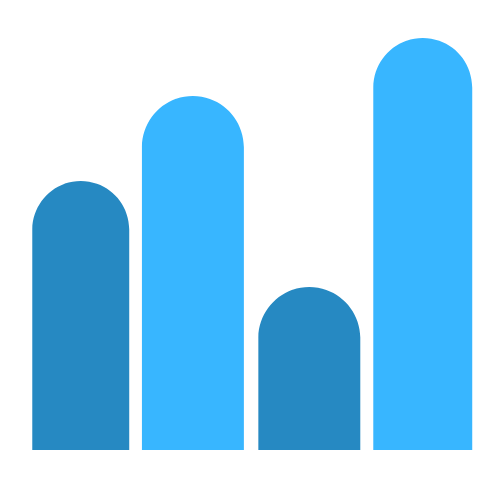
GAMIFI-K FBX



Ilerna

Desenvolupament d’Aplicacions Web 2 (DAW2)

Francesc Vargas

Biel Nadal

Xavier Tomàs

Index

[**DISEÑO Y CODIFICACIÓN**](#_b990jdj1fmpo) **5**

[ESTRUCTURA EN ANGULAR](#_9pf3gadgyrst) 5

[ESTRUCTURA DE LA CARPETA VISTAS](#_pg3jeie1tuo) 6

[Alumnos](#_9o1aa9jd8qvu) 6

[Dark-mode](#_u4j2f5ufu6fc) 6

[Identificación](#_yylti01lgdrh) 6

[Landing](#_f0b11j63hedz) 7

[Profesores](#_op86iv5ftbwe) 7

[RESTO DE ARCHIVOS](#_jxpwhhroetl) 8

[app-routing.module.ts](#_f72fsrkiqmoj) 8

[app.module.ts](#_j4nx75q0g185) 8

[ESTRUCTURA DEL RESTO DE CARPETAS](#_r08glwsw36hk) 9

[Assets](#_d41xhdk0oky3) 9

[Environments](#_atynw1iomf8j) 9

[Scripts](#_yurnte9hvs5p) 9

[Services](#_d7uxtn2an1hu) 9

[ESTRUCTURA CARPETA SERVICES](#_rx3mbf66ocii) 10

[auth](#_nhqwze3f5g87) 10

[auth.service.ts](#_yan6yervxfm5) 10

[token.service.ts](#_kdzbmrdqggpz) 10

[guards](#_21j2hhudfwp) 11

[auth-guard.service.ts](#_5jxpi18o95nq) 11

[role-guard.service.ts](#_2h47tpv77bnn) 11

[interceptors](#_fi300x8iop99) 12

[interceptor.service.ts](#_iidgl5sz4ows) 12

[loading-interceptor.service.ts](#_1k42ypkv80nd) 12

[REST API Y PETICIONES HTTP](#_8eyqdpad9x7y) 13

[ESTRUCTURA EN PHP](#_jpxx6ulaqnjw) 15

[alumnos](#_bmhpaksbh7ej) 15

[Conexión](#_a5uendogcbys) 16

[evaluaciones](#_78f4m1ji8kmf) 16

[identificación](#_z9uf4267wq9) 16

[phpFunctions](#_147vzpemwpah) 17

[profesores](#_3rxccha0nvjw) 17

[rankings](#_1yetgcbhzlf) 17

[alumnos](#_dq6x3skljnp6) 18

[entregas](#_664kugd5oe0k) 18

[profesores](#_qkwmu8gdwqwh) 18

[tokenJWT](#_nzlslj9mvauz) 18

[**LOGIN REGISTRO**](#_6rzgcf50bai6) **19**

[IDENTIFICACIÓN](#_6x4l514wpdrd) 19

[LOGIN ALUMNO](#_jgmdhdk0zpn6) 20

[REGISTRO ALUMNO](#_veenkpc5u2ty) 20

[LOGIN PROFESOR](#_tym6nb8chyeb) 21

[REGISTRO PROFESOR](#_mba11cx7u375) 22

[PERFIL](#_1yoqc4y76uh5) 23

[Perfil](#_13ych824pz4u) 23

[cerrar sesión](#_mg1vqytef0og) 27

[**DASHBOARD PARA PROFESORES**](#_c51tj3vaeuut) **28**

[RANKINGS](#_agpd1q4vvh3q) 29

[CONSULTAR RANKINGS](#_ezmh3zp67jl) 29

[GESTIÓN DE LOS USUARIOS DEL RANKING](#_sjweqzykr8ny) 30

[BUSCADOR](#_21mb97ixugmk) 30

[VER DETALLES DE ALUMNO](#_hrcgotc2vmha) 30

[ELIMINAR ALUMNO DEL RANKING](#_woz5r2enfs5b) 31

[ALUMNOS PENDIENTES](#_66wsyb5raaoo) 31

[OPCIONES DEL RANKING](#_mxwua4vkof6j) 32

[EDITAR RANKINGS](#_6lcbx8q6gi9v) 32

[ELIMINAR RANKINGS](#_40qn6da7bzu9) 32

[CREAR RANKINGS](#_99drflp8al2z) 33

[PRÁCTICAS/ENTREGAS](#_88x15yfnjd04) 34

[CONSULTAR PRÁCTICAS](#_lnuettcyo42u) 34

[GESTIÓN DE LOS USUARIOS DE LAS PRÁCTICAS](#_966r47n0bzt2) 35

[BUSCADOR](#_4pn6ccw11n28) 35

[VER DETALLES](#_sgjmjrrpksva) 35

[PUNTUAR ALUMNOS](#_2yty1km7bvb7) 36

[OPCIONES DE PRÁCTICAS](#_sprezos6l219) 36

[BUSCADOR DE PRÁCTICAS](#_x17i9zy8lb95) 36

[CREAR PRÁCTICAS/ENTREGAS](#_5ru1mgg1waqr) 37

[EVALUACIONES](#_jbrhkx86ckg8) 38

[CONSULTA DE EVALUACIONES](#_5i98mep07aa1) 38

[OPCIONES DE EVALUACIONES](#_xixcsoocaczu) 39

[ELIMINAR EVALUACIONES](#_w2chif3dsbpb) 39

[FILTROS DE EVALUACIONES](#_uymwhrhrp7cv) 40

[**DASHBOARD ALUMNO**](#_kw8r13akamvh) **41**

[LISTADO RANKINGS](#_cs9eq0v9ujgf) 41

[UNIRSE A RANKING](#_ovrr8f2d7fx1) 43

[RANKING SELECCIONADO](#_bgnt6z8kk5j4) 45

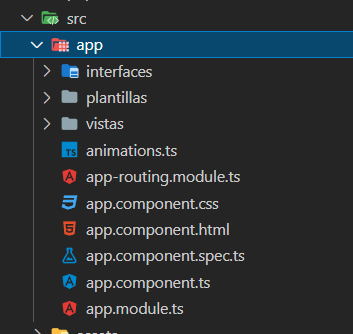
[PUNTUAR COMPAÑEROS](#_jmeircu39wzx) 48

# DISEÑO Y CODIFICACIÓN

Este programa está construido con Angular y PHP.

Angular se encarga del frontend y PHP se encarga del backend.  
  
Para gestionar los paquetes de Angular se utiliza NPM, con el que se instalarán las dependencias, librerías y demás recursos de la aplicación. Es importante hacer npm install para instalar las dependencias y descargar node\_modules.

Para encontrar los archivos principales para comenzar la edición de código de Angular, entramos en la carpeta src > app:



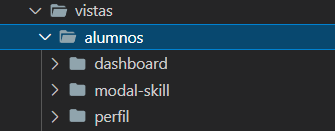
## ESTRUCTURA EN ANGULAR

* **Interfaces:** archivos que definen las interfaces, como alumno y sus propiedades nick, email, nombre, apellidos…
* **Plantillas:** componentes que nos pueden servir para reutilizar en el resto de componentes, como el header, el footer…
* **Vistas:** esta carpeta es la más importante, aquí estarán todos los componentes principales.

## ESTRUCTURA DE LA CARPETA VISTAS

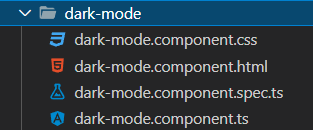
### Alumnos

* + dentro de alumnos tendremos su dashboard, el modal de skills cuando vayan a evaluarse entre ellos y su perfil:



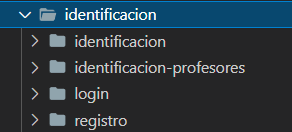
### Dark-mode

* + consta de una única carpeta, con los archivos correspondientes a typescript, html… Se utiliza para el botón y la gestión del cambio de estilos a uno más oscuro:



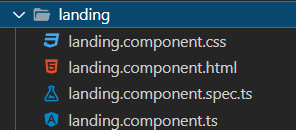
### Identificación

* + para la identificación tenemos la identificación (a secas) de alumnos y la identificación de profesores. Estas dos carpetas se utilizan para construir visualmente los tabuladores de bootstrap y dentro de cada identificación tendremos el componente de login y registro:



### Landing

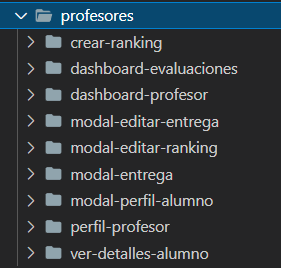
* + carpeta única con la pantalla principal de la web:



### Profesores

Carpetas en relación a los profesores.

