	5.0 KB	977 ms	
bootstrap-timepicker.min.css	200	stylesheet	(index)	1.3 KB	431 ms	
style.css?201701031213	200	stylesheet	(index)	2.3 KB	642 ms	
animate.css	200	stylesheet	(index)	4.8 KB	571 ms	
script.js?201701031213	200	script	(index)	9.3 KB	972 ms	
validator.js	200	script	(index)	4.1 KB	775 ms	
bootstrap-datepicker.js	200	script	(index)	4.3 KB	1.01 s	
style.css	200	stylesheet	(index)	2.3 KB	729 ms	
logo_ _sin_fondo.png	200	png	(index)	206 KB	340 ms	
favicon_ 168.png	200	png	(index)	7.4 KB	305 ms	
nuevo_proyecto.png	200	png	(index)	3.2 KB	22 ms	
maquinaria.png	200	png	(index)	5.2 KB	116 ms	
trabajadores.png	200	png	(index)	3.7 KB	29 ms	
administracion.png	200	png	(index)	29.6 KB	397 ms	
favicon_ 32.png	200	png	Other	1.9 KB	503 ms	
sw.js	(pending)		/sw.js:-Infinity	0 B	Pending	
notificaciones/	Finished	websocket	(index):170	0 B	84 ms	

33 requests | 493 KB transferred | Finish: 4.34 s | DOMContentLoaded: 2.30 s | Load: 2.36 s

Figura 6.5: Cuadro de peticiones con el navegador Google Chrome en tablet mediante Red Doméstica

- **Mozilla Firefox**: De la misma manera, vemos la tabla de solicitudes y respuesta para el navegador Mozilla Firefox en las figuras 6.6, 6.7, 6.8 y 6.9.
  - Ordenador portátil (Figura 6.6)
  - Ordenador sobremesa (Figura 6.7)
  - Smartphone (Figura 6.8)
  - Tableta (Figura 6.9)
- **Microsoft Edge**: Añadimos a la lista de pruebas el nuevo navegador de Microsoft, `Edge`, para comprobar rendimientos dado que es el más reciente. Para este caso únicamente lo probaremos con el ordenador de sobremesa por tener Windows instalado. En la figura 6.10 las peticiones realizadas por este navegador.
  - Ordenador sobremesa (Figura 6.10)
- **Opera**: Por último, comprobamos los mismos valores para el navegador web Opera. En las figuras 6.11 y 6.12 podemos ver las peticiones efectuadas en cada uno de los dispositivos:
  - Ordenador portátil (Figura 6.11)
  - Ordenador sobremesa (Figura 6.12)

En la tabla 6.1 se puede ver un resumen de todos los datos obtenidos tras realizar el experimento en los distintos dispositivos y navegadores a través de la Red Doméstica. En la tabla 6.2 vemos los resultados obtenidos al cargar la página inicial de la aplicación desde distintos



Todos | HTML | CSS | JS | XHR | Tipografía | Imágenes | Medios | Flash | WS | Otros — 31 solicitudes, 1,121.37 KB, 2.20 s

Estado	Método	Archivo	Dominio	Cau...	Tipo	Tra...	Tam...	0 ms
200	GET	bootstrap.min.css	grande.gsyc.u...	stylesheet	css	19.28 KB	118.36 KB	→ 117 ms
200	GET	bootstrap.css	grande.gsyc.u...	stylesheet	css	20.83 KB	142.59 KB	→ 92 ms
200	GET	glyphicons-halflings-regular.eot	grande.gsyc.u...	stylesheet				→ 41 ms
200	GET	jquery-ui.css	code.jquery.com	stylesheet	css	9.67 KB	35.13 KB	→ 37 ms
200	GET	bootstrap-switch.css	grande.gsyc.u...	stylesheet	css	1.42 KB	6.83 KB	→ 57 ms
200	GET	bootstrap-switch.js	grande.gsyc.u...	script	js	4.10 KB	26.23 KB	→ 45 ms
200	GET	bootstrap.min.js	grande.gsyc.u...	script	js	9.60 KB	36.18 KB	→ 41 ms
200	GET	bootbox.min.js	grande.gsyc.u...	script	js	3.63 KB	9.74 KB	→ 34 ms
200	GET	jquery-2.1.1.min.js	code.jquery.com	script	js	33.58 KB	82.27 KB	→ 39 m
200	GET	jquery.min.js	grande.gsyc.u...	script	js	28.86 KB	82.40 KB	→ 1
200	GET	bootstrap-switch.js	grande.gsyc.u...	script	js	4.10 KB	26.23 KB	→ 4
200	GET	nouislider.min.css	grande.gsyc.u...	stylesheet	css	1.03 KB	3.38 KB	→ 6
200	GET	sliders.js	grande.gsyc.u...	script	js	876 B	2.06 KB	→ 3
200	GET	nouislider.min.js	grande.gsyc.u...	script	js	6.66 KB	18.83 KB	→
200	GET	bootstrap-timepicker.min.js	grande.gsyc.u...	script	js	4.52 KB	18.25 KB	
200	GET	bootstrap-timepicker.min.css	grande.gsyc.u...	stylesheet	css	847 B	2.96 KB	
200	GET	style.css?201701031213	grande.gsyc.u...	stylesheet	css	1.83 KB	7.30 KB	
200	GET	animate.css	grande.gsyc.u...	stylesheet	css	4.33 KB	70.57 KB	
200	GET	script.js?201701031213	grande.gsyc.u...	script	js	8.72 KB	50.46 KB	
200	GET	validator.js	grande.gsyc.u...	script	js	3.65 KB	12.70 KB	
200	GET	bootstrap-datepicker.js	grande.gsyc.u...	script	js	3.87 KB	13.45 KB	
200	GET	style.css	grande.gsyc.u...	stylesheet	css	1.83 KB	7.30 KB	
200	GET	logo_ _sin_fondo.p...	grande.gsyc.u...	img	png	205.56 KB	205.56 KB	
200	GET	favicon_ 168.png	grande.gsyc.u...	img	png	6.97 KB	6.97 KB	
200	GET	nuevo_proyecto.png	grande.gsyc.u...	img	png	2.78 KB	2.78 KB	
200	GET	maquinaria.png	grande.gsyc.u...	img	png	4.82 KB	4.82 KB	
200	GET	trabajadores.png	grande.gsyc.u...	img	png	3.33 KB	3.33 KB	
101	GET	/notificaciones/	grande.gsyc.u...	websocket	plain	—	0 B	
200	GET	administracion.png	grande.gsyc.u...	img	png	29.17 KB	29.17 KB	

