

Universidad ORT Uruguay
Facultad de Ingeniería
Escuela de Tecnología

OBLIGATORIO PROGRAMACION 1

DOCUMENTO DE ANÁLISIS



Diego Gagliano - 266155



Gonzalo Pérez - 249454

Grupo: N1D

Docente: Bruno Díaz

Analista en Tecnologías de Información

Fecha de entrega del documento (24-06-2021)

Índice

Contenido

Índice 2

1.	Descripción general del problema a resolver	5
1.1.	Tipos de usuario del sistema	5
1.2.	Listado de funcionalidades	5
2.	Detalle de Funcionalidades	6
2.1	F01 – Registro de usuarios	6
2.1.1	Acceso	6
2.1.2	Descripción	6
2.1.3	Interfaz de usuario	6
2.1.4	Validaciones	6
2.2	F02 – Inicio de sesión	7
2.2.1	Acceso	7
2.2.2	Descripción	7
2.2.3	Interfaz de usuario	7
2.2.4	Validaciones	7
2.3	F03 – Finalizar sesión	8
2.3.1	Acceso	8
2.3.2	Descripción	8
2.3.3	Interfaz de usuario	8
2.4	F04 – Asignación de niveles por alumno	8
2.4.1	Acceso	8
2.4.2	Descripción	8

2.4.3	Interfaz de usuario.....	8
2.4.4	Validaciones.....	8
2.5	F05 – Plantear ejercicios	9
2.5.1	Acceso.....	9
2.5.2	Descripción	9
2.5.3	Interfaz de usuario.....	9
2.5.4	Validaciones.....	9
2.6	F06 – Calificar y comentar tareas	9
2.6.1	Acceso.....	9
2.6.2	Descripción	10
2.6.3	Interfaz de usuario.....	10
2.6.4	Validaciones.....	10
2.7	F07 – Estadísticas de alumnos	10
2.7.1	Acceso.....	10
2.7.2	Descripción	10
2.7.3	Interfaz de usuario.....	10
2.7.4	Validaciones.....	10
2.8	F08 – Buscador de tareas	11
2.8.1	Acceso.....	11
2.8.2	Descripción	11
2.8.3	Interfaz de usuario.....	11
2.8.4	Validaciones.....	11
2.9	F09 – Realizar tarea	11
2.9.1	Acceso.....	11
2.9.2	Descripción	11
2.9.3	Interfaz de usuario.....	12

2.9.4	Validaciones.....	12
2.10	F10 – Ver ejercicios resueltos	12
2.10.1	Acceso.....	12
2.10.2	Descripción	12
2.10.3	Interfaz de usuario.....	12
2.11	F11 – Estadísticas de tareas.....	12
2.11.1	Acceso.....	12
2.11.2	Descripción	12
2.11.3	Interfaz de usuario.....	13
2.11.4	Validaciones.....	13
3.	Informe de Testing.....	13
4.	Código HTML	17
5.	Código Javascript – baseDatos.js	21
6.	Código Javascript – funcionesParametros.js	24
7.	Código Javascript – menuFunciones.js	34
8.	Código Javascript – miCodigoPrincipal.js	39
9.	Datos Precargados	52

1. Descripción general del problema a resolver

La aplicación tiene por objetivo, realizar un entorno virtual dónde el docente pueda gestionar sus alumnos, a los cuales les asignará tareas según su nivel, hacer devoluciones, y llevar un control estadístico de los trabajos realizados. El alumno podrá resolver las tareas asignadas por el profesor, así como ver información estadística de sus tareas realizadas y devoluciones del profesor.

1.1. Tipos de usuario del sistema

Utilizaremos dos tipos de usuarios:

- Alumno
- Profesor

1.2. Listado de funcionalidades

F01 – Registro de usuarios – Usuario/s: alumno, profesor.

F02 – Inicio de sesión – Usuario/s: alumno, profesor.

F03 – Finalizar Sesión – Usuario/s: alumno, profesor.

F04 – Asignación de niveles por alumno – Usuario/s: profesor.

F05 – Plantear ejercicios – Usuario/s: profesor.

F06 – Calificar y comentar tareas – Usuario/s: profesor.

F07 – Estadísticas de alumnos – Usuario/s: profesor.

F08 – Buscador de tareas – Usuario/s: alumno.

F09 – Realizar tarea – Usuario/s: alumno.

F10 – Ver ejercicios resueltos – Usuario/s: alumno.

F11 – Estadísticas de tareas – Usuario/s: alumno.

2. Detalle de Funcionalidades

A continuación, se presenta el detalle de cada una de las funcionalidades a resolver en el obligatorio. Las funcionalidades están ordenadas por tipo de usuario.

2.1 F01 – Registro de usuarios

2.1.1 Acceso

- Profesor
- Alumno

2.1.2 Descripción

Guardará la información personal de un nuevo usuario.

2.1.3 Interfaz de usuario

The image displays two user registration forms side-by-side. The top form, titled 'Registro de Profesor', includes fields for 'Nombre:' (with placeholder 'Ingresar nombre del profesor'), 'Usuario:' (with placeholder 'Ingresar usuario de profesor'), 'Ingresar contraseña:' (with placeholder 'Ingresar password'), and 'Confirmar contraseña:' (with placeholder 'Repetir password'). A 'Registrar' button is at the bottom. The bottom form, titled 'Registro de Alumno', includes fields for 'Nombre:' (with placeholder 'Ingresar Nombre del Alumno'), 'Usuario:' (with placeholder 'Ingresar Nombre de Usuario'), 'Ingresar contraseña:' (with placeholder 'Ingresar Password'), and 'Confirmar contraseña:' (with placeholder 'Repetir password'). It also features a 'Seleccione profesor:' dropdown menu with 'Prof' selected and a 'Registrar' button at the bottom.

2.1.4 Validaciones

- Todos los datos son obligatorios.
- Nombre: Debe indicar el nombre de la persona.
- Usuario: Nombre de usuario con el cual ingresará a la aplicación. Debe ser único.
- Ingresar contraseña: Contraseña que validará el acceso al usuario, la misma debe contener una letra mayúscula, una minúscula y un largo no menor a 4 caracteres.
- Confirmar contraseña: Debe ser idéntica a la contraseña, el objetivo es corroborarla.

- Seleccione profesor: Al momento de registrarse el alumno, deberá indicar a que profesor quiere asignarse.
- En caso de no completar algún campo, el usuario no quedará registrado, y dará un mensaje: “Debe verificar los datos ingresados”.
- Si el nombre de usuario ya existe, no permitirá el registro, y la aplicación informará: “Nombre de usuario ya existente, por favor ingrese otro nombre.”

2.2 F02 – Inicio de sesión

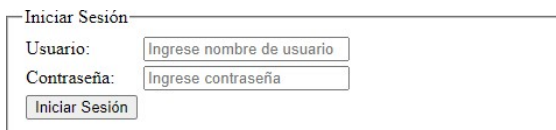
2.2.1 Acceso

- Profesor
- Alumno

2.2.2 Descripción

Permite el acceso del usuario ya registrado a la aplicación.

2.2.3 Interfaz de usuario



The screenshot shows a login form titled "Iniciar Sesión". It contains two input fields: "Usuario:" with the placeholder text "Ingrese nombre de usuario" and "Contraseña:" with the placeholder text "Ingrese contraseña". Below these fields is a button labeled "Iniciar Sesión".

2.2.4 Validaciones

- Todos los campos son obligatorios.
- Usuario: Indicará el nombre de usuario al momento del registro.
- Contraseña: Indicará la contraseña asignada al usuario al momento del registro.
- En caso de que los datos no sean coincidentes, indicará el mensaje: “Datos incorrectos, verifique.”

2.3 F03 – Finalizar sesión

2.3.1 Acceso

- Profesor
- Alumno

2.3.2 Descripción

El usuario cerrará la sesión, mostrando la pantalla inicial.

2.3.3 Interfaz de usuario

Cerrar Sesión

2.4 F04 – Asignación de niveles por alumno

2.4.1 Acceso

- Profesor

2.4.2 Descripción

El usuario tendrá la posibilidad de gestionar a sus alumnos asignándolos a un nivel: Inicial, Intermedio y Avanzado.

2.4.3 Interfaz de usuario

Nombre del alumno	Nivel
Alumno 1	Inicial
Alumno 2	Intermedio
Alumno 3	Avanzado

2.4.4 Validaciones

La asignación de niveles únicamente permitirá el ascenso del alumno, una vez ascendido la aplicación no permitirá el retroceso.

2.5 F05 – Plantear ejercicios

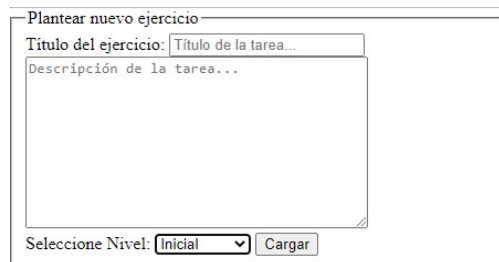
2.5.1 Acceso

- Profesor

2.5.2 Descripción

El usuario podrá plantear diferentes ejercicios a los alumnos que le correspondan, tendrá la posibilidad de asignar la tarea seccionándola por nivel.

2.5.3 Interfaz de usuario



The screenshot shows a web form titled "Plantear nuevo ejercicio". It contains two text input fields: "Título del ejercicio: Título de la tarea..." and "Descripción de la tarea...". Below these fields is a dropdown menu labeled "Seleccione Nivel:" with "Inicial" selected. To the right of the dropdown is a "Cargar" button.

2.5.4 Validaciones

- Título del ejercicio
- Descripción de la tarea
- Selección de nivel

Entre el título y la descripción no pueden tener menos de 20 caracteres, ni superar el máximo de 200.

La tarea será asignada al grupo de alumnos correspondiente al nivel indicado.

2.6 F06 – Calificar y comentar tareas

2.6.1 Acceso

- Profesor

2.6.2 Descripción

Se listarán las tareas que los alumnos hayan terminado y que estén pendientes de evaluar y comentar.

2.6.3 Interfaz de usuario

Calificar Tareas			
Alumno	Título de tarea	Audio	Comentarios
Alumno1	Nombre de la tarea cargada	"audio_alumno1.mp3"	<div>Comentar...</div> <div>Comentar</div>

2.6.4 Validaciones

Deberá ingresar los comentarios pertinentes a la tarea realizada por el alumno. No podrá cerrar una devolución sin dejar un comentario. En ese caso, la aplicación enviará un mensaje: “Debe ingresar un comentario.”

2.7 F07 – Estadísticas de alumnos

2.7.1 Acceso

- Profesor

2.7.2 Descripción

Mostrará la información estadística de las tareas realizadas por los alumnos.

2.7.3 Interfaz de usuario

Estadísticas de los alumnos	
Alumno con más ejercicios resueltos:	
Total ejercicios resueltos:	
Alumno 1 ▾	

2.7.4 Validaciones

- Indicará cual es el alumno con más ejercicios resueltos, el total de ejercicios resueltos por todos sus alumnos.

- Podrá seleccionar un alumno que esté a su cargo, e indicará la cantidad de ejercicios indicados, y la cantidad de resueltos por el mismo.

