

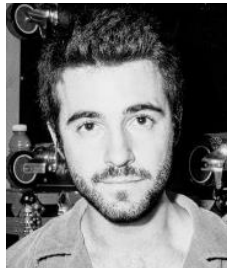
**Universidad ORT Uruguay**  
**Facultad de Ingeniería**  
**Escuela de Tecnología**

**OBLIGATORIO PROGRAMACIÓN 1**

**Documentación**



**[Francisco Aldado – 279264]**



**[Nicolás Bañales – 270543]**

**Grupo: M1D**

***Docente: Gonzalo Gentile***

**AP/ATI**

**Fecha de entrega del obligatorio: 24-06-2021**

# Índice Documentación

1.	Breve descripción del proyecto	7
2.	Como acceder a la aplicación	7
2.1	– Por precarga de usuarios	7
2.1.1	Tabla usuarios precargados de tipo Docente	7
2.1.2	Tabla usuarios precargados de tipo Alumno	8
2.1.3	Recomendación de acceso para probar la página.	8
2.2	– Por registro de nuevo usuario	9
2.2.1	Como registrarse en la aplicación	9
3.	Secciones de la aplicación y navegación	10
3.1.	– Secciones del Docente	10
3.1.1.	Cambiar nivel de alumnos	10
3.1.2.	Plantear Ejercicios	11
3.1.3.	Redactar devoluciones	12
3.1.4.	Ver información estadística	12
3.2.	– Secciones del Alumno	13
3.2.1.	Ver ejercicios planteados	13
3.2.2.	Realizar entrega de ejercicio	14
3.2.3.	Ver ejercicios resueltos	14
3.2.4.	Ver información estadística	15
3.3.	– Cerrar sesión (log out)	15
4.	Detalles del código de este proyecto	16
4.1.	– Archivos del proyecto	16
4.2.	– Precargas, clases y arrays	16

4.2.1.	Precargas	16
4.2.2.	Arrays	17
4.2.3.	Clases	17
4.3.	– Validaciones	18
4.4.	– Login y <i>cuentaActiva</i>	19
4.5.	– Logout y <i>cuentaActiva</i>	20
4.6.	–Ocultar y mostrar elementos	21
4.7.	–Manejo UI (Interfaz de usuario)	22
5.	Lista de funciones	24
6.	Informe de Testing	33
7.	Documento de Análisis	37
7.1.	Descripción del problema a resolver	37
7.1.2.	Tipos de usuario del sistema	37
7.1.3.	Listado de funcionalidades	37
7.2.	Detalle de Funcionalidades	39
7.2.1.	F01 – Registro en el sitio	39
7.2.2.	Acceso	39
7.2.3.	Descripción	39
7.2.4.	Interfaz de usuario	39
7.2.5.	Validaciones	40
7.3.	F02 – Ingreso (Log in) a la aplicación	41
7.3.1.	Acceso	41
7.3.2.	Descripción	41
7.3.3.	Interfaz de usuario	41
7.3.4.	Validaciones	41
7.4.	F03 – Asignar a sus alumnos un nivel	42

7.4.1.	Acceso	42
7.4.2.	Descripción	42
7.4.3.	Interfaz de usuario	42
7.4.4.	Validaciones	42
7.5.	F04 – Asignar ejercicios para sus alumnos de un determinado nivel	43
7.5.1.	Acceso	43
7.5.2.	Descripción	43
7.5.3.	Interfaz de usuario	43
7.5.4.	Validaciones	43
7.6.	F05 – Poder dejar comentarios de devolución de ejercicios entregados por los alumnos	44
7.6.1.	Acceso	44
7.6.2.	Descripción	44
7.6.3.	Interfaz de usuario	44
7.6.4.	Validaciones	44
7.7.	F06 – Visualizar información estadística (general) acerca de alumnos de un docente según ejercicios resueltos	45
7.7.1.	Acceso	45
7.7.2.	Descripción	45
7.7.3.	Interfaz de usuario	45
7.7.4.	Validaciones	45
7.8.	F07 – Poder ver y buscar los ejercicios que su docente plantea según el nivel	46
7.8.1.	Acceso	46
7.8.2.	Descripción	46
7.8.3.	Interfaz de usuario	46
7.8.4.	Validaciones	46

7.9.	F08 – Realizar la entrega de un ejercicio, adjuntando un audio correspondiente a dicha tarea	47
7.9.1.	Acceso	47
7.9.2.	Descripción	47
7.9.3.	Interfaz de usuario	47
7.9.4.	Validaciones	47
7.10.	F09 – Visualizar los ejercicios resueltos	48
7.10.1.	Acceso	48
7.10.2.	Descripción	48
7.10.3.	Interfaz de usuario	48
7.10.4.	Validaciones	48
7.11.	F10 – Visualizar información estadística (individual) acerca de ejercicios resueltos	49
7.11.1.	Acceso	49
7.11.2.	Descripción	49
7.11.3.	Interfaz de usuario	49
7.11.4.	Validaciones	49
7.12.	F11 – Cerrar la sesión (Log out)	50
7.12.1.	Acceso	50
7.12.2.	Descripción	50
7.12.3.	Interfaz de usuario	50
7.12.4.	Validaciones	50
8.	Código completo	51
8.1.	HTML (index.html)	51
8.2.	JavaScript	58
8.2.1.	clases.js	58

8.2.2.	codigo.js	59
8.2.3.	manejoUI.js	78
8.2.4.	precargas.js	108
8.3.	CSS (estilo.css)	121

# Documentación

## 1. Breve descripción del proyecto

El proyecto trata sobre un sitio de gestión de clases de una academia de música (Guitar Academy) que maneja dos tipos de usuario. Docentes y alumnos podrán hacer uso de esta plataforma para facilitar la comunicación y el aprendizaje respectivamente. Tanto docente como alumno, tendrán diferentes roles dentro de la aplicación. El docente podrá plantear ejercicios y el alumno al recibirlos podrá subir un audio con su interpretación. De esta forma, el docente podrá realizar una devolución de la tarea ejecutada. Esta y otras funciones podrán ser realizadas en el sitio, como se detallan en este documento.

## 2. Cómo acceder a la aplicación

Se podrá acceder a la aplicación con dos tipos de usuarios diferentes colocando usuario y contraseña en el login. Para ello, realizamos una precarga de datos con cuatro docentes (Imagen 01) y doce alumnos (Imagen 02). También puede registrarse en la aplicación si desea crear un usuario nuevo.

### 2.1 – Por precarga de usuarios

#### 2.1.1 Tabla usuarios precargados de tipo Docente

Tipo	Nombre	Usuario	Password
docente	Eric Clapton	eric1945	Heaven92
docente	Bob Dylan	bobbydylan	bWind1962
docente	Janis Joplin	lajanis	Rock1
docente	Amy Winehouse	amyrehab83	iSaidno1

*Imagen 1*

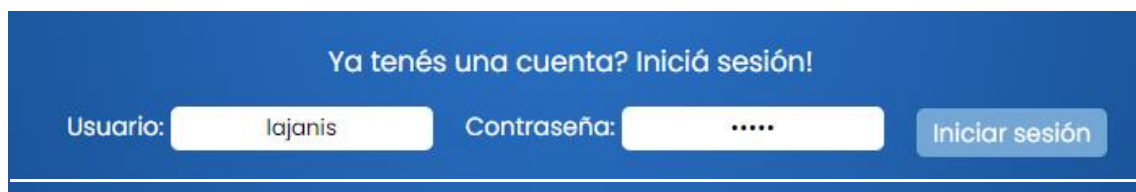
### 2.1.2 Tabla usuarios precargados de tipo Alumno

Tipo	Nombre	Usuario	Password	Nivel	Docente
alumno	Juan Carlos	juanca	Pass1	1	Janis Joplin
alumno	María del Mar	lamarydelcap	Firulais88	2	Eric Clapton
alumno	Roberto Carlos	robbieuru	Roca52	3	Eric Clapton
alumno	Julia Rivera	juliguitarra	iloveTacos1996	1	Eric Clapton
alumno	Federico Gimenez	elfede	Fefo3	1	Janis Joplin
alumno	Laura Gomez	laurasote	Lauchita1	3	Janis Joplin
alumno	Gabriel Richieri	gabo87	gaboteElbebote3	1	Bob Dylan
alumno	Ramiro Ortega	elramaorteguita	iloveMusic24	3	Bob Dylan
alumno	Erica Ramírez	erica7	Lapregunta99	2	Bob Dylan
alumno	Rafa Díaz	rafitamiles	hyperActive17	2	Amy Winehouse
alumno	Ivan Modernell	elivansabe	micChecker21	1	Amy Winehouse
alumno	Nicolas Rodríguez	estudio50	js4Life	1	Amy Winehouse

Imagen 2

### 2.1.3 Recomendación de acceso para probar la página

Desde la sección login (Imagen 03) recomendamos acceder de la siguiente manera. Con el usuario tipo docente **“lajanis”** y contraseña **“Rock1”** para probar las funcionalidades del profesor. Y con el usuario tipo alumno **“juanca”** y contraseña **“Pass1”** para probar las del alumno. Ya que las precargas de estos usuarios fueron pensadas para testear todas las diferentes funcionalidades de la aplicación.



Ya tenés una cuenta? Inicia sesión!

Usuario:  Contraseña:

Imagen 3



## 2.2 – Por registro de nuevo usuario

### 2.2.1 Como registrarse en la aplicación

En la página de inicio haciendo click en el botón “**Registrarme!**”, se abrirá una ventana como la de la Imagen 04, donde podrá llenar los campos requeridos, y seleccionar su tipo de usuario. Si cumplió con todas las validaciones, entrara con éxito a la página de su perfil. Si se registra como alumno, deberá elegir un profesor de la lista desplegable, se le será asignado el nivel Inicial automáticamente.

El formulario de registro, titulado "Registrarme!", contiene los siguientes elementos:

- Un botón de cierre en la esquina superior derecha.
- Campo de texto "Nombre:" con el placeholder "Ingrese su nombre".
- Campo de texto "Usuario:" con el placeholder "Ingrese su usuario".
- Campo de texto "Contraseña:" con el placeholder "Ingrese su contraseña".
- Campo de texto "Confirmar contraseña:" con el placeholder "Confirmar contraseña".
- Selección de rol de usuario:
  - Opción "Soy profesor:" con un botón de radio seleccionado.
  - Opción "Soy alumno:" con un botón de radio no seleccionado.
- Botón de acción "Registrarme" en la parte inferior.

*Imagen 04*

### 3. Secciones de la aplicación y navegación

Para navegar por las cuatro diferentes secciones de la aplicación luego del login o registro, deberá hacer click en alguna de las opciones disponibles en la barra superior de navegación. Dependiendo de qué tipo de usuario ingrese, este podrá ver una u otra cosa. Por defecto, lo que se muestra siempre es la primera sección, al hacer click alguna de las otras, estas aparecerán y se ocultará la anterior sin perder la información ingresada.

#### 3.1.– Secciones del Docente

Accediendo con el perfil docente, podremos ver cuatro secciones diferentes en la barra de navegación, como se ve en la imagen 05.

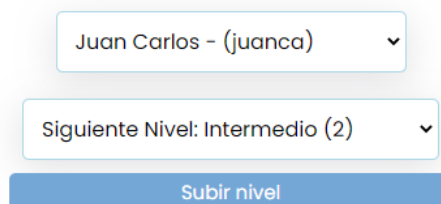


Imagen 5

##### 3.1.1. Cambiar nivel de alumnos

En esta sección el docente podrá elegir a uno de sus alumnos (Imagen 06), y subir al elegido de nivel. Los niveles disponibles son: Inicial, Intermedio y Avanzado. Solo se puede subir de nivel a un alumno de uno en uno, y una vez realizada la acción, no se puede volver al nivel anterior.

**Seleccione a uno de sus alumnos para subir su nivel!**

Un formulario con dos selectores de lista desplegable y un botón. El primer selector muestra "Juan Carlos - (juanca)". El segundo selector muestra "Siguiete Nivel: Intermedio (2)". Debajo de ellos hay un botón azul con el texto "Subir nivel".

Juan Carlos - (juanca) ▼

Siguiete Nivel: Intermedio (2) ▼

Subir nivel

Imagen 6

### 3.1.2. Plantear Ejercicios

El docente podrá plantear ejercicios (Imagen 07) rellenando todos los campos con los requisitos correspondientes.

### Plantear Ejercicios

Seleccione Nivel para Enviar Ejercicio:

Elija Nivel de Alumno:

▼

Título de Ejercicio:

Escriba su Título

Descripción:

Describe su ejercicio aquí:  
Respetar cantidad de caracteres:  
20 (min), 200(max) entre Título y Descripción.

Imagen:

Seleccionar archivo

Ningún archivo seleccionado

Agregar Ejercicio

*Imagen 07*

### 3.1.3. Redactar devoluciones

El docente podrá en esta sección ver, y escuchar las entregas de sus alumnos, así como también redactar devoluciones (Imagen 08).

**Ver entregas recientes y redactar devoluciones a sus alumnos**

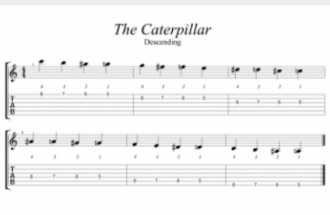

Nombre de alumno	Ejercicio	Imagen	Audio del alumno	Redacte devolución	Enviar entrega
Juan Carlos (Nivel actual: Inicial)	The Caterpillar (Inicial)			<input type="text" value="Escriba devolución"/>	<input type="button" value="Enviar devolución"/>

Imagen 8

### 3.1.4. Ver información estadística

En esta sección el docente podrá ver información estadística acerca de uno o de la totalidad de sus alumnos (Imagen 09).

#### Ver información estadística

El o los alumnos que más resolvieron fueron: **Juan Carlos** (3 tareas) |

El total de tareas entregadas por todos sus alumnos es: **3**

Juan Carlos - (juanca) ▼

El *total* de ejercicios planteados para el nivel de Juan Carlos (Inicial) son: **4**

De todos los ejercicios planteados *ha resuelto* **3** ejercicios.

Imagen 9

## 3.2. – Secciones del Alumno

Accediendo con el perfil alumno, podremos ver cuatro secciones diferentes en la barra de navegación, como se ve en la imagen 10.

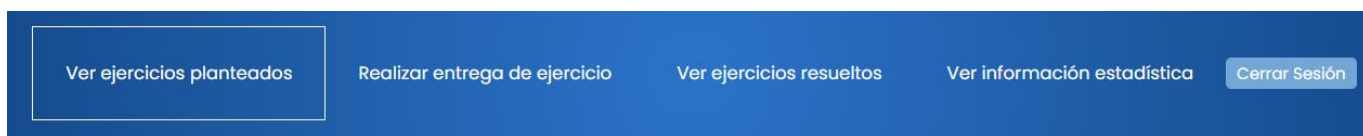


Imagen 10

### 3.2.1. Ver ejercicios planteados

El alumno podrá ver en esta sección los ejercicios que fueron planteados por su docente y para su nivel (Imagen 11). Además contiene un buscador, donde podrá encontrar mediante palabras clave al contenido requerido.

Buscar ejercicios planteados por tu docente **Janis Joplin**, para tu nivel (Inicial):

Buscar


Título del Ejercicio	Descripción	Imagen
Permutaciones (Nivel: Inicial)	Ejercicios secuenciales para trabajar velocidad y ritmo. Repetir 3 veces cada uno con diferentes articulaciones.	

Imagen 11

### 3.2.2. Realizar entrega de ejercicio

En esta sección el alumno podrá realizar una entrega de una grabación de los ejercicios planteados para su nivel por su docente (Imagen 12). No podrá realizar la misma entrega dos veces, ya que una vez entregada, esta no será visible en el desplegable.

#### Realizar entrega de Ejercicios pendientes:

Acordes de Septimas

▼

Adjunte archivo de sonido:

Seleccionar archivo

ej4.m4a

Enviar tarea

Imagen 12

### 3.2.3. Ver ejercicios resueltos

En esta sección el alumno podrá ver todos sus ejercicios entregados sin discriminar por nivel, una especie de historial de todas sus entregas (Imagen 13). Y además podrá leer la devolución de su docente, en caso de que este haya escrito una. Sino se leerá “El docente aun no ha redactado una devolución”.

#### Ver tus ejercicios resueltos


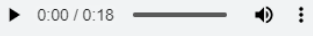
Título del Ejercicio	Descripción	Imagen	Sonido	Devolución del docente
Permutaciones (Nivel: Inicial)	Ejercicios secuenciales para trabajar velocidad y ritmo. Repetir 3 veces cada uno con diferentes articulaciones.			Muy buen trabajo

Imagen 13

#### 3.2.4. Ver información estadística

En esta sección el alumno podrá ver información estadística acerca de sus entregas y sus trabajos plantados para su nivel (Imagen 14).

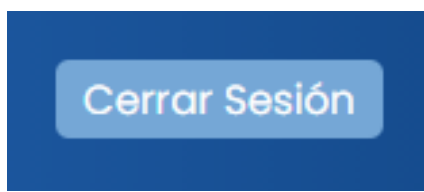
#### Ver información estadística

Porcentaje de ejercicios resueltos para su nivel (Inicial): **75%**  
Total de ejercicios plantados por su docente para su nivel (Inicial): **4**  
De **3** ejercicios resueltos en total, tienen devolucion: **1**

*Imagen 14*

#### 3.3. – Cerrar sesión (log out)

Para cerrar sesión deberá apretar el botón “**Cerrar Sesión**” (Imagen 15). Este lo llevará de vuelta a la página principal donde podrá ingresar con otro usuario o registrar una nueva cuenta.



*Imagen 15*

## 4. Detalles del código de este proyecto

En esta sección se detallará todo lo relacionado al código de este proyecto, así también como bloques importantes para el funcionamiento correcto del programa.

### 4.2.– Archivos del proyecto

Este proyecto cuenta con un archivo HTML (index.html), una carpeta “**js**” que contiene cuatro archivos JavaScript (clases.js, codigo.js, manejoUI.js, precargas.js), una carpeta “**estilo**” que contiene un archivo CSS (estilo.css). Una carpeta “**img**” que contiene tres archivos .jpg (landing1.jpg, landing2.jpg, landing3.jpg) y 2 archivos .svg (iconmonstr-disc-5.svg, iconmonstr-x-mark-7.svg) que forman parte de la página de inicio. Después tenemos la carpeta “**Ejercicios**” que contiene dos carpetas, llamadas “**img**” y “**audio**”. En estas se encuentran 9 archivos de imagen(.jpg) y audio(.m4a) respectivamente que serán utilizadas dentro de la aplicación por los diferentes tipos de usuario para las entregas y los ejercicios planteados por los docentes.

### 4.3.– Precargas, clases y arrays

Este proyecto tiene datos precargados, clases y arrays los cuales son utilizados por el usuario para probar la funcionalidad correcta de todas sus secciones.

