Prototipo de aplicación destinada a la búsqueda de grupos de estudio y mentor.

Grado en Ingeniería Informática

Tutor: Abraham Rodríguez Rodríguez

Autor: Jonay Efrén López Pérez

Enero de 2017

1. INTRODUCCIÓN

1.1. Contexto

Este proyecto nace con la idea de promover el estudio en grupo entre los estudiantes de la ULPGC. El estudio cooperativo es probablemente el paradigma educativo mejor documentado y sobre el que más se ha investigado. Es por lo que se conocen de sobra sus múltiples virtudes. Entre ellas destacamos las siguientes:

- **Promueve la implicación activa del estudiante en el proceso de aprendizaje.**
- **Capitaliza la capacidad que tienen los grupos para incrementar el nivel de aprendizaje mediante la interacción entre compañeros.**
- **Reduce los niveles de abandono de los estudios.**
- **Promueve el aprendizaje independiente y autodirigido.**
- **Permite acomodar los diferentes estilos de aprendizaje de los estudiantes de hoy día.**
- **Facilita un mayor rendimiento académico en las áreas de matemáticas, ciencia y tecnología.**

Y sobre todo prepara a los estudiantes a trabajar en grupo tal y como lo harían en un entorno de trabajo real.

Es por ello por lo que se propone el desarrollo de un prototipo de aplicación móvil mediante la cual los estudiantes podrán encontrar grupos de estudios de la asignatura en la que estén interesados con bastante facilidad. También se puede participar en distintos programas como el de Apadrina un alumno, mediante el cual alumnos veteranos ofrecen ayuda a alumnos de nuevo ingreso o que llevan pocos años en la universidad con temas tanto académicos como de gestión de matrícula, documentación, campus virtual …etc.

1.2. Objetivos

Como resultado del trabajo, los usuarios de nuestra aplicación móvil serán capaces de:

- Buscar grupos de estudio abiertos para una asignatura específica.
- Encontrar un padrino.
- Tener acceso a un repositorio privado para cada grupo de estudio.
- Subir cualquier tipo de ficheros al repositorio de grupo.
- Visualizar perfiles de estudiantes, en los que se encuentran las asignaturas de cada uno.
- Inscribirse como padrino en el programa Apadrina un alumno.

En resumen, el objetivo principal de este proyecto es promover el estudio en grupo entre los alumnos. Hoy en día, los estudiantes cuentan con un sinfín de herramientas software que les sirven como ayuda en el proceso del aprendizaje. Mediante nuestra aplicación móvil no solo serán capaces de encontrar grupos de estudio u otros alumnos que los ayuden con ciertas materias, sino que contarán con herramientas tales como repositorio de ficheros en la nube o una vía de comunicación instantánea.

Así mismo, como estudiante se han propuesto los siguientes objetivos principales:

- Ampliación de conocimiento en las tecnologías de desarrollo Ionic3 y los distintos lenguajes que componen su arquitectura (**Angular4, TypeScript, HTML5, CSS3, SASS**)
- Familiarizarse con un entorno de trabajo que utiliza datos asíncronos en un servidor remoto, como es **FireBase**.

1.3. Metodología

Para el desarrollo de este proyecto se ha seguido una metodología de desarrollo iterativa e incremental.

En un desarrollo iterativo e incremental el proyecto se planifica en diversos bloques temporales (normalmente entre dos semanas y un mes) llamados iteraciones.

En cada una de las iteraciones se evoluciona el producto (entrega incremental) a partir de los resultados de iteraciones anteriores, añadiendo nuevas funcionalidades o realizando modificaciones sobre las ya implementadas si el cliente así lo desea. Podemos observar las fases del modelo iterativo e incremental en la Ilustración 1**.**

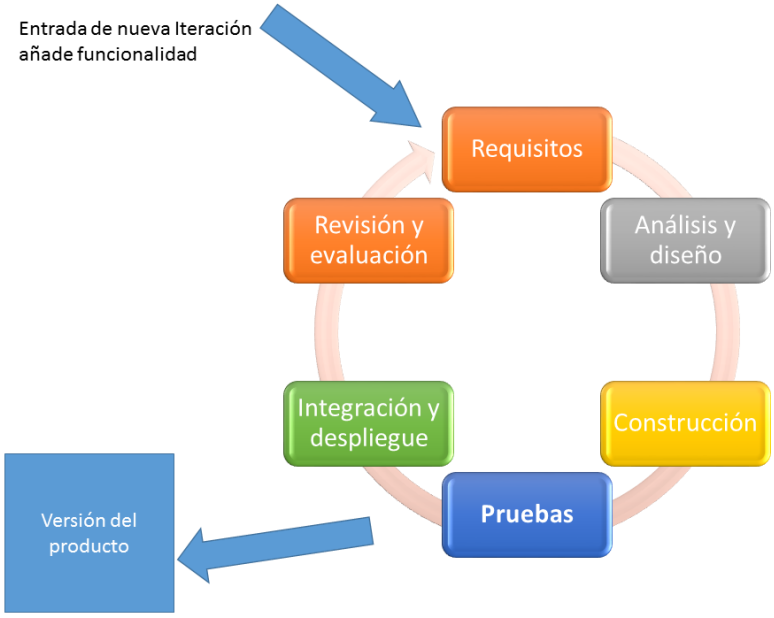

Ilustración 1 - Fases del modelo iterativo e incremental

Se ha elegido esta metodología de desarrollo debido a que desde un principio no todas las funcionalidades o requisitos estaban definidos y se fueron redefiniendo a lo largo del proceso de desarrollo con el tutor, por lo que este tipo de metodología nos facilita la integración de requisitos en cualquier momento. Así mismo, al llevarse a cabo iteraciones en pequeños ciclos se favorece la gestión de las entregas.

Fases	Duración estimada	Tareas
Estudio previo / Análisis	30	**Tarea 1.1: Documentación del lenguaje**
		Tarea 1.2: Análisis estructural de la aplicación
Diseño / Desarrollo / Implementación	180	**Tarea 2.1:Desarrollo de la aplicación**
		Tarea 2.2:Creación Historias de usuario
		Tarea 2.3: Sprint 1, implementación de funcionalidades básicas.
		Tarea 2.4: Sprint 2, Implementación de funcionalidades secundarias
		Tarea 2.5: Sprint 3, mejora de las vistas de la aplicación.
Evaluación / Validación / Prueba	60	**Tarea 3.1: Validación y pruebas de los componentes integrados.**
Documentación/ Presentación	30	**Tarea 4.1: Realización de la memoria del trabajo**

Tabla 1 - Planificación del proyecto

Utilizando esta metodología, la planificación propuesta para el trabajo de fin de grado se ha podido llevar a cabo sin cambios significantes en cuanto al tiempo. Sin embargo, en lo que se requiere a las tareas de la parte de desarrollo, se han producido los siguientes cambios.

Sprint o iteración 1: dedicada a la implementación de inicio de sesión y gestión de grupos de estudio.

Sprint o iteración 2: dedicada a la implementación de funcionalidades de apadrinamiento y de subida de documentos a la nube.

Sprint o iteración 3: dedicada a la mejora de las vistas y de la implementación de funcionalidades requeridas por el tutor.

1.4. Competencias específicas cubiertas

Durante el desarrollo de este proyecto, se han visto cubiertas las siguientes competencias específicas relacionadas con la intensificación Tecnologías de la Información:

TI06.- *Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.*

Esta competencia se ve cubierta ya que el proyecto ha sido desarrollado utilizando tecnologías y servicios web, así como almacenamiento en la nube.

TI03.- *Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.*

Esta competencia se ve cubierta al haberse utilizado una metodología de desarrollo ágil para la organización, desarrollo y evaluación del proyecto realizado.

1.5. Aportaciones al entorno socioeconómico, técnico o científico.

Desde hace unos años y de forma exponencial, los smartphones han ido ganando repercusión en nuestras vidas, hasta el punto en el que son indispensables en nuestro día

a día. Este hecho, no solo crea la oportunidad para desarrollar y concebir aplicaciones para dispositivos móviles, sino que genera la necesidad de llevarlas a cabo.

Con la aplicación desarrollada se busca ofrecer a los estudiantes una herramienta en el entorno académico mediante la cual los estudiantes tengan facilidad de encontrar grupos de estudio y llevar a cabo un proceso de aprendizaje cooperativo.

Con el desarrollo de la aplicación, se tiene la esperanza de que en el entorno social, ayude a estudiantes con barreras de comunicación tales como la timidez o la falta de contactos en el entorno estudiantes, ya que la misma plantea bastantes facilidades a lo hora de buscar estudiantes sin grupo o que estén buscando grupos de estudio de una asignatura común.

1.6. Estructura de la memoria

Debido a que la metodología seguida en el desarrollo es una metodología iterativa e incremental, la estructura de la memoria se adapta a las fases que esta misma conlleva. En primer lugar, se realizará un análisis previo de la aplicación, en el cual se llevará a cabo un estudio de aplicaciones similares para a continuación elaborar una lista de requisitos, tanto funcionales como no funcionales. Para apoyarnos en esta etapa, realizaremos diagramas de casos de uso para las principales funcionalidades.

Una vez finalizados la fase de análisis de nuestra aplicación, describiremos la arquitectura software implementada en el desarrollo del proyecto y el modelo de la base de datos usado.

A continuación, describiremos los requisitos, detalles de implementación y las pruebas realizadas para validar el funcionamiento de la aplicación para cada una de las iteraciones realizadas en el desarrollo del proyecto. Al finalizar esta etapa de implementación de iteraciones se describirán las tecnologías utilizadas en el proyecto y se adjuntará un enlace a BitBucket donde se puede realizar un seguimiento del mismo. Para concluir, se añadirán las conclusiones que hemos extraído del desarrollo del mismo.

2. Análisis previo

En este apartado se describirá la fase de análisis del proyecto realizado. Se realizará un estudio de aplicaciones similares y se desarrollará una lista de requisitos tanto funcionales como no funcionales que nuestra aplicación debe tener. Una vez desarrollada la lista de requisitos de la aplicación, se expondrán unos mockups representativos de la aplicación.

Finalmente, se discutirán los aspectos legales y las normativas que deben ser consideradas en el desarrollo de este proyecto.

2.1. Estudio de aplicaciones similares

En este apartado realizaremos un análisis de aplicaciones similares del mercado. Destacar que, durante la búsqueda de aplicaciones similares no se ha encontrado ninguna tan especializada en la creación de grupos, pero sí se han encontrado aplicaciones destinadas al estudio. De estas aplicaciones se han extraído ciertas funcionalidades que se podrían añadir a nuestro proyecto.

StudyBlue

- Desarrollada por StudyBlue Inc
- Sistema operativo Android 4.0.3 o superiores.
- Versión Actual 5.49
- 1 millón de descargas.
- Última vez actualizada 1 de diciembre de 2017.