2.8 F08 – Buscador de tareas

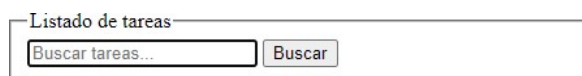
2.8.1 Acceso

- Alumno

2.8.2 Descripción

El usuario tendrá un listado de las tareas que tiene asignadas. Tendrá la opción de buscar la tarea por nombre o descripción.

2.8.3 Interfaz de usuario



The image shows a web interface element titled "Listado de tareas". Below the title is a search bar with the placeholder text "Buscar tareas..." and a button labeled "Buscar".

2.8.4 Validaciones

- Inicialmente el usuario tendrá listadas todas las tareas asignadas para él.
- Podrá buscar por el nombre de la tarea o la descripción de esta.
- En caso de no coincidir, la aplicación informara: “No se encontraron tareas con ese nombre.”

2.9 F09 – Realizar tarea

2.9.1 Acceso

- Alumno

2.9.2 Descripción

El usuario podrá realizar la entrega de la tarea. Seleccionando un audio “.mp3”

2.9.3 Interfaz de usuario

Titulo tarea	Descripción	Archivo
Titulo 1	Descripción 1	Cargar Audio
Titulo 2	Descripción 2	Cargar Audio
Titulo 3	Descripción 3	Cargar Audio

2.9.4 Validaciones

- Deberá cargar un audio en formato “.mp3”, de otra forma no podrá finalizar la tarea.
- En caso de no adjuntar el archivo, indicará el mensaje: “Debe adjuntar audio .mp3.”

2.10 F10 – Ver ejercicios resueltos

2.10.1 Acceso

- Alumno

2.10.2 Descripción

El usuario podrá ver los ejercicios que ha resuelto, y si los mismos han sido evaluados y comentados por su profesor.

2.10.3 Interfaz de usuario

Titulo tarea	Comentario del Profesor
Ejercicio 1	Comentario Ej 1
Ejercicio 2	Comentario Ej 2
Ejercicio 3	Comentario Ej 3

2.11 F11 – Estadísticas de tareas

2.11.1 Acceso

- Alumno

2.11.2 Descripción

Se mostrarán las estadísticas de los ejercicios resueltos

2.11.3 Interfaz de usuario

Porcentaje de ejercicios resueltos
Ejercicios con devolución
Ejercicios sin devolución

2.11.4 Validaciones

- Indicará el porcentaje de ejercicios resueltos sobre los planteados por su profesor.
- Cantidad de ejercicios que tienen comentarios del profesor o no.

3. Informe de Testing

SUITE	DESCRIPCION	RESULTADO ESPERADO	RESULTADO FINAL
Registro de Usuario.	Se ingresa nombre,nombre usuario y tipo de usuario válidos, contraseña en minúscula y números(con 4 o mas caracteres).	No permitir registro por no contener al menos un carácter en mayúscula.	No permite el registro de usuario.
	Se ingresa nombre,nombre usuario, tipo de usuario válido, contraseña con al menos una mayúscula, pero sin números(con 4 o mas caracteres).	No permitir registro por no contener un caracter tipo numero.	No permite el registro de Usuario.
	Se ingresa nombre,nombre usuario y tipo de usuario valido, contraseña de 3 caracteres.	No permitir registro por contener menos de 4 caracteres.	No permite el registro de Usuario.
	Se ingresa Nombre de usuario, tipo de usuario y contraseña valida, campo de nombre vacio.	No permitir registro por dejar el campo de nombre vacio.	No permite el registro de Usuario.
	Se ingresa Nombre, tipo de usuario y contraseña valida, campo de Nombre de Usuario vacio	No permitir registro por dejar el campo de Nombre de Usuario vacio.	No permite el registro de Usuario.
	Se ingresa Nombre, contraseña valida y Nombre de Usuario validos, no se selecciona tipo de usuario	No permitir el registro porque debe seleccionar un tipo de Usuario.	No permite el registro de Usuario.
	Se ingresa Nombre, Nombre de Usuario, contraseña valida y tipo de usuario alumno, pero no se selecciona el profesor responsable	No permitir el registro porque debe seleccionar un profesor.	No permite el registro de Usuario Alumno.

	Se ingresa Nombre de usuario ya existente	No permitir el registro ya que el usuario ya existe.	No permite el registro de Usuario.
	Se ingresa Nombre, nombre de usuario, tipo de usuario valido y contraseña valida.	Registro con éxito.	Permite el registro del usuario.
Login.	Se ingresa usuario previamente registrado, y contraseña no asociada al usuario.	No permitir el ingreso a la plataforma por no coincidir usuario y contraseña.	No permite ingresar a la plataforma.
	Se ingresa usuario previamente registrado, contraseña que cumple con los requisitos de registro, pero no coincidente con el usuario	No permitir el ingreso a la plataforma por no coincidir usuario y contraseña.	No permite ingresar a la plataforma.
	Se ingresa contraseña que cumpla con los requisitos de registro, y se deja vacío el campo nombre de usuario	No permitir el ingreso a la plataforma por no coincidir usuario y contraseña.	No permite ingresar a la plataforma.
	Se ingresa nombre de usuario todo en minúsculas, y contraseña asociada al usuario.	permitir el ingreso ya que el usuario no distingue mayuscula y minuscula.	Permite ingresar a la plataforma.
	Se ingresa usuario todo en Mayuscula y contraseña valida asociada al usuario	permitir el ingreso ya que el usuario no distingue mayuscula y minuscula.	Permite ingresar a la plataforma.
LOGIN PROFESOR.			
Cambiar nivel de los alumnos	Se intenta seleccionar el mismo nivel al ya asignado del alumno.	mensaje de alerta que ya tiene asignado ese nivel.	"nombre del alumno" ya está asignado a ese nivel.
	Se le aumenta el nivel del alumno de inicial a intermedio e inicial a avanzado	cambio de nivel en la columna "nivel actual".	Al usuario "nombre del alumno" se le asigna correctamente el nivel aumentado.
	Se intenta seleccionar un nivel inferior al que ya tiene asignado el alumno	Mensaje de alerta que no es posible cambiar el Nivel al seleccionado.	No es posible el cambio de nivel.
Nuevo Ejercicio.	se ingresa entre 20 y 200 caracteres entre el Titulo y la descripcion de la tarea, se carga la imagen, se selecciona el nivel de alumnos y se tpea en el boton "Cargar Tarea" .	Cargar la tarea.	Se asignó la tarea correctamente al nivel seleccionado.

	Se selecciona el nivel, se carga la imagen, se coloca entre titulo y descripcion mas de 200 caracteres.	No permitir cargar la tarea.	Mensaje de error solicitando cumplir con los limites de caracteres.
	Se cumple la cantidad de caracteres y se deja vacio el título o descripción	No permitir cargar la tarea.	Mensaje de error solicitando que ninguno de los campos queden vacios.
	Se ingresa título y descripción entre 20 y 200 caracteres, se selecciona un archivo de imagen, pero no se selecciona el nivel.	No permitir cargar la tarea.	Debe seleccionar un nivel para asignar la tarea.
Calificar.	Al tapear el boton "Ver" en el suite Calificar.	Ver la tarea asignada por el alumno.	Se ve la tarea que el alumno realizó.
	Posicionado dentro de la tarea que el alumno realizó.	Ver: imagen, audio y text area para comentar la tarea.	Se ve la imagen, el audio y el text area para poder comentar la tarea.
	Se agrega comentario a la tarea y se tapea el botón comentar.	Guardar la tarea comentada	Se agrega el comentario a la tarea.
	Posicionado dentro de la tarea que el alumno realizó, se deja el text Area vacio y se tapea el boton "Comentar".	No debe hacer ninguna devolución.	No realiza ninguna devolucion.
Estadísticas	El profesor se encuentra en el suite Estadísticas.	El profesor deberá visualizar una tabla con el alumno que más ejercicios haya resuelto, el total de tareas entregadas, y un desplegable de alumnos que le corresponden donde visualizará cuantas tareas ha realizado del total que se le plantearon.	El profesor visualiza una tabla con el/los alumnos que más ejercicios hayan resuelto (en caso de no haber ninguno se muestra un mensaje de que no se han resuelto tareas), el total de las tareas entregadas y un desplegable donde puede elegir a los alumnos que le corresponden y ver cuantas tareas entregó de las propuestas en su nivel actual.
LOGIN ALUMNO			
Tareas.	Tapear en el boton "Realizar".	Debe mostrar el div de las tareas, con la imagen y permitir cargar un audio.	se muestra el div , con la imagen y permitir cargar un audio.
	El alumno se encuentra dentro de un ejercicio para realizar , carga un audio y tapea el boton "Entregar".	La tarea se deberia entregar exitosamente.	Muestra un alert diciendo que se entrego la tarea exitosamente.

	Se buscar una cadena de texto que se encuentre en cualquiera de los títulos y descripciones disponibles para el alumno	Debe formar una tabla con todas las tareas que tengan esa cadena de texto solamente en el título. En caso de que no se encuentren resultados en el título, deberá formar una tabla con las tareas que contengan esa cadena de texto en la descripción.	Busca la cadena de texto en el título, si encuentra resultados forma la tabla de tareas, si no encuentra lo busca en la descripción y forma una tabla con los resultados según dónde haya buscado.
	se busca una cadena de texto escrita totalmente en mayúsculas o totalmente en minúsculas.	Debe encontrar y mostrar una tabla con las tareas según el caso anterior.	Encuentra y muestra la tabla según el caso anterior, ya que no distingue entre mayúsculas y minúsculas.
Ejercicios Resueltos.	El alumno se encuentra en el suite Ejercicios Resueltos.	Ver una tabla con título del ejercicio, descripción, Audio cargado de las tareas que haya realizado, y comentarios del profesor en caso de tener devolución.	Se visualiza Título, Descripción, Audio cargado de las tareas que realizó y comentarios en caso de tenerlos.
Estadísticas	El alumno se encuentra en el suite Estadísticas.	El alumno debería visualizar el porcentaje de ejercicios resueltos para su nivel, la cantidad de comentarios que ha recibido por parte de un profesor y los ejercicios que restan comentar para su nivel.	El alumno visualiza el porcentaje de ejercicios resueltos para su nivel, la cantidad de comentarios que ha recibido por parte de un profesor y los ejercicios que restan comentar para su nivel.