#### 4.3.4. Precargas

Estas precargas son de cuatro docentes (con nombre, usuario y contraseña) (Ejemplo en imagen 16), doce alumnos (con nombre, usuario, contraseña, nivel y docente asignado), veinticuatro ejercicios (con nivel, título, id, descripción, imagen y el docente que lo plantea), dieciséis entregas (con id, alumno, ejercicio, docente, sonido y devolución) y tres niveles.

```
// Función para precargar docentes
function preCargaDocentes() {
    crearYGuardarDocente('Eric Clapton', 'eric1945', 'Heaven92');
    crearYGuardarDocente('Bob Dylan', 'bobbydylan', 'bWind1962');
    crearYGuardarDocente('Janis Joplin', 'lajanis', 'Rock1');
    crearYGuardarDocente('Amy Winehouse', 'amyrehab83', 'iSaidno1');
```

Imagen 16



#### 4.3.5. Arrays

Estos datos precargados estarán almacenados en arrays (listas) como objetos (Imagen 17).

```
// Arrays a utilizar
let usuarios = [];
let niveles = [];
let ejercicios = [];
let entregas = [];
```

*Imagen 17*

#### 4.3.6. Clases

Para este proyecto utilizamos cinco clases. Nivel, Docente, Alumno, Ejercicio y Entrega. (Ejemplo en Imagen 18)

```
// Clase para crear Alumno
class Alumno {
  constructor() {
    this.nombre; // string
    this.usuario; // string
    this.password; // string
    this.Nivel; // objeto
    this.Docente; // objeto
    this.tipo = 'alumno';
    this.entregasIndividuales = [];
  }
}

// Clase para crear Ejercicio
class Ejercicio {
  static nro = 1;
  constructor() {
    this.Nivel; // objeto
    this.id = Ejercicio.nro++; // identificador único
    this.tituloEjercicio; // string
    this.descripcionEjercicio; // string
    this.imagen; // imagen
    this.Docente; // objeto
  }
}
```

*Imagen 18*

#### 4.4.– Validaciones

Necesitábamos validar cierta información siguiendo los lineamientos propuestos. Para ello creamos distintas funciones de validación. Las cuales se encargan, por mencionar algunas de ellas, a la limitación de caracteres en algunos campos, requerimientos específicos para contraseñas, la compleción todos los campos, adjuntar el tipo de archivo correcto, entre otras. Algunas de estas funciones de validación se encargarán de mostrar un mensaje al usuario cuando este introduce una información incorrecta (ver Imagen 19).



The image shows a mobile app registration screen titled "Registrarme!". It contains four input fields: "Nombre:" (filled with "Gonza Gentile" and highlighted with a green border), "Usuario:" (placeholder "Ingrese su usuario", red border, with error text "Mínimo 4 caracteres. Sin espacios entremedio."), "Contraseña:" (placeholder "Ingrese su contraseña", red border, with error text "Mínimo 4 caracteres, al menos una mayúscula, una minúscula y un número. Sin espacios entremedio."), and "Confirmar contraseña:" (placeholder "Confirmar contraseña", red border, with error text "Las contraseñas deben coincidir!"). Below the fields are two radio buttons for "Soy profesor:" (selected) and "Soy alumno:". At the bottom is a blue "Registrarme" button.

Imagen 19

Acá vemos un ejemplo de una función de validación que se encarga de comparar que las dos contraseñas ingresadas en el registro sean iguales (Imagen 20).

```
// Esta función compara las contraseñas, y ambas son iguales, devuelve true
function compararPasswordsReg(input1, input2) {
  let validado = false;
  if (
    input1.value.trim() === input2.value.trim() &&
    input2.value.trim().length >= 4
  ) {
    mostrarExitoRegistro(input2);
    validado = true;
  } else {
    mostrarErrorRegistro(input2, 'Las contraseñas deben coincidir!');
  }
  return validado;
}
```

Imagen 20

#### 4.5.– Login y *cuentaActiva*

Los usuarios registrados podrán acceder a la aplicación mediante el login de la página. A través de usuario y contraseña. El usuario puede ser escrito tanto en mayúscula como en minúscula, la contraseña si deberá coincidir en todos sus caracteres por igual (case insensitive). Para ello creamos la siguiente función (Imagen 21):

En esta función se define la variable global *cuentaActiva*. Dependiendo de cual usuario haya ingresado a la aplicación, se mostrarán las secciones del docente (si ingresa un usuario tipo docente) o las secciones del alumno (si ingresa un usuario tipo alumno).

```
function login() {
  let i = 0;
  // Buscamos la cuenta ingresada, si la encontramos pasamos a la contraseña
  while (cuentaActiva === null && i < usuarios.length) {
    // usuarios[i] es cada objeto del array usuarios, el elemento actual en cada iteración
    // Se dejará de buscar una vez que encontramos el objeto
    if (
      usuarios[i].usuario.toLowerCase().trim() ===
      inputUsuarioLogin.value.toLowerCase().trim()
    ) {
      cuentaActiva = usuarios[i];
    }
    i++;
  }
  // Chequeamos por contraseña
  if (
    cuentaActiva !== null &&
    cuentaActiva.password === inputPasswordLogin.value
  ) {
    // Hacemos desaparecer el span de error de Login
    document.querySelector('#mensajeLogin').innerHTML = '';

    // Mostramos el UI segun el usuario
    mostrarUISegunUsuario(cuentaActiva);
  } else {
    // Hacemos aparecer el span del error Login, poniendole un mensaje
    document.querySelector('#mensajeLogin').innerHTML =
      'Usuario o contraseña incorrectos! Ingreselos nuevamente';
  }
}
```

Imagen 21

#### 4.6.– Logout y *cuentaActiva*

El Logout es un botón ubicado en la parte superior derecha en la barra de navegación. Este se encarga de cerrar la sesión de la *cuentaActiva*. Para ello, la redefine a *null* y oculta las ventanas de la aplicación, mostrando nuevamente la página de inicio. Se vacían los inputs del login, y también los del registro (Imagen 22).

```
function logOut() {
  // Reseteamos la cuentaActiva
  cuentaActiva = null;

  // Mostramos la pagina de inicio nuevamente, y ocultamos la de la aplicación
  paginaInicio.classList.remove('ocultar-ventana');
  paginaInicio.classList.add('mostrar-ventana');
  paginaApp.classList.add('ocultar-ventana');

  // Limpiamos inputs
  limpiarUsuarioInput();
  limpiarPasswordInput();
  limpiarInputsRegistro();
}
```

Imagen 22

## 4.7.–Ocultar y mostrar elementos

Para ocultar y mostrar la mayoría de los elementos recurrimos al uso de clases en CSS (Imagen 23). Para el caso de las secciones, para que apareciera solo una al ser elegida desde la barra de navegación, seguimos el tutorial de aulas en donde hace uso directo de la propiedad `.style.display` sobre el elemento (Imagen 24).

```
.mostrar-ventana {
  display: block;
}

.ocultar-ventana {
  display: none;
}
```

Imagen 23

```
let navLinks = document.querySelectorAll('.item-lista');

for (let i = 0; i < navLinks.length; i++) {
  navLinks[i].addEventListener('click', mostrarSeccionIndividual);
}

document.querySelector('#btn-seccion1').click();
}

// Ocultar y mostrar secciones
function mostrarSeccionIndividual() {
  ocultarSecciones();
  let idBoton = this.getAttribute('id');
  let idSeccion = '#' + idBoton.substring(4);

  document.querySelector(idSeccion).style.display = 'block';
}

function ocultarSecciones() {
  let secciones = document.querySelectorAll('.seccion');
  for (let i = 0; i < secciones.length; i++) {
    secciones[i].style.display = 'none';
  }
}
```

Imagen 24

## 4.8.—Manejo UI (Interfaz de usuario)

Para la creación de las secciones recurrimos a funciones que se encargan de mostrar y ocultar las diferentes opciones que tiene cada usuario en la barra de navegación. En el siguiente ejemplo (Imagen 25) se ve la función *mostrarUISegunUsuario* que muestra la navegación y carga las secciones según la *cuentaActiva*. También agrega un mensaje de bienvenida, basado en el nombre de dicha cuenta.

```
function mostrarUISegunUsuario(cuentaActiva) {  
  // Sacamos de vista la página principal de registro y login, agregándole la  
  // clase ocultar-ventana  
  paginaInicio.classList.add('ocultar-ventana');  
  
  mostrarNav(cuentaActiva);  
  cargarSecciones(cuentaActiva);  
  
  // Hacemos aparecer la ventana de la aplicación con los datos y secciones  
  // del docente o alumno, según corresponda  
  paginaApp.classList.remove('ocultar-ventana');  
  
  // Mensaje de bienvenida  
  nombreAMostrar.innerHTML = `Hola de nuevo, ${cuentaActiva.nombre}. <br> <p  
  class="subTituloNombre">Seleccione un botón de la barra superior para ver la  
  sección correspondiente.</p>`;  
}
```

Imagen 25

En la siguiente imagen (Imagen 26) vemos un ejemplo de la creación de una sección. Se crean los elementos, se recorren arrays cumpliendo ciertas condiciones, y se muestra la información personalizada luego en la sección correspondiente con el *innerHTML*.

```
function seccion1Docente(usuario) {
  let titulo = `

## 


```

Imagen 26

## 5. Lista de funciones

En esta lista se detallan todas las funciones del programa. Se encuentran en **negrita** las que consideramos más importantes a criterio personal.

Funciones	Funcionalidad y comentarios
preCargaNiveles()	Esta función llama a crearYGuardarNivel(). Se invoca en misEventos() para precargar los datos de niveles.
preCargaDocentes()	Esta función llama a crearYGuardarDocente(). Se invoca en misEventos() para precargar los datos de los usuarios docentes.
preCargaAlumnos()	Esta función llama a crearYGuardarAlumno(). Se invoca en misEventos() para precargar los datos de los usuarios alumnos.
preCargaEjercicios()	Esta función llama a crearYGuardarEjercicio(). Se invoca en misEventos() para precargar las tareas propuestas por los docentes.
preCargaEntrega()	Esta función llama a crearYGuardarEntrega(). Se invoca en misEventos() para precargar las entregas de tareas realizadas por los alumnos.
crearYGuardarNivel()	Esta función toma como parámetro un número y un nivel (string), y si son validados crea un objeto Nivel, el cual se agrega al array niveles.
crearYGuardarDocente()	Esta función toma como parámetro un nombre, un usuario y un password y si son validados los datos se crea un objeto Docente, el cual se agrega al array usuarios.
crearYGuardarAlumno()	Esta función toma como parámetro un nombre, un usuario, un password, un nivel y un docente y si son validados los datos se crea un objeto Alumno, el cual se agrega al array usuarios.
crearYGuardarEjercicio()	Esta función toma como parámetro un nivel, un título, una descripción, un nombre de archivo(imagen) y un docente y si son validados los datos se crea un objeto Ejercicio, el cual se agrega al array ejercicios.
crearYGuardarEntrega()	Esta función toma como parámetro un nivel, un alumno, un ID de ejercicio, un docente un nombre de archivo (audio) y devolución y si son validados los datos se crea un objeto Entrega, el cual se agrega al



	array entregas y al array entregasIndividuales dentro del objeto Alumno que entró como parametro.
<b>obtenerDocenteConUsuario()</b>	Esta función toma un nombre de usuario y devuelve el objeto de Docente con ese Usuario único (en caso de existir) Nos aseguramos que sea de ese tipo poniéndole el .tipo = 'docente' en el if statement
<b>obtenerAlumnoConUsuario()</b>	Esta función toma un nombre de usuario y devuelve el objeto de Alumno con ese Usuario único (en caso de existir) Nos aseguramos que sea de ese tipo poniéndole el .tipo = 'alumno' en el if statement
<b>obtenerEjercicioConID()</b>	Esta función toma un id y devuelve el objeto de Ejercicio de acuerdo con ese id único, en caso de existir.
<b>obtenerNivelConNumero()</b>	Esta función toma un número y devuelve el objeto de Nivel de acuerdo con ese número, en caso de existir.
<b>obtenerEntregaConID()</b>	Esta función toma un id y devuelve el objeto de Entrega de acuerdo con ese id en caso de existir.
<b>registro()</b>	En esta función capturamos los elementos html de los inputs y los mandamos a funciones de validación. En caso de ser validados todos los campos, según valor del radio button, se crea o un objeto Alumno, o un objeto Docente.
<b>finRegistroExitoso()</b>	Esta función muestra un mensaje de éxito en nuestro formulario de registro. Se define la cuentaActiva como la última del array de usuarios. Se ejecuta la función mostrarUISegunUsuario con un delay de 2.5 segundos. Esto es para que se vea el mensaje de éxito, y no saltemos inmediatamente a la página de las secciones del usuario.
<b>valorRadioBtnRegistro()</b>	Esta función nos devuelve el valor del radio button, el que este "checked".
<b>login()</b>	En esta función capturamos los valores del html que nos llegan desde el sector de login. Primero comparamos el usuario, y si existe comparamos la contraseña. Aquí se define cuentaActiva, la cual será clave para mostrar las secciones personalizadas según cada usuario y su tipo, llamando a mostrarUISegunUsuario()
<b>logout()</b>	En esta función dejamos la cuentaActiva como null, y mostramos la página de inicio nuevamente, y ocultamos la de la aplicación. Limpiamos los inputs como si se fuera a entrar por primera vez, dos de

	login, y registro (por si se había registrado previamente).
<b>validarNombreReg()</b>	Esta función es para validar el campo “Nombre” del formulario de registro. No debe contener números, ni símbolos, y ser mayor o igual a cuatro caracteres. En caso de que cumpla las validaciones, se llamará a la función mostrarExitoRegistro() y se devuelve true. Sino, se llama a la función mostrarErrorRegistro() y se devuelve false. Esta función como parámetro toma el elemento html del input de nombre.
<b>validarUsuarioReg()</b>	Esta función es para validar el campo “Usuario” del formulario de registro. Debe ser mayor o igual a cuatro caracteres, y no tener espacio entre medio y no ser solo un número. En caso de que cumpla las validaciones, se llamará a la función mostrarExitoRegistro() y se devuelve true. Sino, se llama a la función mostrarErrorRegistro() y se devuelve false. Esta función como parámetro toma el elemento html del input de usuario.
<b>validarPasswordReg()</b>	Esta función es para validar el campo de Contraseña, debe ser mayor o igual a cuatro caracteres, contener una minúscula, una mayúscula y un número. En caso de que cumpla las validaciones, se llamará a la función mostrarExitoRegistro() y se devuelve true. Sino, se llama a la función mostrarErrorRegistro() que devuelve false. Esta función como parámetro toma el elemento html del contraseña.
<b>compararPasswordsReg()</b>	Esta función toma como parámetro dos elementos y compara sus valores (.value), si son iguales devuelve true junto con la función de mostrarExitoRegistro() Sino, se llama a la función mostrarErrorRegistro() y se devuelve false.

<b>mostrarErrorRegistro()</b>	<p>Esta función se encarga de avisar al usuario que tuvo algún error en alguno de los campos de registro. Toma como parámetro el elemento html del input y un mensaje (string). Encuentra al elemento padre del input (el div con clase <code>.ítem-formulario</code> en todos los casos) y le agrega la clase <code>.error</code>. Esto lo que hace es pintar el borde del input de donde vino el error de color rojo. Además se encuentra al span del elemento padre, y se le agrega el mensaje personalizado de error. También le saca la clase <code>.exito</code> si la tenía previamente.</p>
<b>mostrarExitoRegistro()</b>	<p>Esta función se encarga de avisar al usuario que su ingreso en el input de registro fue exitoso. Toma como parámetro el elemento html del input. Encuentra al elemento padre del input (el div con clase <code>.ítem-formulario</code> en todos los casos) y le agrega la clase <code>.exito</code>. Esto lo que hace es pintar el borde del input de color verde. Además se encuentra al span del elemento padre y le borra cualquier mensaje de error que hubiese. También le saca la clase <code>.error</code> si la tenía previamente.</p>
<b>validarImagenArchivo()</b>	<p>Esta función toma como parámetro un string de un nombre de archivo. Captura la extensión en una variable y la compara con las que necesita que valide. Si es jpg o png, devuelve true, sino devuelve false.</p>
<b>validarSonidoArchivo()</b>	<p>Esta función toma como parámetro un string de un nombre de archivo. Captura la extensión en una variable y la compara con las que necesita que valide. Si es mp3, m4a, wav o flac, devuelve true, sino devuelve false.</p>
<b>validarPassword()</b>	<p>Esta función la usamos para validar los ingresos de las precargas inicialmente, solo toma como parámetro</p>

	un string y si cumple con todas las condiciones (Igual o más de 4 caracteres al menos una mayúscula, una minúscula, y un número, y no tiene espacios entre medio) se devuelve true, sino se devuelve false.
<b>validarUsuario()</b>	Esta función la usamos para validar los ingresos de las precargas inicialmente, solo toma como parámetro un string y si cumple con todas las condiciones (Igual o más de 4 caracteres, sin espacios entremedio) devuelve true, sino se devuelve false.
<b>validarNombre()</b>	Esta función la usamos para validar los ingresos de las precargas inicialmente, solo toma como parámetro un string y si cumple con todas las condiciones (Igual o más de 4 caracteres, sin números ni símbolos) devuelve true, sino se devuelve false.
<b>tieneEspacioEntreMedio()</b>	Esta función toma como parámetro un string, y si tiene algún espacio entre medio, devuelve true, sino devuelve false.
<b>validarUsuarioExistente()</b>	Esta función toma como parámetro un nombre de usuario y lo compara con todos los usuarios que están el array usuarios. En caso de existir, devuelve true, si no existe aún, devuelve false.
<b>quitarFakePath()</b>	Esta función toma como parámetro un nombre de archivo por ejemplo capturado de un input tipo file, y devuelve una versión más corta, sin el "path" del archivo.
<b>misEventos()</b>	Esta función llama a todas las precargas (preCargaNiveles(), preCargaDocentes, preCargaAlumnos, preCargaEjercicios(), preCargaEntregas()).
<b>mostrarUISegunUsuario()</b>	Esta función oculta la pagina de inicio, y llama a las funciones mostrarNav() y cargarSecciones(), además muestra un mensaje de bienvenida según el usuario que ingresó.
<b>mostrarNav()</b>	Esta función crea la variable nav, y según el tipo de usuario que ingreso, crea el elemento HTML con las secciones correspondientes y lo pone en la sección con id #navegacion. Además le agrega un evento a cada uno de sus links, y con la función mostrarSeccionIndividual() solo mostrar la sección que esta seleccionada y ocultamos el resto.
<b>mostrarSeccionIndividual()</b>	Esta función primero llama a ocultarSecciones() que oculta todas las secciones de la página, luego le añade la propiedad style.display = "block" a la sección que fue elegida, para que sea visible solamente esta última.