* + **crear-ranking:** modal para la creación del ranking.
  + **dashboard-evaluaciones:** componente principal donde consultar las evaluaciones.
  + **dashboard-profesor:** componente principal donde consultar y gestionar los rankings.
  + **modal-editar-entrega:** modal para la edición de entregas.
  + **modal-entrega:** modal para la creación de entregas/prácticas.
  + **perfil-profesor:** el perfil del profesor.
  + **ver-detalles-alumno:** modal para consultar los detalles del alumno.



## 

## RESTO DE ARCHIVOS

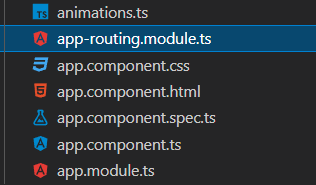
Los archivos más importantes son:

### app-routing.module.ts

* + se encarga de especificar las rutas de cada componente y así poder acceder a ellos.

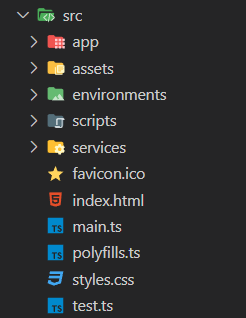
### app.module.ts

* + gestiona cada componente y sus librerías.



## ESTRUCTURA DEL RESTO DE CARPETAS

También existen otras carpetas, fuera de app, que son importantes:



### Assets

* + en esta carpeta se guardarán las imágenes y recursos que necesitemos para la aplicación.

### Environments

* + carpeta generada por angular.

### Scripts

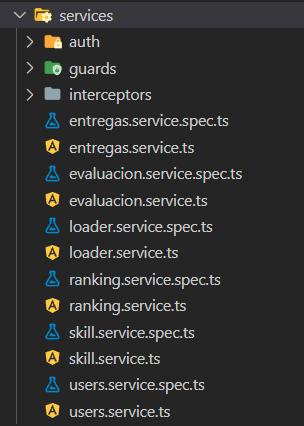
* + carpeta donde guardar scripts, como la preview de los avatares en los perfiles en javascript.

### Services

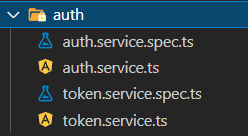
* + esta carpeta es esencial. Aquí dentro tendremos los servicios, que nos proporcionarán los datos que consumimos del PHP.

## ESTRUCTURA CARPETA SERVICES

Como hemos dicho antes, services es una carpeta importante, aquí gestionamos las consultas http y sin estos archivos no tendríamos conectividad con la api.



### auth



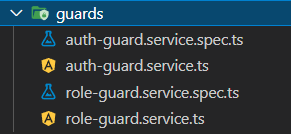
#### auth.service.ts

* + gestiona el login y el registro del alumno y del profesor.
  + guarda en localstorage la sesión del usuario.
  + controla que haya autentificación.
  + controla el rol del usuario (si es alumno o profesor) para saber si el user que ha iniciar sesión puede entrar en ciertos componentes o no.
  + controla la semana en la que estamos y así saber si restaurar los puntos semanales del alumno que ha iniciado sesión (100 por semana).

#### token.service.ts

* + gestiona el token del usuario: lo obtiene, lo guarda, comprueba su expiración.
  + para comprobar la expiración del token se utiliza esta librería: <https://github.com/auth0/angular2-jwt>

### guards



Estos servicios implementan CanActivate. CanActivate es una clase que puede especificar en app-routing.module.ts y retorna true o false dependiendo de lo que especifiquemos dentro de estos servicios. Por ejemplo, en auth-guard, controlamos que el usuario haya iniciado sesión, si no es así, no nos dejará utilizar la ruta.

#### auth-guard.service.ts

* + controla la sesión del usuario, si no ha iniciado sesión no nos debería dejar entrar en las rutas de la web.

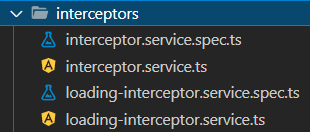
#### role-guard.service.ts

* + controla si el usuario es alumno o profesor.
  + si es alumno e intenta entrar en un componente que es de profesores, se le redireccionará al componente de alumnos, ejemplo: alumno intenta entrar a dashboard-profesores, en vez de eso, se le redireccionará a dashboard.

### 

### 

### interceptors



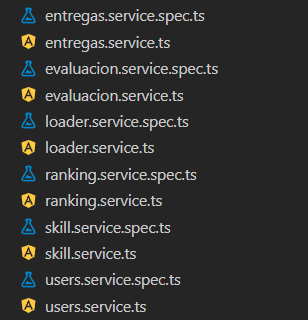
Estos servicios interceptan cualquier petición http que ocurra en la web. Son muy útiles para controlar la sesión del usuario o implementar tiempos de carga/loading para esperar por las peticiones.

#### interceptor.service.ts

* + controla que exista token y si está expirado.

#### loading-interceptor.service.ts

* + por cada petición http, hace que muestre una imagen, icono o animación de carga/loading. En versiones previas de la página teníamos implementado un loading, pero se ha decidido quitarlo. A pesar de funcionar, no ayudaba visualmente.



El resto de servicios (salvo loader.service.ts que se encargaba de mostrar o esconder el loader) se utilizan para la obtención de datos.

## REST API Y PETICIONES HTTP

Cuando hablamos de **REST API** o **RESTful API**, de lo que hablamos es de una arquitectura o más bien de unas limitaciones, unos **endpoints**.

Hacemos una solicitud y obtenemos un **JSON**. No tiene por qué ser JSON, también puede ser HTML, XML, Raw text… Pero en esta aplicación utilizaremos JSON, ya que es un formato popular y muy fácil de utilizar.  
  
Para hacer estas **solicitudes HTTP**, tenemos diversos métodos:

* **GET:** pedirle algo a la API, ejemplo “/alumno/3” nos debería obtener el alumno con id de alumno 3.
* **POST:** para la creación de contenido, aunque también se puede utilizar para actualizar, consultar, eliminar… POST, a comparación con GET, no necesita metadatos o parametrizar la información por la URL.
* **PUT:** reemplaza contenido por otro.
* **DELETE:** para eliminar contenido.

En cada uno de los archivos tenemos una estructura similar:

* entregas.service.ts
* evaluacion.service.ts
* ranking.service.ts
* skill.service.ts
* users.service.ts



Imagen de ejemplo del archivo users.service.ts

En cada función hacemos una petición HTTP, ya sea post, get, put o delete, y dependiendo de eso, le pasaremos un parámetro o convertiremos los datos a JSON para enviarlos con POST.

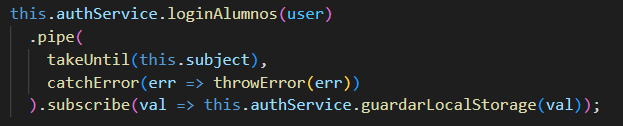
Esto nos devuelve un observable al que podemos subscribirnos.

* Observable: nos proporciona un flujo de datos, que pueden llegar ahora o más tarde. Estos no harán nada a menos que indiquemos que hay que seguirle la pista con una subscripción.
* Subscribe: con esto consumimos del observable.

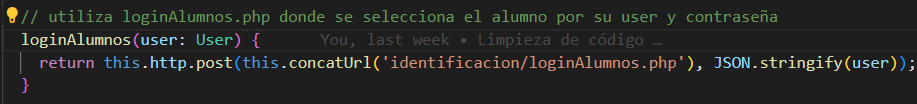
Estas herramientas nos adentran en el mundo asíncrono y de la programación reactiva, es un tema complejo, que difícilmente se entenderá a la primera, que está en constante es necesario empaparse con mucha práctica y lectura.

Recursos:

* <https://www.learnrxjs.io/learn-rxjs/concepts/rxjs-primer>
* <https://angular.io/guide/observables-in-angular#observables-in-angular>
* <https://rxjs.dev/guide/overview>



Con los datos que obtenemos, podemos hacer lo que queramos, como en este ejemplo en el que guardo la sesión del usuario en localstorage que obtenemos de “this.authService.loginAlumnos(user)”:



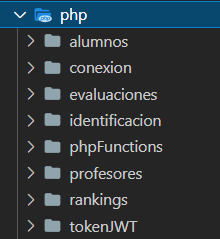
## ESTRUCTURA EN PHP

Fuera de las carpetas de src, que es donde está angular, tenemos una llamada php y dentro diversas carpetas con todo lo relacionado con la API.

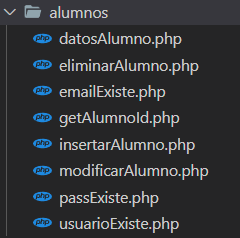
En futuras versiones, probablemente desechemos la manera en la que hemos realizado esta estructura, ya que no es cómoda y no es segura. Lo ideal sería utilizar un framework para PHP con el que podamos tener una lista de peticiones REST, con middlewares para procesar y sanitizar los parámetros y clases en PHP para no tener tantos archivos individuales desorganizados.