.StudyBlue se basa en que los usuarios crean contenidos educativos en forma de tarjetas flash y Quiz que pueden compartir con otros usuarios. Al crearse un grupo de tarjetas o de quiz, se forma un grupo o “clase” al que los distintos tipos de estudiantes pueden ingresar. Esta aplicación es interesante porque cuenta con profesores de universidad que desarrollan sus propios contenidos para la aplicación.

StudyBlue 1 - Resultados de Quiz

- Desarrollada por Brainly Inc.
- Sistema operativo Android 4.0.3 o superiores.
- Versión actual 4.2.0.5.
- 5 millones de descargas.
- Última vez actualizada 18 de diciembre de 2017.

Brainly es una comunidad de grupos creada por estudiantes de todo el mundo. Permite a los usuarios realizar consultas en grupos generales de asignaturas o temas específicos. A estos grupos puede entrar cualquier persona y lo conforman varios miles de estudiantes. Aun así, los estudiantes pueden crear sus grupos privados e invitar a otros estudiantes que estén interesados en el mismo tema. Sin embargo, Brainly no cuenta con un repositorio de documentos privado.

Brainly 1 - Preguntas

Como resultado del análisis de estas dos aplicaciones, se ha decidido que, aunque no entra dentro del alcance de implementación de nuestro prototipo, se implementará la funcionalidad extraída de BrainLy de poder formular preguntas dentro de una asignatura y tema que podrá ser contestada y vista por los demás usuarios de esa aplicación. Podemos ver un ejemplo en la figura BrainLy 1.

Por parte de StudyBlue, se ha decidido implementar una de sus funcionalidades algo modificada. En StudyBlue, los alumnos pueden crear contenidos de tipo quiz o examen, que los demás miembros de los grupos o de la comunidad pueden ver. Se ha pensado que para nuestra aplicación sería interesante que, al crearse este tipo de contenidos, los profesores pudiesen verificar como válidos o aptos, para dotar a estos contenidos de credibilidad. Vemos un pantallazo de la funcionalidad de quiz en la figura StudyBlue 1, donde se recogen los resultados del quiz.

2.2. Requisitos

El primer paso a la hora de identificar los requisitos es determinar aquellas funcionalidades que consideramos necesarias en nuestra aplicación.

A continuación, se expone a modo de catálogo de requisitos dichas funcionalidades.

Requisitos funcionales

Un requisito funcional define una función del sistema de software o sus componentes.

- **Inicio de sesión.**
  - El inicio de sesión debe realizarse mediante los datos de login institucionales de la ULPGC.
  - La cuenta del usuario debe ser recordada después de haber iniciado sesión y debe iniciar sesión automáticamente cada vez que se accede a la misma hasta que el usuario cierre sesión.
  - El sistema debe validar los datos de inicio de sesión.
  - Se redirigirá a la vista principal si el inicio de sesión ha sido exitoso.
  - Se mostrará un mensaje de error en caso de que los datos introducidos no sean correctos.

- **Grupos de estudio.**
  - Todos los estudiantes deben ser capaces de crear grupos de estudio.
  - Cada grupo de estudio tendrá nombre, una breve descripción y se crearán para una única asignatura.
  - Cada grupo de estudio contará con su propio repositorio de documentos privado.
  - Todos los miembros del grupo podrán subir documentos al repositorio y descargarlos.
  - Los miembros del grupo tendrán acceso al muro del grupo donde podrán ver y enviar mensajes en modo de chat.
  - El creador del grupo debe considerarse como dueño o administrador y podrá expulsar miembros del grupo o eliminar el grupo.
  - Los estudiantes ajenos al grupo solo verán la descripción del grupo y podrán enviar una solicitud de entrada al mismo.
  - Los grupos privados no aparecerán en ninguna lista de grupos, y solo se podrá formar parte de ellos mediante invitación.
  - Al invitar a alguien a un grupo, le llegará una notificación que deberá ser aceptada o rechazada.

  - Al enviar una solicitud de entrada a un grupo, le llegará una notificación al administrador del mismo que deberá ser aceptada o rechazada.
  - Toda actividad del grupo debe quedar registrada en un log.

- **Programa Apadrina.**
  - Los estudiantes deben poder inscribirse en el programa.
  - Deben tener un entorno privado en el que padrino y apadrinado sean capaz de comunicarse y compartir ficheros.
  - Los estudiantes deben poder ver una lista de padrinos para poder enviar una solicitud de apadrinamiento.
  - Se debe poder dejar de participar en el programa en cualquier momento.

- **Requisitos generales.**
  - Se debe poder acceder a una lista de grupos públicos.
  - La lista de grupos públicos debe poder filtrarse por asignatura.
  - Los usuarios pueden buscar grupo de forma pasiva, apuntándose a listados de usuarios que buscan grupo para una asignatura en concreta.
  - Los administradores de grupos deben poder invitar a usuarios que estén buscando grupos cuya asignatura sea la misma que la del grupo del administrador.
  - El administrador de grupo puede borrar ficheros del repositorio.
  - Los usuarios pueden cerrar sesión en cualquier momento.

Requisitos no funcionales

Un requisito no funcional o atributo de calidad es un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.

Rendimiento

Una de las grandes desventajas de las aplicaciones hibridas es que el rendimiento suele ser menor que el de las aplicaciones desarrolladas de forma nativa, ya que en este caso ionic levanta un navegador local a pantalla completa e inicia la aplicación, notándose cierta demora en el arranque.

Para cada vista se esperan tiempos de carga bajos, no mayores de un segundo al navegar por la aplicación. (A excepción de la subida y descarga de ficheros)

Los cálculos realizados por la aplicación deben ser ligeros y que no supongan una carga importante para el dispositivo, esperándose un rendimiento óptimo en este sentido y haciendo que la aplicación pueda ser ejecutada por una amplia gama de dispositivos móviles.

Interfaz y usabilidad

La aplicación debe contar con una interfaz sencilla e intuitiva, que haga uso en su paleta de colores de los colores institucionales.

La aplicación debe contar con un menú lateral por el que el usuario pueda moverse por las distintas secciones de la aplicación.

Seguridad

Las contraseñas de los usuarios deben cifrarse en todo momento.

2.3. Diagramas de casos de uso

Una vez definidos los requisitos y las características de nuestra aplicación se presenta el diagrama de casos de uso principales para un usuario autenticado, ya que a un usuario no logeado en el sistema solo se le permite ver un listado de grupos públicos.

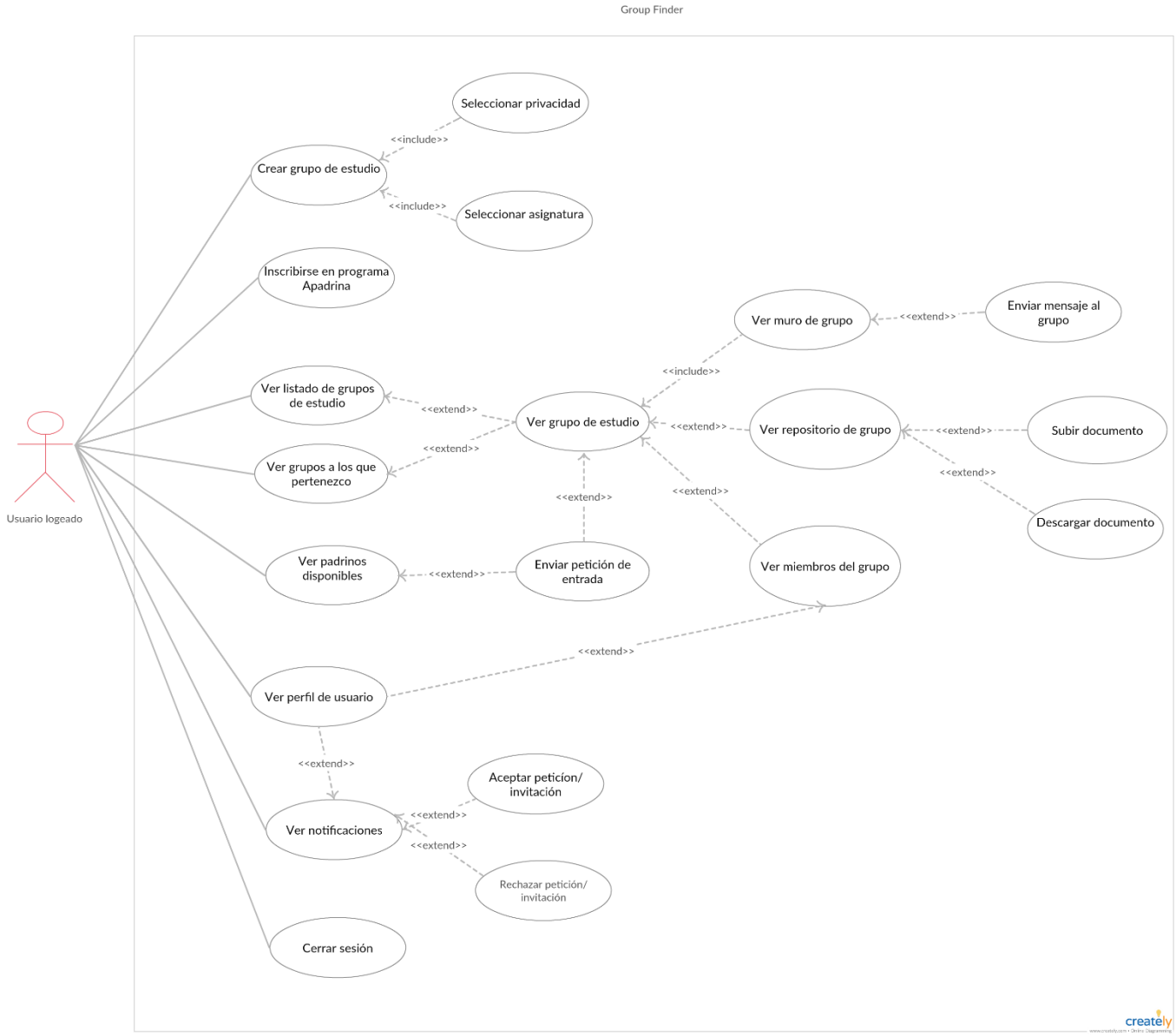

Diagrama 1 - Casos de uso para un usuario autenticado

Para la creación del diagrama de casos de uso, se ha hecho uso de la herramienta web CreateLy (https://creately.com).

Atendiendo al Diagrama 1, describiremos los principales casos de uso implementados en la aplicación.

A continuación, un listado de los casos de uso implementados:

- Iniciar sesión: el usuario debe identificarse en la aplicación mediante el uso de un correo electrónico y una contraseña. La tecnología de autenticación utilizada es la de FireBase Authentication, y se ha intentado "*mockear*" el uso de cuentas institucionales

- Crear grupo de Estudio: los usuarios podrán crear grupos de estudio especificando una asignatura, una breve descripción y la privacidad del mismo.