<b>ocultarSecciones()</b>	Esta función oculta todas las secciones agregándole la propiedad <code>style.display = "none"</code> mediante un loop por que itera por todas las secciones que tienen la clase <code>.seccion</code> .
<b>cargarSecciones()</b>	Esta función toma como parámetro el usuario recién ingresado, y según su tipo, carga todas las secciones.
<b>seccion1Docente()</b>	Esta función crea los elementos HTML de la sección (un título y un select), donde el docente podrá subir de nivel a alguno de sus alumnos. Para ello, itera por el array usuarios y muestra un desplegable de todos los alumnos asignados a ese docente. Si se selecciona alguno, se llama a la función <code>mostrarSelectCambiarNivel()</code> .
<b>mostrarSelectCambiarNivel()</b>	Esta función captura el alumno seleccionado del select de la sección 1 del docente utilizando la función <code>obtenerAlumnoConUsuario()</code> , crea el select con los niveles disponibles, basados en los niveles del alumno seleccionado (utilizando if statements), también el botón de cambiar nivel y un div para dar un mensaje.
<b>btnCambiarNivel()</b>	Esta función captura el alumno seleccionado, así como también el nivel seleccionado, al ser presionado, se efectúa el cambio de nivel. Se le informa al usuario que fue exitoso, o si hubo algún problema (si ya no tenía más niveles para subir). Llama a la sección 3 y 4 para actualizar la información instantáneamente sobre el cambio de nivel.
<b>seccion2Docente()</b>	Esta función crea todos los elementos HTML para crear la sección para plantear ejercicios. Un select para elegir un nivel, campos para título y descripción, y un input para adjuntar un archivo tipo imagen. Luego un botón para enviarlo. Añadimos un evento al botón de agregar ejercicio que llama a <code>btnAgregarEjercicio()</code> .
<b>btnAgregarEjercicio()</b>	Esta función captura todos los valores de la sección 2 docente, y si son válidos los datos, se crea el objeto <code>Ejercicio</code> que será agregado al array <code>ejercicios</code> . Luego vaciamos todos los inputs de la sección. En caso de haber algún error en los datos, se mostrarán mensajes correspondientes.
<b>seccion3Docente()</b>	Esta función crea los elementos necesarios para ver las entregas recientes de los alumnos asignados a ese docente y dejar una devolución. Para ello crea una tabla que las contiene. Primero itera por el array usuarios buscando a los tipo alumno, después filtra a

	que estos sean sus alumnos asignados y busca sus entregas sin devolución. Cada fila de la tabla contiene la propiedad data-id con el id único de cada entrega. Luego le añade un evento a todos los botones de añadir devolución.
<b>enviarDevolucion()</b>	Esta función está en el botón de “Enviar devolución”. Primero captura cual de los botones fue apretado y encuentra a la fila. Así capturamos el input de esa fila y es lo que se le añade al objeto de la entrega como devolución. Luego se actualiza la sección 3 para mostrar nuevamente la tabla, esta vez sin esa tarea, ya que ahora tiene devolución. Se tiene que escribir algo si o si como devolución, no puede quedar vacío.
<b>mensajeSiTablaVacia()</b>	Esta función cuenta la cantidad de filas que tiene la tabla de la sección 3 del docente, en caso de tener solo 1, significaría que no tiene contenido, entonces mostramos un mensaje de que no hay tareas pendientes, en vez que mostrar una tabla solo con sus <th></th> (títulos de la tabla)
<b>seccion4Docente()</b>	En esta sección se crean los elementos necesarios para mostrar las estadísticas requeridas. Se itera por el array de usuarios, se filtra los alumnos que corresponden con los del profesor, y se buscan los datos que son pedidos en la letra. Como la mayor cantidad de ejercicios resueltos, y el alumno que más resolvió ejercicios. Además se crea un select con los alumnos de ese docente, para poder ver información estadística específica de ese alumno.
<b>mostrarEstadisticasIndividuales()</b>	Esta función captura el valor del select de la sección 4, y así obtiene al objeto del alumno seleccionado, con la función obtenerAlumnoConUsuario(), ya con el objeto del alumno obtenido, accedemos a sus datos y los mostramos en un div.
<b>seccion1Alumno()</b>	En esta sección se crean los elementos necesarios para la sección 1 del alumno, la cual incluye un input para el buscador y una tabla con todos los ejercicios planteados por su docente para su nivel. Primero capturamos el objeto Docente que le corresponde a ese alumno, y luego iteramos con el array de ejercicios en busca de los que son de ese docente, y también los que tienen el mismo nivel del alumno.
<b>btnBuscarEjercicio()</b>	Esta función se dispara al apretar “Buscar”. Captura el input del buscador y filtrando con if statements los ejercicios del docente y del nivel del alumno, busca primero en el título y luego en la descripción de los

	ejercicios. Para ello utilizamos indexOf(), si es distinto a -1 es que hay una coincidencia. Si hay coincidencia, llenamos el div de #resultadoBusqueda con los elementos. Acá decidimos no mostrar como una tabla, sino la imagen grande y el título/descripción.
chequearTablaEjerciciosVacia()	Esta función cuenta la cantidad de filas que tiene la tabla de la sección 1 del alumno, en caso de tener solo 1, significaría que no tiene contenido, entonces mostramos un mensaje de que no hay tareas planteadas para el nivel del alumno, en vez que mostrar una tabla solo con sus <th></th> (títulos de la tabla)
seccion2Alumno()	En esta sección se crean los elementos HTML necesarios para la sección 2 del alumno, la cual incluye un select de todos los ejercicios planteados por su docente para su nivel, que no se encuentren en el array de entregas, y no coincidan con el usuario del alumno. Para eso usamos la función checkObjetoRepetidoEntregas(). También hay un input donde cargar un audio. Ambos campos tienen que ser validados. Luego del envío exitoso de la tarea, se actualiza la sección para que se actualice el select de las tareas pendientes.
btnEntregarTarea()	Esta función se llama al enviar una tarea, esta captura el valor del select, y así obtiene el objeto Ejercicio que se va a entregar, luego captura el archivo de sonido, y con la función quitarFakePath le limpiamos el path. Si todo se valida llamamos a la función crearYGuardarEntrega() que agrega la entrega al array de tareas, y al array de entregas individuales del alumno.
seccion3Alumno()	En esta función se crean los elementos HTML necesarios para la sección 3 del alumno. La cual es una tabla con todas las tareas entregadas, sin discriminar por nivel que hizo el alumno. Para ello iteramos por el array individual del alumno entregasIndividuales y simplemente creamos la tabla.
chequearTablaEntregasVacia()	Esta función cuenta la cantidad de filas que tiene la tabla de la sección 3 del alumno, en caso de tener solo 1, significaría que no tiene contenido, entonces mostramos un mensaje de que no hay tareas entregadas aun, en vez que mostrar una tabla solo con sus <th></th> (títulos de la tabla)
seccion4Alumno()	En esta sección se crean los elementos necesarios para mostrar las estadísticas requeridas. Se itera por

	el array de las entregasIndividuales, y también por el de ejercicios, y se incrementan contadores para contabilizar lo requerido por la letra.
abrirVentanaRegistro()	Muestra la ventana del registro (.contenedor-ventana-registro) al agregarle la clase .mostrar-ventana.
cerrarVentanaRegistro()	Oculto la ventana del registro registro (.contenedor-ventana-registro) al quitarle la clase .mostrar-ventana.
radioAlumnoChecked()	Crea un select con los docentes disponibles en el formulario de registro, se llama cuando la opción de alumno esta seleccionada.
radioDocenteChecked()	Oculto el select con los docentes disponibles en el formulario de registro, se llama cuando la opción de docente esta seleccionada.
limpiarUsuarioInput()	Vacía el valor del input usuario del sector de login, también el mensaje de error, si hubiese alguno.
limpiarPasswordInput()	Vacía el valor del input password del sector de login, también el mensaje de error, si hubiese alguno.
limpiarInptusRegistro()	Limpia todos los valores del formulario de registro, también le quita la clase .exito así le quitamos el borde verde del anterior registro exitoso.



## 6. Informe de Testing

Que estoy probando	Resultado Esperado	¿Con que prueba?	¿Pasa la prueba?
Inicio: Botón Registrarme	Que se abra el Formulario Registro	Haciendo Click en el botón “Registrarme!”	SI
Inicio: Formulario Registro, Nombre	Probar validación de Nombre ingresado	Completar el campo Nombre con 3 caracteres	NO
Inicio: Formulario Registro, Nombre	Probar validación de Nombre ingresado	Completar el campo Nombre con 4 o más letras en minúsculas y/o mayúsculas	SI
Inicio: Formulario Registro, Nombre	Probar validación de Nombre ingresado	Completar el campo Nombre con 3 letras y 1 numero (o símbolo)	NO
Inicio Formulario Registro, Usuario	Ver aviso de error al registrar un usuario ya registrado anteriormente	Escribir el campo Usuario con un usuario ya registrado anteriormente.	SI
Inicio: Formulario Registro, Usuario	Probar validación de Usuario ingresado	Escribir el campo Usuario con solo 4 o más números	NO
Inicio: Formulario Registro, Usuario	Probar validación de Usuario ingresado	Escribir el campo Usuario con solo 4 o más símbolos	SI
Inicio: Formulario Registro, Usuario	Probar validación de Usuario ingresado	Escribir el campo Usuario con solo 1 letra y 3 o más números	SI
Inicio: Formulario Registro, Usuario	Probar validación de Usuario ingresado	Escribir el campo Usuario con solo 1 número y 3 o más símbolos	SI
Inicio: Formulario Registro, Usuario	Probar validación de Usuario ingresado	Escribir el campo Usuario con 2 palabras con espacio entre medio	NO
Inicio: Formulario Registro, Contraseña	Probar validación de Contraseña ingresada	Escribir en el campo Contraseña: 3 letras	NO
Inicio: Formulario Registro, Contraseña	Probar validación de Contraseña ingresada	Escribir en el campo Contraseña: 1 may., 1 min., 1 núm. y cualquier otro carácter	SI
Inicio: Formulario Registro, Contraseña	Probar validación de Contraseña ingresada	Escribir en el campo Contraseña: 4 números	NO

Inicio: Formulario Registro, Contraseña	Probar validación de Contraseña ingresada	Escribir en el campo Contraseña con 1 may, 1 min., 1 núm., 1 espacio y cualquier otro carácter	NO
Inicio: Formulario Registro, Confirmar Contraseña	Probar validación de Confirmar Contraseña	Escribir en el campo Confirmar Contraseña, una contraseña diferente a la del campo anterior	NO
Inicio: Formulario Registro, Confirmar Contraseña	Probar validación de Confirmar Contraseña	Escribir en el campo Confirmar Contraseña, una contraseña igual a la del campo anterior	SI
Inicio: Formulario Registro, Radio Button	Probar realizar el registro para Docente	Elijo “Soy profesor” en el registro y completo todos los demás campos correctamente.	SI
Inicio: Formulario Registro, Radio Button	Probar realizar el registro para Alumno	Elijo “Soy alumno” en el registro y completo todos los demás campos correctamente.	SI
Inicio: Formulario Registro, Alumno, Select Docente	Probar seleccionar docente para el Alumno y asignación correcta de profesor	Elijo un docente de la lista y completo todos demás campos correctamente. Al registrarme veo si mi docente fue asignado correctamente viendo el mensaje de la sección 1 de búsqueda de ejercicios.	SI
Inicio: LogIn o Inicio de Sesión	Probar ingreso de usuario	Usar un usuario y contraseña que no existen	NO
Inicio: LogIn o Inicio de Sesión	Probar ingreso de usuario	Usar un usuario que existe, con contraseña incorrecta	NO
Inicio: LogIn o Inicio de Sesión	Probar ingreso de usuario	Usar un usuario que existe, con contraseña correcta	SI
Alumno: Buscar ejercicio	Que encuentre un resultado en su búsqueda	Uso una palabra que no existe en título ni en descripción	NO
Alumno: Buscar ejercicio	Que encuentre un resultado en su búsqueda	Uso una palabra que existe solo en título	SI
Alumno: Buscar ejercicio	Que encuentre un resultado en su búsqueda	Uso una palabra que existe solo en descripción	SI

Alumno: Buscar ejercicio	Que encuentre varios resultados en su búsqueda	Uso una palabra que existan en 2 o más ejercicios	SI
Alumno: Enviar Tarea	Enviar la tarea del alumno a su docente	Elegir tarea y enviar sin adjuntar archivo	NO
Alumno: Enviar Tarea	Enviar la tarea del alumno a su docente	Elegir tarea del select y adjuntar archivo de otro formato o extensión	NO
Alumno: Enviar Tarea	Enviar la tarea del alumno a su docente	No elegir tarea del select y adjuntar archivo en el formato correcto	NO
Alumno: Enviar Tarea	Enviar la tarea del alumno a su docente	Elegir tarea del select y adjuntar archivo en el formato correcto	SI
Alumno: Escuchar Audio entregado	Que se escuche el audio correcto en el ejercicio designado	Click en botón play de reproductor de Audio	SI
Alumno: Ver información estadística	Ver la información y estadística correcta propia del usuario logueado.	No realizo ninguna acción. Esta información aparece directamente en pantalla	SI
Alumno: Botones de Navegación	Que los botones me dirijan a la sección correspondiente	Hago click en cada sección de navegación y veo si es la correcta	SI
Docente: Botones de Navegación	Que los botones me dirijan a la sección correspondiente	Hago click en cada sección de navegación y veo si es la correcta	SI
Navegación: Botón Cerrar Sesión	Que cierre la sesión del usuario y que me lleve a la página principal	Hago click en el botón cerrar sesión	SI
Docente: Select para subir de nivel a un Alumno	Ver el nivel correcto de los Alumnos	Click en select para ver el nivel	SI
Docente: Select para subir de nivel a un Alumno	Subir de nivel a un Alumnos	Selecciono un nivel superior al que tiene el alumno	SI
Docente: Select para subir de nivel a un Alumno	Bajar de nivel a un Alumnos	Ver si se muestra un nivel inferior al que tiene el alumno	NO
Docente: Plantear Ejercicios a los alumnos	Ver los niveles disponibles para mandar tareas en el select	Haciendo click en el select.	SI
Docente: Plantear Ejercicios a los alumnos	Plantear ejercicios a los alumnos	Completo todos los campos menos 1 y planteo ejercicios	NO

Docente: Plantear Ejercicios a los alumnos	Plantear ejercicios a los alumnos	Elijo nivel y lleno todos los campos disponibles, pero entre titulo y descripción uso menos de 20 o mas de 200 caracteres	NO
Docente: Plantear Ejercicios a los alumnos	Plantear ejercicios a los alumnos	Elijo nivel y lleno todos los campos disponibles, pero coloco un archivo que no es .jpg o .png	NO
Docente: Plantear Ejercicios a los alumnos	Plantear ejercicios a los alumnos	Elijo nivel y lleno todos los campos disponibles correctamente	SI
Docente: Escuchar tareas de alumnos en sección Redactar Devoluciones	Verificar precargas del audio enviado por el alumno corresponde a la tarea planteada	Click en play del reproductor de audio	SI
Docente: Escribir devolución de tareas de alumnos	Verificar que devolución entregada se envíe e ingrese en el array correcto	Escribo devolución en el input correspondiente y clickeo botón enviar devolución	SI
Docente: Select de alumnos de Ver información estadística	Ver la información estadística correcta del alumno correspondiente	Hago click en el select y selecciono un alumno	SI
Docente: Ver información estadística de alumnos	Ver la información estadística del alumno que más ejercicios resolvió	No realizo ninguna acción. Esta información aparece directamente en pantalla	SI

## **7. Documento de Análisis**

### **7.2.Descripción del problema a resolver**

Crear una aplicación web para gestionar a distancia el aprendizaje y enseñanza de música, para así facilitar el uso tanto de varios docentes como alumnos en diferentes niveles.

#### **7.1.2. Tipos de usuario del sistema**

Tipos de usuario:

- 1- Docentes
- 2- Alumnos (A – Inicial, B- Intermedio, C-Avanzado)

#### **7.1.3. Listado de funcionalidades**

F01 – Registro en el sitio – Usuario/s: Docente y Alumno

F02 – Ingreso (Log in) a la aplicación – Usuario/s: Docente y Alumno

F03 – Asignar a sus alumnos un nivel (A – Inicial, B- Intermedio, C-Avanzado) –  
Usuario: Docente

F04 – Asignar ejercicios para sus alumnos de un determinado nivel (A – Inicial, B-  
Intermedio, C-Avanzado) – Usuario: Docente

F05 – Poder dejar comentarios de devolución de ejercicios entregados por los alumnos –  
Usuario: Docente

F06 – Visualizar información estadística (general) acerca de alumnos de un docente según  
ejercicios resueltos – Usuario: Docente

F07 – Poder ver y buscar los ejercicios que su docente plantea según el nivel – Usuario:  
Alumno

F08 – Realizar la entrega de un ejercicio, adjuntando un audio correspondiente a dicha  
tarea – Usuario: Alumno

F09 – Visualizar los ejercicios resueltos –Usuario: Alumno

F10 – Visualizar información estadística (individual) acerca de ejercicios resueltos –  
Usuario: Alumno

F11 – Cerrar la sesión (Log out) para volver a la pantalla de inicio donde este la opción  
de Iniciar sesión nuevamente – Usuario: Docente y alumno

## 7.2. Detalle de Funcionalidades

A continuación, se presenta el detalle de cada una de las funcionalidades a resolver en el obligatorio.

### 7.2.2. F01 – Registro en el sitio

### 7.2.3. Acceso

Tipos de usuario: Docente y Alumno.

### 7.2.4. Descripción

Para poder registrarse al sitio ambos tipos de usuario deberán ingresar un nombre, nombre de usuario y contraseña. Si el tipo de usuario es alumno, deberá seleccionar un docente. Habrá opciones para cada formulario, ‘Registrarse como alumno’, y ‘Registrarse como docente’.

### 7.2.5. Interfaz de usuario

The image displays two mobile-style registration forms side-by-side. The left form is titled 'Registrarme como Alumno' and includes fields for 'Nombre de usuario', 'Contraseña', and 'Confirmar contraseña'. It also features a dropdown menu labeled 'Seleccione un profesor' with a list of names: 'Elija un profesor disponible', 'Paul McCartney', 'Bob Dylan', and 'John Lennon'. The right form is titled 'Registrarme como Docente' and includes fields for 'Nombre de usuario', 'Contraseña', and 'Confirmar contraseña'. Both forms have a blue 'Registrarme!' button at the bottom.

### Registrarme como Alumno

Nombre de usuario

Contraseña

Confirmar contraseña

Seleccione un profesor

Elija un profesor disponible

Elija un profesor disponible

Paul McCartney

Bob Dylan

John Lennon

Registrarme!

### Registrarme como Docente

Nombre de usuario

Contraseña

Confirmar contraseña

Registrarme!

### 7.2.6. Validaciones

Todos los campos son obligatorios (nombre, nombre de usuario y contraseña, y confirmación de contraseña).

No pueden existir dos nombres de usuarios iguales.

La contraseña deberá tener como mínimo 4 caracteres, contando con al menos una mayúscula, una minúscula y un número.

Para el perfil alumno: Validar que haya seleccionado un docente.

**Extra:** Validar ingreso de confirmación de contraseña.



## 7.3.F02 – Ingreso (Log in) a la aplicación

### 7.3.2. Acceso

Tipos de usuario: Docente y Alumno

### 7.3.3. Descripción

Se ingresará nombre de usuario y contraseña.

### 7.3.4. Interfaz de usuario



The image shows a login form titled "Acceder". It contains two input fields: "Nombre de usuario" and "Contraseña". Below the fields is a blue button labeled "Acceder". The form is centered on a light gray background.

### 7.3.5. Validaciones

Todos los campos son obligatorios (nombre de usuario y contraseña).

Validar que usuario este registrado previamente.

Validar nombre de usuario en minúsculas.

Validar contraseña en case sensitive.

Validar que nombre de usuario y contraseña se correspondan.

## 7.4.F03 – Asignar a sus alumnos un nivel

### 7.4.2. Acceso

Tipos de usuario: Docente

### 7.4.3. Descripción

Cada docente podrá cambiar el nivel de cada uno de sus alumnos (A – Inicial, B- Intermedio, C-Avanzado) en cualquier momento, pero solamente desde inicial a intermedio y de intermedio a avanzado, nunca al revés.

### 7.4.4. Interfaz de usuario



The screenshot shows a web interface titled "Cambiar nivel de alumno". It features two dropdown menus. The first dropdown menu displays "Juan pedro" with a downward arrow. The second dropdown menu displays "Avanzado" with a downward arrow and a list of options: "Inicial", "Intermedio (Actual)", and "Avanzado". The "Avanzado" option is highlighted in blue. Below the dropdown menus is a blue button labeled "Cambiar nivel".

### 7.4.5. Validaciones

Validar nivel actual del alumno, para mostrar el siguiente disponible en el select, los demás estarán “disabled”.

Si el nivel es el mas alto (avanzado) no se puede cambiar.

Validar para que no se pueda bajar de nivel.

Validar para que solo se pueda subir de a 1 nivel.

## 7.5.F04 – Asignar ejercicios para sus alumnos de un determinado nivel

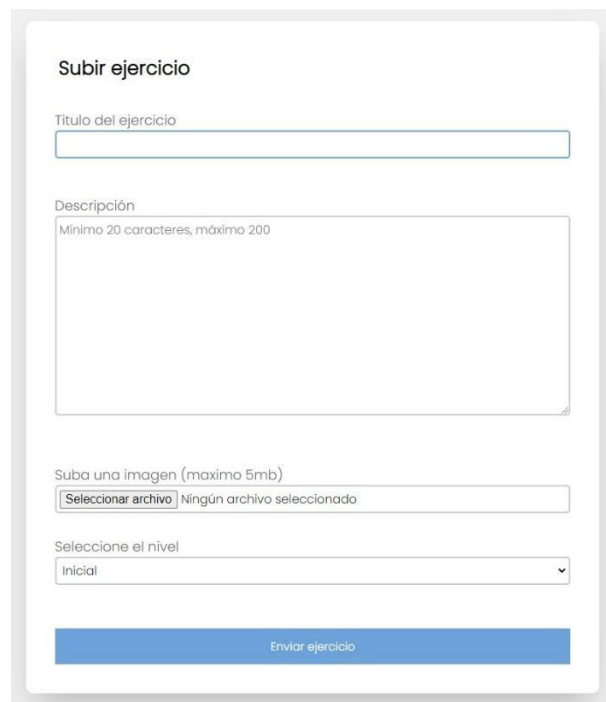
### 7.5.2. Acceso

Tipos de usuario: Docente

### 7.5.3. Descripción

Cada docente podrá plantear un ejercicio basado en uno de los 3 niveles. Este deberá tener un título, una imagen y una descripción en donde se plantea el objetivo a resolver.