Recursos:

* <https://laravel.com/>
* <https://www.slimframework.com/docs/v4/>



### alumnos



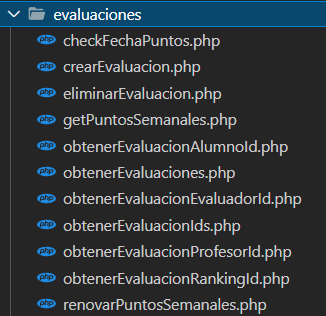
* + Obtener todos alumnos.
  + Obtener alumnos específicos mediante su ID.
  + Modificar alumnos.
  + Eliminar alumnos.
  + Comprobar la existencia de emails o nicknames.

### Conexión



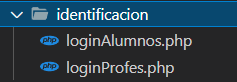
* + Clase BD para gestionar el servidor, usuario, base de datos.
  + Iniciamos la conexión y obtenemos una conexión mysqli.

### evaluaciones



* + Crear o insertar evaluaciones en base de datos.
  + Obtener evaluaciones, de alumnos evaluados, de evaluadores, mediante la id del profesor asignado al ranking o la id del ranking.
  + Eliminar evaluaciones.
  + Controlar los puntos de las evaluaciones, la fecha en la que se hizo la última evaluación, la renovación de puntos en base de datos y su obtención.

### identificación



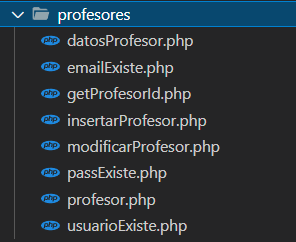
* + Pasamos el nombre de usuario y la contraseña (encriptada), buscamos que esté en la base de datos y si el resultado es correcto devolvemos la sesión del usuario.
  + Tanto del alumno como del profesor. En una primera versión de la aplicación teníamos el login unido y te buscaba tanto en la tabla de alumnos como de profesores (para tener un solo archivo de login), pero decidimos separarlo por comodidad.

### phpFunctions



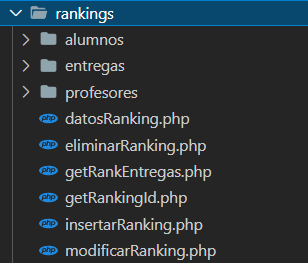
* + Se encarga de encriptar las contraseñas de los usuarios.

### profesores



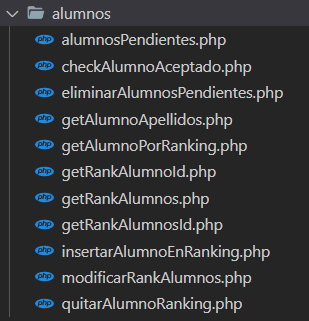
* + Igual que el de alumnos, se encarga de obtener todos los profesores o utilizar su id para buscarlos.
  + Modificar sus datos.
  + Comprobar la existencia de usuarios.

### rankings



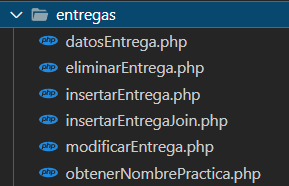
* + CRUD de rankings. Obtener entregas, obtener entregas por su id.
  + Eliminar entregas, modificar o insertar nuevas entregas.

##### alumnos



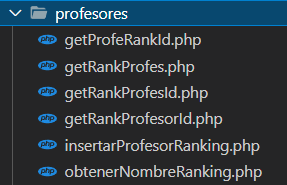
* + - En esta carpeta se controlan los alumnos en relación a los rankings. Obtener alumnos mediante el ranking, comprobar los pendientes de aceptar, quitar del ranking…

##### entregas



* + - Obtener entregas, eliminarlas, insertar nuevas entregas…

##### profesores



* + - Obtener profesores en relación al ranking, insertar nuevos profesores, comprobar la existencia de rankings con el mismo nombre…

### tokenJWT



* Genera un token con la ayuda de la librería:

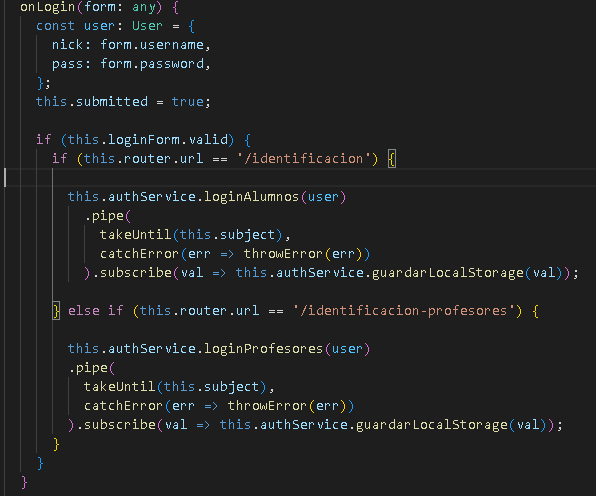
<https://github.com/firebase/php-jwt>

* Aquí podemos especificar el tiempo de expiración. Está puesto a 7200 para que la sesión se cierre a las 2 horas de haber iniciado sesión.

# LOGIN REGISTRO

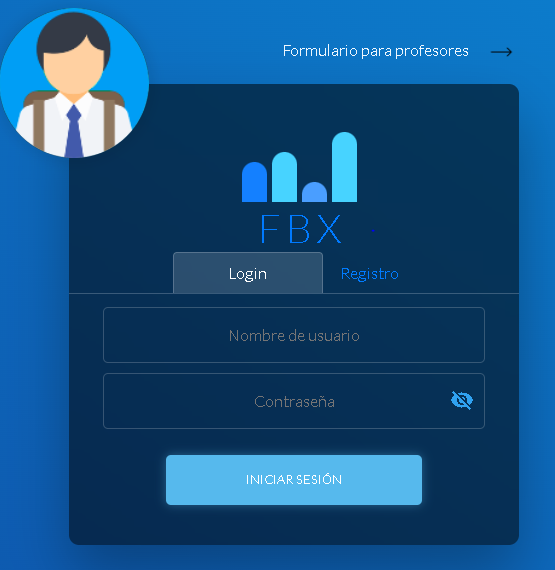
## IDENTIFICACIÓN

En la IDENTIFICACIÓN la función onLogin cojemos los datos recogidos en el formulario de inicio de sesión y lo que primero miramos es si form(loginForm) sea válido es decir si contiene valores dentro de él. posteriormente se comprobara en que url esta en la (/identificacion) se realizará la comprobación y entrara a la cuenta de alumno, (/identificacion-profesores) se realizará la comprobación y entrara a la cuenta de alumno.



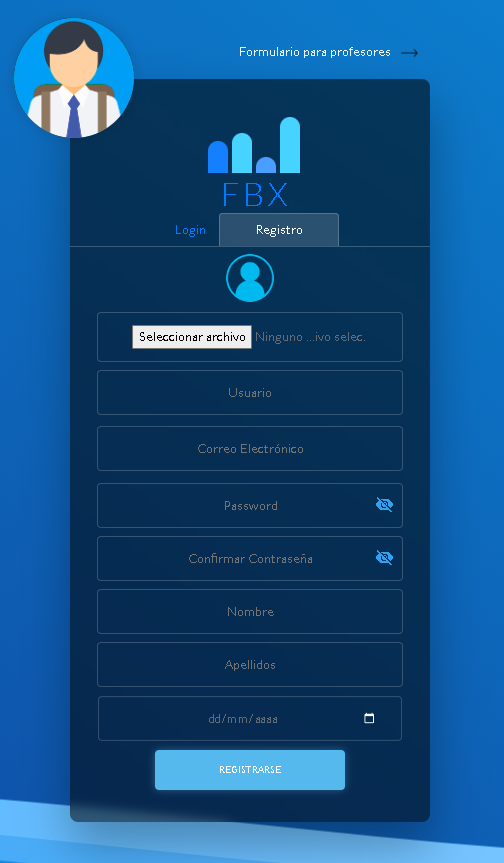
#### LOGIN ALUMNO

En la parte del centro de la pantalla verá el login de alumnos donde podremos iniciar sesión. Donde poniendo el nombre del usuario y su contraseña podrá iniciar sesión.



#### REGISTRO ALUMNO



Al pulsar el botón registro en la parte de alumno saldrá un formulario de registro a rellenar para poder registrarse.

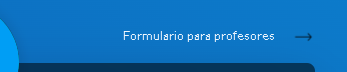
## 

## 

## 

#### 

#### LOGIN PROFESOR

para poder entrar al formulario de profesores tendremos de pulsar. 

