

TRABAJO FIN DE GRADO

**PROYECTO DE DESARROLLO DE APLIACIONES MULTIPLATAFORMA**

CURSO 2022/2023

Aplicación móvil: Teach Cloud

Alumno/a:

Marco Fernández Barreiro

Tutor/a:

Iván Martín Sardor

AFA FORMACIÓN CONTINUA

**CICLO FORMATIVO DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA**

ÍNDICE

[1. INTRODUCCIÓN. 4](#_Toc134979259)

[1.1. Presentación y Objetivos. 4](#_Toc134979260)

[1.2. Contexto. 4](#_Toc134979261)

[1.3. Planteamiento del Problema. 5](#_Toc134979262)

[2. PLAN DE EMPRESA 5](#_Toc134979263)

[2.1. Introducción y descripción 5](#_Toc134979264)

[2.2. Análisis del entorno 6](#_Toc134979265)

[2.3. Análisis DAFO 7](#_Toc134979266)

[2.4. Análisis de viabilidad 9](#_Toc134979267)

[2.5. Estrategia de negocio y comunicación 9](#_Toc134979268)

[3. ESPECIFICACIÓN DE REQUISITOS. 13](#_Toc134979269)

[3.1. Introducción. 13](#_Toc134979270)

[3.2. Descripción general. 13](#_Toc134979271)

[3.3. Requisitos Específicos. 13](#_Toc134979272)

[3.3.1. Requerimientos Funcionales. 13](#_Toc134979273)

[3.3.2. Requerimientos de Interfaces Externas 14](#_Toc134979274)

[3.3.3. Requerimientos de Rendimiento 14](#_Toc134979275)

[3.3.4. Obligaciones de diseño 15](#_Toc134979276)

[4. ANÁLISIS 16](#_Toc134979277)

[4.1. Introducción 16](#_Toc134979278)

[4.2. Diagrama de clases 16](#_Toc134979279)

[4.3. Diagrama de Casos de Uso 17](#_Toc134979280)

[5. DISEÑO 27](#_Toc134979281)

[5.1. Introducción 27](#_Toc134979282)

[5.2. Capa de presentación 27](#_Toc134979283)

[5.3. Capa de Negocio o Lógica de la aplicación 37](#_Toc134979284)

[5.4. Capa de persistencia de datos 45](#_Toc134979285)

[6. IMPLEMENTACIÓN 47](#_Toc134979286)

[6.1. Tecnologías utilizadas en el desarrollo del proyecto 47](#_Toc134979287)

[6.2. Descripción del proyecto 48](#_Toc134979288)

[6.2.1. Capa de presentación 48](#_Toc134979289)

[6.2.2. Capa de negocio o Lógica de la aplicación 49](#_Toc134979290)

[6.2.3. Capa de persistencia de datos 49](#_Toc134979291)

[7. PRUEBAS 50](#_Toc134979292)

[7.1. Introducción 50](#_Toc134979293)

[7.1.1. Capa de presentación 50](#_Toc134979294)

[7.1.2. Capa de negocio 51](#_Toc134979295)

[7.1.3. Capa de persistencia 51](#_Toc134979296)

[8. CONCLUSIÓN 51](#_Toc134979297)

[8.1. Valoración Personal del Trabajo Realizado 51](#_Toc134979298)

[8.2. Posibles ampliaciones 52](#_Toc134979299)

**TABLA DE ILUSTRACIONES**

[1Estudio de mercado 8](#_Toc135234796)

[2 Análisis DAFO 8](#_Toc135234797)

[3Plan de producción 12](#_Toc135234798)

[4Guía de colores 16](#_Toc135234799)

[5Diagrama de clases 17](#_Toc135234800)

[**6** Diagrama de casos de uso 19](#_Toc135234801)

[7Pantalla de log-in 29](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234802)

[8Pantalla de registro 31](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234803)

[9Pantalla de inicio 32](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234804)

[10Filtro de anuncios 34](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234805)

[11Anuncios filtrados 35](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234806)

[12Creación de anuncios 36](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234807)

[13Perfil de usuario 37](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234808)

[14Diálogo de cambio de datos de usuario 38](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234809)

[15 Organización del empaquetado en la API 40](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234810)

[16Ejemplo clase en paquete “model” 41](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234811)

[17Ejemplo clase en paquete "modelDto" 42](file:///C:\Users\marco\OneDrive\Escritorio\AFA%20segundo\Proyecto%20de%20desarrollo%20de%20aplicaciones%20multiplataforma\Anticipo%20proyecto%20Marco%20Fernandez.docx#_Toc135234812)

[18Ejemplo clase en paquete "repository" 43](#_Toc135234813)

[19Ejemplo clase service en paquete "service" 44](#_Toc135234814)

[20Ejemplo clase ServiceImpl en paquete "service" 45](#_Toc135234815)

[21 Ejemplo clase en paquete "controller" 46](#_Toc135234816)

# INTRODUCCIÓN.

## Presentación y Objetivos.

Mi proyecto sobre la aplicación móvil Teach Cloud pretende ayudar a los alumnos que cursan bachillerato y tienen algún tipo de dificultad con asignaturas específicas a encontrar profesores particulares con facilidad que les ayuden a llevar el curso con éxito. El objetivo será poder tener todo este nicho centrado en un solo lugar, ya que las maneras actuales de encontrar un profesor particular no han avanzado en gran medida, debido a que se le ha dado más importancia a las academias de enseñanza.

Para ello se realizará una aplicación móvil en la cual tanto alumnos como profesores podrán darse de alta y anunciarse, ya sea buscando un servicio u ofreciéndolo. Esto permitirá a ambas partes entablar una fácil conexión y hacer más ameno todo el proceso de búsqueda.

## Contexto.

La idea de este proyecto viene a raíz de mi propia época de estudiante de bachillerato. Durante estos cursos, tuve problemas para llevar adelante algunas de las asignaturas que se me pedían, y cuando decidí buscar alguien que me pudiese brindar ayudas personalizadas, no fue nada fácil. No era sencillo encontrar profesores particulares, y cuando encontrabas alguno, podía darse el caso de que nuestros horarios no coincidiesen, no llegásemos a acordar un precio, y muchos otros problemas.

Con esto lo que se pretende es suavizar todo el proceso de búsqueda y encapsularlo en un solo lugar.

Tener una manera sencilla de poder ver los prerrequisitos que tiene un alumno o un profesor en cuando a su disposición horaria, movilidad, cercanía, etc… cortará el tiempo de búsqueda significativamente, permitirá llegar a un acuerdo de manera más efectiva y será más fácil evitar malos entendidos.

Por otra parte, a la hora de buscar un profesor en la actualidad, es muy difícil tener una idea de cómo de efectivas pueden ser esas clases a no ser que alguien conocido haya tenido alguna experiencia anterior. Gracias a tenerlo todo centrado en la aplicación, se podrán ver también valoraciones que tendrán, haciendo que la decisión que tomes en función de tus necesidades sea lo más acertada posible.

## Planteamiento del Problema.

Mediante este proyecto, lo que se pretende es crear un entorno específicamente para alumnos y profesores de bachillerato. Un servicio que les permita entrar en contacto de manera rápida y eficaz sin tener que recurrir a otros lugares. No todos los sitios donde poder anunciarse están enfocados en este grupo de enseñanza, por lo que podrían ser fácilmente eclipsados por ofertas que no tienen nada que ver con su campo.

En la actualidad, la manera más común de encontrar este tipo de servicios es en páginas no especializadas en este grupo de enseñanza, en tablones de anuncios en institutos o mediante el boca a boca, lo cual para la época en la que vivimos, no es tan avanzado como debería ser.

Con el paso de los años, más gente busca clases extraescolares que hace unos años, debido a que había cierto estigma social alrededor, pero esta gente acaba gravitando más hacia las academias, las cuales se centrar en resolver los problemas más generales de una clase que en centrarse en los alumnos de manera específica.

Mediante la división de estas ofertas a una aplicación específica para ellas, se pretende resolver la dificultad a la hora de encontrar un ofertante, y proporcionar los medios para que se pueda llegar a un acuerdo.

El principal competidor para esta aplicación serían las academias, muchas de las cuales llevan implantadas desde hace ya bastante tiempo y les precede cierta reputación. La ventaja que dan las clases particulares sobre estas, es tener alguien completamente centrado en las dificultades del cliente en lugar de las de un conjunto de ellos. Al resolver estas dudas de manera individual aseguras que le sacas todo el partido posible a las clases: no pierdes tiempo resolviendo problemas más generales de los cuales tu no tienes necesidad, y puedes poner todo el tiempo que desees en una necesidad especifica sin arrastrar contigo a un grupo entero de personas.

# PLAN DE EMPRESA

## Introducción y descripción

El proyecto es una aplicación móvil que servirá a los alumnos de bachillerato que tengan dificultades con asignaturas específicas, a encontrar un profesor particular que les guíe.

Tiene la posibilidad de registro para alumnos y profesores.

Se incluirá información como email o numero de teléfono para habilitar la comunicación entre usuarios.

Los usuarios tendrán un tablón donde anunciarse un buscador que les ayude en la búsqueda del profesor adecuado a sus necesidades.

El objetivo principal del proyecto es ayudar a alumnos de bachillerato a encontrar profesorado de apoyo, ya que, en numerosas ocasiones los estudiantes tienen dificultades para cursar asignaturas de elevada exigencia.

La aplicación está especializada en proporcionar a los alumnos de bachillerato, profesorado que les guíe, tanto en cómo aprobar la asignatura como en la superación de la EBAU.

## Análisis del entorno

Vivimos tiempos de cambios constantes en las leyes de educación que obligan a profesores y alumnos a rápidas adaptaciones que no siempre se llevan a cabo con la deseada eficacia.

Socialmente, parece que la única opción aceptable, es cursar estudios universitarios. Esto provoca un enorme estrés en los alumnos de bachillerato, con una competición constante a fin de lograr las mejores calificaciones que les permitan acceder a los estudios deseados.

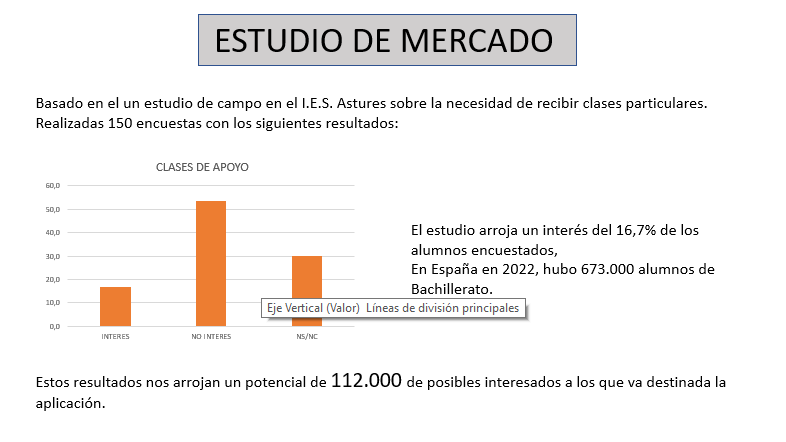
Este sistema es flexible y permite a alumnos y profesores gestionar sus necesidades y llegar a acuerdos individuales (días de clase, precios, horas, etc.)

Hay otro sistema que es el de las academias. Pero son tipos de clases con estructura más rígida y que no siempre se pueden adaptar a las necesidades de un alumno en concreto.

