

# Documentación

## Proyecto Final de Ciclo 2018



I.E.S. Enrique Tierno Galván

Tareas para alumnos de 1º E.S.O.

Actividad o tarea	Asignatura	Fecha	Horas	Actividad
Problema	Problema	2018-09-10	1h	1º B
Problema	Asignatura	2018-09-10	1h	1º B
Examen de ingreso	Examen de ingreso	2018-09-10	Examen de ingreso	1º B
Examen de ingreso	Examen de ingreso	2018-09-10	Examen de ingreso	1º B
Examen de ingreso	Examen de ingreso	2018-09-10	Examen de ingreso	1º B

**Daniel Gutiérrez Arias**

**Listado de Tareas**



## ÍNDICE DE CONTENIDO

1.Introducción.....	4
1.1.Objetivo del proyecto.....	4
1.2.¿Por qué?.....	4
1.3.¿Cómo funciona?.....	4
2.Herramientas utilizadas.....	5
2.1.Programas.....	5
2.2.Librerías.....	5
3.Instalación.....	6
4.Estructura de archivos.....	6
5.Evolución.....	7
5.1.Concepto inicial.....	7
5.2.Primer versión.....	8
5.3.Segunda versión.....	9
5.4.Tercera versión.....	10
5.5.Cuarta versión.....	13
6.Base de datos.....	14
6.1.Cursos. SQL Dump.....	15
6.2.Tareas. SQL Dump.....	15
7.Usabilidad.....	16
8.Accesibilidad.....	16
9.Responsive Web Design.....	17
10.Páginas consultadas.....	18

## 1. INTRODUCCIÓN

### 1.1. OBJETIVO DEL PROYECTO

El proyecto consiste en añadir una sección a la nueva intranet que permita a los alumnos ver las fechas de entrega de tareas, trabajos y exámenes correspondientes al curso que están estudiando.

Estas tareas podrán ser añadidas, modificadas y borradas por el administrador o por un usuario con rol de profesor.

### 1.2. ¿POR QUÉ?

A lo largo de mi estancia en el instituto, observé que cuando un profesor fijaba una fecha de examen, el delegado de la clase escribía esa información en un papel pegado a la puerta del aula. Dado que una de las opciones de desarrollo de proyectos era la creación de una sección de la nueva intranet, pensé que podría ser buena idea añadir una sección donde se pudiera consultar esta información en todo momento fuera del aula, por ejemplo por si un alumno faltara a clase un día. O por si un profesor, a la hora de decidir cuándo poner sus exámenes, quiere consultar qué otras tareas de otras asignaturas tiene asignadas un curso.

### 1.3. ¿CÓMO FUNCIONA?

A la hora de consultar una tarea, el usuario deberá registrarse y acceder con sus credenciales al acceso restringido de la intranet. Posteriormente, deberá seleccionar la opción de “Lista de tareas”, que mostrará todos los cursos disponibles que imparte el centro. Una vez seleccionado el curso, se mostrará el listado de las tareas de dicho curso, con la particularidad de que si el usuario con el que se ha accedido tiene permisos de profesor o de administrador, aparecerán opciones extra para realizar el CRUD de tareas (operaciones de ver, insertar, editar y borrar) y gestionarl as como se desee.

## 2. HERRAMIENTAS UTILIZADAS

### 2.1. PROGRAMAS

- **NetBeans:** Todo el código de la sección tareas ha sido desarrollado en este IDE.
- **MySQL Workbench:** Para la visualización, creación y edición de tablas en base de datos.
- **GitHub:** La plataforma de control de cambios donde todos los desarrolladores de la intranet hemos subido nuestros avances centralizados en un único repositorio.
- **Adobe Photoshop:** Lo he usado para la creación de la portada de este documento. Ha sido el único tratamiento de imágenes en el proyecto, junto con los recortes de pantallazos presentes en este documento.

### 2.2. LIBRERÍAS

- **jQuery:** Una extensión de JavaScript que añade nuevas funcionalidades, facilitando por ejemplo la creación de llamadas Ajax y que ofrece compatibilidad con todos los navegadores.
- **Bootstrap:** Framework de HTML, CSS y JS que facilita la creación de páginas con diseño responsive y para móviles.
- **Twig:** Sistema de plantillas para PHP utilizado para las vistas de la aplicación, que permite hacer operaciones de condiciones, bucles y mostrar variables al cargar la página.
- **Font Awesome:** Librería que ofrece gran cantidad de iconos muy fácil de implementar mediante clases CSS.