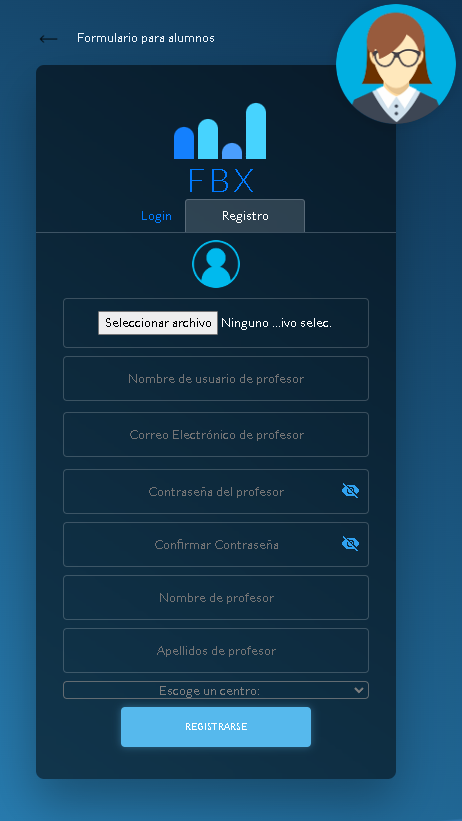
Donde llevara a una página verá el login de profesores donde podremos iniciar sesión. Donde poniendo el nombre del usuario y su contraseña podrá iniciar sesión.



## 

#### REGISTRO PROFESOR



Al pulsar el botón registro en la parte de profesores saldrá un formulario de registro a rellenar para poder registrarse.

# 

## 

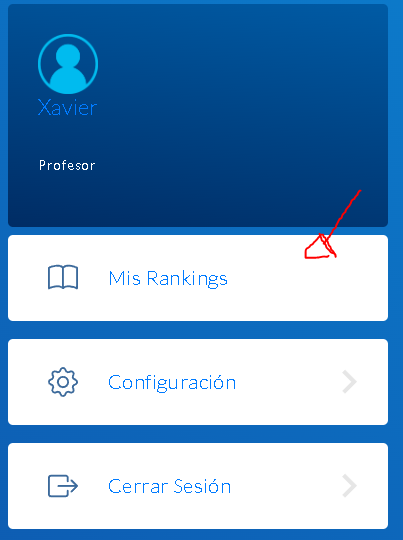
## 

## 

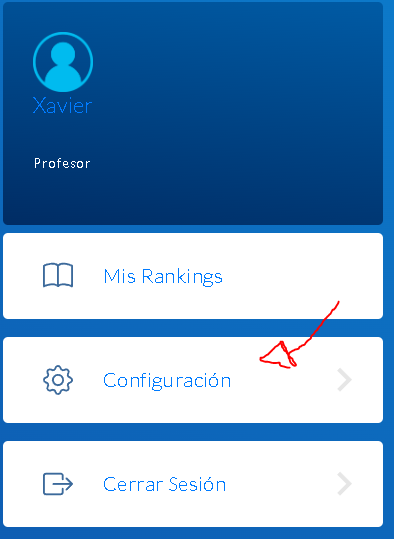
## PERFIL

#### Perfil

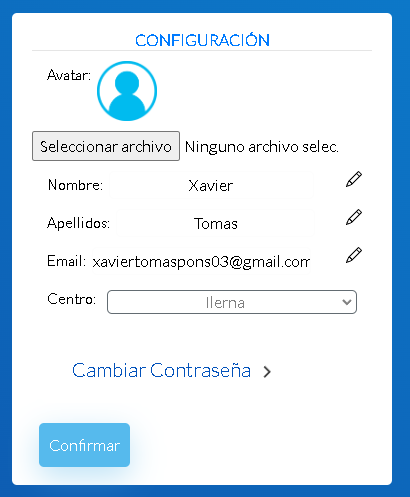
En el perfil encontraremos tres opciones la primera (Mis Rankings) se redirigirá al dashboard en cuestión del profesor o alumno.



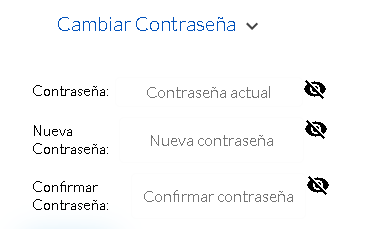
En la segunda opción podremos abrir la configuración de dicha cuenta.



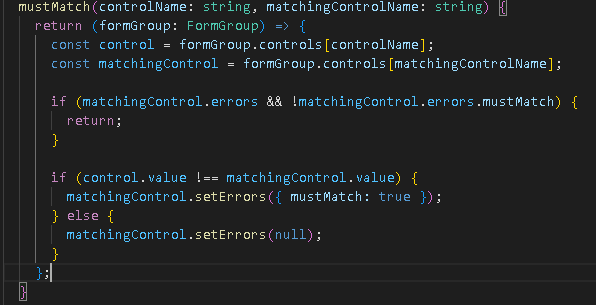
Al pulsar en el lápiz podremos cambiar los datos del usuario en cuestión: avatar, nombre, apellidos, email, centro y cambiar la contraseña.



Al cambiar la contraseña tendremos que poner la contraseña actual y posteriormente tendremos que poner la contraseña nueva dos veces para comprobar si son las mismas para más seguridad.



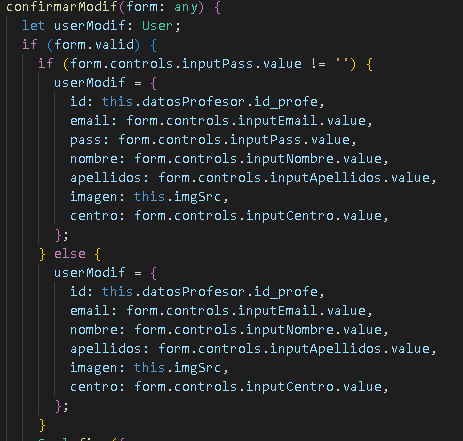
Función para controlar que los campos de password sean iguales.



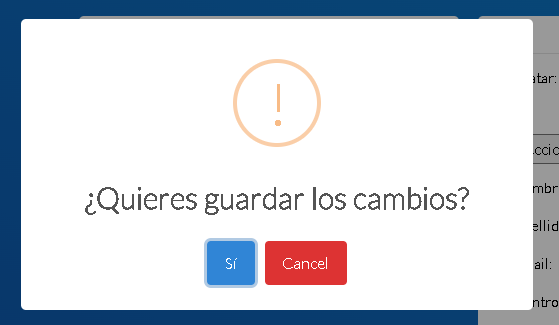
Al pulsar el botón confirmar cambiaremos los nuevos datos por los viejos.

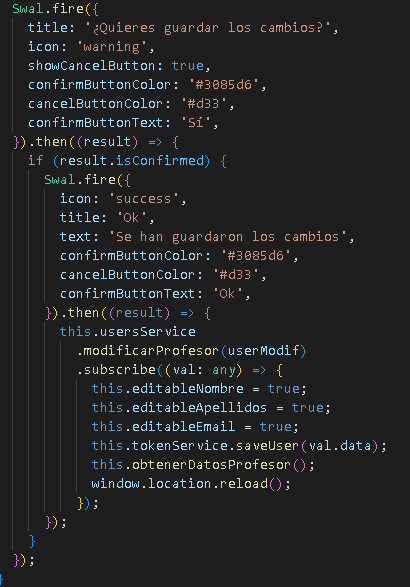


Función para modificar los datos del usuario en el primer if sirbe si hay una imagen y el esle es para cuando no haya una foto para modificar.



Posteriormente saldrá un sweetalert donde tendras de pulsar en ' si ' si quiere modificar los datos anteriores del usuario en cuestión, sino no se modificarán si continuaran los datos anteriores.





#### cerrar sesión

Al pulsar en cerrar sesión aparecerá un div donde tendremos dos opciones

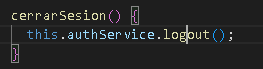


en el div tendremos dos opciones cerrar sesión o cancelar.

si pulsamos cerrar sesión llamara a la función (cerrarSesion()), en cambio si puslamos en cancelar se cerrará el div.

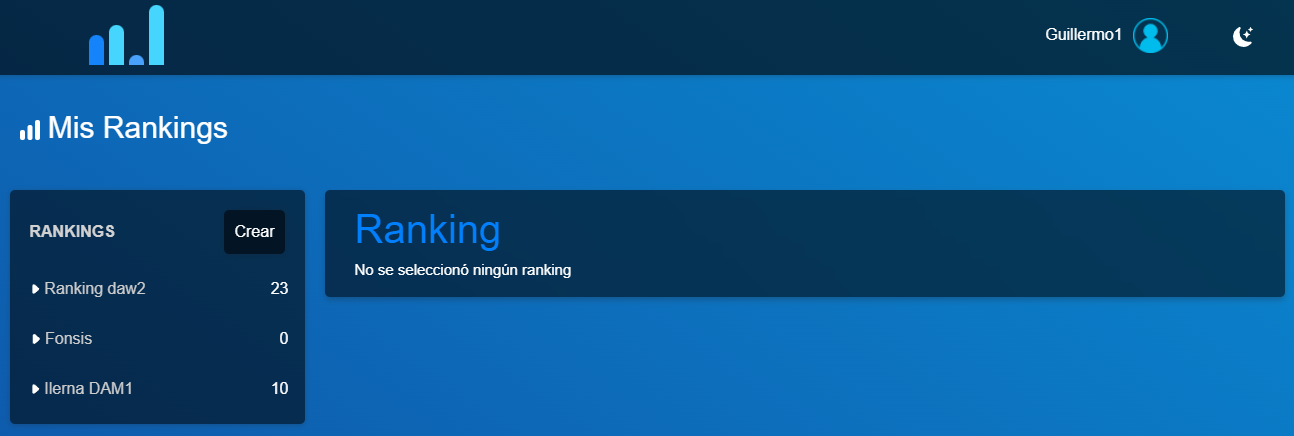


Función cerrarSesion() sirve para borrar los datos del token.