El uso es gratuito tanto para alumnos como para profesores

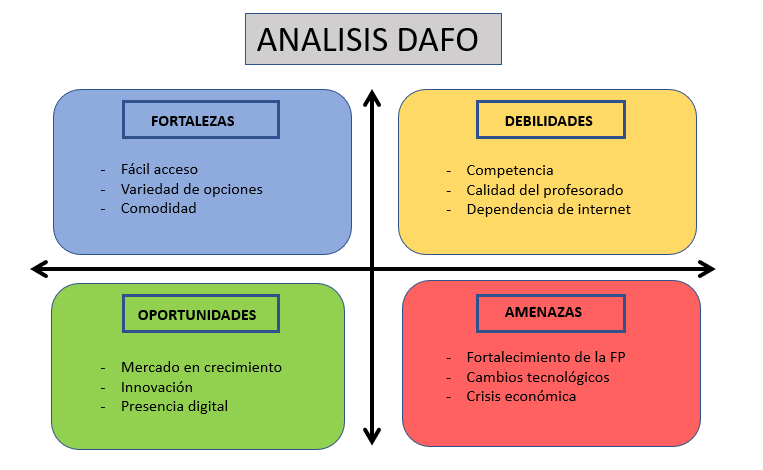
El coste de las clases se establecerá en función del acuerdo a que lleguen alumno y profesor.

En principio las clases serán presenciales u on-line (vía herramientas de reuniones on-line) en función de las preferencias del alumno y de la disponibilidad del profesor.

****

1Estudio de mercado

## Análisis DAFO

****

2 Análisis DAFO

**Fortalezas:**

Fácil acceso: al ser una aplicación móvil, es fácil para los usuarios descargarla y buscar profesores particulares desde cualquier lugar.

Variedad de opciones: la aplicación puede ofrecer una amplia variedad de profesores particulares con diferentes especialidades, lo que permite a los usuarios elegir el que mejor se adapte a sus necesidades.

Comodidad: la aplicación hace que sea muy cómodo encontrar un profesor particular en función de la ubicación y el horario del usuario.

**Oportunidades:**

Mercado en crecimiento: la demanda de profesores particulares está en constante crecimiento debido a la necesidad de mejorar el rendimiento académico y las habilidades laborales.

Innovación: la aplicación puede ser mejorada continuamente para agregar nuevas funciones que atraigan a más usuarios y proporcionen una mejor experiencia de usuario.

Presencia digital: la publicidad en línea puede ayudar a llegar a un público más amplio y atraer a nuevos usuarios a la aplicación.

**Debilidades:**

Competencia: hay muchas aplicaciones similares disponibles en el mercado, lo que significa que la aplicación debe competir para atraer y retener a los usuarios.

Calidad del profesorado: la calidad de los profesores particulares puede variar y algunos usuarios pueden tener malas experiencias, lo que puede afectar la reputación de la aplicación.

Dependencia de internet: La aplicación requiere una conexión a internet constante para funcionar, lo que puede ser un problema en áreas con mala conectividad.

**Amenazas:**

Fortalecimiento de la FP: el auge de la Formación Profesional, puede disminuir el número de alumnos potenciales de la aplicación.

Cambios tecnológicos: Los avances tecnológicos pueden hacer que la aplicación sea obsoleta o menos atractiva para los usuarios.

Crisis económica: La crisis económica que parecemos atravesar, puede disminuir la demanda de profesores particulares, lo que puede afectar la cantidad de usuarios de la aplicación.

## Análisis de viabilidad

La viabilidad del proyecto viene dada por la recuperación de la inversión inicial y de los rendimientos esperados por el desarrollador.

Costes iniciales:

* Ordenador y periféricos: 500 €
* Consumo energético: 5 €/mes
* Coste fibra internet: 35 € / mes
* Publicación en Play Store: 25 €
* Despreciamos el resto de costes, ya que se utilizará el domicilio del impulsor del proyecto.

Ingresos esperados: la aplicación será de uso gratuito tanto para alumnos como para profesores y su financiación correrá a cargo de los anunciantes a través del banner de la propia aplicación.

Los “clicks” están valorados entre 0.01 € y 0.05 €, lo que nos daría una media de 0.03 €.

Hemos visto en el estudio de mercado que tendríamos 112.000 posibles interesados en la aplicación.

Añadimos un factor multiplicador a los accesos, ya que es casi imposible que sólo con el primer acceso se cierre el acuerdo con el profesor. Este factor sería de 2 x, asumiendo que la primera vez se informa y la segunda, se contacta.

Esto nos daría unos ingresos esperados el primer año de 6.720 €, los arrojaría un superávit de 5.715 €

## Estrategia de negocio y comunicación

Dentro de la estrategia de negocio, se tocarán diversos puntos:

**Plan de marketing**

Definición del público objetivo*:* en principio va destinada a alumnos de bachillerato con proyección universitaria. En función de la acogida, se expandirá hacia alumnos de FP, Universitarios e interesados en idiomas.

Estrategia en redes sociales: se dará a conocer el proyecto a través de las redes sociales con perfiles adecuados a los fines perseguidos con la aplicación.

Alianzas: establecimiento de alianzas (en principio locales) con colegios e institutos de proximidad para la recomendación de la aplicación (códigos QR en tablones de anuncios).

Posicionamiento SEO:establecer y mantener un posicionamiento adecuado en buscadores (asegurar palabras clave, funcionamiento fluido de la aplicación, mediciones y análisis web).

Valor añadido: aplicación fácil de usar, gratuita y con un diseño atractivo y moderno.

**Plan de Producción**

El plan de producción ha seguido los siguientes pasos:

Se han identificado las necesidades y requerimientos de los usuarios basados en la experiencia personal del desarrollador y de la investigación a través de internet.

Se ha definido la estructura y el diseño de la aplicación

Se ha diseñado la interfaz de usuario con un diseño atractivo y fácil de usar tanto para los alumnos como para los profesores.

Se han realizado pruebas de usabilidad y se han corregido los errores encontrados.

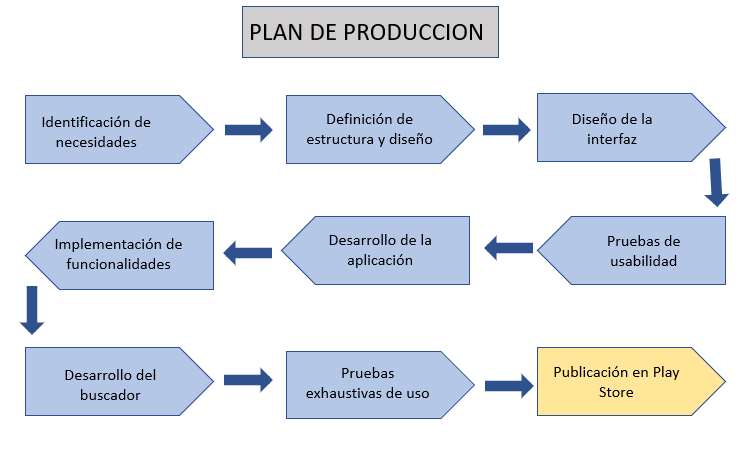
Se ha procedido a desarrollar la aplicación utilizando las herramientas adecuadas y al alcance del desarrollador.

Se han implementado las funcionalidades definidas y se han probado.

Se ha desarrollado el buscador que facilita los resultados a los usuarios.

Se ha procedido a probar y lanzar la aplicación con pruebas exhaustivas a fin de asegurarse de que la aplicación funciona correctamente y no hay errores.

Posteriormente, se procederá a la publicación de la aplicación en Play Store y dará comienzo la promoción a fin de atraer usuarios.

****

3Plan de producción

**Plan de recursos humanos**

El desarrollo inicial y puesta en marcha del proyecto será realizado en exclusiva por mi, por lo que no es necesario un plan de RRHH como tal.

En función de la acogida y expansión del proyecto (ampliación de objetivos) se estudiarían las necesidades y sus costes correspondientes.

**Plan económico financiero**

La inversión inicial es mínima, ya que se centra en mi esfuerzo y tiempo invertido.

El retorno de inversión, por tanto, es casi inexistente ya que se trataría de los costes de un ordenador y la publicación de la aplicación en Play Store (25 €)

El planteamiento salarial depende exclusivamente de mis expectativas y como tal serán variables según avance el proyecto.

**Plan de prevención**

En este caso, sólo tendríamos dos variables:

*Fatiga visual:*

*Iluminación adecuada:* Es importante que el lugar de trabajo cuente con una iluminación adecuada para evitar la fatiga visual. La iluminación debe ser suficiente, sin producir deslumbramientos y sin crear sombras excesivas.

Ergonomía: Es importante que la pantalla del ordenador o cualquier otro dispositivo electrónico se encuentre a la altura adecuada, para evitar que la vista se esfuerce en exceso. Además, es importante que el puesto de trabajo tenga una postura correcta y cómoda para evitar la tensión ocular.

*Descansos periódicos*: Es importante hacer descansos periódicos, para evitar la fatiga visual. Se recomienda descansar durante unos minutos cada hora, para relajar los ojos y evitar el esfuerzo continuo.

*Ejercicios oculares:* Hacer ejercicios oculares puede ayudar a relajar los ojos y prevenir la fatiga visual. Se pueden realizar ejercicios como parpadear, mover los ojos de un lado a otro, enfocar objetos cercanos y lejanos, etc.

*Pausas activas*: Realizar pausas activas que incluyan actividades físicas puede ayudar a reducir el riesgo de fatiga visual. Estas actividades pueden incluir estiramientos, caminatas o cualquier otro tipo de actividad física.

*Sobrecarga postural:*

*Ergonomía del puesto de trabajo*: Se deben adaptar la silla, la mesa y el monitor a las necesidades del trabajador, de manera que permitan una postura cómoda y natural. La silla debe tener un respaldo ajustable y un reposabrazos para apoyar los brazos en posición horizontal. La mesa debe estar a la altura adecuada para evitar que el trabajador deba levantar los hombros para teclear. El monitor debe estar situado a la altura de los ojos, a una distancia de unos 50 a 70 cm, y ajustado en ángulo de inclinación para evitar reflejos y deslumbramientos.

*Descansos regulares:* Se recomienda hacer pausas breves y regulares para levantarse, estirarse y hacer ejercicios de relajación. Se recomienda hacer una pausa de unos 10 a 15 minutos cada 2 horas de trabajo en el ordenador.

*Ejercicios de estiramiento:* Es importante realizar ejercicios de estiramiento que involucren cuello, hombros, espalda, brazos y manos. Los estiramientos deben ser suaves y sencillos, y deben durar unos pocos segundos. Es recomendable que se realicen con regularidad.

*Iluminación adecuada:* La iluminación debe ser suficiente, sin producir deslumbramientos y sin crear sombras excesivas. La pantalla debe tener un brillo adecuado y debe estar libre de reflejos.

# ESPECIFICACIÓN DE REQUISITOS.

## Introducción.

El campo de la enseñanza es fundamental en la sociedad de hoy en día, donde se premia a los estudiantes con las mejores calificaciones sobre los demás. La mayoría de los estudiantes no pueden alcanzar estas metas ellos solos por lo que tener una herramienta para encontrar tutores que les puedan ayudar parece algo fundamental.

Pese a esto, lo que más arraigado está son las academias, y muy por debajo, las clases particulares, las cuales permiten despejar las dudas específicas que pueda tener el alumno y le ayudarán a entender mejor la materia que se le impone aprender.

## Descripción general.

La aplicación será un tablón de anuncios donde tanto alumnos como profesores podrán contactar entre sí para solicitar u ofrecer servicios. Mediante un inicio de sesión tendrás que introducir tanto tus datos como lo que desees buscar, en caso de ser un alumno, u ofertar en caso de ser profesor.

Desde esta pantalla que funcionará como menú, se te permitirá acceso a las funciones de la aplicación, como pueden ser publicar un anuncio, buscar otros anuncios de interés, ver y modificar datos de tu perfil etc…

## Requisitos Específicos.

* + 1. Requerimientos Funcionales.