Figura 6.6: Cuadro de peticiones con el navegador Firefox en ordenador portátil mediante Red Doméstica

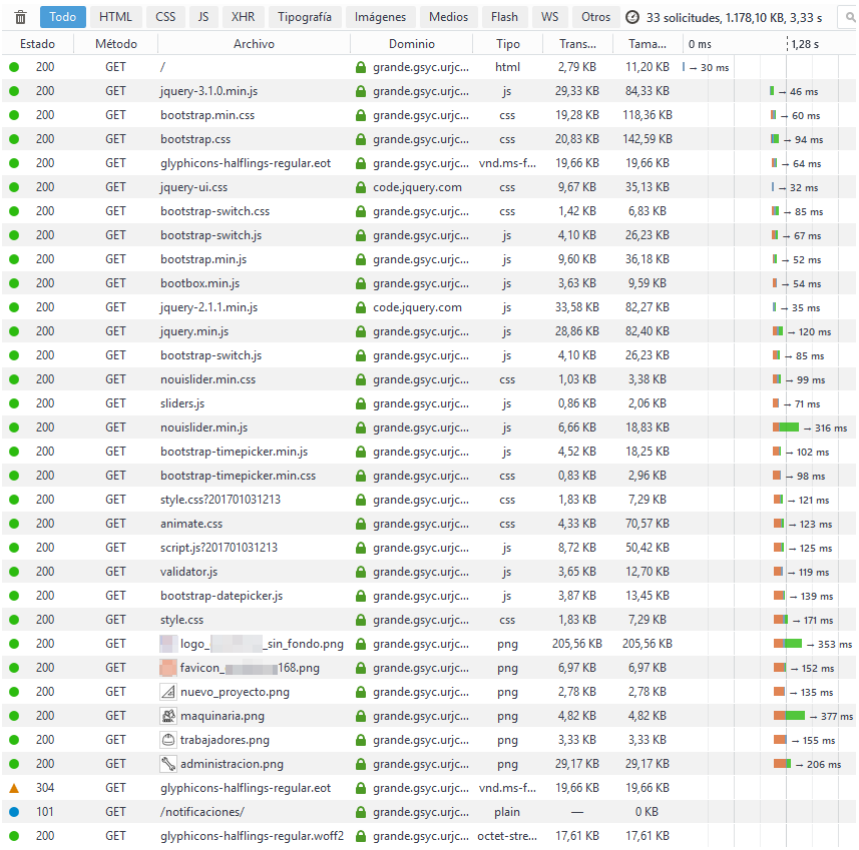


Figura 6.7: Cuadro de peticiones con el navegador Firefox en ordenador de sobremesa mediante Red Doméstica

