

14 de marzo de 2020

Jesus Ceacearo JIMENO, JOSE manuel terrero

CONVALID APP

2 DAM

Contenido

[Introducción 3](#_Toc36230257)

[Plan de empresa 3](#_Toc36230258)

[Justificación 3](#_Toc36230259)

[Nombre y logo 4](#_Toc36230260)

[Producto 4](#_Toc36230261)

[Modelo de negocio 5](#_Toc36230262)

[Consumidores y posibles clientes 5](#_Toc36230263)

[Competencia 6](#_Toc36230264)

[DAFO 6](#_Toc36230265)

[Publicidad y promoción 7](#_Toc36230266)

[Descripción detallada del sistema 7](#_Toc36230267)

[Descripción 7](#_Toc36230268)

[Aplicación web 7](#_Toc36230269)

[Aplicación Android 8](#_Toc36230270)

[Requisitos funcionales 8](#_Toc36230271)

[Historias de usuario 8](#_Toc36230272)

[Modelado 9](#_Toc36230273)

[Sketching 9](#_Toc36230274)

[Andoid 9](#_Toc36230275)

[Angular 16](#_Toc36230276)

[Diagrama de clases del modelo. 25](#_Toc36230277)

[Diseño 25](#_Toc36230278)

[Esquema de clases diseñado para la base de datos. 25](#_Toc36230279)

[Diseño de los servicios web. 26](#_Toc36230280)

[Courses 26](#_Toc36230281)

[Historial 27](#_Toc36230282)

[Module 28](#_Toc36230283)

[Schedule. 29](#_Toc36230284)

[User 29](#_Toc36230285)

[Implementación 32](#_Toc36230286)

[API 32](#_Toc36230287)

[Controllers 32](#_Toc36230288)

[Middleware 33](#_Toc36230289)

[Models 33](#_Toc36230290)

[Routes 34](#_Toc36230291)

[Android 35](#_Toc36230292)

[Commons 35](#_Toc36230293)

[Data: 35](#_Toc36230294)

[Models 35](#_Toc36230295)

[Retrofit 36](#_Toc36230296)

[UI 36](#_Toc36230297)

[Angular: 37](#_Toc36230298)

[Signin: 37](#_Toc36230299)

[Registro-final: 37](#_Toc36230300)

[Services 37](#_Toc36230301)

[Models: 37](#_Toc36230302)

[Dashboard: 37](#_Toc36230303)

# Introducción

Esta aplicación esta basada en el acceso y la salida de alumnos del centro escolar. Puesto que hay muchos alumnos en todas las ramas del instituto que tienen algunas asignaturas o módulos convalidados, el acceso y salidas del centro no es el mismo para todos.

Con esta aplicación queremos facilitar el trabajo a los trabajadores de dicho centro y al alumnado, acercarles a ambos la información necesaria en todo momento para poder tener claro que horas son las convalidadas de cada uno.

Los trabajadores podrán acceder a los horarios de los alumnos para comprobar las horas libres que tienen y si tienen permiso para poder salir en ellas del centro, puesto que hay alumnos que son menores de edad y aunque dispongan de dichas horas libres no podrían salir del centro sin autorización de sus tutores legales.

Los alumnos podrán acceder a la aplicación con su cuenta de correo electrónico corporativo facilitado por el centro. Con este correo podrán acceder a su horario día a día y les saldrán resaltadas las horas libres que tienen convalidadas.

# Plan de empresa

## Justificación

Actualmente, es muy difícil controlar la salida y entrada de los alumnos al centro, puesto que los porteros de dichos centros no tienen un control de los alumnos que pueden y que no pueden salir de este.

Nuestra idea de negocio es crear una aplicación donde los porteros tengan una lista de los alumnos matriculados en el centro en el cual aparezca un horario de los cursos, y en dichos cursos quien puede salir y quien no, y a la hora a la que puede salir.

A día de hoy, desconozco de alguna aplicación similar a esta, puesto que existe el Séneca, pero esta aplicación/página web está dedicada especialmente para los profesores y alumnos, pero nada hecho también pensando en los porteros para controlar quien sale del centro.

Este producto creo que se venderá, y por lo cual la diferencia de las demás, por la principal razón explicada anteriormente, los porteros no saben el horario de los alumnos, entonces este no sabe si algún alumno sale del centro porque no tiene ese modulo (lo tiene convalidado y/o aprobado) o simplemente se quiere saltar alguna hora por cualquier motivo.

## Nombre y logo

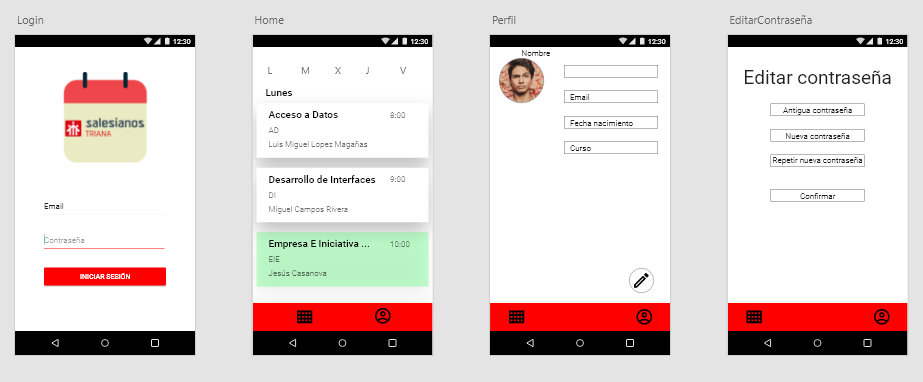


El nombre de la aplicación es **ConvalidApp**. Son un logo y un nombre simples, pero a la vez muy conciso que explica muy bien de que se trata. En el logo se representa la silueta de un calendario/horario con el nombre del centro al que esté dedicada esta aplicación. De esta manera la aplicación queda retratada con solo echarle un vistazo al logo, puesto que en él se representan las virtudes de nuestra aplicación.

## Producto

Este producto ofrecerá un servicio tanto al portero como a los profesores y alumnado para hacerles saber qué módulos tienen que dar y a la hora a la que tienen que darlas.

Aquí podemos ver una captura sobre la interfaz de la aplicación.



La finalidad y las necesidades que cubre esta aplicación/servicio es prácticamente las mencionadas ya, llevar un control sobre los alumnos para identificar quien está autorizado a salir del centro para facilitarles el trabajo tanto a los profesores como al portero del centro para que tenga una lista de dichos alumnos más a mano y fácil de localizar. Esto también está pensado para que los alumnos tengan más a la vista un horario donde poder ver los módulos que tienen que impartir y las horas que cubre dichos módulos.