### 7.5.4. Interfaz de usuario



Subir ejercicio

Título del ejercicio

Descripción

Minimo 20 caracteres, máximo 200

Suba una imagen (maximo 5mb)

Seleccionar archivo | Ningún archivo seleccionado

Seleccione el nivel

Inicial

Enviar ejercicio

### 7.5.5. Validaciones

La suma total de caracteres entre título y descripción no puede superar los 200 caracteres ni ser menor a 20 caracteres. Ninguno de los campos puede quedar vacío. Debe haber adjunta una imagen si o si antes de enviar el ejercicio. Tiene que haber un nivel seleccionado.

## 7.6.F05 – Poder dejar comentarios de devolución de ejercicios entregados por los alumnos

### 7.6.2. Acceso

Tipos de usuario: Docente

### 7.6.3. Descripción

Cada docente podrá redactar una devolución de las tareas entregadas por los alumnos. Cada docente recibirá las tareas de sus alumnos que no tengan devolución. Una vez realizada la devolución, la tarea dejará de ser visible.

### 7.6.4. Interfaz de usuario



### 7.6.5. Validaciones

La devolución no puede estar vacía al ser enviada.

## 7.7.F06 – Visualizar información estadística (general) acerca de alumnos de un docente según ejercicios resueltos

### 7.7.2. Acceso

Tipos de usuario: Docente

### 7.7.3. Descripción

Cada docente podrá ver en un panel distintos datos estadísticos sobre sus alumnos y la cantidad de tareas entregadas de los mismos. La estadística también podrá ser visualizada de manera individual (por alumno).

### 7.7.4. Interfaz de usuario



**Estadísticas**

El alumno que mas entregó tareas es: **Rodrigo Bueno (210)**

Se han entregado: **1337** tareas en total para usted por parte de sus alumnos

Seleccione uno de sus alumnos para ver sus datos de entregas

Julieta Venegas ▼

Julieta ha entregado: **31** tareas en total, y se han planteado **33** para su nivel

### 7.7.5. Validaciones

Los alumnos mostrados en las estadísticas son los correspondientes a ese docente.

## 7.8.F07 – Poder ver y buscar los ejercicios que su docente plantea según el nivel

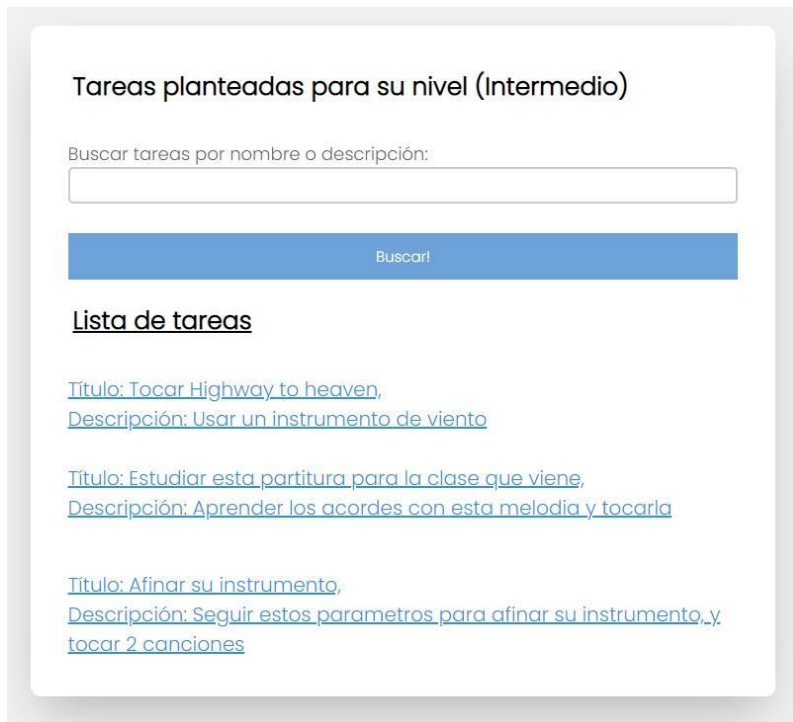
### 7.8.2. Acceso

Tipos de usuario: Alumno.

### 7.8.3. Descripción

Cada alumno podrá ver los ejercicios que su docente le plantea. También podrá buscar por título/descripción.

### 7.8.4. Interfaz de usuario



The screenshot shows a web interface titled "Tareas planteadas para su nivel (Intermedio)". Below the title is a search bar with the placeholder text "Buscar tareas por nombre o descripción:". A blue button labeled "Buscar!" is positioned below the search bar. Underneath the button is the heading "Lista de tareas". The list contains three items, each with a title and a description, both underlined and in blue text:

- Título: Tocar Highway to heaven,  
Descripción: Usar un instrumento de viento
- Título: Estudiar esta partitura para la clase que viene,  
Descripción: Aprender los acordes con esta melodía y tocarla
- Título: Afinar su instrumento,  
Descripción: Seguir estos parametros para afinar su instrumento, y tocar 2 canciones

### 7.8.5. Validaciones

Las búsquedas serán en los títulos de los ejercicios, en caso de no tener resultado se buscará en las descripciones. En caso de no haber coincidencia avisar al alumno.

Las tareas mostradas son las correspondientes al nivel y al docente.

## 7.9.F08 – Realizar la entrega de un ejercicio, adjuntando un audio correspondiente a dicha tarea


### 7.9.2. Acceso

Tipos de usuario: Alumno.

### 7.9.3. Descripción

Cada alumno podrá cargar un archivo de audio de la tarea correspondiente.

### 7.9.4. Interfaz de usuario



**Enviar Tarea**

Titulo: Tocar escalas (nivel: Intermedio)

Descripción: Lograr tocarlas 3 veces seguidas

Subir archivo de audio, maximo 7mb (se recomienda mp3)

Ningún archivo seleccionado

### 7.9.5. Validaciones

El alumno no puede entregar dos veces el mismo ejercicio.

Si no hay archivo adjunto, no se puede enviar la tarea.

Extra: Agregar una confirmación de envió, con mensaje: ¿Está seguro de enviar tarea34.mp3? No podrás modificar el archivo una vez enviado.

## 7.10. F09 –Visualizar los ejercicios resueltos

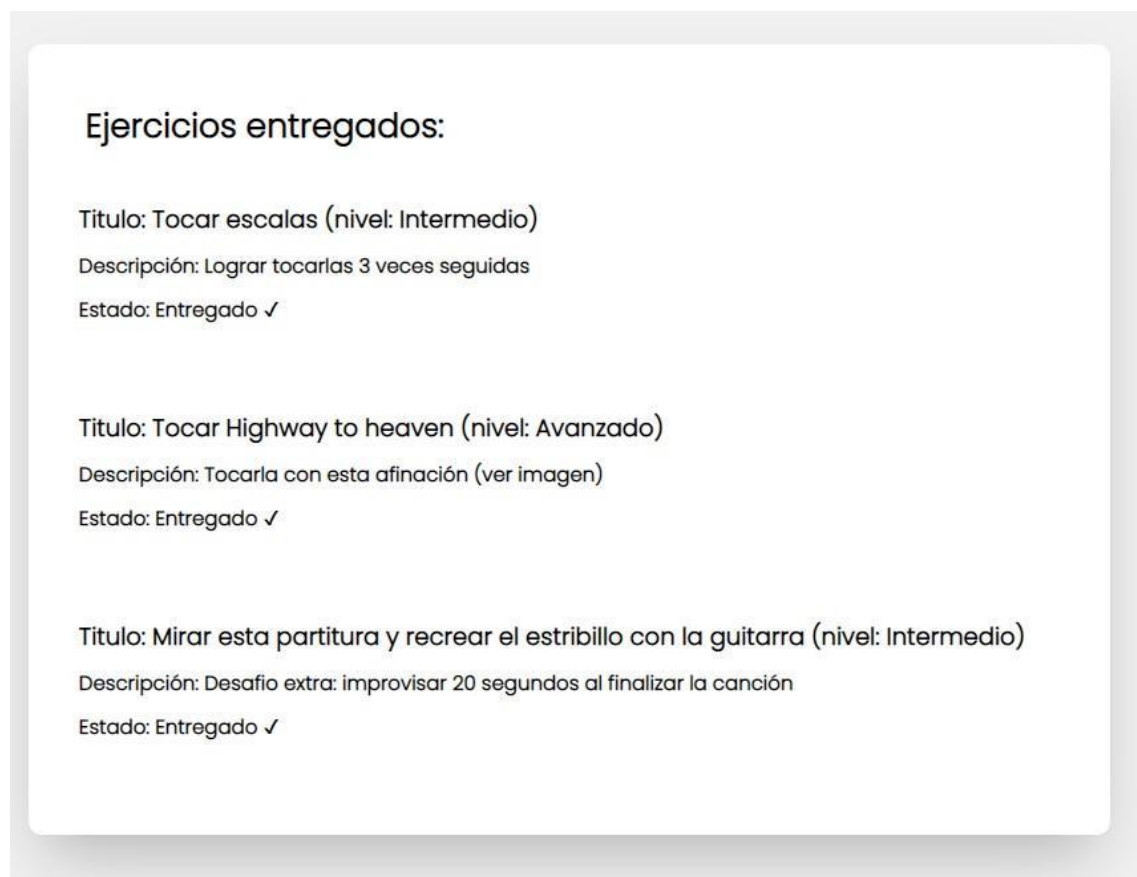
### 7.10.2. Acceso

Tipos de usuario: Alumno.

### 7.10.3. Descripción

Cada alumno podrá ver todos los ejercicios que ha entregado.

### 7.10.4. Interfaz de usuario



### 7.10.5. Validaciones

Si no ha entregado ningún ejercicio, se verá un mensaje que lo indique. Que los ejercicios mostrados sean los entregados por el alumno.



## 7.11. F10 – Visualizar información estadística (individual) acerca de ejercicios resueltos

### 7.11.2. Acceso

Tipos de usuario: Alumno.

### 7.11.3. Descripción

Cada alumno podrá ver la información estadística de sus ejercicios planteados por su docente.

### 7.11.4. Interfaz de usuario



### 7.11.5. Validaciones

Si no ha entregado ningún ejercicio, se verá un mensaje que lo indique. Que la información se corresponda con el alumno y su nivel.

## 7.12. F11 – Cerrar la sesión (Log out)

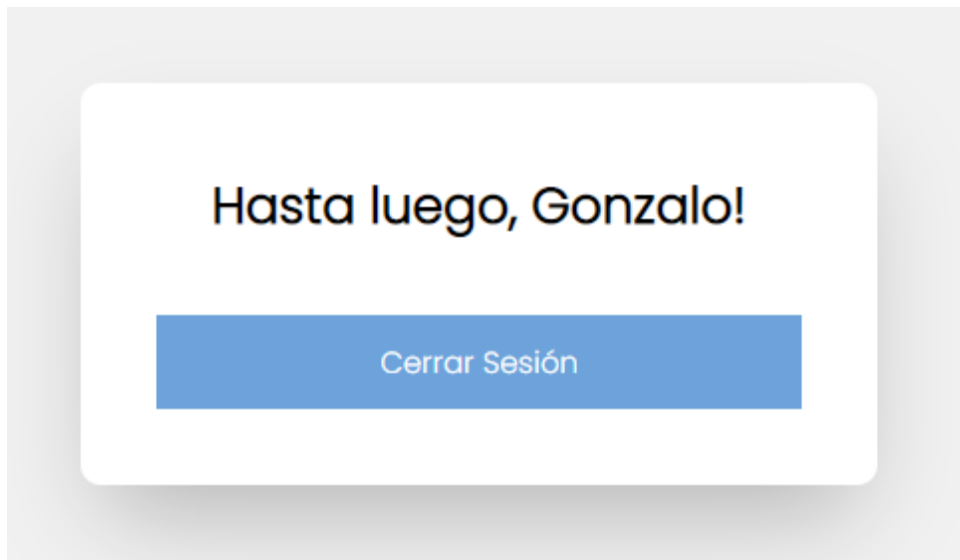
### 7.12.2. Acceso

Tipos de usuario: Docente y alumno.

### 7.12.3. Descripción

Botón para cerrar la sesión. Se volverá a la pantalla de inicio donde esté la opción de Iniciar sesión nuevamente.

### 7.12.4. Interfaz de usuario



### 7.12.5. Validaciones

Solo se muestra luego de haber iniciado sesión previamente.

## 8. Código completo

### 8.1.HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="icon" href="img/iconmonstr-disc-
5.svg" type="image/icon type" />
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <link rel="stylesheet" href="/estilo/estilo.css" />
    <title>
      Obligatorio - Nicolás Bañales - Francisco Aldado - Universidad ORT
-
      Docente Gonzalo Gentile
    </title>
  </head>
  <body>
    <!-- ##### PÁGINA INICIO ##### -->

    <div id="pagina-inicio" class="pagina-inicio">
      <!-- Header -->
      <header>
        <h1>Bienvenido a <b>Guitar Academy</b></h1>
        <h2>
          La mejor forma de aprender cuando estas encerrado <br />
          Y cuando no, también!
        </h2>

        <div class="btn-registro-contenedor">
          <button id="btnAbrirVentanaRegistro" class="btn-registro">
            Registrarme!
          </button>
```

```

</div>

<p>Ya tenés una cuenta? Iniciá sesión!</p>
<!-- Iniciar sesión -->
<div class="contentedor-iniciar-sesion">
  <!-- Log in item usuario -->
  <div class="item-log-in">
    <label for="logInUsuarioInput">Usuario:</label>
    <input
      type="text"
      id="logInUsuarioInput"
      placeholder="Ingrese su usuario"
      value="juanca"
    />
  </div>

  <!-- Log in item password -->
  <div class="item-log-in">
    <label for="logInPasswordInput">Contraseña:</label>
    <input
      type="password"
      id="logInPasswordInput"
      placeholder="Ingrese su contraseña"
      value="Pass1"
    />
  </div>

  <button id="btnLogin">Iniciar sesión</button>
  <br />
</div>

<span id="mensajeLogIn" class="mensajeLogIn"></span>
</header>
<!-- Main -->
<main>
  <div class="img-contentedor">
    <div class="img-columna">

```

```



<h2>Aprende Guitarra y mejora tu canto!</h2>
<p>
    Prueba nuestras lecciones y aprende a tocar tus temas prefe
ridos
    al mismo tiempo que te diviertes cantando! Sorprende a tu f
amilia
    y amigos en tiempo récord!
</p>
</div>

<div class="img-columna">
    

    <h2>Haz aprendido guitarra antes ?</h2>
    <p>
        Tal vez mas de una vez? Podemos ayudarte a alcanzar tus met
as
        mucho mas rápido! Contamos con un destacado plantel docente
que te
        pondran en el escenario antes de que lo sueñes.
    </p>
</div>

<div class="img-columna">
    

    <h2>Nuevo contenido todas las semanas!</h2>
    <p>
        Aprender guitarra no es solo un destino sino todo un viaje!
Te
        ayudaremos atravesar cada dificultad y a disfrutar el recor
rido.
        Sorprendete a ti mismo y a los demas mientras aprendes con
nosotros. Que esperas ?

```

```

        </p>
    </div>
</div>
</main>

<!-- Footer -->
<footer>
    Desarrollado con ♥ por Francisco y Nicolás <br> Universidad ORT -
    Materia: Programación 1 - Docente: Gonzalo Gentile
</footer>
</div>

<!-- ##### FORMULARIOS DE REGISTRO ##### -->

<!-- Formulario de registro -->
<div class="contenedor-ventana-registro">

    <!-- Solo formulario -->
    <div id="registroVentana" class="registroVentana">
        

        <!-- Nombre -->

        <h3>Registrarme!</h3>
        <label id="mensajeRegistroExitoso" class="mensajeRegistroExitoso"
></label>

        <div class="item-formulario">
            <label for="inputNombreRegistro">Nombre:</label>

            <input
                type="text"
                id="inputNombreRegistro"
                placeholder="Ingrese su nombre"
            />

```

```

        <span></span>
    </div>

    <!-- Usuario -->
    <div class="item-formulario">
        <label for="inputUsuarioRegistro">Usuario:</label>
        <input
            type="text"
            id="inputUsuarioRegistro"
            placeholder="Ingrese su usuario"
        />
        <span></span>
    </div>

    <!-- Contraseña -->
    <div class="item-formulario">
        <label for="inputPasswordRegistro">Contraseña:</label>
        <input
            type="password"
            id="inputPasswordRegistro"
            placeholder="Ingrese su contraseña"
        />
        <span></span>

    </div>

    <!-- Confirmacion Contraseña -->

    <div class="item-formulario" >
        <label for="confirmPasswordRegistro">Confirmar contraseña:</lab
el>

        <input
            type="password"
            id="confirmPasswordRegistro"
            placeholder="Confirmar contraseña"
        />
        <span></span>
    </div>

```

```

    </div>

    <!-- Eleccion de tipo de usuario -->
    <div class="item-formulario" id="radios">
        <div class="radioBtn">
            <label for="profesor">Soy profesor:</label>
            <input type="radio" id="profesorRadio" name="tipoUsuario" value
="docente" checked >
            <label for="alumno">Soy alumno:</label>
            <input type="radio" id="alumnoRadio" name="tipoUsuario" value="
alumno" >
        </div>

        <div class="item-formulario" >
            <div id="selectDocentesParaRegistro"></div>
            <span></span>

        </div>

    </div>

    <button id="btnRegistro">Registrarme</button>
</div>

</div>

<!-- ##### -->
<!-- ##### ** INTERIOR APLICACIÓN ** ##### -->
<!-- ##### -->
<div
    id="contenedor-aplicacion"
    class="contenedor-aplicacion ocultar-ventana"
>
    <!-- NAVEGACION INTERIOR APLICACIÓN -->

```



```

<div id="navegacion" class="navegacion"></div>
<h1 id="nombreUsuario" class="nombreUsuario"></h1>

<!--
- SECCION 1. DOCENTE: Cambiar nivel, ALUMNO: Ver ejercicios planteados --
>

<div class="seccion seccion1" id="seccion1"></div>

<!--
- SECCION 2. DOCENTE: Plantear Ejercicios, ALUMNO: Realizar entrega planteados -->
<div class="seccion seccion2" id="seccion2"></div>

<!--
- SECCION 3. DOCENTE: Redactar devolucion, ALUMNO: Ver ejercicios resueltos -->
<div class="seccion seccion3" id="seccion3"></div>

<!--
- SECCION 4. DOCENTE: Estadisticas de docente, ALUMNO: Estadisticas alumno -->
<div class="seccion seccion4" id="seccion4"></div>
</div>

<script src="js/clases.js"></script>
<script src="js/codigo.js"></script>
<script src="js/precargas.js"></script>
<script src="js/manejoUI.js"></script>
</body>
</html>

```

## 8.2. JavaScript

### 1.1.1. clases.js

```
// Clase para crear Nivel
class Nivel {
  constructor() {
    this.numero; // number
    this.nombre; // string
  }
}

// Clase para crear Docente
class Docente {
  constructor() {
    this.nombre; // string
    this.usuario; // string
    this.password; // string
    this.tipo = 'docente';
  }
}