4. Código HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>School of Rock</title>
  <link rel="stylesheet" href="css/estiloPagina.css">
</head>
<body style="background-image: url(img/webImg/wallpaper.png);">
  <div id="contenedor">
    <div id="cabezal" class="clearfix">
      
      <ul id="navPrincipal">
        <li id="btnParaIniciarSesion">
          <a>Iniciar Sesión</a>
        </li>
        <li id="btnParaRegistrarse">
          <a>Registrarse</a>
        </li>
        <li id="btnParaCerrarSesion">
          <a>Salir</a>
        </li>
      </ul>
      <div id="aux">
        <a id="btnListaDeAlumnos">Lista de Alumnos | </a>
        <a id="btnCrearNuevoEjercicio">Nuevo Ejercicio | </a>
        <a id="btnCalificar">Calificar | </a>
        <a id="btnVerTareas">Tareas | </a>
        <a id="btnVerEjerciciosResueltos">Ver Ejercicios Resueltos
|
        <a id="btnEstadisticas">Estadísticas</a>
      </div><!-- FIN AUX -->
    </div><!-- FIN CABEZAL -->
    <div id="contenido">
      <!-- INICIO: DIVS DE REGISTRO E INICIO DE SESION -->
      <div id="divRegistrarse">
        <h3>Registrarse</h3>
        <table>
          <tr>
            <td>Nombre:</td>
            <td><input type="text" id="txtNombreRegistrarse" pl
aceholder="Ingresa su nombre"></td>
          </tr>
          <tr>
            <td>Nombre de Usuario:</td>
```

```

        <td><input type="text" id="txtNombreUsuarioRegistra
rse" placeholder="Usuario"></td>
    </tr>
    <tr>
        <td>Contraseña:</td>
        <td><input type="password" id="txtPasswordRegistrar
se" placeholder="Ingrese contraseña"></td>
    </tr>
    <tr>
        <td>Tipo de Usuario:</td>
        <td>
            <select id="slctTipoDeUsuario">
                <option value="" selected>Seleccione...</op
tion>
                <option value="soyProf">Profesor</option>
                <option value="soyAlumno">Alumno</option>
            </select>
        </td>
    </tr>
</table>
<div id="divSelectorTipoDeUsuario"></div>
<input type="button" id="btnGuardarUsuarioRegistrado" value
="Registrar">
    <div id="mensajeDivError"></div>
</div>

<div id="divIniciarSesion">
    <h3>Iniciar Sesión</h3>
    <table>
        <tr>
            <td>Usuario:</td>
            <td><input type="text" id="txtIniciarSesionUsuario"
placeholder="Nombre de usuario..."></td>
        </tr>
        <tr>
            <td>Contraseña:</td>
            <td><input type="password" id="txtIniciarSesionPass
word" placeholder="Contraseña..."></td>
        </tr>
        <tr>
            <td><input type="button" id="btnIniciarSesion" valu
e="Iniciar Sesión"></td>
        </tr>
    </table>
    <div id="divMsjErrorAlIniciarSesion"></div>
</div>
<!-- FIN: DIVS DE REGISTRO E INICIO DE SESION -->
<div id="menuPrincipalProfesor">
    <div id="divListaDeAlumnosPorProfesor">

```

```

        <div id="tablaAsignacion"></div>
        <div id="msjCambioNivel"></div>
    </div>

    <!-- PLANTEO DE EJERCICIOS DEL PROFESOR -->
    <div id="divCrearNuevoEjercicio">
        <label for="tituloDelNuevoEjercicio">Ingresar Titulo: <
/label>

        <br>
        <input type="text" id="tituloDelNuevoEjercicio" placeho
lder="Ingresar Titulo">
        <br>
        <textarea id="descripcionNuevoEjercicio" placeholder="I
ngresar descripcion....."></textarea>
        <br>
        <label for="slctNivelDelEjercicio">Seleccionar Nivel: <
/label>

        <br>
        <select id="slctNivelDelEjercicio">
            <option value="" selected>Selecione...</option>
            <option value="Inicial">Inicial</option>
            <option value="Intermedio">Intermedio</option>
            <option value="Avanzado">Avanzado</option>
        </select>
        <br>
        <label for="imgNuevaTarea">Cargar Imagen</label>
        <br>
        <input type="file" id="imgNuevaTarea">
        <br><br>
        <input type="button" id="btnNuevaTarea" value="Cargar T
area">

        <div id="mensajeCrearTarea"></div>
    </div>
    <div id="divComentar">
        <div id="tablaEjerciciosParaCalificar"></div>
    </div>
    <div id="mostrarPantallaEjercicioParaCalificar">
        <div id="divVerEjercicioParaCalificar"></div>
        <br>
        <textarea id="txtComentarioProfesor"></textarea>
        <br>
        <input type="button" id="btnComentarTarea" value="Comen
tar">

        <div id="msjErrorCalificar"></div>
    </div>
    <div id="divEstadisticasProfesor">
        <div id="alumnosMasTareasEntregadas"></div>
        <div id="totalTareasEntregadas"></div>
        <select id="slctInfoAlumnos"></select>

```

```

        <div id="divMsjInfoAlumno"></div>
    </div>
</div>

<div id="menuPrincipalAlumno">
    <div id="divTareasAsignadasAlAlumno">
        <div id="divBuscadorDeTareas">
            <label for="txtBuscarTareaCargadaPorProfesor">Busca
r: </label>
            <input type="text" id="txtBuscarTareaCargadaPorProf
esor" placeholder="Buscar tarea...">
            <input type="button" id="btnBuscarTareaParaRealizar
" value="Buscar">
        </div>
        <br>
        <div id="tablaTareasAsignadasPorAlumno"></div>
    </div>
    <div id="divRealizarNuevoEjercicio">
        <div id="funcionGenerarNuevoEjercicio"></div>
        <input type="file" id="btnAudiosDeAlumno">
        <input type="button" id="btnEntregarTareaRealizada" val
ue="Entregar">
        <p id="parrafoMsj"></p>
    </div>
    <div id="divVerEjerciciosResueltos"></div>
    <div id="divEstadisticasAlumno"></div>
</div>

</div><!-- FIN CONTENIDO -->
    <div id="pie">
        <a href="https://www.ort.edu.uy">Diego Gagliano -
Gonzalo Pérez</a>
    </div><!-- FIN PIE -->
</div><!-- FIN CONTENEDOR -->

<script type="text/javascript" src="js/baseDatos.js"></script>
<script type="text/javascript" src="js/funcionesParametros.js"></sc
ript>
<script type="text/javascript" src="js/menuFunciones.js"></script>
<script type="text/javascript" src="js/miCodigoPrincipal.js"></scri
pt>
</body>
</html>

```

5. Código Javascript – baseDatos.js

```
let usuarioSesionIniciada = null;
let usuariosGuardados = [];

let pIdClaseUsuarios = 0;
class ClaseUsuarios {
    //CLASE que utilizamos para formar el objeto de usuario.

    constructor(pNombre, pUsuario, pPass, pTipoUsuario, pProfesorResponsable, pNivelAlumno) {

        this.id = pIdClaseUsuarios;
        this.nombrePersona = pNombre;
        this.nombreUsuario = pUsuario;
        this.passwordUsuario = pPass;
        this.tipoUsuario = pTipoUsuario;
        this.profesorResponsable = pProfesorResponsable;
        this.nivelAlumno = pNivelAlumno;

        pIdClaseUsuarios++;
    }
}

let crearTarea = [];
let entregarTarea = null;

let pIdClaseTarea = 0;
class ClaseTareas {
    //CLASE que utilizaremos para formar distintas etapas de la tarea.

    constructor(pTituloTarea, pDescripcionTarea, pImgTarea, pProfesorResponsable, pNivelAlumno, pUsuarioDeEntrega, pAudioTarea, pEntrega, pComentarios) {

        this.id = pIdClaseTarea;
        this.titulo = pTituloTarea;
        this.descripcion = pDescripcionTarea;
        this.img = pImgTarea;
        this.profesorResponsable = pProfesorResponsable;
        this.nivelAlumno = pNivelAlumno;
        this.usuarioQueEntrega = pUsuarioDeEntrega;
        this.audio = pAudioTarea;
        this.comentado = pEntrega;
        this.comentario = pComentarios;
        pIdClaseTarea++;
    }
}
```

```

}

let tareaEntregada = [];
let comentarTarea = null;

class ClaseTareasEntregadas {
    constructor(pId, pTituloTarea, pDescripcionTarea, pImgTarea, pProfesorResponsable, pNivelAlumno, pUsuarioDeEntrega, pAudioTarea, pEntrega, pComentarios) {
        this.id = pId;
        this.titulo = pTituloTarea;
        this.descripcion = pDescripcionTarea;
        this.img = pImgTarea;
        this.profesorResponsable = pProfesorResponsable;
        this.nivelAlumno = pNivelAlumno;
        this.usuarioQueEntrega = pUsuarioDeEntrega;
        this.audio = pAudioTarea;
        this.comentado = pEntrega;
        this.comentario = pComentarios;
    }
}

function precargaDatos() {
    //Precarga de usuarios para iniciar sistema.
    validarGuardandoUsuarioProfesor("Bruno Díaz", "BruceWayne", "Bdiaz1", "soyProf");
    validarGuardandoUsuarioProfesor("Alejandro Martinis", "aMartinis", "AleMartinis1", "soyProf");
    validarGuardandoUsuarioAlumno("Gonzalo Pérez", "GonzaP", "Gonza28294", "soyAlumno", "BruceWayne", "Inicial");
    validarGuardandoUsuarioAlumno("Diego Gagliano", "DiegoG", "Diego123", "soyAlumno", "BruceWayne", "Inicial");
    validarGuardandoUsuarioAlumno("Nataly Delfino", "NatyD", "Naty1", "soyAlumno", "aMartinis", "Inicial");
    validarGuardandoUsuarioAlumno("Gabriela Brum", "GabiB", "Gabi1", "soyAlumno", "BruceWayne", "Intermedio");
    validarGuardandoUsuarioAlumno("Pablo Fernández", "PabloF", "Pablo1", "soyAlumno", "aMartinis", "Avanzado");
    validarGuardandoUsuarioAlumno("Bruno da Silva", "BrunoLic", "Bruno1", "soyAlumno", "aMartinis", "Avanzado");
    validarGuardandoUsuarioAlumno("Manuel Tolosa", "ManuT", "Manu1", "soyAlumno", "aMartinis", "Intermedio");
    validarGuardandoUsuarioAlumno("Daniel Segovia", "ManuS", "Manu1", "soyAlumno", "BruceWayne", "Intermedio");
}

```

```

validarGuardandoTareaPendiente("Ejercicio 1", "Tocar con cualquier instrume
nto", "Inicial", "ej1.png", "brucewayne");
validarGuardandoTareaPendiente("Ejercicio 2", "Tocar con cualquier instrume
nto", "Intermedio", "ej2.png", "brucewayne");
validarGuardandoTareaPendiente("Ejercicio 3", "Tocar con cualquier instrume
nto", "Intermedio", "ej3.png", "brucewayne");
validarGuardandoTareaPendiente("Ejercicio 4", "Tocar con cualquier instrume
nto", "Avanzado", "ej4.png", "brucewayne");
validarGuardandoTareaPendiente("Ejercicio 5", "Tocar canción con saxo", "In
icial", "ej5.png", "amartinis");
validarGuardandoTareaPendiente("Ejercicio 6", "Tocar canción con bateria",
"Intermedio", "ej6.png", "amartinis");
validarGuardandoTareaPendiente("Ejercicio 7", "Tocar canción con bajo", "Av
anzado", "ej7.png", "amartinis");
validarGuardandoTareaPendiente("Ejercicio 8", "Tocar canción con guitarra",
"Avanzado", "ej8.png", "amartinis");

let nuevaTareaEntregada1 = new ClaseTareasEntregadas(0, "Ejercicio 1", "Toc
ar con cualquier instrumento", "ej1.png", "brucewayne", "Inicial", "gonzap"
, "ej1.m4a", false, "");
let nuevaTareaEntregada2 = new ClaseTareasEntregadas(0, "Ejercicio 1", "Toc
ar con cualquier instrumento", "ej1.png", "brucewayne", "Inicial", "diegog"
, "ej1.m4a", false, "");
let nuevaTareaEntregada3 = new ClaseTareasEntregadas(5, "Ejercicio 6", "Toc
ar canción con bateria", "ej6.png", "amartinis", "Intermedio", "manut", "ej
6.m4a", false, "");
let nuevaTareaEntregada4 = new ClaseTareasEntregadas(7, "Ejercicio 8", "Toc
ar canción con guitarra", "ej8.png", "amartinis", "Avanzado", "pablof", "ej
2.m4a", false, "");
let nuevaTareaEntregada5 = new ClaseTareasEntregadas(4, "Ejercicio 5", "Toc
ar canción con saxo", "ej5.png", "amartinis", "Inicial", "natyd", "ej5.m4a"
, false, "");
let nuevaTareaEntregada6 = new ClaseTareasEntregadas(2, "Ejercicio 3", "Toc
ar con cualquier instrumento", "ej3.png", "brucewayne", "Intermedio", "gabi
b", "ej3.m4a", true, "Excelente, continúe.");

tareaEntregada.push(nuevaTareaEntregada1);
tareaEntregada.push(nuevaTareaEntregada2);
tareaEntregada.push(nuevaTareaEntregada3);
tareaEntregada.push(nuevaTareaEntregada4);
tareaEntregada.push(nuevaTareaEntregada5);
tareaEntregada.push(nuevaTareaEntregada6);