- Inscribirse en programa Apadrina: los usuarios podrán inscribirse en el programa Apadrina, creando un grupo vacío desde el que pueden recibir peticiones de unión o desde el que pueden invitar a alumnos en busca de mentor. Deberán especificar información de por qué participa en el programa y que ofrece.

- Ver listado de grupos: el usuario tendrá a disposición una lista de grupos públicos a los que poder enviar una petición de unión.

- Cerrar sesión: el usuario será capaz cerrar sesión en cualquier momento.

- Ver grupo: los usuarios podrán acceder a una vista de un grupo en concreto, que se divide en tres pestañas: una para el muro del grupo, otra para miembros y otra para el repositorio. Estas dos últimas son solo accesibles para los miembros del grupo, los usuarios que no pertenezcan al grupo solo verán el muro, donde se encontrarán con una descripción del grupo y un botón para enviar solicitud de entrada.

- Ver repositorio de grupo: los usuarios miembros de un grupo podrán acceder al repositorio privado del mismo.

- Ver grupos a los que pertenezco: los usuarios tendrán en el menú lateral acceso a una página donde vean y puedan acceder directamente a los grupos en los que son miembros

- Ver Padrinos disponibles: los usuarios dispondrán de una lista de grupos de padrinos que estén participando en el programa Apadrina.

- Ver perfil de usuario: los usuarios son capaces de visualizar tanto su perfil como el de otros usuarios.

- Abandonar programa Apadrina: los usuarios que participen en el programa Apadrina, dispondrán de un botón que servirá para dejar de participar en el mismo.

- Cambiar foto de perfil: los usuarios tienen a disposición en su perfil de usuario de un botón desde el que podrán cambiar su foto de perfil.

- Filtrar grupos por asignatura: en la vista de listado de grupos de estudio, los usuarios tendrán a su disposición un mecanismo de filtro que se utilizará para filtrar los grupos por asignatura.

- Ver notificaciones: los usuarios tendrán en el menú lateral una página dedicada a las notificaciones, mostrándose un icono de alerta cada vez que el usuario tenga notificaciones nuevas.

- Enviar petición de entrada a grupo: los usuarios que no pertenezcan a un grupo público podrán desde la vista de muro de este enviar una petición de entrada.

- Enviar mensaje a muro de grupo: los miembros de un grupo tienen la posibilidad de enviar mensajes en tiempo real desde el muro de este.

- Subir documentos a un repositorio: los usuarios pertenecientes a un grupo tendrán la posibilidad de subir documentos al repositorio.

- Descargar documento del repositorio: los usuarios pertenecientes a un grupo tendrán la posibilidad de descargar documentos al repositorio.

- Ver miembros de un grupo: los usuarios tienen a su disposición una lista de miembros de un grupo, dentro de la vista del grupo.

- Ver log de actividad de grupo: los miembros del grupo podrán acceder desde la pestaña de miembros a un archivo de log que contiene la actividad del grupo.

- Borrar grupo: los dueños de los grupos tienen la posibilidad de borrar el grupo. Lo que realmente se hace es desactivar el grupo, pero no se borra ningún tipo de información referente al mismo.

- Ver actividad reciente de grupos: el usuario tendrá en la página inicial de la aplicación un panel donde ver la actividad reciente de los grupos de estudio.

- Invitar usuario a grupo: los dueños de un grupo pueden acceder a un listado de alumnos que buscan grupos de estudio para una asignatura en concreto, y desde ese listado pueden enviarles una invitación.

- Expulsar usuario de grupo: los administradores serán capaces de eliminar usuarios de un grupo desde la vista de miembros.

- Borrar documento del repositorio: los administradores son capaces de eliminar ficheros desde la vista de repositorio de un grupo.

- Responder petición de grupo: todos los usuarios en la página de notificaciones cuentan con una serie de botones que les permiten aceptar o rechazar los distintos tipos de notificaciones que les lleguen. Estas notificaciones que se podrán admitir o rechazar responden a las notificaciones de petición de ingreso o de invitación a un grupo.

- Abandonar grupo: los usuarios pertenecientes a un grupo son capaces de abandonar el grupo en cualquier momento.

2.4. Mockups y diseño de interfaz

A la hora de elaborar una interfaz o diseños previos al desarrollo de la aplicación, se optó por el uso de Ionic Creator.

Mediante esta herramienta web, fuimos capaces de realizar los primeros pasos en cuanto a diseño de interfaz de nuestra aplicación, y aunque muchos de los mocks han sufrido importantes cambios, sirvieron para realizar una primera aproximación en cuanto a diseño. En cuanto a la paleta de colores, se ha optado por el uso de los colores institucionales de la ULPGC(tonos azules).

A continuación podemos observar algunos mockups representativos de nuestra aplicación.

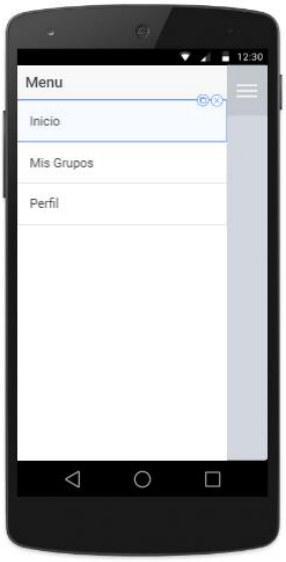

Ilustración 2 - Mockup menú lateral

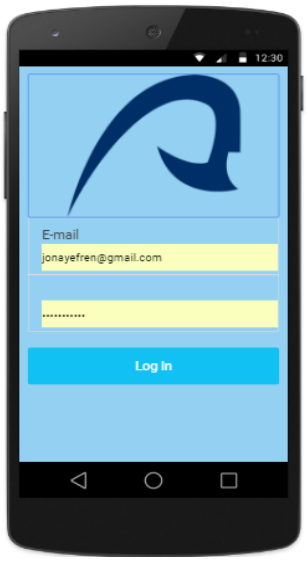

Ilustración 3 - Mockup login

Ilustración 4 - Mockup vista perfil de usuario

Ilustración 5 - Mockup muro de grupo

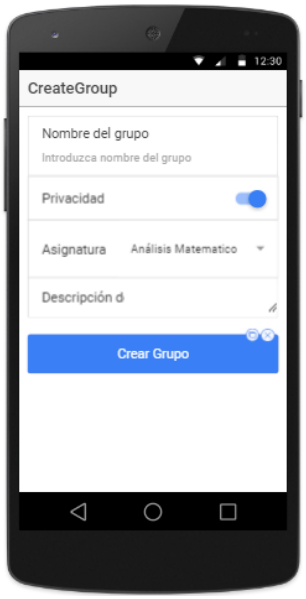

Ilustración 6 - Mockup creación de grupo

2.5. Normativa y legislación

Las aplicaciones móviles pueden recoger gran cantidad de datos de carácter personal. Por este motivo, el grupo wp29 (Grupo de trabajo compuesto por un representante de la autoridad de protección de datos de cada Estado miembro de la UE, el Supervisor Europeo de Protección de Datos y la Comisión Europea) dejan muy claro el marco jurídico aplicable al uso de aplicaciones para dispositivos inteligentes, remarcando que la

normativa sobre protección de datos es totalmente aplicable en las mismas. Este dictamen se puede encontrar en el siguiente enlace:
https://www.agpd.es/portalwebAGPD/revista_prensa/revista_prensa/2013/notas_prensa/common/marzo/130314_NP_Dictamen_aplicaciones.pdf

A continuación se describen las cuestiones legislativas que nuestro proyecto debe cumplir.

- En primer lugar, debemos poseer las licencias de los recursos o tecnologías utilizados en el desarrollo del proyecto. En nuestro caso, al ser el proyecto un TFG y el carácter educativo que tiene, contamos con las licencias gratuitas para aficionados de Google Cloud Platform, lo que nos permite utilizar y tener acceso a los servicios en la nube de Google de Realtime Database y de Google Cloud Storage.

- Al ser los desarrolladores de la aplicación, hemos de responsabilizarnos con el tratamiento de los datos de carácter personal que pudiera tener la aplicación. El proyecto debe cumplir con el Reglamento de Seguridad del Real Decreto 994/1999 de la Ley Orgánica de Protección de Datos (LOPD) y nos obliga a comunicar a la Agencia Española de Protección de datos (AEPD) todos los datos de carácter personal que recabemos, según nuestra actividad y dónde los almacenamos o que tratamiento les damos.

- Solicitar consentimiento con carácter previo a que la aplicación comience a recoger o almacenar información del dispositivo.

- Nuestra aplicación debe respetar la Ley de Propiedad Intelectual «BOE» núm. 97, de 22/04/1996, ya que los usuarios pueden subir cualquier tipo de material en los repositorios. Los usuarios deben aceptar las condiciones de uso, donde aceptan que si infringen la ley de propiedad intelectual se les bloqueará la cuenta de forma permanente.

- Solicitar consentimiento específico para cada uno de los datos personales a los que la aplicación accede.

- Permitir a los usuarios rescindir su consentimiento y desinstalar la aplicación, así como la supresión de datos.

- Facilitar una inteligible y fácilmente accesible política de privacidad, que advierta a los clientes al menos sobre: quiénes son, qué categorías de datos de carácter personal recogen y procesan, por qué deben realizar el procesamiento de datos y para qué se van a utilizar, en caso de que sean cedidos a terceros, una específica descripción acerca de a quién van a ser cedidos y los derechos de los usuarios, en lo referido a la revocación del consentimiento y la supresión de datos.

- Facilitar a los usuarios el ejercicio de sus derechos de acceso, rectificación, oposición y cancelación del tratamiento de datos y avisarles sobre de la existencia de estos mecanismos.

3. Diseño arquitectónico y modelo de la base de datos

3.1. Diseño arquitectónico

Nuestra aplicación desarrollada mediante el *framework* ionic, sigue una arquitectura basada en componentes, heredada debido al desarrollo del código mediante Angular. Esto quiere decir que nuestra aplicación está compuesta por un árbol de componentes que trabajan entre sí para la conclusión de distintos objetivos globales.

Se ha probado que es una arquitectura que proporciona una fácil escalabilidad y mantenibilidad, siendo una de las más usadas en el panorama actual.

Los componentes están pensados para integrar modularidad y resolver pequeños problemas, como por ejemplo un simple botón que desempeña una funcionalidad. Integrando distintos tipos de componentes podemos concebir aplicaciones grandes y complejas. Ionic 3 nos provee de base una cantidad bastante amplia de componentes que están preparados para ser utilizados en pantallas táctiles.

A continuación, describiremos la arquitectura de Angular, que es heredada por Ionic. El siguiente diagrama muestra la arquitectura de Angular.