### 3. INSTALACIÓN

El apartado de tareas no requiere ninguna consideración específica a la hora de ponerse en marcha con respecto al resto de la aplicación, una vez instalado el servidor Apache con PHP 7 y MySQL. Lo único que se necesita es la creación de las tablas en base de datos, descritas más adelante en el documento.

### 4. ESTRUCTURA DE ARCHIVOS

El apartado tareas consta de la siguiente estructura interna:

- **src/controllers/TareasController.php**

El controlador es la parte principal de la sección, ya que incluye las operaciones básicas del proyecto, tales como la carga de la lista de cursos, la lista de tareas, las operaciones de insertar, editar y borrar.

- **src/servicios/tareas/**

Los servicios contienen funciones que hacen operaciones con objetos enviados por el controlador y envían un resultado.

- **src/dao/tareas/**

El DAO incluye funciones llamadas desde los servicios que hacen operaciones con las tablas en base de datos.

- **css/tareas/**

Los estilos CSS utilizados en el proyecto.

- **js/tareas/**

El código Javascript que se ha utilizado para las operaciones en el lado del cliente.

- **vista/tareas/**

Finalmente, las plantillas Twig que una vez interpretadas y cargadas, generarán el código HTML que verá el usuario final.

## 5. EVOLUCIÓN

### 5.1. CONCEPTO INICIAL

Originalmente, a la hora de realizar el pre-proyecto, mi idea original era que:

- El administrador: pueda ver todos los cursos y editar todas las tareas de cada uno
- Un profesor: pueda ver sólo los cursos en los que está dando clase, y modificar las tareas de estos.
- Un alumno delegado: pueda ver sólo el curso que está cursando, y modificar todas sus tareas.
- Un alumno normal: sólo pueda ver las tareas del curso que está estudiando.

No obstante, esto requería permitir a múltiples usuarios con distintos permisos en lugar de simplemente un usuario por rol (un usuario alumno, un usuario profesor, etc), tal y como estaba planteada la intranet original. En clase se comentó que permitir el registro a un número ilimitado de usuarios implicaría una mayor gestión por parte del administrador de la nueva intranet, al tener por ejemplo que dar de baja cada vez que un alumno o profesor deja el centro. Por eso, se optó por simplificar el proyecto tal y como se ha descrito anteriormente.

Originalmente, el diseño de tablas previsto inicialmente se parecía a esto:

Usuarios			
id	nombre	pass	id_permiso
1	user_admin	(hash)	1
2	user_profe	(hash)	2
3	user_delegado	(hash)	4
4	user_alumno	(hash)	3

Permisos	
id	tipo
1	administrador
2	profesor
3	alumno
4	alumno_delegado

Usuarios-Curso	
id_usuario	id_curso
2	1
2	2
3	1
4	1

Curso	
id	nombre
1	1º ESO
2	2º ESO
3	3º ESO
4	4º ESO

Tarea		
id_curso	nombre	fecha
1	Examen matemáticas	05-03-2018
2	Ejercicios inglés pág 70	10-03-2018
3	Entrega trabajo literatura	15-03-2018
4	Examen biología	20-03-2018

Finalmente, la tabla “Usuarios-Curso” fue desechada y las tablas usuarios y permisos han sido creadas por mis compañeros, por lo que sólo he tenido que crear “Cursos” y “Tareas”, con algunas mejoras y campos añadidos.

## 5.2. PRIMERA VERSIÓN

En la primera versión del proyecto, subida el 2 de Mayo de 2018, aún no se había concretado del todo qué base de datos usar, por lo que hice un boceto inicial de cómo sería el diseño utilizando un DAO simulado. Por ejemplo, la función del DAO que obtenía las tareas de un determinado curso, según el valor que recibía, utilizaba un switch case creando y devolviendo un objeto con unos valores fijos, imitando más o menos los valores que recibiría si se usara una base de datos real.

La funcionalidad se reducía simplemente a poder ver una lista de cursos, y al pinchar en uno de ellos, ver las tareas que contiene. No era posible editar nada.