## Modelo de negocio

Para nuestro servicio/aplicación lo principal y lo primero que deberíamos hacer sería realizar campañas de publicidad sobre nuestro producto para darnos a conocer y que los centros educativos, que serían nuestros principales objetivos, se fijen en nuestro producto. Una vez captada la atención de los centros, las ganancias con nuestro producto se conseguirán mediante suscripciones anuales a nuestro servicio, ya que es lo que dura un curso escolar.

Por otro lado, el darle un buen servicio al alumnado y a los trabajadores del centro se otra de nuestras maneras de comercio. Puesto que si ellos están contentos con el uso de nuestra aplicación la demanda de que esta siga en funcionamiento año tras año será mayor, de esta manera podremos mantener a los clientes durante más tiempo.

## Consumidores y posibles clientes

Nuestros consumidores y/o posibles clientes de nuestro servicio serán principalmente centros escolares y/o educativos. La zona en la que residen los consumidores es un poco indiferente, puesto que nosotros el servicio se lo ofrecemos directamente al centro en general, no a alguna persona en concreto.

Nuestro producto es utilizable en cualquier localización puesto que podemos amoldar datos y funcionabilidades a las necesidades del cliente.

## Competencia

Como ya comentamos en el primer apartado, no hay servicios como tal que sean idénticos como el nuestro, si es verdad que está Séneca, pero ese servicio está preparado específicamente para los maestros del centro y nuestro servicio va más allá, nuestro servicio da un soporte tanto a los profesores como al alumnado y al portero del centro. De esta manera ayudaríamos a la gestión del centro de una forma muy productiva. Nuestro servicio es único y eficaz.

## DAFO

|  |  |
| --- | --- |
| **DEBILIDADES** | **AMENAZAS** |
| Al empezar se cuenta con poco presupuesto, ya que al empezar la inversión es un poco alta. | Al empezar como nuevos, somos pocos conocidos en el mercado en el que nos movemos. |
| Al ser un grupo reducido y nuevo, se cuenta con poca experiencia del personal | Al ser nuevos, puede haber una posible desconfianza sobre servicio. |
| **FORTALEZAS** | **OPORTUNIDADES** |
| No empezamos prácticamente de cero, puesto que contamos con la experiencia de otros trabajadores. | Comunicación con otras empresas, junto con la posibilidad de crear una posible alianza con dicha empresa. |
| Posibilidad de presentar un modelo nuevo e innovador al empezar de cero y ver los ya existentes. |  |

## Publicidad y promoción

La publicidad que tendría esta aplicación seria mediante entornos sociales como LinkedIn y varios relacionados con el trabajo, así como redes sociales como Instagram o Twitter, en los cuales suelen estar más activas las empresas y la población. Asimismo, también se le podría dar promoción mediante conferencias o charlas con los directivos de los centros para presentarles nuestro producto.

# Descripción detallada del sistema

## Descripción

Nuestra aplicación servirá para gestionar la entrada y salida de alumnos del centro para horas convalidadas. De esta manera el portero sabrá si tienen permiso o no dichos alumnos para la salida del centro.

Solo se podrá acceder a la aplicación con los correos corporativos (@triana.salesianos.edu) de esta manera solo podrán iniciar sesión los alumnos, maestros, conserjes y administradores del colegio.

Con nuestra aplicación pretendemos facilitar el trabajo del dentro, tanto para alumnos teniendo a su disposición en todo momento su horario como para los trabajadores de el mismo.

También se llevará un registro de salidas del centro con esta app, de esta manera se sabrá en todo momento si el alumno estaba o no dentro del centro a una hora determinada.

### Aplicación web

Con respecto a la aplicación web, servirá para gestionar la administración de la aplicación Android, tanto para insertar cuentas de alumnos como horarios de clase y clases convalidadas de alumnos.

Al entrar en la aplicación web, te saldrá un formulario para poder loguearte. En dicha aplicación web solo podrán acceder los administradores, de esta manera podrán acceder a toda la gestión de la base de datos de la aplicación, para que tanto el conserje como los alumnos puedan acceder a los datos necesarios.

### Aplicación Android

Con respecto a la aplicación Android, servirá para gestionar el acceso a datos tanto de los conserjes como los alumnos.

De esta manera los conserjes pueden acceder a los horarios de cada curso y a los datos de horarios convalidados para cada alumno en específico, y los alumnos podrán acceder a sus horarios para ver sus horas convalidadas.

Ambos usuarios podrán acceder a su perfil para poder ver sus datos.

### Requisitos funcionales

1. **Gestión de usuarios**
   1. **Alta de un usuario**: el administrador podrá dar de alta a los usuarios en su aplicación rellenando un formulario con sus datos.
   2. **Editar perfil**: esta opción la podrá utilizar el administrador para cambiar el curso del usuario o para cambiar su contraseña de acceso.
   3. **Eliminar usuario**: esta opción la podrá utilizar el administrador para eliminar antiguos alumnos del centro o aquellos alumnos que se han des matriculado.
2. **Gestión de cursos**
   1. **Alta de un curso**: esta opción la podrá utilizar el administrador para dar de alta nuevos cursos en el centro.
   2. **Editar un curso**: esta opción la podrá utilizar el administrador para modificar un curso.
   3. **Eliminar un curso**: Esta opción la podrá utilizar el administrador para poder eliminar aquellos cursos que no existan en el centro.
3. **Gestión de módulos**
   1. **Alta de un módulo**: esta opción la podrá utilizar el administrador para dar de alta un nuevo módulo.
   2. **Editar un módulo**: esta opción la podrá utilizar el administrador para poder modificar un módulo.
   3. **Eliminar un módulo**: esta opción la podrá utilizar el administrador para poder eliminar un módulo.

## Historias de usuario

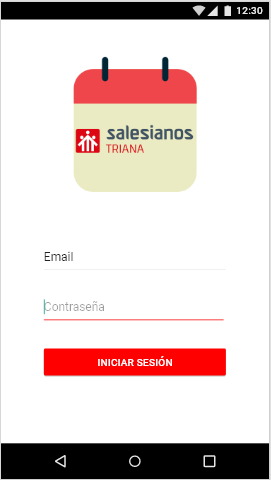
Las historias de usuario detalladas se encuentran en el documento DSP al cual se puede acceder pulsado [aquí](DSP%20-%20Documento%20de%20seguimiento%20del%20proyecto.xlsx) .