}

```

6. Código Javascript – funcionesParametros.js

```
//valida que la password cumpla con los requisitos exigidos por la letra.
function ControlarPassword(pPass) {
    let resultadoValido = false;
    let minuscula = false;
    let mayuscula = false;
    let numero = false;

    for(let i = 0; i < pPass.length; i++) {
        if(funcionCharCodeAt(pPass, i, 47, 58)) { //codigo ASCII numeros.
            numero = true;
        } else if(funcionCharCodeAt(pPass, i, 64, 91)) { //codigo ASCII letras mayusculas
            mayuscula = true;
        } else if(funcionCharCodeAt(pPass, i, 96, 123)) { //codigo ASCII letras minusculas
            minuscula = true;
        }
    }

    if(numero && mayuscula && minuscula && pPass.length > 3) {
        resultadoValido = true;
    }
    return resultadoValido;
}

//Funcion que controla los datos ingresados, si todos son correctos guarda al profesor
function validarGuardandoUsuarioProfesor(pNombre, pUsuario, pPassword, pTipoUsuario) {
    let datosValidos = pNombre != "" && pUsuario != "" && ControlarPassword(pPassword) && !existeUsuario(pUsuario);

    if(datosValidos) {
        let usuarioLowerCase = pUsuario.toLowerCase();
        let nuevoProfesor = new ClaseUsuarios(pNombre, usuarioLowerCase, pPassword, pTipoUsuario, "", "");
        usuariosGuardados.push(nuevoProfesor);
        console.log(usuariosGuardados);
    }
}

//busca que el usuario ingresado no existe de antemano.
function existeUsuario(pUsuario) {
    let usuarioExistente = false;
    let i = 0;
    let usuarioExistenteLowerCase = pUsuario.toLowerCase();
```



```

        while(i < usuariosGuardados.length && !usuarioExistente) {
            if(usuariosGuardados[i].nombreUsuario == usuarioExistenteLowerCase)
            {
                usuarioExistente = true;
            }
            i++;
        }

        return usuarioExistente;
    }
}
//Funcion que controla los datos ingresados, si todos son correctos guarda
al alumno
function validarGuardandoUsuarioAlumno(pNombre, pUsuario, pPassword, pTipoU
suario, pProfesorResponsable, pNivel) {
    let datosValidos = pNombre != "" && pUsuario != "" && ControlarPassword
(pPassword) && existeUsuario(pProfesorResponsable) && !existeUsuario(pUsuar
io);
    if(datosValidos) {
        let usuarioLowerCase = pUsuario.toLowerCase();
        let nuevoAlumno = new ClaseUsuarios(pNombre, usuarioLowerCase, pPas
sword, pTipoUsuario, pProfesorResponsable.toLowerCase(), pNivel);
        usuariosGuardados.push(nuevoAlumno);
        console.log(usuariosGuardados);
    }
}
//funcion para simplificar el charcodeAT (codigo ASCII)
function funcionCharCodeAt(pPass, pContador, pCodigo1, pCodigo2) {
    resultado = false;
    if(pPass.charCodeAt(pContador) > pCodigo1 && pPass.charCodeAt(pContador
) < pCodigo2) {
        resultado = true;
    }

    return resultado;
}
//funcion que retorna mensajes de error en la pantalla de registro.
function msjsDeErrorDivRegistro(pNombre, pUsuario, pPass) {
    let msjDivReg = "";

    if(pNombre == "") {
        msjDivReg = "Ingrese un nombre por favor.";
    } else if(pUsuario == "") {
        msjDivReg = "Ingrese un nombre de usuario por favor.";
    } else if(ControlarPassword(pPass) == false) {
        msjDivReg = `
        Debe ingresar una contraseña con los siguientes requisitos:
        <ul>
            <li>4 caracteres o más</li>
            <li>Debe contener al menos una mayúscula</li>

```

```

        <li>Debe contener al menos una minúscula</li>
        <li>Debe contener al menos un número</li>
    </ul>
    `;
} else {
    msjDivReg = "Registro completado."
}

return msjDivReg;
}

//se genera la tabla para asignar el nivel de los alumnos, según el profeso
r que esté loggeado.
function generarTablaAsignarNivelDeAlumnos(pProfesor) {
    let mostrarTabla = "";
    let alMenosUnAlumno = false;
    mostrarTabla += `
        <table border=1px bordercolor="#ED4028" cellpadding="0">
            <tr>
                <th>Nombre</th>
                <th>Usuario</th>
                <th>Nivel Actual</th>
                <th>Seleccionar Nivel</th>
                <th bgcolor="#ED4028"></th>
            </tr>
    `;

    for(let i = 0; i < usuariosGuardados.length; i++) {
        if(usuariosGuardados[i].profesorResponsable == pProfesor) {
            alMenosUnAlumno = true;
            mostrarTabla += `
                <tr>
                    <td>${usuariosGuardados[i].nombrePersona}</td>
                    <td>${usuariosGuardados[i].nombreUsuario}</td>
                    <td>${usuariosGuardados[i].nivelAlumno}</td>
                    <td>
                        <select id="slctCambiarNivelDeAlumno_${i}">
                            <option value="" selected>Selecione...</option>
                            <option value="Inicial">Nivel Inicial</option>
                            <option value="Intermedio">Nivel Intermedio</op
tion>
                            <option value="Avanzado">Nivel Avanzado</option>
                        </select>
                    </td>
                    <td>
                        <input type="button" id=
btnCambiarNivel="${i}" class="btnCambiarNivel" value="Cambiar">
                    </td>
                </tr>
            `;
        }
    }
}

```

```

        </tr>
    `;
    }
}
if(alMenosUnAlumno) {
    mostrarTabla += "</table>";
} else {
    mostrarTabla = "No tienes alumnos asignados.";
}

return mostrarTabla;
}

//Busca el último caracter, lo utilizamos para quitar el fakepath en este caso estaríamos buscando una contrabarra.
function buscarUltimoCaracterEnTexto(textoDondeBuscar, caracterBuscado) {
    let resultado = 0;
    let bandera = true;

    let i = textoDondeBuscar.length - 1;
    while (i >= 0 && bandera) { //se busca de derecha a izquierda la posición de la primera barra (con iterador i--)
        if (textoDondeBuscar[i] == caracterBuscado) {
            resultado = i;
            bandera = false;
        }

        i--;
    }
    if(bandera) {
        resultado = (-1);
    }
    return resultado; //DEVUELVE EN FORMATOS NUMERO, LA POSICION DE LA BARRA
} //combinado con buscarultimocaracter, determinamos la posición donde arranca el nombre del archivo guardado.
function cortarTextoDesdePosicion(texto, posicion) {
    let textoParaRetornar = "";

    for (let i = posicion; i < texto.length; i++) {
        textoParaRetornar += texto[i]; //EMPIEZA A ARMAR EL TEXTO DESDE LA BARRA ENCONTRADA (ULTIMA BARRA), HASTA EL FINAL , ACA DE IZQUIERDA A DERECHA.
    }

    //RETORNA EL TEXTO ARMADO DESDE LA BARRA ENCONTRADA
    return textoParaRetornar;
} //Elimina el fakepath ccon ayuda de las otras dos funciones.
function eliminarFakePath(pathCompleto) {

```

```

    let posicionUltimaBarra = buscarUltimoCaracterEnTexto(pathCompleto, "\\");
    let nombreArchivo = cortarTextoDesdePosicion(pathCompleto, posicionUltimaBarra + 1);

    return nombreArchivo;
}

//controla y verifica que los datos de tareas sean validos. y los pushea en el array
function guardarTarea(pTituloPush, pDescripcionPush, pNivelPush, pImgPush, pProfPush) {
    let guardarTarea = new ClaseTareas(pTituloPush, pDescripcionPush, pImgPush, pProfPush, pNivelPush, null, null, false, "");
    crearTarea.push(guardarTarea);
}

//controla los limites de caracteres por letra.
function limiteCaracteres(pTitulo, pDescripcion) {
    let tituloSinEspacios = pTitulo.trim();
    let descripcionSinEspacios = pDescripcion.trim();

    let validar = false;
    let condicion = (tituloSinEspacios.length + descripcionSinEspacios.length) <= 200 && (tituloSinEspacios.length + descripcionSinEspacios.length) >= 20 && tituloSinEspacios != "" && descripcionSinEspacios != "";
    if(condicion) {
        validar = true;
    }
    return validar;
}

//verifica que la tarea pendiente tenga datos válidos para poder guardarla en el array
function validarGuardandoTareaPendiente(pTituloTarea, pDescripcionTarea, pNivelTarea, pImgTarea, pProfesorResp) {
    let validacion = false;

    let imgSinFP = eliminarFakePath(pImgTarea);

    let i = 0;
    let encontrado = false;

    while (i < crearTarea.length && !encontrado) {
        let crearTareaTituloLC = crearTarea[i].titulo.toLowerCase();
        let tituloTareaLC = pTituloTarea.toLowerCase();
        let crearTareaDescrpLC = crearTarea[i].descripcion.toLowerCase();
        let descripcionTareaLC = pDescripcionTarea.toLowerCase();
        let aValidar = crearTareaTituloLC == tituloTareaLC && crearTareaDescrpLC == descripcionTareaLC && crearTarea[i].profesorResponsable == pProfesorResp && crearTarea[i].nivelAlumno == pNivelTarea;
    }
}

```