El proyecto estaría planteado para que se subiese a la Google Play Store para que pudiese ser descargado, probado y puesto en funcionamiento por usuarios reales.

En un principio, estaría pensado para funcionar solo a nivel nacional y, por tanto, en castellano. En función del éxito de la aplicación, se planteará la posibilidad de crear sus propias versiones adaptadas a los estudios previos a la universidad de otros países.

El proyecto contará con:

* **Inicio de sesión o creación de cuenta:** los usuarios podrán acceder a la aplicación introduciendo sus credenciales o bien si eres nuevo, crear una cuenta introduciendo los datos pertinentes.
* **Creación de anuncios:** Si es profesor, se podrá poner un anuncio en el que oferten sus clases particulares. Si es alumno, podrá decir que materias está cursando y que es exactamente lo que busca.
* **Búsqueda de anuncios:** mediante un buscador se podrá filtrar que es lo que estás buscando.
* **Fácil comunicación:** la aplicación tratará de ser intuitiva a fácil de utilizar para todo tipo de usuarios, permitiendo una rápida y sencilla comunicación entre los mismos.
* **Entorno agradable para el usuario:** la interfaz de usuario será diseñada de una forma clara y que permita que todo tipo de usuarios tengan una estancia agradable durante el tiempo que necesiten usarla.
  + 1. Requerimientos de Interfaces Externas

En esta sección trataremos las interfaces externas que necesitará nuestra aplicacion.

La principal sería la interfaz de usuario, la cual trataremos de hacer lo más intuitiva posible, resaltando las opciones de los menus y proporcionando la infomacion necesaria para que el usaurio sepa manejar la aplicación de manera cómoda.

En cuanto a la interfaz de hardware, la apliación tendrá que conectarse con la base de datos mediante una Api Rest para su conrrecto funcionamiento.

* + 1. Requerimientos de Rendimiento

En cuanto a los requisitos de rendimiento, serán los que un usuario estándar podría esperar de una aplicación móvil.

* **Estabilidad:** la aplicación mantendrá un flujo de ejecución aceptable, se evitarán fallos, cuelgues, bloqueos y otro tipo de malfuncionalidades en la medida de lo posible para garantizar un uso placentero.
* **Rendimiento:** la aplicación cargará de manera rápida, o en caso de no ser asi, proporcionará algún tipo de mensaje al usuario informandole de que el proceso está tardando más de la cuenta, nunca dejando al usuario sin una respuesta y dandole a pensar que puede estar no funcionando correctamente.
* **Batería:** debido a que la aplicación tiene la finalidad de conectar a la gente entre si, tratando de hacer el proceso lo más simple posible para los usuarios menos expermientados, no consumirá grandes recursos en cuanto a la batería del dispositivo.
* **Seguridad:** la app solicitará el número mínimo de permisos necesarios para su uso correcto, para asi no dar la impresión al usuario de que se le está pidiendo más de la cuenta a la hora de usar la app.
* **Accesibilidad:** los iconos o botones serán adecuados para todo tipo de personas, ya sea con dificultad de visión o con dispositivos de pantalla más pequeña. En un futuro se podrían implementar más funciones de accesibilidad como un tema oscuro.
  + 1. Obligaciones de diseño

Para el diseño de la aplicación, se requiere que esta sea intuitiva, facil de utilizar y con la información suficiente para que un usuario con conocimientos basicos de tecnoligía pueda utilizar la aplicación sin ningun tipo de problema.

Se utilizarán menus desplegables para la navegación por la aplicación, botones de un tamaño adecuado y el uso de imagenes descriptivas en lugar de texto para partes especificas de la aplicación.

Se utlizará la gama de colores mostrada a continuación para de diseño general de la aplicación, utilizando variantes sutiles de los mismos en caso de necesidad, pero nunca rompiendo la estética establecida previamente.



4Guía de colores

# ANÁLISIS

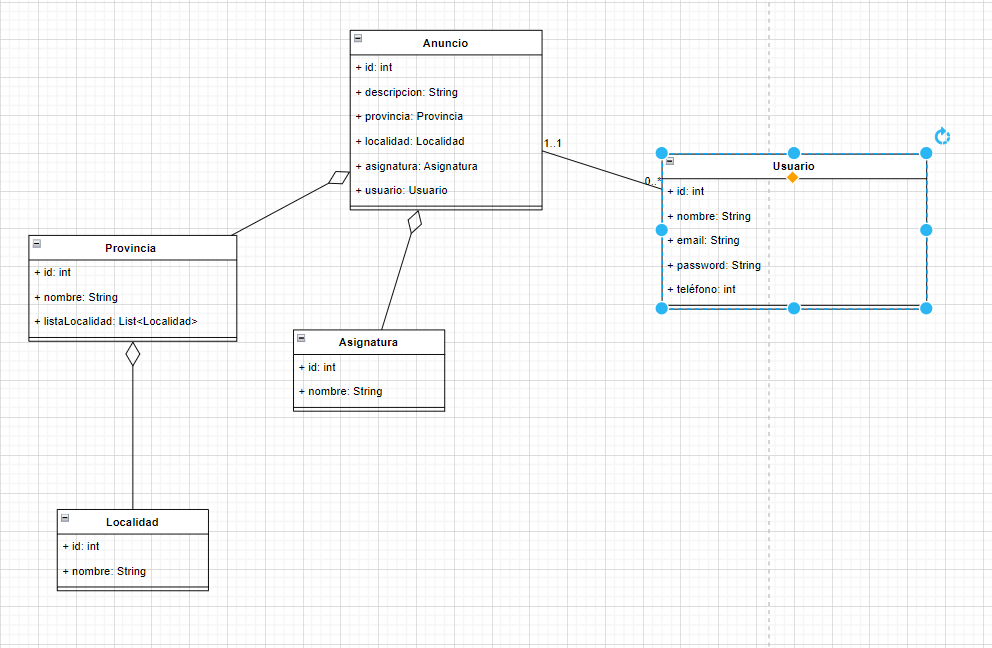
## Introducción

En el análisis de la aplicación veremos como va a ser desarrollada la misma: trataremos de ver y explicar desde el diseño que tendrá el diagrama de clases en el que se basará la aplicación hasta el diagrama de casos de uso y los requisitos funcionales y no funcionales que estos necesitan.

En cuanto a los casos de uso, en este punto describiremos el diagrama y los casos más relevantes, de una manera breve, dejando este apartado para su posterior realización en el anexo, donde si se entrará en detalles de cada uno de los casos de uso. Lo mismo se realizará con los requisitos.

## Diagrama de clases

Para nuestra aplicación, utilizaremos el diagrama de clases que se verá a continuación.



5Diagrama de clases

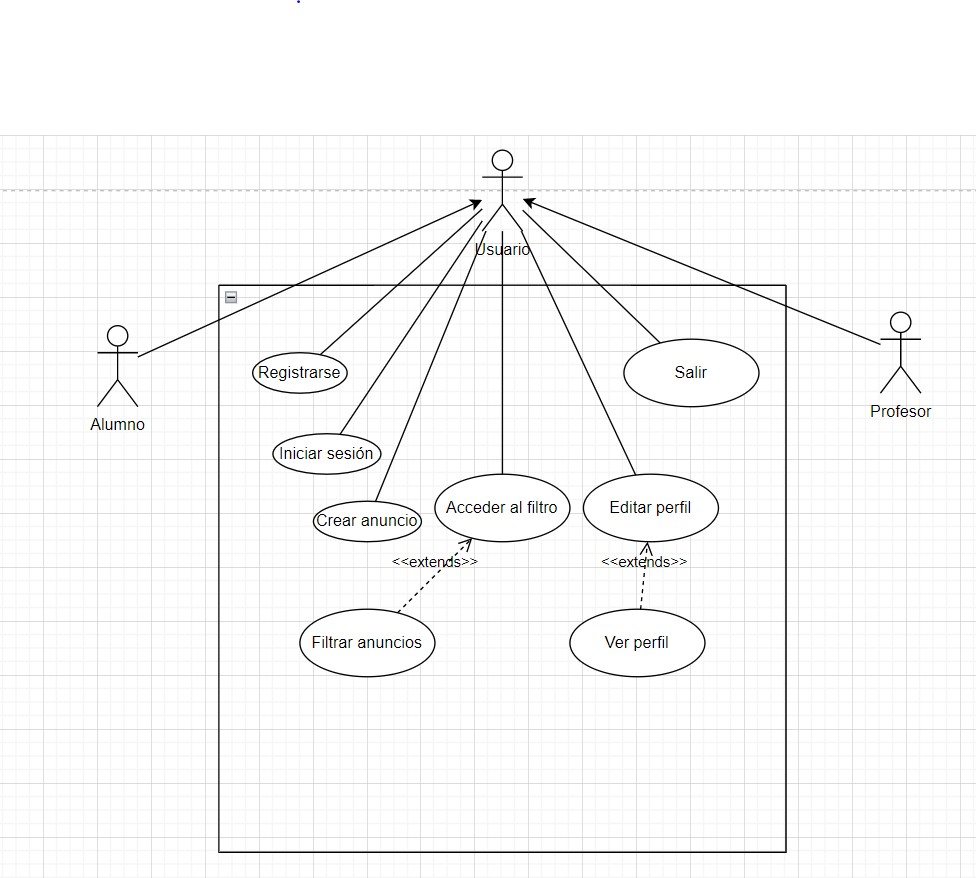
Como se puede apreciar en la imagen, este diagrama constará de las siguientes clases:

* **Usuario:** será una clase donde se guardarán los datos de los usuarios. Constará de los atributos necesarios para la creación de un usuario (id, nombre, email, contraseña y teléfono).
* **Anuncio:** clase que podrá crear un usuario, ya sea alumno o profesor, aunque ambos contienen lo mismo. Una asignatura (asignatura), la provincia (provincia) en la cual se está (la cual luego se dividirá en cuidades), una localidad o ciudad(localidad), una descripción (descripción) y el usuario que ha puesto el anuncio(usuario) .
* **Asignatura:** esta clase sirve para definir una asignatura en cuestión, la cual tendrá un nombre (nombre).
* **Provincia:** sirve para marcar una zona aproximada del país en la que reside el usuario, para poder centrar las búsquedas. Esta clase tendrá a su vez una lista de localidades en las que el usuario podría dar la clase, para terminar de concretar una zona (listaLocalidad).
* **Localidad:** clase para marcar una zona más especifica de la provincia en la que se podría dar la clase. Constará del nombre de la localidad (nombre).

## Diagrama de Casos de Uso

Para los diagramas de casos de uso, nos centraremos en la imagen a continuación mostrada.

Este no será el diagrama final, pero si una aproximación con lo que se tiene hasta ahora.