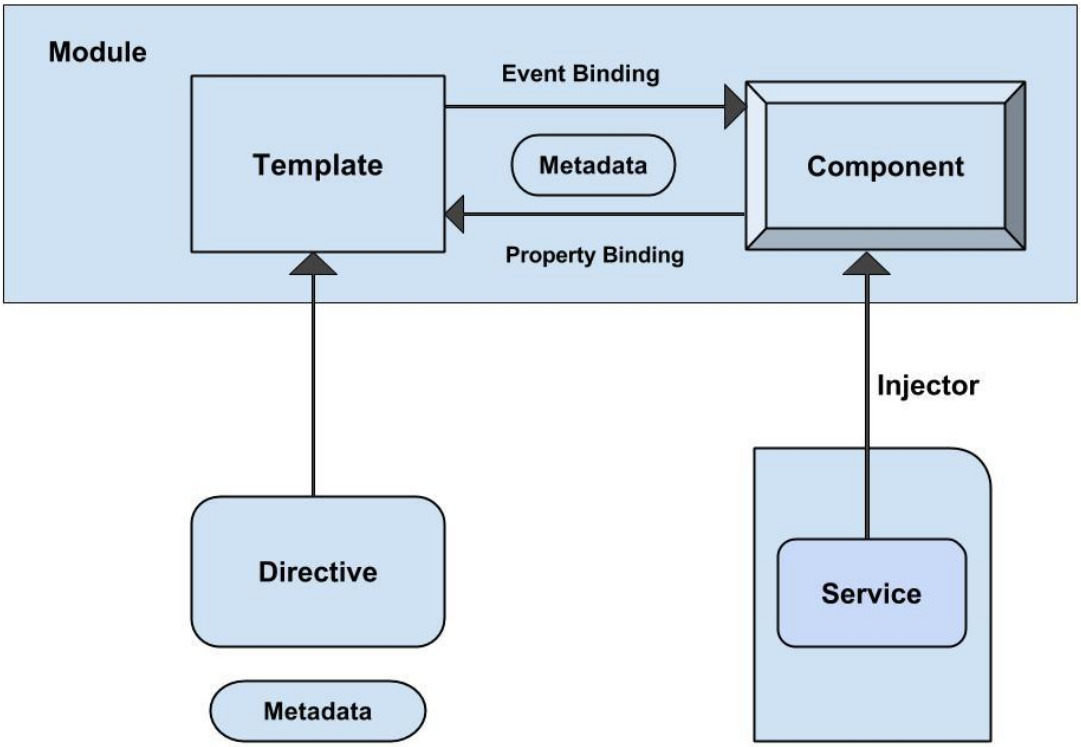

Diagrama 2 – Arquitectura de Angular

La arquitectura de Angular consta de los siguientes módulos o partes:

- Módulos
- Componentes
- Plantillas
- Metadatos
- Enlace de datos (Data binding)
- Servicios
- Directivas
- Inyección de dependencias

A continuación, describiremos cada una de estas partes.

3.2. Módulos

Las aplicaciones de Angular son modulares como ya hemos comentado, gracias a su propio sistema de módulos llamado Angular Modules. Los módulos en Angular son una pieza fundamental en el desarrollo y son agrupaciones de componentes. Nos ayudan a mantener un orden y a encapsular funcionalidades para crear aplicaciones desacopladas con bloques reutilizables.

Toda aplicación de Angular tiene al menos un módulo, el módulo principal o root module.

Los módulos son definidos por el decorador *NgModule* al definirse ciertos metadatos. Los más importantes son los siguientes:

- **declarations**: Las vistas que pertenecen a tu módulo. Hay 3 tipos de clases de tipo vista: componentes, directivas y pipes.
- **exports**: Conjunto de declaraciones que deben ser accesibles para plantillas de componentes de otros módulos.
- **imports**: Otros NgModules, cuyas clases exportadas son requeridas por plantillas de componentes de este módulo.
- **providers**: Los servicios que necesita este módulo, y que estarán disponibles para toda la aplicación.
- **bootstrap**: Define la vista raíz. Utilizado solo por el root module.

3.3. Componentes

Un componente es una clase de controlador de una vista asociada. El componente define propiedades y métodos que están disponibles en su plantilla.

En nuestra aplicación, utilizamos varios componentes ofrecidos por Ionic, tales como:

- *Ionic Cards*
- *Floating Action Buttons*
- *Ionic Lists*
- *Menus*
- *Tabs*

3.4. Plantillas

Las plantillas nos permiten en Angular definir la vista de un componente. Las plantillas de Angular son HTML, pero decorados con distintos componentes o expresiones de Angular que enriquecen el comportamiento de la plantilla.

Algunas de estas expresiones nos permiten trabajar con código TypeScript directamente desde las plantillas, tales como:

- *ngFor : estructura de control de flujo iterativo
- {{object.property}} : forma de acceder a objetos del controlador

3.5. Metadatos

Los metadatos son una forma de procesar las clases en Angular. Angular procesa los distintos tipos de clase mediante los metadatos. Todas las clases para Angular son solo eso, clases, hasta que se le indique si se trata de un componente u otro tipo de clase gracias a los metadatos de Angular.

La forma de añadir metadatos a una clase en Angular es mediante el patrón decorador, en la declaración de la clase.

3.6. Enlace de datos o Data Binding

El enlace de datos o 'Data Binding' es un proceso de coordinación de los procesos de intercambio de información entre las plantillas y los componentes a los que pertenecen. Escribir toda esa lógica a mano es tedioso y propenso a errores. Angular lo resuelve por nosotros gracias al Data Binding.

Angular dispone de cuatro formas de Data Binding:

- Interpolación: (Del componente al DOM) : podemos acceder a valores establecidos en el componente mediante {{variable}}.
- Property Binding (Del componente al DOM) : Se pueden establecer propiedades de distintos componentes en la plantilla mediante [propiedad]=variable".
- Event binding (Del DOM al componente) : Podemos indicar a Angular que cuando se produzca un evento en una parte concreta de nuestra plantilla, se ejecute alguna función en el componente, pudiendo pasar parámetros adicionales. Por ejemplo (click)="ejecutarFuncion($event)" ejecutaría la función ejecutarFuncion del componente y le pasaría como parámetro el evento ocasionado.
- Two-way binding (Desde el DOM al componente y viceversa): En este caso se combinan event binding y property binding, el valor de la propiedad fluye desde el componente al elemento HTML, pero puede ser modificado por el usuario, actualizando el valor de dicha propiedad. Se usa de la siguiente manera: [(ngModel)]="propiedad"

3.7. Servicios

Los servicios son funciones que se encargan de realizar una tarea específica. Los servicios se inyectan utilizando el mecanismo de inyección de dependencia.

Juegan un papel fundamental en las aplicaciones de Angular, ya que mantienen el código organizado y modularizado.

3.8. Directivas

Las directivas de Angular permiten la manipulación y reutilización de código HTML. Cuando Angular renderiza las directivas, transforma el DOM en base a las instrucciones que se encuentren en ellas.

Encontramos los diferentes tipos de directivas:

- Directivas estructurales: Alteran el *layout* añadiendo, eliminando o reemplazando elementos en el DOM. Comienzan por asterisco y son, por ejemplo *ngIf, funcionando como un *if* tradicional, si la condición se cumple, el elemento se inserta en el DOM. En caso contrario, se elimina del mismo.

- Directivas Atributo: Alteran la apariencia o el comportamiento de un elemento del DOM. En las plantillas juegan el papel de atributos HTML normales. Alguna de estas directivas es por ejemplo no-padding para eliminar relleno de un contenedor

3.9. Inyección de dependencias

La inyección de dependencias es un mecanismo para proporcionar nuevas instancias de una clase con todas aquellas dependencias que requiere plenamente formadas.

La mayoría de dependencias suelen ser servicios, y Angular usa la inyección de dependencias para proporcionar nuevos componentes con los servicios que necesitan.

Gracias a TypeScript, cuando un componente requiere de inyección de dependencias, Angular sabe de qué servicios depende con tan solo mirar su constructor.

3.10. Modelo de la base de datos

En el proyecto realizado, no hemos trabajado con bases de datos relacionales. En nuestro proyecto utilizamos Realtime Database de FireBase, producto de Google.

Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Cuando nuestra aplicación es compilada, todos los usuarios comparten una instancia de Realtime Database y reciben actualizaciones automáticamente con los datos más recientes.

Sin embargo, a modo de guía se realizó un diagrama entidad-relación.

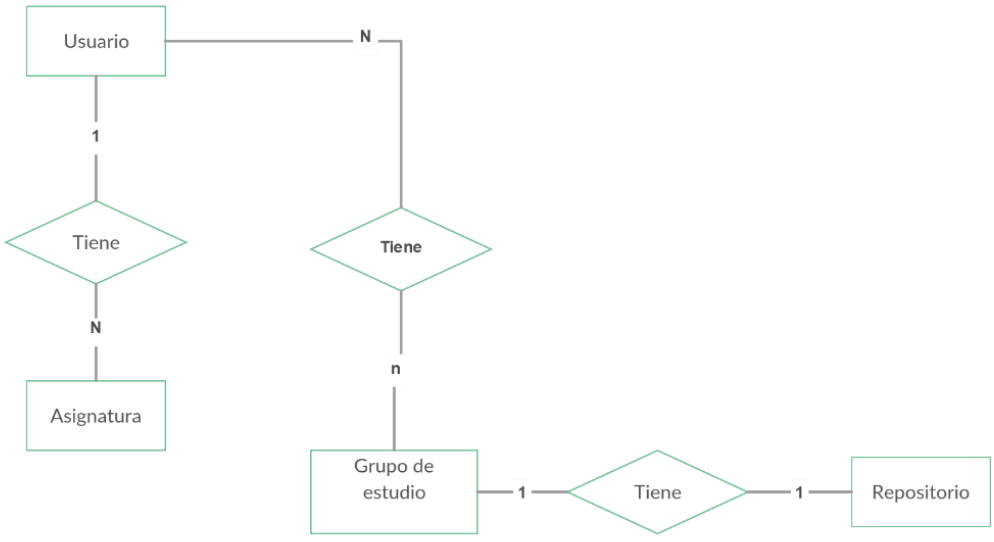

Diagrama 3- Modelo entidad relación

4. Iteraciones

En este apartado repasaremos el alcance de la implementación de nuestro proyecto y enumeraremos las características implementadas en nuestra aplicación.

A continuación, detallaremos las iteraciones del proyecto y que características han sido implementadas en que iteración. También mostraremos algún que otro detalle de implementación de las características más importante.

4.1. Alcance de la implementación

En este proyecto se ha desarrollado un prototipo de aplicación que cumple con los requisitos detallados en la fase de análisis y diseño. Si bien es un prototipo y puede tener más funcionalidades, se trata de un prototipo que integra las funcionalidades básicas totalmente operativo y funcional.

