Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Una app multiplataforma para medir las emociones en clase

Trabajo Fin de Grado
Grado en Ingeniería Informática

Autor: [Chen Peng]
Tutor: [Vicente Javier Julián Inglada
Elena Val
Juan Miguel Alberola]
[2016-2017]

Resumen

El estado anímico de los estudiantes tiene una influencia directa en el proceso de enseñanza-aprendizaje. En concreto, un estado positivo permite mejorar aspectos como la motivación y los logros conseguidos. A pesar de que el estado anímico representa un estado a largo plazo, este puede ser modificado a través de las emociones que tienen los estudiantes en función de las clases.

El objetivo del Proyecto será desarrollar una aplicación multiplataforma para medir las emociones en clase, orientada al estudio del estado de ánimo de los alumnos en determinados momentos de una clase. La aplicación permitirá al profesor crear encuestas o grupos de encuestas y a los que los alumnos suscribirse a la encuesta correspondiente. A través de las encuestas, la aplicación capturará los datos de los alumnos que serán almacenados para ser analizados. El profesor podrá accede al resultado del análisis asociado a su encuesta creada.

Palabras clave: aplicación Web, emoción, clase, multiplataforma, estudio.

Abstract

The animic state of the students have a direct influence in the process of education-learning. In concrete, a positive attitude allows you to improve aspects like motivation and completed achievements. Despite that the animic state represents a state in a larger number, this can be modified through emotions that the students have in respect to the classes.

The aim of the project is to develop a multi-platform application to measure emotions in class, oriented to the study of the mood of the students of any academic level. The application will have a system that will allow the management of groups of surveys and the result that would be obtained through the surveys. The application will be interacted through a database where will be stored the information and result of surveys.

Keywords: Web Application, emotion, class, multi-platform, study.

Tabla de contenidos

1. Introducción

1.1 Contexto y Motivación

Hoy en día, con la evolución tecnológica a la que estamos expuestos somos más propensos a tener una gran cantidad de herramientas que nos dan la posibilidad de orientar nuestras actividades diarias buscando la máxima eficiencia. Por una parte, Internet es una de las herramientas que impactaron la vida humana, hoy en día, con más de tres mil millones y medio de personas conectadas a la Red. Por otra parte, los dispositivos móviles junto con sus aplicaciones móviles son otras de estas herramientas gracias a sus movilidades que nos permiten una conexión con el tiempo real en cualquier momento, acceder al contenido en cualquier lugar, interactuar y conocer con muchas personas de todas las partes del mundo y acorta la distancia a través de la comunicación.

La evolución de las aplicaciones móviles siempre está acompañada con la evolución de los dispositivos móviles. Con la aparición del famoso "featurephones" (Como la imagen 1) en los años 90, aparecieron las primeras aplicaciones móviles que eran como la agenda, arcade games y los editores de ringtone…etc. Cumplían funciones muy elementales y su diseño era bastante simple. A partir de ese momento, la evolución de las aplicaciones se dio rápidamente gracias a las innovaciones en tecnología WAP (Wireless Application Protocol) y la transmisión de data EDGE (Enhanced Data Rates for Global Evolution), esto vino acompañado de un desarrollo muy fuerte de los celulares.

La gran evolución y el crecimiento de las aplicaciones móviles empieza en el año 2007, Apple lanzó su primera versión de lo que sería la revolución en dispositivos móviles: el *iPhone*. A este ecosistema se juntaba el gigante en crecimiento, Google, lanzó también en 2007 Android, un sistema operativo específico para móviles. Su primer dispositivo se lanzó un año después ("HTC Dream", lanzó en el octubre de 2008). A partir de ese momento apareció un tipo de dispositivo denominado "Smartphone" que provocó el boom de las aplicaciones, juegos, noticias, diseño, arte, fotografía, medicina…etc.

Desde el momento del lanzamiento de "Smartphone" juntos con los dos sistemas operativos para el dispositivo móvil hasta la actualidad, tanto el IOS y el Android hubo un impresionante crecimiento y una rapidez de ocupación en el mercado mundial del sistema operativo para los dispositivos móviles. Hasta el año pasado según la investigación de la empresa consultora Gartner, IOS y Android se concentran un abrumador 99,1 % del mercado, dentro de ese dato, 86,2% para Android y 12,9 % para IOS. Lo podemos ver en más detalle en la siguiente tabla:

Operating System	2Q16 Units	2Q16 Market Share (%)	2Q15 Units	2Q15 Market Share (%)
Android	296,912.8	86.2	271,647.0	82.2
iOS	44,395.0	12.9	48,085.5	14.6
Windows	1,971.0	0.6	8,198.2	2.5
Blackberry	400.4	0.1	1,153.2	0.3
Others	680.6	0.2	1,229.0	0.4
Total	**344,359.7**	**100.0**	**330,312.9**	**100.0**

Tabla 1: tabla de estudio del porcentaje de ocupación de OS (Gartner).

Gracias a las plataformas que están ofreciendo la empresa Google y Apple, hoy en día las aplicaciones se rodean y se basan en estas dos plataformas, junto con las plataformas web constituyen las tres principales herramientas del ser humano para obtener las informaciones.

Dado este escenario, nos encontramos con un ambiente diverso de sistemas operativos, frameworks y lenguajes de programación que pareciera tener un crecimiento exponencial. Y es que, bajo la creciente demanda de aplicaciones móviles y *web*, el desarrollador actual debe convertirse en un experto en todos los frentes anteriormente nombrados y, además, el desarrollo de una aplicación nativa tanto que sea iOS+Swift o Android+Java puede tardar unos dos o tres meses, ya que tener expertos de tecnologías nativas puede ser posible para empresas grandes, pero para la mayoría de Startups esta opción no es viable. Sin embargo, la alta demanda exige cortos *time-to-market* y es ahí donde un ecosistema con baja mano de obra y alta demanda de tecnología comienza a colapsar. ¿Cómo desarrollar hasta cuatro aplicaciones en un corto plazo, con recursos humanos y económicos limitados? Hoy en día una de las formas para solucionar este problema es implementar una aplicación web, al igual que aplicaciones móviles para Android, iOS y Windows Phone como mínimo, y reduce el tiempo de la acogida en el público y sea un éxito.

Desarrollar una aplicación web que sea "responsive" proporciona mucha flexibilidad para que el usuario pueda utilizarla tanto en dispositivos móviles como en ordenadores o tables. Sin embargo, aunque proporciona flexibilidad, también tiene sus inconvenientes:

- Están altamente limitadas en el acceso al hardware del dispositivo.
- Tiempos de respuesta y experiencia de usuario propios de una navegación web.
- No se encuentran en las Stores, siendo este el lugar donde la gente está acostumbrada a buscar aplicaciones.

Para solucionar todos los problemas que hemos mencionado anteriormente, en los últimos años surgió otra manera de hacer las aplicaciones multiplataforma que recibe el nombre "**Aplicación Híbrida**".

¿Qué son las aplicaciones híbridas?

Las aplicaciones híbridas son aplicaciones desarrolladas usando un único *stack* y empaquetadas para ser desplegadas en múltiples dispositivos, con diferentes tamaños de pantalla y fabricantes. Estas aplicaciones permiten al desarrollador construir sus productos en tecnologías simples como HTML, CSS y Javascript (JS). En otros casos, utilizando plataformas *server-side* para su implementación, por medio de lenguajes como C# y VB.NET.

En ese proyecto se decide la creación de una aplicación multiplataforma utilizando este tipo de framework y aprovechando flexibilidad respecto a las aplicaciones s nativas y mejoras de usabilidad respecto a la aplicación web, por último, enfocando el desarrollo de dicha aplicación a la formación e investigación en el ámbito de un Trabajo de Fin de Grado.

