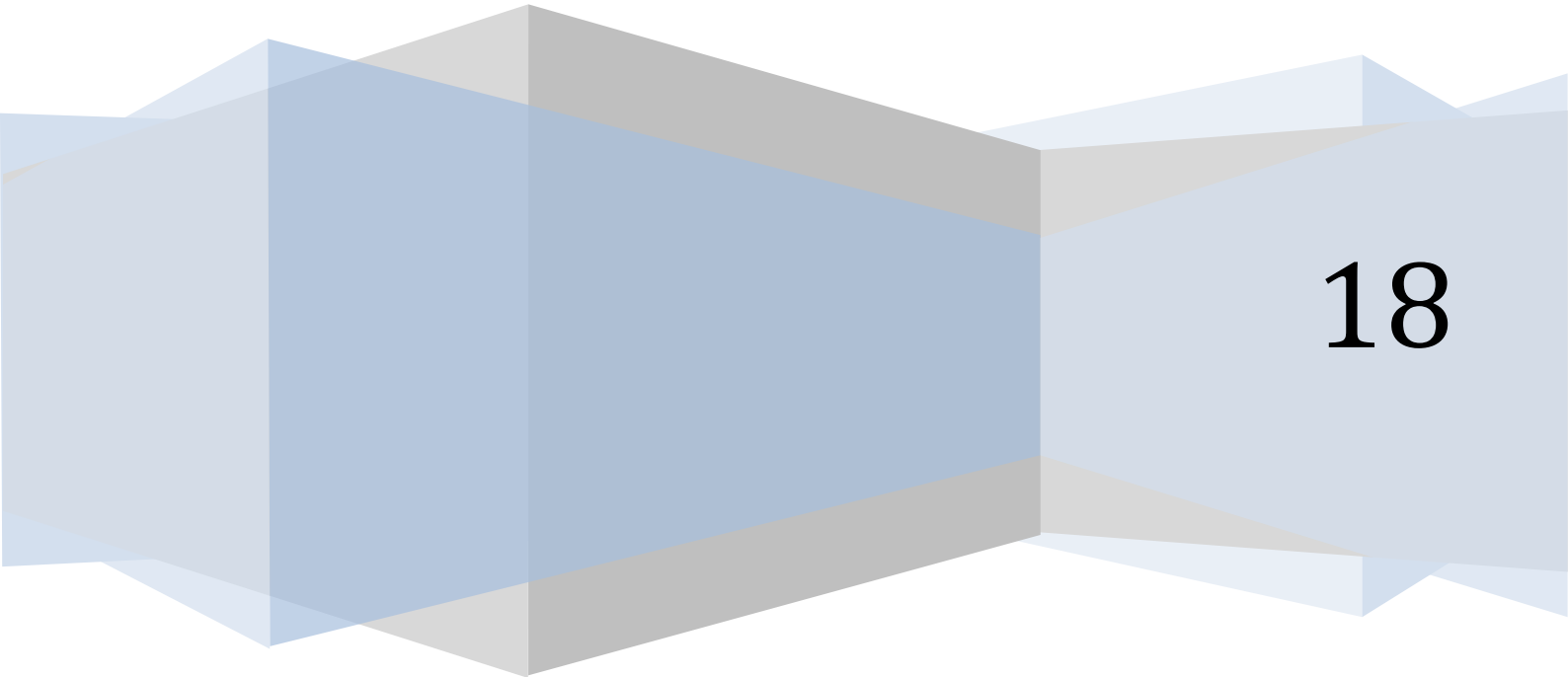


Instituto Enrique Tierno Galván.

# Proyecto final grado DAW

Seguimiento de Programaciones.

Sergio González Negrete



18



---

## **ÍNDICE.**

---

<b>1. <u>Introducción al proyecto.</u></b>	
1.1 ¿Qué es el seguimiento de programaciones?.....	4
1.2 ¿Para que sirve el seguimiento de programaciones? .....	4
<b>2. <u>Desarrollo.</u></b>	
2.1 Herramientas utilizadas. ....	4
2.2 Técnicas utilizadas. ....	5
2.3 Organización de los archivos. ....	5
2.4 Desarrollo de la interfaz. ....	6
2.5 Desarrollo del servidor. ....	9
2.6 Base de datos. ....	13
<b>3. <u>Uso.</u></b>	
3.1 Uso general de la app. ....	13
<b>4. <u>Accesibilidad.</u></b> .....	16
<b>5. <u>Problemas del desarrollo</u></b> .....	17
<b>6. <u>Páginas de ayuda.</u></b> .....	19

---

# INTRODUCCIÓN DEL PROYECTO.

---

## 1. ¿QUÉ ES EL SEGUIMIENTO DE PROGRAMACIONES?.

El seguimiento de programaciones consiste en sistema que utilizan los profesores para poder controlar de una manera informatizada y cómoda como de avanzados o retrasados van con sus programaciones respecto a su idea principal.

## 2. ¿Cómo se utilize el seguimiento de programaciones?.

El sistema de programaciones se utiliza de una manera muy sencilla.