4.2. Iteración 1: Implementación de funcionalidades de grupo y usuario básicas.

En esta iteración nos hemos centrado en la implementación de las funcionalidades de usuario, tales como el inicio de sesión y cambiar una foto de perfil, así como las funcionalidades relacionadas con los grupos de estudio. Las exponemos todas a continuación:

- Iniciar sesión
- Ver perfil de usuario
- Crear grupo de estudio
- Ver listado de grupos
- Ver grupo
- Cerrar sesión
- Cambiar foto de perfil
- Enviar mensaje a muro grupo
- Ver notificaciones
- Filtrar grupos por asignatura

4.2.1. Detalles de implementación

Esta iteración supuso la puesta en marcha del proyecto, y hubo algún que otro contratiempo debido a que no tenía experiencia con esta versión de Angular. Había ciertas partes del código que mantenían suscripciones a objetos de la base de dato abiertas, por lo que al usuario cerrar sesión se producía un error de permisos. Se solventó cerrando todas las subscripciones al cumplir con el objetivo de las mismas. A continuación, se exponen ciertas partes de código de las características implementadas que se piensas relevantes.

```
<ion-list>
  <ion-item-divider no-padding>
    <ion-row>
      <ion-col col-9 color="primaryDark">
        <h2>Nombre</h2>
      </ion-col>
      <ion-col col-3>
        <h2>Asignatura</h2>
      </ion-col>
    </ion-row>
  </ion-item-divider>
  <ng-container *ngFor="let group of groups | async" >
  <ion-item *ngIf="!group.private"   no-padding (click)="goToGroup(group.id)">
    <ion-row>
      <ion-col col-9>
        {{group.name}}
      </ion-col>
      <ion-col col-3>
        {{group.cod}}
      </ion-col>
    </ion-row>
  </ion-item>
 </ng-container>
 </ion-list>
</ion-content>
```

Ilustración 7 - Código referente a la lista de grupos de estudio

En la ilustración 7, podemos ver la estructura utilizada para listar todos los grupos de estudio públicos. Se utiliza la estructura de control *for* para recorrer el listado de grupos e irlos mostrando de forma asíncrona. Al clickar en algún elemento de la lista de grupos, accederemos a la vista de ese grupo concreto.

En la ilustración 8 podemos ver una de las vistas añadidas a esta iteración, correspondiente al código de la ilustración 7.

Ilustración 8 - Vista de lista de grupos

A continuación veremos un segmento de las estructura JSON con la que se ha trabajado en esta iteración.

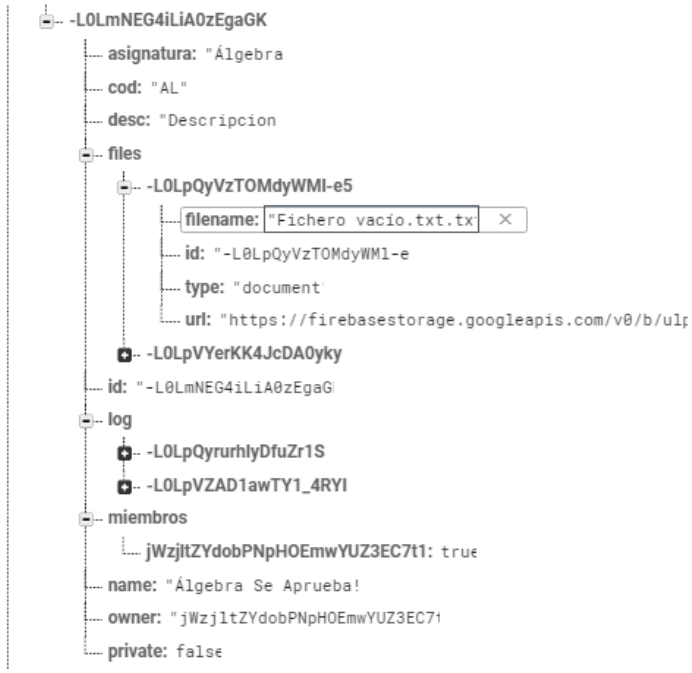

Ilustración 9 - Extracto JSON perteneciente a grupo de estudio

En la ilustración 9 podemos observar la estructura que compone un grupo de estudio. El grupo de estudio aparte de estar formado por campos simples como descripción, asignatura o nombre tiene asignado un campo "owner" que hace referencia al id del usuario que creó y por ende es el dueño del grupo. Así mismo, el grupo contiene un conjunto de elementos llamado "files", que se utilizan para implementar cada repositorio de grupo. Estos objetos contienen una url de descarga, un nombre de fichero, el tipo de fichero que es (pueden existir directorios) y su identificador único. Los grupos de estudio cuentan con una lista de ids de usuario referente a los miembros del grupo, así como un *booleano* que nos indica si el grupo es privado o público.

4.3. Iteración 2: Implementación de repositorio y funcionalidades de Grupo Secundarias.

En esta iteración se han implementado las funcionalidades referentes a la participación en el programa Apadrina y se han seguido implementando funcionalidades referentes a los grupos de estudio. Son las siguientes:

- Inscribirse en programa apadrina
- Ver repositorio de grupo
- Ver padrinos disponibles
- Abandonar programa Apadrina
- Responder petición de grupo
- Enviar petición de entrada a grupo
- Borrar documento del repositorio
- Subir documentos a un repositorio
- Descargar documento del repositorio
- Ver miembros de grupo
- Invitar usuario a grupo
- Ver notificaciones

4.3.1. Detalles de implementación

En esta iteración, se optó por implementar el módulo del programa Apadrina de la siguiente forma: cuando un estudiante se inscribe en el programa, se crea un grupo privado propio, facilitando la implementación de la lógica de este módulo. Se distingue de los grupos normales en que su asignatura es la de Mentor, para facilitar la separación entre los distintos tipos de grupo. Cuando el usuario solicita ver una lista de padrinos, lo que en realidad ve es una lista de grupos pertenecientes a padrinos que están participando en el programa, pudiendo enviar una solicitud si la descripción del padrino les convence.

```
uploadFile(event: any) {
  let loader = this.loadingCtrl.create({
    content: "Subiendo archivo"
  });
  loader.present();
  var file: File = event.target.files[0];
  const medaTada = { contentType: (file as any).type };
  const storeRef: firebase.storage.Reference = firebase
    .storage()
    .ref("/groups/" + this.gid + "/" + (file as any).name);
  storeRef.put(file, medaTada).then(snapshot => {
    this.db.list("/groups/" + this.gid + this.actualDir).push({
      filename: (file as any).name,
      url: snapshot.downloadURL,
      type: "document"
    }).then((item) => {
      this.db.object("/groups/" + this.gid + this.actualDir+'/'+item.key).update({id : item.key});
    });

    this.auth.authState.subscribe((usersna) => {
      if (usersna) {
        var file: File = event.target.files[0];
        this.db.object("/users/" + usersna.uid).valueChanges().subscribe((user) => {
          this.db.list("/groups/" + this.gid + '/log').push({
            message: "El usuario "+(user as any).name+" ha subido "+(file as any).name,
            date: Date.now(),
            reversedate: 0-Date.now(),
            type: "upload",
            color:"green"
          });
        })
      }
    })
    loader.dismiss();
  });
}
```

Ilustración 10 - Código referente a la subida de ficheros

En esta iteración queremos destacar el código desarrollado para implementar la subida de ficheros, ya que nunca habíamos tenido experiencia trabajando con este tipo de tecnologías.

Para subir un fichero, el usuario debe hacer click sobre un botón que abrirá el navegador de ficheros del dispositivo. Una vez el usuario haya seleccionado un documento, se ejecutará la función uploadfile, recibiendo como parámetro el evento generado.

A través del evento, somos capaces de obtener el documento seleccionado y procederíamos a subirlo al sistema de almacenamiento. Para ello, creamos una referencia en Firebase Cloud Storage, para luego proceder a subir el archivo mediante esa referencia.

Una vez completada la subida del documento, se procede a almacenar en el grupo de estudio del repositorio el fichero subido.

Finalmente se genera una entrada en el log de actividad del grupo.

A continuación veremos un segmento de las estructura JSON con la que se ha trabajado en esta iteración, concretamente con la de notificaciones de un usuario concreto

Ilustración 11 - Estructura JSON referente a las notificaciones de usuario

En la ilustración 11 nos encontramos con la estructura de las notificaciones de usuario. Cada usuario tiene asociado a su perfil un listado de notificaciones. Las notificaciones se generan al recibir una petición de entrada a grupo, al invitar a un usuario especifico al grupo o al ser admitido o rechazado en un grupo. Es por ello por lo que se distinguen tres valores para el campo *type* : req, inv o msg dependiendo del carácter de la notificación. El id del usuario que envía la notificación se usa para añadirlo al nuevo grupo en caso de aceptación, mientras que el campo de *receiver* se utiliza para eliminar la notificación del usuario que la recibe una vez contestada

4.4. Iteración 3 : Implementación de funcionalidades monitorización y administración de grupo

En esta iteración se mejoraron las vistas y se utilizaron iconos de librerías externas. Se implementaron funcionalidades requeridas por el tutor, tales como un log de actividad para cada grupo y ciertas funcionalidades de gestión de grupo que solo puede realizar el administrador, como borrar un grupo o expulsar a un usuario. Son las siguientes:

- Borrar grupo
- Ver log de actividad de grupo
- Expulsar usuario de grupo

- Ver actividad reciente de grupo

4.4.1. Detalles de implementación

En esta iteración, el requisito añadido del log supuso que para varios aspectos relacionados con el almacenamiento en la base de datos. Para facilitar el registro de actividades en el log, se guardaba con cada entrada una fecha con símbolo negativo, lo que facilitaba el ordenamiento de actividades. A continuación expondremos partes del código consideradas relevantes.

```
removeUser(id:any) {
  this.db.object('/users/'+id+'/groups').update({[this.gid] : null });
  this.db.object('/groups/'+this.gid+'/miembros').update({[id]:null});
  let logsub = this.db.object("/users/" + id).valueChanges().subscribe((user) => {
    this.db.list("/groups/" + this.gid + '/log').push({
      message: "El usuario "+(user as any).name+" ha sido expulsado",
      date: Date.now(),
      reversedate: 0-Date.now(),
      type: "times",
      color:"danger"
     });
     logsub.unsubscribe();
  })
```

Ilustración 12 - Código referente a la expulsión de usuario de un grupo

En la ilustración 12 vemos como se ha implementado la función utilizada para expulsar a un usuario de un grupo, pudiendo llevarse a cabo solo por el administrador del grupo.