// Clase para crear Alumno
class Alumno {
  constructor() {
    this.nombre; // string
    this.usuario; // string
    this.password; // string
    this.Nivel; // objeto
    this.Docente; // objeto
    this.tipo = 'alumno';
    this.entregasIndividuales = [];
  }
}

// Clase para crear Ejercicio
```

```

class Ejercicio {
  static nro = 1;
  constructor() {
    this.Nivel; // objeto
    this.id = Ejercicio.nro++; // identificador único
    this.tituloEjercicio; // string
    this.descripcionEjercicio; // string
    this.imagen; // imagen
    this.Docente; // objeto
  }
}

// Clase para crear Entrega
class Entrega {
  static nro = 1;
  constructor() {
    this.id = Entrega.nro++; // identificador unico
    this.Alumno; // objeto
    this.Ejercicio; // objeto
    this.Docente; // objeto
    this.sonido; // audio file
    this.devolucion; // string
  }
}

```

### 1.1.2. codigo.js

```

// Arrays a utilizar
let usuarios = [];
let niveles = [];
let ejercicios = [];
let entregas = [];

// Elementos útiles para funciones de Login:

```

```

// Input Login Usuario
let inputUsuarioLogIn = document.querySelector('#logInUsuarioInput');
// Input Login Password
let inputPasswordLogIn = document.querySelector('#logInPasswordInput');
// Elemento página de inicio (registro y login)
let paginaInicio = document.querySelector('#pagina-inicio');
// Elemento página interior aplicacion (luego del log in exitoso)
let paginaApp = document.querySelector('#contenedor-aplicacion');
// Nombre del usuario que se visualiza al hacer login exitoso
let nombreAMostrar = document.querySelector('#nombreUsuario');

// Esta cuenta va a ser la que logre pasar el login, y segun su tipo (doc
ente o alumno) se mostraran diferentes datos en las secciones.
let cuentaActiva = null;

// ##### Funciones crear y guardar #####
#####

// Función para crear un nuevo nivel (objeto) y agregarlo al array nivele
s
function crearYGuardarNivel(numero, nombre) {
  if (!isNaN(numero) && nombre.trim().length > 0 && isNaN(nombre)) {
    // Convertimos a número antes de agregarlo al objeto
    número = Number(numero);
    let unNivel = new Nivel();
    unNivel.numero = numero;
    unNivel.nombre = nombre;
    // Agregamos el objeto nivel al array de niveles
    niveles.push(unNivel);
  }
}

// Función para crear un nuevo docente (objeto) y agregarlo al array usua
rios
function crearYGuardarDocente(nombre, usuario, password) {
  if (

```

```

        validarNombre(nombre) &&
        validarUsuario(usuario) &&
        validarPassword(password) &&
        !validarUsuarioExistente(usuario)
    ) {
        let unDocente = new Docente();
        unDocente.nombre = nombre;
        unDocente.usuario = usuario.toLowerCase();
        unDocente.password = password;
        // Agregamos el objeto Docente al array de docentes
        usuarios.push(unDocente);
    }
}

// Función para crear un nuevo alumno (objeto) y agregarlo al array de usuarios
function crearYGuardarAlumno(nombre, usuario, password, nivel, docente) {
    if (
        validarNombre(nombre) &&
        validarUsuario(usuario) &&
        validarPassword(password) &&
        docente !== null &&
        nivel !== null &&
        !validarUsuarioExistente(usuario)
    ) {
        let unAlumno = new Alumno();
        unAlumno.nombre = nombre;
        unAlumno.usuario = usuario.toLowerCase();
        unAlumno.password = password;
        unAlumno.Nivel = nivel;
        unAlumno.Docente = docente;
        usuarios.push(unAlumno);
    }
}

```

```

// Función para crear un nuevo ejercicio (objeto) y agregarlo al array de
ejercicios
function crearYGuardarEjercicio(
    nivelObj,
    tituloEjercicio,
    descripcionEjercicio,
    imagen,
    docenteObj
) {
    if (
        nivelObj !== null &&
        docenteObj !== null &&
        tituloEjercicio.trim().length + descripcionEjercicio.trim().length >=
20 &&
        tituloEjercicio.trim().length + descripcionEjercicio.trim().length <=
200 &&
        imagen.trim().length > 4 &&
        tituloEjercicio.trim().length > 0 &&
        descripcionEjercicio.trim().length > 0 &&
        validarImagenArchivo(imagen)
    ) {
        let unEjercicio = new Ejercicio();
        unEjercicio.Nivel = nivelObj;
        unEjercicio.tituloEjercicio = tituloEjercicio;
        unEjercicio.descripcionEjercicio = descripcionEjercicio;
        unEjercicio.imagen = imagen;
        unEjercicio.Docente = docenteObj;
        ejercicios.push(unEjercicio);
    }
}

function crearYGuardarEntrega(
    nivel,
    alumno,
    ejercicio,
    docente,

```

```

    sonido,
    devolucion
  ) {
    if (
      nivel !== null &&
      alumno !== null &&
      ejercicio !== null &&
      docente !== null &&
      sonido.length > 4 &&
      validarSonidoArchivo(sonido)
    ) {
      let unaEntrega = new Entrega();
      unaEntrega.Nivel = nivel;
      unaEntrega.Alumno = alumno;
      unaEntrega.Ejercicio = ejercicio;
      unaEntrega.Docente = docente;
      unaEntrega.sonido = sonido;
      unaEntrega.devolucion = devolucion;

      // Agregamos la entrega al array general de entregas y al personal de
      // el alumno que entregó
      entregas.push(unaEntrega);
      alumno.entregasIndividuales.push(unaEntrega);
    }
  }

// ##### Funciones para obtener objetos con
// usuario, número o ID #####

// Esta función toma un nombre de usuario y devuelve el objeto de Docente
// con ese Usuario unico (en caso de existir) Nos aseguramos que sea de ese
// tipo poniendole el .tipo = 'docente' en el if statement
function obtenerDocenteConUsuario(usuario) {
  let docente = null;
  let i = 0;
  while (i < usuarios.length && docente === null) {

```

```

    // Chequeamos que coincida el usuario y el tipo para asegurarnos que
    devuelve un docente y no un alumno por error
    if (
        usuarios[i].usuario.toLowerCase() === usuario.toLowerCase() &&
        usuarios[i].tipo === 'docente'
    ) {
        docente = usuarios[i];
    }
    i++;
}
return docente;
}

// Esta función toma un nombre de usuario y devuelve el objeto de Alumno
con ese Usuario unico (en caso de existir) Nos aseguramos que sea de ese
tipo poniendole el .tipo = 'alumno' en el if statement
function obtenerAlumnoConUsuario(usuario) {
    let alumno = null;
    let i = 0;
    while (i < usuarios.length && alumno === null) {
        // Chequeamos que coincida el usuario y el tipo para asegurarnos que
        devuelve un alumno y no un alumno por error
        if (
            usuarios[i].usuario.toLowerCase() === usuario.toLowerCase() &&
            usuarios[i].tipo === 'alumno'
        ) {
            alumno = usuarios[i];
        }
        i++;
    }
    return alumno;
}

// Esta función toma un id y devuelve el objeto de Ejercicio de acuerdo a
ese id único
function obtenerEjercicioConID(id) {

```



```

    let ejercicio = null;
    let i = 0;
    while (i < ejercicios.length && ejercicio === null) {
        if (ejercicios[i].id === id) {
            ejercicio = ejercicios[i];
        }
        i++;
    }

    return ejercicio;
}

// Esta función toma un número y devuelve el objeto de Nivel de acuerdo a ese número
function obtenerNivelConNumero(numero) {
    let nivel = null;
    let i = 0;
    while (i < niveles.length && nivel === null) {
        if (niveles[i].numero === numero) {
            nivel = niveles[i];
        }
        i++;
    }
    return nivel;
}

// Esta función toma un id y devuelve el objeto de Entrega de acuerdo a ese id
function obtenerEntregaConID(id) {
    let entrega = null;
    let i = 0;
    while (i < entregas.length && entrega === null) {
        if (entregas[i].id === id) {
            entrega = entregas[i];
        }
        i++;
    }

```

```

    }
    return entrega;
}

// ##### Funcionalidad Registro #####
#####

function registro() {
    // No capturamos los value directamente porque nos interesa acceder luego al parentElement de cada elemento, para mostrar distintos errores en sus correspondientes span, y agregarle la class .error para pintar su borde de rojo
    let nombreInput = document.querySelector('#inputNombreRegistro');
    let usuarioInput = document.querySelector('#inputUsuarioRegistro');
    let passwordInput = document.querySelector('#inputPasswordRegistro');
    let confirmPasswordInput = document.querySelector('#confirmPasswordRegistro');
    let radioValor = valorRadioBtnRegistro();

    // Llamamos a estas funciones por si el usuario apreta "registrarme" se pondrían en rojo con su correspondiente mensaje en todos los campos que no validen, al mismo tiempo.
    validarNombreReg(nombreInput);
    validarUsuarioReg(usuarioInput);
    validarPasswordReg(passwordInput);
    compararPasswordsReg(passwordInput, confirmPasswordInput);

    if (
        validarNombreReg(nombreInput) &&
        validarUsuarioReg(usuarioInput) &&
        validarPasswordReg(passwordInput) &&
        compararPasswordsReg(passwordInput, confirmPasswordInput) &&
        radioValor !== undefined
    ) {
        // Tipo docente
        if (radioValor === 'docente') {
            // guardamos al docente en el array usuarios

```

```

    crearYGuardarDocente(
        nombreInput.value.trim().toLowerCase(),
        usuarioInput.value.trim(),
        passwordInput.value.trim()
    );

    // Esta funcion muestra nuestro UI y define la nueva cuentaActiva
    finRegistroExitoso();

    // Tipo alumno
} else if (radioValor === 'alumno') {
    let selectDocente = document.querySelector(
        '#selectDocenteRegistro'
    ).value;

    if (selectDocente !== -1) {
        // guardamos al alumno en el array usuarios
        crearYGuardarAlumno(
            nombreInput.value.trim(),
            usuarioInput.value.trim().toLowerCase(),
            passwordInput.value.trim(),
            obtenerNivelConNumero(1),
            obtenerDocenteConUsuario(selectDocente)
        );
        // Esta funcion muestra nuestro UI y define la nueva cuentaActiva
        finRegistroExitoso();
    }
}
}
}

function finRegistroExitoso() {
    document.querySelector(
        '#mensajeRegistroExitoso'
    ).innerHTML = `Registro exitoso ☒ Estamos creando tu perfil...⌚`;
}

```

```

let botonRegistro = document.querySelector('#btnRegistro');

// La cuenta activa es ahora el ultimo elemento del array usuarios
cuentaActiva = usuarios[usuarios.length - 1];

// Desactivamos el boton para que la función no pueda ser ejecutada nuevamente, mientras mostramos el mensaje de éxito
botonRegistro.setAttribute('disabled', 'true');
// Tardamos 2 segundos y medio para que se pueda ver el mensaje exitoso de registro
setTimeout(ocultarVentanaRegistro, 2500);
setTimeout(function () {
    mostrarUISegunUsuario(cuentaActiva);
}, 2500);
}

// Nos devuelve el valor del radio button que este "cheked"
function valorRadioBtnRegistro() {
    let radios = document.getElementsByName('tipoUsuario');
    let valor;
    for (i = 0; i < radios.length; i++) {
        if (radios[i].checked) {
            valor = radios[i].value;
        }
    }
    return valor;
}

// ##### Funcionalidad Login y Logout #####
#####

function logIn() {
    let i = 0;
    // Buscamos la cuenta ingresada, si la encontramos vamos a la contraseña
    while (cuentaActiva === null && i < usuarios.length) {

```

```

    // usuarios[i] es cada objeto del array usuarios, el elemento actual
    en cada iteración

    // Se dejará de buscar una vez que encontramos el objeto
    if (
        usuarios[i].usuario.toLowerCase().trim() ===
        inputUsuarioLogIn.value.toLowerCase().trim()
    ) {
        cuentaActiva = usuarios[i];
    }
    i++;
}

// Chequeamos por contraseña
if (
    cuentaActiva !== null &&
    cuentaActiva.password === inputPasswordLogIn.value
) {
    // Hacemos desaparecer el span de error de login
    document.querySelector('#mensajeLogIn').innerHTML = '';

    // Mostramos el UI segun el usuario
    mostrarUISegunUsuario(cuentaActiva);
} else {
    // Hacemos aparecer el span del error login, poniendole un mensaje
    document.querySelector('#mensajeLogIn').innerHTML =
        'Usuario o contraseña incorrectos! Ingreselos nuevamente';
}
}

function logOut() {
    // Reseteamos la cuentaActiva
    cuentaActiva = null;

    // Mostramos la pagina de inicio nuevamente, y ocultamos la de la aplic
    ación
    paginaInicio.classList.remove('ocultar-ventana');
    paginaInicio.classList.add('mostrar-ventana');
}

```

```

paginaApp.classList.add('ocultar-ventana');

// Limpiamos inputs
limpiarUsuarioInput();
limpiarPasswordInput();
limpiarInputsRegistro();
}

// ##### Validaciones #####
#####

// Se permite cualquier nombre, no debe contener números ni caracteres es
peciales.
function validarNombreReg(inputNombre) {
    let validado = false;

    if (isNaN(inputNombre.value) && inputNombre.value.trim().length >= 4) {
        let bandera = true;
        let i = 0;
        while (bandera && i < inputNombre.value.trim().length) {
            if (
                inputNombre.value.charCodeAt(i) >= 33 &&
                inputNombre.value.charCodeAt(i) <= 64
            ) {
                bandera = false;
                mostrarErrorRegistro(
                    inputNombre,
                    'No debe contener números! Tampoco simbolos (!, #, $, &, @)'
                );
            } else {
                mostrarExitoRegistro(inputNombre);
                validado = true;
            }
            i++;
        }
    } else {

```

```

        mostrarErrorRegistro(inputNombre, 'Mínimo 4 caracteres. Solo letras.'
    );
    }

    return validado;
}

// Esta función valida al usuario, no debe existir previamente en el array de usuarios, se permiten numeros, pero no espacios, debe ser mayor a 4 caracteres
function validarUsuarioReg(inputUsuario) {
    let validado = false;

    if (
        isNaN(inputUsuario.value) &&
        inputUsuario.value.trim().length >= 4 &&
        !tieneEspacioEntreMedio(inputUsuario.value.trim())
    ) {
        // Si valida todo lo de arriba, aún debera validar que es único
        if (!validarUsuarioExistente(inputUsuario.value)) {
            validado = true;
            mostrarExitoRegistro(inputUsuario);
        } else {
            mostrarErrorRegistro(
                inputUsuario,
                'Ese nombre de usuario ya existe! Probá otro distinto.'
            );
        }
    } else {
        mostrarErrorRegistro(
            inputUsuario,
            'Mínimo 4 caracteres. Sin espacios entremedio.'
        );
    }

    return validado;
}

```

```

}
// Esta función valida la contraseña ingresada, debe contener al menos 1
// numero, 1 mayuscula y 1 minuscula, no debe tener espacios entremedio, may
// or a 4 caracteres
function validarPasswordReg(inputPassword) {
    let hayMayuscula = false;
    let hayMinuscula = false;
    let hayNumero = false;
    let validado = false;

    if (inputPassword.value.trim().length >= 4) {
        for (let i = 0; i < inputPassword.value.trim().length; i++) {
            // Chequea el carácter por en busca de mayusculas
            if (
                inputPassword.value.charCodeAt(i) >= 65 &&
                inputPassword.value.charCodeAt(i) <= 90
            ) {
                hayMayuscula = true;
            }
            // Chequea el carácter por en busca de minúsculas
            if (
                inputPassword.value.charCodeAt(i) >= 97 &&
                inputPassword.value.charCodeAt(i) <= 122
            ) {
                hayMinuscula = true;
            }
            // Chequea el carácter por en busca de números
            if (
                inputPassword.value.charCodeAt(i) >= 48 &&
                inputPassword.value.charCodeAt(i) <= 57
            ) {
                hayNumero = true;
            }
        }
    }
}

```



```

if (
    hayMayuscula &&
    hayMinuscula &&
    hayNumero &&
    !tieneEspacioEntreMedio(inputPassword.value.trim())
) {
    validado = true;
    mostrarExitoRegistro(inputPassword);
} else {
    mostrarErrorRegistro(
        inputPassword,
        'Mínimo 4 caracteres, al menos una mayuscula, una minúscula y un número. Sin espacios entremedio.'
    );
}
return validado;
}

// Esta función compara las contraseñas, y ambas son iguales, devuelve true
function compararPasswordsReg(input1, input2) {
    let validado = false;
    if (
        input1.value.trim() === input2.value.trim() &&
        input2.value.trim().length >= 4
    ) {
        mostrarExitoRegistro(input2);
        validado = true;
    } else {
        mostrarErrorRegistro(input2, 'Las contraseñas deben coincidir!');
    }
    return validado;
}

```

```

// Función para validar tipo de archivo de imagen
function validarImagenArchivo(nombreArchivo) {
    let siguienteIndicePunto = nombreArchivo.lastIndexOf('.') + 1;

    let extensionArchivo = nombreArchivo
        .substr(siguienteIndicePunto, nombreArchivo.length)
        .toLowerCase();

    if (extensionArchivo == 'jpg' || extensionArchivo == 'png') {
        return true;
    } else {
        return false;
    }
}

// Función para validar tipo de archivo de sonido
function validarSonidoArchivo(nombreArchivo) {
    let siguienteIndicePunto = nombreArchivo.lastIndexOf('.') + 1;
    let extensionArchivo = nombreArchivo
        .substr(siguienteIndicePunto, nombreArchivo.length)
        .toLowerCase();

    if (
        extensionArchivo == 'mp3' ||
        extensionArchivo == 'm4a' ||
        extensionArchivo == 'wav' ||
        extensionArchivo == 'flac'
    ) {
        return true;
    } else {

```

```

    return false;
  }
}

// Función para validar contraseñas (al menos: 4 caracteres, una minúscula, una mayúscula, un número y sin espacios entre medio)
function validarPassword(password) {
  let mayuscula = false;
  let minuscula = false;
  let numero = false;

  if (password.length >= 4) {
    for (let i = 0; i < password.length; i++) {
      // Chequea el carácter por en busca de mayúsculas
      if (password.charCodeAt(i) >= 65 && password.charCodeAt(i) <= 90) {
        mayuscula = true;
      }
      // Chequea el carácter por en busca de minúsculas
      if (password.charCodeAt(i) >= 97 && password.charCodeAt(i) <= 122)
    {
      minuscula = true;
    }
      // Chequea el carácter por en busca de números
      if (password.charCodeAt(i) >= 48 && password.charCodeAt(i) <= 57) {
        numero = true;
      }
    }
  }

  if (mayuscula && minuscula && numero && !tieneEspacioEntreMedio(password)) {
    return true;
  }
  return false;
}

```

*// Función para validar que el usuario no tenga espacios entre medio y se a mayor a 4 caracteres (puede incluir algun número si se quiere, ya que no se especifica en el obligatorio(Por ej: ericClapton1996)).*

```
function validarUsuario(usuario) {  
  //Le sacamos el espacio de los costados  
  usuario = usuario.trim();  
  if (  
    usuario.length >= 4 &&  
    isNaN(usuario) &&  
    !tieneEspacioEntreMedio(usuario) &&  
    !validarUsuarioExistente(usuario)  
  ) {  
    return true;  
  }  
  
  return false;  
}
```

*// El nombre no puede contener números ni símbolos, pero si puede contener espacios entre medio (Por ej: JC Martinez Gimenez). Debe ser mayor o igual a 4 caracteres*

```
function validarNombre(nombre) {  
  let validado = false;  
  
  if (isNaN(nombre) && nombre.trim().length >= 4) {  
    validado = true;  
    let i = 0;  
    while (validado && i < nombre.trim().length) {  
      if (nombre.charCodeAt(i) >= 33 && nombre.charCodeAt(i) <= 64) {  
        validado = false;  
      }  
      i++;  
    }  
  }  
}
```