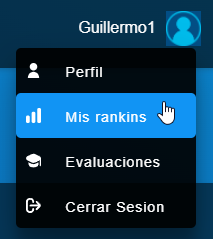
# DASHBOARD PARA PROFESORES

Nada más iniciar sesión como profesor, la página se redireccionará automáticamente al dashboard de profesores, donde verán este contenido:



Aquí un profesor puede ver, en el header, el logotipo de la página, su nick de usuario, su avatar y un botón para cambiar la visualización a modo oscuro.

Si coloca el cursor sobre su nombre/avatar, se desplegará automáticamente una lista de enlaces, que le servirá para navegar por la web o cerrar sesión:



## 

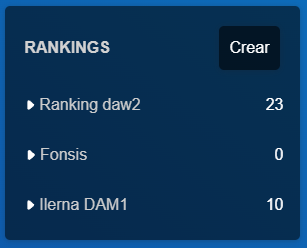
## 

## 

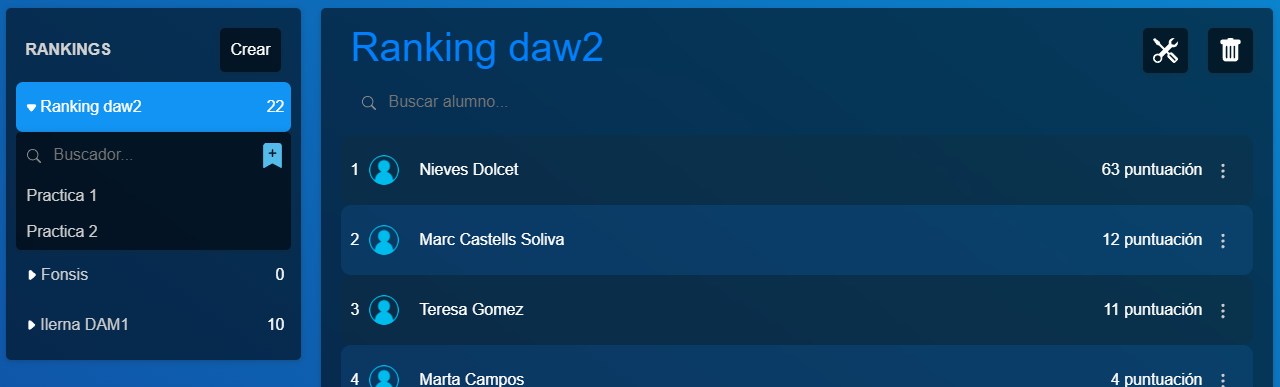
## 

## RANKINGS

### CONSULTAR RANKINGS

En la parte izquierda verá una lista con todos los rankings creados previamente por ese mismo profesor que ha iniciado sesión, también verá un botón para crear un nuevo ranking y al lado la cantidad de alumnos que hay en ese momento dentro del ranking:   


Se ve un mensaje que indica que no se ha seleccionado ningún ranking. Cuando pulsamos en la lista sobre alguno de ellos se verán los alumnos, ordenados de mayor a menor puntuación:

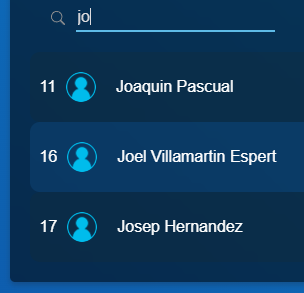


### 

### GESTIÓN DE LOS USUARIOS DEL RANKING

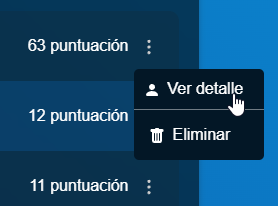
#### BUSCADOR

Encima de la lista de alumnos se encuentra un buscador, tiene en cuenta tanto el nombre como el apellido:

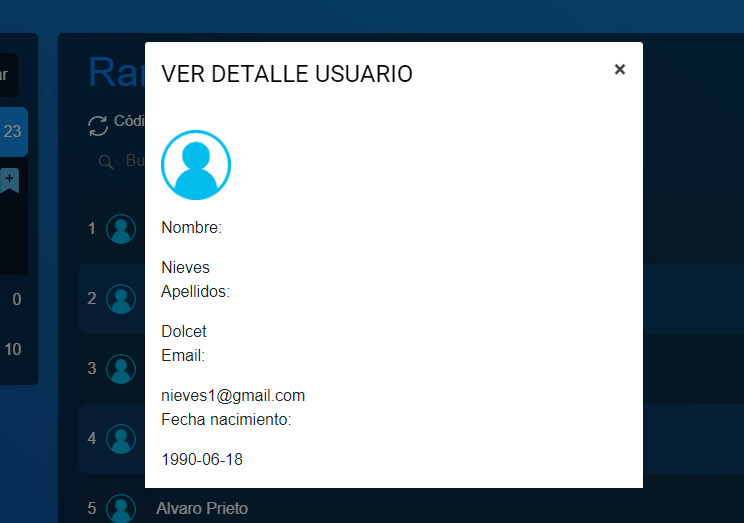


#### VER DETALLES DE ALUMNO

En los rankings, podemos ver el detalle del alumno pulsando primero en el desplegable con los tres puntitos:



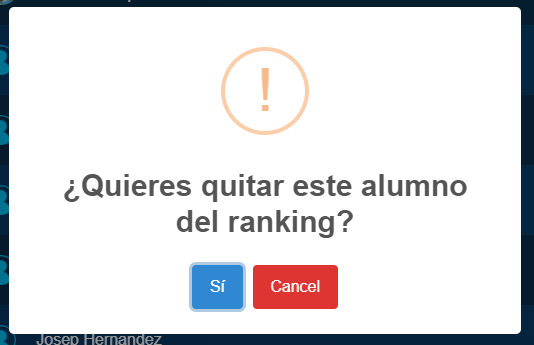
Al pulsar en “Ver detalle” nos aparecerá un modal con más datos:



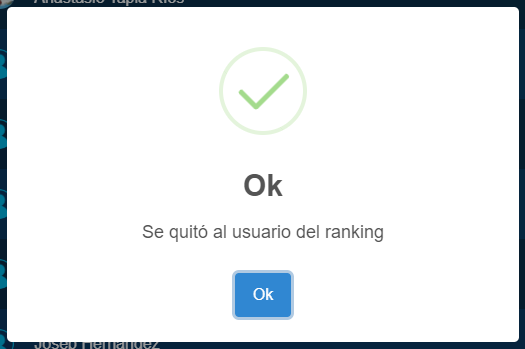
#### 

#### ELIMINAR ALUMNO DEL RANKING

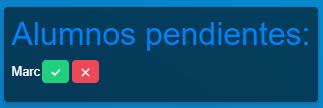
En cambio, al pulsar en eliminar, nos saltará un Sweetalert con un mensaje de advertencia:



Al pulsar en sí, confirmaremos la eliminación del usuario del ranking en el que estamos:



#### ALUMNOS PENDIENTES

Si un alumno se une a un ranking, el profesor puede comprobar la lista de alumnos en espera y poder aceptarles o denegarles cuando vaya a pulsar en el botón del ranking específico:  


Si lo acepta, se actualizará la página y se verá el alumno nuevo reflejado en el nuevo ranking. Se le asignarán las prácticas que existan en el ranking a ese alumno que se acaba de unir:  


### 

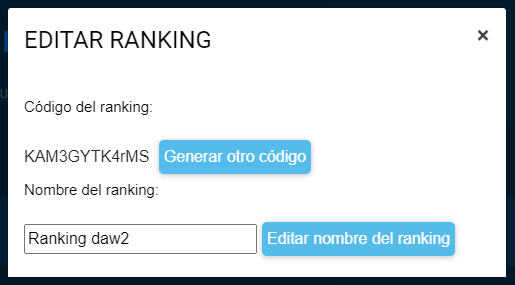
### OPCIONES DEL RANKING

Modificar el título o eliminar el ranking:



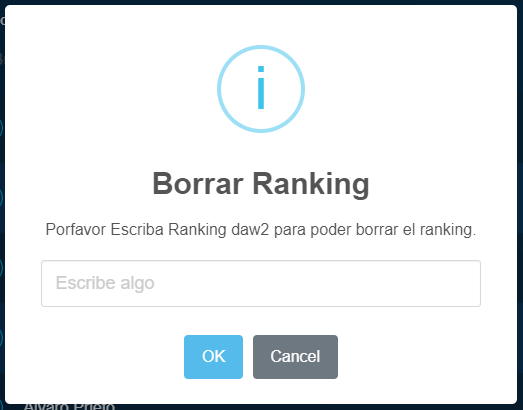
#### EDITAR RANKINGS

Al pulsar en el botón de modificación nos saltará un modal, dentro veremos dos opciones, generar un nuevo código de ranking o editar el nombre del ranking:



Cuando se genera otro código, este cambio no afectará a los alumnos que ya estén unidos al ranking, sin embargo, nuevos alumnos que quieran unirse tendrán que utilizar el nuevo código, el antiguo dejará de servir.