# Modelado

## Sketching

### Andoid

**Login**

****

La primera pantalla que nos encontramos es un Activity de Login. Esta pantalla cuenta con un ImageView en el que se mostrará el logo de la aplicación, un TetView en el que se imprimirá el nombre de la misma y un CardView con el formulario de logueo. En el cardview se imprimirá un mensaje de corrección si el usuario no ha introducido una cuenta de correo corporativa para que no intenten loguearse con una cuenta incorrecta, también habrá un botón de login y uno de registro.

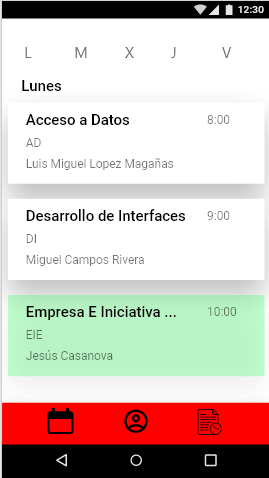
Desde esta pantalla de login podremos ir a:

* Pantalla principal de usuario
* Pantalla principal de conserje
* Pantalla principal de profesor

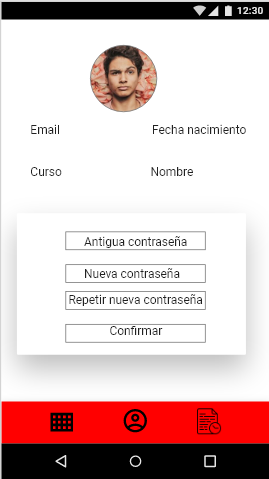
**Cliente**

La parte de la app para cliente esta destinada a los alumnos del centro, que son los que mas van a utilizar nuestra aplicación.

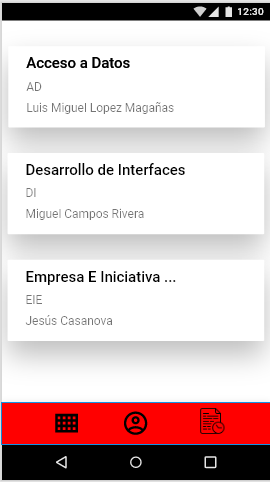
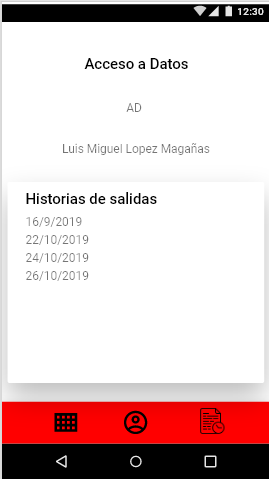
Esta parte será un Activity nuevo con un navegador inferior con tres pestañas: horario , perfil e historial. Cada pestaña carga su fragment correspondiente.



Por defecto se cargará la pestaña de horario. En esta pestaña se mostrará el horario del día de la semana en el que estamos. Las asignaturas se mostrarán en un listado modelado con RecyclerView y con su adapter personalizado, en su fragment correspondiente. En la parte superior tendremos un menú con los días de la semana de lunes a viernes, cuando seleccionemos el día que queremos ver la lista se recargara con el horario del día que seamos ver. También habrá dos pestañas para seleccionar si quiere mostrar el horario del curso al que pertenece o su horario personal.



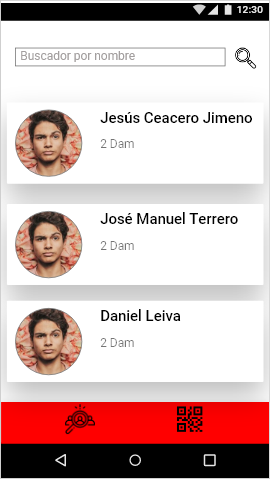
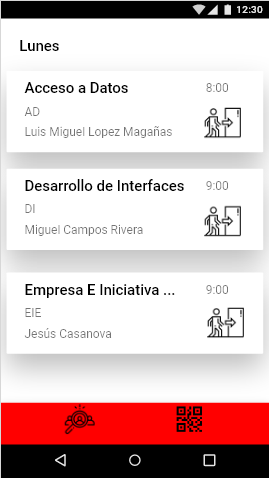
En la pestaña de perfil carga un fragmente donde abra un ImageView que mostrará la imagen que tenemos de perfil, este ImageView servirá también para abrir el explorador de archivos para seleccionar otra foto. Una vez seleccionemos la foto aparecerá dos iconos, uno para aceptar el cambio y otro para descartarlo. Debajo de la foto se mostrarán los datos del usuario.   
A continuación, habrá un CardView que contendrá un formulario para el cambio de contraseña que es el único dato que el usuario podrá modificar.

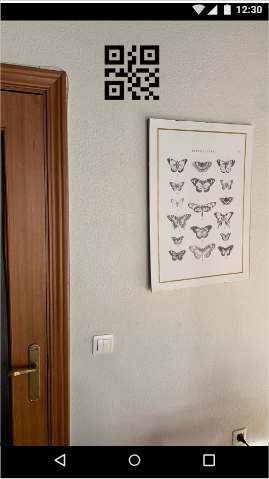
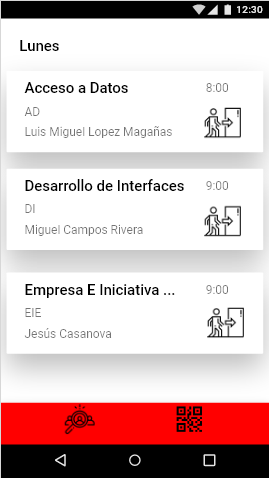
En la pestaña de historial se cargará un fragment con un RecyclerView y su adapter correspondiente, este mostrara un listado en Cardview donde se mostrarán las asignaturas convalidadas del alumno y sus datos. Al pinchar sobre alguna de ellas nos llevara a un Activity nuevo en el saldrán los datos de la asignatura y un poco más abajo un CardView con un listado de las salidas del centro de dicho alumno en esa asignatura.

**Conserje**

La parte de la app para el conserje está destinada a la comprobación de las asignaturas convalidadas y el registro de salidas. Esta parte abrirá un Activity con un navegador inferior en el cual habrá dos pestañas: listado de alumnos y código QR.