Podremos crear una asignatura asociada a un curso ( está orientada al departamento de informática, pero se puede utilizar con resto ya que también están incluidos en la base datos), una vez creada la asignatura, podremos agregarle los temas que queramos, así como asociarlos a una evaluación.

---

# DESARROLLO

---

## 1. Herramientas utilizadas.

Las herramientas utilizadas para este proyecto han sido:

1. NetBeans→Editor de código para php, html, js entre otros, con el se ha realizado el código del proyecto.
2. MySQL WORKBENCH → Cliente para MySQL que nos permite accede a la base de datos de una manera gráfica y muy fácil de utilizar, imprescindible.
3. GitHub→ Herramienta imprescindible para la unión de las distinta partes de la aplicación.
4. Navegadores→ Esencial para poder probar la funcionalidad y ver las pantallas de la aplicación.

## 2. Técnicas usadas en el desarrollo.

La técnica más utilizada en esta parte de la aplicación ha sido *Ajax* que nos permite conectarnos con el servidor de una manera rápida y sin necesidad de tener que recargar la página, pudiendo actualizar la parte de la misma que nosotros creamos conveniente.

Por parte de la vista, las técnicas más utilizadas son las ventanas modales (se explicaran mejor en puntos posteriores) y las tabs, que son pestañas dentro de la misma página (se explicaran mejor en puntos posteriores), de esta manera nos ahorramos los tiempos de carga entre ventanas.

## 3. Organización de los archivos.

La organización de los archivos hecha por Oscar Novillo, nuestro profesor de Desarrollo en entorno Web.

Los archivos se organizan en diferentes carpetas

1. **Controllers**→ Son los receptor de la información procedente del la página web, dentro de esta carpeta hay una carpeta llamada **SeguimientoProgramaciones** donde están los controladores para la aplicación.
2. **DAO**→ Son la conexión con la base de datos, al igual que en la carpeta anterior, dentro de esta carpeta hay otra con los archivos de la app.
3. **Servicios**→ Es la conexión entre los DAO y los Controllers, misma organización que anteriormente
4. **Vistas**→ Aquí se guardan los archivos para las templates (plantillas), que servirán para la vista de nuestra app. Misma organización.
5. **CSS**→ Carpeta para guardar los estilos de la vista. Misma organización.
6. **JS**→ Carpeta para guardar los js, que sirven para darle dinamismo a las pantallas de la app. Misma organización.

## 4. Desarrollo de la interfaz.

Como se ha dicho anteriormente, la interfaz está basada en el uso de tabs y modales, aquí vamos a ver como están desarrollados.

### 1. Modales:

Los modales o ventanas modales son un conjunto de etiquetas de html con unos estilos para darlos “vida”:



La imagen sería un ejemplo de uno de los modales de la aplicación, como se ve, el modal queda por encima del resto de elementos, y pone una “barrera”, tras él para que solo él sea *clickable*, el modal se puede cerrar pulsando el aspa de arriba a la derecha o el botón oportuno.

Los modales son técnicas muy útiles para pedir información al usuario, como el nombre, email, dirección, etc, sin necesidad de cargar otra página, ya que el modal se forma al cargar la página, pero está Escondido.

### 2. Tabs:

La técnica de las tabs o pestañas, consiste en un conjunto de etiquetas con estilos muy definidos, con ellas se permite tener mucha información dentro de nuestra página pero no saturar al usuario con ella, también nos evitan tener que poner enlaces a otras páginas, lo que implicaría tiempos de carga entre una ventana y otra.

Como se ve en la imagen existen dos pestañas en nuestra aplicación, la que pestaña que está activada se muestra con un fondo blanco al igual que contenedor de información que se muestra, así damos seguridad al usuario de que está viendo la información que ha solicitado.

A parte de las pestañas y los modales, la información relacionada con los temas y las asignaturas se organizan en tablas.

Estas tablas se modifican de manera dinámica al realizar una de las operaciones, como por ejemplo:

- Seleccionando un curso y una asignatura la tabla de las unidades se rellenará con los temas de esa asignatura, además el cajón de las asignaturas también se rellena de manera dinámica cuando se cambia de curso.
- También se va a cambiar la tabla de manera dinámica cuando se cree una nueva asignatura o tema (siempre y cuando los cajones estén seleccionados), y a parte también se modifica cuando se actualiza una asignatura o un tema, así como con el borrado de los mismos.