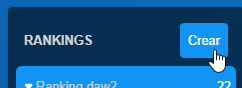
#### ELIMINAR RANKINGS

Y si pulsamos en el botón de eliminar ranking, también nos aparecerá un modal, si queremos confirmar la eliminación del ranking tendremos que escribir el nombre del ranking:  
  


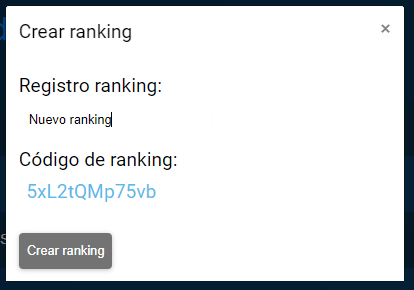
#### 

#### CREAR RANKINGS

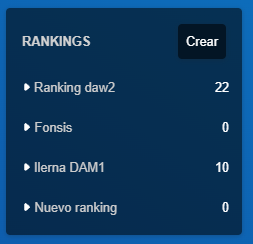
Si queremos crear un ranking, podemos pulsar en el botón Crear que está encima de las listas de rankings:



Se abrirá un modal donde especificar el nombre y veremos un código que se genera automáticamente:



Se actualizará la página y veremos el cambio reflejado en la lista:



# 

## PRÁCTICAS/ENTREGAS

### CONSULTAR PRÁCTICAS

Además, en la lista de rankings, se despliega otra lista, pero de entregas/prácticas que contiene dicho ranking. Podemos pulsar en cualquier práctica para ver su contenido:



Aquí se verán los alumnos ordenados alfabéticamente por su apellido y al lado derecho tendremos una casilla para introducir su puntuación.

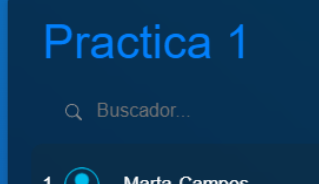
Si queremos volver al ranking y no a su entrega, podemos volver a pulsar sobre el nombre del Ranking.

### 

### GESTIÓN DE LOS USUARIOS DE LAS PRÁCTICAS

#### BUSCADOR

Al igual que con los rankings, también se pueden buscar alumnos dentro de las prácticas:

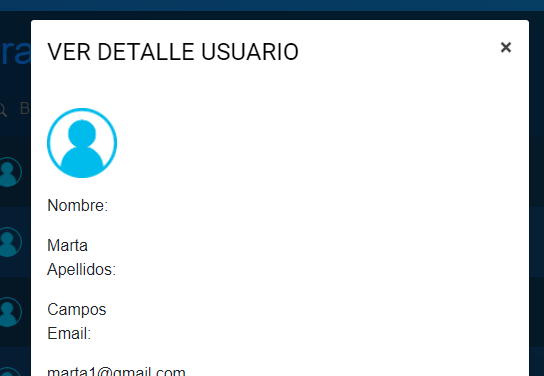


#### VER DETALLES

A diferencia de los rankings, al pulsar en los tres puntitos, aquí tan solo podremos ver su detalle, no está la opción de eliminar del ranking:



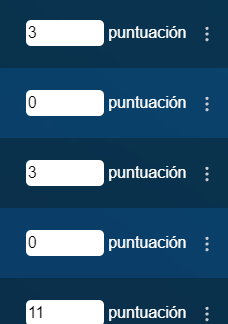
Se verá su detalle:



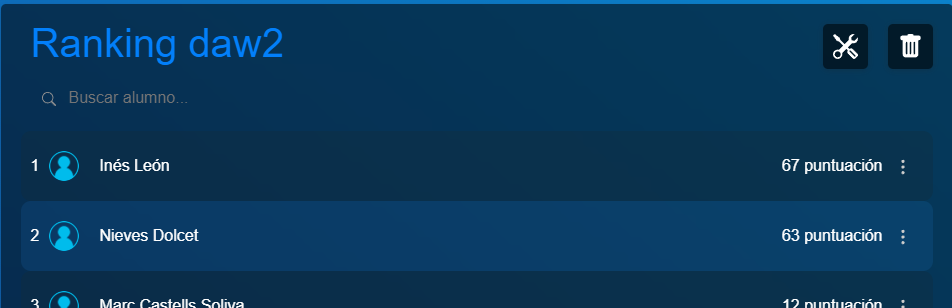
#### 

#### PUNTUAR ALUMNOS

Para puntuar alumnos, podemos escribir una o más puntuaciones a la vez. Al pulsar enter, se insertarán la puntuación o puntuaciones en la base de datos:



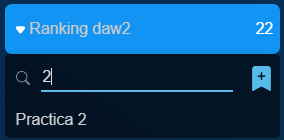
Todas estas puntuaciones de cada práctica/entrega se verán reflejadas al volver al ranking:



### OPCIONES DE PRÁCTICAS

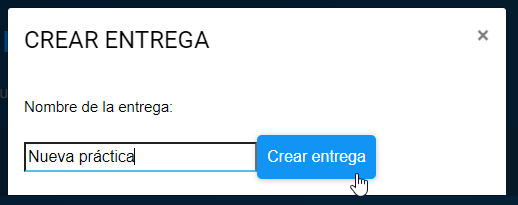
#### BUSCADOR DE PRÁCTICAS

En el desplegable de cada ranking, veremos sus prácticas, un buscador y un botón para agregar una nueva práctica a ese Ranking:



#### CREAR PRÁCTICAS/ENTREGAS

Al pulsar sobre el botón de crear se nos abrirá un modal donde especificar el nombre de la práctica/entrega:



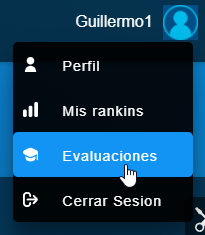
Una vez pulsemos en crear entrega, podremos ver reflejado el cambio volviendo a pulsar sobre el ranking donde hemos creado dicha entrega:



Dentro estarán todos los alumnos que hay en el ranking.

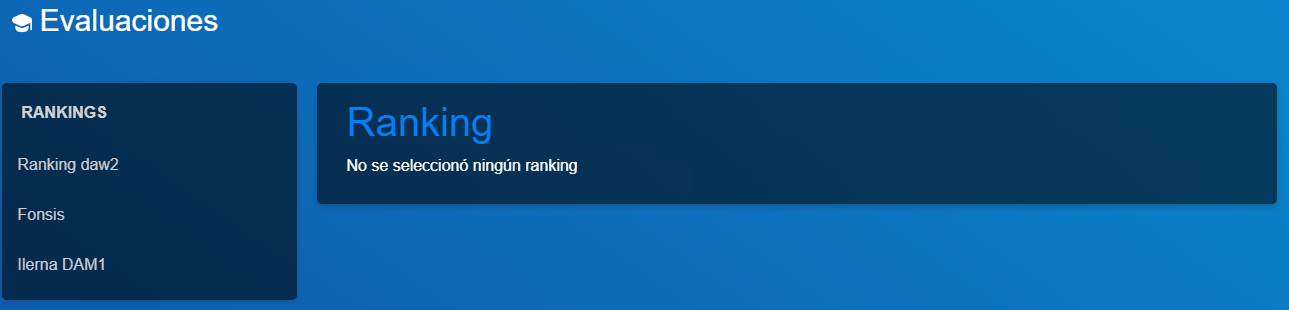
## EVALUACIONES

El profesor también tiene otro dashboard en el que puede consultar todas las evaluaciones que los alumnos se hacen entre ellos. Puede viajar a dicha página a través del desplegable de su nick/avatar:

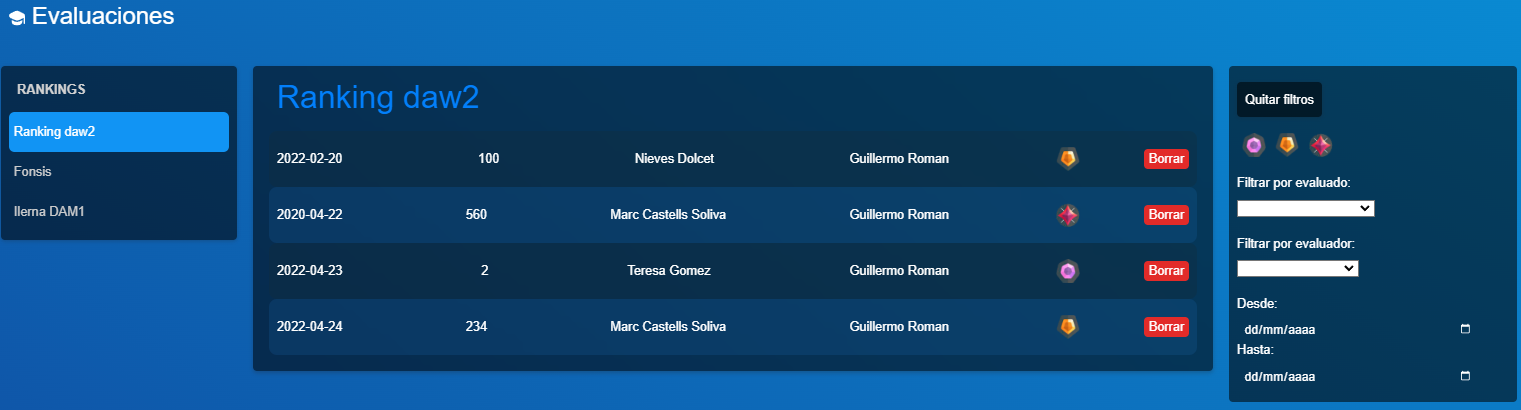


### CONSULTA DE EVALUACIONES

Dentro verá una página similar a los rankings, con su lista de rankings de ese profesor que ha iniciado sesión:

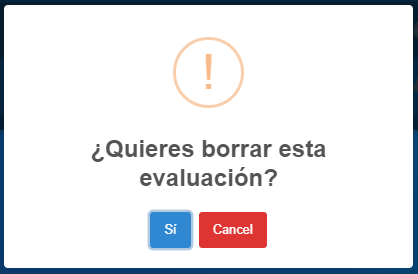


Al pulsar sobre un ranking veremos las evaluaciones que los alumnos de ese ranking se han hecho entre ellos:



### OPCIONES DE EVALUACIONES

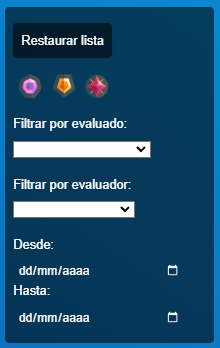
#### ELIMINAR EVALUACIONES

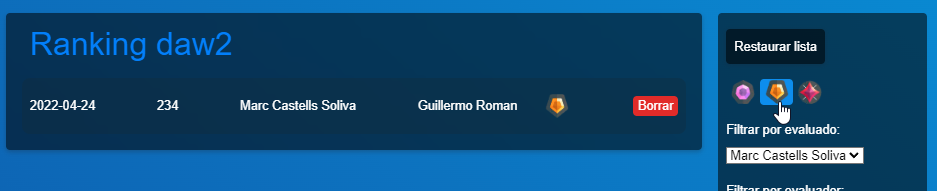
Se pueden eliminar evaluaciones pulsando en borrar:  
  
Aparecerá un modal de confirmación:  


#### 

#### FILTROS DE EVALUACIONES

También contamos con una lista de filtros que el profesor puede utilizar:





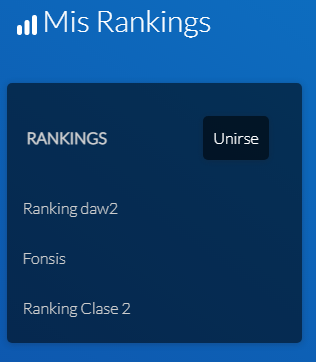
El botón de restaurar lista devolverá la lista a como estaba originalmente. Hay que tener en cuenta que los rangos de fechas “Desde” y “Hasta” se tienen que cambiar o borrar manualmente:

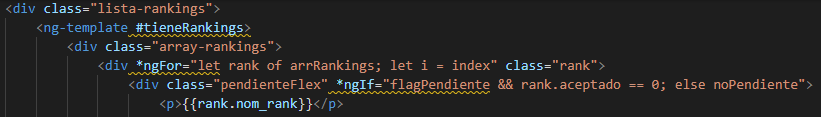


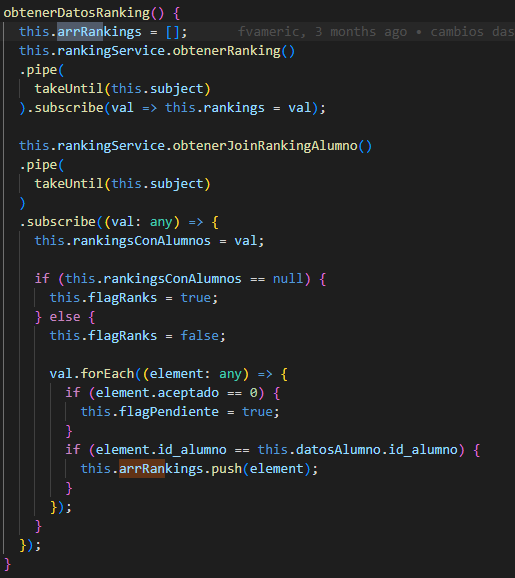
# DASHBOARD ALUMNO

## LISTADO RANKINGS

Este apartado, permite al alumno poder visualizar todos los rankings a los que este se ha unido y por lo tanto, a los que pertenece, también permite al alumno agregarse a nuevos rankings.



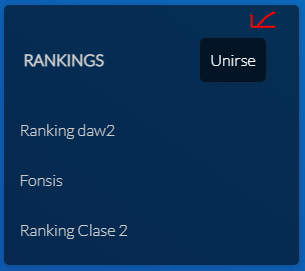


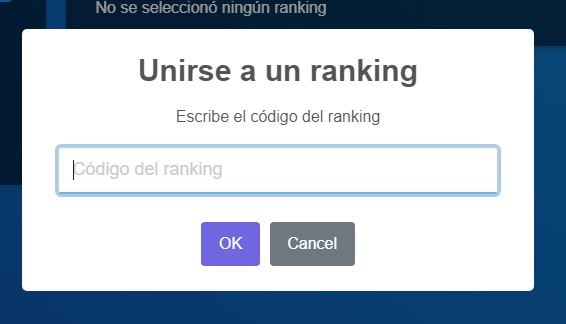


De esta forma y mediante la función “obtenerDatosRanking()”, es posible mostrar los rankings del alumno que ha iniciado sesión.

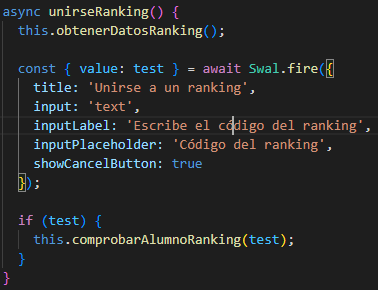
Extrayendo de la base de datos, todos los rankings en los que el alumno pertenezca, guardando los datos de cada ranking en un array llamado “arrRankings[]” y después, mostrando el array mediante el HTML.

## UNIRSE A RANKING

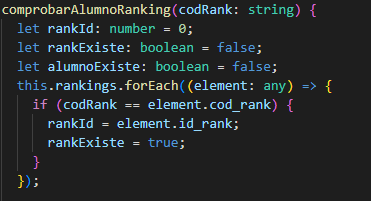




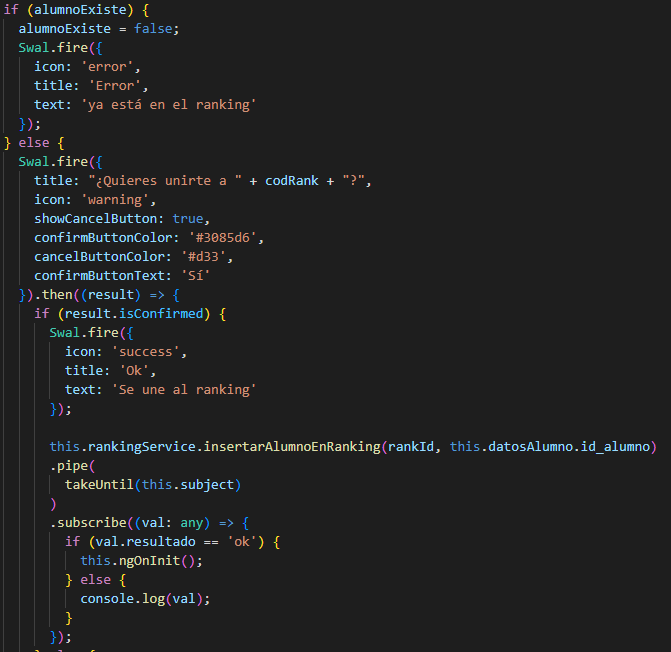
Para poder unirse a un ranking, se debe de utilizar el código que se ha generado cuando el profesor ha creado el ranking.



Seguidamente, se escribirá el código del ranking en el input del Sweetalert y se comprobará que el código escrito, va referenciado a algún ranking.



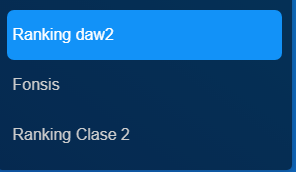
En caso de que el ranking exista, se comprobará que el alumno que intenta unirse al ranking no pertenezca ya a este.

