**PROYECTO FIN DE CICLO**

Home

Departamento de Informática y Comunicaciones

Técnico Superior en Desarrollo de Aplicaciones Multiplataforma

IDEApp

Autoras

**María Inmaculada Campillo Soto**

**Estíbaliz Echeverría Aristu**

Tutor

**Antonio Alcaraz Cárcel**

Alquerías, 15 de junio de 2020

**Índice**

[0. Introducción. 1](#_Toc43058540)

[1. Objetivos del Proyecto. Definición de este. 1](#_Toc43058541)

[1.1. Descripción del Proyecto. 1](#_Toc43058542)

[1.2. Definición de los objetivos. 2](#_Toc43058543)

[1.3. Definición del alcance y las restricciones del sistema. 2](#_Toc43058544)

[Alcance 2](#_Toc43058545)

[Restricciones 2](#_Toc43058546)

[1.4. Beneficios. 3](#_Toc43058547)

[2. Análisis del sistema. 3](#_Toc43058548)

[2.1. Obtención de Información. 3](#_Toc43058549)

[2.2. Recursos necesarios. 3](#_Toc43058550)

[2.2.1. Recursos humanos. 4](#_Toc43058551)

[2.2.2. Recursos software. 4](#_Toc43058552)

[2.2.3. Recursos hardware. 4](#_Toc43058553)

[2.3. Planificación y Estudio de Viabilidad. 5](#_Toc43058554)

[2.3.1. Identificación y temporalización de fases de proyecto. 5](#_Toc43058555)

[2.3.2. Estimación de la duración de las actividades. 6](#_Toc43058556)

[2.3.3. Estimación de los costes de las actividades. 7](#_Toc43058557)

[2.3.4. Diagrama Gantt. 7](#_Toc43058558)

[2.3.5. Constitución del equipo y asignación del trabajo correspondiente. 7](#_Toc43058559)

[3. Diseño del sistema. 9](#_Toc43058560)

[3.1. El diseño de los datos. Diagrama Entidad/Relación. 9](#_Toc43058561)

[3.2. Diseño de la Interfaz. 10](#_Toc43058562)

[3.3. Diseño de los procedimientos. 17](#_Toc43058563)

[4. Implementación. 19](#_Toc43058564)

[4.1. Preparación del entorno operativo. 19](#_Toc43058565)

[4.2. Definición de políticas de backup. 22](#_Toc43058566)

[4.3. Instalación de entorno de desarrollo y prueba. 22](#_Toc43058567)

[5. Prueba. 23](#_Toc43058568)

[5.1. Indicadores de calidad para realizar la evaluación. 23](#_Toc43058569)

[5.2. Elaborar una batería de pruebas para detectar errores. 25](#_Toc43058570)

[5.3. Procedimiento para la participación en la evaluación de los usuarios finales 25](#_Toc43058571)

[6. Documentación. 25](#_Toc43058572)

[6.1. Estructura de Bases de Datos. 25](#_Toc43058573)

[6.2. Código de Fuentes. 28](#_Toc43058574)

[6.3. Documentos de Definición Inicial y todas las modificaciones solicitadas y realizadas. 34](#_Toc43058575)

[7. Mantenimiento. 35](#_Toc43058576)

[7.1. Mejorar IDEApp. 35](#_Toc43058577)

[8. Licencias. 35](#_Toc43058578)

[9. Conclusión. 36](#_Toc43058579)

[10. Bibliografía. 37](#_Toc43058580)

[10.1. Cursos. 37](#_Toc43058581)

[10.2. Documentación de Firebase. 37](#_Toc43058582)

[10.3. Documentación de Angular. 37](#_Toc43058583)

# Introducción.

En este documento procederemos a explicar detalladamente la estructura y el funcionamiento, de “IDEApp”, una aplicación orientada a instituciones educativas como el instituto de Alquerías. En pocas palabras, es una plataforma educativa con algunas mejoras que permiten a los alumnos y profesores subir sus propios apuntes a la plataforma.

Explicaremos brevemente qué tecnologías han sido usadas y cómo poder preparar un entorno apropiado.

Además, aquí se encontrará diferente información como la estructura que tienen los datos, el proyecto, una introducción a lo que sería el uso de esta aplicación y se pondrá en contexto cómo hemos implementado “IDEApp”.

# Objetivos del Proyecto. Definición de este.

En primer lugar, describiremos el proyecto junto con los objetivos que pretendemos lograr, así como su alcance, limitaciones y los beneficios del sistema.

## Descripción del Proyecto.

El proyecto consiste en una aplicación web dirigida a las personas que trabajan y estudian en institutos como el IES de Alquerías. Estaaplicación va a permitir a los usuarios acceder a su información personal, así como a diferentes funcionalidades según el tipo de usuario que sea, profesor, alumno, administrador y editor. La principal diferencia entre administrador y editor es que el primero tiene todos los permisos para cambiar datos en IDEApp y el editor será una ayuda para disminuir la carga del administrador, pero no con tantos permisos como éste.

De esta forma, los usuarios con el rol ‘alumno’ podrán consultar sus notas en las diferentes asignaturas, los profesores podrán consultar su departamento, asignaturas y alumnos. Asimismo, los administradores podrán gestionar los datos de la aplicación.

Ahora mismo, cuando pensamos en una plataforma dirigida a los centros educativos se nos ocurre pensar en “Plumier” y en el “Aula Virtual”. En el primero, desde la perspectiva de los alumnos solamente podemos consultar nuestras notas y matriculas tanto activas como pasadas. Por otra parte, en el “Aula Virtual” es donde podemos descargar y subir trabajos relacionados con el curso en el que estamos matriculados actualmente.

Además, a menudo los profesores eligen diferentes aplicaciones como “Classroom” de Google para cubrir esta necesidad. Por lo tanto, en un mismo curso los alumnos tienen que acceder y crear usuarios en varias aplicaciones sólo para poder disponer de los apuntes de todas sus asignaturas.

Como resultado de este problema, hemos decidido crear una aplicación que englobe todas estas necesidades, inspirada en el aula virtual y plumier. La funcionalidad de ésta es, entre otras cosas, la de subir archivos y descargarlos, acceso a la información de las asignaturas y ver/poner calificaciones.

Como hemos dicho anteriormente, en esta aplicación encontramos varios tipos de usuario, alumnos, profesores, administradores y editores. Todos ellos accederán al programa a través de un registro cerrado, por lo que los nombres de usuarios y contraseñas las suministrará el personal administrativo o, lo que es lo mismo, los que tengan el papel de “Administrador” o “Editor”.

Todo el mundo podrá ver la pantalla de inicio de la web, sin embargo, habrá que identificarse para mostrar el resto de las funcionalidades al usuario. Según el tipo de usuario, se puede hacer una cosa u otra, es decir, se puede navegar por diferentes pantallas de la página web. Esto lo veremos en profundidad en próximos puntos.

## Definición de los objetivos.

El objetivo principal del proyecto es crear una aplicación que mejore la experiencia de los alumnos y profesores del centro a la hora de tener que consultar datos, compartir documentos con el resto de los usuarios y subir o visualizar calificaciones.

Nuestra aplicación concentra las funcionalidades de varias aplicaciones dirigidas a la educación. Adaptándose mejor a las necesidades globales del centro y a su vez, a las individuales de alumnos y profesores, haciendo que esta tendencia a diversificar las plataformas que se utilizan en un mismo centro disminuya de tal forma que la información quede en un mismo sitio.

En el caso de los profesores, en un principio, estos podrán ver la información de su departamento y del resto, poner notas y subir documentos a la página. Más adelante se irán añadiendo mejoras a estas funcionalidades básicas.

IDEApp pretende llegar a más de un centro a la vez. Para asegurar que cada institución se sienta representada, se les ofrecerá el poder de personalizar la aplicación para que se adapte a sus exigencias y necesidades. Por eso se crearán URLs diferentes y un proyecto de Firebase individual (aunque con la misma estructura) para cada centro y así cada uno dispondrá de una página web diferenciada con su nombre, logo y colores característicos. Igualmente figurarán en la web enlaces a los distintos centros pertenecientes a una misma fundación (como es el caso de los colegios de Maristas).

Para que todo el mundo pueda acceder con diferentes dispositivos, hemos optado por una aplicación web que se adapte al tamaño de éstos. Por lo tanto, la estructura de ésta será ligeramente diferente de un dispositivo a otro, es decir, es un diseño web “responsive”.

## Definición del alcance y las restricciones del sistema.

Una vez que hemos analizado la aplicación y los objetivos, debemos pasar a examinar como vamos a alcanzar dichos objetivos.

#### Alcance

Al haber elegido una aplicación web, nos aseguramos de que los usuarios puedan acceder desde cualquier dispositivo con acceso a Internet sin importar el sistema operativo que use.

Asimismo, IDEApp no ocupa espacio en disco puesto que todo se aloja en un servidor externo, en este caso Firebase y no necesita ningún tipo de instalación puesto que los dispositivos vienen con un navegador instalado por defecto, por lo que el usuario no deberá estar demasiado preocupado por esto.

#### Restricciones

La principal restricción que genera optar por una aplicación web es no hay funcionalidades offline. Además, IDEApp no podrá estar en tiendas de aplicaciones como “App Store” y “Play Store”

En cuanto a las limitaciones con respecto a la aplicación en sí, una clara es que solamente está orientada al sistema educativo español, concretamente al sistema educativo de la Región de Murcia, en consecuencia, adaptarla a comunidades autónomas supondría un coste adicional, especialmente si tenemos en cuenta a las comunidades bilingües.

Finalmente, en cuanto a restricciones económicas, debemos considerar que Firebase solamente proporciona almacenamiento gratuito hasta cierto límite. Así que, conforme aumente el uso habrá que abonar mensualmente cierta cantidad para mantener los servidores.

## Beneficios.

Como se ha dicho anteriormente, los beneficios para los alumnos son que no tienen que estar registrados en diferentes aplicaciones para obtener el mismo resultado, tener los documentos o notas de las diferentes asignaturas.

En cuanto a los beneficios para los profesores son principalmente tener acceso a los datos de su departamento, poner notas de sus asignaturas y subir documentos. O lo que es lo mismo, concentrar lo que ofrece el “Aula Virtual” y “Plumier” en una sola aplicación donde la navegación es más intuitiva.

# Análisis del sistema.

En este apartado haremos un análisis para el desarrollo del proyecto, con la finalidad de entender las exigencias principales a desarrollar la aplicación web.

## Obtención de Información.

A raíz de nuestra experiencia como alumnado del IES de Alquerías en estos últimos años, hemos podido observar que era común cambiar de plataforma para la entrega de trabajos, visualización de notas y obtención de documentos como los apuntes de una determinada asignatura.

A su vez, como alumnos no disponíamos de una plataforma fija si queríamos compartir apuntes entre compañeros. Como, por ejemplo, el uso de “WhatsApp”, “Telegram” o “Gmail” para esto. Por otra parte, usábamos otras aplicaciones aparte del “Aula Virtual” como “Classroom” y “Edmodo”.

Todo esto nos ha llevado a la conclusión de que hacía falta una aplicación que cubriera estas necesidades y que a su vez sea fácil de usar. Igualmente, cuando IDEApp se ponga en marcha, se pondrá en un periodo de prueba de un año escolar donde los usuarios podrán enviarnos sugerencias de mejoras a desarrollar en un futuro.

## Recursos necesarios.

En primer lugar, para desarrollar esta aplicación necesitamos diferentes recursos. En este apartado describiremos los diferentes tipos y qué materiales son necesarios para llevar a cabo este proyecto.

Al ser el proyecto final, hemos optado por elegir tecnologías que no hemos llegado a dar en clase en profundidad. Así que, para entender como trabajar tanto con Angular como con Firebase, que utiliza bases de datos no relacionales, hemos dedicado un esfuerzo mayor que si hubiéramos elegido desarrollar la aplicación en JavaServer Faces con una base de datos MySQL.

### Recursos humanos.

En concreto, el proyecto ha sido desarrollado por dos personas que actúan como desarrolladoras y administradoras de éste.

Una vez que la aplicación se implemente en el IES de Alquerías, habrá que tener más personal para insertar datos. Es decir, múltiples personas dispondrán de cuentas de usuario del tipo administrador y editor, como el director del centro y personal administrativo, entre otros.

En el caso de la continuación del desarrollo de la aplicación como las mejoras o la adición de funcionalidades, se necesitaría uno o varios desarrolladores más para hacerlo.

### Recursos software.

Como hemos comentado en otros puntos, hemos usado Angular 9 y Firebase para desarrollar IDEApp. El primero es un framework open source desarrollado por Google y Firebase es una plataforma digital para facilitar el desarrollo de aplicaciones web o móviles, también de Google.

Esta combinación hace que el rendimiento de la aplicación sea bastante bueno, ya que nos proporciona herramientas compatibles con Angular como “Firebase Performance Monitoring” con las que obtener un seguimiento del rendimiento personalizado y automático.

Para el uso de estos hemos tenido que instalar “Visual Studio Code” para el desarrollo con Angular. A su vez, hemos instalado “Node.js” y su administrador de paquetes NPM (Node Package Manager). Al usar Angular, mediante NPM, se instaló “@Angular CLI” que facilita el desarrollo de aplicaciones web.

En el caso de Firebase, lo principal es tener una cuenta de Google e iniciar sesión en su [página](https://firebase.google.com/?hl=es). Además, se ha usado “GitHub” para la sincronización del código fuente, “Bootstrap” para facilitar el desarrollo de la estructura de la interfaz de usuario y “Font Awesome” para los iconos, como los de las redes sociales.

Sin embargo, todo esto se verá con más detenimiento en el apartado de “[Implementación](#_Implementación.)”.