1.2 Objetivos

El objetivo general de este trabajo es creación de una aplicación multiplataforma basándose en el paradigma de aplicaciones hibridas que hemos mencionado anteriormente y tendrá como eje fundamental la gestión automática de un conjunto de encuestas siguiendo distintos criterios. Además, debe gestionar adecuadamente usuarios, encuestas, resultado de las encuestas y la estadística del resultado. Para alcanzar el objetivo principal se plantean los siguientes subjetivos:

- Analizar los paradigmas para el desarrollo de aplicaciones multiplataforma
- Diseñar una aplicación escalable que pueda ser ejecutada en múltiples dispositivos atendiendo a los requisitos definidos
- Crear la infraestructura de una aplicación multiplataforma (frontend y backend)
- Definir un entorno para validar la aplicación desarrollada.

El sistema se interactuará con una base de datos externa donde se almacenará tanto la propia información de las encuestas y el resultado del análisis de las encuestas.

Teniendo en cuenta su ámbito de uso, es necesario que la aplicación sea accesible desde cualquier plataforma como Android, iOS, Windows Phone y navegador web o desde dispositivos con diferentes tamaños de pantalla como PC, móvil o Tablet.

2. Estado del Arte

2.1 Introducción

Hoy en día las encuestas son muy útiles en todos los ámbitos de la sociedad. Es un método muy utilizado para recopilar las informaciones, opiniones y datos. Encontramos mucha variedad de herramientas para realizar las encuestas. La mayoría de ellos son aplicaciones web que se ejecutan directamente en la Web, pero también tienen opciones para ejecutarlas en otros dispositivos como móviles o Tablet, ya que los dos tipos de dispositivos tienen la opción de acceder a la Web mediante su navegador por defecto.

2.2 Herramientas similares

A continuación, vamos a ver las aplicaciones web existentes para realizar las encuestas:

2.2.1 Google Form

En primer lugar, encontramos el famoso Google Form, es una aplicación web de Google drive, en la cual podemos realizar formularios y encuestas para adquirir estadísticas sobre la opinión de un grupo de personas o informaciones deseadas. Podemos utilizarla en todos los ámbitos de nuestra vida: educativa, laboral, social, personal, empresarial, de ocio o simplemente pasatiempo.

Google Form nos dan a elegir entre unos montones de opciones de preguntas, desde las de tipo test hasta las listas desplegables con una escala numérica (como se muestra en la imagen 1). También nos dejan añadir imágenes y videos de YouTube o prueba algo más sofisticado con la ramificación de páginas y las preguntas filtro.

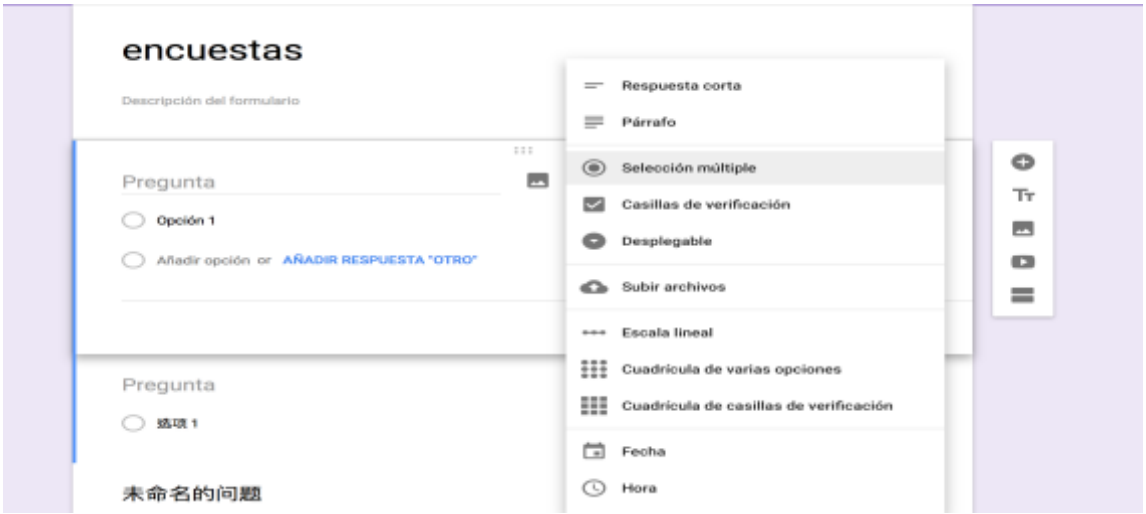

Imagen 1: selección de tipo de preguntas en Google form.

Una vez terminado de realizar la encuesta, se puede enviar nuestra encuesta mediante correo electrónico o compartir un enlace web en las redes sociales como Facebook o Twitter. Las respuestas de la encuesta se recopilan de forma automática y ordenada en

Formularios, con gráficos y datos de las respuestas en tiempo real. Además, también podemos concluir los datos y examinarlos en una hoja de cálculo.

Inconvenientes

- Se tiene que tener acceso internet para poder realizar y contestar la encuesta.
- Existen ciertas limitaciones relativas a las capacidades, ya que es un almacenamiento de nube gratuito.
- Su manejo podría no ser seguro si el usuario no crea una buena contraseña, si el usuario se equivoca al intentar compartir archivos, podría causar que los datos son accesibles a todo público.

2.2.2 Survey Nuts

Survey Nuts es una aplicación web freemium con una interfaz muy intuitiva que nos va a permitir realizar en la versión gratuita pequeñas encuestas de 10 preguntas como máximo y hasta 100 personas podrán responder a dicha encuesta. Las estadísticas serán a tiempo real, podremos ver las respuestas individualizadas y podremos elegir entre varios diseños para dar un mejor aspecto a nuestra encuesta en línea.
En el caso de que queramos opciones más avanzadas, tiene la versión completa a 29 €/ mes, versión de la aplicación web para encuestas con la que no vamos a tener límite alguno, características avanzadas, posibilidad de extraer en un Excel o en un csv y muchas más características extra.

Vemos en la página inicial algunas de las características principales de esta aplicación para encuestas: (Como se ve en la imagen 2)

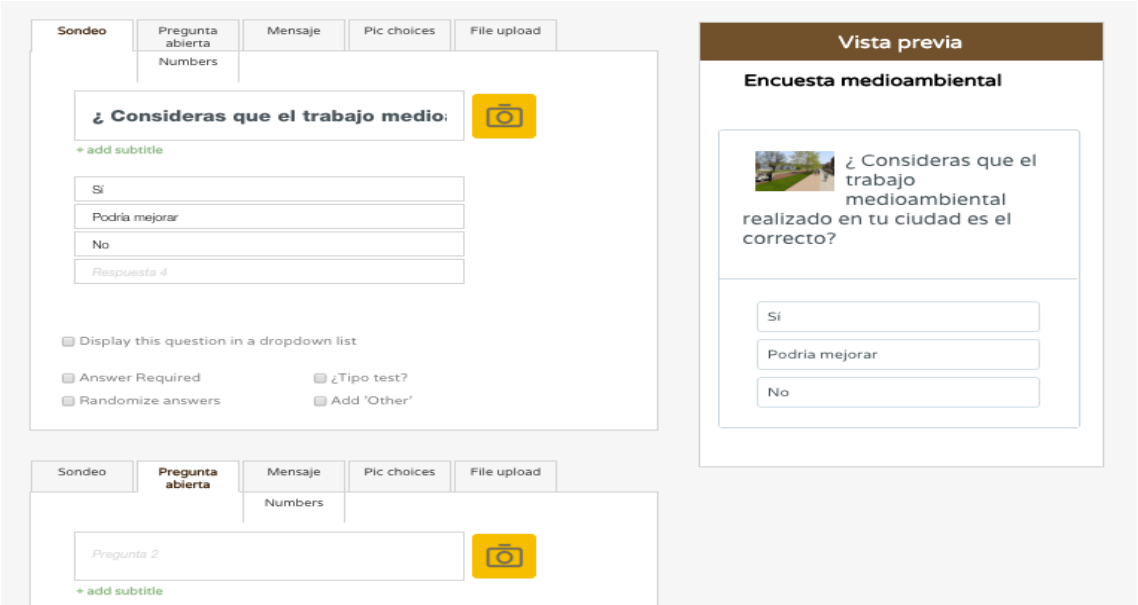

Imagen 2: interfaz de la aplicación Survey Nuts

Como vemos en la imagen superior en primer lugar nos deja elegir entre Sondeo, Pregunta abierta, mensaje, Pic choices y file upload.