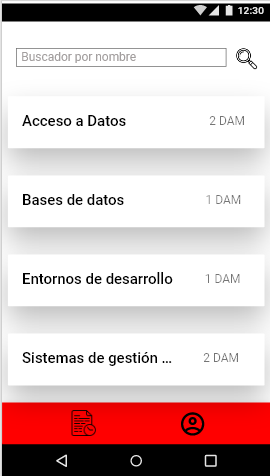
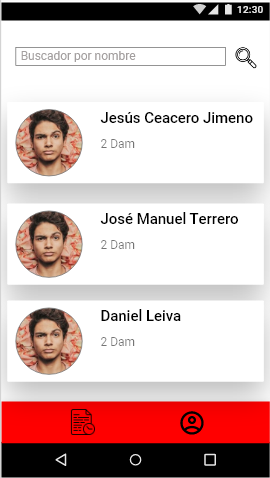
Por defecto este Activity cargará la pestaña de listado de alumnos, donde se cargará un fragment con un RecyclerView y su adapter personalizado. Aquí se mostrará el listado de todos los alumnos en CardView donde se mostrará la foto del alumno, el nombre y el curso. Se podrá filtrar la lista por nombre, para que la busque da un alumno en particular.   
Una vez tengamos al alumno deseado si clicamos sobre él se nos abrirá otro Activity con el listado del horario del curso al que pertenece, de esta forma el conserje podrá darle a una pestaña sobre la asignatura y hora deseada para registrar la salida del alumno del centro.

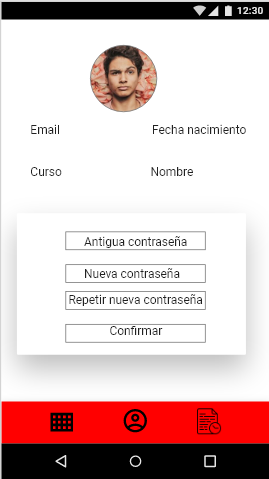
La pestaña de QR lo que realizará será abrir la cámara del teléfono, de esta forma podrá leer el código QR que el alumno tiene en su perfil de la aplicación. Una vez allá leído el código QR, se abrirá el activity con el horario del curso del alumno como en el caso anterior para poder seleccionar la hora a la que le desea registrar la salida.

**Profesor**

La parte de la app para los profesores esta destinada para que cada profesor pueda ver su horario y los alumnos convalidados en las asignaturas. Esta parte abrirá un Activity con un navegador inferior en el cual habrá dos pestañas: listado de asignaturas y perfil.

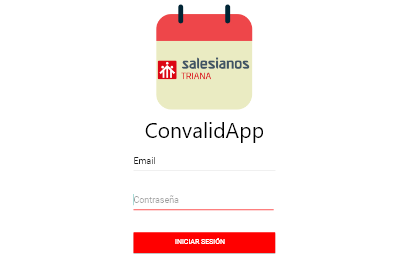
Pro defecto se abrirá la pestaña de listado de asignaturas. Esta pestaña cargara un fragment con un RecyclerView y su adapter personalizador, en él se pintarán todas las asignaturas que el profesor imparte. Al clicar sobre alguna de ellas se le abrirá un Activity nuevo donde habrá un listado de los alumnos que tienen dicha asignatura convalidada.



En la pestaña de perfil carga un fragmente donde abra un imageView que mostrará la imagen que tenemos de perfil, este ImageView servirá también para abrir el explorador de archivos para seleccionar otra foto. Una vez seleccionemos la foto aparecerá dos iconos, uno para aceptar el cambio y otro para descartarlo. Debajo de la foto se mostrarán los datos del usuario.   
A continuación, habrá un CardView que contendrá un formulario para el cambio de contraseña que es el único dato que el usuario podrá modificar.

### Angular

**Login**



La primera pantalla que nos encontramos es el Login. Esta pantalla cuenta con una imagen, que seria el logo de la aplicación, debajo encontramos el nombre de dicha aplicación y posteriormente encontramos unos campos de formulario donde tenemos que introducir el email y la contraseña y solo faltaría darle al botón de iniciar sesión que nos lo encontramos en la parte inferior para entrar en la aplicación. Si algún campo es erróneo o se quedara en blanco saltaría un mensaje en rojo con la información.

Desde esta pantalla de login podremos ir a:

* Pantalla principal de administrador

**Administrador**

La parte de la app para web está destinada al administrador del centro, que son los que van a realizar toda la gestión sobre el centro.



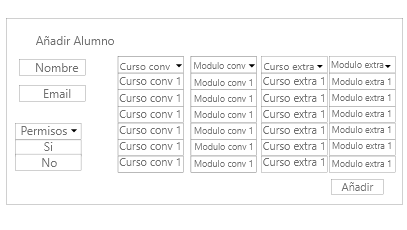
Por defecto se cargará la pantalla de listado de usuarios. Esta parte será una tabla con columnas y filas donde nos encontramos un listado de los usuarios registrados en la base de datos. A la derecha nos encontramos un menú lateral de navegación hacia las diferentes pantallas que contiene la aplicación, que es común en las diferentes pantallas principales de la aplicación. También nos encontramos con un buscador para filtrar por cualquier dato del usuario y en la parte inferior de la pantalla nos encontramos con una paginación.

En esta pantalla nos encontramos con un botón encima de la tabla donde podemos añadir un usuario de cualquiera de los tipos que hay (usuarios, profesores y conserjes).

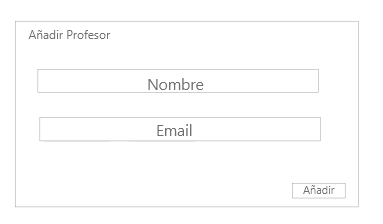


Dependiendo de en cual clickemos nos mandará a un diálogo con un formulario u a otro, pero todos tienen unos campos comunes, que son el nombre y el email.

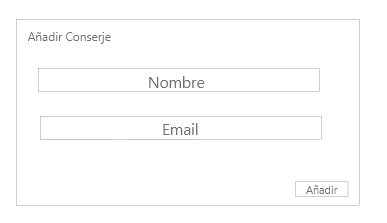
**Registro usuario**



**Registro profesor**

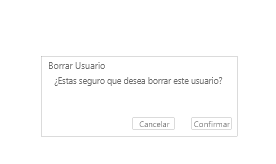


**Registro conserje**



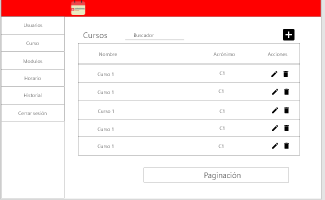
Si volvemos otra vez atrás al listado, más abajo del botón de añadir nos encontramos con una columna de acciones, donde podemos editar y borrar cualquier fila de la tabla.

**Borrar usuario**



En esta pantalla nos encontramos con un dialogo en donde nos muestra un mensaje de confirmación por si estamos seguro de que deseamos eliminar ese usuario.