### Listado de cursos:

#### Tareas para alumnos

Seleccione un curso para mostrar las tareas correspondientes.

- [1º DAW](#)
- [2º DAW](#)
- [1º ESO](#)

(Actualmente se está utilizando un DAO simulado, no conecto aún a la BBDD)

(Sólo he hecho que se visualicen, queda por hacer que si entra un usuario profesor pueda hacer CRUD de las tareas y que un administrador pueda añadir cursos)

### Listado de tareas:

#### Tareas para alumnos de 1º DAW

Descripción de la tarea	Asignatura / Módulo	Fecha
tarea 1	Bases de datos	mañana
tarea 2	Sistemas	pasado mañana
tarea 3	asignatura	fecha

[Volver](#)

### DAO simulado:

```
/* CutreTask Simulator 5.0 */
switch ($curso) {
  case 1:
    $tarea1 = new Tarea();
    $tarea1->descripcion = 'tarea 1';
    $tarea1->asignatura = 'Bases de datos';
    $tarea1->fecha = 'mañana';

    $tarea2 = new Tarea();
    $tarea2->descripcion = 'tarea 2';
    $tarea2->asignatura = 'Sistemas';
    $tarea2->fecha = 'pasado mañana';

    ...
  }
```



### 5.3. SEGUNDA VERSIÓN

El 6 de Mayo, mejoré un poco el diseño en general, e hice una tercera vista para editar tareas.

La idea era que si se tenían permisos de edición de tareas, la columna de editar apareciera, llevando a una página que mostrara todos los campos preparados para ser editados. No estaba del todo seguro de cómo plantearlo y además seguía sin poder usar base de datos, por lo que la funcionalidad seguía siendo nula.

#### Listado de cursos:

Tareas para alumnos					
Seleccione un curso para mostrar las tareas correspondientes.					
Nombre	Tipo	Turno	Ver	Editar	
1º E.S.O.	E.S.O.		Ver	Editar	
2º E.S.O.	E.S.O.		Ver	Editar	
3º E.S.O.	E.S.O.		Ver	Editar	
4º E.S.O.	E.S.O.		Ver	Editar	
1º Bachillerato	Bachillerato		Ver	Editar	
2º Bachillerato	Bachillerato		Ver	Editar	
1º Mantenimiento de vehículos	F.P. Básica	Mañana	Ver	Editar	
2º Mantenimiento de vehículos	F.P. Básica	Mañana	Ver	Editar	

#### Ver:

##### Tareas para alumnos de 1º ESO

Descripción de la tarea	Asignatura / Módulo	Fecha
Examen trigonometría	Matemáticas	2018-05-15
Entrega de comentario de texto	Literatura	2018-05-17
tarea 3	asignatura	2018-06-13

[Volver](#)

#### Editar:

##### Editando: Tareas para alumnos de 1º ESO

	Descripción de la tarea	Asignatura / Módulo	Fecha
<input type="button" value="Borrar"/>	<input type="text" value="Examen trigonometría"/>	<input type="text" value="Matemáticas"/>	<input type="text" value="15/05/2018"/>
<input type="button" value="Borrar"/>	<input type="text" value="Entrega de comentario de texto"/>	<input type="text" value="Literatura"/>	<input type="text" value="17/05/2018"/>
<input type="button" value="Borrar"/>	<input type="text" value="tarea 3"/>	<input type="text" value="asignatura"/>	<input type="text" value="13/06/2018"/>

(Por hacer: conectar a la base de datos con AJAX y poder añadir tareas)

## 5.4. TERCERA VERSIÓN

El 20 de Mayo, las cosas se pusieron interesantes, porque finalmente se acordó usar una base de datos común, por lo que pude empezar a implementar operaciones reales en BD.

La lista de cursos sufrió un ligero cambio, al final decidí no mostrar dos columnas de ver y editar tareas, y que simplemente al pinchar en el nombre del curso te llevara a las tareas del mismo. El modo visualizar y el modo editar lo dictaminaría el permiso del usuario que se ha conectado.

### Tareas para alumnos

Seleccione un curso para mostrar las tareas correspondientes.