```

    return validado;
}

// Función que detecta si hay un espacio entre medio entre un string ingre
esado, si es así, devuelve true
function tieneEspacioEntreMedio(string) {
    let tieneEspacio = false;

    string = string.trim();
    let i = 0;

    while (!tieneEspacio && i < string.length) {
        if (string.charAt(i) === ' ') {
            tieneEspacio = true;
        }
        i++;
    }

    return tieneEspacio;
}

// Esta función retorna true si ya existe ese usuario en el array usuario
s
function validarUsuarioExistente(nombreDeUsuario) {
    let i = 0;
    let yaExiste = false;
    while (!yaExiste && i < usuarios.length) {
        let unUsuario = usuarios[i];
        if (nombreDeUsuario.toLowerCase() === unUsuario.usuario.toLowerCase()
    ) {
            yaExiste = true;
        }
        i++;
    }
    return yaExiste;
}

```

```

}

// Limpia barras
function quitarFakePath(pNom) {
  let nombreOk = '';
  let encuentreBarra = false;
  let posBarra = -1;
  let i = pNom.length - 1;
  while (i >= 0 && !encontreBarra) {
    let car = pNom[i];
    if (car === '\\' || car === '/') {
      encuentreBarra = true;
      posBarra = i;
    }
    i--;
  }

  if (encontreBarra) {
    nombreOk = pNom.substr(posBarra + 1);
  }
  return nombreOk;
}

```

### 1.1.3. manejoUI.js

```

misEventos();

function misEventos() {
  preCargaNiveles();
  preCargaDocentes();
  preCargaAlumnos();
  preCargaEjercicios();
  preCargaEntregas();
}

```

```

}

// ##### User Interface #####
#####

function mostrarUISegunUsuario(cuentaActiva) {
    // Sacamos de vista la página principal de registro y login, agregandol
    e la clase ocultar-ventana
    paginaInicio.classList.add('ocultar-ventana');

    // Hacemos aparecer la ventana de la aplicación con los datos y seccion
    es del docente o alumno, según corresponda
    mostrarNav(cuentaActiva);
    cargarSecciones(cuentaActiva);
    paginaApp.classList.remove('ocultar-ventana');

    // Mensaje de bienvenida
    nombreAMostrar.innerHTML = `Hola de nuevo, ${cuentaActiva.nombre}. <br>
    <p class="subTituloNombre">Seleccione un botón de la barra superior para
    ver la sección correspondiente.</p>`;
}

// Mostramos la navegación según el tipo de usuario logeado correctamente
function mostrarNav(usuario) {
    let nav;

    // tipo Docente
    if (usuario.tipo === 'docente') {
        nav = `<nav>
        <ul>
            <li id="btn-seccion1" class="item-lista">
                <a>Cambiar nivel de alumnos</a>
            </li>
        </ul>
        <ul>
            <li id="btn-seccion2" class="item-lista">

```

```

        <a>Plantear ejercicios</a>
    </li>
</ul>
<ul>
    <li id="btn-seccion3" class="item-lista">
        <a >Redactar devoluciones</a>
    </li>
</ul>
<ul>
    <li id="btn-seccion4" class="item-lista">
        <a>Ver información estadística</a></li>
</ul>
<button id="btnLogOut" class="btnlogOut">Cerrar Sesión</button>
</nav>`;
    // tipo Alumno
} else if (usuario.tipo === 'alumno') {
    nav = `<nav>
<ul>
    <li id="btn-seccion1" class="item-lista">
        <a>Ver ejercicios planteados</a>
    </li>
</ul>
<ul>
    <li id="btn-seccion2" class="item-lista">
        <a>Realizar entrega de ejercicio</a>
    </li>
</ul>
<ul>
    <li id="btn-seccion3" class="item-lista">
        <a>Ver ejercicios resueltos</a>
    </li>
</ul>
<ul>
    <li id="btn-seccion4" class="item-lista">
        <a >Ver información estadística</a></li>
</ul>

```



```

        <button id="btnLogOut" class="btnlogOut">Cerrar Sesión</button>

</nav>`;
    }
    document.querySelector('#navegacion').innerHTML = nav;
    document.querySelector('#btnLogOut').addEventListener('click', logOut);

    let navLinks = document.querySelectorAll('.item-lista');

    for (let i = 0; i < navLinks.length; i++) {
        navLinks[i].addEventListener('click', mostrarSeccionIndividual);
    }

    document.querySelector('#btn-seccion1').click();
}

// Ocultar y mostrar secciones
function mostrarSeccionIndividual() {
    ocultarSecciones();
    let idBoton = this.getAttribute('id');
    let idSeccion = '#' + idBoton.substring(4);

    document.querySelector(idSeccion).style.display = 'block';
}

function ocultarSecciones() {
    let secciones = document.querySelectorAll('.seccion');
    for (let i = 0; i < secciones.length; i++) {
        secciones[i].style.display = 'none';
    }
}

function cargarSecciones(usuario) {
    // Si es docente
    if (usuario.tipo === 'docente') {

```

```

    seccion1Docente(usuario);
    seccion2Docente();
    seccion3Docente();
    seccion4Docente();

    // Si es alumno
} else if (usuario.tipo === 'alumno') {
    seccion1Alumno(usuario);
    seccion2Alumno(usuario);
    seccion3Alumno();
    seccion4Alumno(usuario);
}
}

/* #####
## */
/* ##### *** USER INTERFACE DOCENTE *** #####
## */
/* #####
## */

// ##### Seccion 1 DOCENTE #####

function seccion1Docente(usuario) {
    let titulo = `

## 


```

```

    if (
        unAlumno.tipo === 'alumno' &&
        unAlumno.Docente.usuario === usuario.usuario
    ) {
        selectAlumnos += `<option value="${unAlumno.usuario}"> ${unAlumno.nombre} - (${unAlumno.usuario}) </option>`;
    }
}

selectAlumnos += `</select>`;

document.querySelector('#seccion1').innerHTML =
    titulo + selectAlumnos + divSelectNiveles;

// Evento que se activa al seleccionar un alumno de select, sirve para
cambiar de nivel
document
    .querySelector('#selListaAlumnos')
    .addEventListener('click', mostrarSelectCambiarNivel);
}

function mostrarSelectCambiarNivel() {
    // El value del select de los alumnos del docente de arriba, es el nombre de usuario, así obtenemos el objeto de ese alumno con esta función:
    let alumnoSeleccionado = obtenerAlumnoConUsuario(
        document.querySelector('#selListaAlumnos').value
    );

    let botonCambiarNivel = `<button id="btnCambiarNivel">Subir nivel</button>`;

    let divMensaje = `<div id="mensajeCambioDeNivel" class="mensajeCambioDeNivel"></div>`;

    if (alumnoSeleccionado !== null) {
        let selectNiveles = `<select class="selListaNiveles" id="selListaNiveles"><option disabled value="${alumnoSeleccionado.Nivel.numero}"> Nivel

```

```

actual (${alumnoSeleccionado.Nivel.numero}) : ${alumnoSeleccionado.Nivel.
nombre}</option>`;

    if (alumnoSeleccionado.Nivel.numero === 1) {
        selectNiveles += `<option selected value="${2}"> Siguiente Nivel: I
ntermedio (2)</option>`;
    } else if (alumnoSeleccionado.Nivel.numero === 2) {
        selectNiveles += `<option selected="true" value="${3}"> Siguiente N
ivel: Avanzado (3)</option>`;
    } else {
        selectNiveles += `<option disabled selected value="-
1"> Ya no hay más niveles para avanzar!</option>`;
    }

    // Mostramos el select de los niveles dinámicamente según alumno, y m
ostramos el botón para cambiarlo, junto con un div que puede mostrar mens
ajes
    selectNiveles += `</select>`;

    // Esto esta dentro del div
    document.querySelector('#mostrarNiveles').innerHTML =
        selectNiveles + botonCambiarNivel + divMensaje;

    document
        .querySelector('#btnCambiarNivel')
        .addEventListener('click', btnCambiarNivel);
} else {
    document.querySelector('#mostrarNiveles').innerHTML = ' ';
}
}

function btnCambiarNivel() {
    // Capturamos en una variable el alumno (objeto) seleccionado
    let alumnoSeleccionado = obtenerAlumnoConUsuario(
        document.querySelector('#selListaAlumnos').value
    );

```

```

// Capturamos en una variable el nivel (objeto) seleccionado
let nivelSeleccionado = obtenerNivelConNumero(
    Number(document.querySelector('#selListaNiveles').value)
);

if (
    alumnoSeleccionado.Nivel !== nivelSeleccionado &&
    nivelSeleccionado !== null
) {
    // Le cambiamos el nivel efectivamente al usuario
    alumnoSeleccionado.Nivel = nivelSeleccionado;

    // Se actualiza el select de niveles
    mostrarSelectCambiarNivel();
    document.querySelector(
        '#mensajeCambioDeNivel'
    ).innerHTML = `Nivel cambiado a ${alumnoSeleccionado.Nivel.numero} (<
b>${alumnoSeleccionado.Nivel.nombre})</b> con éxito!`;
} else {
    document.querySelector(
        '#mensajeCambioDeNivel'
    ).innerHTML = `Este es el máximo nivel! <b>No puedes cambiarlo</b>`;
}

// Actualizamos la seccion 3 y 4 (ya que ahí se ve el nivel de los alum
nos también)
seccion3Docente();
seccion4Docente();
}

// ##### Seccion 2 DOCENTE #####

function seccion2Docente() {
    let titulo = `<h2>Plantear Ejercicios</h2>`;

```

```

    let selectNivel = `<label for="txtTitulo">Título de Ejercicio: </label><input id="txtTitulo" type="text"
      placeholder="Escriba su Título"><br>`;

    let textAreaPlantearEjercicios = `<label for="archivoImagen">Imagen:</label><input id="fileImagen" type="file"><br>`;

    let botonEnviarEjercicio = `

```

```

        titulo +
        selectNivel +
        inputTituloEjercicio +
        textAreaPlantearEjercicios +
        archivoImagen +
        botonEnviarEjercicio +
        divMensajeEjercicioPlanteado +
        `</div>`;

document
    .querySelector('#btnAgregarEjercicio')
    .addEventListener('click', btnAgregarEjercicio);
}

function btnAgregarEjercicio() {
    let msg;

    let nivelSeleccionado = obtenerNivelConNumero(
        Number(document.querySelector('#selListaNivelesEjS2').value)
    );

    let titulo = document.querySelector('#txtTitulo').value;
    let descripcion = document.querySelector('#txtDescripcion').value;
    let imagen = document.querySelector('#fileImagen').value;
    let archImagen = quitarFakePath(imagen);

    if (
        nivelSeleccionado !== null &&
        titulo.trim().length + descripcion.trim().length <= 200 &&
        titulo.trim().length + descripcion.trim().length >= 20 &&
        titulo.trim().length > 0 &&
        descripcion.trim().length > 0 &&
        archImagen.length > 4 &&
        validarImagenArchivo(archImagen)
    ) {
        msg = `Se agregó correctamente.✅`;
    }
}

```

```

crearYGuardarEjercicio(
    nivelSeleccionado,
    titulo,
    descripcion,
    archImagen,
    cuentaActiva
);

// Vaciamos campos luego de agregar exitosamente
document.querySelector('#txtTitulo').value = '';
document.querySelector('#txtDescripcion').value = '';
document.querySelector('#fileImagen').value = '';
document.querySelector('#selListaNivelesEjs2').value = '-1';
} else if (nivelSeleccionado === null) {
    msg = 'Debe seleccionar un nivel ✕';
} else if (
    titulo.trim().length + descripcion.trim().length > 200 ||
    titulo.trim().length + descripcion.trim().length < 20 ||
    titulo.trim().length === 0 ||
    descripcion.trim().length === 0
) {
    msg =
        'El título y la descripción deben tener como mínimo 20 caracteres,
máximo 200. Ambos campos deben contener texto ✕';
} else if (archImagen.length < 4) {
    msg = 'Debe seleccionar una imagen ✕';
} else if (!validarImagenArchivo(archImagen)) {
    msg = 'Debe adjuntar una imagen (jpg o png) ✕';
}

document.querySelector('#divMensajeEjercicioPlanteado').innerHTML = msg
;
}

// ##### Seccion 3 DOCENTE #####

```



```

function seccion3Docente() {
    let titulo = `

## 


```

```

    }
  }
}
tabla += `</table>`;

document.querySelector('#seccion3').innerHTML = titulo + divMensaje + t
tabla;

// Esta función comprobará si la tabla esta vacia (tiene solo 1 fila, o
sea los title headers), en caso de estarlo, mostrara un mensaje de que no
hay tareas pendientes para corregir
mensajeSiTablaVacía();





// Hacemos un nodeList de todos los botones de la tabla para agregarles
un evento
let botonesEnviarDevolucion = document.querySelectorAll('.btnFila');

for (let j = 0; j < botonesEnviarDevolucion.length; j++) {
  botonesEnviarDevolucion[j].addEventListener('click', enviarDevolucion
);
}
}

function enviarDevolucion() {
  // Accedemos a los elementos de la fila del boton apretado, como al inpu
t que necesitamos.
  let botonApretado = this;
  let fila = botonApretado.parentElement.parentElement; // el padre del p
adre del boton es su fila
  // Capturamos el value de el input de la fila
  let inputDeFila = fila.querySelector('input').value;
  let divMensaje = document.querySelector('#msgAlEntregarTarea');
  // Accedemos al objeto único de entrega usando el dataset del id y la f
uncion obtenerEntregaConID
  let ejercicioDeLaFilaObjeto = obtenerEntregaConID(Number(fila.dataset.i
d));

```

```



if (inputDeFila.trim().length > 0 && isNaN(inputDeFila)) {
    ejercicioDeLaFilaObjeto.devolucion = inputDeFila;
    // Actualizamos la sección 3, no se deberán ver las tareas que tengan
    devolución
    divMensaje.innerHTML =
        '<p>  Devolución enviada con éxito ,  ¡Actualizando tareas!
 </p>';

```

*//NOTA IMPORTANTE: Lo que nos pasaba acá era que al actualizar la sección (para ver las siguientes tareas sin devolución( llamando a seccion3Docente ), no se mostraba nunca el mensaje de éxito de arriba obviamente, entonces le pusimos un pequeño delay de 2.5 segundos, para que se vea el mensaje de que fue enviada, mientras anunciamos que actualizaremos la tabla. Desactivamos los botones e inputs de esa fila para que no se pueda escribir nada durante esos 2.5 segundos.*

```

fila.querySelector('input').setAttribute('disabled', 'true');
botonApretado.setAttribute('disabled', 'true');

// Se actualiza la tabla luego de 2.5 segundos
setTimeout(seccion3Docente, 2500);
} else {
    divMensaje.innerHTML = '<p>  Debe escribir algo de devolución!  <
/p>';
}
}

```

```

function mensajeSiTablaVacia() {
    let tabla = document.querySelector('#tablaTareasPorCorregir');

    if (tabla.rows.length === 1) {
        tabla.innerHTML =
            '<p>Por ahora no tiene tareas pendientes para corregir!</p>';
    }
}

```

```
// ##### Seccion 4 DOCENTE #####

function seccion4Docente() {
  let titulo = '<h2>Ver información estadística</h2>';
  let contadorTotalTareasEntregadas = 0;
  let mayorCantidadDeEjResueltos = Number.NEGATIVE_INFINITY;
  let alumnoQueMasResolvio = '';
  let selectAlumnos = `<select id="selectAlumnoEstadisticas"><option valu
e="-1">Elija uno de sus alumnos para ver sus estadísticas</option>`;
  let divMensaje = `<div id="mensajeEstadisticasIndividuales"></div>`;
  // Mostrar al alumno que ha resuelto más ejercicios

  //Capturamos el máximo de tareas entregadas
  for (let j = 0; j < usuarios.length; j++) {
    let unUsuario = usuarios[j];

    if (unUsuario.tipo === 'alumno' && unUsuario.Docente === cuentaActiva
) {
      for (let x = 0; x < unUsuario.entregasIndividuales.length; x++) {
        cantidadEntregas = unUsuario.entregasIndividuales.length;

        if (mayorCantidadDeEjResueltos < cantidadEntregas) {
          mayorCantidadDeEjResueltos = cantidadEntregas;
        }
      }
    }
  }

  // Mostramos el o los alumnos que tengan ese máximo
  for (let i = 0; i < usuarios.length; i++) {
    let unUsuario = usuarios[i];
    if (unUsuario.tipo === 'alumno' && unUsuario.Docente === cuentaActiva
) {
      if (
        unUsuario.entregasIndividuales.length === mayorCantidadDeEjResuel
tos

```

```

    ) {
        alumnoQueMasResolvio +=
            '<b>' +
            unUsuario.nombre +
            '</b>' +
            ` (${mayorCantidadDeEjResueltos} tareas)` +
            ' | ';
    }
}

// Mostrar cantidad total de ejercicios entregados
for (let i = 0; i < entregas.length; i++) {
    let unaEntrega = entregas[i];

    if (unaEntrega.Docente.usuario === cuentaActiva.usuario) {
        contadorTotalTareasEntregadas++;
    }
}

// Select por alumnos

for (let i = 0; i < usuarios.length; i++) {
    let unAlumno = usuarios[i];

    if (
        unAlumno.tipo === 'alumno' &&
        unAlumno.Docente.usuario === cuentaActiva.usuario
    ) {
        selectAlumnos += `<option value="${unAlumno.usuario}"> ${unAlumno.n
ombre} - (${unAlumno.usuario}) </option>`;
    }
}

selectAlumnos += `</select>`;

```

```

document.querySelector('#seccion4').innerHTML =
    titulo +
    `

El o los alumnos que más resolvieron fueron: ${alumnoQueMasResolvio} <br/> El total de tareas entregadas por todos sus alumnos es: <b> ${contadorTotalTareasEntregadas}</b> </p>` +
    selectAlumnos +
    divMensaje;

document
    .querySelector('#selectAlumnoEstadisticas')
    .addEventListener('click', mostrarEstadisticasIndividuales);
}

function mostrarEstadisticasIndividuales() {
    let alumnoSeleccionado = obtenerAlumnoConUsuario(this.value);
    let todosLosEjPlanteados = 0;
    let resueltosDeSuNivel = 0;
    let divMensaje = document.querySelector('#mensajeEstadisticasIndividuales');

    if (alumnoSeleccionado !== null) {
        for (let i = 0; i < ejercicios.length; i++) {
            let unEjercicio = ejercicios[i];

            if (
                unEjercicio.Docente === alumnoSeleccionado.Docente &&
                unEjercicio.Nivel === alumnoSeleccionado.Nivel
            ) {
                todosLosEjPlanteados++;
            }
        }

        for (let j = 0; j < alumnoSeleccionado.entregasIndividuales.length; j++) {
            let unaEntregaIndividual = alumnoSeleccionado.entregasIndividuales[j];


```

```

        if (unaEntregaIndividual.Ejercicio.Nivel === alumnoSeleccionado.Nivel) {
            resueltosDeSuNivel++;
        }
    }

    divMensaje.innerHTML = `

De <b> ${todosLosEjPlanteados} </b> ejercicios planteados para ${alumnoSeleccionado.nombre} (Nivel: ${alumnoSeleccionado.Nivel.nombre}) <br> Ha resuelto: <b>${resueltosDeSuNivel} </b>ejercicios.</p>`;
    } else {
        // Vaciamos el div de las estadísticas, para que no quede la del alumno anteriormente seleccionado
        divMensaje.innerHTML = '';
    }
}

/* #####
## */

/* ##### *** USER INTERFACE ALUMNO *** #####
## */

/* #####
## */

// ##### Seccion 1 ALUMNO #####

function seccion1Alumno(usuario) {
    // Capturamos el objeto docente del alumno
    let docenteDelAlumno = usuario.Docente;

    let titulo = `

##


```

```

    let inputBuscador = `

```



```

chequearTablaEjerciciosVacía();