### Recursos hardware.

Como es de esperar, además del software hay que usar hardware. Para el desarrollo del proyecto ha bastado con usar los ordenadores que tenemos. Ambos tienen algunas de las siguientes características:

* Windows 10 de 64 bits
* Una memoria RAM entre 8 y 16 GB.
* Memoria ROM mínima de 265 GB, aunque en realidad el proyecto no ocupa toda esta cantidad.
* Procesador Intel Core i7-8550U.

Al ser un desarrollo para una aplicación web, no ha ocupado muchos recursos hardware al igual que para los usuarios, no necesitan nada excepto un dispositivo con conexión a internet y un navegador web.

## Planificación y Estudio de Viabilidad.

En este apartado detallaremos las diferentes fases del proyecto. Es decir, una breve explicación de la planificación que se ha llevado a cabo.

### Identificación y temporalización de fases de proyecto.

Para el desarrollo de la aplicación nos encontramos con diferentes etapas, las cuales se encuentran en el siguiente listado:

* Análisis de la estructura del proyecto.
* Formación.
* Diseño de la aplicación.
* Implementación.
* Documentación.
* Mantenimiento y futuras ampliaciones.

Análisis de la estructura del proyecto

En esta primera fase se investigan las diferentes tecnologías que se podrían usar para la aplicación, al igual que la estructura de esta. Es decir, las diferentes pantallas y otros elementos a considerar para obtener una interfaz adecuada con el “back-end”. Así mismo, ver si el proyecto, tal y como lo teníamos planteado, era viable.

En cuanto a las tecnologías que se barajaron, varían desde Java, una tecnología de back-end con la que hemos trabajado de forma consistente durante los últimos dos años, hasta JavaScript o Angular, tecnologías “front-end” que, hasta el momento hemos visto brevemente. Al empezar a pensar en esto, nuestra primera opción fue usar Firebase como almacenamiento de datos y Node.js, siendo un motor de ejecución de JavaScript.

Todo esto, nos llevó a analizar los pros y contras de elegir una base de datos no relacional y tecnologías que no dominábamos. Sin embargo, tras una deliberación optamos por elegir las tecnologías más novedosas. Esta sería Angular, basado en Typescript, que a su vez nace de JavaScript. Esto supondría dedicar un tiempo específico para la formación, fase que explicaremos más adelante.

Sin embargo, la estructura de la aplicación supuso un mayor coste de tiempo puesto que al ser compleja se necesitaba tiempo para la maduración de conceptos y aclaración de estos. En el apartado “[Documentación](#_Documentación.)” explicaremos la estructura de cómo es en una base de datos no relacional como es el caso de “Firebase” o más concretamente, un servicio que nos ofrece, “Realtime Database”.

Documentación

La siguiente fase es la de documentar todo lo referente a la aplicación. Como veremos en el diagrama de Gantt, esta fase se ha ido desarrollando de manera paralela al resto a lo largo de todo el proyecto. La fase de documentación conlleva, además de esta memoria, a comentar el código para un mejor entendimiento. Siendo esto último una gran ventaja puesto que el proyecto se hace entre dos personas.

Las diversas actividades que se llevan en esta etapa son parecidas, es decir, se basan en el análisis de los contenidos a explicar y detallarlos con la mejor claridad posible.

Formación

En esta importante fase, nos pusimos a investigar diferentes maneras de aprender sobre el tratamiento de datos para su almacenamiento y la manera de desarrollarlo.

En primer lugar, mientras hacíamos cursos prácticos de Firebase, hacíamos la estructura de la base de datos orientada a una relacional. Entre los que hemos realizado, se encuentra “Firebase y Angular” de “OpenWebinars” que, a su vez, nos introdujo a Angular. Lo cual nos llevó a pensar que “íbamos por el buen camino”.

De manera similar, hicimos cursos de “Angular” en la misma plataforma además de otros encontrados en otras como puede ser “YouTube”.

Todo ello nos llevó un poco más de lo esperado puesto que en esta fase nos encontrábamos con el inicio del estado de alarma debido al COVID-19 que, en nuestro caso supuso tener que detener el desarrollo del proyecto.

Implantación

Esta fase fue casi en paralelo con la anterior para poner en práctica nuestros conocimientos sobre Angular y Firebase. También se hacia la adaptación de una base de datos relacional como lo era en su origen, “MySQL”, a una no relacional como lo es “Realtime Database” de Firebase.

La primera actividad de esta etapa era hacer la estructura del proyecto que, siendo desarrollada en Angular, se divide en componentes y servicios que explicaremos en la parte de “[Implementación](#_4._Implementación)”. Esta actividad se llevó a cabo de diferentes maneras, es decir, se crearon diferentes estructuras y versiones de estas hasta dar con la que tenemos actualmente.

Una vez teniendo la estructura adecuada, la siguiente actividad era empezar a implementar todo el “back-end” de la aplicación y al mismo tiempo hacer una interfaz de usuario adecuada.

Mantenimiento y futuras ampliaciones

En esta última fase se llevará a cabo una vez puesta en marcha la aplicación. En otras palabras, consiste en hacer que el proyecto sea más eficiente y minimización de los posibles errores que puedan dar en el futuro.

En el caso de que “IDEApp” llegue a la fase de producción, es decir, llegue a usarse en el día a día, esta etapa de mantenimiento y futuras ampliaciones es la más importante. Además de que, en ese caso, se necesitarían recursos para costear el mantenimiento de datos en Firebase y personal para crear las ampliaciones.

### Estimación de la duración de las actividades.

La actual sección puede dividirse en una estimación del tiempo que las actividades y fases hubieran durado y la que en realidad es debido a la situación generada por el COVID-19.

En primer lugar, teniendo en cuenta la fase del análisis de la estructura del proyecto, la formación y la realización de esta estructura pudo haberse hecho en un mes puesto que teníamos suficientes herramientas para llevarla a cabo.

Siendo un total de alrededor de tres meses, nos quedarían dos para la implementación de la aplicación y la documentación de esta.

Sin embargo, tras la declaración del estado de alarma en marzo, se generó un clima de incertidumbre respecto a cómo se realizarían los proyectos finales y las prácticas obligatorias. Durante este periodo no se pudo continuar con el proyecto puesto que se generaron propuestas alternativas con diferentes modos de evaluación que afectaban de manera directa a cada miembro del equipo. Finalmente, elegimos continuar con el desarrollo de IDEApp, aunque, al disponer de un menor tiempo nos supuso el replanteamiento de la aplicación y acelerar los tiempos de finalización de las tareas que nos habían asignado.

### Estimación de los costes de las actividades.

Puesto que el proyecto está basado en tecnologías “open source” y gratuitas, los costes en cuanto a esa parte son mínimos.

Por otra parte, nos encontramos con los costes del material utilizado, un portátil, un ordenador de sobremesa y un par de pantallas auxiliares. El costo de la energía que estos materiales hace que se incremente el coste total.

Asimismo, tenemos el caso de que son dos personas las que participan en el desarrollo de “IDEApp” por lo que, en cuanto al coste de recursos humanos también se ven afectados. Para calcular la tarifa por hora hemos tomado el salario mínimo por hora de 2020.

Un cálculo aproximado sería el siguiente:

|  |  |  |  |
| --- | --- | --- | --- |
| Coste aproximado | Euros | Cantidad (por persona) | TOTAL |
| Materiales | 850 € |  | 850,00 € |
| Luz, Internet | 50 € | X 3 meses | 150,00 € |
| Recursos humanos | 7,43 € /hora | X 70 horas | 520,10 € |

Todo esto hace un total de 1520,10€ por persona. Al ser dos las que desarrollan el proyecto, el resultado final sería de 3.040,20€.

### Diagrama Gantt.

Vamos a mostrar el diagrama de Gantt, en la siguiente página, con la planificación de tiempo para las fases y tareas que se describieron en apartados anteriores. Para ello, hemos hecho una lista con estas actividades y etapas y el tiempo estimado de la realización de cada una de ellas.

### Constitución del equipo y asignación del trabajo correspondiente.

Este apartado es clave, puesto que el proyecto no lo hace una sola persona. Aunque en algunas partes tenemos las mismas tareas, en otras son diferentes para poder abarcar más apartados en la implementación de la aplicación.

Algunas tareas como la estructura de la base de datos se dividen en una parte básica en las que colaboran ambas partes, pero cuando se trata de hacer algo más concreto, como el Entidad/Relación se ha encargado Estíbaliz.

Por otra parte, en cuanto a la adaptación de una base de datos relacional a una no relacional y la estructura del proyecto en Angular la realizó María.

En cuanto a la implementación, ambas hicimos nuestra parte de la aplicación. En algunas partes teníamos que programar de manera conjunta, pero en otras nos centramos en nuestras respectivas tareas para así dar más funcionalidades a “IDEApp”.

# Diseño del sistema.

En esta sección de la memoria, nos extenderemos más en cuanto a la estructura de los datos de la aplicación, cómo se ha diseñado la interfaz y el diseño de procedimientos para conocer el flujo de interacción de “IDEApp”.

## El diseño de los datos. Diagrama Entidad/Relación.

Ahora vamos a poner el planteamiento del que surge el E/R que veremos más adelante:

Tenemos, por lo tanto, dos tipos de usuarios principales, alumno y profesor. Ambos tendrán un email y contraseña propia para acceder a esta aplicación y, aparte, se guardará su información personal. Además, tenemos a los administradores y editores que serán el personal de secretaria y el equipo directivo para controlar la aplicación. La diferencia entre estos dos es, básicamente, los permisos y que estos últimos están para disminuir la sobrecarga de trabajo de los administradores.

Tanto alumnos como profesores pueden subir documentos a la plataforma.

Respecto a los primeros, éstos se agruparán formando cursos (ej. 1 DAM, 2 DAM, etc.) en un determinado año académico (ej. 2019/2020). También estarán matriculados en diferentes asignaturas en sus años académicos correspondientes y tendrán sus respectivas notas por evaluación.

Por otra parte, los profesores pueden impartir varias asignaturas y ser tutores de un grupo de alumnos, además de que pertenecen a un determinado departamento.

Al tener que dividir a los usuarios en diferentes roles, tenemos dos estructuras diferentes. Una para la relación con entre los usuarios y los papeles que pueden tener y otra con estos mismos y la aplicación.

Como podemos observar en la Figura 1 tenemos al “Usuario” como elemento principal, el cual puede ser “Profesor” o “Alumno”. Hemos dejado a un lado al personal administrativo y equipo directivo puesto que son estos los que van a administrar la aplicación, es decir, crear usuarios, cursos y asignaturas, entre otras cosas. Por otra parte, tenemos a los alumnos y profesores que son los que realmente interactúan con la aplicación en el contexto que son los que pueden subir documentos o subir/ver notas dependiendo de su rol.

Además, en esta Figura 1, para poder ver mejor la estructura de la Figura 2, hemos decidido poner a “Documento” ya que es una funcionalidad común que tienen.

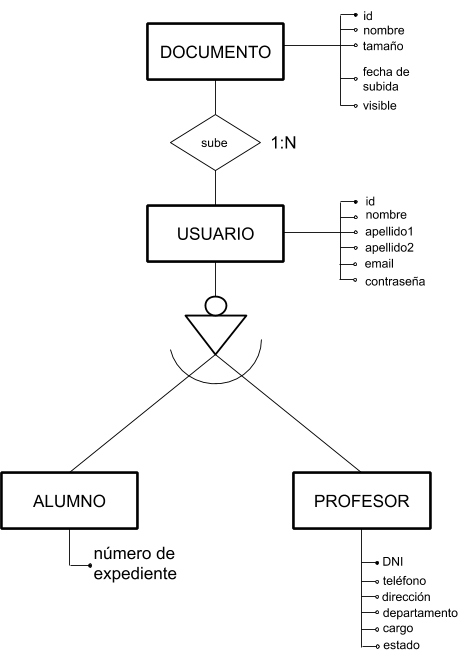


Figura . Relación entre los usuarios.

Por otra parte, tenemos la Figura 2, la cual es más compleja que la primera puesto que los elementos del sistema educativo están relacionados todos entre sí.

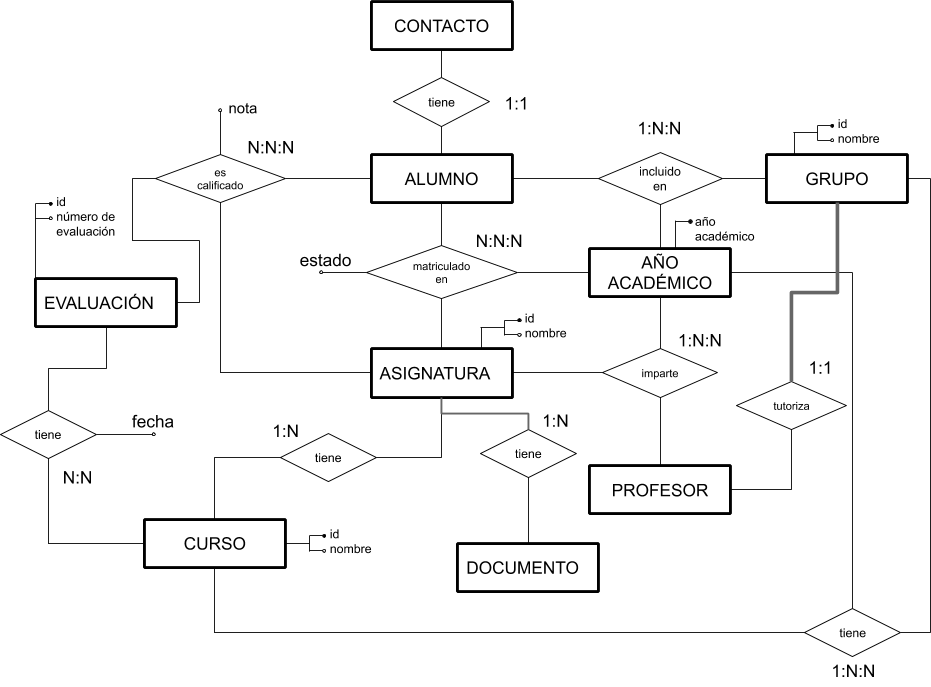


Figura . Modelo Entidad/Relación.