2.2.3 SurveyMonkey

SurveyMonkey (Imagen 3) una herramienta freemium que nos va a ayudar a crear, recopilar y analizar nuestras encuestas. En este caso nos tendremos que registrar en la plataforma o iniciar sesión con nuestra cuenta de Facebook o Google, si no queremos agregar datos.

Al igual que Google Drive, es una herramienta muy intuitiva y por tanto sencilla de usar. Tras registrarnos veremos rápidamente el botón crear encuesta y empezaremos la creación de una nueva encuesta, aunque también tenemos la opción de copiar una que hayamos hecho anteriormente para algo similar, o utilizar alguna de las que nos vienen por defecto en la plataforma

Imagen 3: Interfaz de crear encuesta de SurveyMonkey

También vamos a tener gran variedad de tipos de pregunta en nuestra encuesta, como vemos a continuación, y podremos ir añadiéndolas de forma similar a los formularios de Google Drive

2.2.4 Kahoot!

Es una herramienta gratuita que permite la creación de diferentes tipos de actividades educativas a modo de juego, en el que los participantes pueden competir entre ellos y recibir retroalimentación inmediata sobre su desempeño.

La plataforma permite el diseño de cuatro tipos distintos de actividades:

- **Quiz (cuestionario):** presenta una serie de preguntas de selección múltiple con sus respectivas opciones. Al finalizar, se muestra la puntuación obtenida por cada jugador en una tabla de posiciones de acuerdo con el número de aciertos y a la velocidad de respuesta.
- **Discussion (foro):** funciona a modo de votación. Se inserta una sola pregunta con las respectivas opciones de respuesta y arroja los porcentajes que obtuvo cada una al finalizar la actividad. Esta modalidad permite iniciar un debate o discusión en torno a un tema específico.

- **Survey (encuesta):** presenta una o más preguntas con las respectivas opciones de respuesta. Funciona como herramienta para recolectar información sobre la opinión de los participantes respecto a un tema, mas no arroja una calificación.
- **Jumble (orden correcto):** crea una pregunta o enunciado y define las opciones de respuesta ofrecidas como una secuencia de pasos que lleva un orden específico. El jugador debe indicar el orden correcto en el que van las opciones para acertar.
  El docente puede asignar una cantidad definida de puntos y tiempo límite a cada pregunta de las actividades. Además, puede complementar los enunciados y problemas con imágenes y videos.

Para hacer uso de las herramientas de creación, el usuario (generalmente el docente) debe crear una cuenta personal en la plataforma. Una vez ha diseñado su Kahoot, la plataforma genera un pin de acceso para los participantes, quienes pueden hacer parte de las actividades sin necesidad de tener una cuenta propia o descargar la aplicación a sus dispositivos.

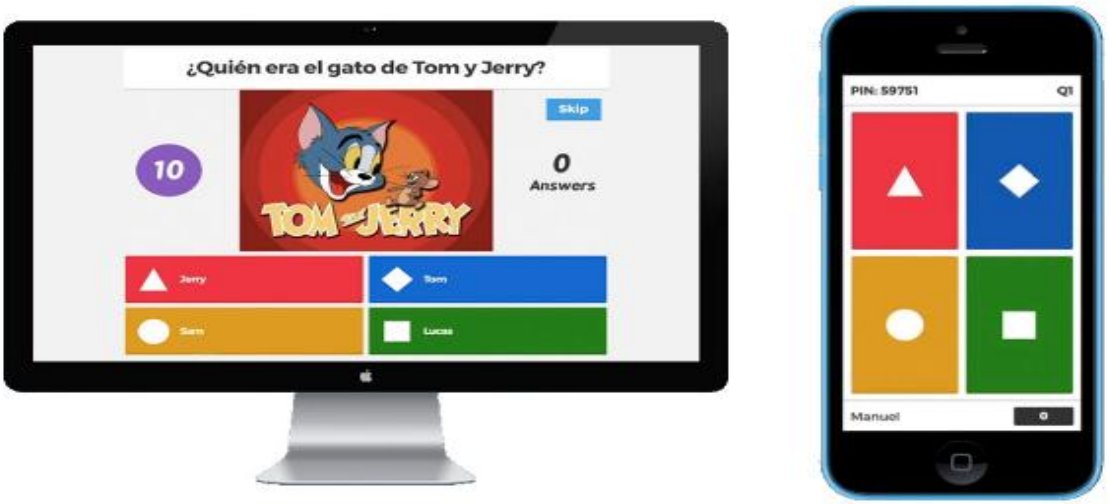

Imagen 4: la encuesta de Kahoot!

2.3 Análisis

Después de exponer diferentes tipos de herramientas existentes en el mercado que trabajan con las encuestas vamos a proceder a analizarlas para extraer más información entre estas cuatro herramientas y poder compararlas con mayor precisión con nuestra aplicación multiplataforma.

En este apartado del capítulo nos centraremos en dos tipos de análisis, según sus características cuantitativas y sus características cualitativas.

Para las características cualitativas se valorará a calidad de dichos programas de forma subjetiva mediante características capaces de medir según nuestro criterio. Para las características cuantitativas se emplearán técnicas muy objetivas; como la observación controlada y estructurada de la que se extraen inferencias externas más allá de los datos. Tanto para las características cuantitativas o cualitativas se verán reflejadas en dos tablas gráficas.

2.3.1 Características cualitativas

Este análisis lo hemos medido con las siguientes características que permiten elaborar una tabla objetiva valorando sus puntos fuertes como débiles en el ámbito cualitativo. Estos datos se recogerán en la tabla 1.

- **Diseño:** Aspecto y estética del sistema.
- **Facilidad de aprendizaje**: Capacidad para un usuario que desconozca la herramienta para aprender a utilizar toda la información.
- **Facilidad de uso**: Capacidad del usuario para manejar la herramienta. Esfuerzo que requiere la herramienta para su utilización.
- **Contenido**: Medición del contenido de la herramienta al sistema

Para valorar estas características he utilizado la escala de Likert puesto que permite definir mejor los criterios anteriormente mencionados. La escala es la siguiente:

1. total desacuerdo.
2. en desacuerdo.
3. ni de acuerdo ni en desacuerdo.
4. de acuerdo.
5. totalmente de acuerdo.

Herramientas	Diseño	Facilidad de aprendizaje	Facilidad de uso	Contenido
Wentis	4	5	4	3
Google Form	3	4	4	3
Survey nuts	3	3	4	3
SurveyMonkey	4	3	4	4
Kahoot!	4	3	4	4

Tabla 2: Características cualitativas

2.3.2 características cuantitativas

Para elaborar un análisis cuantitativo de las cuatro herramientas propuestas existentes en el mercado y también nuestra aplicación multiplataforma, hemos clasificado una serie de características detalladas a continuación y que mostraremos en la tabla 2:

Precio: Determina si la herramienta es de pago indicando su coste o gratuita.
Autentificación: Determina si es necesario autentificarse con una cuenta o no.
Cuentas: Determina si gestiona varias redes sociales o solo una.
Plataforma: Determina si la herramienta soporta una o más de una plataforma.