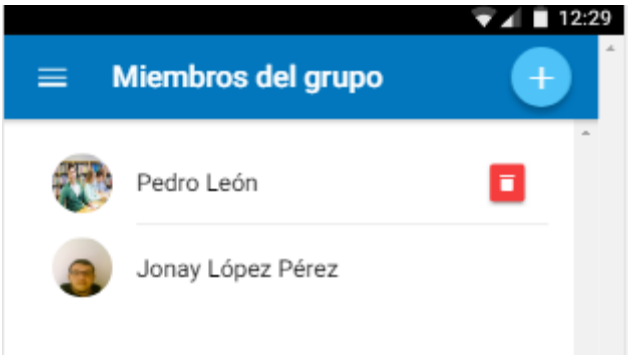

Ilustración 13 - Vista miembros del grupo

En la ilustración 13 vemos como al lado de un miembro de grupo, si somos el administrador, nos aparecerá un botón que se utiliza para expulsar. Una vez pulsado el botón, se llamará a la función removeUser(), que se encargará de eliminar el grupo del

listado de grupos del usuario, así como de eliminar al usuario del listado de miembros del grupo.

Ilustración 14 - Estructura JSON de log de actividad de grupo

En ilustración 14 vemos de que atributos se componen los objetos de tipo log. Constan de un color y de un tipo, campos utilizados para construir los iconos que aparecen en el log de actividad de grupo. El campo reversedate es el utilizado para mostrar las entradas de log de más recientes a más antiguas.

4.5. Pruebas del prototipo

Además de la prueba de funcionalidades una vez el prototipo fue acabado, en cada una de las iteraciones se ha ido comprobando que todas las características funcionan de forma correcta y adecuada.

La aplicación ha sido probada en el navegador Google Chrome, sin embargo, ciertas funcionalidades como la descarga de documentos han sido implementadas tanto como para navegador web como de forma nativa para Android. Es por eso por lo que también se han realizado pruebas de todas las funcionalidades en dos dispositivos Android de distinto tamaño de pantalla, comprobando que la aplicación responde de forma correcta ante distintas pantallas de smartphones.

Las pruebas se han realizado en un Sony Xperia Z y en un Samsung Galaxy S5.

5. Tecnologías

En este apartado de la memoria describiremos las diferentes herramientas y tecnologías utilizadas a lo largo del desarrollo del proyecto. Trataremos el *framework* Ionic y las tecnologías que a su vez lo componen, así como de FireBase Realtime Database y Cloud Storage.

5.1. Ionic 3

Ilustración 15 - Logo Ionic 3

Ionic3 es un *framework* gratuito y de código abierto que se utiliza para el desarrollo de aplicaciones multiplataforma híbridas, apoyándose en HTML5, CSS y Cordova como base. Es uno de los frameworks del momento ya que utiliza AngularJS para gestionar las aplicaciones, asegurando que éstas sean rápidas y fácilmente escalables.

Podemos encontrar documentación sobre esta tecnología e información relacionada en el siguiente enlace: https://ionicframework.com

5.2. Angular 4

Ilustración 16 - Logo Angular 4

Angular 4 es un framework para el desarrollo de aplicaciones basadas en tecnologías web de TypeScript de código abierto, mantenido por Google. Se utiliza para concebir y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador, en un esfuerzo para hacer que el desarrollo y las pruebas del producto sean más fáciles.

Podemos encontrar documentación sobre esta tecnología e información relacionada en el siguiente enlace: https://angular.io/

5.3. HTML5 y CSS3

Ilustración 17 - Logo HTML5

HTML5 es la quinta versión importante del lenguaje básico de la World Wide Web HTML, lenguaje de marcas utilizado para la elaboración de páginas web. Es un estándar que sirve de referencia en la elaboración de páginas web.

Ilustración 18 - Logo CSS3

CSS3 es la última versión del lenguaje de Hojas de Estilo en Cascada (Cascading Style Sheets). CSS es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcas.

5.4. TypeScript

Ilustración 19 - Logo TypeScript

TypeScript (https://www.typescriptlang.org/) es un lenguaje de programación de código abierto desarrollado por Microsoft. Es un superconjunto de JavaScript, que añade tipado de datos y objetos basados en clases.

5.5. Cordova

Ilustración 20 - Logo Cordova

Apache Cordova es un entorno de desarrollo de aplicaciones móviles de código abierto, originalmente creado por Nitobi. Cordova permite construir aplicaciones móviles utilizando CSS3, HTML5 y JavaScript en vez de utilizar APIS nativas como las de Android o iOS.

5.6. Angularfire2

Ilustración 21 - Logo AngularFire

Angularfire2 es simplemente una librería para Angular que nos facilita las transacciones con Firebase RealTime Database de Google.

Una de las tecnologías utilizadas es FireBase como base de datos, ya descrita en el apartado Modelo de la base de datos.

5.7. Firebase Cloud Storage

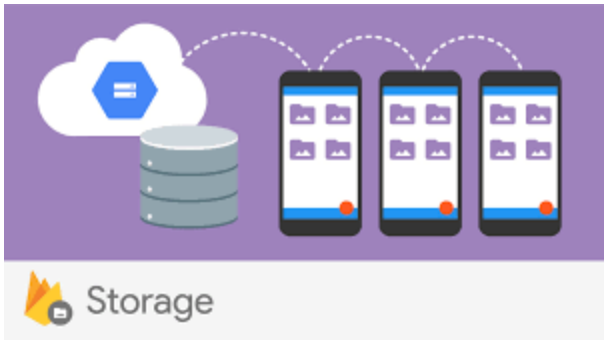

Ilustración 22 - Cloud Storage Logo

Cloud Storage se creó para programadores de aplicaciones que necesitan almacenar y publicar contenido generado por usuarios.

Cloud Storage para Firebase es un servicio de almacenamiento de objetos potente, simple y rentable construido por Google. Los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y descarga de archivos para tus aplicaciones de Firebase, sin importar la calidad de la red.

5.8. FontAwesome

Ilustración 23 - Logo FontAwesome

Font Awesome es un framework de fuentes e iconos basados en CSS y LESS. Proporciona iconos escalables que pueden ser customizados de manera instantánea mediante etiquetas CSS.

6. Acceso al código y despliegue

El código del proyecto es accesible mediante el siguiente repositorio público en BitBucket: https://bitbucket.org/jonayLo/group-finder .

7. Conclusiones

Una vez finalizado el proyecto, y analizando los resultados obtenidos, se ha llegado a la conclusión de que el trabajo dedicado al desarrollo del proyecto ha servido para alcanzar los objetivos marcados y el prototipo realizado cumple con los requisitos que hemos descrito en la fase de análisis.

Como posibles mejoras en la aplicación, implementar un sistema de búsqueda de grupo más exhaustivo supondría un gran avance en la aplicación. Este sistema de búsqueda ayudaría a los usuarios a encontrar de forma más rápida que cumplan con las características que buscan, tales como poder buscar grupos mediante la descripción del desempeño de los grupos.

También se ha pensado en que sería una funcionalidad bastante beneficiosa para los usuarios la de implementar un sistema mediante el que los grupos de estudio pudiesen reservar aulas libres de la facultad, pudiendo acceder a un horario en el que se vería cada aula y las franjas horarias de la misma.

Como conclusión final, el desarrollo de este proyecto me ha servido para trabajar con tecnologías modernas y en auge, con las que no estaba del todo familiarizado y para darme de que la programación para dispositivos móviles es un entorno en el que me siento cómodo y en el que seguramente siga trabajando.

Sin embargo, llegar hasta el final del proyecto de forma exitosa no ha sido fácil, pues las tecnologías con las que se ha trabajado son tecnologías relativamente nuevas, no todas con abundando documentación. En particular lo que más me ha costado ha sido la adecuación a trabajar con una base de datos asíncrona. Desde un punto de vista personal, cambia totalmente la forma de trabajar en muchos casos.

Otro aspecto que ha sido problemático pero que nos ha enseñado mucho ha sido el trabajar con una base de datos no relacional y redundante, como hemos dicho antes es una forma totalmente distinta de trabajar y que se me resistió en el comienzo del proyecto.

8. Fuentes de información

- Arquitectura Angular
  http://www.w3ii.com/angular2/angular2_architecture.html
  Última vez accedido en diciembre de 2017

- Curso de Ionic y Angular por Steve Micholatti
  https://www.pluralsight.com/courses/building-mobile-apps-ionic-framework-angularjs
  Última vez accedido en septiembre de 2017

- Documentación AngularFire
  https://github.com/angular/angularfire2
  Última vez accedido en noviembre de 2017

- Documentación FireBase
  https://firebase.google.com/docs/
  Última vez accedido en diciembre de 2017

- Documentación Ionic3
  https://ionicframework.com/docs/
  Última vez accedido en diciembre de 2017

- Guía de protección de datos para desarrolladores de aplicaciones móviles
  https://ayudaleyprotecciondatos.es/2016/06/06/normativa-lopd-aplicaciones-moviles/
  Última vez accedido en diciembre de 2017

- Introducción a Angular
  https://blog.enriqueoriol.com/2017/03/introduccion-angular-modulo-y-componente.html
  Última vez accedido en diciembre de 2017

- ¿Por qué aprendizaje cooperativo?
  https://www.upc.edu/rima/es/grupos/giac-grupo-de-interes-en-aprendizaje-cooperativo/bfpor-que-aprendizaje-cooperativo
  Última vez accedido en diciembre de 2017

Anexos

i. Manual de usuario

Al iniciar la aplicación, se mostrará la página de inicio que muestra dos paneles. El primero nos invita a crear o encontrar grupos de estudios y el segundo es un panel que en otras ocasiones puede cambiar para promocionar eventos de la escuela o noticias importantes. En este caso se ha utilizado para promocionar el programa Apadrina.