Nombre	Tipo	Turno
1º E.S.O.	E.S.O.	
2º E.S.O.	E.S.O.	
3º E.S.O.	E.S.O.	
4º E.S.O.	E.S.O.	
1º Bachillerato	Bachillerato	

Además, así decidí unificar las vistas de visualizar y editar tareas en una sola. Gracias al uso de las plantillas Twig, es muy fácil indicar que ciertas cosas aparezcan o no según el valor de una variable.

El problema es que por entonces, no teníamos aún usuarios ni permisos implementados, por lo que para hacer pruebas, hice que para entrar en modo “edición” hubiera que poner un parámetro determinado en la URL, y puse un link al final de la tabla para ahorrar tiempo de pruebas.

### Tareas para alumnos de 2º DAW

Descripción de la tarea	Asignatura / Módulo	Fecha
Nueva tarea	Proyecto final de curso	2018-06-13

(TODO: mejorar la recarga cutre al añadir tareas. Quizá implemente datatables.)

[Volver](#)

["Obtener permisos de profesor" \(pruebas\)](#)

Al entrar en modo edición, aparece una cuarta columna al final con los botones “editar” y “borrar”, así como una fila al final del todo con un botón para añadir tareas.

### Tareas para alumnos de 2º DAW

Descripción de la tarea	Asignatura / Módulo	Fecha	
Nueva tarea	Proyecto final de curso	2018-06-13	<input type="button" value="Editar"/> <input type="button" value="Borrar"/>
<input type="button" value="Añadir"/>			

Al pinchar en editar, hice que en esa fila concreta, por medio de Javascript, apareciera una etiqueta input en lugar de cada campo editable. En realidad, por cada celda lo que hice fue incluir el texto plano en un span, un input escondido para introducir los datos, y un input hidden que nunca se visualiza con el contenido original del campo.

### Tareas para alumnos de 2º DAW

Descripción de la tarea	Asignatura / Módulo	Fecha	
<input type="text" value="Nueva tarea"/>	<input type="text" value="Proyecto final de curso"/>	<input type="text" value="13/06/2018"/>	<input type="button" value="X"/>
<input type="button" value="Añadir"/>			

Así, al dar a editar, el span con el texto se oculta, y muestra el input. Cuando se vacía un campo y se pincha fuera perdiendo el foco, hice que se rellenara con el valor que tenía ese campo originalmente, para ayudar a que no hubiera valores nulos, a pesar de que ya existen validaciones al enviar por Javascript y en el servidor. Al editar una fila, sólo aparece un botón de cancelar (que restaura los campos a sus versiones originales de la misma forma descrita antes), pero cuando se ha editado un campo, aparece un botón para validar la edición.

Cuando se ha realizado la edición correctamente, aparece un mensaje de alerta preguntando si realmente se desea editar esa fila. Al confirmar, sin recargarse la página aparece una barra horizontal mostrando el resultado de la operación Ajax, y los textos span son actualizados con los nuevos valores.

## Tareas para alumnos de 2º DAW

Se ha modificado la tarea correctamente

Descripción de la tarea	Asignatura / Módulo	Fecha		
Nueva tarea MODIFICADA	Proyecto final de curso	2018-06-13	Editar	Borrar

Al borrar una fila, pregunta con una alerta si deseamos realizar la operación, y si la operación ha sido correcta, se muestra una animación jQuery para que esa fila de la tabla se oculte al usuario. Asimismo, al darle a añadir una fila, aparece suavemente una fila con nuevos campos que rellenar para poder crear.

## Tareas para alumnos de 2º DAW

Descripción de la tarea	Asignatura / Módulo	Fecha		
Nueva tarea	Proyecto final de curso	2018-06-13	Editar	Borrar
<input type="text" value="Nueva tarea"/>	<input type="text" value="Asignatura"/>	<input type="text" value="dd/mm/aaaa"/>	<input type="checkbox"/>	<input type="checkbox"/>

Lo cual derivó en un problema que no había previsto. ¿Qué debería pasar al crear la fila? Internamente, en la carga inicial de la página, al leer la lista de tareas, se recoge el id de cada tarea para que así, al pintar cada fila, cada etiqueta guarde en su id el número de tarea. Pero al crear una nueva tarea, para pintarla en la tabla, aún recogiendo el valor por Ajax del último valor, no tenemos la garantía de que esa ID sea válida si otro usuario simultáneamente está añadiendo tareas.