**6** Diagrama de casos de uso

Apoyándonos en la ayuda visual que nos proporciona el diagrama, podemos ver que consta de los siguientes casos de uso:

|  |  |  |
| --- | --- | --- |
| CU1: Registrarse | | |
| Versión | Versión 1.0 | |
| Descripción | Un nuevo usuario, ya bien sea alumno o profesor, se registrará en la aplicación introduciendo la información necesaria para el registro. | |
| Precondiciones | Entrar en la aplicación y no tener una cuenta ya creada | |
| Secuencia normal | **Paso** | **Acción** |
| **1** | Se introducirá el nombre |
| **2** | Se introducirá el email |
| **3** | Se introducirá la contraseña y confirmar contraseña |
| **4** | Se introducirá el teléfono |
|  | **5** | Si todos los campos son correctos, el sistema registrará al usuario en la base de datos y accederá a la aplicación. |
| Excepciones | **Paso** | **Acción** |
| **5** | Si alguno de los campos no está relleno. El sistema muestra un error. Ir a paso 1. |
| **5** | Si el email está ya en uso. El sistema muestra un mensaje. Ir a paso 2. |
| **4** | Si la contraseña y su confirmación no son iguales. El sistema muestra un mensaje. Ir a paso 3. |
| Postcondiciones | Si los datos de registro son válidos, se creará una nueva cuenta para el usuario y se accederá a la aplicación. | |
| Actores | Usuario. | |
| Importancia | Vital | |
| Urgencia | Inmediata | |

**Tabla 1: Descripción del caso de uso CU1**

|  |  |  |
| --- | --- | --- |
| CU2: Iniciar sesión | | |
| Versión | Versión 1.0 | |
| Descripción | Un usuario que ya tenga un registro previo introducirá sus credenciales de inicio de sesión y accederá a la aplicación. | |
| Precondiciones | Estar en la aplicación y haber realizado un registro previamente | |
| Secuencia normal | **Paso** | **Acción** |
| **1** | Se introducirá el email |
| **2** | Se introducirá la contraseña |
| **3** | Se pulsará el botón de Log in. |
| **4** | Si los datos son correctos, el sistema accederá a la aplicación. |
| Excepciones | **Paso** | **Acción** |
| **4** | Si el email no se encuentra en la base de datos. El sistema muestra mensaje. Ir a paso 1. |
| **4** | Si la contraseña no es correcta. El sistema muestra un mensaje. Ir a paso 2. |
| Postcondiciones | Si los datos de acceso son correctos, se entrará a la aplicación | |
| Actores | Usuario. | |
| Importancia | Vital | |
| Urgencia | Inmediata | |

**Tabla 2: Descripción del caso de uso CU2**

|  |  |  |
| --- | --- | --- |
| CU3: Crear anuncio | | |
| Versión | Versión 1.0 | |
| Descripción | Un usuario introducirá los datos necesarios para crear un anuncio y el anuncio se creará. | |
| Precondiciones | Haber iniciado sesión con una cuenta en la aplicación | |
| Secuencia normal | **Paso** | **Acción** |
| **1** | Introducir descripción |
| **2** | Introducir localidad |
| **3** | Introducir provincia |
| **4** | Introducir asignatura |
|  | **5** | Si todos los campos son correctos, el sistema creará el anuncio |
| Excepciones | **Paso** | **Acción** |
| **5** | Si alguno de los campos no está relleno. El sistema muestra un error. Ir a paso 1. |
| Postcondiciones | Si los datos introducidos son correctos, se creará un nuevo anuncio del usuario registrado. | |
| Actores | Usuario. | |
| Importancia | Vital | |
| Urgencia | Inmediata | |

**Tabla 3: Descripción del caso de uso CU3**

|  |  |  |
| --- | --- | --- |
| CU4: Acceder al filtro | | |
| Versión | Versión 1.0 | |
| Descripción | El usuario accederá al filtro para los anuncios | |
| Precondiciones | Haber iniciado sesión con una cuenta en la aplicación | |
| Secuencia normal | **Paso** | **Acción** |
| **1** | Pulsar el botón que llevará al filtro. |
| **2** | El sistema accederá a la pantalla con el filtro. |
| Excepciones | **Paso** | **Acción** |
| **2** | No debería haber excepciones en este caso |
| Postcondiciones | Se accederá a la pantalla de filtro de anuncios. | |
| Actores | Usuario. | |
| Importancia | Vital | |
| Urgencia | Inmediata | |

**Tabla 4: Descripción del caso de uso CU4**

|  |  |  |
| --- | --- | --- |
| CU5: Filtrar anuncios | | |
| Versión | Versión 1.0 | |
| Descripción | El usuario introducirá una serie de parámetros de búsqueda en la aplicación para poder encontrar un anuncio que le interese | |
| Precondiciones | Haber iniciado sesión con una cuenta en la aplicación y estar en el filtro | |
| Secuencia normal | **Paso** | **Acción** |
| **1** | El usuario entrará al filtro |
| **2** | El usuario podrá elegir la localidad |
| **3** | El usuario podrá elegir la provincia |
| **4** | El usuario podrá elegir las asignaturas |
|  | **5** | El sistema enseñará los anuncios que cumplan las condiciones impuestas por el usuario |
| Excepciones | **Paso** | **Acción** |
| **5** | No hay anuncio con esas condiciones. El sistema informa al usuario. Cambiar filtros de búsqueda. Volver a paso 1 |
| Postcondiciones | El usuario podrá buscar entre todos los anuncios de la aplicación en función de lo que le interese. | |
| Actores | Usuario. | |
| Importancia | Vital | |
| Urgencia | Inmediata | |

**Tabla 5: Descripción del caso de uso CU5**

|  |  |  |
| --- | --- | --- |
| CU6: Editar perfil | | |
| Versión | Versión 1.0 | |
| Descripción | El usuario dispondrá de la posibilidad de editar los campos que desee de su perfil y cambiarlos en función de nuevas condiciones. | |
| Precondiciones | Haber iniciado sesión con una cuenta en la aplicación | |
| Secuencia normal | **Paso** | **Acción** |
| **1** | Cambiar nombre en la aplicación |
| **2** | Cambiar teléfono en la aplicación |
| **3** | Si los nuevos datos introducidos son correctos, el sistema los actualizará en la base de datos |
| Excepciones | **Paso** | **Acción** |
| **3** | Teléfono no es válido. Se avisa al usuario. Ir a paso 2 |
| Postcondiciones | EL usuario podrá cambiar algunos de los datos de su cuenta | |
| Actores | Usuario. | |
| Importancia | Vital | |
| Urgencia | Inmediata | |

**Tabla 6: Descripción del caso de uso CU6**

|  |  |  |
| --- | --- | --- |
| CU7: Entrar al perfil | | |
| Versión | Versión 1.0 | |
| Descripción | El usuario podrá entrar en su perfil para ver sus datos | |
| Precondiciones | Haber iniciado sesión con una cuenta en la aplicación | |
| Secuencia normal | **Paso** | **Acción** |
| **1** | Se pulsará el botón de acceso al perfil |
| **2** | Entrará a la pantalla de perfil de usuario. |
| Excepciones | **Paso** | **Acción** |
| **4** | No debería haber excepciones |
| Postcondiciones | El usuario podrá ver sus datos. | |
| Actores | Usuario. | |
| Importancia | Vital | |
| Urgencia | Inmediata | |

**Tabla 7: Descripción del caso de uso CU7**

|  |  |  |
| --- | --- | --- |
| CU8: Salir | | |
| Versión | Versión 1.0 | |
| Descripción | El usuario podrá salir de la aplicación desde el menú principal | |
| Precondiciones | Haber iniciado sesión con una cuenta en la aplicación | |
| Secuencia normal | **Paso** | **Acción** |
| **1** | Se pulsará el botón de salir de la aplicación |
| **2** | El sistema cerrará la sesión del usuario. |
|  |  |  |
| Excepciones | **Paso** | **Acción** |
| **3** | No debería haber excepciones. |
| Postcondiciones | El alumno habrá salido exitosamente de la aplicación. | |
| Actores | Usuario. | |
| Importancia | Vital | |
| Urgencia | Inmediata | |

**Tabla 8: Descripción del caso de uso CU8**

Estos son los casos de uso de los que describiremos en este punto. Otros que se puedan ir añadiendo se añadirán en los anexos para no extender de más el proyecto.

Con esto casos de uso vienen también una lista de requisitos funcionales y no funcionales.

# DISEÑO

## Introducción

Este apartado explicará cuales han sido los pasos en el diseño de las distintas capas de la aplicación.

Empezaremos por la capa de presentación, donde se explicará y argumentará el porque del diseño elegido para la parte del Front-end de la aplicación, realizado con Android Studio.

Después pasaremos a la capa de negocio o lógica de la aplicación, donde se empezará a entrar en detalle de la realización de la API con Spring boot y lenguaje JAVA.

Finalmente, en la capa de persistencia de datos se tocará la base de datos: como se creó, Scripts y tablas necesarias para su correcto uso y enlace con la API y el Front-end.

## Capa de presentación

En este apartado entraremos en detalle de como se ha creado el Front-end de la aplicación, que recursos hemos utilizado, y se explicarán detalladamente las diferentes pantallas con las que cuenta la aplicación.

Empezando por lo básico, la aplicación se realizó con Android Studio, un entorno de desarrollo especializado en aplicaciones para dispositivos móviles que cuenten con el sistema operativo “Android”. También cabe mencionar que, Android Studio soporta diversos lenguajes como JAVA, Kotlin o C++, como bien se **puede ver en la página web de Android Developer** *(Android Developer, s.f.).* Para este proyecto el lenguaje elegido será JAVA, ya que es de el que más dominio se tiene y el que ha sido utilizado durante el curso.

* **Pantalla de inicio**

Empecemos por lapantalla de inicio. En ella podemos ver el logo de la aplicación sobre una banda negra, un cuadrado blanco y un fondo naranja. Esta elección de colores viene dada por la guía de estilos referenciada anteriormente, en la que se deja en claro los colores que serán principalmente usados en la aplicación y a la cual se mantiene fidelidad. Sobre el cuadrado blanco se podrá ver unos campos los cuales se podrán rellenar. Estos serán el “email” y la “contraseña”, y serán necesarios para realizar la autentificación del usuario antes de acceder a la aplicación. Además de estos campos, en la esquina inferior derecha se pueden ver otros 2 objetos: un enlace tal que: “¿No tienes cuenta? Regístrate” y un botón negro con el texto “Log in”. El primero servirá para redirigir a los usuarios que no dispongan todavía de una cuenta en la aplicación a un formulario del que hablaremos más adelante, y el botón servirá para que esos que si están ya registrados puedan acceder a la aplicación con sus credenciales. A continuación, se mostrará una imagen del conjunto de esta pantalla.

7Pantalla de log-in

En cuanto a la programación que hay detrás de esta pantalla, tendremos una clase JAVA en la cual inicializaremos todos los componentes necesarios de la pantalla, y se moldeará la lógica de el enlace y del botón de acceso.

El enlace tiene una lógica simple: en el momento en que lo toques, abrirá la actividad que contiene el formulario de inscripción, el cual pasaremos a explicar a continuación.

El botón por otra parte, tendrá una lógica algo más compleja. Al ser pulsado, deberá recoger los datos de se introdujeron en los campos de “email” y “contraseña” y hacer una llamada a la API.

Primero se buscará un registro de ese email en la base de datos, y en caso de que se encuentre, realizará la comprobación de la contraseña. Cabe mencionar que la contraseña será encriptada por motivos de seguridad, aunque ese tema se tocará más en profundidad cuando expliquemos la API. Por ahora lo que necesitamos saber es que, en caso de que la contraseña coincida con la de la base de datos asociada al email dado, se otorgará acceso a la aplicación. En caso contrario, aparecerá un mensaje diciendo que las credenciales son incorrectas.

* **Formulario de registro**

Después de tocar el enlace mencionado anteriormente, se nos abrirá una nueva actividad con un formulario. En el podremos ver el logo de la aplicación en un fondo blanco, múltiples campos donde introducir datos con sus correspondientes etiquetas y un botón de registro (color naranja).

Funcionará como cualquier formulario: introduces los datos que quieres para tu cuenta en la aplicación y pulsas el botón de enviar. Se deben rellenar todos los campos y estos serán:

**-Nombre:** el nombre que se verá en la aplicación. Preferiblemente debería ser nombre real o algo lo suficientemente descriptivo, como un diminutivo.