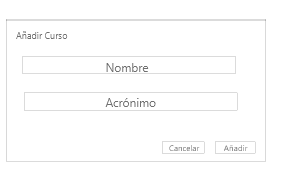
**Listado cursos**



Esta parte será una tabla con columnas y filas donde nos encontramos un listado de los cursos registrados en la base de datos. A la derecha nos encontramos con el menú lateral de navegación. Nos encontramos con un buscador para filtrar por cualquier dato de un curso en específico y en la parte inferior de la pantalla nos encontramos con una paginación.

En esta pantalla nos encontramos con un botón encima de la tabla donde podemos añadir un curso.

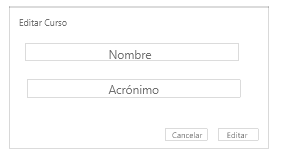
**Añadir curso**



Esta pantalla es un dialogo en donde nos encontramos con los campos que tenemos que rellenar para crear un curso y, una vez rellenados, podemos pinchar en el botón de añadir y el curso estará añadido o en el de cancelar por si no queremos añadir un curso y volver al listado de cursos.

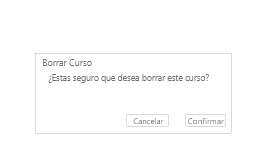
Si volvemos otra vez atrás al listado, más abajo del botón de añadir nos encontramos con una columna de acciones, donde podemos editar y borrar cualquier fila de la tabla.

**Editar curso**



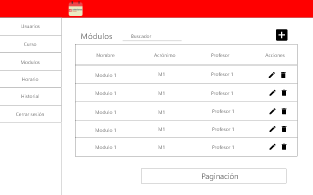
Esta pantalla es un diálogo en donde podemos editar cualquier campo de un curso en concreto. Una vez ya rellenados los campos de dicho formulario si pinchamos en el botón de Editar realizaremos la edición de dicho curso y volveríamos al listado de los cursos con dicho curso ya editado o, si queremos volver al listado sin editar nada podemos pinchar en el botón de Cancelar.

**Borrar curso**



En esta pantalla nos encontramos con un dialogo en donde nos muestra un mensaje de confirmación por si estamos seguro de que deseamos eliminar ese curso en concreto.

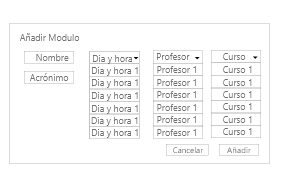
**Listado módulos**



Esta parte será una tabla con columnas y filas donde nos encontramos un listado de los módulos registrados en la base de datos. A la derecha nos encontramos con el menú lateral de navegación. Nos encontramos con un buscador para filtrar por cualquier dato de un módulo en específico y en la parte inferior de la pantalla nos encontramos con una paginación.

En esta pantalla nos encontramos con un botón encima de la tabla donde podemos añadir un módulo.

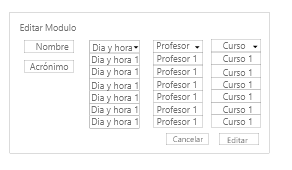
**Añadir módulo**



Esta pantalla es un dialogo en donde nos encontramos con los campos (input básicos y desplegables con diferentes opciones para elegir) que tenemos que rellenar para crear un módulo y, una vez rellenados, podemos pinchar en el botón de añadir y el módulo estará añadido o en el de cancelar por si no queremos añadir un módulo y volver al listado de módulos.

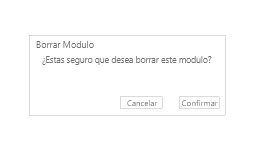
Si volvemos otra vez atrás al listado, más abajo del botón de añadir nos encontramos con una columna de acciones, donde podemos editar y borrar cualquier fila de la tabla.

**Editar módulo**



Esta pantalla es un diálogo en donde podemos editar cualquier campo de un módulo en concreto. Una vez ya rellenados los campos de dicho formulario si pinchamos en el botón de Editar realizaremos la edición de dicho módulo y volveríamos al listado de los módulos con dicho módulo ya editado o, si queremos volver al listado sin editar nada podemos pinchar en el botón de Cancelar.

**Borrar módulo**



En esta pantalla nos encontramos con un dialogo en donde nos muestra un mensaje de confirmación por si estamos seguro de que deseamos eliminar ese módulo en concreto.

**Listado horarios**



Esta parte será una tabla con columnas y filas donde nos encontramos un listado de los horarios registrados en la base de datos. A la derecha nos encontramos con el menú lateral de navegación. Nos encontramos con un buscador para filtrar por cualquier dato de un horario en específico y en la parte inferior de la pantalla nos encontramos con una paginación.

En esta pantalla nos encontramos con un botón encima de la tabla donde podemos añadir un horario.

**Añadir horario**



Esta pantalla es un dialogo en donde nos encontramos con los campos (desplegables con los días y las horas de un día escolar) que tenemos que rellenar para crear un horario y, una vez rellenados, podemos pinchar en el botón de añadir y el horario estará añadido o en el de cancelar por si no queremos añadir un horario y volver al listado de horarios.

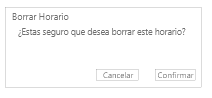
Si volvemos otra vez atrás al listado, más abajo del botón de añadir nos encontramos con una columna de acciones, donde podemos editar y borrar cualquier fila de la tabla.

**Editar horario**



Esta pantalla es un diálogo en donde podemos editar cualquier campo de un horario en concreto. Una vez ya rellenados los campos de dicho formulario si pinchamos en el botón de Editar realizaremos la edición de dicho horario y volveríamos al listado de los horarios con dicho horario ya editado o, si queremos volver al listado sin editar nada podemos pinchar en el botón de Cancelar.

**Borrar horario**



En esta pantalla nos encontramos con un dialogo en donde nos muestra un mensaje de confirmación por si estamos seguro de que deseamos eliminar ese horario en concreto.

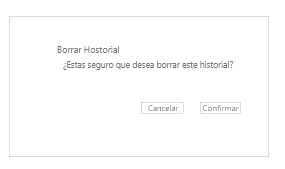
**Listado historial**



Esta parte será una tabla con columnas y filas donde nos encontramos un listado de las veces que han salido los alumnos del centro, a qué hora de la mañana y en que modulo, a modo de historial, registrados en la base de datos. A la derecha nos encontramos con el menú lateral de navegación. Nos encontramos con un buscador para filtrar por cualquier dato de un historial en específico y en la parte inferior de la pantalla nos encontramos con una paginación.

En esta pantalla únicamente nos encontramos en la columna de acciones la opción de borrar un historial por si, cuando acabe el curso o dicho alumno se borrara del centro, desean limpiar los valores de la tabla.