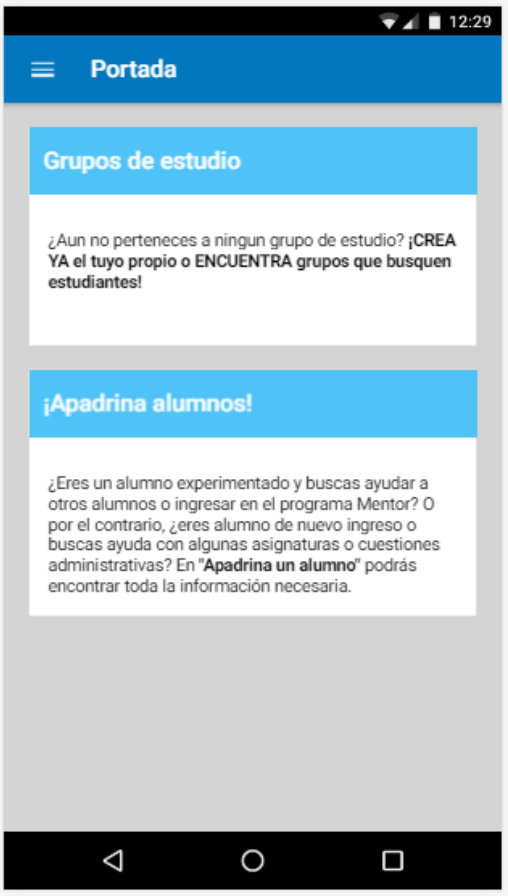

Ilustración 24 - Vista página principal sin logear

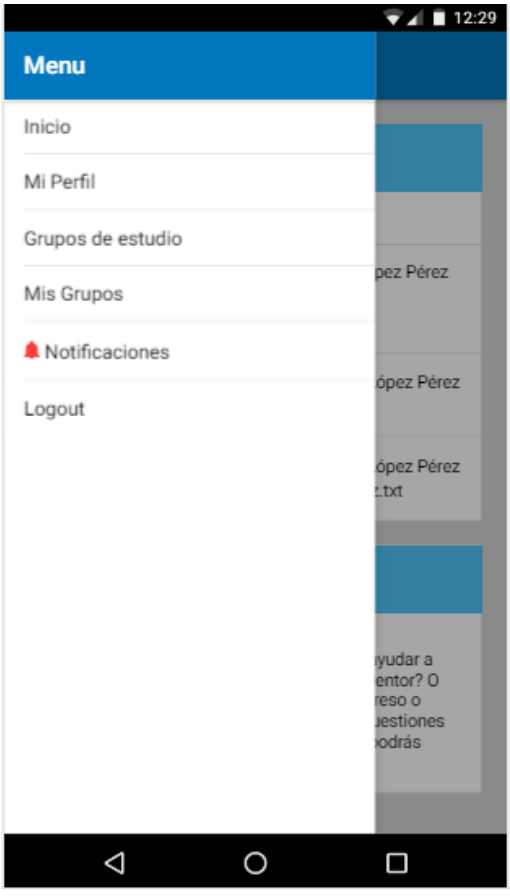

Ilustración 24.1 -Menú lateral de la aplicación

Fijándonos en la ilustración 24 podemos observar ambos paneles. Al hacer click en panel de grupos de estudio se nos redirigirá a la página de listado de grupos públicos, mientras que al ahcer click en el panel de Apadrina se nos requerirá iniciar sesión.

En la ilustración 25 vemos como sería la página principal si estuviéramos logeados y perteneciéramos a algún grupo, ya que si perteneces a algún grupo el panel que te invita a buscar grupos de estudio o a crear el tuyo se convierte en un panel que contiene la actividad reciente de grupos de estudio. Al hacer click en el panel de actividad reciente accederíamos a la vista de ese grupo

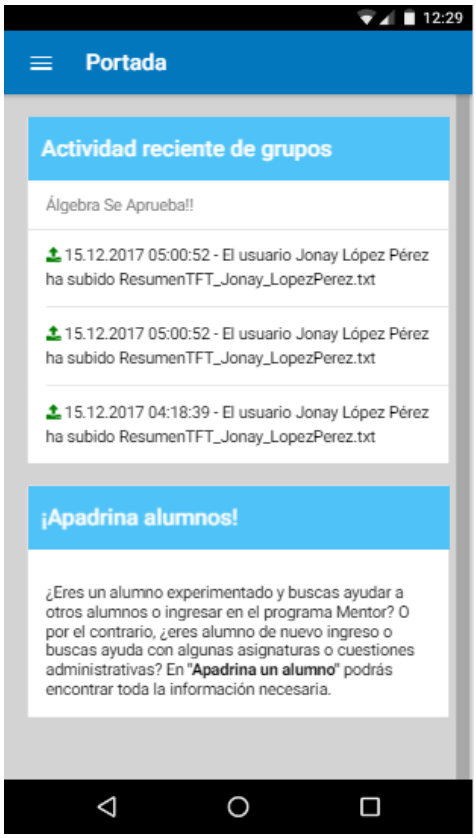

Ilustración 22 - Vista de página principal logeado

.

A continuación describiremos la página de login.

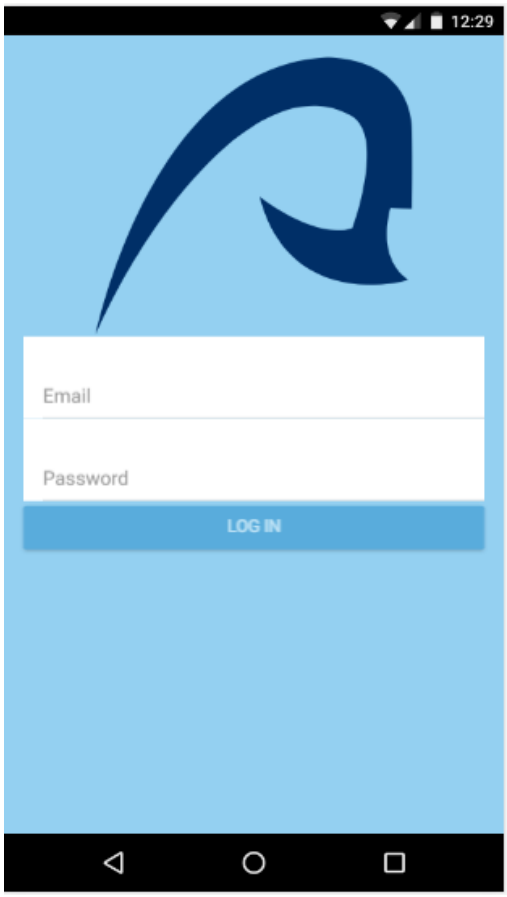

Ilustración 23 - Vista de login

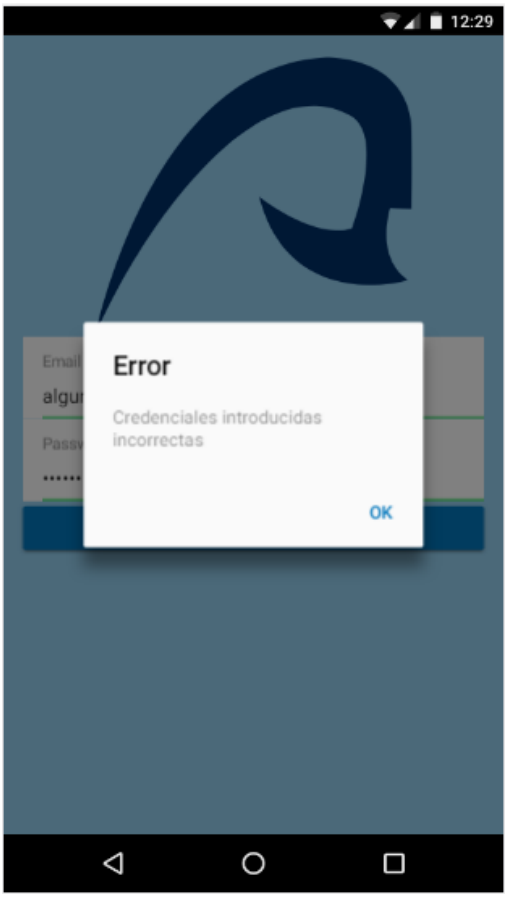

Ilustración 24 - Error al logearse

En la ilustración 26 nos encontramos con la vista de login de la aplicación. Cuenta con dos campos de entrada que cuentan con dos validadores, el campo de email debe seguir una estructura de email válida. Ambos campos son necesarios, de lo contrario el botón de Log In no se habilitará hasta que lo sean.

En la ilustración 27 vemos lo que sucedería en caso de que el usuario introdujera datos de inicio de sesión equivocados

A continuación describiremos la vista de perfil de usuario. Esta vista contiene la información de cada usuario, mostrando su nombre y E-mail, asi como sus asignaturas.

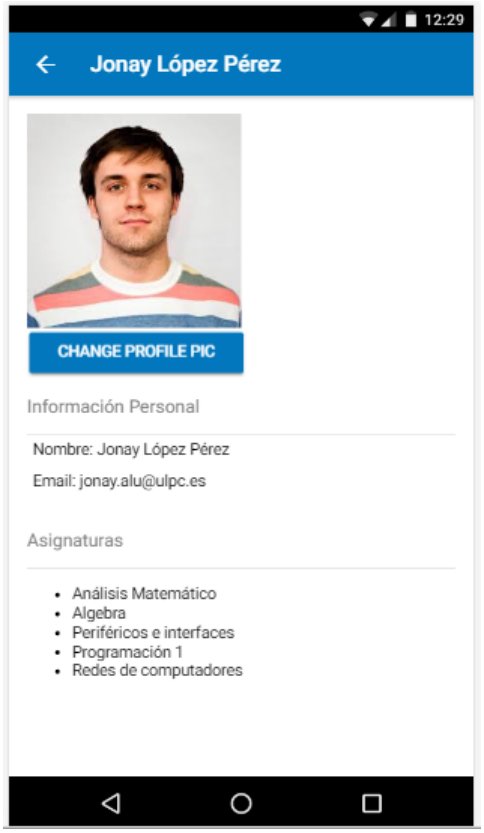

Ilustración 28 - Vista de perfil de usuario

Como podemos ver en la ilustración 28, los usuarios contarán en su propio perfil con un botón que les permite cambiar su foto de usuario.

La siguiente página a mostrar es la de grupos de estudio. En esta página se recogen los distintos grupos de estudio públicos, mostrando las asignaturas en las que está matriculado

Ilustración 29 - Vista de grupos de estudio

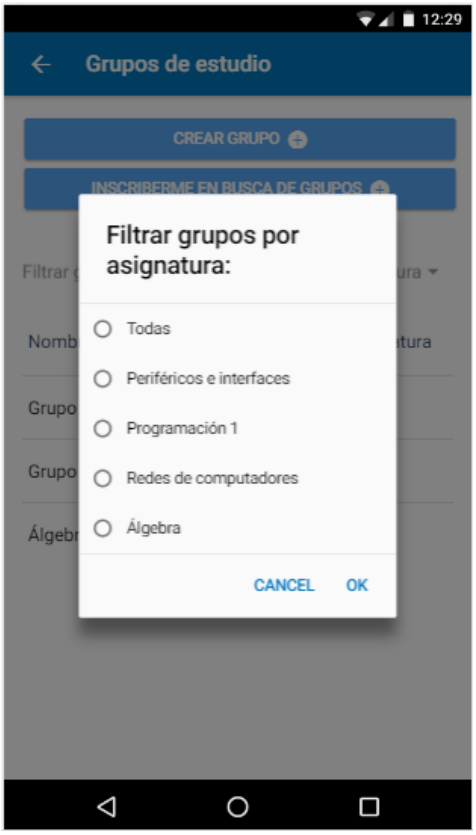

Ilustración 30 - Vista de filtrado de grupos

En la página de grupos de estudio mostrada en la ilustración 29 nos encontramos principalmente con dos botones, uno nos sirve para redirigirnos a la página de creación de grupos y el otro para buscar grupo de forma pasiva. Los grupos de estudio se muestran en forma de lista, acompañados de su asignatura. Haciendo click en cada elemento de la lista accederemos a la vista del grupo seleccionado. En la ilustración 30 observamos el desplegada que se acciona al hacer click sobre el filtro de grupos por asignatura.