Para realizar esta modificación de la tabla ha sido necesario realizar dos funciones, una para cada tabla, combinando jQuery, así como el DOM de JavaScript, más concretamente se han usado las funciones de *createElement*, *deleteRow*, *appendChild* y *setAttribute*.

```

function cambiar_tabla_asignaturas() {
    curso = $("#select_cursos-g-a").val();
    var tabla = document.getElementById("tabla_gestion_asignaturas");
    var longitud_tabla_inicial = tabla.rows.length;
    if (tabla.rows.length > 0) {
        for (var i = longitud_tabla_inicial; i > 1; i--) {
            tabla.deleteRow(i - 1);
        }
    }
    $.ajax({
        url: "/index.php?c=seguimiento_programaciones&destino=asignaturas&a=get_asignatura_curso",
        data: {"id_curso": curso},
        type: "POST",
        success: function (data) {
            var parseodata = JSON.parse(data);
            if (parseodata.error === undefined) {
                if (parseodata.asignaturas[0] !== undefined) {
                    for (asignatura in parseodata.asignaturas) {
                        var fila = document.createElement("tr");
                        var celda = document.createElement("td");
                        celda.setAttribute("data-id", parseodata.asignaturas[asignatura].ID);
                        celda.setAttribute("data-curso", curso);
                        celda.setAttribute("class", "name_asignatura");
                        celda.innerHTML = parseodata.asignaturas[asignatura].NOMBRE;
                        fila.appendChild(celda);
                        tabla.appendChild(fila);
                    }
                    var filaboton = document.createElement("tr");
                    var celdaBoton = document.createElement("td");
                    var boton = document.createElement("button");
                    boton.setAttribute("value", "Crear Asignatura");
                    boton.appendChild(document.createTextNode("Crear Asignatura"));

```

```

                    boton.setAttribute("id", "abrir_modal_add_asignatura");
                    boton.setAttribute("class", "btn btn-primary col-sm-12");
                    boton.setAttribute("type", "button");
                    celdaBoton.setAttribute("colspan", "3");
                    celdaBoton.appendChild(boton);
                    filaboton.appendChild(celdaBoton);
                    tabla.appendChild(filaboton);
                    $("table").delegate(".name_asignatura", "click", fn_mostrar_modal_actualizar);
                    $("table").delegate("#abrir_modal_add_asignatura", "click", fn_mostrar_modal_crear);
                } else {
                    var fila = document.createElement("tr");
                    var filaboton = document.createElement("tr");
                    var celda = document.createElement("td");
                    var celdaBoton = document.createElement("td");
                    var boton = document.createElement("button");
                    boton.setAttribute("value", "Nueva Asignatura");
                    boton.appendChild(document.createTextNode("Nueva Asignatura"));
                    boton.setAttribute("id", "abrir_modal_add_asignatura");
                    boton.setAttribute("class", "btn btn-primary col-sm-12");
                    boton.setAttribute("type", "button");
                    celda.setAttribute("colspan", "3");
                    celda.innerHTML = "No hay datos";
                    celdaBoton.setAttribute("colspan", "3");
                    celdaBoton.appendChild(boton);
                    fila.appendChild(celda);
                    filaboton.appendChild(celdaBoton);
                    tabla.appendChild(fila);
                    tabla.appendChild(filaboton);
                }
            }

```



A la hora de crear modificar de esta forma las tablas (borrando las filas que hubiera y añadiéndolas mediante el DOM) las clases y los id pierden los eventos, ya que han sido creados después de que el document esté cargado, por lo que hay que dárselos de nuevo.

¿Cómo se hace esto? Muy sencillo, jQuery tiene un evento llamado *delegate*,

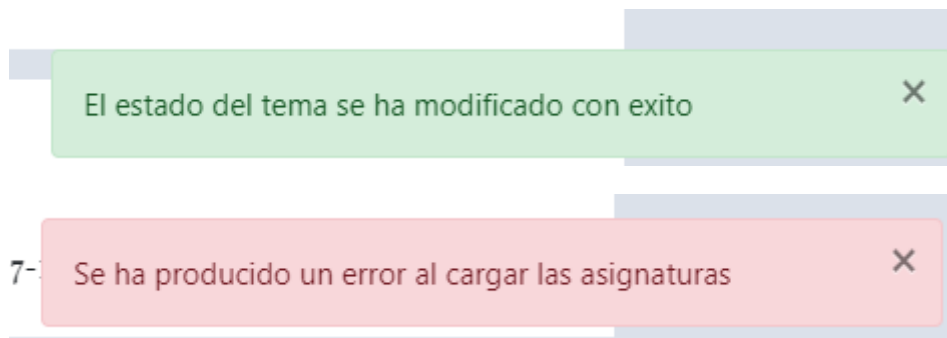
```
$("#table").delegate(".abrir-modal-update-tema", "click", fn_mostrar_modal_upt_temas);  
$("#table").delegate("#abrir_modal_add_tema", "click", fn_mostrar_modal_nuevo_tema);  
$("#table").delegate("#cambiar_estado_unidad", "click", marcar_como_hecho_tema);
```

Lo que le indicas es un selector, en este caso es la etiqueta table, y luego le indicas el id o la clase a la que quieras que se le asocie un evento, este viene a continuación seguido de la funcion que quieres que se ejecute cuando se produzca ese evento.

Cabe destacar que debido a que los estilos de la interfaz están realizados con la librería Bootstrap de css el diseño es totalmente responsive.

Para concluir esta parte es necesario destacar el uso de la libreria *Notify* de Bootstrap, esto es un penqueño mensaje que sale en la parte inferior derecha (se puede poner dónde se quiera).