**-Email:** el email que se asociará a tu cuenta y con el que deberás autentificarte durante el registro.

**-Contraseña y confirmar contraseña:** la contraseña será junto con el email lo necesario para autentificarte en la aplicación. Confirmar contraseña es un mecanismo de seguridad para que en caso de que la primera vez se haya metido una contraseña no deseada por error, se pueda avisar al usuario.

**-Teléfono:** Será la manera que tendrán de contactar entre ellos los usuarios de la aplicación.



8Pantalla de registro

La clase JAVA simplemente inicializará los componentes necesarios e implementará la lógica detrás del botón de registro. En cuanto a la lógica del mismo, en el momento de pulsarlo se harán ciertas comprobaciones, como que todos los campos estén debidamente rellenados, que no exista una cuenta con el mismo email en la base de datos y que contraseña y confirmar contraseña sean iguales. Si todas son correctas, se añadirá el usuario a la base de datos y se accederá automáticamente a la aplicación con sus credenciales.

* **Página principal**

Ya teniendo una cuenta y habiendo accedido a la aplicación, estaremos en la página principal. En esta, veremos un texto que nos da la bienvenida personalmente, junto con más información en la pantalla, todo sobre un fondo naranja, como llevamos viendo hasta ahora. (Añadir lo que vaya a poner aquí).

Además, en la esquina superior derecha se ve un menú. Este contará con 3 opciones:

* Crear anuncio.
* Ver perfil.
* Salir.

Pasaremos por todas estas opciones a su debido tiempo, por ahora seguiremos hablando de la pantalla principal.

9Pantalla de inicio

Por último, en la esquina inferior derecha vemos un botón con una lupa. Este será el que nos lleve al buscador de anuncios de la aplicación.

La lógica detrás de esta capa es: múltiples Textviews donde se resalta toda la información que vemos, incluido el nombre de usuario que será sacado de los datos recibidos al iniciar la sesión, un menú de opciones en con las ya mencionadas y una función para que según cual pulses te lleve a distintas partes de la aplicación, y por último un FAB (floating action button) que al pulsar sobre el se creará la siguiente pantalla.

Ahora, seguiremos por la parte del FAB del filtro de anuncios.

* **Filtro de anuncios**

Empezamos a entrar en las zonas mas complejas de la aplicación. En la pantalla del filtro se verá un cuadro de texto que nos informa de lo que tenemos que hacer, unos cuantos textos de apoyo y 3 desplegables de opciones. Estos desplegables se alimentan a partir de la información almacenada en la base de datos. Y tienen un procedimiento algo complejo.

Los desplegables o “spinners” de provincia y localidad van enlazados entre sí: según la provincia que tengas seleccionada, se actualizarán las opciones de las localidades. Esto se hace comprobando cual de las opciones está seleccionada en el desplegable de las provincias, y haciendo una llamada a la API con esta información para recibir y cargar las localidades necesarias, todo aunque las opciones vayan cambiando.

El desplegable de las asignaturas sigue la misma filosofía, solo que sin estar enlazado a los otros 2.

En el momento en que las opciones por las que se desea filtrar sean las preferidas, podemos pulsar el botón de filtrar, el cual nos cambiará de actividad a una donde estará la lista con los anuncios que queremos ver.



10Filtro de anuncios

* **Lista de anuncios ya filtrados**

Llegaremos aquí después de elegir los filtros en la pantalla anterior. Al momento de entrar a esta pantalla, se hará una petición a la API con las opciones de los filtros, y se traerá una lista de anuncios para mostrar. Estos se mostrarán en un RecyclerView, un componente para mostrar listas de objetos las cuales se pueden personalizar. En este caso, lo que mostraremos en una lista de “layouts” en los cuales desplegaremos toda la información que se necesite sobre los anuncios que se han traído desde la base de datos.

Esta información será:

* Nombre del usuario
* Teléfono del usuario
* Provincia
* Localidad
* Asignatura
* Descripción del anuncio

Estos serán mostrados con un fondo de un tablón de anuncios, y se podrá hacer “scroll” hacia abajo para ver todos los que haya.

11Anuncios filtrados

En cuanto a la lógica, después de hacer la llamada a la Api para que nos traiga a información, crearemos un adaptador de esos anuncios y le pasaremos el layout donde queremos que vaya la información, rellenaremos el layout con la información que nos llegue de cada anuncio, y por último lo mostraremos.

* **Crear anuncio**

Volviendo a la pantalla principal, vamos a entrar más en detalle de las opciones de menú superior, empezando por crear anuncio.

Al entrar, veremos una pantalla con un montón de campos a rellenar. Estos serán una descripción del anuncio, la provincia, la localidad y la asignatura, todos debidamente marcados para que el usuario sepa en todo momento que es cada cosa. Las opciones de localidad, provincia y asignatura siguen la misma lógica que hacían en la pantalla anterior: reciben la información de la base de datos y través de una petición, y se actualizan en función de ellas mismas. El campo de la descripción es u simple texto plano donde se podrá escribir.

12Creación de anuncios

Cuando todos los datos de creación del anuncio sean los adecuados, se pulsará el botón de “crear anuncio”. Este recogerá la información de los desplegables, el cuadro de texto y de la sesión iniciada para crear un anuncio asociado al usuario que ha iniciado sesión, y las opciones de desplegable. Este se guardará en la base de datos y se volverá a la pantalla principal.

* **Ver perfil**

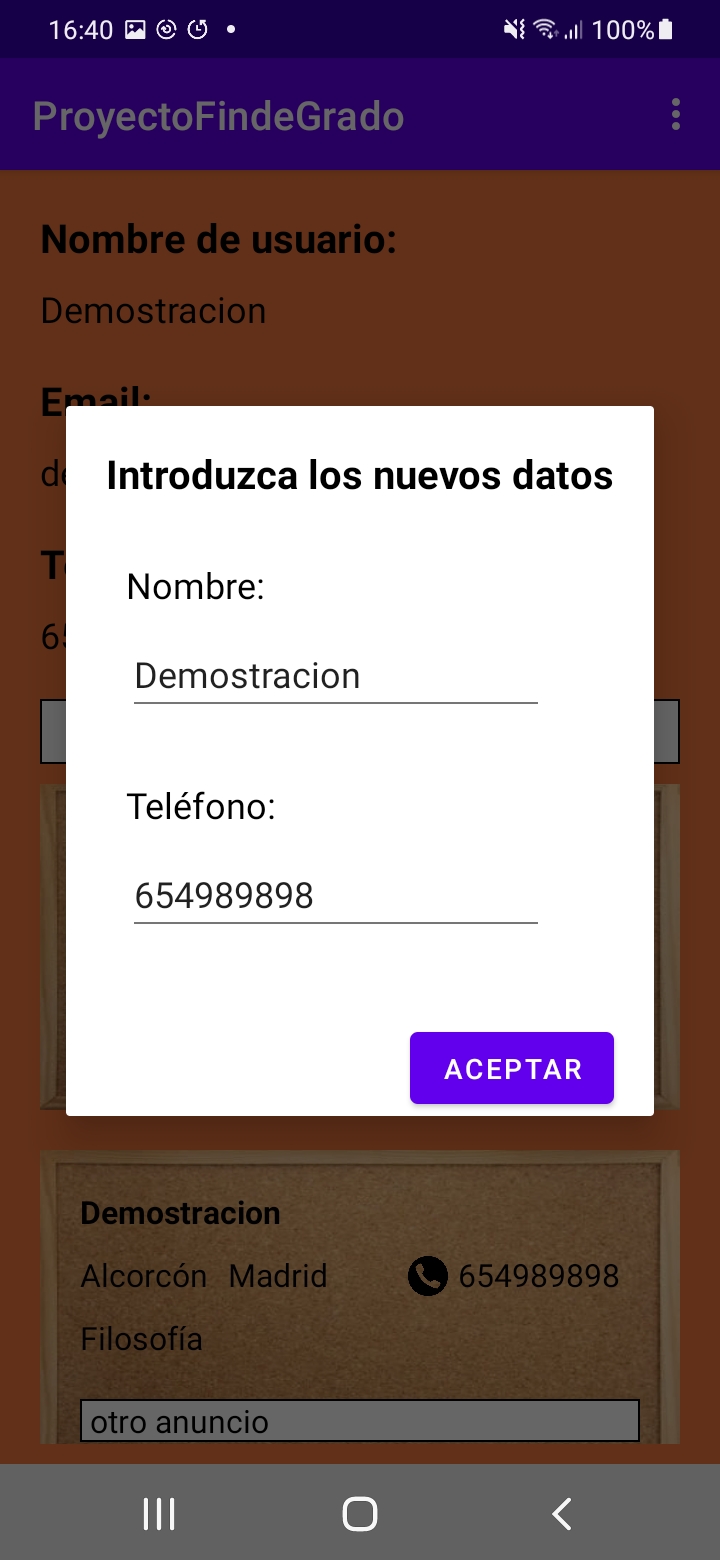
Ahora que tenemos un anuncio, querremos ver como ha quedado. Esto lo podremos ver si utilizamos la siguiente opción del menú principal: Perfil. Al pulsar esta opción nos llevará a una pantalla que sigue el diseño de todas las que hemos visto hasta ahora: cuadros de texto donde se estará desplegando toda la información necesaria.

Al entrar en esta pantalla, se rellenarán los campos de nombre de usuario, email y teléfono con los datos de nuestro usuario, y se hará una llamada a la API con la petición de que recupere todos los anuncios de nuestro usuario. Estos se recibirán y se mostrarán tal y como se hizo anteriormente, así que no se entrará mucho en los detalles.

13Perfil de usuario

La nueva opción que propone esta pantalla, además de poder ver tus datos y anuncios creados, es la de poder cambiar algunos datos de tu perfil.

En el menú de la esquina superior derecha, veremos un menú como el anterior, pero esta vez con una sola opción: Editar perfil.

Al pulsar sobre esta opción, se abrirá un cuadro de diálogo con las opciones que podemos cambiar, que por ahora serán nuestro nombre y número de teléfono. Cambiaremos lo que los apetezca y pulsaremos aceptar. Esto nos devolverá a la pantalla principal donde veremos qué en el mensaje de bienvenida personalizado, nuestro nombre se habrá cambiado en caso de haberlo editado.

14Diálogo de cambio de datos de usuario

Si vuelves a entrar en el perfil, se podrán ver reflejados los cambios también, ya que habrán sido enviados a la base de datos.

* **Salir**

Con esto ya hemos hecho una visita completa por todo lo que nos ofrece la aplicación. Solo nos queda despedirnos. Si pulsamos la última opción que nos queda en el menú de la pantalla principal, salir, se cerrará nuestra sesión y la aplicación volverá a la pantalla de log in, lista para que se vuelva a repetir todo el proceso de ciclo de vida de la aplicación.

## Capa de Negocio o Lógica de la aplicación

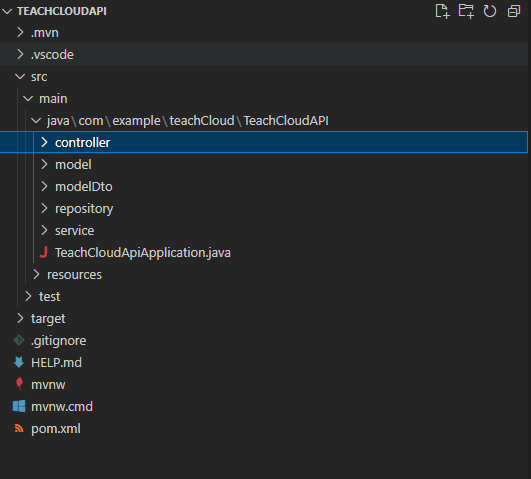
En la capa de negocio trataremos la API con la que se hará la transferencia de información entre la base de datos y el Front-end. Será una API Rest programada en Spring boot, con lenguaje JAVA y como proyecto de Maven.