Herramientas	Precio	Autenticación	Cuentas	Plataforma
WenTis	Gratuito	Si / No	Varias	Todos
Google Form	Gratuito	Si	Google	Web
SurveyNuts	Gratuito o 29€/completa	Si	Propia	Web
SurveyMonkey	Gratuito 35€/mes	Si	Varias	Todos
Kahoot!	Gratuito	Si	Varias	Todos

Tabla 3: Características cuantitativas

Como conclusión de los sistemas analizados anteriormente podemos extraer como información útil que las herramientas visualizadas en el análisis presentan una mayor dificultad de aprendizaje en los sistemas más completos, es decir, aquellos sistemas que incorporan más funcionalidades. También podemos apreciar que las herramientas de pago trabajan con un numero de redes sociales más amplio orientado hacia uso profesional, permitiendo trabajar a los usuarios en distintos ámbitos. En general, son herramientas más elaboradas que están en continuo desarrollo y perfeccionamiento. Por otra parte, en las herramientas gratuitas como Google Form y Kahoot son más populares y de mayor influencia en la red que las otras. En el caso de Google Form, aunque el contenido no es muy abundante que las otras, pero tiene un mayor número de usuario debido que es una aplicación web propia de Google y su facilidad de uso. El Kahoot se destaca debido a su abundante contenido y su diseño de interfaz, pero en la parte de encuestas no se centra tanto como las demás funcionalidades.

Con nuestra aplicación multiplataforma(WenTis) obtendremos datos de interés únicamente se centra en el sector de la educación logrando que la vida estudiantil mejore tanto para el docente como para el alumnado. Además, gracias a su carácter multiplataforma se adapta a todos los tipos de dispositivos y plataformas. Por último, al diferenciarse con las demás herramientas, nuestra aplicación multiplataforma será completamente gratuita.

2.4 Tecnologías

2.4.1 Nuevos FrameWorks

Para la realización del proyecto debemos repasar todos los frameworks disponibles en la actualidad para implementar la aplicación hibrida de forma que elijamos la que mejor se adapte a nuestras necesidades.

Frameworks	Ventajas	Inconvenientes
Ionic 2	Ofrece una amplia variedad de componentes. Con la misma base de código es capaz de funcionar en las mayorías plataformas. El coste y el tiempo de desarrollo es sensiblemente menor.	No tiene acceso a las funciones de los componentes hardware. El rendimiento suele ser menor que una aplicación nativa. Al ejecutarse en una web view, puede ser menos ricas y potentes que las disponibles en nativos.
ChocolateChipUI	Está basado en JQuery. Presenta una amplia variedad de componentes. Presenta temas muy semejantes al nativo.	Como Ionic no tiene componentes para acceder al hardware del dispositivo por lo que hay que utilizarlos plugin de Apache Córdova.
RatChet	Presenta temas IOS y Android. Permite utilizar knockout, AngularJS, DurandalJs etc.	Componentes reducidas. Hay que utilizar plugin para acceder a la funcionalidad hardware.

	Puede utilizarse para realizar una web aplicación.	No funciona en otra plataforma que no sea IOS o Android.
Intel's App	Permite utilizar AngularJS, Backbone y DurandalJs. Es ligero y rápido. Soporta JQuery.	Hay que utilizar plugins para acceder a la funcionalidad hardware.

Tablas 4: Descripción de las características de cada tecnología.

Tras realizar un análisis de las ventajas e inconvenientes de los frameworks disponibles, hemos decidido utilizar el framework "Ionic 2[1]" para nuestro proyecto. Además, revisaremos todas las tecnologías necesarias a la hora de implementar la aplicación utilizando Ionic2.

Ionic 2 es un framework para el desarrollo de aplicaciones híbridas, inicialmente pensado para móviles y Tablets, aunque ahora también es capaz de implementar aplicaciones web e incluso dentro de pocas aplicaciones de escritorio multiplataforma. Su característica fundamental es que usa por debajo de Angular 2 [6] y una cantidad de componentes enorme, que facilita mucho el desarrollo de las aplicaciones.

En el caso de nuestra aplicación se implementaría utilizando el framework Ionic 2 y con su lenguaje de programación TypeScript [5] (Angular 2) para la implementación de los componentes de la aplicación en la que se definirán la lógica de la aplicación.

2.4.2 HTML

"*HTML*, siglas en inglés de *HyperText Markup Language* (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web." (Wikipedia: *HTML*)

A través HTML implementamos la parte interfaz de la aplicación. En concreto, la quinta versión HTML 5, que introduce nuevas características para ayudar a desarrollador de aplicaciones web. En nuestro caso, utilizaremos para implementar la parte de las vistas, al igual que CSS.

2.4.3 CSS

Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en *HTML* o *XML* (y por extensión en *XHTML*)" (Wikipedia: Hoja de estilos en cascada)

Las hojas de estilo ya llevan años presentes en el mundo de la programación en la web. Su utilidad y facilidad a la hora de estilizar un sitio online las ha convertido en la herramienta indispensable de cualquier desarrollador web. Hoy en día recibimos su tercera versión CSS 3 que incorpora más funcionalidades para implementar más estilo de páginas. En nuestro caso podemos decir que dicho lenguaje es la esencia de nuestra aplicación, es decir, gracias al CSS3 la interfaz es visualmente agradable.

2.4.4 TypeScript

TypeScript es un lenguaje de programación libre y código abierto desarrollado y mantenido por Microsoft. Es un supe conjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases.

Extiende la sintaxis de JavaScript, por tanto, cualquier código JavaScript existente debería funcionar sin problemas. Está pensado para grandes proyectos, los cuales a través de un compilador de TypeScript se traducen a código JavaScript original.

Como ya hemos mencionado antes, TypeScript es el lenguaje de programación que estamos utilizando para implementar la parte funcional de la aplicación. Como durante la carrera hemos estudiado y utilizado el conocimiento sobre JavaScript, entonces eso nos facilita bastante a la hora de programar en TypeScript, ya que TypeScript es muy similar al JavaScript y de hecho en algunas ocasiones resulta más simple que el JavaScript.

2.4.5 Angular 2

Angular 2 es un framework para las aplicaciones web de TypeScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

Aunque en nuestro caso no estamos utilizando el framework Angular 2 pero el uso de Ionic 2 es muy similar al uso de Angular 2, ya que Ionic 2 está basado en Angular 2 pero es más orientado a los dispositivos móviles.

2.4.6 Firebase (Google)

Firebase [2] se trata de una plataforma móvil creada por Google, cuya principal función es desarrollar y facilitar la creación de aplicaciones de elevada calidad de una forma rápida, con el fin de ayudar a facilitar la gestión de la aplicación, el aumento de la base de usuarios y, por ende, su monetización. Está disponible para diferentes plataformas como IOS, Android y Web.

Para realizar una aplicación multiplataforma requiere de una base de datos que pueda acceder la parte web y la parte móvil, por lo cual, tenemos que descargar soluciones en un servidor externo. Si esta base de datos la implementaremos con un SGBD SQL como MySQL, tendríamos que realizar un API de conexión. Sin embargo, existen soluciones en el Mercado que han aparecido en los últimos años que te facilitan este desarrollo (MongoDB y Firebase), puesto que se ofrecen una API para acceder a los datos y únicamente solo tenemos que ocuparnos del diseño. Después de un análisis de las ventajas e inconvenientes sobre las bases de datos en el Mercado, hemos decidido de utilizar Firebase como la base de datos de nuestra aplicación.