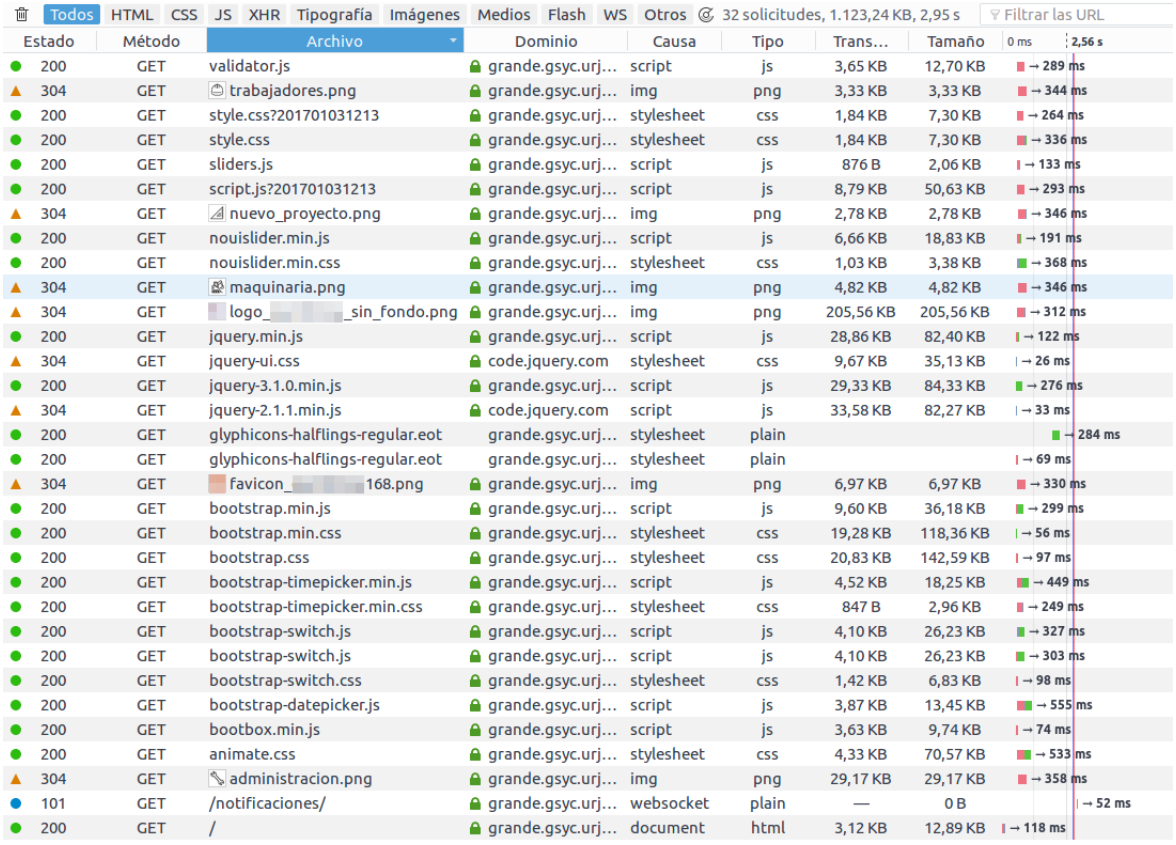


Figura 6.8: Cuadro de peticiones con el navegador Firefox en smartphone mediante Red Doméstica

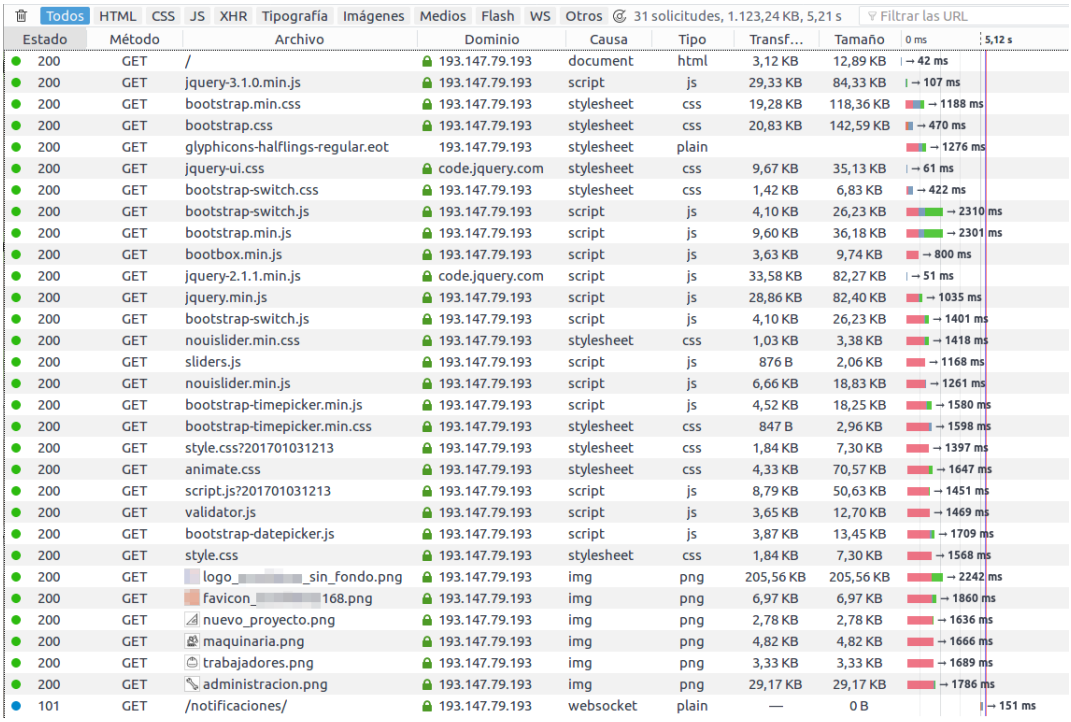


Figura 6.9: Cuadro de peticiones con el navegador Firefox en tableta mediante Red Doméstica