Tiene varios colores, pero los utilizados aquí son el verde(operación realizada con éxito), el rojo (algún tipo de fallo) y el azul (información).



## 5. Desarrollo del servidor.

La parte del servidor se ha desarrollado con el lenguaje PHP.

Como se ha contado anteriormente la parte del servidor está dividida en tres parte, para que así que mas estructurado el código, de manera que sea más fácil realizar el mantenimiento.

### 1. Controller:

El controller está estructurado de manera que se recogen los parametros de la url y se actua sobre ellos, lo primero que se mira es la acción que se va a realizar, y mediante la estructura *switch* de php se ejecutará una cosa u otra, una vez esté la acción comprobada se mirará el destino de la acción, es decir, si la acción se realizará sobre las asignaturas, temas o cursos.

```
if (isset($_REQUEST[Constantes::PARAMETER_NAME_ACTION]) && $_REQUEST[seguiminetoConstantes::DESTINO]) {
    $accion = $_REQUEST[Constantes::PARAMETER_NAME_ACTION];
    $destino = $_REQUEST[seguiminetoConstantes::DESTINO];
    switch ($accion):
        case seguiminetoConstantes::ACCION_INSERTAR:
            switch ($destino):
                case seguiminetoConstantes::ASIGNATURAS:
                    $mensaje = new stdClass;
                    $json_asignatura = $_REQUEST[seguiminetoConstantes::OBJETO_ASIGNATURA_JSON];
                    $servicios = new seguimientoServices();
                    $objetoAsig = $servicios->parseo_json($json_asignatura);
                    if ($objetoAsig->nombre != "" && $objetoAsig->id != "") {
                        $asignatura_insertada = $servicios->insertar_asignatura($servicios->parseo_json($json_asignatura));
                        echo $asignatura_insertada;
                    } else {
                        $mensaje->error = \utils\seguimientoProgramaciones\constantasMensajes::PARAMETROS_NO_RECIBIDOS;
                        echo json_encode($mensaje);
                    }
                    break;
                case seguiminetoConstantes::UNIDADES:
                    $json_unidad = $_REQUEST[seguiminetoConstantes::OBJETO_UNIDAD_JSON];
                    $servicios = new seguimientoServices();
                    $objetoUnidad = $servicios->parseo_json($json_unidad);
                    if ($objetoUnidad->nombre != "" && $objetoUnidad->evaluacion != "" && $objetoUnidad->id_asignatura != "") {
                        $unidad_insertada = $servicios->insertar_unidad_trabajo($servicios->parseo_json($json_unidad));
                        echo $unidad_insertada;
                    } else {
                        $mensaje->error = \utils\seguimientoProgramaciones\constantasMensajes::PARAMETROS_NO_RECIBIDOS;
                        echo json_encode($mensaje);
                    }
            }
        }
    }
```

## 2. Servicios.

Con los servicios lo que se consigue es separar los dao (conexión a la base de datos) y la recepción de datos.

Este archivo consiste en llamadas a las funciones del dao, y en el controller se llaman a las funciones de los servicios pertinentes.

```
class seguimientoServices {
    public function parseo_json($json) {
        $json_parseado = json_decode($json);
        return $json_parseado;
    }
    public function insertar_asignatura($asignatura) {
        $dao = new crudAsignaturas();
        return $dao->crear_asignatura($asignatura);
    }
    public function modificar_asignatura($asignatura) {
        $dao = new crudAsignaturas();
        return $dao->modificar_asignatura($asignatura);
    }
    public function borrar_asignatura($asignatura) {
        $dao = new crudAsignaturas();
        return $dao->borrar_asignatura($asignatura);
    }
    public function borrado_total($asignatura) {
        $dao = new crudAsignaturas();
        return $dao->borrado_total($asignatura);
    }
}
```

### 3. DAO

En estos archivos se realizarán las operaciones con la base de datos, inserción, lectura, borrado, actualización.