Incluso, aun si pudiera realmente generar una fila nueva en la vista con la id correcta, no tendría sentido hacer una función Javascript compleja para que pinte una nueva fila editando el DOM de la página, teniendo un sistema de plantillas como Twig ya funcional. Y es más, no podríamos ver si alguien ha introducido tareas después de nuestra última carga de la página, sólo podríamos ver lo que ha hecho nuestro usuario.

Finalmente, lo planteé de forma que esa fila de creación de tareas no tuviera una id concreta, y que al darle a Aceptar, se recargara la página completamente, eliminando esa sensación dinámica de que todos los cambios se realicen sin recargas, como así pasaba al editar y borrar.

También dejé caer la posibilidad de desechar todo el sistema de tablas que tenía hecho e implementar el plugin jQuery “DataTables”, que recoge un objeto JSON y lo muestra automáticamente con posibilidad de añadir filtros, ordenar columnas, etc, pero no quería volver a tener que empezar de cero.

















## 5.5. CUARTA VERSIÓN

Fue entonces cuando pensé “pero claro, al modificar o borrar una fila, tampoco estarías obteniendo resultados reales, ya que sólo reflejas los cambios que ha realizado el usuario que estás usando”, con lo que tenía que conseguir una forma de recargar sólo la tabla de forma dinámica, y a ser posible repintándolo aprovechando Twig, sin regenerar todo por JavaScript.

Y así surgió la idea de separar la tabla a su propia plantilla separada. En la plantilla de la vista de tareas, en lugar de la tabla, habría un div con un include a la plantilla de la tabla, y el Javascript a la hora de terminar de añadir, editar o borrar, lo que haría sería recargar esa plantilla por Ajax y dibujar el resultado html dentro del div. No sé si es la mejor solución, pero al menos elimino ese problema con las recargas.

Así pues, el 3 de Junio supuso una reestructuración masiva del listado de tareas.

### Tareas para alumnos de 1º E.S.O.

Descripción de la tarea ▾	Asignatura / Módulo ▾	Fecha ▾	Prioridad ▾	Acciones
Prueba	Asignatura	2018-05-06	Finalizó hace 34 días	 
Hoy	Matemáticas	2018-06-02	Finalizó hace 7 días	 
Examen tema 2	Literatura	2018-06-03	Finalizó hace 6 días	 
Prueba	Prueba	2018-06-09	Hoy	 
Prueba2	Asignatura	2018-06-10	Mañana	 
Entrega de trabajo	Tecnología	2018-06-13	Quedan 4 días	 
Examen de recuperación	Matemáticas	2018-06-15	Quedan 6 días	 
Ejercicios 1, 2, 3, 4, 6, 8, 10 de página 170 del libro	Asignatura tal	2018-06-30	Quedan 21 días	 
<input type="checkbox"/> Ocultar tareas pasadas		1 / 2	Nº tareas/página: 10	

Volver

Ahora que ya podía recargar la tabla sin miedo, incorporé la funcionalidad de paginación e indicar el número de tareas a mostrar por página, mediante el uso de LIMIT y OFFSET en base de datos, así como la posibilidad de reordenar columnas ascendentemente o descendentemente cuando se hace click en el encabezado. También descubrí que algunos de mis compañeros usaban la librería de iconos

Font Awesome que permite mostrar iconos vectorizados en una página web, así que cambié los botones que tenía originalmente por iconos de dicha librería.

Añadí una columna prioridad que indica de forma más visual el tiempo restante de expiración de la tarea. Cuando la tarea finaliza hoy o mañana, se visualiza en rojo para alertar al alumno rápidamente de que tiene que finalizar su trabajo urgentemente o que un examen está muy próximo. Cuando una tarea forma parte del pasado, no se borra, pero no aparece a primera vista al cargar la página ya que puede distraer a la hora de ver las tareas de urgente vencimiento. No obstante, añadí un checkbox para ocultar/mostrar las tareas anteriores al día actual, las cuales se muestran en gris para distinguirse fácilmente de las que quedan pendientes por realizar. Un usuario con permiso de edición puede decidir si borrar las tareas pasadas, o dejarlas para futura referencia. Al ocultarse automáticamente las tareas pasadas, no debería ser necesario borrar las tareas de años anteriores, con lo que con esta opción se gana en mantenimiento.