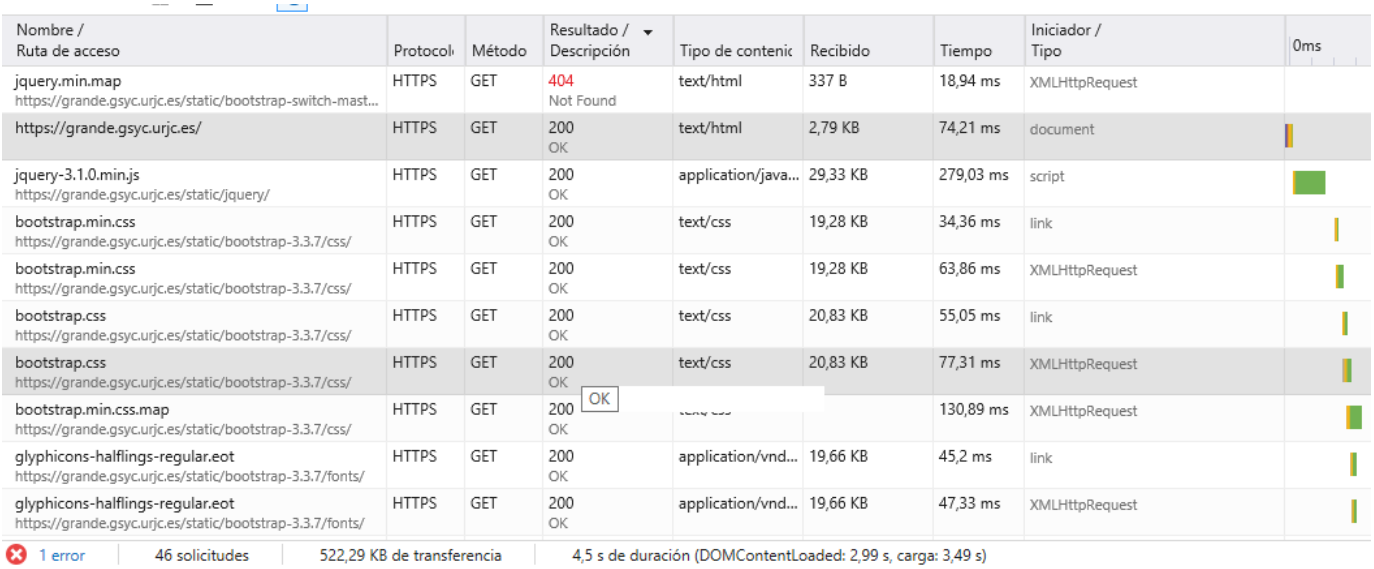


Figura 6.10: Cuadro de peticiones con el navegador Edge en ordenador de sobremesa mediante Red Doméstica