Utilizaremos la versión 2.7.10 por elección del programador, y añadiremos las dependencias pertinentes para que funcione correctamente, además de las que vendrán por defecto:

* Spring web
* Spring data JPA
* Lombok
* Mysql-connector-java
* Spring security crypto

Empecemos a explicar la lógica detrás de la API. Lo primero de todo, hablaremos de los paquetes.

La API contará de 5 paquetes, los cuales almacenarán las clases correspondientes a la acción que diga cada paquete.

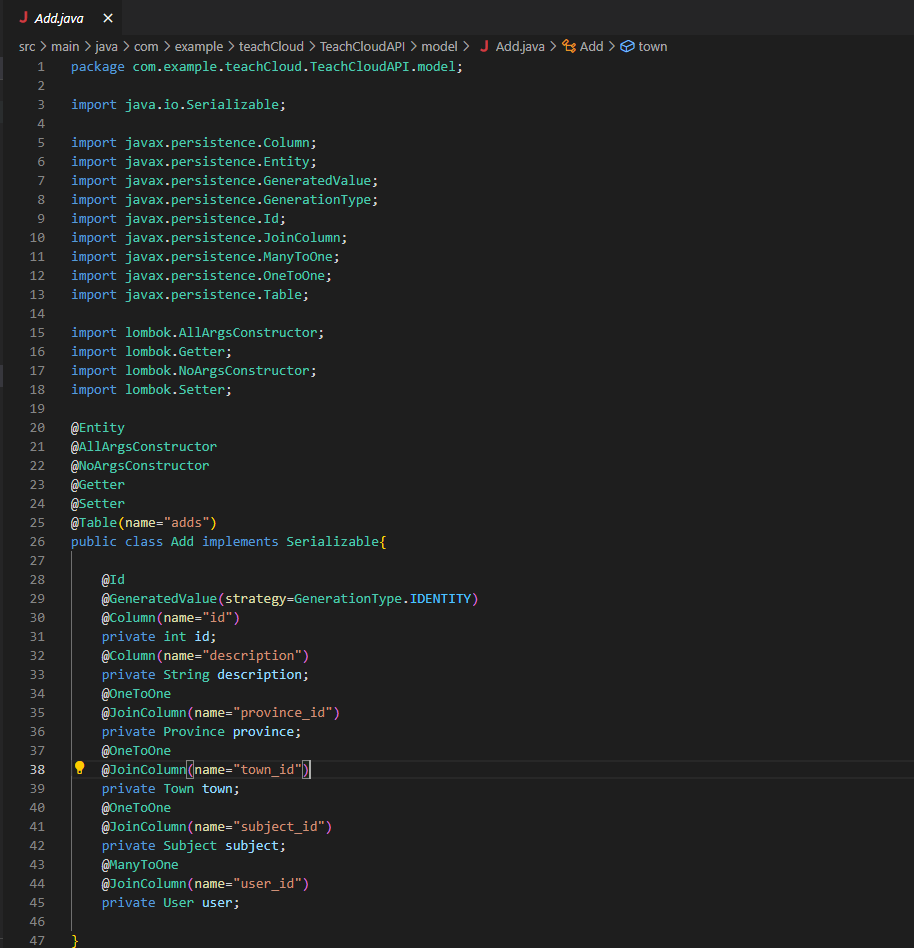


15 Organización del empaquetado en la API

El primero del que hablaremos será el paquete “model”. Este paquete tendrá la declaración de todas las clases que clasificaremos como entidades. Cada una de estas clases contará con:

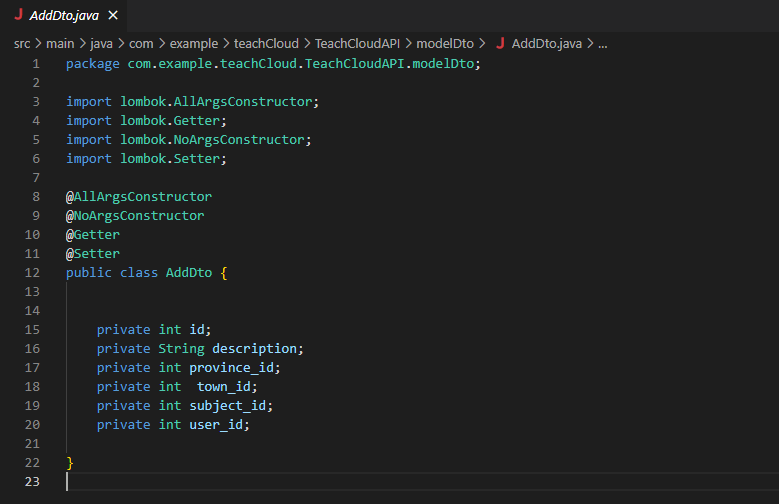
* La declaración de los atributos respectivos de cada entidad.
* Etiquetas necesarias para que se puedan referenciar como entidades y añadirlas posteriormente a la base de datos (@Entity, @Table, @Id, @Column).
* Etiquetas dadas por la dependencia de Lombok, que nos permitirán autogenerar constructores, getters y setters (@NoArgsConstructor, @AllArgsConstructor, @Setter, @Getter).
* Etiquetas de unión entre entidades si se diese el caso de que algunas de estas entidades tienen como atributo otra de ellas (@OneToOne, @OneToMany, @ManyToOne, @JoinColumn). Todas estas son dadas por la importación de “javax.persistence” en esta versión de Spring boot, y permiten que las entidades se puedan reflejar luego en nuestra base de datos. En versiones posteriores de Spring boot (a partir de la 3.0.0), estas dependencias vendrán dadas por “jakarta.persistence”.

Todas estas clases, además deben de implementar “Serializable”.



16Ejemplo clase en paquete “model”

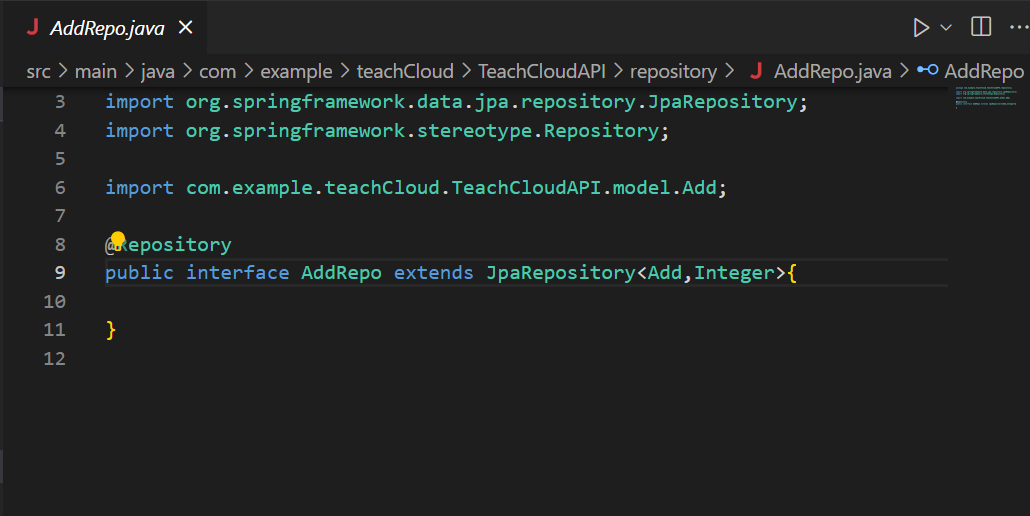
El siguiente paquete será “modelDto”. Es un paquete similar a model: guardará las clases DTO (Data Transfer Object) de cada una de las entidades que luego irán a la base de datos. Estas clases serán una copia de las entidades a las que representan, pero se quitarán todas las etiquetas menos las que pertenezcan a Lombok, la implementación de “Serializable” y se sustituirán todos los atributos que sean otra entidad, solo por su Id.



17Ejemplo clase en paquete "modelDto"

Estas clases se utilizarán a la hora de recibir y mandar datos a la Front-end de la aplicación, en vez de mandar el objeto marcado como entidad directamente.

Continuaremos con el paquete “repository”. Este contendrá una interfaz que servirá de repositorio por cada una de las entidades que tengamos. Este repositorio contará con la etiqueta @Repository, para marcarle a Spring boot que será una clase repositorio, y extenderá de JPARepository<>, recibiendo dentro 2 parametros: primero el tipo de objeto del que será repositorio, y segundo el tipo de la clave primaria de dicho objeto (véase un ejemplo en la próxima imagen).



18Ejemplo clase en paquete "repository"

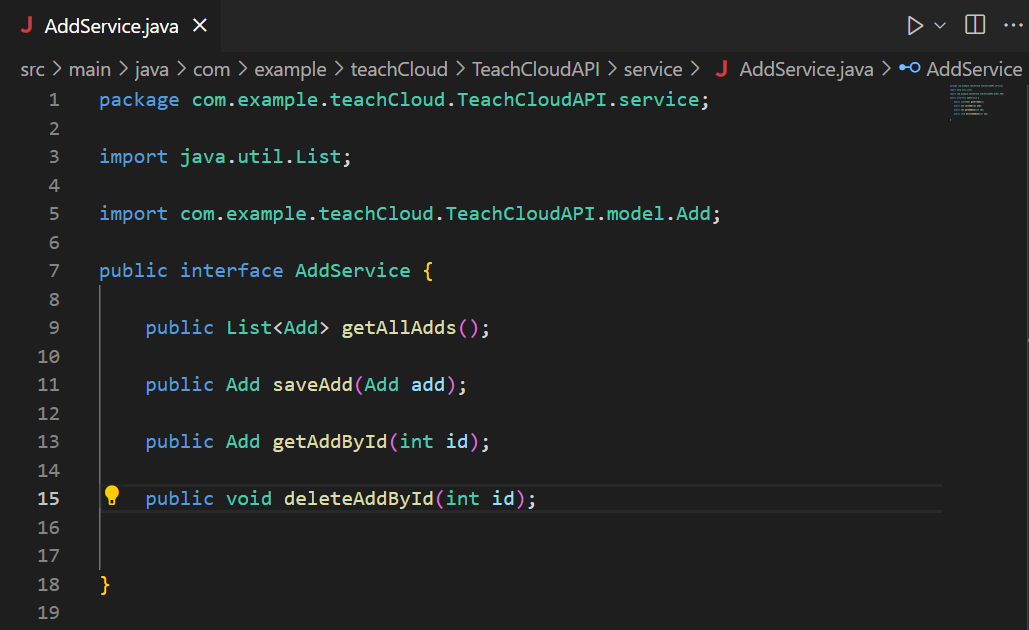
Esta clase servirá para que luego la utilice una clase “service”, de la cual hablaremos ahora, y utilizará los métodos que extiende de JPARepository.

En este caso, la interfaz no tiene ningún método, y no los necesita, pero en caso de que fuese necesario dado que JPARepository no tuviese un método del cual vamos a hacer uso, se podrían implementar sin problema.