3. Propuesta

3.1 Análisis de requisitos

En este capítulo de análisis de requisitos se pretende determinar la funcionalidad completa de la aplicación multiplataforma de una manera integrada pero también hemos seguido una metodología ágil con Sprint de una semana para tener un MVP (Mínimo Producto Viable). Es decir, durante la semana se trabaja por mi cuenta y reuníamos un día de cada semana para resolver las dudas y planificar el siguiente Sprint. Para organizar el trabajo hemos utilizado la herramienta Trello (Imagen 5):

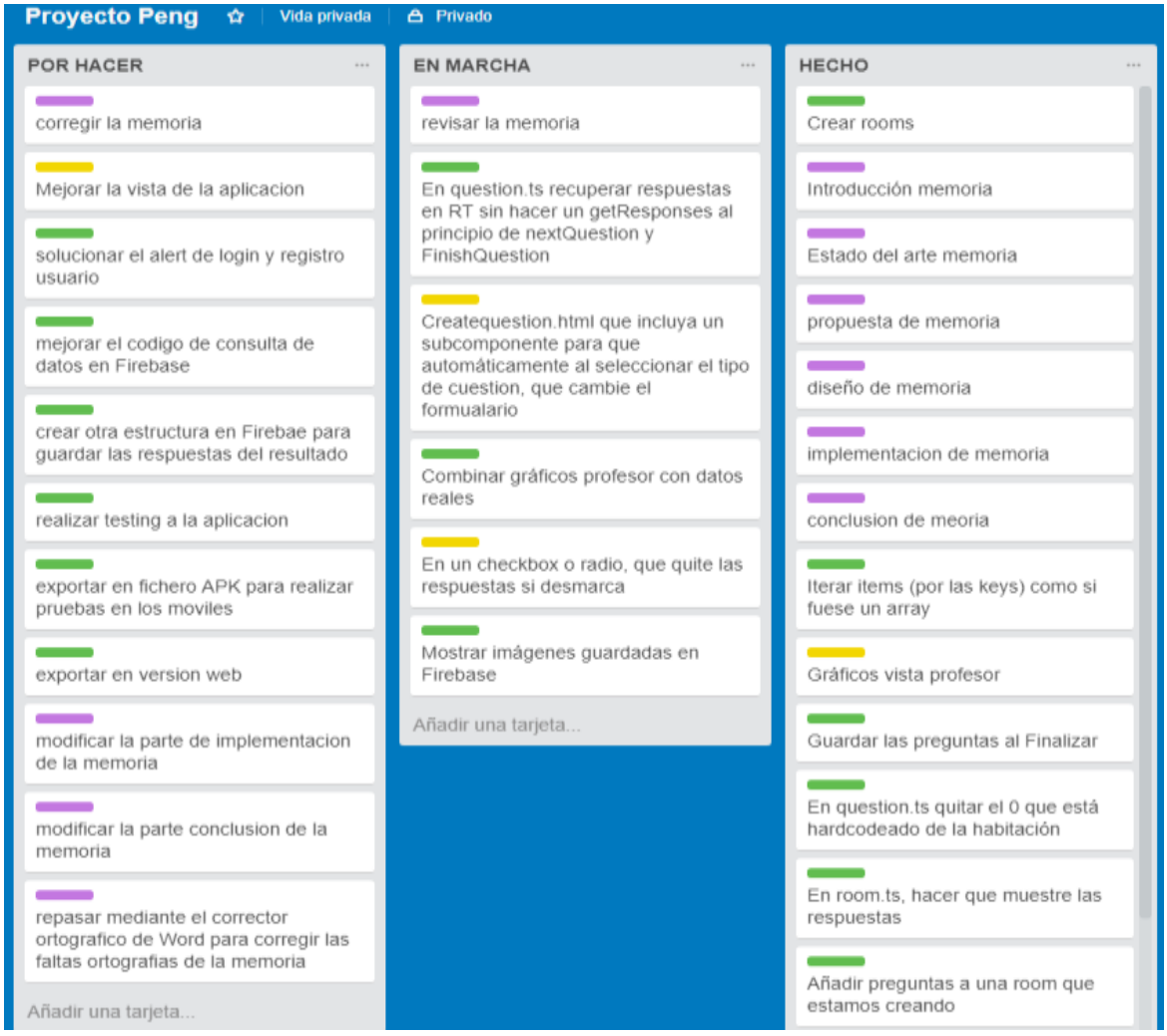

Imagen 5: la organización de trabajo Trello.

En la aplicación existen dos tipos de usuarios: los usuarios profesores y los usuarios alumnos. En esta aplicación únicamente solo los usuarios profesores tienen la posibilidad de acceder a todos los tipos de funcionalidad de la aplicación puesto que ellos son los encargados de crear y publicar las encuestas. A continuación, vamos a ver los roles y funciones que tienen cada uno de ellos.

Usuarios	Roles o Funciones
Profesores	Iniciar y registrar en la aplicación. Crear sala de encuestas Entrar en la sala y crear las encuestas Visualizar el resultado de la encuesta.
Alumno	Con el PIN entrar en sala y realizar la encuesta que han publicado los profesores.

Tabla 5: Rol de usuarios

3.1.1 Ámbito de la aplicación.

Esta aplicación que hemos desarrollado recibe el nombre "**WenTis**", porque hemos cogido la pronunciación de la palabra "Pregunta" en Chino Mandarín. Este proyecto tiene el objetico de cumplir todos los requisitos del proyecto que hemos propuesto, y está enfocada en medir el estado emocional y la satisfacción de los alumnos universitarios o cualquier ámbito académico. A continuación, vamos a detallar las funcionalidades que implementa la aplicación y las que no implementa.

La aplicación permitirá:

- Iniciar sesión con una cuenta propia de la aplicación. En el caso de no tener cuenta la aplicación tiene la posibilidad de registrarse desde la aplicación.
- La aplicación podría crear una sala [1]donde el usuario puede ir creando encuesta que le convenga.
- Cada sala se generará manualmente o automáticamente un numero PIN en el que luego los alumnos escogen ese número PIN para entrar en la sala y realizar la encuesta de dicha sala.
- El usuario (Profesor) puede ver las estadísticas del resultado de la encuesta mediante la vista gráfica.
- El usuario puede publicar un número indeterminado de salas o encuestas.

El sistema no permitirá:

- Iniciar la sesión mediante la cuenta de las redes sociales.
- Eliminar las salas y encuestas creadas.

Con este proyecto se espera tener un mayor conocimiento sobre la calidad de las clases de los profesores ayudándoles a mejorar y a favorecer un ámbito de aprendizaje más ameno en el que ambas partes, tanto profesor como alumno sean capaces de sacar lo mejor de sí.

3.2 Descripción de la aplicación

Como ya hemos mencionado anteriormente, la aplicación multiplataforma forma parte de un proyecto colectivo de los profesores que se utilizara para medir las emociones de los alumnos universitarios. Por lo tanto, este proyecto estará desarrollado por los actores mencionados en el análisis de los requisitos, así como este facilitará datos para otros proyectos.

[1] **Sala:** Una habitación o una Room donde se sitúa la encuesta y su información.

3.2.1 Funciones del producto

Las funciones principales de la aplicación las podemos recoger en los diferentes casos de uso que se detallaran en la imagen 6. En esta imagen se indicarán los casos de uso de la aplicación completa.