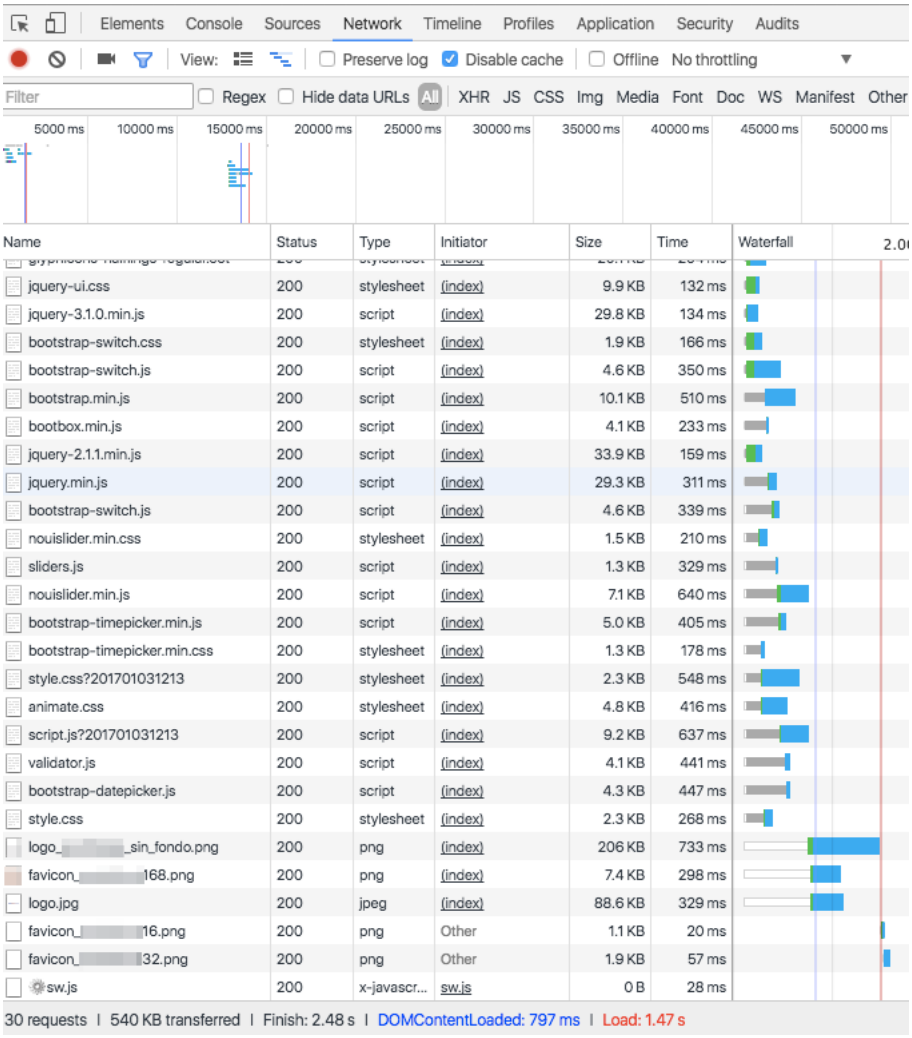
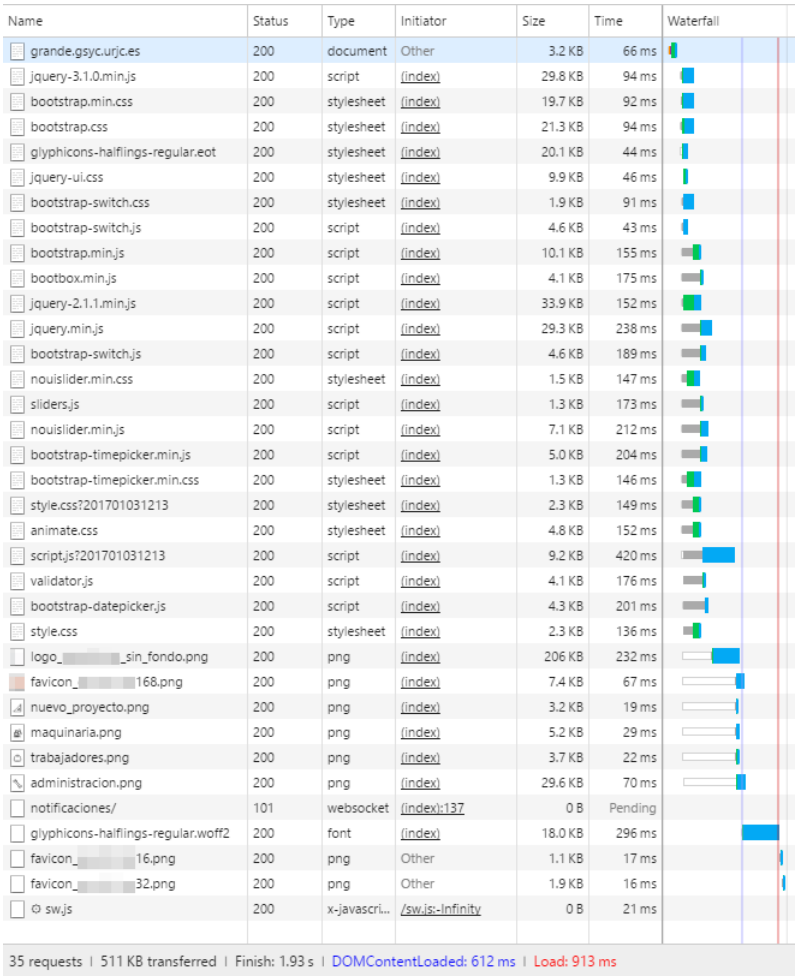


Figura 6.11: Cuadro de peticiones con el navegador Opera en ordenador portátil mediante Red Doméstica



Name	Status	Type	Initiator	Size	Time	Waterfall
grande.gsyc.urjc.es	200	document	Other	3.2 KB	66 ms	
jquery-3.1.0.min.js	200	script	(index)	29.8 KB	94 ms	
bootstrap.min.css	200	stylesheet	(index)	19.7 KB	92 ms	
bootstrap.css	200	stylesheet	(index)	21.3 KB	94 ms	
glyphicons-halflings-regular.eot	200	stylesheet	(index)	20.1 KB	44 ms	
jquery-ui.css	200	stylesheet	(index)	9.9 KB	46 ms	
bootstrap-switch.css	200	stylesheet	(index)	1.9 KB	91 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	43 ms	
bootstrap.min.js	200	script	(index)	10.1 KB	155 ms	
bootbox.min.js	200	script	(index)	4.1 KB	175 ms	
jquery-2.1.1.min.js	200	script	(index)	33.9 KB	152 ms	
jquery.min.js	200	script	(index)	29.3 KB	238 ms	
bootstrap-switch.js	200	script	(index)	4.6 KB	189 ms	
nouislider.min.css	200	stylesheet	(index)	1.5 KB	147 ms	
sliders.js	200	script	(index)	1.3 KB	173 ms	
nouislider.min.js	200	script	(index)	7.1 KB	212 ms	
bootstrap-timepicker.min.js	200	script	(index)	5.0 KB	204 ms	
bootstrap-timepicker.min.css	200	stylesheet	(index)	1.3 KB	146 ms	
style.css?201701031213	200	stylesheet	(index)	2.3 KB	149 ms	
animate.css	200	stylesheet	(index)	4.8 KB	152 ms	
script.js?201701031213	200	script	(index)	9.2 KB	420 ms	
validator.js	200	script	(index)	4.1 KB	176 ms	
bootstrap-datepicker.js	200	script	(index)	4.3 KB	201 ms	
style.css	200	stylesheet	(index)	2.3 KB	136 ms	
logo_[redacted]_sin_fondo.png	200	png	(index)	206 KB	232 ms	
favicon_[redacted]168.png	200	png	(index)	7.4 KB	67 ms	
nuevo_proyecto.png	200	png	(index)	3.2 KB	19 ms	
maquinaria.png	200	png	(index)	5.2 KB	29 ms	
trabajadores.png	200	png	(index)	3.7 KB	22 ms	
administracion.png	200	png	(index)	29.6 KB	70 ms	
notificaciones/	101	websocket	(index):137	0 B	Pending	
glyphicons-halflings-regular.woff2	200	font	(index)	18.0 KB	296 ms	
favicon_[redacted]16.png	200	png	Other	1.1 KB	17 ms	
favicon_[redacted]32.png	200	png	Other	1.9 KB	16 ms	
sw.js	200	x-javascri...	/sw.js:-Infinity	0 B	21 ms	