**Borrar historial**



En esta pantalla nos encontramos con un dialogo en donde nos muestra un mensaje de confirmación por si estamos seguro de que deseamos eliminar esa fila del historial en concreto.

## Diagrama de clases del modelo.



# Diseño

## Esquema de clases diseñado para la base de datos.

const courseSchema = new mongoose.Schema({

name: {type: String},

acronym: {type: String},

modules: [{type: Schema.ObjectId, ref: 'Module'}]

});

const scheduleSchema = new mongoose.Schema({

user: {type: Schema.ObjectId, ref: 'User'},

module: {type: Schema.ObjectId, ref: 'Module'},

horario: {type: Schema.Types.ObjectId, ref: 'Schedule'},

fechayhora: {type: Date}

});

const moduleSchema = new mongoose.Schema({

name: {type: String},

acronym: {type: String},

teacher: {type: Schema.Types.ObjectId, ref: 'User'},

horario: [{type: Schema.Types.ObjectId, ref: 'Schedule'}],

course: {type: Schema.Types.ObjectId, ref: 'Course'}

});

const moduleSchema = new mongoose.Schema({

user: {type: Schema.Types.ObjectId, ref: 'User'}

});

const scheduleSchema = new mongoose.Schema({

dia: {type: String, enum: ['Lunes', 'Martes','Miercoles', 'Jueves', 'Viernes']},

hora: {type: String, enum: ['8:00', '9:00','10:00', '11:30', '12:30', '13:30']}

});

const userSchema = new mongoose.Schema({

email: {type: String},

fullname: {type: String},

role: {type: String, enum: ['USER', 'CONSERJE','PROFESOR', 'ADMIN'], default: "USER"},

password: {type: String},

birthdate: {type: String},

permissions: {type: Boolean, default: false},

photo: imgSchema,

course: {type: Schema.ObjectId, ref: 'Course'},

convalidados: [{type: Schema.ObjectId, ref: 'Module'}],

extras: [{type: Schema.ObjectId, ref: 'Module'}],

imparte: [{type: Schema.ObjectId, ref: 'Module'}]

});

## Diseño de los servicios web.

### Courses

* **(POST) addCourse: “{{base\_url}}/courses/add”:**
  + Crea un curso
  + Se le pasará como cuerpo el curso creado
  + Devolverá el curso creado
* **(GET) findAll: “{{base\_url}}/courses/”:**
  + Busca los cursos en la base de datos.
  + Devolverá un array con todos los cursos de la base de datos.
* **(GET) findOne: “{{base\_url}}/courses/:id”:**
  + Busca un curso en la base de datos.
  + Se le pasará como parámetro el id del curso.
  + Devolverá el curso buscado.
* **(PUT) updateCourse: “{{base\_url}}/courses/edit/:id”:**
  + Busca y modifica un curso.
  + Se le pasará como parámetro el id del curso.
  + Se le pasará como cuerpo el curso modificado.
  + Devolverá el curso modificado.
* **(DELETE) deleteCourse: “{{base\_url}}/courses/delete/:id”:**
  + Elimina un curso.
  + Se le pasará como parámetro el id del curso.
  + No devolverá nada.

### Historial

* **(POST) addHistorial: “{{base\_url}}/historial/add”:**
  + Crea un Historial.
  + Se le pasará como cuerpo el historial credo.
  + Devolverá el historial creado.
* **(GET) findAll: “{{base\_url}}/historial/”:**
  + Busca todos los historiales creados.
  + Devolverá un listado de
* **(GET) findOne: “{{base\_url}}/historial/:id”:**
  + Buscara un historial en la base de datos
  + Se le pasará como parámetro el id del historial.
  + Devolverá el historial buscado
* **(PUT) updateHistorial: “{{base\_url}}/historial/edit/:id”:**
  + Busca y modifica un historial.
  + Se le pasará como parámetro el id del historial.
  + Se le pasara como cuerpo el historial modificado.
  + Devolverá el historial modificado.
* **(DELETE) deleteHistorial: “{{base\_url}}/historial/delete/:id”:**
  + Borra un historial.
  + Se le pasará como parámetro el id del historial.
  + No devolverá nada.

### Module

* **(POST) addModule: “{{base\_url}}/modules/add”:**
  + Crea un modulo.
  + Al crear el modulo vinculara este con el profesor que lo imparte.
  + Al crear el modulo vinculara este con el curso en el que está.
  + Se le pasará como cuerpo el módulo creado.
  + Devolverá el módulo creado.
* **(GET) findAll: “{{base\_url}}/ modules /”:**
  + Busca todos los módulos creados.
  + Devolverá el historial creado.
* **(GET) findOne: “{{base\_url}}/ modules /:id”:**
  + Busca un módulo.
  + Se le pasará como parámetro el id del módulo.
  + Devolverá el módulo buscado.
* **(PUT) updateModule: “{{base\_url}}/ modules /edit/:id”:**
  + Busca y modifica un módulo.
  + Se le pasara como parámetro el id del módulo.
  + Se le pasará como cuerpo el módulo modificado.
  + Devolverá el módulo modificado.
* **(PUT) deleteModule: “{{base\_url}}/ modules /delete/:id”:**
  + Busca y elimina un módulo.
  + Se le pasara como parámetro el id del módulo.
  + No devuelve nada.
* **(POST) findListCourse: “{{base\_url}}/ modules /list/”:**
  + Busca un listado de módulos
  + Se le pasará como cuerpo el listado con las ids de los módulos.
  + Devolverá el listado de módulos buscado

### Schedule.

* **(POST) addSchedule: “{{base\_url}}/ schedules /add”:**
  + Crea un horario
  + Se le pasará como cuerpo el horario creado.
  + Devolverá el horario creado.
* **(GET) findAll: “{{base\_url}}/ schedules /”:**
  + Busca todos los horarios
  + Devolverá el listado de todos los horarios.
* **(GET) findOne: “{{base\_url}}/ scheludes /:id”:**
  + Busca un horario
  + Se le pasara como parámetro el id del horario.
  + Devolverá el horario buscado.
* **(PUT) updateSchedule: “{{base\_url}}/ schedules /edit/:id”:**
  + Busca y modifica un horario.
  + Se le pasara como parámetro el id del horario.
  + Se le pasará como cuerpo el horario modificado.
  + Devolverá el horario modificado.
* **(DELETE) updateSchedule: “{{base\_url}}/ schedules /delete/:id”:**
  + Busca y elimina un horario.
  + Se le pasara como parámetro el id del horario.
  + No devolverá nada.