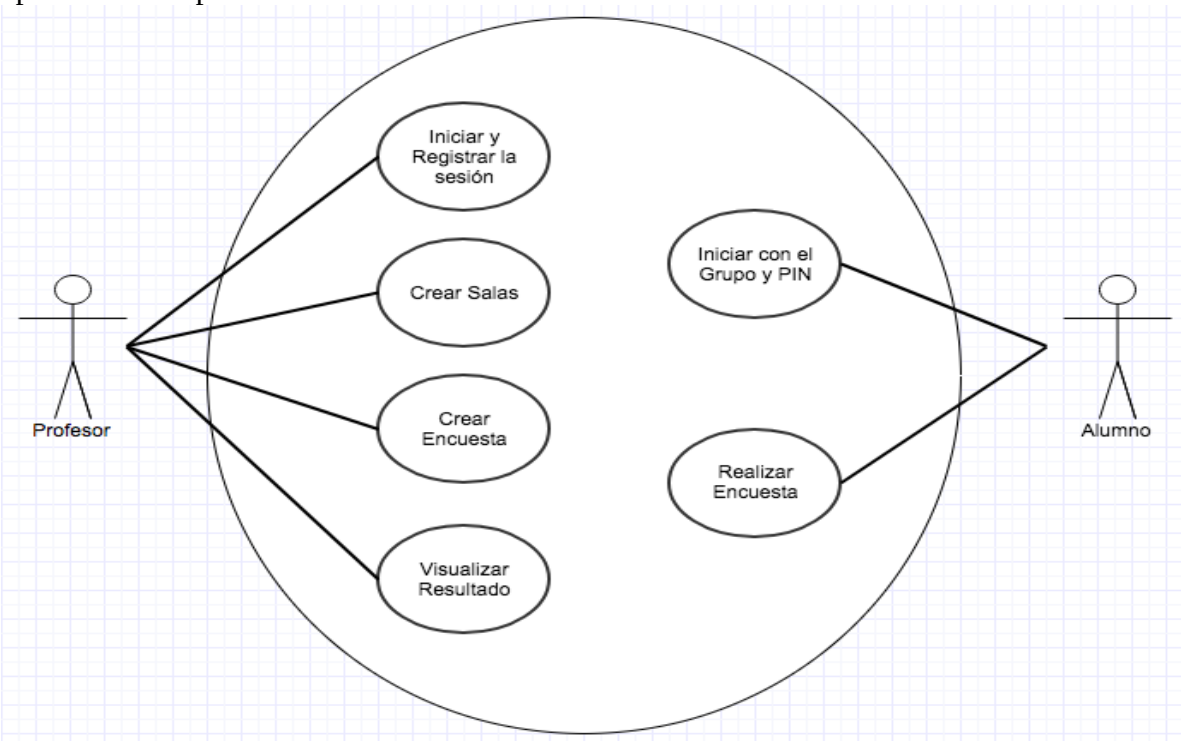

Imagen 6: Diagrama de casos de uso.

Caso de Uso	Descripción
CU01	Iniciar y Registrar la sesión.
CU02	Visualizar y Crear una sala
CU03	Crear una Encuesta
CU04	Visualizar el Resultado
CU05	Iniciar con el Grupo y el PIN
CU06	Realizar Encuestas

Tabla 6: Casos de uso

3.2.2 Restricciones

En esta aplicación que estamos desarrollando se identifican dos tipos de restricciones, las restricciones de usuarios profesores y las restricciones de usuarios alumnos.

Restricciones de los profesores:

- En cada sala el usuario profesor solo puede crear una encuesta.
- El usuario profesor necesitara obligatoriamente una cuenta para entrar en la aplicación.
- El usuario profesor necesitara en todo momento el acceso a Internet.

Restricciones de los alumnos:

- El usuario alumno no puede hacer login ni registrar en la aplicación.
- El usuario al no está registrado no se puede utilizar unas determinadas funciones de la aplicación.
- Los alumnos pueden visualizar el resultado de la encuesta que ellos habían realizado.
- Los alumnos necesitaran en todo momento el acceso a Internet.

3.2.3 Suposiciones y dependencias

Destacamos que, en este proyecto, para el correcto funcionamiento de la aplicación, necesitamos un número notable de alumnos que valoren una clase para poder tener estadísticas más precisas.

Por otra parte, necesitamos a lo largo del proceso dependencia de Internet, puesto que no utilizamos un servidor privado, utilizamos una base de datos en línea para intercambiar información sobre distintas aplicaciones.

3.2.4 Requisitos futuros

Para mejorar el sistema en futuras versiones podríamos incluir los siguientes requisitos:

- Añadir más funcionalidades de las redes sociales como: compartir resultados de encuestas, foros o chats entre los usuarios.
- Para cada pregunta se podría ofrecer la opción de subir imágenes.
- Exportar la aplicación a diferentes plataformas y subirla en sus tiendas de aplicaciones.
- Ampliar el servicio para docentes de diferentes escuelas o universidades.

4. Diseño

4.1 Introducción

En este apartado vamos a detallar de donde se surge la idea de inspiración para el diseño de nuestra aplicación multiplataforma. Posteriormente se ha elaborado un esquema conceptual en el que se mostrara de forma genérica como funciona nuestro sistema. Una vez detallados todos estos aspectos se redactarán de forma más precisa todos los componentes de nuestro sistema. Para ello se han elaborado una serie de diagramas que permitirán explicar con más claridad dichos componentes. Finalmente， se ha elaborado una conclusión para resumir aspectos y matices importantes

4.2 Descripción de la arquitectura

En este apartado vamos a mostrar el ciclo funcionamiento de la aplicación y los distintos componentes involucrados.

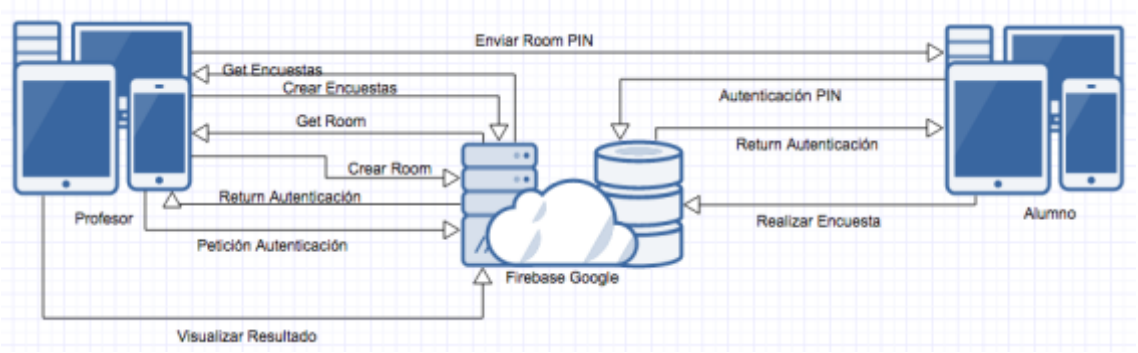

Imagen 7: Diagrama esquema conceptual

Como se muestra en este esquema de la imagen 7, el usuario de tipo profesor se identifica directamente a través de la cuenta propia del servidor Firebase(Petición Autenticación y return Autenticación). Una vez con la sesión activa en la aplicación móvil, el usuario de tipo profesor puede utilizar una sala existente a lo largo de la sesión o también se puede crear una sala nueva. Una vez estamos dentro de una sala, el usuario profesor puede crear la lista de las preguntas de la encuesta de dicha sala. También se puede añadir más preguntas a la encuesta o incluso crear una nueva encuesta.

Por otro lado, el usuario de tipo alumno se autentica con el servidor mediante el nombre del usuario y el número PIN que ha obtenido desde el usuario de tipo profesor. El servidor comprueba el número PIN y devuelve la sala que corresponde a ese número PIN al usuario de tipo alumno. El alumno obtiene la sala y puede empezar a realizar la encuesta de dicha sala. Finalmente, una vez que el usuario de tipo alumno finaliza la encuesta y guarda el resultado en el servidor, el usuario de tipo profesor puede visualizar la estadística del resultado que ha obtenido el usuario de tipo alumno.

4.3 Especificación Formal

Para este apartado se especificarán formalmente los diferentes componentes que interactúan en el sistema de forma detallada. Para poder elaborar bien la fase de diseño de especificación formal, hemos decidido utilizar el modelo por capas en el que se han distinguido tres capas, la capa de presentación, la capa lógica y la capa de persistencia.

4.3.1 Capa de presentación

En la capa de presentación vamos a especificar mediante la herramienta mockUps el diseño de la interfaz externa que vera el alumno cuando utilice la aplicación multiplataforma. También vamos a detallar las diferentes interacciones para poder navegar por la aplicación y de esta forma poder entender mejor el funcionamiento del sistema a través de esta capa externa y que es una de las más significativas para los usuarios de la aplicación.