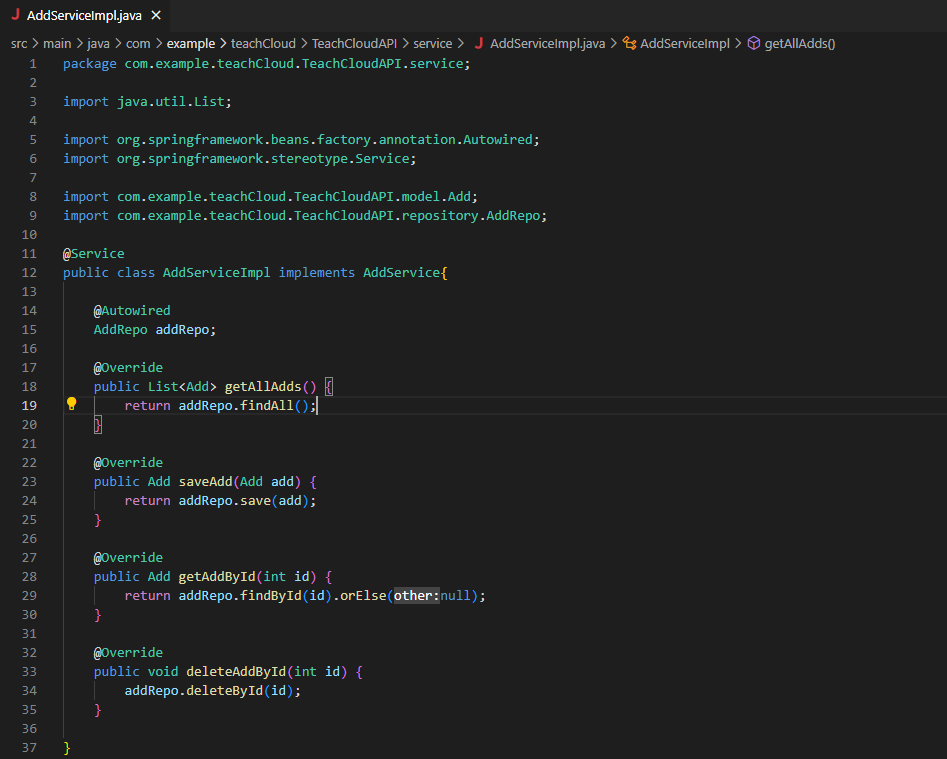
Continuando con el paquete “service”, este contendrá 2 clases por cada entidad: una llamada Service, y otra ServiceImpl.

La clase Service será una interfaz, después implementada por ServiceImpl, en la cual declararemos los métodos que necesitaremos para cada entidad. Por norma general, serán los métodos de un CRUD (Create, Read, Update, Delete), pero se añadirán más si fuese necesario.

La clase ServiceImpl será la que implemente los métodos definidos en Service. Tendrán una lógica simple y se utilizará la etiqueta @Autowired hacia el repositorio para llamar a los métodos de JPA y que estos introduzcan o devuelvan los datos. También ha de tener la etiqueta @Service para marcar que es un servicio.



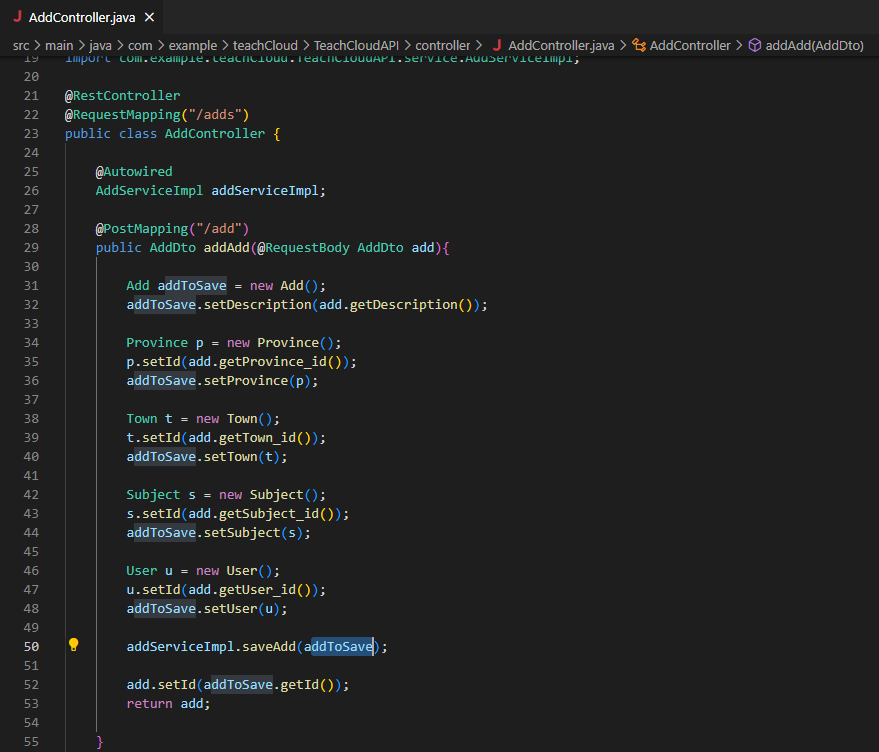
19Ejemplo clase service en paquete "service"



20Ejemplo clase ServiceImpl en paquete "service"

Por último, tenemos el paquete “controller”. En este paquete se guardarán los controladores de todas las entidades, y estos serán los que se encarguen de recibir y manejar las peticiones HTTP que se hagan hacia la API.

Cada entidad tendrá su propio controlador, el cual deberá tener la etiqueta @RestController, @RequestMapping con la dirección de este controlador específico, y las diversas peticiones HTTP, además de la anteriormente mencionada etiqueta @Autowired, esta vez hacia la clase ServiceImpl de cada entidad. Utilizaremos una imagen de controlador de los anuncios y el método de añadir un anuncio a la base de datos como ejemplo para ayudar con la explicación.



21 Ejemplo clase en paquete "controller"

Con la imagen de arriba de referencia, podemos ver como se implementan todas las etiquetas mencionadas anteriormente y el método de añadir a la base de datos. @PostMapping nos dice que tiene que recibir una petición POST, y en el cuerpo vemos que recibe un objeto Dto, como se mencionó en el apartado dedicado a este tipo de clases.

Al recibir esto, crearemos un objeto Add, introduciremos todos los datos que nos han llegado pro parte del Front en el objeto Dto y llamaremos al método save de addServiceImpl para que añada el nuevo objeto a la base de datos con los datos que nos llegaron en el cuerpo de la petición.

Después, como respuesta de que todo ha salido correcto, devolveremos el objeto creado como objeto Dto al Front.

Hemos utilizado de ejemplo la clase Add y todo lo relacionado con ella para explicar el funcionamiento básico de la API. Esta contará por supuesto con el resto de entidades que son necesarias para el funcionamiento de la aplicación, y todas seguirán la misma estructura que que hemos visto para la clase Add, variando ligeramente en algunas cosas si se fuese a necesitar de métodos propios o de alguna otra cosa.

Teniendo claro el esquema de nuestra API y todo lo que va asociado a una entidad específica, se debería poder replicar esta explicación para cada una de las diferentes entidades con las que contaremos, así que no repetiremos la explicación anterior con todas ellas.

En cuanto al resto de ficheros, mucho de ellos no son de nuestro interés en este momento ya que son generado automáticamente al crear el proyecto de Spring boot. Aun así, hay otros 3 ficheros de los que debemos hacer mención.

El primero será TeachCloudAplication.java. Este es uno de los ficheros autogenerados por el Spring Initializr, y es el que contiene el método Run() para que nuestra aplicación funcione. No hace falta hacer nada con él, pero es necesario saber que es el fichero que empieza la ejecución de nuestra API.

Los 2 que nos quedan son ficheros de configuración: aplication.properties (dentro del paquete resources), u Pom.xml.

Aplication.properties es donde definiremos la conexión con nuestra base de datos: ruta, nombre de usuario, contraseña, en que puerto va a estar ejecutándose… Es necesario saber la función de este fichero para que, en caso de querer cambiar a que base de datos se conectará nuestra API, tener claro donde debemos referenciarlo.

Por último, tenemos el Pom.xml. Este fichero será donde se introduzcan todas las dependencias que sean necesarias para nuestro proyecto, y es necesario saber su utilidad para que en el caso de querer añadir más dependencias, o saber cuales estamos utilizando, tener claro donde ir a buscarlas.

## Capa de persistencia de datos

En esta capa hablaremos del diseño que tiene la base de datos. Primero de todo, estará hecha en MySQL, un modelo de bases de datos relacional. Se ha escogido este tipo de base de datos frente a otras bases de datos relacionales como PostgreSQL e incluso sobre bases de datos no relacionales como MongoDB debido a la experiencia acumulada de utilizar este tipo de bases de datos durante un par de años, frente a 1 mes de formación en MongoDB y unas pinceladas de PostgreSQL.

Antes de explicar las tablas de la base de datos, cabe mencionar que en la primera versión de la aplicación algunas de estas no estarán completas, como las próximamente mencionadas tablas de provincias, asignaturas, etc. Se utilizará una base de datos de prueba para la primera versión de la aplicación, y se implementarán los datos necesarios en actualizaciones futuras.

La base de datos constará de las siguientes tablas:

* **Users:** será la tabla donde se guarden los datos de los usuarios. Esta contará con los campos necesarios para la autentificación de los usuarios y para cierta personalización de su perfil. La tabla contará con los siguientes campos:
* **Id(int):** el identificador unitario de cada usuario.
* **name(varchar):** el nombre con el que se verá cada usuario
* **email(varchar):** el email asociado a cada cuenta de usuario
* **password(varchar):** la contraseña de cada usuario (la cual se encontrará encriptada dentro de la base de datos).
* **phone(varchar):** el numero de teléfono con el que se podrá contactar con los usuarios
* **Provinces:** tabla donde se guardarán todas las provincias que estarán disponibles a elegir dentro de la aplicación. Estas luego tendrán asociadas ciertas ciudades o localidades para luego ser utilizadas en los anuncios.

Tendrá los siguientes campos:

* **id(int):** el identificador unitario de cada provincia.
* **Name(varchar):** el nombre de la provincia en cuestión.
* **Towns:** tabla donde se guardarán todas las ciudades o localidades que estarán disponibles a elegir dentro de la aplicación. Estas irán asociadas a una provincia mediante un identificador.

Tendrá los siguientes campos:

* **id(int):** el identificador unitario de cada ciudad o localidad.
* **name(varchar):** el nombre de la ciudad en cuestión.
* **province\_id(int):** el id de la provincia a la que va asociada esta ciudad.
* **Subjects:** tabla donde se guardarán todas las asignaturas que estarán disponibles al crear un anuncio.

Tendrá los siguientes campos:

* **id(int):** el identificador unitario de cada asignatura.
* **name(varchar):** el nombre de la asignatura.
* **Adds:** tabla donde se guardarán todos los anuncios que pongan los usuarios. Esta tabla tendrá asociada mediante claves foráneas el usuario que puso el anuncio, una provincia, ciudad y asignatura.

Tendrá los siguientes campos:

* **id(int):** el identificador unitario de cada anuncio.
* **description(varchar):** la descripción que el usuario le haya querido poner al anuncio.
* **Province\_id (int):** el id de la provincia asociada al anuncio.
* **Town\_id(int):** el id de la localidad o ciudad asociada al anuncio**.**
* **Subject\_id(int):** el id de la asignatura que se pretende impartir en el anuncio.
* **User\_id(int):** el id del usuario que ha puesto el anuncio.

Esta base de datos estará alojada en local por el momento, y utilizaremos Docker Desktop para guardar y correr la imagen de MySQL necesaria además de HeidiSQL para poder visualizar la base de datos.

# IMPLEMENTACIÓN

## Tecnologías utilizadas en el desarrollo del proyecto

Aquí listaremos la batería de tecnologías utilizadas durante el desarrollo de la aplicación, desde el planteamiento de la idea hasta su final despliegue:

**Diseño de la base de datos**

Una pieza fundamental de cualquier aplicación es la base de datos que irá asociada a ella. Para crear el esquema de la misma, así como todos los diagramas entidad-relación, modelos relacionales, etc… utilizaremos la extensión de Draw.io para el Visual Studio Code.