### User

* **(POST) register: “{{base\_url}}/ users /register”:**
  + Registra un usuario en la base de datos.
  + Se le pasará como cuerpo el usuario creado.
  + Devolverá el usuario registrado.
* **(POST) login: “{{base\_url}}/ users /login”:**
  + Comprueba el logueo del usuario
  + Se le pasará como cuerpo el username y password del usuario.
  + Devolverá el usuario logueado y un token.
* **(POST) loginAdmin: “{{base\_url}}/ users /login/admin”:**
  + Comprueba el logueo del usuario administrador.
  + Se le pasará como cuerpo el username y password del administrador.
  + Devolverá el administrador logueado y un token.
* **(GET) findAll: “{{base\_url}}/ users /”:**
  + Busca todos los usuarios.
  + Devolverá el listado de todos lo usuarios.
* **(GET) findAllProfesor: “{{base\_url}}/ users /profesor”:**
  + Busca todos los usuarios con rol PROFESOR o ADMIN.
  + Devolverá el listado de todos los usuarios con rol PROFESOR o ADMIN.
* **(GET) findAllUser: “{{base\_url}}/ users /users”:**
  + Busca todos los usuarios con rol USER
  + Devolverá el listado con todos los usuarios con rol USER.
* **(GET) findOne: “{{base\_url}}/ users /:id”:**
  + Busca un usuario.
  + Devolverá el usuario buscado.
* **(GET) findAllConvProfesor: “{{base\_url}}/ :id /profesor”:**
  + Busca un listado de usuarios con un módulo convalidado.
  + Se le pasara como parámetro el id del módulo convalidado.
  + Devolverá el listado de usuarios con el módulo convalidado.
* **(GET) getImage: “{{base\_url}}/ users /:id/img”:**
  + Busca la imagen de perfil de un usuario
  + Se le pasara como parámetro el id del usuario.
  + Devolverá la imagen del usuario.
* **(PUT) updateUser: “{{base\_url}}/ users /:id”:**
  + Busca y modifica un usuario.
  + Se le pasara como parámetro el id del usuario.
  + Se le pasará como cuerpo el usuario modificado.
  + Devolverá el usuario modificado.
* **(PUT) updatePassword: “{{base\_url}}/ users /:id/password”:**
  + Busca y modifica la contraseña de un usuario.
  + Se le pasara como parámetro el id del usuario.
  + Se le pasará como cuerpo el email y la nueva contraseña.
  + Devolverá el usuario modificado.
* **(PUT) updateImg: “{{base\_url}}/ users /:id/img”:**
  + Busca y modifica la imagen de un usuario.
  + Se le pasara como parámetro el id del usuario.
  + Se le pasará como cuerpo la imagen de usuario.
  + Devolverá el usuario modificado.
* **(DELETE) deleteUser: “{{base\_url}}/ users /:id”:**
  + Busca y elimina un usuario.
  + Se le pasara como parámetro el id del usuario.
  + No Devolvera nada.
* **(DELETE) deleteImg: “{{base\_url}}/ users /:id/img”:**
  + Busca y elimina la imagen de un usuario
  + Se le pasara como parámetro el id del usuario.
  + No devolverá nada.
* **(GET) getRegister: “{{base\_url}}/ users /register/:id”:**
  + Busca un registro de usuario.
  + Se le pasara como parámetro el id del registro.
  + Devolverá el registro buscado
* **(POST) updateModule: “{{base\_url}}/ modules /edit/:id”:**
  + Busca un registro y un usuario, elimina un registro y modifica un usuario.
  + Se le pasara como parámetro el id del registro.
  + Se le pasará como cuerpo el usuario modificado.
  + Devolverá el usuario modificado.

# Implementación

## API

En este apartado se describe los paquetes y las clases de la aplicación node.

### Controllers

Este paquete esta para contener las clases de los controladores de las diferentes clases pojo que tenemos en nuestro proyecto.

Estos controladores manejan que hace la aplicación con respecto a las peticiones de las diferentes aplicaciones que se conecten a ella.

Course:

Este controlador maneja las peticiones que se realicen con respecto a la clase de cursos.

En el se gestionan todas las acciones que se deben de realizar (borrado, creado, editado, consultas …). El controlado es el único que accede a la base de datos directamente.

Error\_types.:

Este controlador tiene lo tipos de errores que pueden devolver las peticiones para poder utilizarlos.

Historial:

Este controlador maneja las peticiones que se realicen con respecto a la clase de historial.

En él se gestionan todas las acciones que se deben de realizar (borrado, creado, editado, consultas …). El controlado es el único que accede a la base de datos directamente.

Module:

Este controlador maneja las peticiones que se realicen con respecto a la clase de modulo.

En él se gestionan todas las acciones que se deben de realizar (borrado, creado, editado, consultas …). El controlado es el único que accede a la base de datos directamente.

Schedule:

Este controlador maneja las peticiones que se realicen con respecto a la clase de horario.

En él se gestionan todas las acciones que se deben de realizar (borrado, creado, editado, consultas …). El controlado es el único que accede a la base de datos directamente.

User:

Este controlador maneja las peticiones que se realicen con respecto a la clase de Usuario.

En él se gestionan todas las acciones que se deben de realizar (borrado, creado, editado, consultas …). El controlado es el único que accede a la base de datos directamente.

### Middleware

Este paquete contiene los middlewares que controla la seguridad de la aplicación.

Auth:

Este middleware contiene el método para la autenticación en la aplicación y otro método que maleja los tipos de errores posibles del api.

Has\_role\_admin:

Este middleware controla el rol del usuario, controla que el usuario que realiza la petición en la que se añade sea un usuario con rol de ADMIN.

### Models

Este paquete contiene las clases con los modelos pojo de nuestra base de datos.

Course:

Esta clase contiene el esquema de la clase curso, con el dictaminamos los atributos y elementos que contiene esta clase en la base de datos (tipos de los parámetros, restricciones, exigencias, …).

Historial:

Esta clase contiene el esquema de la clase historial, con el dictaminamos los atributos y elementos que contiene esta clase en l base de datos (tipos de los parámetros, restricciones, exigencias, …).

Module:

Esta clase contiene el esquema de la clase modulo, con el dictaminamos los atributos y elementos que contiene esta clase en l base de datos (tipos de los parámetros, restricciones, exigencias, …).

Register:

Esta clase contiene el esquema de la clase registro, con el dictaminamos los atributos y elementos que contiene esta clase en l base de datos (tipos de los parámetros, restricciones, exigencias, …).

Schedule:

Esta clase contiene el esquema de la clase horario, con el dictaminamos los atributos y elementos que contiene esta clase en l base de datos (tipos de los parámetros, restricciones, exigencias, …).