A la hora de crear grupos, dispondremos de un formulario compuesto por varios campos de entrada. Dos campos de entrada de texto para definir la descripción y el nombre de grupo, un desplegable para establecer la asignatura del grupo y un input de tipo *toggle* que nos permite establecer la privacidad del grupo. Todos estos campos son requeridos para poder enviar el formulario. En ese entonces se podrá hacer click sobre el botón de crear grupo. Podemos ver la vista de creación de grupo en la ilustración 31.

Ilustración 31 - Vista de creación de grupo

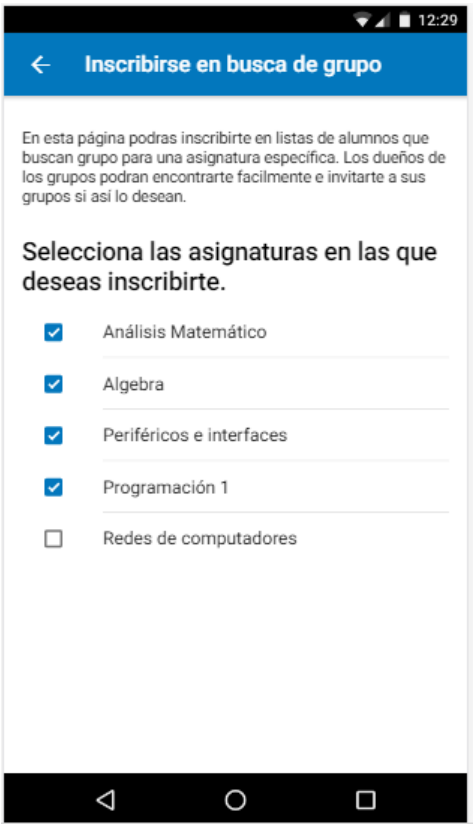

Ilustración 32 - Vista de búsqueda de grupos

En la ilustración 32 vemos la vista relacionada con la búsqueda de grupos de forma pasiva. Cuenta con una breve descripción de la funcionalidad de la página, y se le presentan al usuario las asignaturas en las que está matriculado. El usuario podrá inscribirse o darse de baja en la búsqueda de cada asignatura.

A continuación describiremos las páginas de vista de grupo. La vista de un grupo se divide en tres pestañas. La pestaña del muro del grupo, donde los usuarios podrán interactuar entre si mediante un chat, la pestaña de miembros en la que los usuarios pueden acceder al perfil de los otros miembros del grupo, acceder al log de actividad del grupo o abandonar el mismo. Los administradores en el apartado de miembros podrán invitar nuevos usuarios, expulsar a miembros del grupo o incluso borrar el mismo. En la pestaña del repositorio, los usuarios podrán acceder a los documentos almacenados, así como subir nuevos documentos o crear carpetas. Podemos ver dicha distribución en las ilustraciones 33, 34 y 35.

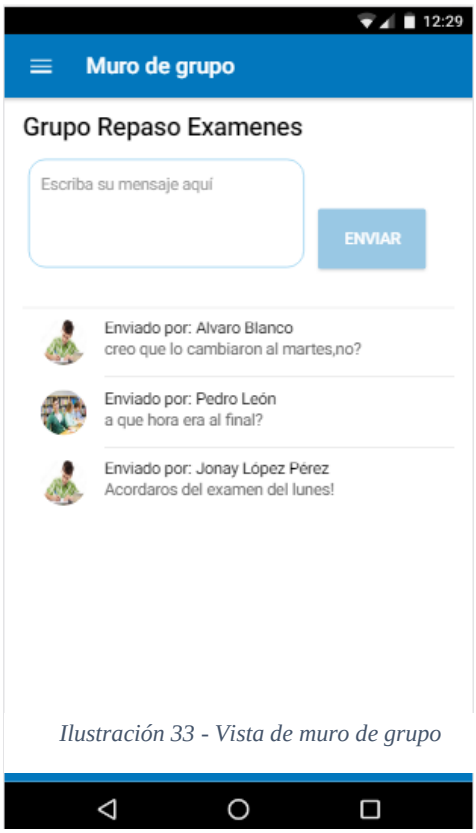

Ilustración 33 - Vista de muro de grupo

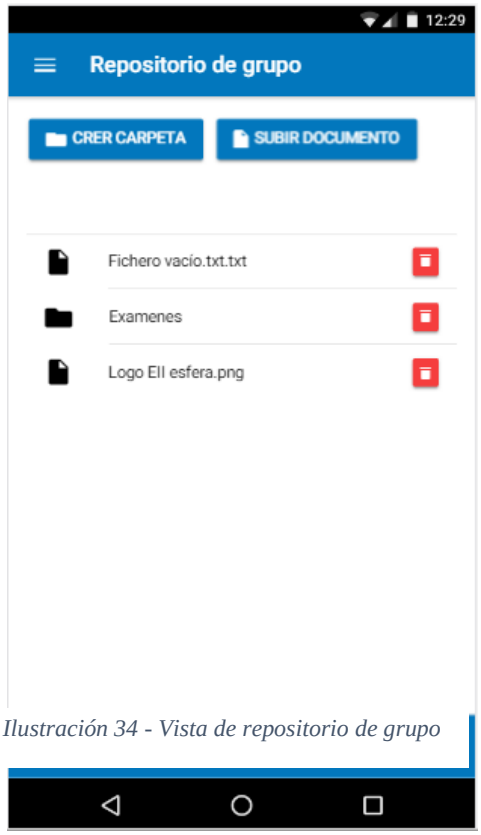

Ilustración 34 - Vista de repositorio de grupo

Ilustración 35 - Vista de miembros de grupo

A continuación describiremos la vista de “Mis grupos”. En esta vista se presentan los grupos a los que el usuario pertenece, pudiendo acceder a los mismos desde esta vista. Los grupos se muestran mediante las cartas de Ionic(como podemos ver en la ilustración 36), donde se especifica la descripción del grupo, su nombre y la asignatura con la que están relacionados.

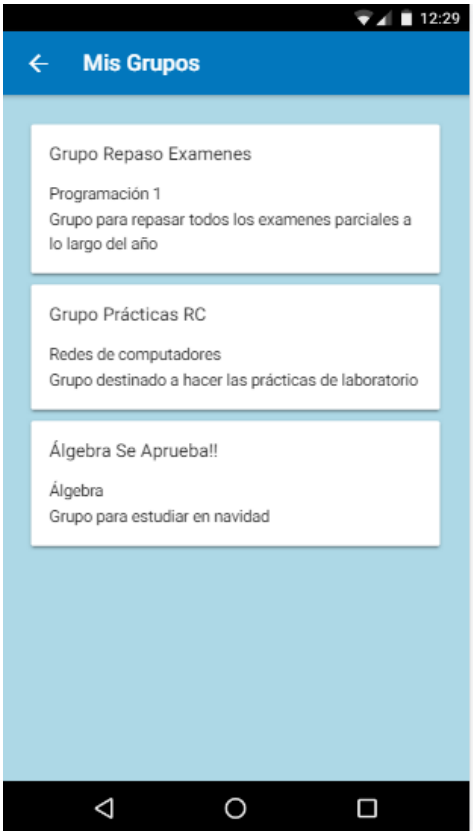

Ilustración 36 - Vista de "mis grupos"

Para terminar con el capítulo de manual de usuario, presentamos las vistas referentes a las notificaciones y al log de actividad de un grupo.

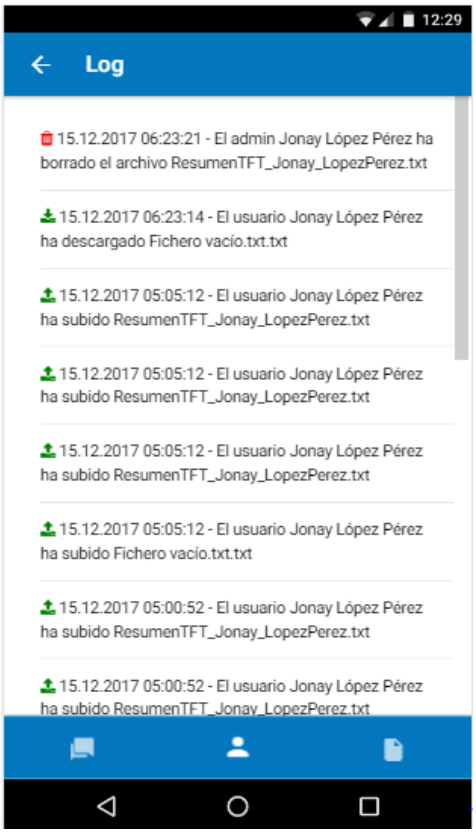

Ilustración 37 - Vista de log de actividad de grupo

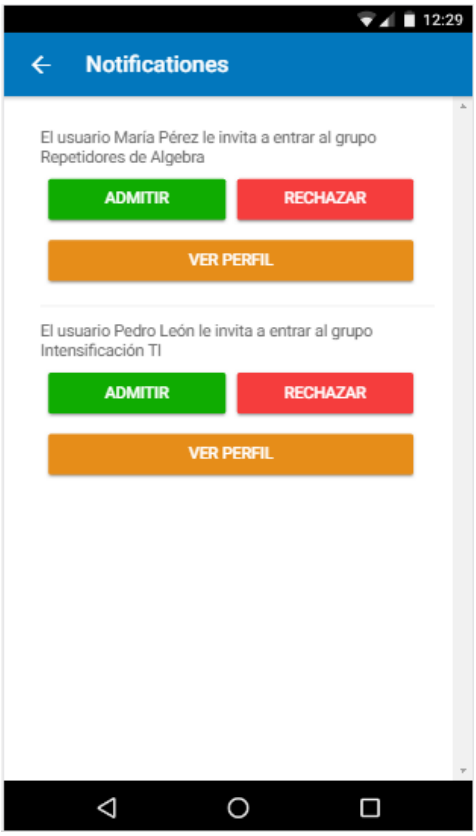

Ilustración 38 - Vista de notificaciones de usuario

En la ilustración 38 vemos la página enfocada a las notificaciones de usuario. Estas notificaciones son o bien de petición de entrada a un grupo en el que el usuario es administrador, o invitaciones a entrar a un grupo de estudio. Se disponen de tres botones, uno para aceptar la petición o invitación, otro para rechazarlas y un último botón que nos sirve para visualizar el perfil de la persona que ha generado la notificación.