35 requests | 511 KB transferred | Finish: 1.93 s | DOMContentLoaded: 612 ms | Load: 913 ms

Figura 6.12: Cuadro de peticiones con el navegador Opera en ordenador de sobremesa mediante Red Doméstica

	Solicitudes	Información Transferida	Tiempo total de renderización
Chrome Portátil	34	509Kb	2.49 segundos
Chrome Sobremesa	34	510Kb	2.09 segundos
Chrome Smartphone	32	491Kb	2.71 segundos
Chrome Tableta	33	493Kb	4.34 segundos
Firefox Portátil	31	1121Kb	2.20 segundos
Firefox Sobremesa	33	1178Kb	3.33 segundos
Firefox Smartphone	32	1123Kb	2.95 segundos
Firefox Tableta	31	1123Kb	5.21 segundos
Edge Sobremesa	46	522Kb	4.5 segundos
Opera Portátil	30	540Kb	2.48 segundos
Opera Sobremesa	35	511Kb	1.93 segundos

Tabla 6.1: Tabla con los datos obtenidos con la red de Doméstica

	Solicitudes	Información Transferida	Tiempo total de renderización
Chrome Portátil	33	492Kb	773 milisegundos
Chrome Sobremesa	33	492Kb	557 milisegundos
Chrome Smartphone	32	491Kb	1.27 segundos
Chrome Tableta	32	491Kb	1.72 segundos
Firefox Portátil	32	1123Kb	1.16 segundos
Firefox Sobremesa	32	1123Kb	1.96 segundos
Firefox Smartphone	32	1123Kb	2.36 segundos
Firefox Tableta	32	1123Kb	3.67 segundos
Edge Portatil	46	522Kb	3.4 segundos
Opera Portátil	34	494Kb	686 milisegundos
Opera Sobremesa	33	492Kb	838 milisegundos

Tabla 6.2: Tabla con los datos obtenidos con la red de la universidad

dispositivos y navegadores pero en esta ocasión desde la Red de la Universidad. En la tabla 6.3 se presentan los datos obtenidos a través de la red móvil 4G. Esta red es especialmente interesante ya que este tipo de red es frecuentemente utilizada desde dispositivos móviles actualmente. Por último, en la tabla 6.4 los datos obtenidos a través de la red doméstica pero en esta ocasión utilizando la memoria caché de los navegadores.

A la vista de los resultados obtenidos, con respecto a los navegadores utilizados para el experimento, vemos que el navegador `Opera` es el más rápido en hacer las solicitudes al servidor y renderizar la página para su visualización. Por contra, el navegador de Microsoft es el que más lentamente realiza el proceso. Como navegadores a la vanguardia de los estándares, `Chrome`, `Firefox` y `Opera` son los óptimos para la aplicación.

Como conclusión global podemos extraer que salvo alguna excepción, los tiempos de carga de la aplicación son similares en los navegadores más utilizados en la actualidad, y que éstos son bastante aceptables incluso sin hacer uso de la memoria caché (aproximadamente 2-3 segundos) para una aplicación de este tipo y que si hacemos uso de ella los tiempos de carga disminuyen significativamente (salvo excepciones, a aproximadamente 1-2 segundos) por lo que ofrecerán una experiencia al usuario buena.