```

        if(aValidar) {
            encontrado = true;
        }
        i++;
    }
    if(!encontrado) {
        if(imgSinFP != "" && limiteCaracteres(pTituloTarea, pDescripcionTarea) && pNivelTarea != "") {
            guardarTarea(pTituloTarea, pDescripcionTarea, pNivelTarea, imgSinFP, pProfesorResp);
            validacion = true;
        }
    }

    return validacion;
} //le agrega el audio a la tarea cargada por el profesor, y se guarda en nuevo array.
function validarGuardandoTareaParaCalificar(pAudio) {
    let entregado = false;

    if(tareaEntregada.length == 0) {
        let nuevaTareaEntregada = new ClaseTareasEntregadas(entregarTarea.id, entregarTarea.titulo, entregarTarea.descripcion, entregarTarea.img, entregarTarea.profesorResponsable, usuarioSesionIniciada.nivelAlumno, usuarioSesionIniciada.nombreUsuario, pAudio, false, "")
        tareaEntregada.push(nuevaTareaEntregada);
    } else {
        let i = 0;
        while (i < tareaEntregada.length && !entregado) {
            if(tareaEntregada[i].id == entregarTarea.id && tareaEntregada[i].usuarioQueEntrega == usuarioSesionIniciada.nombreUsuario) {
                entregado = true;
            }
            i++;
        }
        if(!entregado) {
            let nuevaTareaEntregada = new ClaseTareasEntregadas(entregarTarea.id, entregarTarea.titulo, entregarTarea.descripcion, entregarTarea.img, entregarTarea.profesorResponsable, usuarioSesionIniciada.nivelAlumno, usuarioSesionIniciada.nombreUsuario, pAudio, false, "")
            tareaEntregada.push(nuevaTareaEntregada);
        }
    }
    return entregado;
}
//busca subcadenas de texto para los titulos y descripciones
function buscarSubCadenas(pCadena, pSubCadena) {
    let resultadoBuscador = false;
    let esCadena = "";

```

```

    let cadenaLowerCase = pCadena.toLowerCase();
    let subCadenaLowerCase = pSubCadena.toLowerCase();
    let i = 0;
    let j = 0;
    //Primero hace la busqueda en los titulos, en caso de falso pasa a la d
    escripcion.
    while(i < cadenaLowerCase.length && !resultadoBuscador) {
        if(cadenaLowerCase[i] == subCadenaLowerCase[j]) {
            let k = i;

            while(j < subCadenaLowerCase.length && cadenaLowerCase[k] == su
            bCadenaLowerCase[j]) {
                esCadena += cadenaLowerCase[k];
                k++;
                j++;
            }

            if(esCadena == subCadenaLowerCase) {
                resultadoBuscador = true;
            } else {
                esCadena = "";
                j = 0;
            }
        }
        i++;
    }
    return resultadoBuscador;
}

//Cuenta cantidad de tareas para las estadísticas.
function contarTareas(pArrayDeTareas) {
    let cantidad = 0;

    for(let i = 0; i < pArrayDeTareas.length; i++) {
        if(usuarioSesionIniciada.profesorResponsable == pArrayDeTareas[i].p
        rofesorResponsable && usuarioSesionIniciada.nivelAlumno == pArrayDeTareas[i
        ].nivelAlumno) {
            cantidad++;
        }
    }
    return cantidad;
}

//cuenta las tareas comentadas por el profesor para las estadísticas
function contarTareasComentadas(pArrayDeTareas) {
    let cantidad = 0;
    for(let i = 0; i < pArrayDeTareas.length; i++) {
        if(usuarioSesionIniciada.nombreUsuario == pArrayDeTareas[i].usuario
        QueEntrega && usuarioSesionIniciada.profesorResponsable == pArrayDeTareas[i
        ].profesorResponsable && usuarioSesionIniciada.nivelAlumno == pArrayDeTarea
        s[i].nivelAlumno && pArrayDeTareas[i].comentado) {
            cantidad++;
        }
    }
}

```

```

    }
}
return cantidad;

}

//Hace la búsqueda de el o los alumnos que mas tareas entregaron por profesor.
function alumnoMasTareasEntregadas(pProfesor) {
    let variableInfinitaNegativa = Number.NEGATIVE_INFINITY;
    let msj = "";
    let alMenosUnAlumno = false;
    for(let i = 0; i < usuariosGuardados.length; i++) { //si el usuario que
        está preguntando entregó mas tareas, sustituye el actual por este.
        if(usuariosGuardados[i].profesorResponsable == pProfesor) {
            alMenosUnAlumno = true;
            if(contarTareasPorAlumno(usuariosGuardados[i]) > variableInfinitaNegativa) {
                variableInfinitaNegativa = contarTareasPorAlumno(usuariosGuardados[i]);
                msj = `
                <table border="1px" cellspacing="0" bordercolor="#ED4028">
                <tr>
                <th>Nombre</th>
                <th>Cantidad Ejercicios resueltos</th>
                </tr>
                <tr>
                <td>${usuariosGuardados[i].nombrePersona}</td>
                <td align="center">${contarTareasPorAlumno(usuariosGuardados[i])}</td>
                </tr>
                `;
            } else if(contarTareasPorAlumno(usuariosGuardados[i]) == variableInfinitaNegativa){ // si el usuario que está preguntando entregó la misma cantidad de tareas, se agrega al anterior.
                msj += `
                <tr>
                <td>${usuariosGuardados[i].nombrePersona}</td>
                <td align="center">${contarTareasPorAlumno(usuariosGuardados[i])}</td>
                </tr>
                `;
            }
        }
    }
    //en caso de no haber resultados, muestra msj de error
    if(variableInfinitaNegativa == 0){
        msj = "Los alumnos no han resuelto tareas.";
    }
}

if (!alMenosUnAlumno) {
    msj = "No tienes alumnos asignados.";
}

```

```

    }
    return msj;
}
//se le pasa usuarios dinamicos y les calcula cuantas tareas tiene hechas.
En la function anterior es utilizada.
function contarTareasPorAlumno(pAlumno) {
    let contador = 0;

    for(let i = 0; i < tareaEntregada.length; i++) {
        if(tareaEntregada[i].usuarioQueEntrega == pAlumno.nombreUsuario) {
            contador++;
        }
    }

    return contador;
}
//Muestra las estadisticas de cuantas tareas fueron entregadas para el.
function totalTareasEntregadasPorProfesor(pProfesor) {
    let contador = 0;

    for(let i = 0; i < tareaEntregada.length; i++) {
        if(tareaEntregada[i].profesorResponsable == pProfesor) {
            contador++;
        }
    }

    return contador;
}
//Selector de alumnos en la pantalla del profesor para que muestre sus esta
disticas.
function slctInfoAlumnos(pProfesor) {
    let opciones = `
        <option value="" selected>Seleccione alumno...</option>
    `;
    for(let i = 0; i < usuariosGuardados.length; i++) {
        if(pProfesor == usuariosGuardados[i].profesorResponsable) {
            opciones += `
                <option value="${usuariosGuardados[i].nombreUsuario}">${usu
ariosGuardados[i].nombrePersona}</option>
            `;
        }
    }

    return opciones;
}
//calcula cuantas tareas fueron propuestas para ese alumno en ese nivel
function tareasPropuestasParaElAlumno(pAlumno) {
    let nivel = obtenerNivelAlumno(pAlumno);
    let tareasContadas = contarTareasPorNivelyProfesor(nivel, usuarioSesion
Iniciada.nombreUsuario);

```



```

    return tareasContadas;
}
//Cuenta las tareas para ese nivel segun el profesor responsable.
function contarTareasPorNivelyProfesor(pNivelTareas, pProfesorResponsable)
{
    let contador = 0;

    for(let i = 0; i < crearTarea.length; i++) {
        if(pNivelTareas == crearTarea[i].nivelAlumno && pProfesorResponsable == crearTarea[i].profesorResponsable) {
            contador++;
        }
    }
    return contador;
}
//cuenta las tareas entregadas por el alumno que se consulta en el nivel que pertenece.
function contarTareasEntregadasPorAlumnoPorNivel(pAlumno) {
    let contador = 0;
    let nivel = obtenerNivelAlumno(pAlumno);
    for(let i = 0; i < tareaEntregada.length; i++) {
        if(tareaEntregada[i].usuarioQueEntrega == pAlumno && tareaEntregada[i].nivelAlumno == nivel) {
            contador++;
        }
    }
    return contador;
}
//Devuelve el nivel del alumno que se está consultando
function obtenerNivelAlumno(pAlumno) {
    let encontrado = false;
    let i = 0;
    let nivel = "";
    while(i < usuariosGuardados.length && !encontrado) {
        if(usuariosGuardados[i].nombreUsuario == pAlumno) {
            nivel = usuariosGuardados[i].nivelAlumno;
            encontrado = true;
        }
        i++;
    }
    return nivel;
}

```

7. Código Javascript – menuFunciones.js

```
function ocultarTodo() { // funcion que utilizamos para ocultar todas las p
antallas y botones de la aplicacion, para luego mostrar solo lo que necesit
e.
    //Botones de navegacion principal y auxiliar
    mostrarOcultarBotonera("none", "none", "none", "none", "none", "none",
"none", "none", "none")
    //DIVS principales
    mostrarOcultar("divRegistrarse", "none");
    mostrarOcultar("divIniciarSesion", "none");
    //DIVS de profesor
    mostrarOcultar("menuPrincipalProfesor", "none");
    mostrarOcultar("divListaDeAlumnosPorProfesor", "none");
    mostrarOcultar("divCrearNuevoEjercicio", "none");
    mostrarOcultar("divComentar", "none");
    mostrarOcultar("mostrarPantallaEjercicioParaCalificar", "none");
    mostrarOcultar("divEstadisticasProfesor", "none");
    mostrarOcultar("divMsjInfoAlumno", "none");
    mostrarOcultar("msjErrorCalificar", "none");
    //DIVS de alumno
    mostrarOcultar("menuPrincipalAlumno", "none");
    mostrarOcultar("divTareasAsignadasAlAlumno", "none");
    mostrarOcultar("divRealizarNuevoEjercicio", "none");
    mostrarOcultar("divVerEjerciciosResueltos", "none");
    mostrarOcultar("divEstadisticasAlumno", "none");
}
//funcion de parametros, que utilizamos para sustituir al style.display...
//le pasamos id y forma de mostrar la pantalla (block, none, inline-block)
function mostrarOcultar(pId, pFormaDeMostrar) { // Funcion que utilizamos p
ara mostrar y ocultar pantallas
    document.querySelector(`#${pId}`).style.display = pFormaDeMostrar;
}
//sustituye al addEventListener para dar evento a los botones pasandole id
y nombre de la funcion.
function agregarEventoDeBotones(pIdBotones, pNombreFuncion) { // Funcion qu
e utilizamos para agregar evento a los botones
    document.querySelector(`#${pIdBotones}`).addEventListener("click", pNom
breFuncion);
}
//En cada parámetro se elegirá entre: "none" / "block" / "inline-block"
function mostrarOcultarBotonera(pIniciarSesion, pRegistro, pCerrarSesion, p
ListaAlumnos, pCrearEj, pCalificar, pVerTareas, pVerEjerciciosResueltos, pE
stadisticas) {
    mostrarOcultar("btnParaIniciarSesion", pIniciarSesion);
    mostrarOcultar("btnParaRegistrarse", pRegistro);
    mostrarOcultar("btnParaCerrarSesion", pCerrarSesion);
    mostrarOcultar("btnListaDeAlumnos", pListaAlumnos);
    mostrarOcultar("btnCrearNuevoEjercicio", pCrearEj);
}
```