## Diseño de la Interfaz.

Para el diseño de la interfaz, destacamos el uso de Bootstrap para que IDEApp se adapte a todo tipo de pantallas. Puesto que al ser una aplicación web debemos tener en cuenta que los usuarios pueden acceder a ella desde cualquier dispositivo.

Esta aplicación está basada en dos colores, blanco y azul, siendo este último el que caracteriza al IES de Alquerías. Además, hemos creado un logo que junta el nombre del centro con “App” y así hacer la plataforma propia de éste.

Durante esta sección iremos por las diferentes pantallas, algunas son iguales para todos los tipos de usuarios, pero al tener diferentes permisos estas mismas pueden sufrir cambios.

Para empezar, veremos la pantalla principal, como se muestra en la Figura 3, todo el mundo puede acceder a ella, incluso los que no han iniciado sesión. Esto es una ventaja para que en un futuro podamos poner noticias. Aquí nos encontramos con una imagen del centro y un enlace a los tweets en tiempo real. En la parte de arriba tenemos una barra de navegación que irá cambiando según el rol que tenga el usuario en ese momento. Iremos viendo las diferencias en las siguientes pantallas.



Figura . Pantalla de inicio.

Continuamos por las asignaturas, aquí los estudiantes y profesores solamente tienen permiso para verlas. Por otra parte, tenemos al administrador y a los editores que son capaces de añadir, modificar y eliminar las que ellos quieran. Esto se puede observar en la Figura 4. Como podemos observar, en la barra de navegación, podemos tener diferentes enlaces dependiendo del rol del usuario.

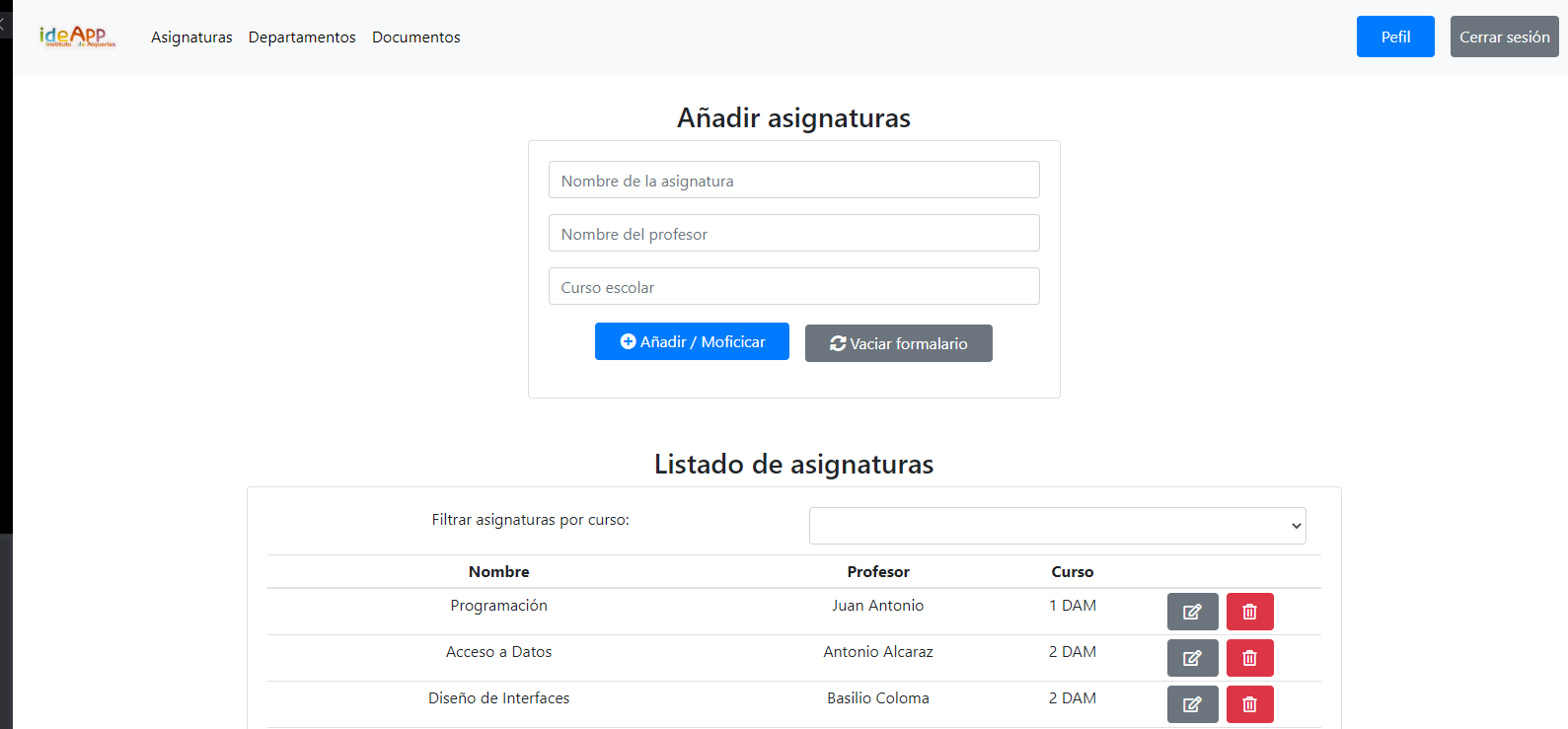


Figura . Pantalla de asignaturas según un administrador y un estudiante.



La siguiente pantalla es la de “Departamentos”, la cual es solamente accesible si eres profesor, administrador o editor. Además, estos dos últimos son los que pueden cambiar la información de la base de datos, es decir, si el jefe de departamento cambia, se puede modificar. En la Figura 5 tenemos un ejemplo de la vista de un administrador cambiando el nombre del jefe de departamento.

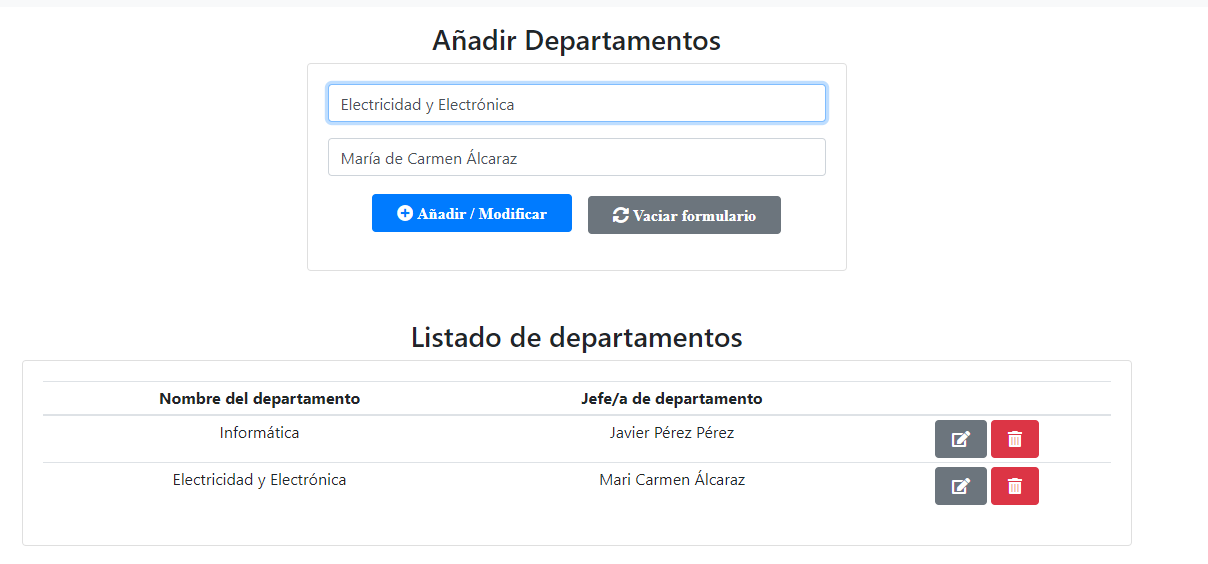


Figura . Pantalla de "Departamentos".

A continuación, vamos a mostrar la pantalla de documentos que es similar al resto. Con la única diferencia de que todos los usuarios menos los editores pueden subir archivos. La razón por la que los editores no tienen permiso para subir es que su papel es el de acelerar el proceso de inserción de datos como usuarios, cursos, asignaturas, entre otras cosas. En el caso de que surgiera esta necesidad, la añadiríamos en una futura actualización.

En la Figura 6 nos encontramos con esta sección de documentos con la sesión de un editor.



Figura . "Documentos" siendo un editor.

En cuanto a la navegación del usuario, quedaría la ventana donde se puede ver los datos personales, también llamada “Perfil”. Dependiendo de que usuario con el que hemos iniciado sesión, podemos acceder a cierta información y hacer ciertas tareas. Por ejemplo, siendo administrador o editor puedes crear usuarios y cursos dentro de esta ventana. Sin embargo, si eres profesor o alumno, solamente tendrás acceso a tu propia información. En la figura 8 tienes varias pantallas que corresponden a diferentes tipos de usuarios.

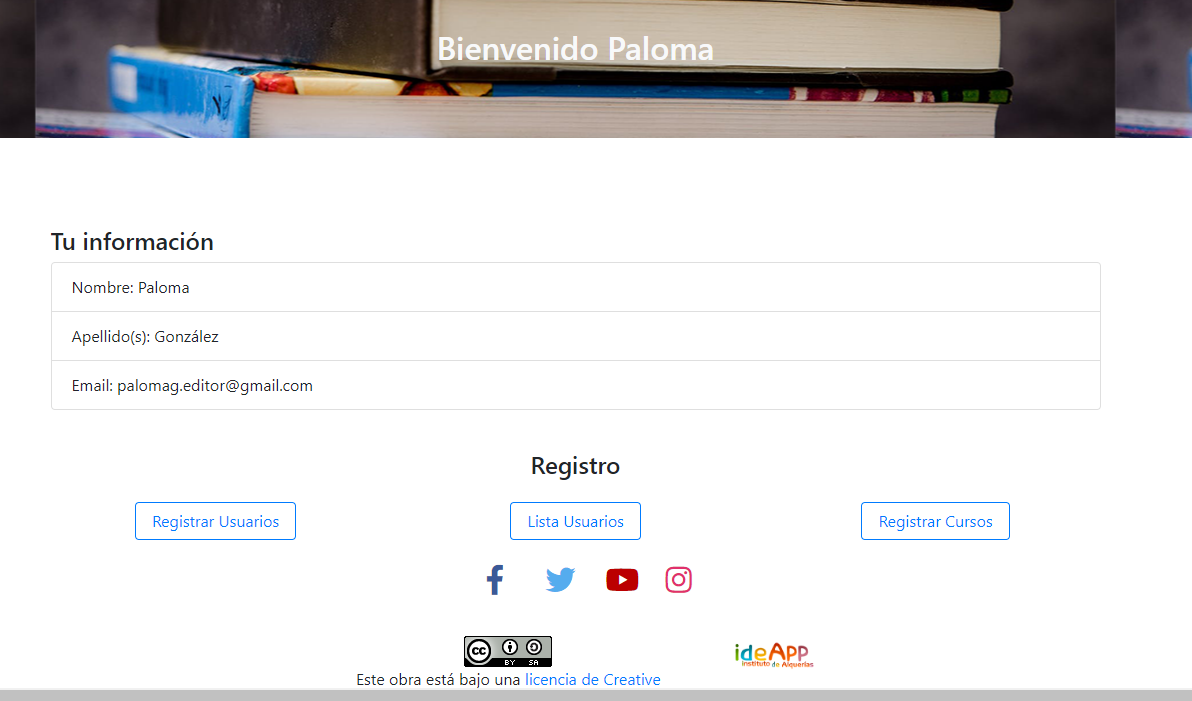


Figura . Perfil del usuario siendo administrador y un alumno.

A continuación, vamos a mostrar las pantallas acceso a la aplicación, el registro de cursos y el de registro y listado de los usuarios.

En la pantalla del acceso, Figura 8, tenemos simplemente un pequeño formulario.

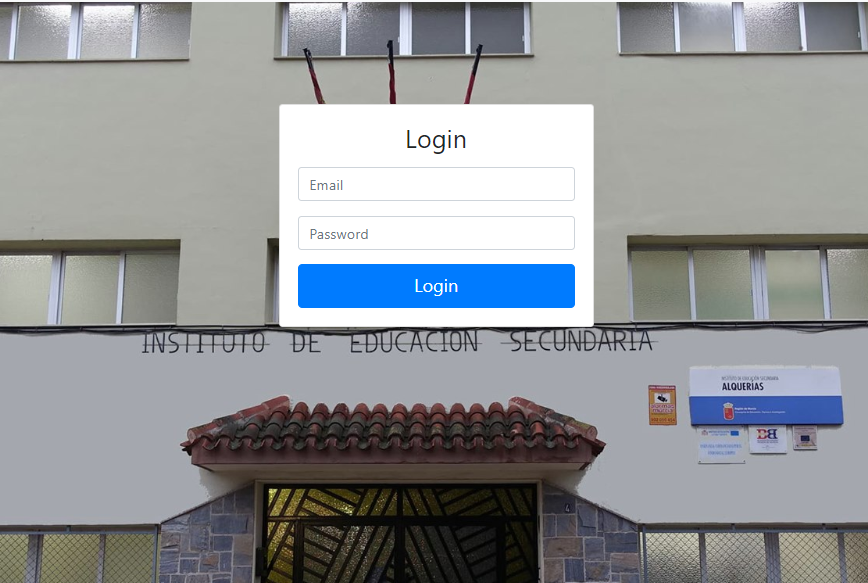


Figura . Pantalla de login

En cuanto al registro de los cursos, tenemos un listado de los cursos que ya hemos insertado en la base de datos y un pequeño formulario para insertar los nuevos. Como observamos en la Figura 9, esta pantalla es parecida a las pantallas de “Asignatura” y de “Departamento”.