User:

Esta clase contiene el esquema de la clase usuarios, con el dictaminamos los atributos y elementos que contiene esta clase en l base de datos (tipos de los parámetros, restricciones, exigencias, …).

### Routes

Este paquete contiene las clases que administran a que controlador va cada ruta de las peticiones al api. Con el podemos gestionar las peticiones de una manera eficiente, no solo diciendo al controlador que tiene que acceder cada ruta, sino que también le decimos a que middleware tiene que acceder para comprobar roles y autenticaciones.

Historial:

Esta clase contiene las rutas que serán destinadas al apartado de historial. En el gestionaremos si es necesario autenticación para acceder a ella, roles y a que controlador va.

Module:

Esta clase contiene las rutas que serán destinadas al apartado de modulo. En el gestionaremos si es necesario autenticación para acceder a ella, roles y a que controlador va.

Schedule:

Esta clase contiene las rutas que serán destinadas al apartado de horario. En el gestionaremos si es necesario autenticación para acceder a ella, roles y a que controlador va.

Usher:

Esta clase contiene las rutas que serán destinadas al apartado de usuario. En el gestionaremos si es necesario autenticación para acceder a ella, roles y a que controlador va.

Coures:

Esta clase contiene las rutas que serán destinadas al apartado de cursos. En el gestionaremos si es necesario autenticación para acceder a ella, roles y a que controlador va.

## Android

En este apartado se describe los paquetes y las clases de la aplicación Android.

### Commons

Este paquete sirve para las diferentes funcionalidades comunes de la aplicación

Constants:

Es esta clase sirve para guardas las constantes que se necesitan para el funcionamiento de la aplicación.

MyApp:

Clase que sirve para mostrar en el contexto que estamos

SharedPreferencesManager:

Clase para guardar Strings que pueden ser rescatados desde cualquier parte de la aplicación

### Data:

Este paquete contiene los paquetes de repositorios y viewmodel.

Repositorios:

Estas clases contiene todas las peticiones al api conectadas a través del servicio.

ViewModel:

Esta clase contiene todos los ViewModel que conectan los repositorios y las clases. También conectan los MainActivity con los Fragments.

### Models

Este paquete contiene todas las clases con los modelos que las clases que recibe del api, también tiene los DTO necesarios para mandar datos al api.

### Retrofit

Contiene todo lo necesario para que funcione retrofit.

Servicios:

Estas clases sirven para generar los servicios necesarios para los logueo y autenticaciones para el api.

Servicio:

Esta interface es la que conecta los repositorios con el api, definiendo el tipo de petición los parámetros, cuerpo, …

### UI

Este paquete contiene los paquetes de las pantallas de la aplicación.

Asignatura:

Este paquete contiene el fragment y el adapter para la pantalla de asignaturas convalidadas,

Asignatura\_prof:

Este paquete contiene el fragment y el adapter para la pantalla de asignaturas del profesor.

Auth:

Este paquete contiene el activity de login.

Profile:

Este paquete contiene el fragment del perfin de usuario.

QR:

Este paquete contiene el fragment y los activity necesarios para la generación y lector de los códigos qr.

Schedule:

Este paquete contiene el fragment y el adapter necesarios para la pantalla del horario del alumno.

Users:

Este paquete contiene los fragment, adapters y activity necesarios apara las pantallas de usuarios.

También se encuentran los tres MainActivity correspondientes a los roles implementados en el api, cada uno con su navegación correspondiente.

## Angular:

En este apartado se describe los paquetes y las clases de la aplicación angular.

### Signin:

Este paquete contiene las clases necesarias para el componente de logueo de la aplicación.

### Registro-final:

Este paquete contiene las clases necesarias para el componente del registro final.

### Services

Este paquete contiene los servicios de casa una de las clases de la base de datos, necesarios para la conexión al api.

### Models:

Este paquete contiene todos los modelos de las clases de la base de datos y los DTO necesarios para mandar y recibir datos al api.

### Dashboard:

En este paquete se encuentran todos los componentes que se utilizan en la aplicación una vez que nos hallamos logueado.

Cursos-add-dialog:

Este es el componente que abre el dialogo para la creación de un curso.

Cursos-borrar-dialog:

Este es el componente que abre el dialogo para el borrado de un curso.

Cursos-edit-dialog:

Este es el componente que abre el dialogo para la edición de un curso.

Cursos-listado:

Este es el componente que genera el listado de lo cursos con el filtrado y la paginación correspondiente. Y que enlaza con el borrado, editado y creación de los mismos.

Historial-borrar-dialog:

Este es el componente que abre el dialogo para el borrado de un historial.

Historial-listado:

Este es el componente que se encarga del listado de historiales. Con su filtrado correspondiente y paginado.

Horario-add-dialog:

Este paquete contiene las clases necesarias para el componente de dialogo para la creación de un nuevo historial.

Horario-borrar-dialog:

Este paquete contiene las clases necesarias para el componente de dialogo de borrado de un horario.

Horario-edit-dialog:

Este es paquete contiene las clases necesarias para el componente de dialogo de editado de un horario.

Horarios-listado:

Este paquete contiene las clases necesarias para el componente de listado de los horarios de la base de datos, con su filtrado y paginación correspondientes.

Modulos-add-dialog:

Este es el paquete que contiene las clases necesarias para el componente de creación de un nuevo módulo.

Modulos-borrar-dialog:

Este es el paquete que contiene las clases necesarias para el componente de borrado de un módulo.

Modulos-edit-dialog:

Este es el paquete que contiene las clases necesarias para el componente de editado de un módulo.

Modulos-listado:

Este componente contiene las clases necesarias para el componente de listado de módulos, con su filtrado y paginación correspondientes.

Usuario-borrar-dialog:

Este paquete contiene las clases necesarias para la el componente de borrado de un usuario.

Usuarios-add-alumno:

Este paquete contiene las clases necesarias para el componente del formulario de registro de un alumno.

Usuarios-add-conserje:

Este paquete contiene las clases necesarias para el componente del formulario de registro de un conserje.

Usuarios-add-dialog:

Este paquete contiene las clases necesarias para el componente del dialog para seleccionar que tipo de usuario se quiere crear.

Usuarios-add-profesor:

Este paquete contiene las clases necesarias para el componente del formulario de registro de un profesor.

Usuarios-edit-dialog:

Este paquete contiene las clases necesarias para el componente del dialogo para el editado de un usuario.

Usuarios-listado:

Este paquete contiene las clases necesarias para el componente de listado de usuarios, con su filtrado y paginado correspondiente.