Caso especial son los datos obtenidos mediante la red de la universidad. Estos tiempos son

	Solicitudes	Información Transferida	Tiempo total de renderización
Chrome Portátil	33	492Kb	2.94 segundos
Chrome Sobremesa	33	492Kb	3.15 segundos
Chrome Smartphone	33	493Kb	3.72 segundos
Chrome Tableta	32	491Kb	4.96 segundos
Firefox Portátil	33	1100Kb	2.68 segundos
Firefox Sobremesa	32	1110Kb	2.53 segundos
Firefox Smartphone	32	1123Kb	3.22 segundos
Firefox Tableta	32	1123Kb	4.38 segundos
Edge Sobremesa	46	544Kb	4.1 segundos
Opera Portátil	34	494Kb	2.80 segundos
Opera Sobremesa	33	448Kb	2.49 segundos

Tabla 6.3: Tabla resumen con los datos obtenidos a través de la red móvil 4G

	Solicitudes	Información Transferida	Tiempo total de renderización
Chrome Portátil	33	193Kb	575 milisegundos
Chrome Sobremesa	33	79.9Kb	1.08 segundos
Chrome Smartphone	33	3.6Kb	1.17 segundos
Chrome Tableta	32	3.6Kb	1.64 segundos
Firefox Portátil	32	458Kb	928 milisegundos
Firefox Sobremesa	31	458Kb	1.06 segundos
Firefox Smartphone	32	458Kb	3.22 segundos
Firefox Tableta	32	458Kb	2.34 segundos
Edge Sobremesa	46	162.8Kb	3.3 segundos
Opera Portátil	34	71.6Kb	1.09 segundos
Opera Sobremesa	33	71.6Kb	706 milisegundos

Tabla 6.4: Tabla resumen con los datos optenidos utilizando la memoria caché

mucho más bajos, rondando 1 segundo, que los medidos a través de las otras dos redes. Esto es debido a que la red desde la que se hicieron las peticiones es la misma red en la se sitúa el propio servidor de la aplicación por lo que los tiempos de transmisión de los archivos se reducen radicalmente.

Con respecto a la cantidad de información transferida ésta es menos constante, alrededor de 500Kb sin memoria caché, y de menos de 100Kb para la mayoría de los navegadores, valores también bastante aceptables para este proyecto.

6.2. Experiencia de Usuario

Este experimento tiene como objetivo fundamental recoger la opinión y comentarios de los usuarios reales de la aplicación para evaluar el grado de aceptación que ha generado, así como determinar aquellos aspectos a mejorar o posibles funcionalidades a incluir en el futuro.

Para llevar a cabo este experimento se han utilizado dos herramientas diferentes. En primer lugar, se ha solicitado al grupo de usuarios que han colaborado durante todo el desarrollo de la aplicación y posterior periodo de *testing* la redacción de un informe en el que nos ofrecieran, desde su punto de vista, los puntos positivos y negativos de la aplicación, así como posibles mejoras que añadirían. Estos usuarios, al haber estado involucrados durante el desarrollo, conocen ampliamente la aplicación, por lo que consideramos que su opinión es especialmente importante.

En este informe manifiestan con respecto a la interfaz de usuario, que facilita el uso de la aplicación, al tener un diseño *"facil e intuitivo, permitiendo el cambio de campo dentro de la interfaz con mucha facilidad"*. Consideran que la interfaz se adapta correctamente a los distintos niveles de usuarios *"impidiendo posibles problemas que pudieran ocasionar usuarios de rango inferior y siempre corregibles desde usuarios de rango superior"*. Por último remarcan que la interfaz de la aplicación en dispositivos móviles, como *smartsphones* o tabletas, está *"satisfactoriamente desarrollada, permitiendo realizar perfectamente las mismas tareas que en la versión de escritorio"*.

En relación al manejo de la aplicación, manifiestan que *"el manejo sencillo de la misma era uno de los puntos clave para su realización, dado que sería utilizada por todo tipo de usuario, muchos de ellos sin conocimientos extensos en informática"*. Respecto a este objetivo expresan que se ha sido satisfecho ya que *"se ha conseguido estructurar la misma de manera sencilla y amena, realizando la introducción de los partes por pasos, haciendo su introducción simple y rápida"*.

Como conclusión de este informe podemos extraer que aquellos usuarios piloto que han ayudado en el desarrollo quedan satisfechos con el trabajo realizado ya que consideran que se adapta correctamente a la idea inicial que tenía la empresa de la aplicación y que el resultado final de este proyecto permite un uso simple e intuitivo lo que ayudará en gran medida en la implantación con los usuarios finales.

En segundo lugar, se ha creado una breve encuesta destinada a los usuarios finales que han comenzado a utilizar la aplicación en su día a día y el objetivo de la encuesta es medir su valoración de ella, así como saber cuál ha sido el impacto que ha tenido tanto en la tarea principal de crear Partes de Obra como en tareas auxiliares como la de consultar las bases de

datos de Personal y Maquinaria.

Los resultados arrojados por esta en cuesta son, en general, muy positivos. Concretamente los usuarios finales consideran que la aplicación, tanto globalmente, como en concreto su interfaz son, *"Buenas"*. En segundo lugar expresan que la aplicación les ayuda *"Bastante"* en la tarea de creación de Partes de Obra y que les ayuda *"Mucho"* en tareas como la consulta de personal y maquinaria a través de ella. Para finalizar, se les proporcionó un campo donde podían introducir, mediante texto, de qué formas se podría mejorar la aplicación. En este punto la mayoría de las aportaciones ha sido para reportar errores menores que ya han sido solventados en la nueva versión.