En el caso de esta app tiene 3 archivos, uno para Asignaturas, otro para Temas y otro para Cursos

```
public function modificar_asignatura($asignatura_modificar) {
    $asignaturaObjeto = new \stdClass;
    $mensaje = new \stdClass;
    $connectionDB = new DBConnection();
    $conn = $connectionDB->getConnection();
    try {
        $conn->beginTransaction();
        $stmt = $conn->prepare(ConstantsBD::actualizar_asignatura);
        $stmt->bindParam(1, $asignatura_modificar->nombre);
        $stmt->bindParam(2, $asignatura_modificar->id_asignatura);
        $stmt->execute();
        $stmt = $conn->prepare(ConstantsBD::update_asignatura_curso);
        $stmt->bindParam(1, $asignatura_modificar->id_curso);
        $stmt->bindParam(2, $asignatura_modificar->id_asignatura);
        $stmt->execute();
        $conn->commit();
        $mensaje->exito = constantesMensajes::ACTUALIZACION_HECHA;
    } catch (\Exception $ex) {
        $conn->rollback();
        $mensaje->error = constantesMensajes::ERROR_GENERAL;
        echo json_encode($mensaje);
    } finally {
        $connectionDB->disconnect();
    }
    return json_encode($mensaje);
}
```

```

class crudUnidadesTrabajo {
    public function crear_unidad_trabajo($unidad_trabajo_crear){
        $mensaje = new \stdClass;
        $unidad_trabajo_crear->hecho = 0;
        $connectionDB = new DBConnection();
        $conn = $connectionDB->getConnection();
        try{
            if ($this->comprobar_unidad_trabajo($unidad_trabajo_crear)==0){
                $conn->beginTransaction();
                $stmt = $conn->prepare(ConstantsBD::insert_tema);
                $stmt->bindParam(1, $unidad_trabajo_crear->nombre);
                $stmt->bindParam(2, $unidad_trabajo_crear->evaluacion);
                $stmt->bindParam(3, $unidad_trabajo_crear->hecho);
                $stmt->execute();
                $last_tema = $conn->lastInsertId();
                $stmt = $conn->prepare(ConstantsBD::asociar_tema_asignatura);
                $stmt->bindParam(1, $unidad_trabajo_crear->id_asignatura);
                $stmt->bindParam(2, $last_tema);
                $stmt->execute();
                $conn->commit();
                $mensaje->exito = constantesMensajes::INSERCIÓN_HECHA_TEMA;
            }
            else{
                $mensaje->error = constantesMensajes::ERROR_UNIDAD_DUPLICADA;
            }
        }catch(\Exception $ex){
            if ($conn != null){
                $conn->rollback();
            }
            $mensaje->error = constantesMensajes::ERROR_GENERAL;
        }
    }
}

class cursoDAO {

    public function get_all_cursos() {
        try {
            $connectionDB = new DBConnection();
            $conn = $connectionDB->getConnection();
            $stmt = $conn->prepare(ConstantsBD::GET_ALL_CURSOS);
            $stmt->execute();
            $cursos = $stmt->fetchAll(PDO::FETCH_OBJ);
            return $cursos;
        } catch (Exception $ex) {

        } finally {
            $connectionDB->disconnect();
        }
    }
}

```

El resto del servidor está completado con archivos de constants, lo que supone una comodidad a la hora del mantenimiento.

## 6. Base de Datos.

La base de datos consta de diversas tablas para las diversas funciones que esta tiene.

En el caso de mi parte son necesarias X:

1. Usuarios → Login y permisos.
2. Asignaturas → Con un nombre y un id.
3. Cursos → Con un nombre, un id, un tipo y un turno.
4. Cursos\_asignaturas → Con el id\_curso, para saber el curso que esta relacionado con la asignatura, identificada por un id\_asignatura, ambos son claves foráneas.
5. Unidades\_Trabajo → Con un id, un nombre, la evaluación, y Unidad\_hecha, que es un Boolean para saber si esa asignatura se ha dado.
6. Asignaturas\_unidadesTrabajo → Con un id\_asignaturas para saber la relación entre ella y sus temas, identificados mediante un id\_unidad\_trabajo.

---

## USO

---

Tal y como se ha dicho anteriormente, la pagina esta compuesta por modales, tablas y dos tabs (pestañas).

En la primera pestaña vamos a poder gestionar las unidades de trabajo (temas), de manera que vamos a poder crear, modificar y borrar, así como modificar el estado de este (poner su estado en realizado, es decir, tema dado en clase).

Gestión Temas   Gestión Asignaturas

### Gestión Temas

Listado Cursos: 1º Carrocería

Listado Asignaturas: Nuevisima

Nombre	Evaluación	Realizado
TemaModificado	Primera Evaluación	<input checked="" type="checkbox"/>

Crear Tema

Para crear una nueva asignatura debemos, seleccionar la pestaña que pone gestión de asignaturas y clicar en el botón que se encuentra en la última fila de la table, de esta forma se nos abrirá una ventana modal en la que podremos introducir los datos para crear la asignatura.

Gestión Temas Gestión Asignaturas

### Gestión principal/Temas asignatura

Listado Cursos

Nombre

Seleccione un curso

Nueva Asignatura

Una vez tengamos creada la asignatura podremos empezar a crear temas para ellas, así que debemos volver a la pestaña anterior.