```

    mostrarOcultar("btnCalificar", pCalificar);
    mostrarOcultar("btnVerTareas", pVerTareas);
    mostrarOcultar("btnVerEjerciciosResueltos", pVerEjerciciosResueltos);
    mostrarOcultar("btnEstadisticas", pEstadisticas);
}
//Se agregan todos los botones cuyo evento principal sea cambiar pantalla
function botonesCambiosDePantallas() {
    agregarEventoDeBotones("btnParaRegistrarse", mostrarDivRegistro);
    agregarEventoDeBotones("btnParaIniciarSesion", mostrarDivInicioSesion);
    agregarEventoDeBotones("btnListaDeAlumnos", mostrarDivAsignarNivelDeAlu
mnos);
    agregarEventoDeBotones("btnCrearNuevoEjercicio", mostrarDivCrearNuevaTa
rea);
    agregarEventoDeBotones("btnCalificar", mostrarDivParaComentarTareas);
    agregarEventoDeBotones("btnVerTareas", mostrarDivListadoTareas);
    agregarEventoDeBotones("btnVerEjerciciosResueltos", mostrarDivTodasLasT
areasEntregadasDelAlumno);
    agregarEventoDeBotones("btnEstadisticas", mostrarDivEstadisticas);
}
//botones con funcionalidades varias e importantes
function botonesConFuncionalidades() {
    agregarEventoDeBotones("btnGuardarUsuarioRegistrado", registrarNuevoUsu
ario);
    agregarEventoDeBotones("btnIniciarSesion", iniciarSesion);
    agregarEventoDeBotones("btnParaCerrarSesion", cerrarSesion);
    agregarEventoDeBotones("btnNuevaTarea", guardarNuevaTareaPendiente);
    agregarEventoDeBotones("btnEntregarTareaRealizada", entregarTareaPendie
nteDeCalificar);
    agregarEventoDeBotones("btnComentarTarea", calificarTarea);
    agregarEventoDeBotones("btnBuscarTareaParaRealizar", buscadorDeTareas);
}
//Pertenece a funcion con evento change, en el menú de registro. Segun el t
ipo de usuario muestra o no el selector de profesores.
function mostrarSelectorDeProfesoresRegistro() {
    let tipoUsuarioAlumno = document.querySelector("#slctTipoDeUsuario").va
lue;
    let mostrarComboProfesores = "";
    if(tipoUsuarioAlumno == "soyAlumno") {
        mostrarComboProfesores += `
        <label for="slctProfesorResponsable">Profesor Responsable:</lab
el>

        <select id="slctProfesorResponsable">
            <option value="" selected>
                Seleccione...
            </option>
            `
        for(let i = 0; i < usuariosGuardados.length; i++) {
            if(usuariosGuardados[i].tipoUsuario == "soyProf") {

```

```

        mostrarComboProfesores += `
            <option value="${usuariosGuardados[i].nombreUsuario}">
                ${usuariosGuardados[i].nombrePersona} (${usuariosGu
ardados[i].nombreUsuario})
            </option>
        `;
    }
}

mostrarComboProfesores += "</select>"

} else {
    mostrarComboProfesores += "";
}
document.querySelector("#divSelectorTipoDeUsuario").innerHTML = mostrar
ComboProfesores;
}
//muestra pantalla de registro de nuevos usuarios.
function mostrarDivRegistro() {
    ocultarTodo();
    mostrarOcultarBotonera("block", "block", "none", "none", "none", "none"
, "none", "none", "none");
    mostrarOcultar("divRegistrarse", "block");
}
//muestra pantalla de inicio de sesion para todos los usuarios
function mostrarDivInicioSesion() {
    ocultarTodo();
    mostrarOcultarBotonera("block", "block", "none", "none", "none", "none"
, "none", "none", "none");
    mostrarOcultar("divIniciarSesion", "block");
}
//funcion que utilizamos para ir cambiando las pantallas dentro de la sesio
n de profesor
function mostrarDivSesionIniciadaProfesor() {
    ocultarTodo();
    mostrarOcultarBotonera("none", "none", "block", "inline-block", "inline-
block", "inline-block", "none", "none", "inline-block");
    mostrarOcultar("menuPrincipalProfesor", "block");
}
//funcion que utilizamos para ir cambiando las pantallas dentro de la sesio
n de alumno
function mostrarDivSesionIniciadaAlumno() {
    ocultarTodo();
    mostrarOcultar("menuPrincipalAlumno", "block");
    mostrarOcultarBotonera("none", "none", "block", "none", "none", "none",
"inline-block", "inline-block", "inline-block");
}
//muestra la pantalla donde se generará la tabla para asignarles niveles a
los alumnos.

```

```

function mostrarDivAsignarNivelDeAlumnos() {
    mostrarDivSesionIniciadaProfesor();
    mostrarOcultar("divListaDeAlumnosPorProfesor", "block");

    let profesorResponsable = usuarioSesionIniciada.nombreUsuario;
    let tabla = generarTablaAsignarNivelDeAlumnos(profesorResponsable);

    document.querySelector("#tablaAsignacion").innerHTML = tabla;

    let btnCambiarNivelAlumno = document.querySelectorAll(".btnCambiarNivel");
    for(let i = 0; i < btnCambiarNivelAlumno.length; i++) {
        btnCambiarNivelAlumno[i].addEventListener("click", confirmarNivelDeAlumno);
    }
}
//Muestra la pantalla para que el profesor pueda crear una nueva tarea.
function mostrarDivCrearNuevaTarea() {
    mostrarDivSesionIniciadaProfesor();
    mostrarOcultar("divCrearNuevoEjercicio", "block");
    document.querySelector("#mensajeCrearTarea").innerHTML = "";
}
//muestra la pantalla para que el profesor pueda comentar las tareas entregadas por los alumnos.
function mostrarDivParaComentarTareas() {
    mostrarDivSesionIniciadaProfesor();
    mostrarOcultar("divComentar", "block");
    generarTablaParaComentarEjercicios();
}
//muestra pantalla para ver ejercicio seleccionado y poder calificarlo.
function divVerEjercicio() {
    mostrarDivSesionIniciadaProfesor();
    mostrarOcultar("mostrarPantallaEjercicioParaCalificar", "block");
    mostrarEjercicioResuelto();
}
//Muestra el listado de tareas para que pueda realizar el alumno.
function mostrarDivListadoTareas() {
    mostrarDivSesionIniciadaAlumno();
    mostrarOcultar("divTareasAsignadasAlAlumno", "block");
    document.querySelector("#txtBuscarTareaCargadaPorProfesor").value = "";
    generarTablaDeTareas();
}
//abre nueva pantalla donde se genera el ejercicio para entregar
function divMostrarRealizarNuevoEjercicio() {
    mostrarDivSesionIniciadaAlumno();
    mostrarOcultar("divRealizarNuevoEjercicio", "block");
    pantallaEntregarEjercicio();
}
//Se genera el ejercicio para entregar.
function pantallaEntregarEjercicio() {
    let pantalla = `
        <h3>${entregarTarea.titulo}</h3>
        <p>${entregarTarea.descripcion}</p>
        
    `
}

```

```

        <br>
        document.querySelector("#funcionGenerarNuevoEjercicio").innerHTML = pantalla;
    } // Muestra pantalla donde se genera la tabla con todas las tareas entregadas por el alumno y si fueron o no comentadas por el profesor.
    function mostrarDivTodasLasTareasEntregadasDelAlumno() {
        mostrarDivSesionIniciadaAlumno();
        mostrarOcultar("divVerEjerciciosResueltos", "block");
        tablaDeTareasEntregadas();
    } // muestra la pantalla de estadísticas dependiendo del tipo de usuario que está loggeado.
    function mostrarDivEstadisticas() {
        if(usuarioSesionIniciada.tipoUsuario == "soyProf") {
            mostrarDivSesionIniciadaProfesor();
            mostrarDivEstadisticasProfesor();
        } else {
            mostrarDivSesionIniciadaAlumno();
            mostrarDivEstadisticasAlumno();
        }
    } // Mensajes de informacion que se muestran segun el metodo change del selector o desplegable de la página.
    function mostrarDivInfoAlumnos() {
        let opcionAlumno = document.querySelector("#slctInfoAlumnos").value;
        let msj = "";
        let tareasPropuestasParaSuNivel = tareasPropuestasParaElAlumno(opcionAlumno);
        let tareasEntregadasPorAlumnoEnSuNivel = contarTareasEntregadasPorAlumnoPorNivel(opcionAlumno);

        if(opcionAlumno != "") {
            mostrarOcultar("divMsjInfoAlumno", "block");
            msj = `El alumno ha entregado ${tareasEntregadasPorAlumnoEnSuNivel} de ${tareasPropuestasParaSuNivel} tareas propuestas en su nivel actual.`;
        } else {
            msj = "";
        }

        document.querySelector("#divMsjInfoAlumno").innerHTML = msj;
    }
}

```

8. Código Javascript – miCodigoPrincipal.js

```
inicializar();

function inicializar() {
    //Hacemos la precarga de datos, y hacemos llamada de todos los botones
    que dan funcionalidad a la aplicación
    ocultarTodo();
    mostrarOcultarBotonera("block", "block", "none", "none", "none", "none"
, "none", "none", "none");
    botonesCambiosDePantallas();
    botonesConFuncionalidades();
    precargaDatos();

    //Evento de cambio de visualizaciones para los combo selects.
    document.querySelector("#slctTipoDeUsuario").addEventListener("change",
mostrarSelectorDeProfesoresRegistro);
    document.querySelector("#slctInfoAlumnos").addEventListener("change", m
ostrarDivInfoAlumnos);
    console.log(usuariosGuardados);
    console.log(tareaEntregada);
    console.log(crearTarea);
}

//Función que recibe los datos del HTML y permite o no el registro de un nu
evo usuario.
function registrarNuevoUsuario() {
    let nuevoNombrePersona = document.querySelector("#txtNombreRegistrarse"
).value;
    let nuevoNombreUsuario = document.querySelector("#txtNombreUsuarioRegis
trarse").value;
    let nuevoPassword = document.querySelector("#txtPasswordRegistrarse").v
alue;
    let tipoUsuario = document.querySelector("#slctTipoDeUsuario").value;
    let msjDivReg = "";

    //Pregunta si el usuario es profesor o Alumno, para luego validar según
    el tipo de usuario si los datos son correctos.
    if(tipoUsuario == "soyProf") {
        //Pregunta si ya hay un usuario guardado con ese nombre (de usuario
    ).

        if(!existeUsuario(nuevoNombreUsuario)) {
            //En caso de que no exista, y los datos ingresados sean validos
            guarda el nuevo usuario PROFESOR.
            validarGuardandoUsuarioProfesor(nuevoNombrePersona, nuevoNombre
Usuario, nuevoPassword, tipoUsuario);
            msjDivReg = msjsDeErrorDivRegistro(nuevoNombrePersona, nuevoNom
breUsuario, nuevoPassword);
        }
    }
}
```

```

        } else {
            msjDivReg = "Nombre de usuario ya utilizado, ingrese otro por favor.";
        }
    } else if(tipoUsuario == "soyAlumno") {
        let profesorResponsable = document.querySelector("#slctProfesorResponsable").value;
        if(!existeUsuario(nuevoNombreUsuario)) {
            //Si no existe ese nombre de usuario, y el tipo de usuario es alumno, guarda nuevo alumno en el nivel inicial. (Si los datos cumplen la validación).
            validarGuardandoUsuarioAlumno(nuevoNombrePersona, nuevoNombreUsuario, nuevoPassword, tipoUsuario, profesorResponsable, "Inicial");
            if(profesorResponsable == "") {
                msjDivReg = "Debe elegir un profesor.";
            } else {
                msjDivReg = msjsDeErrorDivRegistro(nuevoNombrePersona, nuevoNombreUsuario, nuevoPassword);
            }
        } else {
            msjDivReg = "Nombre de usuario ya utilizado, ingrese otro por favor.";
        }
    } else {
        msjDivReg = "Seleccione tipo de usuario.";
    }

    document.querySelector("#mensajeDivError").innerHTML = msjDivReg;
    document.querySelector("#txtNombreRegistrarse").value = "";
    document.querySelector("#txtNombreUsuarioRegistrarse").value = "";
    document.querySelector("#txtPasswordRegistrarse").value = "";
}