// Le agregamos un evento al boton de busqueda
document
  .querySelector('#buscaEjercicio')
  .addEventListener('click', btnBuscarEjercicio);
}

function btnBuscarEjercicio() {
  let docenteDelAlumno = cuentaActiva.Docente;

  let inputBusquedaTxt = document.querySelector('#inputBusquedaTxt').value;

  let resultado = '';
  let mensaje = '';
  let encontroPalabra = false;

  if (inputBusquedaTxt.trim().length > 0) {
    for (let i = 0; i < ejercicios.length; i++) {
      let unEjercicio = ejercicios[i];
      let img = ``;

      if (
        unEjercicio.Docente.usuario === docenteDelAlumno.usuario &&
        unEjercicio.Nivel === cuentaActiva.Nivel
      ) {
        if (!encontroPalabra) {
          mensaje =
            '<p>No encontramos más resultados que coincidan con su búsqueda!</p>';
        }
        if (
          unEjercicio.tituloEjercicio
            .toLowerCase()

```

```

        .indexOf(inputBusquedaTxt.toLowerCase()) !== -1
    ) {
        encontroPalabra = true;
        resultado += `<h2>${unEjercicio.tituloEjercicio}</h2> <br> <p>
${unEjercicio.descripcionEjercicio}</p><br> ${img} <br>`;
    } else if (
        unEjercicio.descripcionEjercicio
            .toLowerCase()
            .indexOf(inputBusquedaTxt.toLowerCase()) !== -1
    ) {
        encontroPalabra = true;
        resultado += `<h2>${unEjercicio.tituloEjercicio}</h2> <br> <p>
${unEjercicio.descripcionEjercicio}</p><br> ${img} <br>`;
    }
}
}
} else {
    resultado = '<p>Escriba algo antes de buscar</p>';
}

document.querySelector('#resultadoBusqueda').innerHTML = resultado + me
nsaje;
}

function chequearTablaEjerciciosVacía() {
    let tabla = document.querySelector('#tablaEjerciciosPlanteados');

    if (tabla.rows.length === 1) {
        tabla.innerHTML = '<p>No tiene ejercicios planteados para su nivel</p>
>';
    }
}

// ##### Sección 2 ALUMNO #####

function seccion2Alumno(usuario) {
    let titulo = `<h2>Realizar entrega de Ejercicios pendientes:</h2>`;

```

```

let docenteDelAlumno = usuario.Docente;
let selectDeEjercicios =
  '<select id="selectEjercicios"><option value="-
1">Elija un ejercicio a entregar</option>';
let botonEntrega = `<button id="btnEntrega" class="btnEntrega">Enviar t
area</button>`;
let divMensajeEntrega =
  '<div id="mensajeEntregaTarea" class="divMensajes"></div>';
for (let i = 0; i < ejercicios.length; i++) {
  unEjercicio = ejercicios[i];

  if (
    unEjercicio.Docente.usuario === docenteDelAlumno.usuario &&
    unEjercicio.Nivel === usuario.Nivel &&
    !checkObjetoRepetidoEnEntregas(unEjercicio, usuario)
  ) {
    selectDeEjercicios += `<option value="${unEjercicio.id}"> ${unEjerc
icio.tituloEjercicio}</option>`;
  }
}

let archivoSonido = `<br/><label for="archivoSonido">Adjunte archivo de
sonido:</label><input id="fileSonido" type="file"><br>`;

selectDeEjercicios += `</select>`;

document.querySelector('#seccion2').innerHTML =
  titulo +
  selectDeEjercicios +
  archivoSonido +
  botonEntrega +
  divMensajeEntrega;

document
  .querySelector('#btnEntrega')
  .addEventListener('click', btnEntregarTarea);

```

```

    seccion3Alumno();
    seccion4Alumno();
}

function checkObjetoRepetidoEnEntregas(obj, usuario) {
    for (let i = 0; i < entregas.length; i++) {
        if (entregas[i].Ejercicio === obj && entregas[i].Alumno === usuario)
        {
            return true;
        }
    }
    return false;
}

function btnEntregarTarea() {
    let docenteDelAlumno = obtenerDocenteConUsuario(cuentaActiva.Docente.usuario);

    let ejercicioAEntregar = obtenerEjercicioConID(
        Number(document.querySelector('#selectEjercicios').value)
    );
    let sonido = document.querySelector('#fileSonido').value;
    let archivoSonido = quitarFakePath(sonido);
    let mensaje = '';
    let devolucion = '';

    if (
        ejercicioAEntregar !== null &&
        archivoSonido.length > 4 &&
        validarSonidoArchivo(archivoSonido)
    ) {
        crearYGuardarEntrega(
            cuentaActiva.Nivel,
            cuentaActiva,
            ejercicioAEntregar,

```

```

        cuentaActiva.Docente,
        archivoSonido,
        devolucion
    );

    // Llamamos a seccion 2 de nuevo, para que se actualize el select con
    SOLO las tareas pendientes
    seccion2Alumno(cuentaActiva);

    mensaje = 'Tarea enviada con éxito ✅';
} else if (ejercicioAEntregar === null) {
    mensaje = 'Debe elegir una tarea ❌';
} else if (archivoSonido.length < 4) {
    mensaje = 'Debe adjuntar un archivo ❌';
} else if (!validarSonidoArchivo(archivoSonido)) {
    mensaje =
        'Debe adjuntar solo un audio con el formato correcto (m4a, mp3, wav
o flac) ❌';
}

document.querySelector('#mensajeEntregaTarea').innerHTML = mensaje;
}

// ##### Seccion 3 ALUMNO #####

function seccion3Alumno() {
    let titulo = `

## 


```

```

    let img = ``;

    if (
        unaEntrega.Alumno.usuario === cuentaActiva.usuario &&
        unaEntrega.devolucion !== ''
    ) {
        tabla += `<tr><td>${unaEntrega.Ejercicio.tituloEjercicio} <br> (Nivel: ${unaEntrega.Ejercicio.Nivel.nombre}) </td><td>${unaEntrega.Ejercicio.descripcionEjercicio}</td><td>${img}</td><td>${sonido}</td><td>${unaEntrega.devolucion}</td></tr>`;
    } else if (
        unaEntrega.Alumno.usuario === cuentaActiva.usuario &&
        unaEntrega.devolucion === ''
    ) {
        tabla += `<tr><td>${unaEntrega.Ejercicio.tituloEjercicio} <br> (Nivel: ${unaEntrega.Ejercicio.Nivel.nombre}) </td><td>${unaEntrega.Ejercicio.descripcionEjercicio}</td><td>${img}</td> <td>${sonido}</td><td>Aún el docente no redactó una devolución</td></tr>`;
    }
}

document.querySelector('#seccion3').innerHTML = titulo + tabla;

chequearTablaEntregasVacía();
}

function chequearTablaEntregasVacía() {
    let tabla = document.querySelector('#tablaTareasResueltas');

    if (tabla.rows.length === 1) {
        tabla.innerHTML = '<p>No ha entregado ninguna tarea</p>';
    }
}

// ##### Seccion 4 ALUMNO #####

```

```

function seccion4Alumno() {
  let titulo = `<h2>Ver información estadística</h2>`;
  let divMensaje;
  let docenteDelAlumno = obtenerDocenteConUsuario(cuentaActiva.Docente.usuario);
  let totalEjPlanteados = 0;

  let totalResueltos = cuentaActiva.entregasIndividuales.length;

  let totalResueltosParaSuNivel = 0;

  let contadorEntregasConDevolucion = 0;
  let porcentaje;

  for (let i = 0; i < ejercicios.length; i++) {
    unEjercicio = ejercicios[i];

    if (
      unEjercicio.Docente == docenteDelAlumno &&
      unEjercicio.Nivel === cuentaActiva.Nivel
    ) {
      totalEjPlanteados++;
    }
  }

  for (let j = 0; j < cuentaActiva.entregasIndividuales.length; j++) {
    if (cuentaActiva.entregasIndividuales[j].devolucion !== '') {
      contadorEntregasConDevolucion++;
    }

    if (
      cuentaActiva.entregasIndividuales[j].Ejercicio.Nivel ===
      cuentaActiva.Nivel
    ) {
      totalResueltosParaSuNivel++;
    }
  }
}

```

```

    }
}

if (totalEjPlanteados === 0) {
    porcentaje = 0;
} else {
    porcentaje = (totalResueltosParaSuNivel * 100) / totalEjPlanteados;
}

divMensaje = `

<p>Porcentaje de ejercicios resueltos para su nivel
(${cuentaActiva.Nivel.nombre}):<b> ${porcentaje}%</b><br>Total de ejercic
ios planteados por su docente para su nivel (${cuentaActiva.Nivel.nombre}
): <b>${totalEjPlanteados}</b> <br>De <b>${totalResueltos}</b> ejercicios
resueltos en total (para todos los niveles), tienen devolucion: <b>${con
tadorEntregasConDevolucion}</b></p></div>`;

document.querySelector('#seccion4').innerHTML = titulo + divMensaje;
}

// ##### Evento en el boton de login y en el
// de registrarme #####

// Boton Login
document.querySelector('#btnLogin').addEventListener('click', login);

// Boton registro (dentro del formulario)
document.querySelector('#btnRegistro').addEventListener('click', registro
);

// Al hacer click se limpian los inputs de login, usuario y contraseña
inputUsuarioLogin.addEventListener('click', limpiarUsuarioInput);
inputPasswordLogin.addEventListener('click', limpiarPasswordInput);

// Al apretar la opcion (radio button) alumno en el formulario de registr
o, se abre el desplegable de docentes para elegir uno
document


```



```

        .querySelector('#alumnoRadio')
        .addEventListener('click', radioAlumnoChecked);

// Al apretar la opcion (radio button) de docente en el formulario de reg
istro, se oculta el desplegable de docentes
document
    .querySelector('#profesorRadio')
    .addEventListener('click', radioDocenteChecked);

// ##### Mostrar y ocultar ventanas #####
#####

// Abrir y cerrar ventana de registro
let btnCerrarVentanaRegistro = document.querySelector('.cerrarBtnForm');

btnAbrirVentanaRegistro.addEventListener('click', abrirVentanaRegistro);

btnCerrarVentanaRegistro.addEventListener('click', cerrarVentanaRegistro)
;

function abrirVentanaRegistro() {
    let ventanaRegistroDocente = document.querySelector(
        '.contenedor-ventana-registro'
    );

    ventanaRegistroDocente.classList.add('mostrar-ventana');

    document.querySelector('#btnRegistro').disabled = false;
}

function cerrarVentanaRegistro() {
    let ventanaRegistroDocente = document.querySelector(
        '.contenedor-ventana-registro'
    );

    ventanaRegistroDocente.classList.remove('mostrar-ventana');
}

```

```

function ocultarVentanaRegistro() {
    let ventanaRegistroDocente = document.querySelector(
        '.contenedor-ventana-registro'
    );

    ventanaRegistroDocente.classList.remove('mostrar-ventana');
}

// Mostramos el select de profesores para el registro
function radioAlumnoChecked() {
    let selectDocentes = `<select id="selectDocenteRegistro">`;
    let label = `<label>Seleccione un profesor</label>`;
    for (let i = 0; i < usuarios.length; i++) {
        unUsuario = usuarios[i];

        if (unUsuario.tipo === 'docente') {
            selectDocentes += `<option value="${unUsuario.usuario}">${unUsuario
.nombre} - (${unUsuario.usuario}) </option>`;
        }
    }

    selectDocentes += `</select>`;

    document.querySelector('#selectDocentesParaRegistro').innerHTML =
        label + selectDocentes;
}

// Ocultamos el select de profesores para el registro si esta la opcion d
ocente seleccionada
function radioDocenteChecked() {
    document.querySelector('#selectDocentesParaRegistro').innerHTML = '';
}

```

*// Esta función toma el elemento html del input, y le pone la class .exit o a su parentElement para así pintar de verde su borde y limpiar el span del error*

```
function mostrarExitoRegistro(input) {  
    let elementoPadre = input.parentElement;  
    elementoPadre.classList.remove('error');  
    elementoPadre.classList.add('exito');  
  
    let span = elementoPadre.querySelector('span');  
  
    span.innerHTML = '';  
}
```

*// Esta función toma el elemento html del input, y le pone la class .erro r a su parentElement para así pintar de rojo su borde y las letras del sp an dentro del mismo*

```
function mostrarErrorRegistro(input, mensaje) {  
    let elementoPadre = input.parentElement;  
  
    elementoPadre.classList.remove('exito');  
    elementoPadre.classList.add('error');  
  
    let span = elementoPadre.querySelector('span');  
  
    span.innerHTML = mensaje;  
}
```

*// Limpieza de input de Usuario (Login) y mensaje*

```
function limpiarUsuarioInput() {  
    document.querySelector('#mensajeLogIn').innerHTML = '';  
    inputUsuarioLogIn.value = '';  
}
```

*// Limpieza de input de Password (Login) y mensaje*

```
function limpiarPasswordInput() {  
    document.querySelector('#mensajeLogIn').innerHTML = '';  
    inputPasswordLogIn.value = '';
```

```

}

// Limpia los inputs del registro
function limpiarInputsRegistro() {
    document.querySelector('#inputNombreRegistro').value = '';
    document.querySelector('#inputUsuarioRegistro').value = '';
    document.querySelector('#inputPasswordRegistro').value = '';
    document.querySelector('#confirmPasswordRegistro').value = '';
    document.querySelector('#mensajeRegistroExitoso').innerHTML = '';

    // Le sacamos el borde verde de la clase .exito que habia quedado previ
    o a algún registro exitoso en los inputs del formulario
    let itemsRegistro = document.querySelectorAll('.item-formulario');
    for (let i = 0; i < itemsRegistro.length; i++) {
        itemsRegistro[i].classList.remove('exito');
    }
}
}

```

#### 1.1.4. precargas.js

```

// Precarga Niveles
function preCargaNiveles() {
    crearYGuardarNivel(1, 'Inicial');
    crearYGuardarNivel(2, 'Intermedio');
    crearYGuardarNivel(3, 'Avanzado');
}

// Función para precargar docentes
function preCargaDocentes() {
    crearYGuardarDocente('Eric Clapton', 'eric1945', 'Heaven92');
    crearYGuardarDocente('Bob Dylan', 'bobbydylan', 'bWind1962');
}

```

```

    crearYGuardarDocente('Janis Joplin', 'lajanis', 'Rock1');
    crearYGuardarDocente('Amy Winehouse', 'amyrehab83', 'iSaidno1');
}

// Función para precargar alumnos
function preCargaAlumnos() {
    crearYGuardarAlumno(
        'Juan Carlos',
        'juanca',
        'Pass1',
        obtenerNivelConNumero(1),
        obtenerDocenteConUsuario('lajanis')
    );
    crearYGuardarAlumno(
        'María del Mar',
        'lamarydelcap',
        'Firulais88',
        obtenerNivelConNumero(2),
        obtenerDocenteConUsuario('eric1945')
    );
    crearYGuardarAlumno(
        'Roberto Carlos',
        'robbieuru',
        'Roca52',
        obtenerNivelConNumero(3),
        obtenerDocenteConUsuario('eric1945')
    );
    crearYGuardarAlumno(
        'Julia Rivera',
        'juliguitarra',
        'iloveTacos1996',
        obtenerNivelConNumero(1),
        obtenerDocenteConUsuario('eric1945')
    );
    crearYGuardarAlumno(
        'Federico Gimenez',

```

```

    'elfede',
    'Fefo3',
    obtenerNivelConNumero(1),
    obtenerDocenteConUsuario('lajanis')
);
crearYGuardarAlumno(
    'Laura Gomez',
    'laurasote',
    'Lauchita1',
    obtenerNivelConNumero(3),
    obtenerDocenteConUsuario('lajanis')
);
crearYGuardarAlumno(
    'Gabriel Richieri',
    'gabo87',
    'gaboteElbebote3',
    obtenerNivelConNumero(1),
    obtenerDocenteConUsuario('bobbydylan')
);
crearYGuardarAlumno(
    'Ramiro Ortega',
    'elramaorteguita',
    'iloveMusic24',
    obtenerNivelConNumero(3),
    obtenerDocenteConUsuario('bobbydylan')
);
crearYGuardarAlumno(
    'Erica Ramírez',
    'erica7',
    'Lapregunta99',
    obtenerNivelConNumero(2),
    obtenerDocenteConUsuario('bobbydylan')
);
crearYGuardarAlumno(
    'Rafa Diaz',
    'rafitamiles',

```

```

        'hyperActive17',
        obtenerNivelConNumero(2),
        obtenerDocenteConUsuario('amyrehab83')
    );
    crearYGuardarAlumno(
        'Ivan Modernell',
        'elivansabe',
        'micChecker21',
        obtenerNivelConNumero(1),
        obtenerDocenteConUsuario('amyrehab83')
    );
    crearYGuardarAlumno(
        'Nicolas Rodriguez',
        'estudio50',
        'js4Life',
        obtenerNivelConNumero(1),
        obtenerDocenteConUsuario('amyrehab83')
    );
}

// Función para precargar Ejercicios
function preCargaEjercicios() {
    crearYGuardarEjercicio(
        obtenerNivelConNumero(1),
        'Permutaciones',
        'Ejercicios secuenciales para trabajar velocidad y ritmo. Repetir 3 v
eces cada uno con diferentes articulaciones.',
        'ej1.png',
        obtenerDocenteConUsuario('lajanis')
    );
    crearYGuardarEjercicio(
        obtenerNivelConNumero(2),
        'Repeticiones',
        'Ejercicios repetitivos para mejorar la velocidad y ritmo. Repetir 5
veces con diferentes articulaciones.',
        'ej2.png',

```

```

    obtenerDocenteConUsuario('lajanis')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(1),
    'The Caterpillar',
    'Ejercicios cromaticos descendentes para trabajar lentamente la digitacion por todo el traste de la guitarra. Realizar a 60bpm, 75bpm y 90bpm por cada negra.',
    'ej3.png',
    obtenerDocenteConUsuario('lajanis')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(1),
    'Cromaticos a velocidad',
    'Ejercicios cromaticos para trabajar velocidad de la digitacion en todo el traste. Realizar con diferentes articulaciones a 80, 90 y 100 bpm por cada tiempo de negra.',
    'ej4.png',
    obtenerDocenteConUsuario('lajanis')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(1),
    'Acordes de Septimas',
    'Ejercicios de acordes con septimas mayor y menor. Repetir varias veces cada uno hasta memorizar la secuencia completa.',
    'ej5.png',
    obtenerDocenteConUsuario('lajanis')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(3),
    'Yankee Doodle',
    'Aprenderemos este tema clásico del folk estadounidense y que gracias a las series de dibujos animados y a las películas de Hollywood es conocida en muchas partes del mundo.',
    'ej6.png',
    obtenerDocenteConUsuario('lajanis')
);

```



```

);
crearYGuardarEjercicio(
    obtenerNivelConNumero(3),
    'Cromaticos a alta velocidad',
    'Ejercicios cromaticos para trabajar velocidad de la digitacion en to
do el traste. Realizar con diferentes articulaciones a 80, 90 y 100 bpm p
or cada tiempo de negra.',
    'ej4.png',
    obtenerDocenteConUsuario('bobbydylan')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(2),
    'Acordes de Septimas',
    'Ejercicios de acordes con septimas mayor y menor. Repetir varias vec
es cada uno hasta memorizar la secuencia completa.',
    'ej5.png',
    obtenerDocenteConUsuario('bobbydylan')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(3),
    'El regreso de Doodle',
    'Aprenderemos este tema clásico del folk estadounidense y que gracias
a las series de dibujos animados y a las películas de Hollywood es conoc
ida en muchas partes del mundo.',
    'ej6.png',
    obtenerDocenteConUsuario('bobbydylan')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(1),
    'Old McDonald has A Farm',
    'El viejo MacDonald tenía una granja es una popular canción infantil
perteneciente al folklore musical estadounidense, de autor anónimo. Tocar
de memoria',
    'ej7.png',
    obtenerDocenteConUsuario('bobbydylan')
);

```