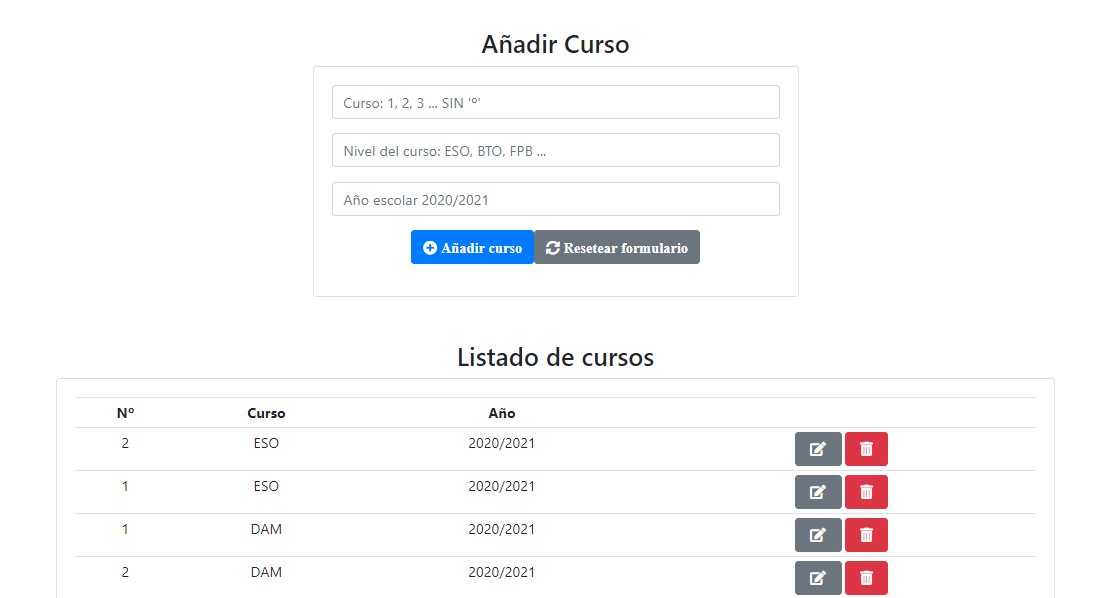


Figura . Registro de curso.

Continuamos por el listado de alumnos, como mostramos en la Figura 10, podemos filtrarlos y modificar la información. Sin embargo, no hemos dado la opción de eliminar los usuarios desde la web para evitar que se suprima un usuario de manea errónea. En el caso de que fuera necesario, un administrador puede acceder a “Realtime Database” que es la base de datos que aloja la información.

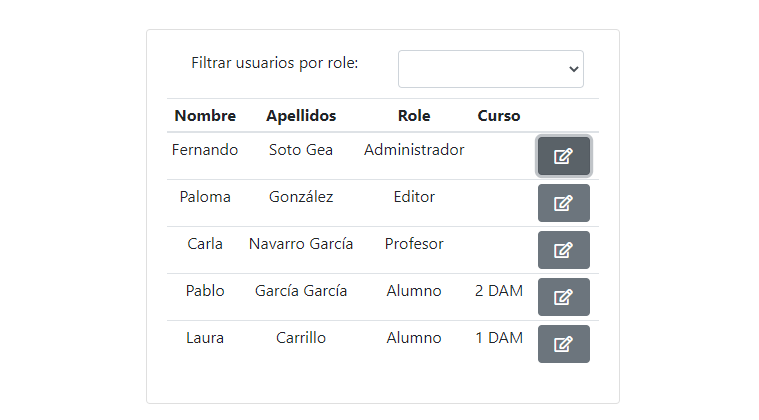


Figura . Lista de usuarios.

Por último, la pantalla más importante de la aplicación, el registro de usuarios. Éste tiene un papel importante porque puede crear a todo tipo de usuarios desde el administrador al usuario. Sin embargo, cuando se va a crear el perfil de una persona con el rol de administrador o de editor, aparecerá una alerta. En la Figura 11 tenemos el registro para un estudiante, un profesor y un editor (puesto que este último tiene el mismo formulario que el administrador).

Como hemos adelantado antes, cuando creamos un estudiante o un profesor e ingresamos un curso. Si éste no está en la base de datos, se crea automáticamente. Además, hay campos, aunque pocos, que no son obligatorios, como el contacto del segundo tutor del alumno y los segundos apellidos puesto que puede haber personas con un solo apellido.

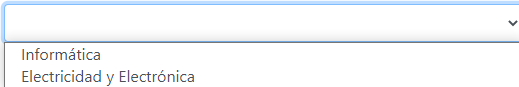
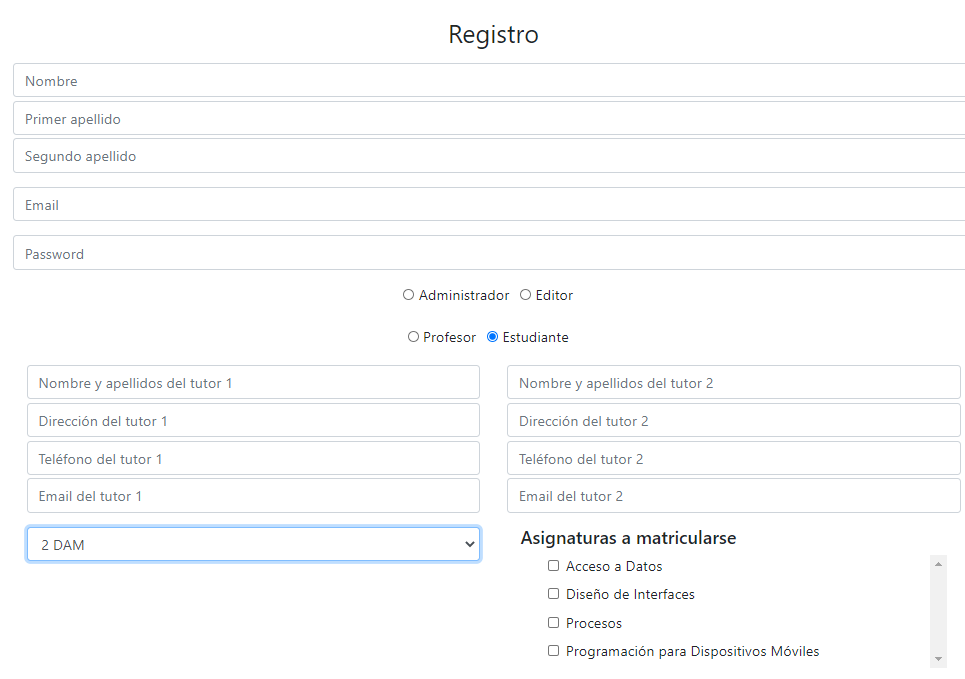
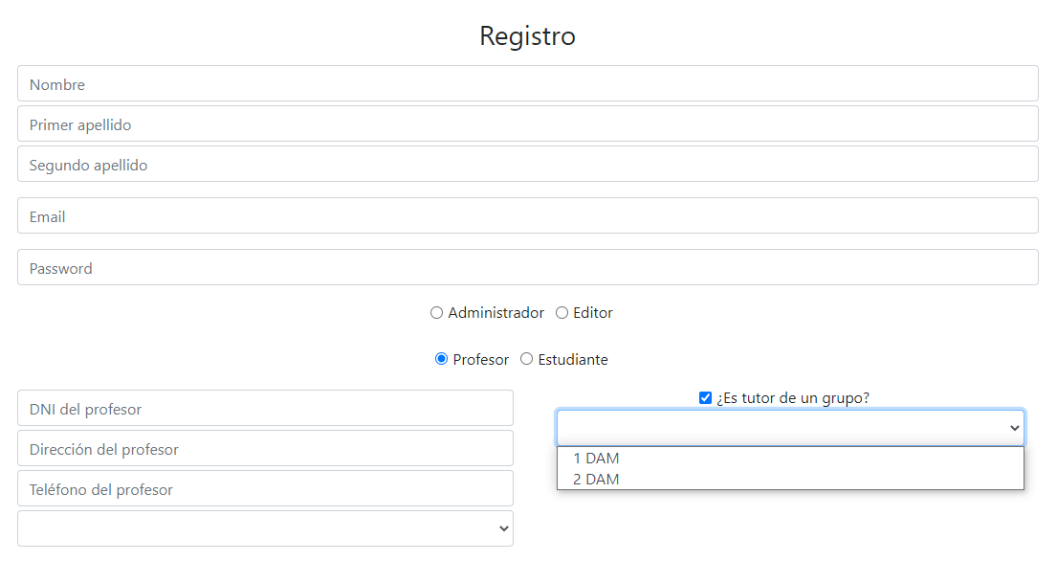
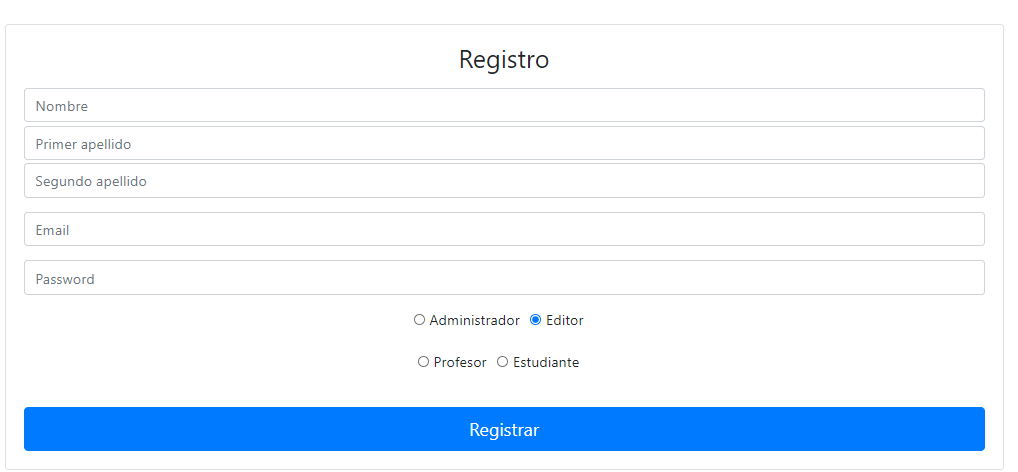


Figura . Registro de usuarios.

## Diseño de los procedimientos.

En esta sección mostraremos diferentes flujos de navegación y de funcionamiento que tiene la aplicación. Éstos son diferentes para cada tipo de usuario, puesto que dependiendo del papel que desempeñan tienen una funcionalidad u otra. Como hay muchos posibles flujos, en esta sección vamos a simplificarlos para que se pueda tener una idea más clara.

En primer lugar, nos pondremos en el papel de administrador y de editor, ya que este último es un “ayudante” del primero y ambos tienen el mismo flujo de navegación. Como se muestra en la Figura 12 tenemos las posibilidades de navegación entre las pantallas que tiene IDEApp.

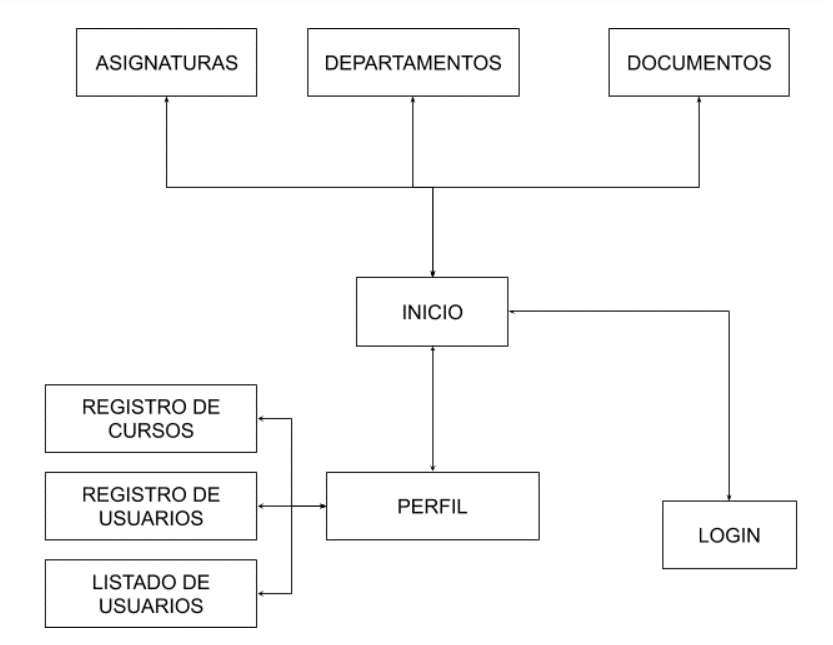


Figura . Flujo de navegación siendo administrador o editor.

Al tener enlaces en la barra de navegación de las pantallas de las asignaturas, documentos y departamentos al igual que el inicio (haciendo clic sobre el logo) podemos acceder a ellas en cualquier momento, muchas de estas relaciones no están marcadas, como de perfil y asignatura. Lo que también queda claro con esta imagen es que para poder hacer registros importantes como lo son los usuarios y los cursos necesitas acceder al perfil del usuario.

Asimismo, como hemos mencionado antes, la pantalla de inicio se puede acceder estando registrado o no.