//Funcion que recibe los datos del HTML y busca si existe el usuario y su contraseña es correcta.
function iniciarSesion() {
    let nombreUsuario = document.querySelector("#txtIniciarSesionUsuario").value.toLowerCase();
    let pass = document.querySelector("#txtIniciarSesionPassword").value;

    let i = 0;
    let usuarioEncontrado = false;
    let msjError = "";
    //Busca el usuario.
    while(i < usuariosGuardados.length && !usuarioEncontrado) {
        if(nombreUsuario == usuariosGuardados[i].nombreUsuario) {
            if(pass == usuariosGuardados[i].passwordUsuario) {
                usuarioEncontrado = true;
                usuarioSesionIniciada = usuariosGuardados[i];
            }
        }
    }
}

```



```

    }
    i++;
}
//Si lo encuentra, dependiendo del tipo de usuario, ingresará al div que le corresponda.
if(usuarioEncontrado) {
    if(usuarioSesionIniciada.tipoUsuario == "soyProf") {
        mostrarDivSesionIniciadaProfesor();
    } else {
        mostrarDivSesionIniciadaAlumno();
    }
} else {
    msjError = "Usuario y/o contraseña errónea. Intente nuevamente.";
    document.querySelector("#divMsjErrorAlIniciarSesion").innerHTML = msjError;
}
document.querySelector("#txtIniciarSesionUsuario").value = "";
document.querySelector("#txtIniciarSesionPassword").value = "";
console.log(usuarioSesionIniciada);
}
//Cierra cualquier sesion iniciada ocultando todos los menús y borrando los datos de las variables globales.
function cerrarSesion() {
    ocultarTodo();
    mostrarOcultarBotonera("block", "block", "none", "none", "none", "none", "none", "none", "none");
    usuarioSesionIniciada = null;
    variableInfinitaNegativa = Number.NEGATIVE_INFINITY;
}
//Función que permite el cambio de nivel del alumno.
function confirmarNivelDeAlumno() {
    //Llama a todos los botones que tienen el atributo "id-btnCambiarNivel" para darle evento a los botones dinámicos.
    let btnCambiarNivel = this.getAttribute("id-btnCambiarNivel");
    let nivelSeleccionado = document.querySelector(`#slctCambiarNivelDeAlumno_${btnCambiarNivel}`).value;
    let i = 0;
    let bandera = true;
    let msj = "";

    while(i < usuariosGuardados.length && bandera) {
        if(btnCambiarNivel == usuariosGuardados[i].id) {
            //Si el nivel seleccionado es intermedio o avanzado, y el usuario pertenece al inicial.
            if(usuariosGuardados[i].nivelAlumno == "Inicial" && (nivelSeleccionado == "Intermedio" || nivelSeleccionado == "Avanzado")) {
                usuariosGuardados[i].nivelAlumno = nivelSeleccionado;
            }
        }
    }
}

```

```

        bandera = false;
        msj = `
            Al usuario ${usuariosGuardados[i].nombrePersona} se le
asignó el nivel ${nivelSeleccionado} correctamente.
        `;

        //Si el usuario es nivel intermedio y seleccion el avanzado
, me permite cambiarlo
    } else if(usuariosGuardados[i].nivelAlumno == "Intermedio" && n
ivelSeleccionado == "Avanzado") {
        usuariosGuardados[i].nivelAlumno = nivelSeleccionado;
        bandera = false;
        msj = `
            Al usuario ${usuariosGuardados[i].nombrePersona} se le
asignó el nivel ${nivelSeleccionado} correctamente.
        `;
    } else {
        //si no se selecciona nivel da error.
        if(nivelSeleccionado == "") {
            msj = "Debe seleccionar un nivel.";
            //Si se selecciona el mismo nivel que ya tiene asignado
, da otro error.
        } else if(nivelSeleccionado == usuariosGuardados[i].nivelAl
umno) {
            msj = `
                ${usuariosGuardados[i].nombrePersona} ya está asign
ado al nivel ${nivelSeleccionado}.
            `;
            //Dejamos el else para aquellos cambios de retroceso de
nivel.
        } else {
            msj = `
                No es posible cambiar a ${usuariosGuardados[i].nombrePe
rsona} del nivel ${usuariosGuardados[i].nivelAlumno} al nivel ${nivelSelec
cionado}.
            `;
        }
    }
}
i++;
}
//actualiza la tabla de alumnos y su nivel correspondiente.
mostrarDivAsignarNivelDeAlumnos();
document.querySelector("#msjCambioNivel").innerHTML = msj;
}
//Funcion para guardar nuevas tareas en el array de tareas.
function guardarNuevaTareaPendiente() {
    let tituloTarea = document.querySelector("#tituloDelNuevoEjercicio").va
lue;

```

```

    let descripcionTarea = document.querySelector("#descripcionNuevoEjercicio").value;
    let nivelTarea = document.querySelector("#slctNivelDelEjercicio").value;
;
    let imgTarea = document.querySelector("#imgNuevaTarea").value;
    let profesorResp = usuarioSesionIniciada.nombreUsuario;

    let mensaje = "";
    //Si los datos ingresados en el HTML son válidos, pushea la tarea en el array.
    if(validarGuardandoTareaPendiente(tituloTarea, descripcionTarea, nivelTarea, imgTarea, profesorResp)) {
        mensaje = `
            Tarea guardada al nivel ${nivelTarea} exitosamente.
        `;

        //Se vacían los campos.
        document.querySelector("#tituloDelNuevoEjercicio").value = "";
        document.querySelector("#descripcionNuevoEjercicio").value = "";
        document.querySelector("#imgNuevaTarea").value = "";
        document.querySelector("#slctNivelDelEjercicio").value = "";

    } else {
        //Por si la tarea no cumple con los requisitos de título y descripción.
        if(!limiteCaracteres(tituloTarea, descripcionTarea)) {
            mensaje =
                "Para guardar la tarea, al menos debe haber 20 caracteres entre título y descripción, y no más de 200. Además no pueden quedar vacíos."
        };

        } else if(imgTarea == "") {
            mensaje = "Debe seleccionar una imagen para cargar la tarea.";
        } else if(nivelTarea == "") {
            mensaje = "Debe seleccionar un nivel para asignar.";
        } else {
            mensaje = `Ya existe la tarea para el nivel ${nivelTarea}.`;
        }

    }

    console.log(crearTarea);

    document.querySelector("#mensajeCrearTarea").innerHTML = mensaje;
}
//Tabla con las tareas finalizadas por alumno para cada profesor, que estén pendientes de calificar.
function generarTablaParaComentarEjercicios() {
    let tabla = `
        <table border="1" cellspacing="0" bordercolor="#ED4028">
            <tr>

```

```

        <th>Título</th>
        <th>Descripción</th>
        <th>Alumno</th>
        <th bgcolor="#ED4028"></th>
    </tr>
`;
    let contadorTareas = 0;
    let tareasEncontradas = false; //variable que usamos como bandera para
    decir que encontramos tareas para el profesor

    if(tareaEntregada.length != 0) {
        for(let i = 0; i < tareaEntregada.length; i++) {
            if(tareaEntregada[i].profesorResponsable == usuarioSesionInicia
da.nombreUsuario && !tareaEntregada[i].comentado) {
                tareasEncontradas = true;
                contadorTareas++;
                tabla += `
                    <tr>
                        <td>${tareaEntregada[i].titulo}</td>
                        <td>${tareaEntregada[i].descripcion}</td>
                        <td>${tareaEntregada[i].usuarioQueEntrega}</td>
                        <td>
                            <input type="button" id-
btnVerEjercicio="${i}" class="btnVerEjercicio" value="Ver">
                        </td>
                    </tr>
                `;
            }
        }
    }

    tabla += "</table>";

    if(contadorTareas == 0 || !tareasEncontradas) { // esta condicion se ar
mó, por si no hay tareas que mostrar, pisamos la tabla que fuimos creando.
        tabla = "No se encuentran tareas para calificar.";
    }

    document.querySelector("#tablaEjerciciosParaCalificar").innerHTML = tab
la;

    let btnVerEjercicios = document.querySelectorAll(".btnVerEjercicio");
    for(let i = 0; i < btnVerEjercicios.length; i++) {
        btnVerEjercicios[i].addEventListener("click", verEjercicio);
    }
}

//Llama a la función que muestra el ejercicio para comentar.
function verEjercicio() {
    let btnVerEjercicio = this.getAttribute("id-btnVerEjercicio");

```

```

    let i = 0;
    let bandera = true;

    while(i < tareaEntregada.length && bandera) {
        if(btnVerEjercicio == i) {
            comentarTarea = tareaEntregada[i];
            divVerEjercicio(); // Llama a la función que oculta y muestra p
antallas según lo que necesita mostrar, para ir a comentar el ejercicio.
            bandera = false;
        }
        i++;
    }
}

//Muestra en pantalla el ejercicio resuelto para comentar.
function mostrarEjercicioResuelto() {
    let nuevaPantalla = `
    <h3>${comentarTarea.titulo}</h3>
    <p>${comentarTarea.descripcion}</p>
    
    <br>
    <audio controls>
        <source src="audio/${comentarTarea.audio}" type="audio/mp3">
    </audio>
    `;

    document.querySelector("#divVerEjercicioParaCalificar").innerHTML = nuevaPa
ntalla;
}

//Función que le da evento a a comentar las tareas realizadas por los alumn
os.
function calificarTarea() {

    let comentarioProfe = document.querySelector("#txtComentarioProfesor").
value;
    let i = 0;
    let bandera = true;
    let msjError = "";
    //Pregunta si el comentario del profesor no es vacío.
    if(comentarioProfe != "") {
        //Cambia el estado de tarea comentada a true, y pisa el comentario
"vacío" de esa tarea.
        while(i < tareaEntregada.length && bandera) {
            if(tareaEntregada[i].id == comentarTarea.id && tareaEntregada[i
].alumnoQueEntrega == comentarTarea.alumnoQueEntrega && tareaEntregada[i].c
omentado == false) {
                tareaEntregada[i].comentario = comentarioProfe;
                tareaEntregada[i].comentado = true;
                bandera = false;
            }
        }
    }
}

```