## 6. BASE DE DATOS

La funcionalidad del listado de tareas añade dos tablas en base de datos:

- Una tabla con la lista de cursos, con los campos:
  - **id\_curso:** una clave identificativa autoincremental
  - **nombre\_curso:** autoexplicativo
  - **tipo:** para indicar si es un curso de la E.S.O., Bachillerato, F.P. Medio, F.P. Superior, etc
  - **turno:** “m” para mañana, “t” para tarde
- Y otra tabla para la lista de tareas
  - **id\_tarea:** clave identificativa autoincremental
  - **id\_curso:** para saber a qué curso corresponde la tarea
  - **descripcion:** en qué consiste la tarea
  - **asignatura:** una cadena de texto indicando a qué asignatura corresponde
  - **fecha:** fecha de vencimiento de la tarea

## 6.1. CURSOS. SQL DUMP

```
--
-- Table structure for table `cursos`
--

DROP TABLE IF EXISTS `cursos`;

CREATE TABLE `cursos` (
  `id_curso` int(11) NOT NULL AUTO_INCREMENT,
  `nombre_curso` varchar(250) COLLATE utf8mb4_unicode_ci NOT NULL,
  `tipo` varchar(250) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `turno` varchar(1) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id_curso`)
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Dumping data for table `cursos`
--

LOCK TABLES `cursos` WRITE;

INSERT INTO `cursos` VALUES (1,'1° E.S.O.','ESO','m'),(2,'2° E.S.O.','ESO','m'),(3,'3° E.S.O.','ESO','m'),(4,'4° E.S.O.','ESO','m'),(5,'1° Bachillerato','BACH','m'),(6,'2° Bachillerato','BACH','m'),(7,'1° Mantenimiento de vehículos','FPB','m'),(8,'2° Mantenimiento de vehículos','FPB','m'),(9,'1° Mantenimiento de vehiculos','FPB','t'),(10,'2° Mantenimiento de vehículos','FPB','t'),(11,'1° Carrocería','FPGM','m'),(12,'2° Carrocería','FPGM','m'),(13,'1° Instalaciones Eléctricas y Automáticas','FPGM','m'),(14,'2° Instalaciones Eléctricas y Automáticas','FPGM','m'),(15,'1° Instalaciones de Producción de Calor','FPGM',NULL),(16,'2° Instalaciones de Producción de Calor','FPGM','m'),(17,'1° ASIR','FPGS','t'),(18,'2° ASIR','FPGS','t'),(19,'1° Mantenimiento de Instalaciones Térmicas y Fluidos','FPGS','t'),(20,'2° Mantenimiento de Instalaciones Térmicas y Fluidos','FPGS','t'),(21,'1° Automoción','FPGS','t'),(22,'2° Automoción','FPGS','t'),(23,'1° DAW','FPGS','m'),(24,'2° DAW','FPGS','m');

UNLOCK TABLES;
```

## 6.2. TAREAS. SQL DUMP

```
--
-- Table structure for table `tareas`
--

DROP TABLE IF EXISTS `tareas`;

CREATE TABLE `tareas` (
  `id_tarea` int(11) NOT NULL AUTO_INCREMENT,
  `id_curso` int(11) NOT NULL,
  `descripcion` varchar(255) NOT NULL,
  `asignatura` varchar(50) NOT NULL,
  `fecha` date NOT NULL,
  PRIMARY KEY (`id_tarea`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `tareas`
--

LOCK TABLES `tareas` WRITE;

UNLOCK TABLES;
```

## 7. Usabilidad

La usabilidad hace referencia a la calidad de la experiencia del usuario a la hora de utilizar la página, en función de si resulta cómoda de utilizar y si su manejo es intuitivo.

He intentado hacer que el funcionamiento de la página sea sencillo y fácil de entender. Solamente consta de dos listados, y he procurado que tenga un aspecto que recuerde fácilmente a otros sistemas de CRUD. He intentado que los textos sean lo suficientemente descriptivos y he usado iconos con significados estandarizados (papelera para borrar) y con tooltips que se muestran al pasar el cursor por encima por si no quedara claro cuál es su función.

He usado colores claros pero contrastados para resaltar cada elemento, que cambian de color cuando se pasa el cursor por encima, y el botón de volver al listado de cursos es lo suficientemente visible para facilitar la navegación.