Continuemos con la navegación, pero esta vez desde un usuario con el rol de estudiante, en la Figura 13 observamos que es más sencillo que el anterior puesto que, al no tener tantos permisos, las páginas que se pueden acceder son limitadas. Para el profesor sería lo mismo que la Figura 13 pero con acceso a la ventana de “Departamentos”.

Por otra parte, tenemos tres flujos de datos relacionados con Firebase puesto que este tiene tres servicios independientes para la autentificación de los usuarios, el almacenamiento de datos como las asignaturas y otro diferente para los documentos.



Figura . Navegación de IDEApp siendo un estudiante.

Primero veremos el flujo que tiene en la aplicación con “Authentication”, el servicio para la autentificación de usuarios. Este flujo va en paralelo con “Database”, la base de datos donde se guarda la información de los usuarios, las asignaturas, departamentos,… En la Figura 14 tenemos la representación es este caso.

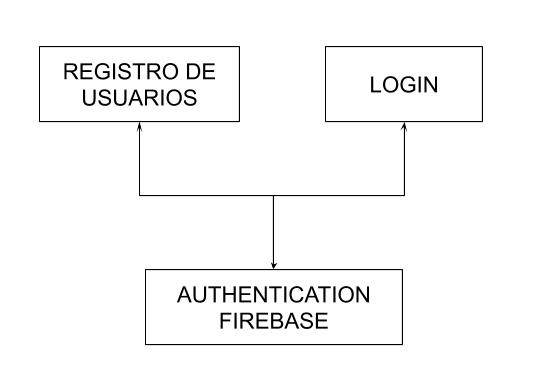


Figura . Servicio de Authentication.

También tenemos el servicio de “Storage” el cual nos permite subir documentos a diferentes carpetas, dependiendo de la asignatura. Como mostramos en la Figura 15, es un flujo sencillo. Este servicio va también en paralelo con el servicio de base de datos que nos ofrece Firebase.

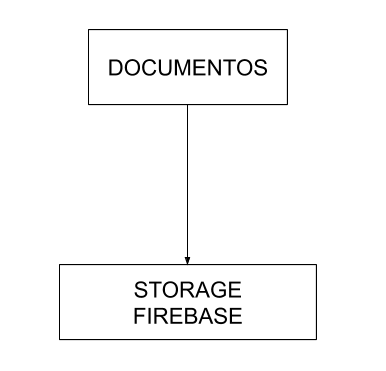


Figura . Servicio de Storage.

Como hemos mencionado antes, los dos anteriores y este que vamos a presentar, el de “Database” van en paralelo puesto que es el encargado de guardar la información de toda la aplicación. En la Figura 16 tenemos en cuenta tanto los servicios de Firebase como las pantallas de navegación de la aplicación.

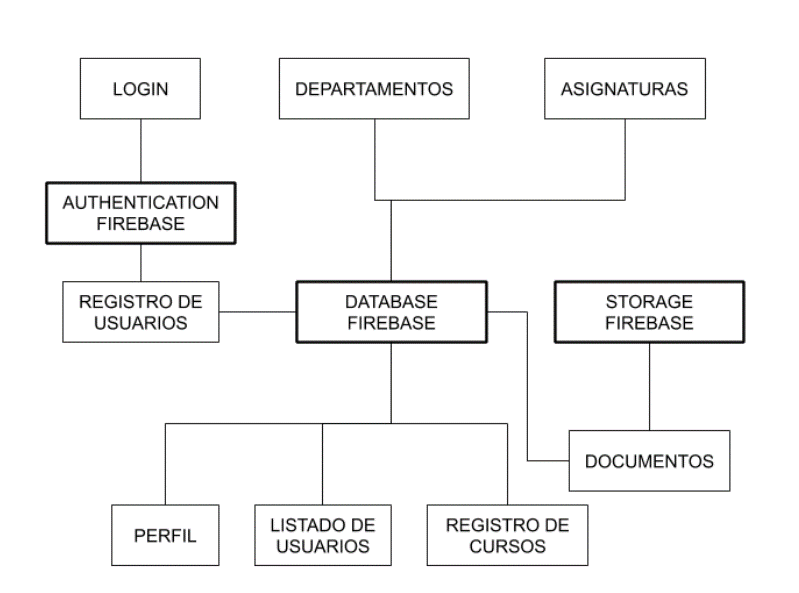


Figura . IDEApp y los servicios de Firebase

# Implementación.

En esta sección explicaremos como preparar el entorno para el desarrollo de la aplicación y las políticas de “backup”. Una vez sabido esto, instalaremos lo necesario y haremos pruebas.

## Preparación del entorno operativo.

Para la preparación de nuestro entorno operativo hemos optado por utilizar Firebase como proveedor de hosting, ya que nos ofrece un plan gratuito siempre y cuando no superemos los límites establecidos respecto a datos almacenados, lectura de documentos, etc. Otro beneficio es que permite trabajar remotamente en la aplicación como copropietarias desde nuestras respectivas cuentas de usuario de Google, por lo que no ha hecho falta la creación de una cuenta exclusiva para desarrollar el proyecto. Además, usando Firebase hemos podido concentrar todas las necesidades de almacenamiento (hosting, base de datos, documentos) de la aplicación en una única plataforma.

Firebase nos permite utilizar JavaScript, Node.js, Android, iOS, Unity, C++ y nos ofrece servicios como:

* **Cloud Firestore** 🡪 Una base de datos flexible y escalable para la programación en servidores, dispositivos móviles y la Web desde Firebase y Google Cloud Platform.
* **Cloud Functions** 🡪 Permite ejecutar automáticamente el código de “backend” en respuesta a eventos activados por las funciones de Firebase y las solicitudes HTTPS. El código se almacena en la nube de Google y se ejecuta en un entorno administrado.
* **Authentication** 🡪 Proporciona servicios de “backend”, SDK fáciles de usar y bibliotecas de IU ya elaboradas para autenticar a los usuarios en tu app. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federada populares, como Google, Facebook y Twitter.
* **Firebase Hosting** 🡪 Proporciona hosting seguro y rápido para las aplicaciones web, el contenido dinámico y estático, y los microservicios.
* **Cloud Storage** 🡪 Un servicio de almacenamiento de objetos potente, simple y rentable construido para la escala de Google. Los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y descarga de archivos, sin importar la calidad de la red. Puede usarse para almacenar imágenes, audio, documentos y otros tipos de contenido generado por el usuario. En el servidor, se puede usar Google Cloud Storage para acceder a los mismos archivos.
* **Realtime Database** 🡪 Almacena y sincroniza datos en una base de datos NoSQL alojada en la nube. Los datos se sincronizan con todos los clientes en tiempo real y se mantienen disponibles cuando la app no tiene conexión.
* **Test Lab** 🡪 Una infraestructura de prueba de apps basada en la nube. Con una sola operación, se pueden probar las apps para Android o iOS en una amplia variedad de dispositivos y configuraciones, y ver los resultados (que incluyen registros, videos y capturas de pantalla) en la consola de Firebase.
* **Dynamic Links** 🡪 Permite implementar vínculos que funcionan exactamente como deben, en varias plataformas y sin importar que se haya instalado previamente la aplicación. Por ejemplo, si un usuario abre un Dynamic Link en iOS o Android, puedes ir directamente al contenido vinculado en la aplicación nativa. Si un usuario abre el mismo Dynamic Link en un navegador de escritorio, lo puedes conducir al mismo contenido en el sitio web.

Firebase nos permite centrarnos en el desarrollo de la aplicación, y delegar el funcionamiento y la infraestructura del servidor. Además, nos ofrece múltiples opciones para llevar a cabo el escalado de la aplicación en caso de crecimiento de esta, así como un dominio gratuito.

Los pasos para implementar Firebase en nuestra aplicación web son:

* + **Paso 0:** abrir el proyecto web y acceder a Firebase con una cuenta de Google.
  + **Paso 1:** Crear un proyecto de Firebase
  + **Paso 2:** Registrar nuestra app
  + **Paso 3:** Agregar los SDK de Firebase e inicializar Firebase. Para ello debemos:
    1. Instalar el paquete npm de Firebase y guardarlo en el archivo package.json usando:

npm install --save firebase

* + 1. Inicializar y añadir la configuración de Firebase en la app:

// archivos: environment.ts y environment.prod.ts

// dentro de la constante “environment”

firebaseConfig = {  
  apiKey: "***api-key***",  
  authDomain: "***project-id***.firebaseapp.com",  
  databaseURL: "https://***project-id***.firebaseio.com",  
  projectId: "***project-id***",  
  storageBucket: "***project-id***.appspot.com",  
  messagingSenderId: "***sender-id***",  
  appID: "***app-id***",  
};  
  
// Initializar Firebase, document: app.module.ts  
AngularFireModule.initializeApp(environment.firebaseConfig);

* **Paso 4:** Instalar CLI y, luego, implementar en Firebase Hosting. Para implementar en Firebase, demos usar [Firebase CLI](https://firebase.google.com/docs/cli?hl=es-419), una herramienta de línea de comandos. Podemos instalarla desde cualquier punto de nuestro ordenador usando:

npm install -g firebase-tools

Una vez instalado debemos:

* + 1. Acceder a Google y ejecutar :

firebase login

* + 1. Inicializar el proyecto de Firebase, para ello ejecutamos desde la raíz del directorio local de apps:

firebase init

Este comando de inicialización hace lo siguiente:

* + - * Vincula el directorio local de la app con Firebase.
      * Genera un archivo firebase.json (que es necesario para Firebase Hosting).
      * Solicita que especifiques un directorio raíz público que contenga los archivos estáticos públicos (HTML, CSS, JS, etc.).
    1. Implementa la configuración de hosting y contenido en Firebase Hosting. De forma predeterminada, cada proyecto de Firebase tiene subdominios gratuitos en los dominios web.app y firebaseapp.com. Para realizar la implementación, debemos ir al directorio raíz del proyecto y ejecutar:

firebase deploy

* + 1. Ya podemos acceder a cualquiera de los sitios web predeterminados: “project-id.web.app” y “project-id.firebaseapp.com”
  + Acceder a Firebase en la app. El SDK de Firebase admite:

Por otro lado, como el desarrollo de la aplicación se realizará desde dos computadoras diferentes, hemos optado por la creación de un repositorio privado alojado en GitHub. Al igual que Firebase, GitHub ofrece una versión gratuita llamada ‘GitHub Free’ con funcionalidades básicas y una suscripción mensual que nos daría acceso a la versión ‘Pro’ con más funcionalidades como la creación de ‘wikis’ para los diferentes repositorios o un mayor espacio de almacenaje. En este caso, al ser un proyecto desarrollado por un equipo de dos personas hemos optado con continuar con la versión gratuita ya que cubría perfectamente nuestras necesidades.

Para empezar a trabajar en el desarrollo del proyecto usando GitHub debemos seguir los siguientes pasos:

* **Paso 1:** Crear una cuenta de GitHub para cada miembro del equipo.
* **Paso 2:** Crear un nuevo repositorio. Solo debe crearse un repositorio para el proyecto, el resto del equipo podrá acceder al código como ‘colaborador’. En este caso, como ya hemos mencionado, hemos optado por un repositorio privado.
* **Paso 3:** Subir los archivos locales del proyecto a GitHub. Esto puede hacerse fácilmente haciendo clic en el botón ‘upload files’ y arrastrando los archivos directamente. Finalmente, cuando tengamos todos los archivos, debemos escribir un mensaje que quedará reflejado junto al ‘commit’, seleccionar la rama en la que queremos que se guarden los archivos, en este caso la rama ‘Master’ y pulsar en ‘Commit Changes’
* **Paso 4:** Añadir colaboradores al repositorio. Desde éste, debemos hacer clic en ‘Settings’ y seleccionar ‘Manage Access’. Una vez en la pestaña, hacemos clic en el botón ‘Invite Colaborator’ y escribimos el email o el nombre de usuario de GitHub de la persona que queremos invitar al proyecto. Cada posible colaborador, recibirá una invitación en su email que deberá aceptar para participar en el repositorio.

## Definición de políticas de backup.

Firebase permite a los usuarios del plan de pago ‘Blaze’ configurar copias de seguridad automáticas de Firebase Realtime Database, una función de autoservicio que permite crear copias de seguridad diarias de los datos de la aplicación de Database y de las reglas en formato JSON y almacenarlas en un depósito de Google Cloud Storage.

Sin embargo, al haber optado por el plan ‘Spark’ que es gratuito, mientras se mantenga, deberá de haber un encargado de exportar semanalmente las bases de datos de Firebase y almacenar los datos de manera local. Así si ocurriera un error o quisiéramos restaurar una base de datos anterior, el encargado deberá borrar la base de datos activa y sustituirla por el backup que deseemos.

Desde el punto de vista como desarrolladores, sabemos realizar los backups de manera manual no es lo más ideal, pero es necesario para mantener el coste a 0 del desarrollo de la aplicación. Una vez que la aplicación este desplegada y sea necesario cambiar del plan ‘Spark’ al ‘Blaze’ se procederá a la automatización de las copias de seguridad de las bases de datos.