Gestión Temas Gestión Asignaturas

### Gestión Temas

Listado Cursos

1º Carrocería

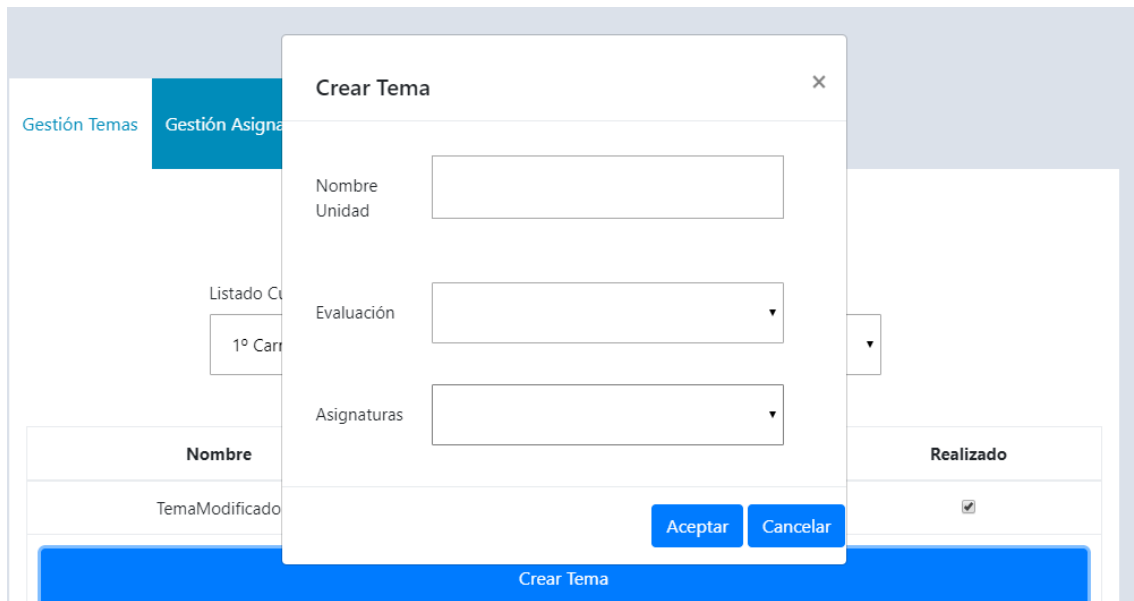
Listado Asignaturas

Nuevisima

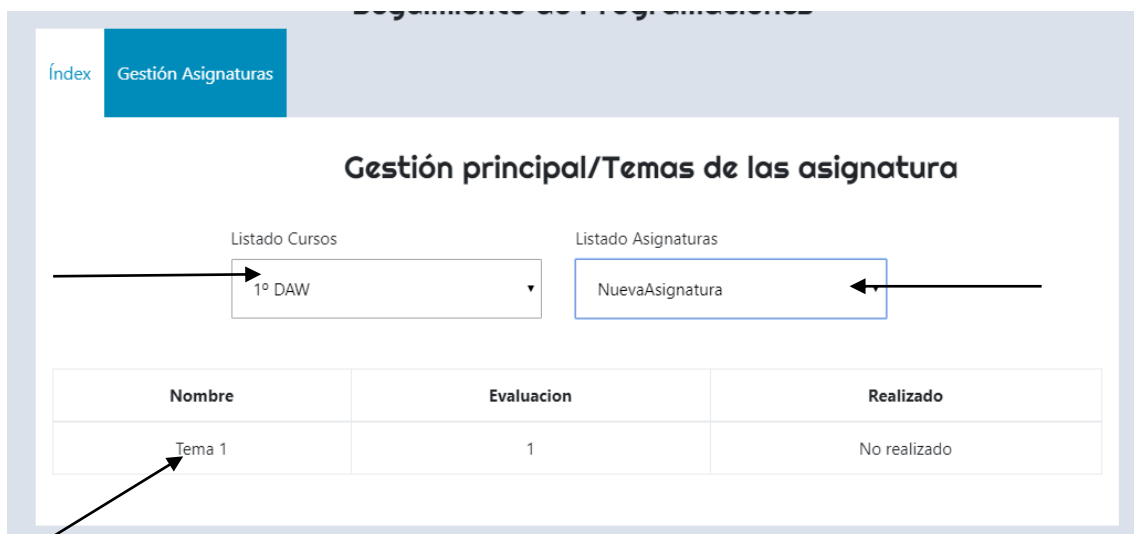
Nombre	Evaluacion	Realizado
TemaModificado	Primera Evaluación	<input checked="" type="checkbox"/>