Como conclusión de este capítulo podemos decir que se ha cumplido su cometido de medir, desde el punto de vista de los usuarios finales, la valoración del proyecto. De los resultados podemos extraer que el grado de aceptación de la aplicación es muy bueno y que cumple ampliamente con las espectactivas de la empresa y que ayudará en gran medida en el trabajo habitual de los usuarios finales. Como conclusión extra, la aplicación está bien pulida y no presenta prácticamente errores en su funcionamiento, ya que el reporte en la última fase de errores ha sido prácticamente nulo.

Capítulo 7

Conclusiones

En los capítulos anteriores se ha analizado en profundidad el Trabajo de Fin de Grado. En primer lugar se proporcionó una introducción a las tecnologías web y a su estado actual. Posteriormente se definieron los objetivos y requisitos que se buscaban solventar con este proyecto así como la metodología y plan de trabajo seguidos durante su gestación. En el tercer capítulo se especifican las herramientas utilizadas para su desarrollo. En el capítulo cuarto se da una extensa visión del interfaz de usuario y sus funcionalidades mientras que en el capítulo cinco se detalla cómo se han programado esas funcionalidades. El capítulo sexto contiene los experimentos realizados sobre el software desarrollado. Para finalizar se expondrán las conclusiones que podemos obtener del proyecto así como los trabajos futuros que se podrían plantear partiendo de él.

7.1. Aportes Principales

Este Trabajo de Fin de Grado queda enmarcado dentro de un proyecto industrial para una empresa constructora, cuyo su objetivo era crear una aplicación que permitiera a los usuarios la gestión digitalizada de partes de obra. En concreto este TFG tiene como cometido desarrollar la parte cliente de esta aplicación.

Podemos concluir que se han cumplido tanto el objetivo principal del Trabajo de Fin de Grado como los subojetivos en los que se divide éste, que se definieron en el capítulo 2. Se ha desarrollado una aplicación web para la recogida de partes diarios de obra plenamente funcional que a día de hoy lleva más de seis meses en producción. La aplicación ha sido desarrollada en comunicación continua con la empresa que lo que ha permitido que el resultado final de la aplicación se adapte muy bien a los requisitos del cliente.

Actualmente está en fase de implantación por parte la empresa y ya tiene registrados más de veinte usuarios finales y diez proyectos, y está siendo usada a diario para la creación de los partes de obra reales de algunos de éstos.

Este objetivo principal se dividió en los siguientes subobjetivos, que han sido alcanzados con éxito el:

- Es capaz de recoger y almacenar todos los datos necesarios y de generar, de manera automática, el Parte de Obra, tanto en formato .xlsx como .pdf manteniendo el diseño que venía usando la empresa hasta este momento.

- Es multiplataforma y se adapta de manera óptima a distintos tamaños y resoluciones de pantalla siendo totalmente usable en ordenadores de sobremesa, portátiles, teléfonos inteligentes y tabletas.
- Tiene un Sistema de Notificaciones que cumple las características especificadas por los usuarios.
- Cuenta con un Sistema de Autoguardado local que facilita al usuario la tarea de crear un nuevo Parte de Obra en varias sesiones.

Por último, y basándonos en los experimentos realizados en el capítulo anterior, también podemos concluir que cumple con los requisitos de aplicación simple de usar e intuitiva.

El proyecto, en su parte cliente, está compuesto por más de 14000 líneas de código.

7.2. Trabajos Futuros

Este Trabajo de Fin de Grado tiene como principal potencial la gran cantidad de datos que se irán almacenando sobre muchos aspectos de la empresa, como la maquinaria, los trabajadores, averías, consumos, etc. Una línea futura atractiva es crear un software de análisis de datos utilizando técnicas de Big Data si se cree conveniente, con la que extraer información valiosa con el fin de optimizar distintos procesos de la empresa

También se podría incluir un sistema de reportes que permita generar informes agregados sobre distintos aspectos, como maquinaria, empleados, etc, de manera automática.

En tercer lugar se podrían integrar otras bases de datos pertenecientes a la empresa y manejar únicamente a través de esta misma aplicación web otros procesos internos relacionados con gestión personal o de equipos.

Bibliografía

[1] Proyecto JdeRobot http://jderobot.org/

[2] World Web Consortium (W3C) https://www.w3.org/

[3] Especificaciones HTML5 https://developer.mozilla.org/es/docs/HTML/HTML5

[4] Página oficial JavaScript https://www.javascript.com/

[5] Documentación DOM https://developer.mozilla.org/es/docs/Referencia_DOM_de_Gecko

[6] Guía básica CSS http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo

[7] Librería noUiSlider https://refreshless.com/nouislider/

[8] Bootstrap http://getbootstrap.com/

[9] Página oficial jQuery https://jquery.com/

[10] Librería Dual List Box https://www.virtuosoft.eu/code/bootstrap-duallistbox/

[11] Bootstrap Select Picker https://silviomoreto.github.io/bootstrap-select/

[12] Librería noUiSlider https://refreshless.com/nouislider/

[13] Libro de GitHub https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones

[14] Django Channels, Librería para WerSocket https://channels.readthedocs.io/en/stable/