## Instalación de entorno de desarrollo y prueba.

Nuestro entorno de desarrollo y prueba se ha llevado a cabo en un sistema operativo Windows 10 en las versiones Home y Pro. Para desarrollar nuestra aplicación web , hemos optado por utilizar las siguientes herramientas:

* Node.JS es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Para utilizarlo debemos descargarlo de la siguiente dirección e instalarlo: <https://nodejs.org/download/release/v12.16.1/> En nuestro caso hemos seleccionado la versión Windows LTS 12.16.1 x64. Hemos elegido esta versión ya que se encuentra como LTS activa y continuará así hasta el 30 de abril de 2022
* Angular 9 como plataforma para desarrollar aplicaciones web. Como ya hemos instalado Node.JS, para instalar angular solo tenemos que abrir un terminal y escribir el siguiente comando : npm install -g @angular/cli
* Visual Studio Code como IDE de desarrollo web, que nos permite codificar y nos avisa de errores en el código antes de compilar. Pinchando en el enlace podemos descargamos la última versión estable para Windows: <https://code.visualstudio.com/>. Una vez descargado ejecutamos el archivo “.exe” y lo instalamos en el equipo.
* Para facilitar el trabajo hemos instalado dentro de Visual Studio dos extensiones. Este paso es opcional. Para acceder a las extensiones de Visual Studio, con el programa abierto podemos pulsar Ctrl+Shift+X o acceder pulsando el icono situado en la barra lateral izquierda. Una vez en la pestaña de extensiones nos aparecerán las recomendadas y habilitadas. En primer lugar, buscamos e instalamos ‘Beautify’, esta extensión nos permite de forma rápida formatear el código para que sea más legible. En segundo lugar, buscamos e instalamos la extensión ‘GitHub Pull Requests and issues’ que nos va a permitir revisar y gestionar ‘pull requests’ y cuestiones de GitHub directamente en Visual Studio Code. Esta última extensión ha sido muy útil a lo largo de la implementación del proyecto, puesto que nos ha permitido mantener el código actualizado de forma rápida.

# Prueba.

En este apartado procederemos a ver las actividades realizadas para identificar posibles errores de funcionamiento o usabilidad de IDEApp. Realizar pruebas a lo largo del desarrollo de la aplicación es necesario para corregir los errores y entregar un producto de buena calidad.

## Indicadores de calidad para realizar la evaluación.

En este punto vamos a conocer los indicadores que hemos ido siguiendo durante el desarrollo del proyecto. Para asegurarnos de crear una web ‘user-friendly’ para los usuarios hemos tenido en cuenta los estándares de usabilidad web y universal. Por ello, hemos intentado crear una web con una interfaz clara que se ajuste a la resolución del dispositivo desde el que se visualice.

En las siguientes tablas veremos las pautas que hemos intentado seguir a la hora de desarrollar la aplicación:

|  |
| --- |
| DISEÑO GENERAL |
| Interfaz responsive, se adapta al tamaño de la pantalla |
| Los elementos tienen el mismo aspecto y funcionamiento en todas las páginas del sitio. |
| Se mantiene una interfaz consistente |
| Los elementos se colocan siempre en el mismo lugar |
| Botones con forma de botón similares en todas las páginas |

|  |
| --- |
| NAVEGACIÓN |
| Hacer que las tareas más comunes de tus usuarios estén fácilmente disponibles. |
| Menús cortos |
| Logotipo: a la izquierda con enlace a inicio. |
| Navegación: Parte superior, secciones. |
| No usar "door slams" |
| Evitar desperdiciar espacio en la página principal (Ej: botón "saber más") |

|  |
| --- |
| CONTENIDO Y LEGIBILIDAD |
| Utiliza colores que contrasten texto y fondo. |
| Usar lenguaje plano, no literario. |
| Utiliza tamaños de letra suficientemente grandes. |
| No muevas el texto, ni lo escales ni hagas que parpadee. |
| Utilizar cabeceras (h1, h2, h3…). |
| Utilizar títulos y subtítulos. |
| Usar tipografías coherentes |
| No a los iconos excesivos ni tipografías extrañas |
| Dividir textos largos en varias secciones. |

|  |
| --- |
| TRATAMIENTO DE ERRORES |
| Pedir la confirmación de operaciones peligrosas |
| Compatibilidad con varios navegadores |
| Deshabilitar/Ocultar controles que realicen acciones no permitidas |
| Evitar controles muy pequeños y menús con muchas opciones |

|  |
| --- |
| REDACCIÓN DEL CONTENIDO |
| Lenguaje sencillo y directo |
| Evitar estructuras complejas, palabras largas y conceptos abstractos. |
| No utilizar el subjuntivo, pasivas, el lenguaje metafórico. |

## Elaborar una batería de pruebas para detectar errores.

La fase de pruebas, también conocida como “testing” comprende las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto al cliente. Es una actividad más en el proceso de control de calidad que, como ya hemos dicho, debe realizarse de forma paralela al desarrollo de la aplicación.

En nuestro caso, hemos optado por difundir varias versiones de prueba cerradas en diferentes escalas a lo largo de un año escolar (9 meses). Una vez que la aplicación este teóricamente finalizada, pasaría a ejecutarse en un entorno real como versión Alfa. Cuando esto ocurra, incorporaremos a los primeros usuarios de nuestra aplicación, estos usuarios serán los alumnos y profesores que formen parte de una clase aleatoria del instituto de Alquerías, más dos personas que el centro seleccionará como administrador y moderador. Con esta prueba inicial podremos localizar problemas ocultos para poder corregirlos antes de llegar a la siguiente versión de la web.

Tras la versión Alfa, que estimamos que podría durar unos 3 meses, se pasaría a lanzar una versión Beta, más refinada, a la que se incorporarían nuevos usuarios progresivamente. El objetivo de esta versión será localizar y corregir errores que surjan al aumentar el tráfico de usuarios simultáneos en la web. La Beta cerrada se espera que finalice tras 6 meses de uso y que en ese momento ya tengan acceso todos los alumnos, profesores y personal administrativo del IES Alquerías. Tras esto, la app, pasará a estar ya en su versión candidata para el lanzamiento.

## Procedimiento para la participación en la evaluación de los usuarios finales.

A lo largo del proceso de pruebas, como hemos dicho, iremos incorporando usuarios a los que se les encargarán diferentes tareas según la fase en la que nos encontremos.

En la fase Alfa, los usuarios valorarán si los indicadores de calidad previamente expuestos se cumplen y podrán sugerir mejoras en los diferentes apartados de estos. Asimismo, los usuarios nos proporcionaran información sobre errores que encuentren en la aplicación y podrán dar su opinión sobre funcionalidades que les gustaría introducir en la web.

En la fase Beta, se espera añadir de forma paulatina nuevos usuarios que, al igual que en la fase anterior, nos proporcionarán los datos de los errores que encuentren y podrán hacer sugerencias para mejorar la aplicación.

# Documentación.

En la siguiente sección hablaremos en profundidad de la base de datos que hemos usado, su estructura y las colecciones (al ser una base de datos no relacional, es así como se estructura la información).

Por otra parte, hablaremos del código que lleva el proyecto. Pese a que está en el “Aula Virtual”, en un apartado de esta sección veremos cómo es la estructura del proyecto y los aspectos importantes para tener en cuenta.

## Estructura de Bases de Datos.

Lo primero de todo hay que saber que los datos se pueden guardar por colecciones, es decir, las asignaturas son una colección y los departamentos otra. Además, los datos se guardan en formato JSON por lo que todos los datos que se guardan son cadenas de texto.

Obviamente, en nuestra base de datos tenemos más colecciones. En la Figura 17 tenemos la lista de las colecciones que hemos usado. Hemos de **advertir** que el código del proyecto y la base de datos la hemos realizado en inglés para así hacer este proyecto más accesible a un público mayor puesto que hoy en día es lo más habitual en las empresas y en los proyectos públicos.



Figura . Colecciones de la base de datos.

En primer lugar, tenemos el nombre de la base de datos de la aplicación “ideapp-tfc-project”. A continuación, podemos ver las diferentes colecciones que conforman al final toda la información de la aplicación.

La primera colección son los cursos, cuyos datos se componen, como se muestra en la Figura 18, de un número de identificación único, y los campos que son: “course” el curso, en número, en el que están los estudiantes, es decir, 1º, 2º, 3º, etc., el nivel (ESO, BTO, FP Básica, …) y por último el año en el que se va a cursar. Esto último está pensado para tener guardado los diferentes cursos a lo largo de los años y su información. Además de una propiedad calculada que, al fin y al cabo, es la unión de los tres campos.



Figura . La colección de los cursos.

La siguiente colección que vamos a ver es la de departamentos o, como figura en la base de datos, “departments”. Al igual que el anterior, tenemos un identificador, que tiene Firebase por defecto, y cada departamento tiene sus propios datos como son el nombre del departamento y el nombre completo de la persona que está a cargo, es decir, el jefe/a de departamento. Un ejemplo claro es el que se muestra en la Figura 19.

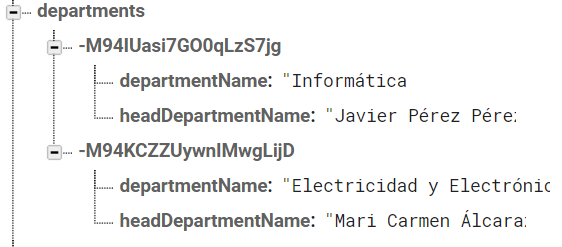


Figura . la colección de departamentos.

Antes de pasar a los documentos, vamos a ver las colecciones de asignaturas y usuarios. La parte de asignaturas es similar a las anteriores, tienen un identificador único y además tienen el nombre de esta, el profesor que la imparte y el curso al que se imparte. En la Figura 20 tenemos un ejemplo.

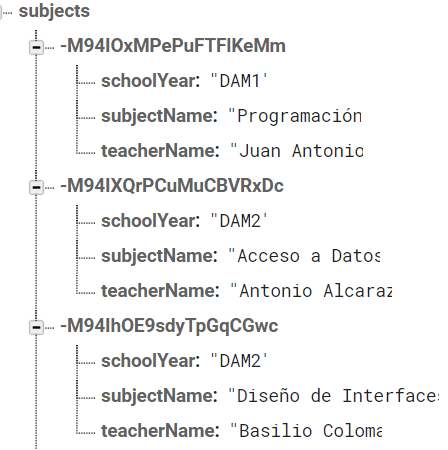


Figura . Colección de asignaturas.

En la colección de usuarios tenemos a todos los usuarios, tanto a los administradores y editores como profesores y alumnos. Estos se diferencian por una propiedad que tienen. En la Figura 21 podemos observar que dependiendo del rol que tengan, tienen unos datos u otros. Sin embargo, comparten propiedades comunes como el email, la contraseña, apellidos y nombre. En este caso, mostramos dos tipos de usuarios, en este orden, profesor y administrador.

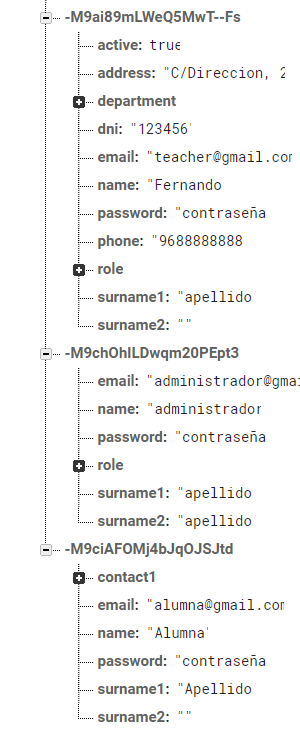


Figura . Colección de usuarios.

En el caso de los documentos, Figura 22, cada uno tiene un identificador propio y un campo, “url”, que es la ruta donde podemos encontrar el documento almacenado en “Firebase Cloud Storage”

A continuación, vamos a explicar la estructura que tiene la colección de matriculados o “sign-up”. Puesto que es el resultado de la relación entre dos colecciones. Es decir, haciendo referencia a la una base de datos relacional, si tenemos una relación N:N se crea una tabla nueva con las claves primarias de las tablas originales. En este caso, viene a ser parecido, creamos una colección con una primera “capa” de las “keys” (como las llama Angular y Firebase) o cadena de caracteres que refleja una identificación única de cada elemento de una de las colecciones originales. Después se crea una segunda “capa” con las “keys” de la otra colección.

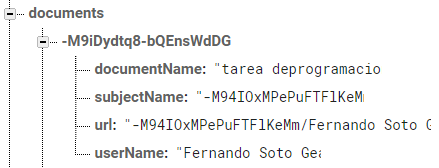


Figura . Colección de documento.

la colección “sign-up”, registrado o matriculado en español. En realidad, es una subcolección con los identificadores de las asignaturas que contienen el identificador de los alumnos que las cursan. Podemos observar en la Figura 23 que estos estudiantes son propiedades de objectos que están igualados a un booleano.

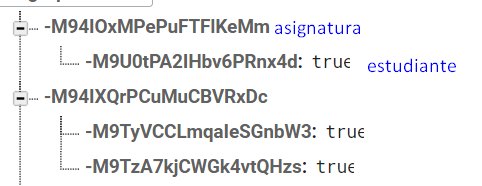


Figura . Colección de alumnos que cursan una asignatura.

## Código de Fuentes.

En esta sección vamos a explicar puntos para tener en cuenta sobre el código. En otras palabras, veremos la estructura del proyecto y qué elementos componen las carpetas más relevantes.

Como podemos observar en la Figura 25, nuestro código se divide bastantes carpetas, sin embargo, las que nosotras hemos desarrollado están dentro de la carpeta “src”. El resto de carpeta contienen los módulos/dependencias necesarias para que la aplicación funcione y haga uso de los servicios correspondientes. Vamos a explicar el funcionamiento de las carpetas que se encuentran en “src”, más concretamente, nos expandiremos más en la subcarpeta “app”, donde se encuentra todo nuestro código.

Queremos recordar que el proyecto está en inglés puesto que así lo hacemos más accesible para los que lo lean, ya que, hoy en día las empresas usan este sistema.

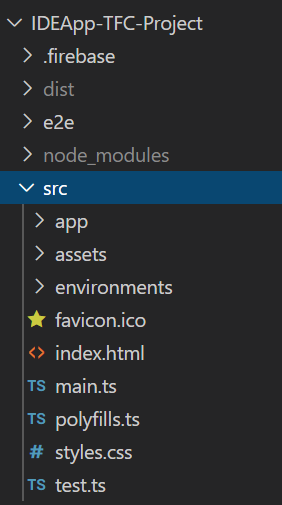


Figura . Estructura del proyecto.