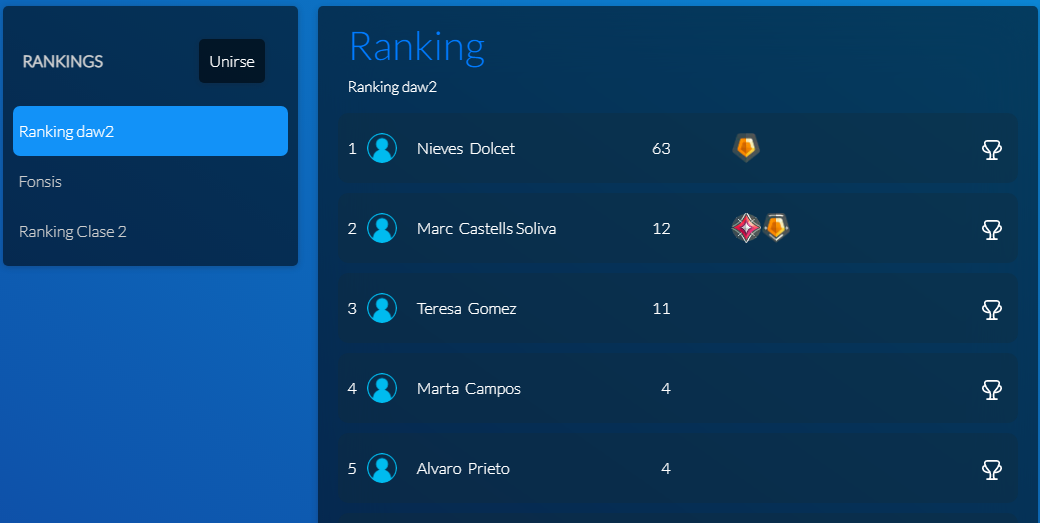


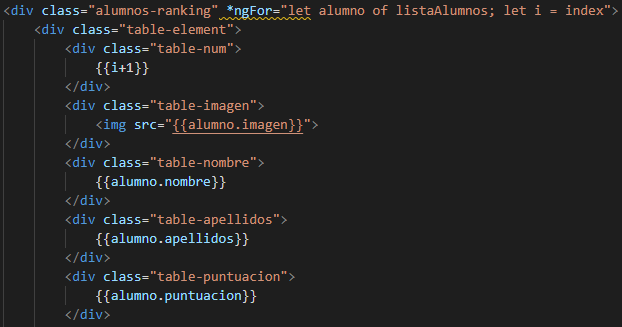
En el caso de que ya pertenezca al ranking, se le va a mostrar una pantalla de error, si no pertenece al ranking, se le va a mostrar una pantalla en la que se le pregunta al alumno si está seguro de unirse al ranking, si este accede, se insertan los datos en la base de datos.

## RANKING SELECCIONADO

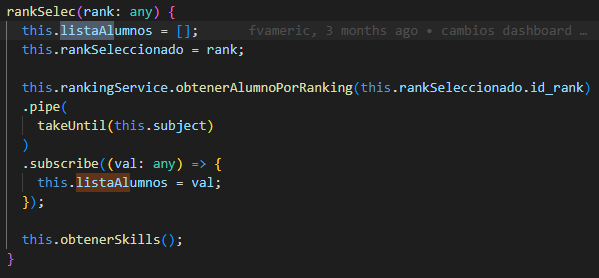
El alumno podrá seleccionar los rankings de su lista, de esta forma se muestra la información detallada a esos rankings, como todos los alumnos que pertenecen a ellos, sus puntuaciones, sus “softskills”…





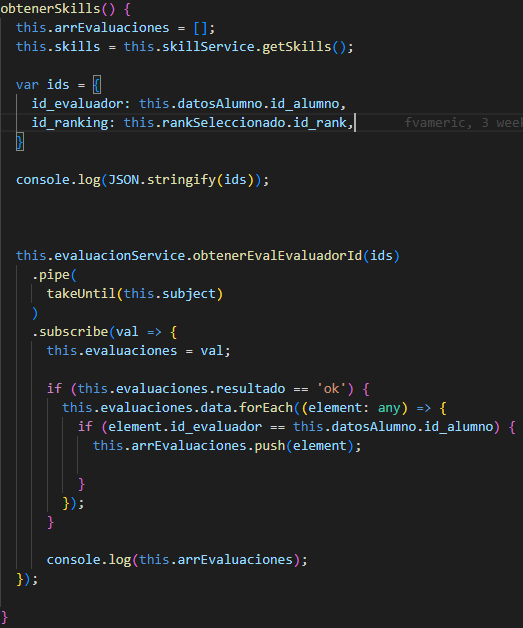


De esta manera se muestran los integrantes del ranking seleccionado, sus respectivos nombres y apellidos, y su puntuación.



Además, también, se muestran las softskills que ha conseguido cada alumno en ese ranking mediante los puntos otorgados por sus compañeros. Las softskills pueden variar el nivel, según los puntos que el alumno posea de cada una.

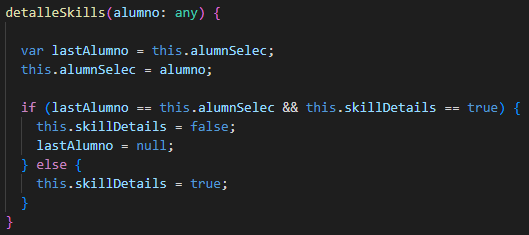




Mediante la siguiente funcion “obtenerSkills()” se recogen todos los puntos de las diferentes skills de cada alumno y se muestran.



Pulsando sobre ellas, se puede visualizar los puntos totales que posee el alumno de esa softskill en concreto.



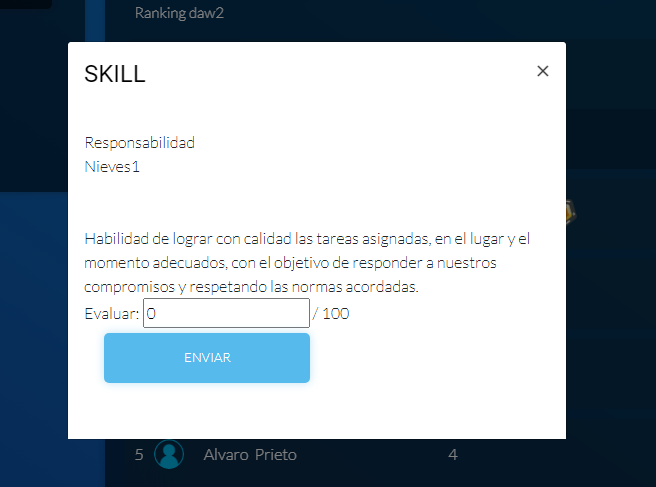
Esto es posible gracias a la siguiente función “detalleSkills()”, a la cual se le pasa una variable de tipo alumno, y se muestran los puntos de la softskill seleccionada mediante un boolean.

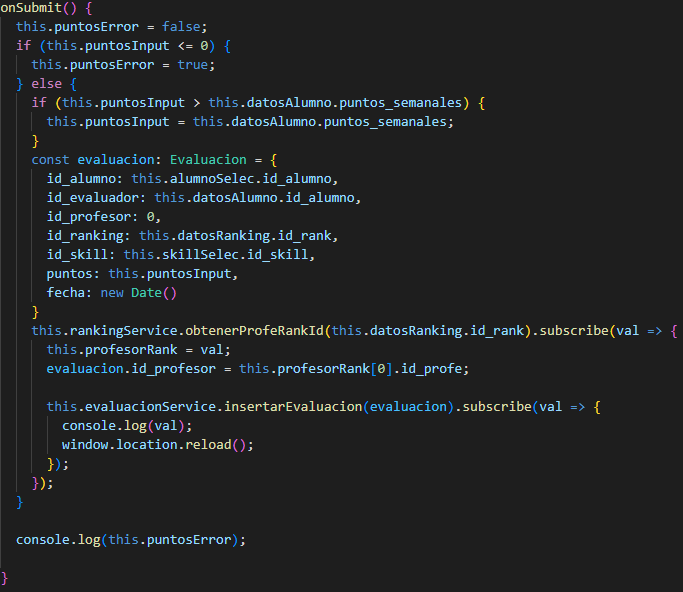
## PUNTUAR COMPAÑEROS

Es posible puntuar a los compañeros de un mismo ranking y asignarle diferentes puntuaciones de cada softskill.



Pulsando sobre el trofeo, se abre una segunda fila debajo de cada alumno.

  
Pulsando sobre cada medalla(que representa una softskill) se abre un menú el cual permite puntuar al alumno con un límite de hasta 100 puntos por softskill, esta votación no puede repetirla continuamente en una softskill para sumarle puntos infinitos, es únicamente aplicable una vez a la semana, el hecho de volver a votar la softskill, lo único que haría es modificar tu puntuación sobre ella.



De esta manera, cuando se pulse sobre el botón “enviar” y los puntos asignados sean mayores a 0, se crearà una variable de tipo Evaluación, con todos los datos referentes y se subirá a la base de datos mediante una función llamada “insertarEvaluacion(evaluacion)”, perteneciente al servicio de evaluaciones.