A la hora de diseñar, tenemos que tener en cuenta que el diseño de la aplicación multiplataforma tendría que ser lo más sencillo e intuitivo para mejorar la facilidad de uso de los usuarios. Se pretende hacer un diseño que sea sencillo de entender por todos y que sea lo más rápido posible.

Antes de ver los diferentes mockUps veremos un diagrama en el que se muestran las principales Pages de la aplicación y la relación que tienen entre ellos. Cada page de la aplicación tiene asociado un layout que mostrara el contenido grafico de cada ventana. Una vez mostrado el diagrama, se procederá con las explicaciones de los diseños gráficos con sus diferentes interacciones para tener una mayor claridad sobre la capa de presentación.

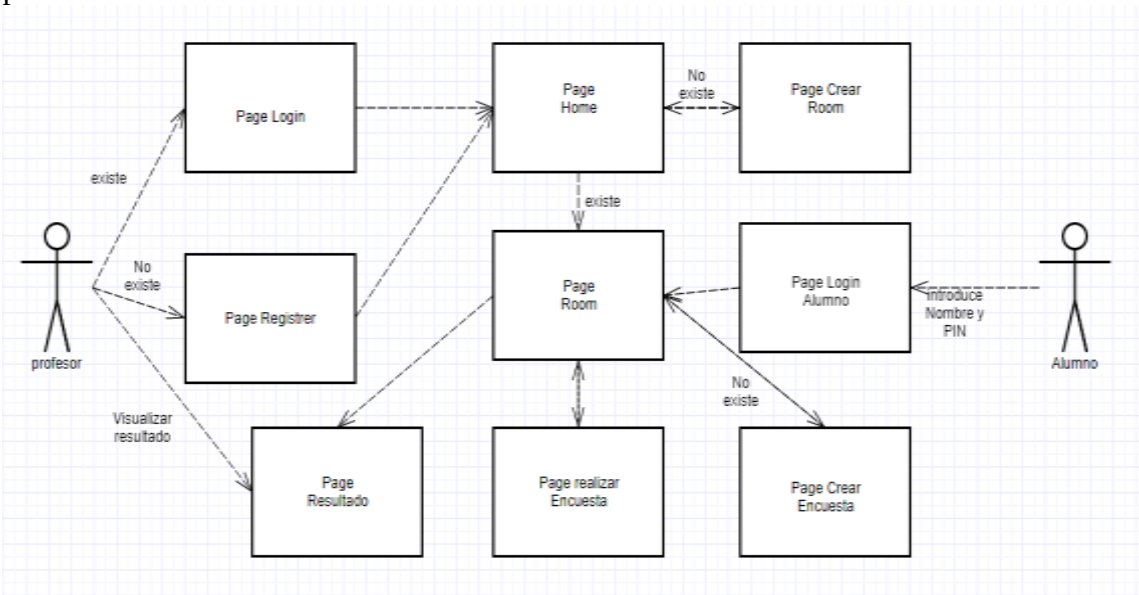

Imagen 8: Diagrama de page de la capa de presentación.

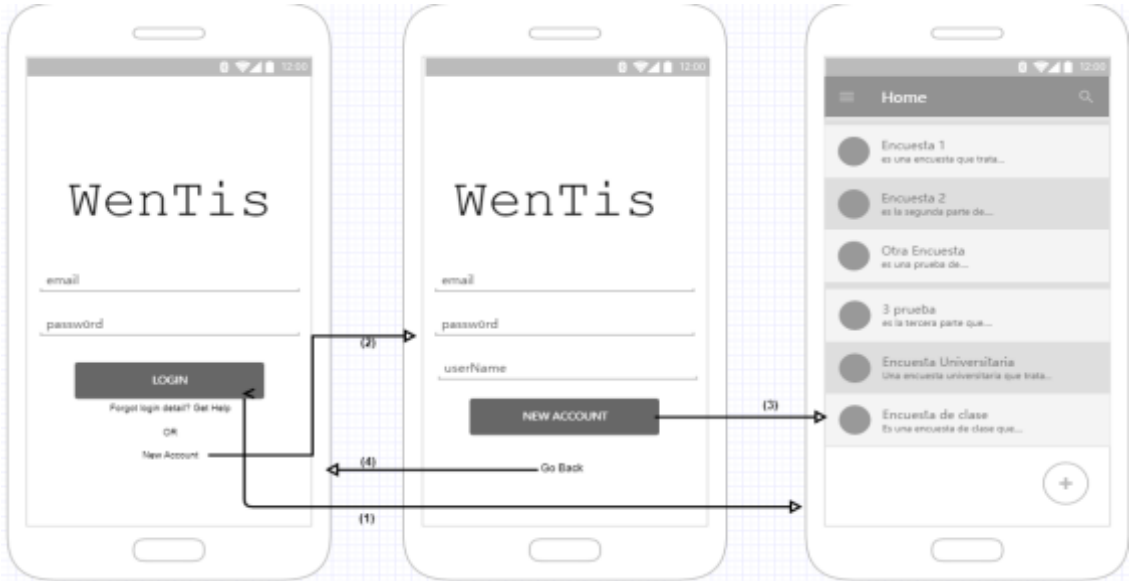

Imagen 9: MockUps de Iniciar sesión y registro

En primer lugar, empezamos con el page profesor que se dirige hacia un el page de iniciar la sesión (CU01), si tenemos una cuenta simplemente tendremos que introducir el email y la contraseña y hacer un clic en el botón LOGIN que nos redirigirá al home page (1). Si no tenemos una cuenta, simplemente elegimos la opción "New Account" (2) y nos llevará al page de registro en la que nos pedirá igual que el page login un email, password y un userName. Una vez terminamos hacemos un clic y nos llevara directamente al home page (3). También existe una opción de retroceder al page login (4).

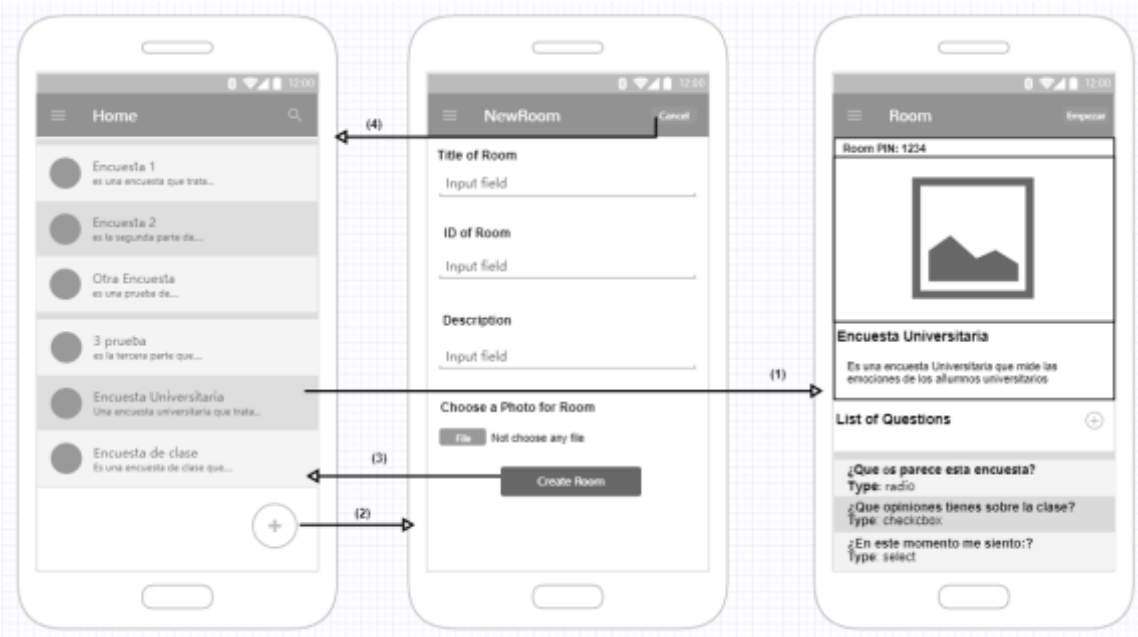

Imagen 10: MockUps de crear una sala (room)

Una vez estamos dentro del home page ya estamos dentro de la parte central de la aplicación. Este page nos demuestra una lista de todas aquellas salas que ha creado el usuario. El usuario puede escoger una sala de la lista y nos llevara al page de Room donde están todas las informaciones de dicha sala (1). Si acaso el usuario tiene la intención de crear otra sala nueva simplemente con el botón "+" se dirige hacia el page "NewRoom"

para crear una sala nueva (2), se introduce los datos de la sala y se finaliza con el botón create room y devuelve la sala nueva al home page (3). También existe la opción de cancelar para volver al Home page (4).