Comencemos por la carpeta “enviroment”. Aquí se encuentran las variables de entorno como la variable para la comunicación entre Firebase y nuestro proyecto. En concreto, tenemos dos archivos, uno que usa las variables de entorno para cuando nos encontramos en desarrollo y otro para producción.

En la carpeta “assets” nos encontramos con todos los recursos, como imágenes que necesitamos. En un próximo punto, “[Licencias](#_Licencias.)”, pondremos las licencias que tienen.

A continuación, antes de adentrarnos en la carpera “app” hay que explicar que el proyecto funciona de la siguiente manera:

Tenemos un archivo “index.html” donde se encuentra la estructura básica de HTML. Esto es como se muestra en la Figura 25. Esta aplicación utiliza sus propios componentes, es decir, crea sus propias etiquetas HTML y así solamente se crea “una” página.

En la carpeta “app” nos encontraremos infinidad de componentes como el de la línea 17 de la Figura 25.

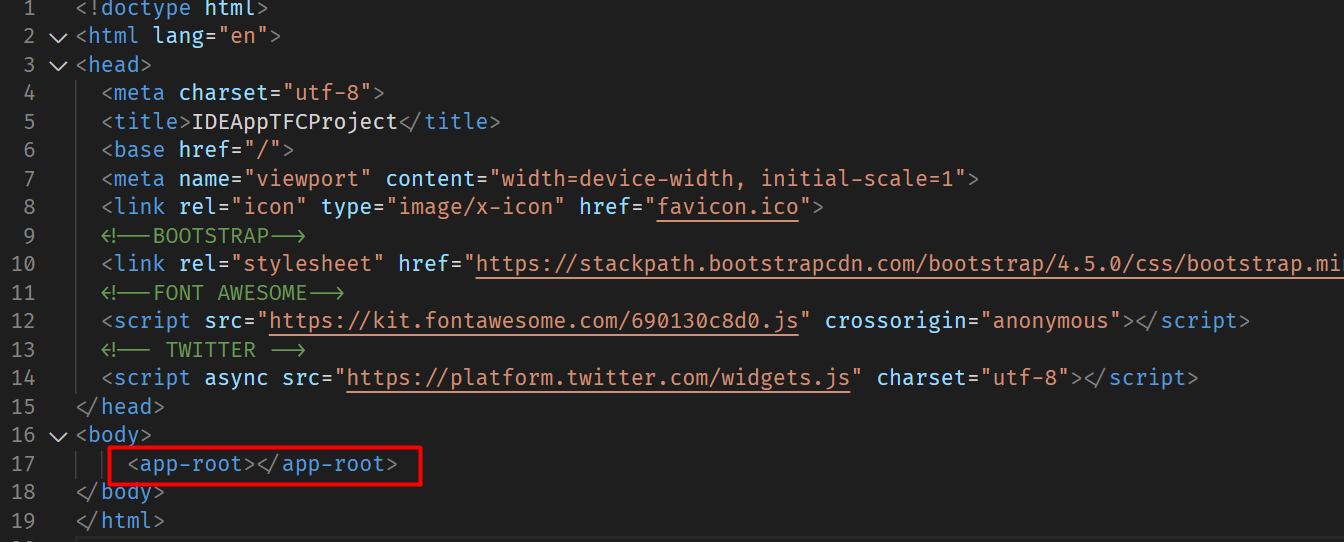


Figura . "index.html".

Como podemos observar en la Figura 26, tenemos la carpeta “app” dividida en subcarpetas que se corresponden con los componentes que hemos mencionado antes, una carpeta con las clases como: “Subject” (asignatura), “User” (usuario) y “Department” (departamento); y otra con los servicios. Éstos últimos son los que acceden a la base de datos, al almacenamiento de documentos y a la autentificación y hacen las modificaciones.

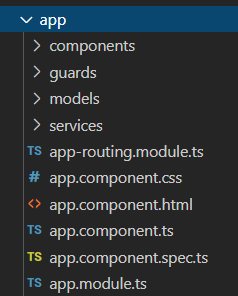


Figura . Carpeta "app".

Aunque antes de ver su contenido vamos a examinar un par de ficheros que son de interés: “app-routing-module.ts” y “app.module.ts”. El primero se utiliza para definir las rutas que usa la aplicación, desde el inicio hasta la famosa ventana del 404. El segundo archivo es usado para la importación de módulos puesto que para usarlo en nuestro proyecto debemos tenerlo aquí. Un módulo, para ponernos en contexto, es una clase con la anotación o decorador “@NgModule”. Es este archivo, “app.module.ts”, en donde declaramos los servicios (como los canales que acceden a Firebase) y los componentes (etiquetas HTML junto con su código). Éstos los veremos a continuación.

En la carpeta “components” tenemos varios componentes. Pongamos, por ejemplo, la pantalla de asignaturas de un usuario que es administrador, tal y como parece en la Figura 27.

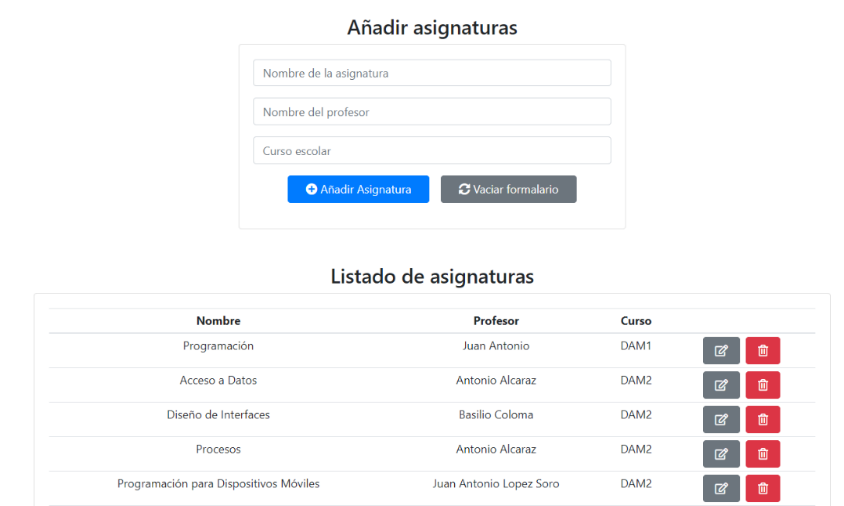


Figura . Pantalla "Asignaturas".

Esta pantalla en concreto, aparte de ser un componente en sí, tenemos dos componentes dentro: el formulario y la lista de las asignaturas. Veamos el código en la Figura 28. Este HTML viene acompañado por un código, el archivo se llama “subjectmain.componet.ts”. Éste solamente comprueba que el usuario haya iniciado sesión y su rol.

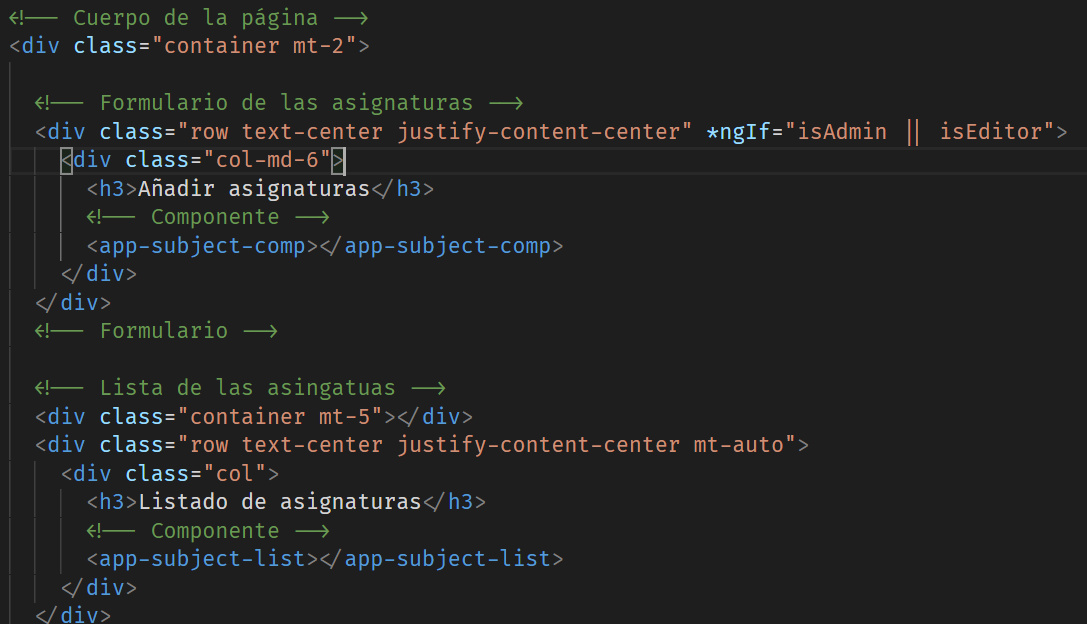


Figura . Código de la pantalla de las asignaturas. Archivo “subjectmain.component.html”

En el caso del segundo componente de la Figura 28, este tiene su propio código en “Typescript” y que vamos a ver en la Figura 29. Este es un ejemplo de componente que llama a un servicio, en concreto, “SubjectService”.

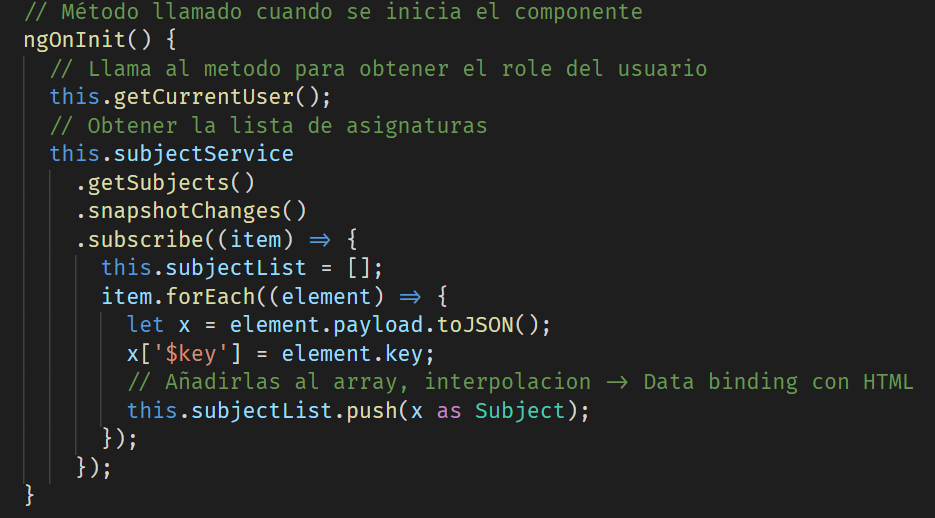
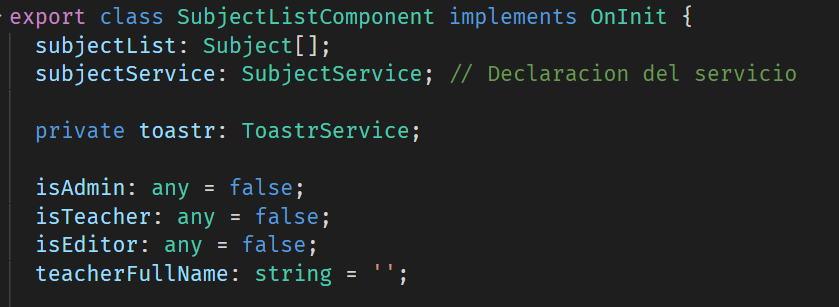


Figura . Código del componente "<app-subject-list>".

Con esto, nos vamos a la carpeta “services” y vemos como es el servicio al que llama el componente del que acabamos de hablar. El código que destacáremos se observa en la Figura 30, es el que llama al servicio de Firebase para obtener lo que haya en la base de datos y mantiene actualizado otra colección que depende de él, “sign-up”.

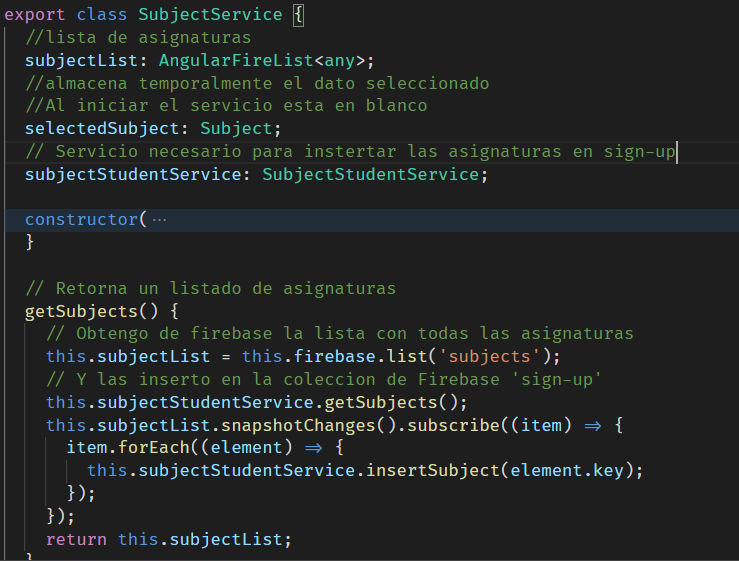


Figura . Código que destacar de "SubjectService".

Por otra parte, tenemos la carpeta “models” que son todas las clases, como “Subject”, la cual tiene las propiedades que vimos en la estructura de la base de datos, se muestra en la Figura 31.