## 8. Accesibilidad

La accesibilidad de una página web implica su capacidad para permitir que personas de edad avanzada o con algún tipo de discapacidad puedan hacer uso normal de ella.

Personalmente, reconozco que es posible que ciertos elementos de mi proyecto no sean del todo accesibles debido al uso de eventos onclick o que el tamaño de ciertos botones no sea lo suficientemente grande para poder pulsarse en dispositivos móviles. Me hubiera gustado poder centrarme en desarrollar mejor estas características, pero he priorizado el desarrollo de la funcionalidad básica.

No obstante, sí he facilitado la posibilidad, a la hora de añadir o editar una tarea, que mediante la pulsación de la tecla enter, se puedan aceptar los cambios. Al igual que, pulsando la tecla escape, se puede cancelar la operación igual que si se hubiera pulsado el botón de cancelar.

También, como curiosidad, si se añade el parámetro ***&idioma=en*** en la url tanto en la lista de cursos como en la lista de tareas, todos los textos de la interfaz saldrán en inglés. No incluye los textos del menú de la intranet ni los datos almacenados en base de datos, puesto que no es una implementación centralizada en toda la aplicación, pero podría potencialmente servir si alguna vez se decide traducir completamente la intranet.

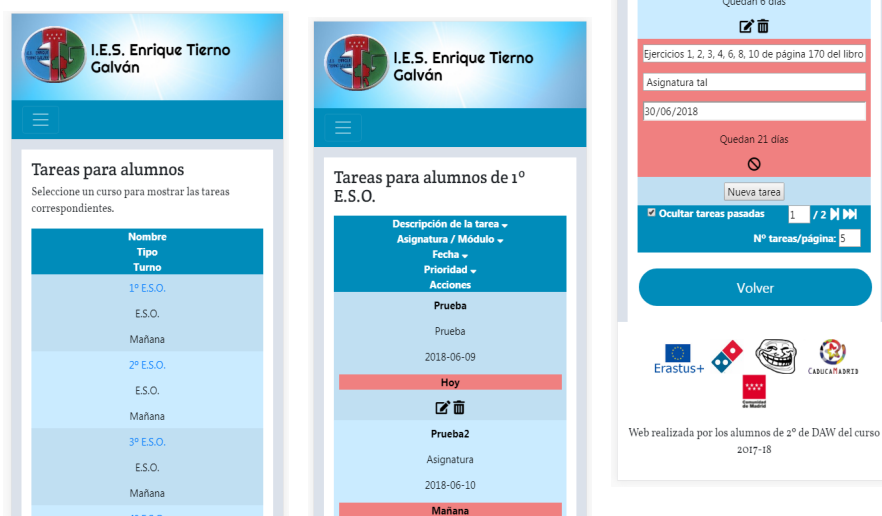


## 9. RESPONSIVE WEB DESIGN

Como podemos observar, la página se redimensiona correctamente, gracias al trabajo de diseño de Erasto y la implementación de bootstrap.



Realmente la única consideración que he tenido que tener respecto a RWD es mostrar las celdas de las tablas en vertical cuando el ancho de la página es menor de 650px con un media-query de CSS.



## 10. PÁGINAS CONSULTADAS

Las páginas web que he consultado con más frecuencia para resolver los problemas con los que me he ido encontrando son las siguientes:

- <https://stackoverflow.com>

Web imprescindible a la hora de programar. Cualquier problema de código buscado en Google habido y por haber siempre lleva a esta página, en la que una gran cantidad de usuarios aportan distintas soluciones a todos los problemas.

- <https://www.w3schools.com/>

Probablemente una de las mejores webs para consultar cualquier duda sobre HTML, CSS o JavaScript.

- <http://php.net>

La principal guía de referencia para todo lo relacionado con PHP.

- <https://twig.symfony.com/>

Muy útil para consultar dudas con las plantillas twig.

- <http://fontawesome.com/>

¿Necesitas iconos en tu página web? Aquí tienen muchos, muy simple y fácil de implementar.

- [https://www.w3schools.com/howto/howto\\_css\\_loader.asp](https://www.w3schools.com/howto/howto_css_loader.asp)

Creación de un loader CSS, lo he usado para mostrar una pantalla de carga mientras se espera el resultado de una llamada Ajax.