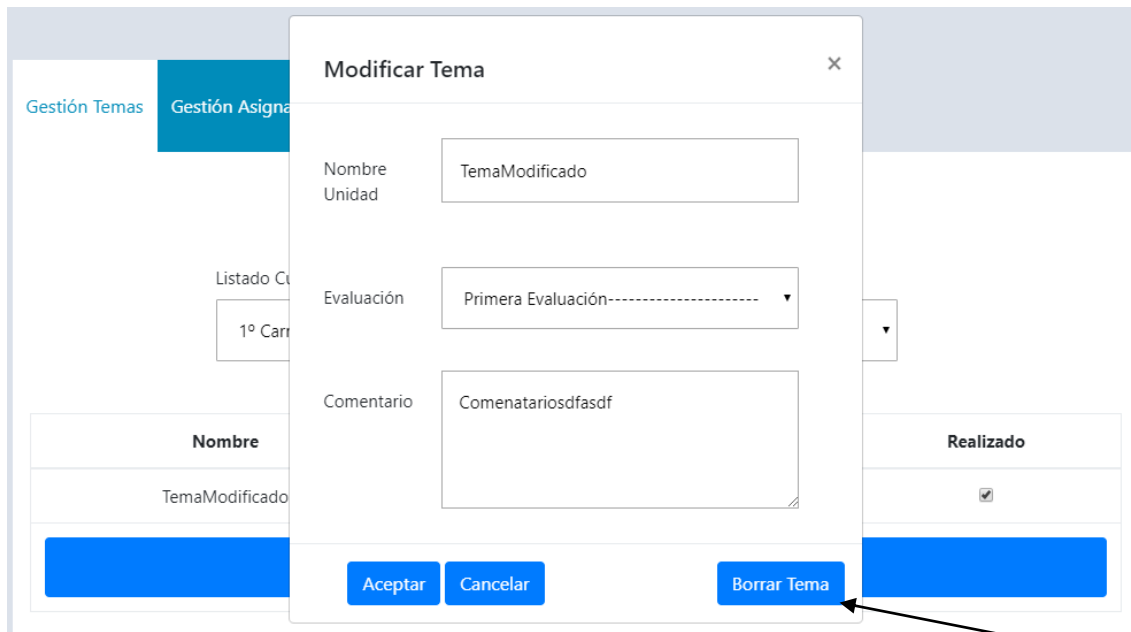
Crear Tema

Para crear un nuevo tema debemos clicar en el botón que se encuentra en la última fila de la tabla, de esta forma se nos abrirá una ventana modal, en la que podremos introducir los datos para crear el tema.



Una vez creamos los temas, podremos modificarlos, para ello deberemos elegir el curso, y la asignatura, de esta forma se nos modificará la table y clickando sobre el nombre del tema se nos desplegará otra ventana modal, como puedes ver, ahora tiene algo más, se trata de un comentario para dejar constancia de porqué se ha cambiado algo de ese tema.





En este mismo modal, abajo a la derecha podemos ver que hay un botón para borrar el tema.

En el caso de las asignaturas es exactamente igual, seleccionaremos un curso, y se nos completará la table, una vez completada pulsaremos sobre el nombre y se nos desplegará un modal, para editarla o borrarla.

---

## ACCESIBILIDAD

---

En cuanto a la accesibilidad si que es posible que la primera vez que veas la página igual puede paracer un poco complicada de utilizar, ya que tiene muchos elementos oculto que si no sabes que están ahí se puede hacer un poco complicado usarlos.

Esto ocurre sobre todo a la hora de una vez creadas las asignaturas y los temas, cuando los busques y aparezcan en la tabla seguramente no sepas que tiene que clickar en el nombre para que te aparezca el modal de editar.

Esto se soluciona cambiado el cursor de esa celda de la tabla.



---

# PROBLEMAS DURANTE EL DESARROLLO

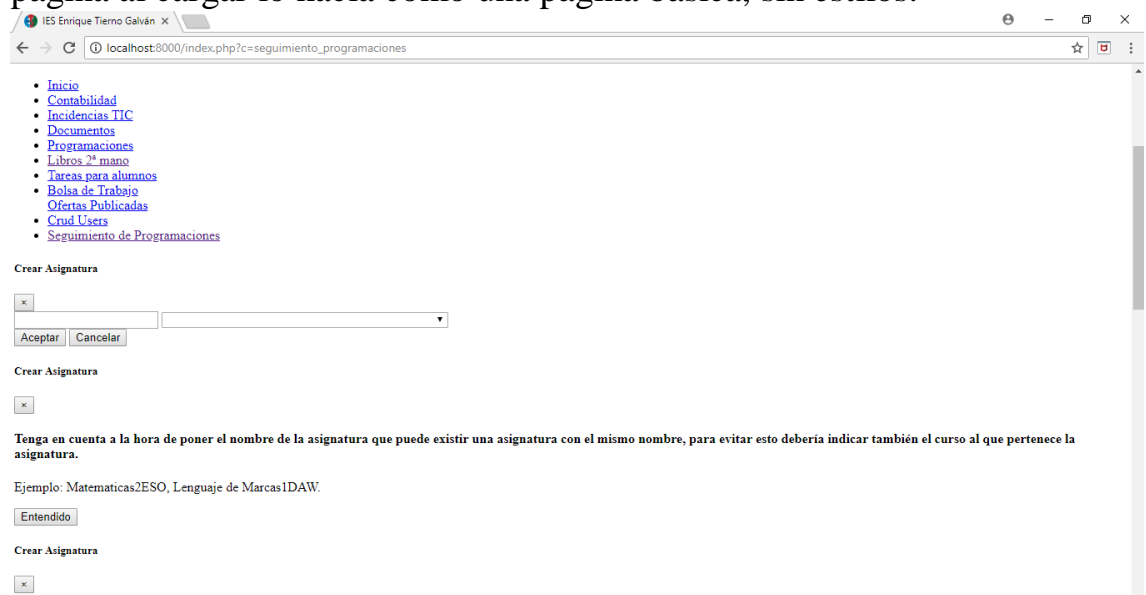
---

Durante el desarrollo de la aplicación han surgido algún que otro problema y aquí voy a proceder a enumerarlos.