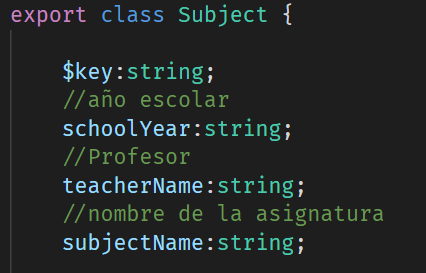


Figura . La clase "Subject".

Respecto al tema de la seguridad de la aplicación debemos evitar que los visitantes accedan a ella o al menos a las paginas destinadas a los usuarios de IDEApp. Para ello hacemos uso de los “Guards” (la última carpeta a comentar). Tal y como se muestra en la Figura 32, su código es sencillo puesto que viene con un método por defecto donde controlamos la salida, así el usuario será redirigido a la página de inicio si no ha iniciado sesión.

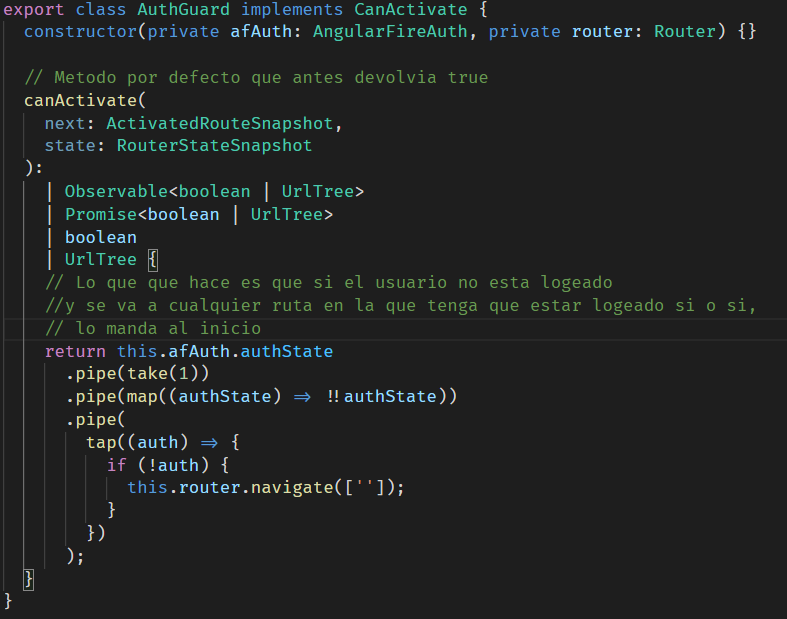


Figura . Código del "AuthGuard".

Para que esto funcione tenemos que modificar las rutas el archivo “app-routing.module.ts”, como se observa en la Figura 34.

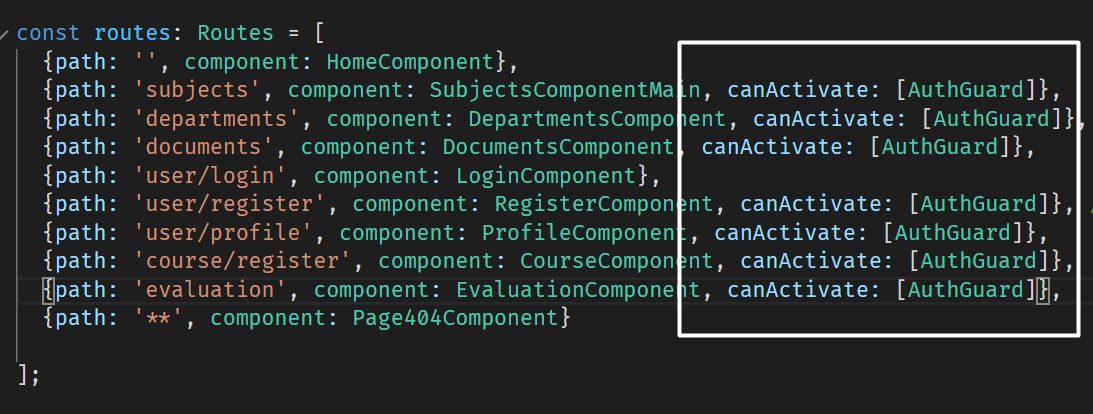


Figura . Modificación de las rutas.

En conclusión, esto es una breve explicación de lo que lleva el proyecto puesto que el resto de los componentes y servicios tienen una estructura similar a excepción del registro de usuarios (que varía por tener que insertarse en subcolecciones), las inserciones al almacenamiento de documentos (parecido a “Realtime Database” pero con métodos propios de “Firebase Cloud Storage”) y la autentificación del “login” (métodos propios del servicio de “Firebase Authentication”).

## Documentos de Definición Inicial y todas las modificaciones solicitadas y realizadas.

Continuemos hablado esta vez de la documentación inicial donde se describían las posibles tecnologías con las que podría desarrollarse este proyecto. Sin embargo, como se cambiaron varias cosas, sobre todo, las tecnologías que al final se usaron, vamos a realizar una comparación de esa documentación inicial y la final.

Primero vamos a resumir lo que se decía en la documentación inicial:

*En cuanto a los roles que tendría el usuario, hubieran sido: administrador, profesor y alumno. Las funcionalidades de estos eran ver el listado de asignaturas, subir documentos, ver departamentos (en caso de los profesores) y edición de contenido (en caso de los administradores).*

*Por otra parte, en lo referente a las tecnologías, se barajó la posibilidad de usar MySQL para la base de datos y las tecnologías: Spring, Java Server Faces, entre otras, para la realización de la aplicación.*

Sin embargo, actualmente se han cambiado tanto las tecnologías como las funcionalidades a las que pueden acceder los usuarios.

En primer lugar, vamos a hablar de las tecnologías para después pasar a los servicios finales que ofrece IDEApp.

Hemos pasado de Java a Angular (Typescript, en realidad) y de MySQL a Firebase. Este cambio es debido a que Angular es bastante estable para hacer aplicaciones web y Firebase se coordina eficaz y eficientemente con este framework. Ambas son tecnologías que se mencionaron en clase, pero con las que no tuvimos oportunidad de trabajar hasta ahora.

Durante este cambio también se unieron a este abanico de posibilidades, Node.js (aunque lo utilizamos en este proyecto) junto con Express.js y Firebase que son útiles para el desarrollo de un servidor web. Sin embargo, durante la fase de formación, nos dimos cuenta de que la mejor opción era usar Angular y Firebase (hablando en general), así que, finalmente IDEApp fue formada con estas tecnologías.

En cuanto a las funcionalidades que al principio se pensó para esta aplicación y las que hay actualmente, son bastante parecidas. Aunque han sufrido diferentes modificaciones, como:

* Tenemos un rol más, el “Editor”, que ayuda al administrador puesto que éste tenía mucha carga de trabajo.
* El listado de asignaturas, de departamentos y de documentos son casi los mismos.
* Listado de todos los usuarios y su modificación.
* Filtrado en lo referente al registro de usuarios y en el listado de usuarios, asignaturas y documentos.
* Se han añadido funcionalidades como que el propio administrador y editor pueden crear a otros con el mismo papel y la posibilidad de insertar más asignaturas, departamentos y cursos nuevos. Ésta última funcionalidad es “nueva” comparado con las que se pensaron al principio.
* Enlace directo a las redes sociales y visualización de post de estas mismas.
* Pantallas nuevas como son las del perfil del usuario.

# Mantenimiento.

Para el mantenimiento de la aplicación, sería conveniente tener una persona a cargo de hacer las posibles mejoras tanto de la interfaz como de la lógica que lleva detrás. Además, si se continua con el uso de Firebase, se necesitaría cambiar al plan “Blaze”, de pago, en lugar de “Spark” puesto que se necesitaría más espacio para guardar toda la información de los usuarios del centro.

## Mejorar IDEApp.

A la hora de ofrecer una mejor experiencia, siempre podemos enviar a los usuarios pequeños cuestionarios para saber que se podría añadir o mejorar a IDEApp.

Es evidente que, en cuanto a la lógica de ésta, siempre se podría mejorar para ofrecer algo más eficiente. Sin embargo, en este apartado vamos a hablar de funcionalidades que a los alumnos y profesores les hubiera gustado tener en un futuro.

* **Aplicación nativa:** para que siga siendo multiplataforma se tendría que realizar una versión para cada sistema operativo. Además, conllevaría a tener IDEApp en las tiendas de aplicaciones propias de cada uno de ellos, así como también, adaptar sus funcionalidades totalmente a las características de cada plataforma. Es una mejora que podría hacerse en un futuro pero que supondría costes adicionales en el desarrollo, puesto que, habría que mantener y actualizar las diferentes aplicaciones y no solamente una.
* **Búsqueda:** esta mejora es útil especialmente para los usuarios ya que puede mejorar significativamente la experiencia a la hora de navegar entre las diferentes pantallas de la web.
* **Continuos cambios en la interfaz:** como es normal, toda aplicación suele actualizar su aspecto. Es una mejora inevitable y recomendable para mantener la sensación de los usuarios de que está actualizada. Es decir, que el usuario sienta que está navegando en una web moderna. Además, puede que muchas partes se puedan automatizar con el paso del tiempo.
* **Ampliación de funcionalidades:** si bien es cierto, los usuarios que usan diariamente IDEApp pueden tener varias ideas o necesidades que ésta pueda cubrir en un futuro cómo, la que se planteó como mejora en la documentación inicial, poner notas a las asignaturas de los alumnos y la posibilidad de compartir los apuntes con otras personas que usan esta plataforma.

# Licencias.

En este apartado vamos a exponer las licencias de todos los recursos que hemos utilizado y la licencia propia de este proyecto.

Para empezar, las imágenes que se ven en la aplicación, más concretamente, la imagen del inicio es del Facebook del IES de Alquerías que cuenta con la licencia “no exclusiva” de esta plataforma y el resto de las imágenes son de la plataforma “[Pixels](https://www.pexels.com/es-es/)”. Su [licencia](https://www.pexels.com/es-es/license/) nos permite usar las imágenes gratuitamente, siendo opcional la mención de la fuente y, asimismo, permite la modificación de estas. Sin embargo, lo que no está permitido hacer es: en el caso de que aparezca una persona no debe resultar desfavorecida, no se pueden vender las fotos o utilizarlas de manera que parezca que se promociona un producto y tampoco redistribuir o venderlas en otras plataformas.

En este proyecto también hemos usado los iconos de “[Font Awesome](https://fontawesome.com/)”, el cual es de código abierto y compatible con GPL (Licencia Pública de GNU). Más concretamente su [licencia](https://fontawesome.com/v4.7.0/license/) para el código es [MIT](http://opensource.org/licenses/mit-license.html), para la documentación es [CC BY 3.0](http://creativecommons.org/licenses/by/3.0/) y para las “Fonts” es [SIL OFL 1.1](http://scripts.sil.org/OFL). Además, esta plataforma nos dice que todos los iconos de marca son marcas registradas de sus respectivos dueños, su uso no indica el respaldo del titular de la marca por parte de “Font Awesome”, no viceversa. Así mismo, estos iconos solo deben usarse para representar a la empresa o producto al que se refieren y no debemos utilizarlo de ninguna otra manera.

Por otra parte, este proyecto tiene la licencia “[CC BY-SA 3.0 ES](https://creativecommons.org/licenses/by-sa/3.0/es/)” la cual podemos ver en el “footer” de IDEApp.

# Conclusión.

Este tipo de aplicaciones, como lo es IDEApp, pueden llegar a tener éxito en los institutos puesto que no suelen tener una página web o aplicación propia que abarque funcionalidades similares. Como es el caso de las universidades, estas tienen una plataforma única donde el alumnado y profesorado pueden tener los documentos, realizar videoconferencias y visualizar las calificaciones, entre otras cosas.

Sin embargo, al ser un proyecto académico, los usuarios no pueden hacer tantas cosas en IDEApp como las haría en otra más profesional como es el caso de lo que hemos comentado hace un momento.

Por otra parte, los editores y administradores de la aplicación deberán pasar por un periodo de adaptación para familiarizarse con la aplicación y deberán modificar su rutina de trabajo para incorporarla en su día a día.

Finalmente, somos conscientes de que para que IDEApp puede llegar a desplegarse en un futuro en un instituto de la comunidad de Murcia sería necesario establecer un diálogo con el ministerio de educación de Murcia y trazar un plan de implementación conjunto con estos.

En conclusión, si al final se desplegara y con la adicción de nuevas mejoras podría llegar a ser un proyecto viable.

# Bibliografía.

## Cursos.

* Dominic Code.(2018). Youtube: “Curso de Angular 5. <https://www.youtube.com/channel/UC3QuZuJr2_EOUak8bWUd74A/videos>
* Antonio Carlos Garcia. OpenWebinars: “Curso de Angular 8 para principiantes”. <https://openwebinars.net/academia/portada/angular/>
* Diego Martínez. OpenWebinars: “Curso de firebase y Angular”. <https://openwebinars.net/academia/portada/angular-firebase/>
* Javier Camarra. OpenWebinars: “Curso de Angular 4”. <https://openwebinars.net/academia/portada/angular-4/>

## Documentación de Firebase.

* Firebase. Página web: <https://firebase.google.com/?hl=es-419>
* Firebase Documentación inicial. <https://firebase.google.com/docs/web/setup?hl=es-419>
* Firebase Authentication. <https://firebase.google.com/docs/auth?hl=es-419>
* Firebase Realtime Database.

<https://firebase.google.com/docs/database?hl=es-419>

* Cloud Storage. <https://firebase.google.com/docs/storage?hl=es-419>
* Firebase. Productos. <https://firebase.google.com/products/firestore?hl=es>

## Documentación de Angular.

* Angular. Página web. <https://angular.io/>
* Angular documentación inicial. <https://angular.io/guide/setup-local>
* Conceptos de Angular. <https://angular.io/guide/architecture>
* Breve tutorial de cómo crear una aplicación. <https://angular.io/tutorial>