```

        alert("Comentarios guardados.")
        console.log(tareaEntregada[i]);
        mostrarDivParaComentarTareas(); // retorna a la pantalla do
nde muestra las tareas pendientes de comentar.
    }
    i++;
}
} //Muestra mensajes de error según el tipo.
if(bandera && comentarioProfe != "") {
    mostrarOcultar("msjErrorCalificar", "block");
    msjError = "La tarea ya fue comentada.";
} else {
    mostrarOcultar("msjErrorCalificar", "block");
    msjError = "Debe ingresar un comentario.";
}

document.querySelector("#msjErrorCalificar").innerHTML = msjError;
document.querySelector("#txtComentarioProfesor").value = "";
}
//Se muestra la tabla de tareas para que el alumno las pueda realizar.
function generarTablaDeTareas() {
    let tabla = `
        <table border="1px" cellspacing="0" bordercolor="#ED4028">
            <tr>
                <th>Título</th>
                <th>Descripción</th>
                <th bgcolor="#ED4028"></th>
            </tr>

`;
    let tareaEncontrada = false;

    for(let i = 0; i < crearTarea.length; i++) {
        if(usuarioSesionIniciada.nivelAlumno == crearTarea[i].nivelAlumno &
        & usuarioSesionIniciada.profesorResponsable == crearTarea[i].profesorRespon
sable) {
            tabla += `
                <tr>
                    <td>${crearTarea[i].titulo}</td>
                    <td>${crearTarea[i].descripcion}</td>
                    <td>
                        <input type="button" id-
btnRealizarEjercicio="${i}" class="btnRealizarEjercicio" value="Realizar">
                    </td>
                </tr>
            `;
            tareaEncontrada = true;
        }
    }
    tabla += "</table>";
    if(!tareaEncontrada) {

```

```

        tabla = "No se encontraron tareas asignadas.";
    }
    document.querySelector("#tablaTareasAsignadasPorAlumno").innerHTML = ta
bla;

    let btnRealizarEjercicios = document.querySelectorAll(".btnRealizarEjer
cicio");
    for(let i = 0; i < btnRealizarEjercicios.length; i++) {
        btnRealizarEjercicios[i].addEventListener("click", mostrarDivEjer
cicio);
    }
}
//le da evento a los botones dinamicos de las tareas a realizar.
function mostrarDivEjercicio() {
    let btnRealizarEjercicio = this.getAttribute("id-
btnRealizarEjercicio");

    let i = 0;
    let bandera = true;

    while(i < crearTarea.length && bandera) {
        if(btnRealizarEjercicio == i) {
            entregarTarea = crearTarea[i];
            divMostrarRealizarNuevoEjercicio();
            console.log(entregarTarea);
            bandera = false;
        }
        i++;
    }
}
//Valida los datos de la tarea y los guarda en el array
function entregarTareaPendienteDeCalificar() {
    let audio = document.querySelector("#btnAudiosDeAlumno").value;
    let audioSinFP = eliminarFakePath(audio);//elimina el C://fakepath que
se agrega al guardar una imagen o audio.
    let msj = "";
    //Exige que haya un audio cargado para poder validar la tarea calificad
a.
    if(audio != "") {
        if(!validarGuardandoTareaParaCalificar(audioSinFP)) {
            console.log(tareaEntregada);
            alert("La tarea se entregó con éxito.");
            mostrarDivListadoTareas();
            document.querySelector("#btnAudiosDeAlumno").value = "";
        } else {
            alert("La tarea ya fue entregada.");
            mostrarDivListadoTareas();
            document.querySelector("#btnAudiosDeAlumno").value = "";
        }
    }
}

```

```

    } else {
        msj = "Debe cargar un archivo de audio.";
    }
    document.querySelector("#parrafoMsj").innerHTML = msj;
}

//Muestra una tabla con las tareas que entregó el alumno y si fueron coment
adas por el profesor responsable.
function tablaDeTareasEntregadas() {
    let tabla = `
        <table border="1" cellspacing="0">
            <tr>
                <th>Título</th>
                <th>Descripción</th>
                <th>Audio</th>
                <th>Comentarios</th>
            </tr>
        `;
    let alMenosUnaTarea = false; //variable que utilizo para validar mostra
r un mensaje u otro.

    for(let i = 0; i < tareaEntregada.length; i++) {
        if(tareaEntregada[i].usuarioQueEntrega == usuarioSesionIniciada.nom
breUsuario) {
            tabla += `
                <tr>
                    <td>${tareaEntregada[i].titulo}</td>
                    <td>${tareaEntregada[i].descripcion}</td>
                    <td>
                        <audio controls>
                            <source src="audio/${tareaEntregada[i].audio}"
type="audio/mp3">
                            <audio>
                        </td>
                    <td>${tareaEntregada[i].comentario}</td>
                </tr>
            `;
            alMenosUnaTarea = true;
        }
    }

    tabla += "</table>";
    if(!alMenosUnaTarea) {
        tabla = "No se encontraron tareas entregadas.";
    }
    document.querySelector("#divVerEjerciciosResueltos").innerHTML = tabla;
}
//Busca las tareas o por titulo, o por descripción, no por ambas.

```



```

function buscadorDeTareas() {
    let txtSubCadena = document.querySelector("#txtBuscarTareaCargadaPorProfesor").value;
    let mostrarTabla = `
        <table>
            <tr>
                <th>Título</th>
                <th>Descripción</th>
                <th bgcolor="#ED4028"></th>
            </tr>
    `;

    let tituloOk = false; // utilizo para decir si encontré una subcadena en el título, y así evitar buscar en la descripción
    let descripcionOk = false;
    // busca en los títulos
    for(let i = 0; i < crearTarea.length; i++) {
        if(buscarSubCadenas(crearTarea[i].titulo, txtSubCadena) && usuarioSesionIniciada.profesorResponsable == crearTarea[i].profesorResponsable && usuarioSesionIniciada.nivelAlumno == crearTarea[i].nivelAlumno) {
            mostrarTabla += `
                <tr>
                    <td>${crearTarea[i].titulo}</td>
                    <td>${crearTarea[i].descripcion}</td>
                    <td><input type="button" id=
            btnRealizarEjercicio="${i}" class="btnRealizarEjercicio" value="Realizar"><
            /td>

                </tr>
            `;
            tituloOk = true;
        }
    }

    // Si no se encontro subcadena en los títulos, pasa a buscar en los comentarios.
    if(!tituloOk) {
        for(let i = 0; i < crearTarea.length; i++) {
            if(buscarSubCadenas(crearTarea[i].descripcion, txtSubCadena) && usuarioSesionIniciada.profesorResponsable == crearTarea[i].profesorResponsable && usuarioSesionIniciada.nivelAlumno == crearTarea[i].nivelAlumno) {
                mostrarTabla += `
                    <tr>
                        <td>${crearTarea[i].titulo}</td>
                        <td>${crearTarea[i].descripcion}</td>
                        <td><input type="button" id=
            btnRealizarEjercicio="${i}" class="btnRealizarEjercicio" value="Realizar"><
            /td>

                    </tr>
                `;
                descripcionOk = true;
            }
        }
    }
}

```

```

    }
  }
}

mostrarTabla += `</table>`;
if(!tituloOk && !descripcionOk) { //si ninguna se encuentra, muestra un
mensaje de error.
  mostrarTabla = "No hay resultados que coincidan con su búsqueda.";
}

document.querySelector("#tablaTareasAsignadasPorAlumno").innerHTML = mo
strarTabla;
//defino la funcion que le agrega evento a los botones dinámicos.
let btnRealizarEjercicios = document.querySelectorAll(".btnRealizarEjer
cicio");
for(let i = 0; i < btnRealizarEjercicios.length; i++) {
  btnRealizarEjercicios[i].addEventListener("click", mostrarDivEjerci
cio);
}
}

//muestra las estadísticas de los alumnos
function mostrarDivEstadisticasAlumno() {
  mostrarOcultar("divEstadisticasAlumno", "block");
  let contadorTareasParaHacer = contarTareas(crearTarea);
  let contadorTareasHechasPorAlumno = contarTareasEntregadasPorAlumnoPorN
ivel(usuarioSesionIniciada.nombreUsuario);
  let contadorTareasComentadas = contarTareasComentadas(tareaEntregada);
  let divMsj = "";
  console.log(contadorTareasParaHacer);
  console.log(contadorTareasHechasPorAlumno);

  let porcentajeEjerciciosResueltos = contadorTareasHechasPorAlumno * 100
/ contadorTareasParaHacer;
  let totalEjerciciosSinComentar = contadorTareasHechasPorAlumno -
contadorTareasComentadas;

  divMsj = `
    <ul>
      <li>Ha resuelto el ${porcentajeEjerciciosResueltos}% de los eje
rcicios planteados para el nivel ${usuarioSesionIniciada.nivelAlumno}.</li>
      <li>Ha recibido ${contadorTareasComentadas} comentarios de ejer
cicios en el nivel ${usuarioSesionIniciada.nivelAlumno}.</li>
      <li>Restan comentar ${totalEjerciciosSinComentar} ejercicios pa
ra el nivel ${usuarioSesionIniciada.nivelAlumno}.</li>
    </ul>
  `;

  if(contadorTareasParaHacer == 0) {

```

```

        divMsj = "No tienes ejercicios planteados para realizar.";
    }

    document.querySelector("#divEstadisticasAlumno").innerHTML = divMsj;
}

//muestra las estadísticas de los profesores.
function mostrarDivEstadisticasProfesor() {
    mostrarOcultar("divEstadisticasProfesor", "block");
    let profesor = usuarioSesionIniciada.nombreUsuario;
    let alumnoMasTareas = alumnoMasTareasEntregadas(profesor);
    let totalTareasEntregadasPorProf = totalTareasEntregadasPorProfesor(profesor);

    let msjTotalTareas = "";
    let infoAlumnos = slctInfoAlumnos(profesor);

    if(totalTareasEntregadasPorProf == 1) {
        msjTotalTareas = `En total se entregó ${totalTareasEntregadasPorProf} tarea.`;
    } else if(totalTareasEntregadasPorProf > 1) {
        msjTotalTareas = `En total se entregaron ${totalTareasEntregadasPorProf} tareas.`;
    }

    document.querySelector("#alumnosMasTareasEntregadas").innerHTML = alumnoMasTareas;
    document.querySelector("#totalTareasEntregadas").innerHTML = msjTotalTareas;
    document.querySelector("#slctInfoAlumnos").innerHTML = infoAlumnos;
}

```

9. Datos Precargados

- Usuario profesor: Bruno Díaz
 - Usuario: BruceWayne
 - Contraseña: Bdiaz1
 - Usuarios alumnos asignados:
 - Usuario: GonzaP - Contraseña: Gonza28294
 - Usuario: DiegoG - Contraseña: Diego123
 - Usuario: GabiB - Contraseña: Gabi1
 - Usuario: ManuS – Contraseña: Manu1

1 tarea asignada al nivel Inicial. – Dos tareas resueltas

2 tareas asignadas al nivel Intermedio – Una tarea resuelta y comentada.

1 tarea asignada al nivel Avanzado.

- Usuario profesor: Alejandro Martinis
 - Usuario: aMartinis
 - Contraseña: AleMartinis1
 - Usuarios alumnos asignados:
 - Usuario: NatyD - Contraseña: Naty1
 - Usuario: PabloF - Contraseña: Pablo1
 - Usuario: BrunoLic - Contraseña: Bruno1
 - Usuario: ManuT – Contraseña: Manu1

1 tarea asignada al nivel Inicial. – Una tarea resuelta pendiente de comentar

1 tarea asignada al nivel Intermedio. – Una tarea resuelta pendiente de comentar

2 tareas asignadas al nivel Avanzado. – Una tarea resuelta pendiente de comentar