```

    crearYGuardarEjercicio(
        obtenerNivelConNumero(2),
        'La marcha de los Santos',
        'Es un himno góspel que toma elementos de música folkórica. Autor des
conocido, y si bien es música espiritual hoy día es tocada por bandas de
jazz. Tocar de memoria.',
        'ej8.png',
        obtenerDocenteConUsuario('bobbydylan')
    );
    crearYGuardarEjercicio(
        obtenerNivelConNumero(1),
        'REM - Everybody Hurts',
        'Es una canción del grupo estadounidense R.E.M. La canción está inclu
ida en el álbum Automatic for the people de 1992 y fue lanzada como singl
e al año siguiente. Tocar de memoria.',
        'ej9.png',
        obtenerDocenteConUsuario('bobbydylan')
    );
    crearYGuardarEjercicio(
        obtenerNivelConNumero(3),
        'Yankee Doodle',
        'Aprenderemos este tema clásico del folk estadounidense y que gracias
a las series de dibujos animados y a las películas de Hollywood es conoc
ida en muchas partes del mundo.',
        'ej6.png',
        obtenerDocenteConUsuario('amyrehab83')
    );
    crearYGuardarEjercicio(
        obtenerNivelConNumero(1),
        'Old McDonald has A Farm',
        'El viejo MacDonald tenía una granja es una popular canción infantil
perteneciente al folklore musical estadounidense, de autor anónimo. Tocar
de memoria',
        'ej7.png',
        obtenerDocenteConUsuario('amyrehab83')
    );

```

```

    crearYGuardarEjercicio(
        obtenerNivelConNumero(2),
        'La marcha de los Santos',
        'Es un himno góspel que toma elementos de música folkórica. Autor des
conocido, y si bien es música espiritual hoy día es tocada por bandas de
jazz. Tocar de memoria.',
        'ej8.png',
        obtenerDocenteConUsuario('amyrehab83')
    );
    crearYGuardarEjercicio(
        obtenerNivelConNumero(3),
        'REM - Everybody Hurts',
        'Es una canción del grupo estadounidense R.E.M. La canción está inclu
ida en el álbum Automatic for the people de 1992 y fue lanzada como singl
e al año siguiente. Tocar de memoria.',
        'ej9.png',
        obtenerDocenteConUsuario('amyrehab83')
    );
    crearYGuardarEjercicio(
        obtenerNivelConNumero(1),
        'Permutaciones',
        'Ejercicios secuenciales para trabajar velocidad y ritmo. Repetir 3 v
eces cada uno con diferentes articulaciones.',
        'ej1.png',
        obtenerDocenteConUsuario('amyrehab83')
    );
    crearYGuardarEjercicio(
        obtenerNivelConNumero(2),
        'Repeticiones',
        'Ejercicios repetitivos para mejorar la velocidad y ritmo. Repetir 5
veces con diferentes articulaciones.',
        'ej2.png',
        obtenerDocenteConUsuario('amyrehab83')
    );
    crearYGuardarEjercicio(
        obtenerNivelConNumero(2),

```

```

    'La marcha de los Santos',
    'Es un himno góspel que toma elementos de música folkórica. Autor des
conocido, y si bien es música espiritual hoy día es tocada por bandas de
jazz. Tocar de memoria.',
    'ej8.png',
    obtenerDocenteConUsuario('eric1945')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(1),
    'REM - Everybody Hurts',
    'Es una canción del grupo estadounidense R.E.M. La canción está inclu
ida en el álbum Automatic for the people de 1992 y fue lanzada como singl
e al año siguiente. Tocar de memoria.',
    'ej9.png',
    obtenerDocenteConUsuario('eric1945')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(1),
    'Permutaciones',
    'Ejercicios secuenciales para trabajar velocidad y ritmo. Repetir 3 v
eces cada uno con diferentes articulaciones.',
    'ej1.png',
    obtenerDocenteConUsuario('eric1945')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(2),
    'Repeticiones',
    'Ejercicios repetitivos para mejorar la velocidad y ritmo. Repetir 5
veces con diferentes articulaciones.',
    'ej2.png',
    obtenerDocenteConUsuario('eric1945')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(1),
    'The Caterpillar',

```

```

    'Ejercicios cromaticos descendentes para trabajar lentamente la digitacion por todo el traste de la guitarra. Realizar a 60bpm, 75bpm y 90bpm por cada negra.',
    'ej3.png',
    obtenerDocenteConUsuario('eric1945')
);
crearYGuardarEjercicio(
    obtenerNivelConNumero(3),
    'Cromaticos a velocidad',
    'Ejercicios cromaticos para trabajar velocidad de la digitacion en todo el traste. Realizar con diferentes articulaciones a 80, 90 y 100 bpm por cada tiempo de negra.',
    'ej4.png',
    obtenerDocenteConUsuario('eric1945')
);
}

// Función para precargar entregas
function preCargaEntregas() {
    crearYGuardarEntrega(
        obtenerNivelConNumero(1),
        obtenerAlumnoConUsuario('juanca'),
        obtenerEjercicioConID(1),
        obtenerDocenteConUsuario('lajanis'),
        'ej1.m4a',
        'Muy buen trabajo'
    ),
    crearYGuardarEntrega(
        obtenerNivelConNumero(1),
        obtenerAlumnoConUsuario('juanca'),
        obtenerEjercicioConID(3),
        obtenerDocenteConUsuario('lajanis'),
        'ej3.m4a',
        ''
    ),
    crearYGuardarEntrega(

```

```

        obtenerNivelConNumero(1),
        obtenerAlumnoConUsuario('juanca'),
        obtenerEjercicioConID(4),
        obtenerDocenteConUsuario('lajanis'),
        'ej4.m4a',
        ''
    );
    crearYGuardarEntrega(
        obtenerNivelConNumero(3),
        obtenerAlumnoConUsuario('elramaorteguita'),
        obtenerEjercicioConID(7),
        obtenerDocenteConUsuario('bobbydylan'),
        'ej4.m4a',
        ''
    );
    crearYGuardarEntrega(
        obtenerNivelConNumero(1),
        obtenerAlumnoConUsuario('gabo87'),
        obtenerEjercicioConID(10),
        obtenerDocenteConUsuario('bobbydylan'),
        'ej7.m4a',
        ''
    );
    crearYGuardarEntrega(
        obtenerNivelConNumero(2),
        obtenerAlumnoConUsuario('erica7'),
        obtenerEjercicioConID(11),
        obtenerDocenteConUsuario('bobbydylan'),
        'ej8.m4a',
        'Muy bien! Cuidado de no trastear al cambiar de cuerda.'
    );
    crearYGuardarEntrega(
        obtenerNivelConNumero(1),
        obtenerAlumnoConUsuario('elivansabe'),
        obtenerEjercicioConID(14),
        obtenerDocenteConUsuario('amyrehab83'),

```

```

    'ej7.m4a',
    'Excelente interpretación de memoria. Sigue así !'
);
crearYGuardarEntrega(
    obtenerNivelConNumero(2),
    obtenerAlumnoConUsuario('rafitamiles'),
    obtenerEjercicioConID(15),
    obtenerDocenteConUsuario('amyrehab83'),
    'ej8.m4a',
    'Que linda interpretacion, casi lloran mis oidos!'
);
crearYGuardarEntrega(
    obtenerNivelConNumero(1),
    obtenerAlumnoConUsuario('elivansabe'),
    obtenerEjercicioConID(17),
    obtenerDocenteConUsuario('amyrehab83'),
    'ej1.m4a',
    ''
);
crearYGuardarEntrega(
    obtenerNivelConNumero(2),
    obtenerAlumnoConUsuario('rafitamiles'),
    obtenerEjercicioConID(18),
    obtenerDocenteConUsuario('amyrehab83'),
    'ej2.m4a',
    ''
);
crearYGuardarEntrega(
    obtenerNivelConNumero(2),
    obtenerAlumnoConUsuario('lamarydelcap'),
    obtenerEjercicioConID(19),
    obtenerDocenteConUsuario('eric1945'),
    'ej8.m4a',
    'Buen Trabajo! Pon cuidado en afinar antes de grabar.'
);
crearYGuardarEntrega(

```

```

    obtenerNivelConNumero(1),
    obtenerAlumnoConUsuario('juliguitarra'),
    obtenerEjercicioConID(20),
    obtenerDocenteConUsuario('eric1945'),
    'ej9.m4a',
    'Si querias hacer sangrar mis oidos... Lo conseguiste!.'
);
crearYGuardarEntrega(
    obtenerNivelConNumero(1),
    obtenerAlumnoConUsuario('juliguitarra'),
    obtenerEjercicioConID(21),
    obtenerDocenteConUsuario('eric1945'),
    'ej1.m4a',
    'Muy bien realizado, prueba tocarlo con guitarra Electrica.'
);
crearYGuardarEntrega(
    obtenerNivelConNumero(2),
    obtenerAlumnoConUsuario('lamarydelcap'),
    obtenerEjercicioConID(22),
    obtenerDocenteConUsuario('eric1945'),
    'ej2.m4a',
    ''
);
crearYGuardarEntrega(
    obtenerNivelConNumero(1),
    obtenerAlumnoConUsuario('juliguitarra'),
    obtenerEjercicioConID(23),
    obtenerDocenteConUsuario('eric1945'),
    'ej3.m4a',
    ''
);
crearYGuardarEntrega(
    obtenerNivelConNumero(3),
    obtenerAlumnoConUsuario('robbieuru'),
    obtenerEjercicioConID(24),
    obtenerDocenteConUsuario('eric1945'),

```



```

    'ej4.m4a',
    ''

);
}

```

## 1.2.CSS (estilo.css)

```

/* Fuente global */
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap');

/* ##### AJUSTES GLOBALES #####
##### */
*,
*:before,
*:after {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

html {
    font-size: 62.5%;
    scroll-behavior: smooth;
}

body {
    font-family: 'Poppins', sans-serif;
}

button {

```

```

font-family: 'Poppins', sans-serif;
font-size: 1.5rem;
cursor: pointer;
border-radius: 5px;
border: none;
padding: 0.3rem 1rem;
background-color: #75a7d6;
color: white;
}

button:hover {
  color: black;
  background-color: #d6dde4;
  transition: 0.4s all;
}

select {
  font-family: 'Poppins', sans-serif;
  cursor: pointer;
}

input {
  font-family: 'Poppins', sans-serif;
  outline: none;
}

/* ##### HEADER #####
*/

header {
  background: radial-gradient(
    circle,
    rgba(44, 115, 201, 1) 0%,
    rgba(10, 56, 112, 1) 100%
  );

```

```

    color: white;
    min-height: 35vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    padding: 2rem;
    text-align: center;
}

header h1 {
    font-size: 4.7rem;
    font-weight: 400;
    margin: 1rem 0rem;
}

header h2 {
    font-size: 2.5rem;
    font-weight: 400;
    margin-bottom: 1rem;
}

header p {
    font-size: 1.7rem;
    margin-top: 3rem;
    margin-bottom: 1rem;
}

header button {
    margin-left: 2rem;
    margin-top: 0.5rem;
}

/* ##### CONTENEDOR INICIAR SESIÓN #####
##### */

```

```

.contentedor-iniciar-sesion {
  display: flex;
  align-items: center;
  margin-bottom: 2rem;
}

.contentedor-iniciar-sesion input {
  border-radius: 5px;
  border: none;
  padding: 3px;
  text-align: center;
}

.mensajeLogIn {
  font-size: 1.3rem;
}

main {
  padding: 3rem;
  min-height: 65vh;
}

.item-log-in {
  font-size: 1.5rem;
  margin-left: 2rem;
}

.img-contenedor {
  display: flex;
  justify-content: center;
  text-align: center;
  margin-top: 4rem;
}

.img-columna {
  width: 30%;
}

```

```

    margin: 0rem 2rem;
}

.img-columna h2 {
    font-size: 2.3rem;
}

.img-columna p {
    font-size: 1.4rem;
}

.img-contenedor img {
    max-width: 260px;
    border-radius: 5px;
}

footer {
    background: radial-gradient(
        circle,
        rgba(44, 115, 201, 1) 0%,
        rgba(10, 56, 112, 1) 100%
    );
    text-align: center;
    color: white;
    font-size: 1.3rem;
    padding: 1rem;
    min-height: 6vh;
    display: flex;
    justify-content: center;
    align-items: center;
}

/* ##### REGISTROS #####
### */

/* ***** Registro docente ***** */

```

```

.contenedor-ventana-registro {
  background-color: rgba(0, 0, 0, 0.8);
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  display: none;
  animation: animacionVentana 0.4s ease-in-out;
}

.registroVentana {
  background-color: white;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  padding: 2rem;
  border-radius: 5px;
  width: 360px;
}

.registroVentana h3 {
  font-size: 2rem;
  text-align: center;
}

.registroVentana p {
  text-align: center;
}

.item-formulario.error input {
  border: 2px solid rgb(243, 100, 100);
}

.item-formulario.exito input {
  border: 2px solid rgb(104, 245, 122);
}

```

```

.item-formulario.error span {
  color: rgb(243, 100, 100);
}

.item-formulario {
  margin-top: 2.3rem;
}

.mensajeRegistroExitoso {
  text-align: center;
  padding: 1rem;
}

.registroVentana select {
  width: 100%;
  border: rgb(219, 219, 219) 2px solid;
  border-radius: 5px;
  padding: 1rem;
  outline: none;
}

.registroVentana select:focus {
  border: rgb(148, 148, 148) 2px solid;
}

.registroVentana label {
  display: block;
  font-size: 1.6rem;
}

.registroVentana input {
  width: 100%;
  border: rgb(219, 219, 219) 2px solid;
  border-radius: 5px;
  padding: 1rem;
}

```

```

input[type='radio'] {
  cursor: pointer;
}

.registroVentana input:focus {
  width: 100%;
  border: rgb(148, 148, 148) 2px solid;
  border-radius: 5px;
  padding: 1rem;
}

.registroVentana button {
  width: 100%;
  margin-top: 2rem;
  padding: 0.8rem 1rem;
}

.radioBtn {
  display: flex;
  justify-content: center;
  align-items: center;
  padding: 1rem;
}

.btn-registro {
  font-size: 1.8rem;
}

.cerrarBtnForm {
  display: block;
  margin-left: 30rem;
  cursor: pointer;
}

.selectNiveles {

```



```

    text-align: center;
}

/* #####
## */
/* ##### *** INTERIOR APLICACIÓN *** #####
## */
/* #####
## */

/* ##### NAVEGACION (NAVBAR) #####
##### */

.navegacion {
    background-color: #cce2f7;
    background: radial-gradient(
        circle,
        rgba(44, 115, 201, 1) 0%,
        rgba(10, 56, 112, 1) 100%
    );
    position: sticky;
    top: 0;
    left: 0;
    width: 100%;
    min-height: 13vh;
    z-index: 10;
}

.navegacion nav {
    width: 80%;
    margin: auto;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 1.65rem;
    padding: 2rem;

```

```

}

.navegacion li {
  cursor: pointer;
}

.item-lista {
  padding: 2rem;
  margin: 1rem 1rem;
  text-align: center;
  list-style: none;
}

.navegacion ul:hover {
  border: 1px solid white;
  color: white;
}

.item-lista a {
  text-decoration: none;
  color: white;
}

.seccion {
  text-align: center;
}

.nombreUsuario {
  text-align: center;
  padding: 1.4rem;
  margin-top: 1.5rem;
  color: rgb(26, 72, 128);
  border-bottom: 1px solid #ccc;
}

```

```

/* ##### SECCION GLOBAL #####
##### */

.seccion h2 {
  font-size: 2rem;
  margin: 2rem;
}

.seccion select {
  margin: 1rem;
  font-size: 1.6rem;
  padding: 1rem;
  border-radius: 5px;
  border: 1px solid lightblue;
  box-shadow: rgba(100, 100, 111, 0.2) 0px 7px 29px 0px;
  outline: none;
}

.seccion input {
  margin: 1rem;
  font-size: 1.6rem;
  padding: 1rem;
  border-radius: 5px;
  border: 1px solid lightblue;
  box-shadow: rgba(100, 100, 111, 0.2) 0px 7px 29px 0px;
  outline: none;
}

.seccion label {
  font-size: 1.6rem;
}

.seccion textarea {
  resize: none;
  font-family: 'Poppins', sans-serif;
  border: 1px solid lightblue;

```

```

outline: none;
border-radius: 5px;
background-color: white;
box-shadow: rgba(100, 100, 111, 0.2) 0px 7px 29px 0px;
padding: 0.3rem;
}

.seccion input[type='file'] {
  cursor: pointer;
}

.subTituloNombre {
  font-size: 1.5rem;
  padding: 1rem;
}

/* ##### SECCION 1 #####
### */

.seccion1 {
  background-color: white;
  min-height: 20vh;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  display: flex;
  padding: 2rem;
}

.selectNiveles {
  display: flex;
  flex-direction: column;
  width: 340px;
  margin: 0 auto;
}

```

```

.mensajeCambioDeNivel {
  font-size: 1.4rem;
  padding: 1rem;
}

/* Seccion 1 alumno */
.seccion1 .resultadoBusqueda {
  margin: 2rem;
}

.seccion1 table {
  margin: 0 auto;
}

.seccion1 .nombreProfe {
  color: rgb(26, 72, 128);
}

.seccion1 p {
  font-size: 1.7rem;
}

.seccion1 .resultadoBusqueda p {
  font-size: 1.6rem;
}

.seccion1 .resultadoBusqueda h2 {
  margin: 1rem;
  color: #3577b4;
}

.seccion1 .resultadoBusqueda img {
  width: 770px;
}

.seccion1 .resultadoBusqueda img {
  width: 770px;
}

```

```

}

.btnBusqueda {
  padding: 0.5rem 1.4rem;
  font-size: 1.8rem;
  width: 150px;
  display: block;
  margin: 0 auto;
}

/* ##### SECCIÓN 2 #####
### */

.seccion2 {
  width: 35%;
  margin: 0 auto;
  padding: 2rem;
  display: flex;
  flex-direction: column;
}

.seccion2 .divMensajes {
  font-size: 1.55rem;
}

.inputBusqueda {
  width: 40%;
}

.agregarEjercicio {
  display: flex;
  flex-direction: column;
}

/* ##### SECCION 3 #####
### */

```

```

.seccion3 {
  text-align: center;
  width: 900px;
  margin: 0 auto;
  margin-bottom: 3rem;
  width: 100%;
}

.seccion3 p {
  font-size: 1.6rem;
  text-align: center;
  margin: 1rem;
}

.tablaDevoluciones {
  margin: 0 auto;
}

/* ##### SECCION 4 #####
### */

.seccion4 {
  margin-bottom: 10rem;
}

.seccion4 p {
  font-size: 1.6rem;
}

/* ##### CLASES DE MOSTRAR Y OCULTAR #####
##### */

.mostrar-ventana {
  display: block;
}

.ocultar-ventana {

```

```

    display: none;
}

/* ##### ESTILO TABLA GLOBAL #####
##### */

table {
    border-collapse: collapse;
    width: 60%;
    border: 1px solid rgb(0, 0, 0);
}

th,
td {
    text-align: left;
    padding: 8px;
}

th {
    font-size: 1.8rem;
    text-align: center;
    border: 1px solid rgb(0, 0, 0);
}

td {
    font-size: 1.6rem;
    text-align: center;
    border: 1px solid rgb(0, 0, 0);
}

tr:nth-child(even) {
    background-color: #ececec;
}

tr:nth-child(odd) {
    background-color: rgb(36, 101, 180);
}

```



```

    color: white;
}

th {
    background-color: rgb(36, 101, 180);
    color: white;
}

table img {
    width: 350px;
}

@media screen and (max-width: 1100px) {
    .seccion2 {
        width: 70%;
    }

    .inputBusqueda {
        width: 60%;
    }
}

@keyframes animacionVentana {
    from {
        opacity: 0%;
    }
    to {
        opacity: 100%;
    }
} ;

```