**Creación de la base de datos**

Para crear la base de datos utilizaremos Heidi SQL, un software de gestión y diseño de base de datos para MySQL.

Esta nos servirá tanto como para el modelado de las tablas como su posterior edición y gestión.

Debido a que tendremos la base de datos en local, utilizaremos Docker para abrir la conexión a la base de datos.

**Docker Desktop**

Una aplicación de escritorio que nos permite guardar imágenes de distintos tipos de bases de datos, como MySQL, PostgreSQL y muchas otras, guardarlas en un contenedor y poder ejecutarlas en segundo plano para poder tener acceso a nuestras bases de datos.

**Desarrollo de la aplicación**

La aplicación será desarrollada en Android Studio, y el lenguaje escogido será Java, un lenguaje de programación orientado a objetos. A la hora de usar Android Studio también será necesario XML, con el que se creará el formato de las distintas pantallas de la aplicación.

**Desarrollo de la API**

La API se desarrollará utilizando Java y el Framework “Spring Boot”. La API se utilizará a la hora de realizar la conexión entre el Front-end en Android y la base de datos, permitiendo al Front-end realizar un CRUD (Create, Read,Update, Delete) sobre las distintas entidades de la base de datos.

## Descripción del proyecto

En este apartado, explicaremos los pasos para instalar nuestra aplicación y poder tenerla corriendo en cualquier sitio. Para esto, lo dividiremos en estas capas.

* + 1. Capa de presentación

En esta capa explicaremos como instalar toda la parte del front-end de la aplicación. Para esto tenemos 2 opciones: utilizar un emulador dentro del android studio, o descargar la aplicación en un dispositivo móvil Andoid. Procederemos a explicar ambas opciones:

* **Emulador en android studio:** para utilizar esta opción, será necesario descargar el programa android studio en nuestro ordenador. Este se puede conseguir desde la página oficial de android studio. Después de descargarlo, abrir el archivo y comenzar la instalación. Con las opciones por defecto debería ser más que suficiente.

Cuando esté ya instalado, se tendrá que ir a Open y abrir el proyecto de la aplicación, que se podrá descargar de la entrega del proyecto. Con el proyecto ya en el android studio, al darle a la fecha de verde de ejecutar debería empezar a correr la aplicación en uno de los emuladores por defecto del android studio.

* **Descargar la aplicación a un dispositivo móvil:** para esta opción descarguese la APK que encontrará en la entrega del proyecto e instalela en su dispositivo movil. En cuanto termine la instalación, deberái estar disponible para utilizar.

Con esto, la aplicación ya estará instalada en nuestro dispositivo pero debido a que actualmente solo funciona el local, todavía hay que hacer más cosas que explicaremos en las siguientes capas.

* + 1. Capa de negocio o Lógica de la aplicación

En esta capa explicaremos como poner en funcionamiento la API Rest de la aplicación. Este proceso es más simple que el de la capa de presentación.

Primero, será descargarnos de la entrega el archivo con la API. En cuanto esté descargada, se tendrá que abrir con un editor de código. En este caso para su desarrollo se ha utilizado el Visual Studio CODE, pero se recomienda que se use el favorito de cada usuario. Otras opciones recomendadas pueden ser el IntelliJ o Eclipse.

Cuando se tenga abierto en el IDE elegido, pulsar el botón de ejecutar deberái poner en marcha la API dentro del localhost del ordenador, pero todavía hay que hacer un último paso antes de que esté todo en funcionamiento.

De manera adicional para la prueba de la api, se adjuntará una colección de consultas para utilizar con la herramienta Postman y poder probar la API sin tener que utilizar la aplicación.

* + 1. Capa de persistencia de datos

Esta será la última capa de instalación. Lo primero será tener un programa de gestión de base de datos y algo para alojar una base de datos en localhost. En este caso, el gestor de base de datos ha sido HeidiSQL y se ha alojado en un container utilizado Docker, pero al igual que para la API, se pueden utilizar otros gestores como MySQL Workbench o Dbeaver, y cualquier otra manera de alojar una base de datos en localhost como el XAMPP.

En cuanto estas 2 cosas esten preparadas, se tendrá que entrar en localhost con usuario root y contraseña “teachpwd” en el gestor de base de datos. Después se cargará un script que irá en la entrega del proyecto y con esto, todos los datos necesarios de la base de datos deberían cargarse.

Teniendo ya todo preparado, y asegurandose de que el dispositivo donde está funcionando la API y la aplicación android esten conectados a la misma red, la aplicación deberái estar en funcionamiento y lista para su uso.

# PRUEBAS

## Introducción

En este apartado se describirán de manera breve las pruebas necesarias para asegurarse de que la aplicación y todos sus componentes funcionan de una marear adecuada.

* + 1. Capa de presentación

La manera que se ha utilizado para realizar las pruebas a la aplicación ha sido las múltiples ejecuciones de la misma para asegurarse de que todo funciona como debe ser. Entre las muchas cosas que se han tenido que probar, las más relevantes serían:

* Asegurarse de que que se manda y recibe información desde la API: la conexión con la API y la base de datos es de vital importancia para el funcionamiento de la aplicación. Se han hecho numerosas pruebas para asegurarse de que la conexión es exitosa, ya que se necesita enviar el formulario de registro, hacer el log in a la aplicación, recibir la información del usuario, y traer los anuncios desde la base de datos. Se ha comprobado que los endpoints y las funciones cumplen con sus respectivos patrones, además de asegurarse de que todos los ficheros de configuración de la conexión tengan los valores correctos.
* Comprobación del funcionaminto de todas las pantallas: la aplicación cuenta con una multitud de pantallas, y se necesita que todas estas funcionen de una manera adecuada. Entre las comprobaciones que se han hecho están, que la aplicación no falle al cambiar entre pantallas, que se persista la información entre las distintas pantallas, que cada cambio de actividad lleve a la siguiente actividad deseada...

Todos estos puntos han sido probados y tienen el funcionamiento esperado.

* Persistencia de datos correcta: la informaciónn se debería mantener entre los cambios de pantallas, y mediante pruebas al cambiar los datos se ha podido probar que se mostrará sin la información adecuada para cada caso en todas las pantallas, ya sean textos explicativos, o información sacada desde la base de datos sobre un usuario o anuncio.
  + 1. Capa de negocio

En esta capa trataremso las pruebas realizadas a la API.

Lo primero que se necesitó probar fue que se ejecutase correctamente mediante la configuración de todas las dependencias y del fichero aplication.properties. Estos tendrán todo l necesario para que funcione correctamente , además de las credenciales de acceso a la base de datos bien configuradas en el fichero de las propiedades, sinedo estas la ruta donde estará la base de datos (localhost) el usuario (root) y la contraseña (teachpwd), además de e puerto donde la Api estará escuchado (9000). Después, utilizamos la herramienta Postman y una serie de peticiones propias para asegurar que todos los endpoints devuelven la información deseada.

Una muestra de estas peticiones estará adjuntada en la entrega en un fichero .JSON para utilizar en el Postman en caso de que se quiera probar.

* + 1. Capa de persistencia

En cuanto a la base de datos, no hay tantas pruebas que realizar como en las otras 2 capas. Realizar la conexión con usuario y contraseña para acceder es lo mínimo. Después realizar las consultas SQL pertinentes para la creación de las tablas y los datos que sean necesarios. Por último, el gestor de base de datos nos dejará ver si esos datos que introdujimos están ahí, por lo que no hay pruebas necesarias.

El único ejemplo de prueba que podemos mostrar en este caso, sería utilizar las peticiones de Postman de las que hablamos en el apartado anterior junto con la API, y ver que la información que nos devolverá es la misma que podemos ver en nuestro gestor.

# CONCLUSIÓN

## Valoración Personal del Trabajo Realizado

Durante la realización de el proyecto me he dado cuenta de que todavía queda mucho por aprender. Soy consciente de cosas mejorables de la aplicación, pero también de que en un tiempo limitado no es posible realizar todo lo que uno tiene en mente.

Ha sido un año complicado, con muchos cambios de ritmo y un montón de cosas nuevas que saber hacer, y aún así, he notado que estamos en la superficie de todo.

Ha sido necesario aprender muchas cosas por cuenta propia para realizar este proyecto, ya que habíamos hecho muchas de estas cosas por separado en distintas ocasiones, peor sin ver como luego todo tiene que encajar a la perfección o las cosas empiezan a desmoronarse.

De no ser por lo aprendido durante el periodo de prácticas esto hubiese sido muy distinto, ya que tuve que aprender de cero como diseñar una API, y estoy seguro de que es algo que no hubiese sido capaz de hacer de no ser por el tiempo empleado para aprenderlo en el trabajo para realizar las tareas que se me mandaban. Hubiese tenido que recurrir a alguna alternativa a la hora de conectar el Front con la base de datos como he visto a otros compañeros tener que hacer.

Teniendo todo en cuenta: el limitado tiempo, la necesidad de emprender múltiples nuevas tecnologías y trucos, el estrés por ser la primera vez que me enfrento a un trabajo de esta magnitud… ha sido una experiencia difícil, pero también enriquecedora y de la cuál he sido capaz de aprender mucho para futuros proyectos, ya sea en el ámbito laboral o personal.

## Posibles ampliaciones

Este proyecto tiene un margen de ampliación bastante grande y me hubiese gustado poder implementar más funcionalidad a la aplicación, pero por falta ya sea de tiempo, de experiencia o de conocimiento no ha sido posible.

Entre las diversas ampliaciones que podríamos añadir al proyecto se encuentran:

**-Implementación de la aplicación en otros dialectos:** debido a la naturaleza de la aplicación de querer ayudar a los alumnos a encontrar profesores de bachillerato, parece algo innecesario tratar de llevarla a otros países, por lo que traducirla no parece una muy buena idea. Sin embargo, dentro del territorio español contamos con diversos e idiomas y dialectos qué seguro que apreciarían poder encontrar la aplicación en catalán, gallego, vasco, etc.

**-Diferenciación más marcada entre alumnos y profesores:** en la versión 1.0 de la aplicación, la diferencia entre quien es un alumno y quien un profesor, viene marcada en el anuncio, y no en el usuario como tal.

Para una versión próxima, se implementaría una diferenciación más marcada entre estos 2 roles, introduciendo otros parámetros como las asignaturas que un profesor puede dar, que sea el perfil el que pueda recibir reseñas en lugar del anuncio que pone, entre otras muchas.

Por parte del alumno, poder poner en su perfil que bachillerato es el que está estudiando, las asignaturas con las que tiene dificultades y alguna otra cualidad.

**- “Quality of life improvements”:** es un término utilizado comúnmente para referirse a mejoras con la interfaz, hacer la navegación más amigable etc. Un ejemplo de estas mejoras podría ser la implementación de un modo oscuro para la gente que lo prefiera.

**-Añadir imágenes a los perfiles:** esto es un claro ejemplo de una funcionalidad que no se ha podido realizar por falta de tiempo y experiencia. Añadir una imagen a los perfiles daría un toque extra de profesionalidad y de personalización al perfil de cada usuario.

**-Reseñas:** la idea de poder poner una serie de puntuaciones ya bien sea al perfil de un usuario o a un anuncio se barajó en un principio, peor debido a la falta de tiempo se tuvo que descartar. Aun así, sería una de las ampliaciones con más prioridad de esta lista, ya que ayudaría al usuario a ver que profesores tienen mejores recomendaciones de otros usuarios.

# BIBLIOGRAFÍA Y WEBGRAFÍA

- Introducción a Android Studio. (s. f.). *Android Developers*. https://developer.android.com/studio/intro?hl=es-419

- *Genuine Coder - Learn to write genuine programs*. (2022, 4 julio). Genuine Coder. https://genuinecoder.com/