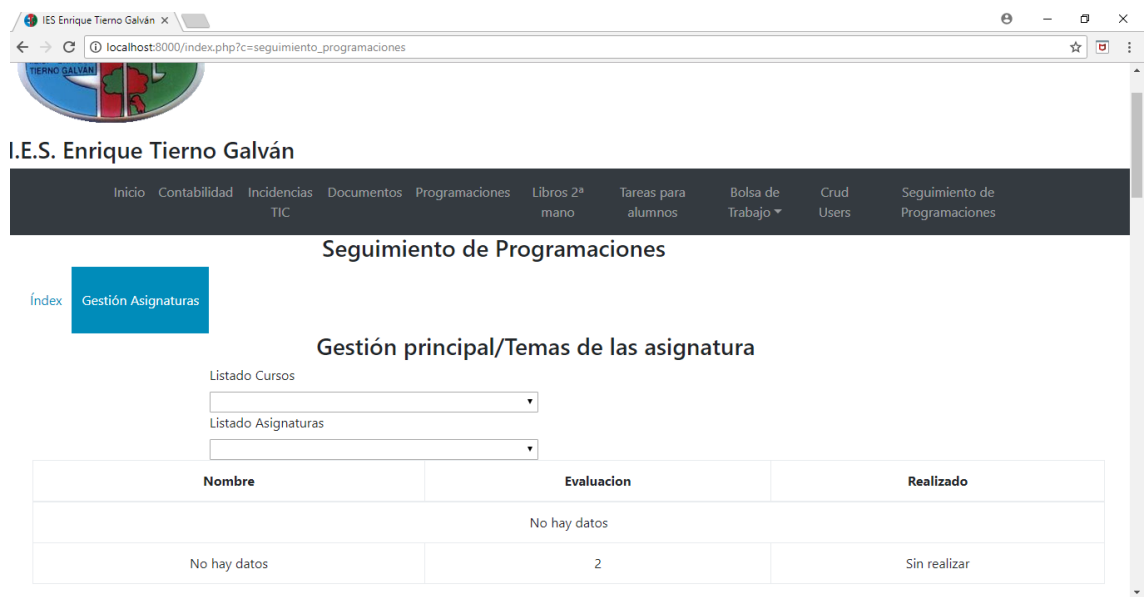
## 1. Extrema lentitud de la app:

Por una razón desconocida la aplicación carece de una velocidad aceptable para un trabajo fluido.

Esta lentitud ha llegado al punto de que en alguna de las ocasiones en las que necesitaba recargar la página para actualizar el JavaScript la página al cargar lo hacía como una página básica, sin estilos.



En otras ocasiones, si cargaba los estilos (no todos), pero no el js lo que hacía imposible, por ejemplo, abrir los modales, por lo que no se podía trabajar.



Se ha de destacar que estos problemas se han dado probando el proyecto en local, tanto servidor como base de datos. La base de datos tuve que usar la local ya que no me conectaba la aplicacion con la base de datos externa.

Para concluir este punto, el ultimo problema que me surgió fue el problema de que el Bootstrap no funcionaba en su totalidad, por ejemplo, a la hora de mostrar sus modales, Bootstrap tiene sus propias funciones pero no funcionaban, para ser más concreto lo único que funcionaba de Bootstrap de lo que yo he probado eran los *col-tamaño* y la clase *row*. Para solucionarlo tuve que crear mi propio sistema de pestañas y modales.

```
$(document).ready(function () {
    $(".tab").css("border", "0px");
    $("#mostrar_contenedor_index").css("background", "white");
    $("#mostrar_contenedor_index").css("color", "#008cba");
    $("#mostrar_contenedor_asignaturas").css("background", "#008cba");
    $("#mostrar_contenedor_asignaturas").css("color", "white");
    $(".tab").click(function () {
        var id = $(this).attr("data-target");
        $("." + id).show();
        $(this).css("background", "white");
        if ($(this).attr("data-target") == "div_contenedor_index") {
            $(".contenedor_gestion_asignaturas").hide();
            $("#mostrar_contenedor_asignaturas").css("background", "#008cba");
            $("#mostrar_contenedor_asignaturas").css("color", "white");
            $("#mostrar_contenedor_index").css("color", "#008cba");
        } else {
            $(".div_contenedor_index").hide();
            $("#mostrar_contenedor_index").css("background", "#008cba");
            $("#mostrar_contenedor_index").css("color", "white");
            $("#mostrar_contenedor_asignaturas").css("color", "#008cba");
        }
    });
});
```

---

## PÁGINAS CONSULTADAS

---

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

Viene muy bien para consultas rápidas, sobretodo por poder probar lo que vas a hacer sin perder mucho tiempo.

- <https://es.stackoverflow.com/>

Si eres programador conoces y has usado esta página. Imprescindible.

- <https://dev.mysql.com/doc/>

Para mirar como hacer algún tipo de consulta esta es tu página.

- <http://php.net/manual/es/intro-what-is.php>

Cualquier duda sobre php está aquí.

- <https://getbootstrap.com/docs/4.0/getting-started/introduction/>

Como hacer el diseño de tu página, con ejemplos codificados.