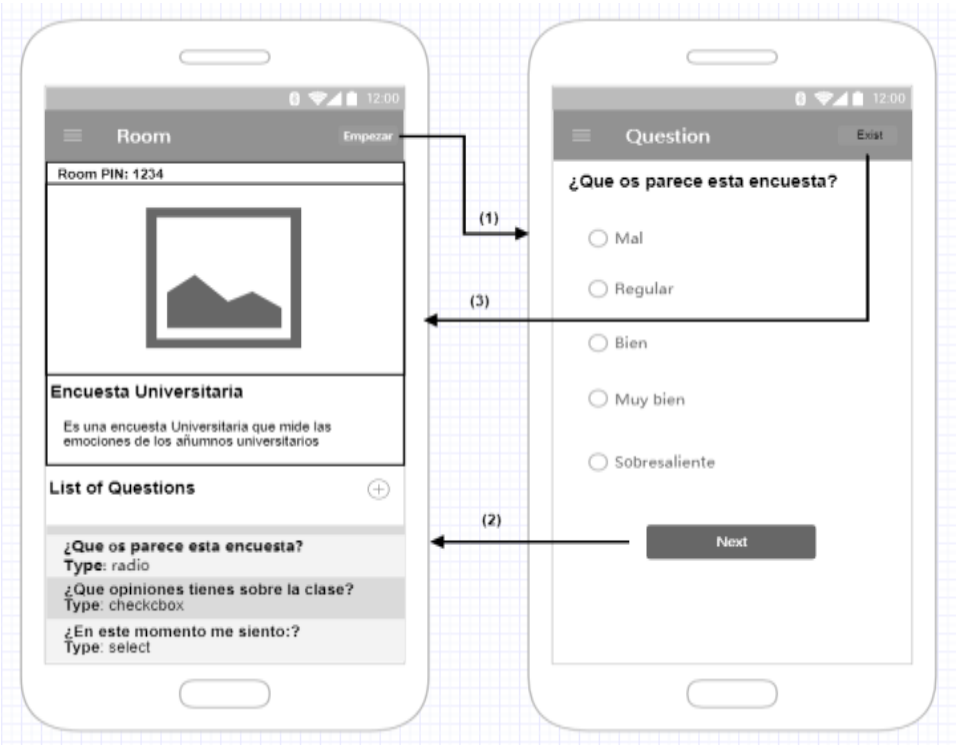

Imagen 11: MockUps para realizar la encuesta.

Desde el page sala, con el botón “Empezar” nos dirige hacia el page de realizar las encuestas (1). Como lo podemos ver en la imagen 13, dicho page está formado por una pregunta y un número determinado de respuestas que pueden ser de distintos tipos dependiendo del tipo de pregunta. Cada vez que terminamos de responder una pregunta es necesario hacer un clic al botón “Next” para dirigir a la siguiente pregunta hasta llegar la última pregunta donde el botón “Next” se cambiara a “Finish” para finalizar la encuesta (2). También existe la opción de salir y volver al page sala (3). El resultado de la encuesta se almacenará inmediatamente en la BBDD gracias al almacenamiento en tiempo real de Firebase.

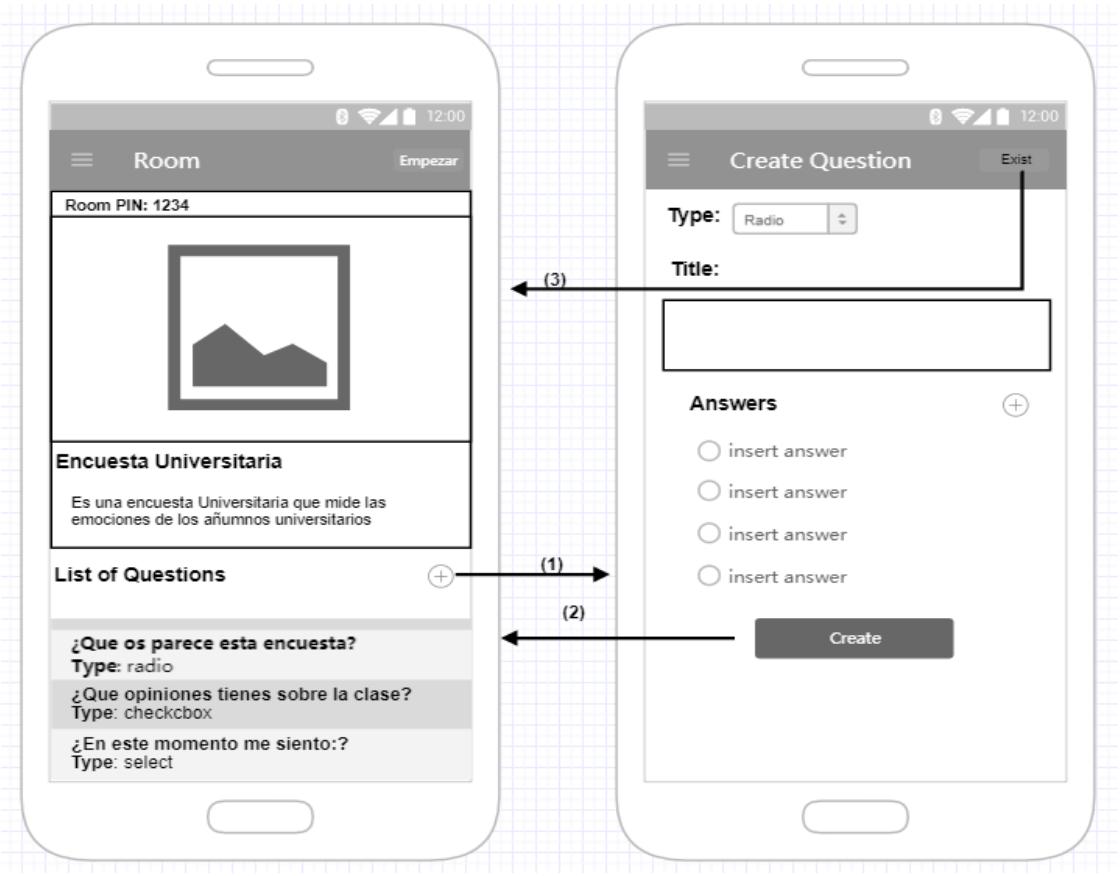

Imagen 12: MockUps para crear Questions

Por otra parte, si no existen preguntas o el usuario quiere añadir más preguntas, simplemente con el botón "+" se dirige hacia el page crear preguntas para una pregunta nueva (1). Como lo podemos ver en el page, en la parte superior hay una opción type para que el usuario elija el tipo de pregunta que quiere crear, de momento hay: radio, checkbox, select y respuesta corta. Por debajo, está un input-box para introducir la pregunta y opciones de respuesta del tipo que ha elegido el usuario, puede añadir más número de respuestas haciendo clic sobre el botón "+". Por último, para finalizar simplemente hacer falta hacer un clic con el botón "Create" y se devuelve un objeto pregunta al home page (2). También existe la opción de cancelar la creación de pregunta (3).

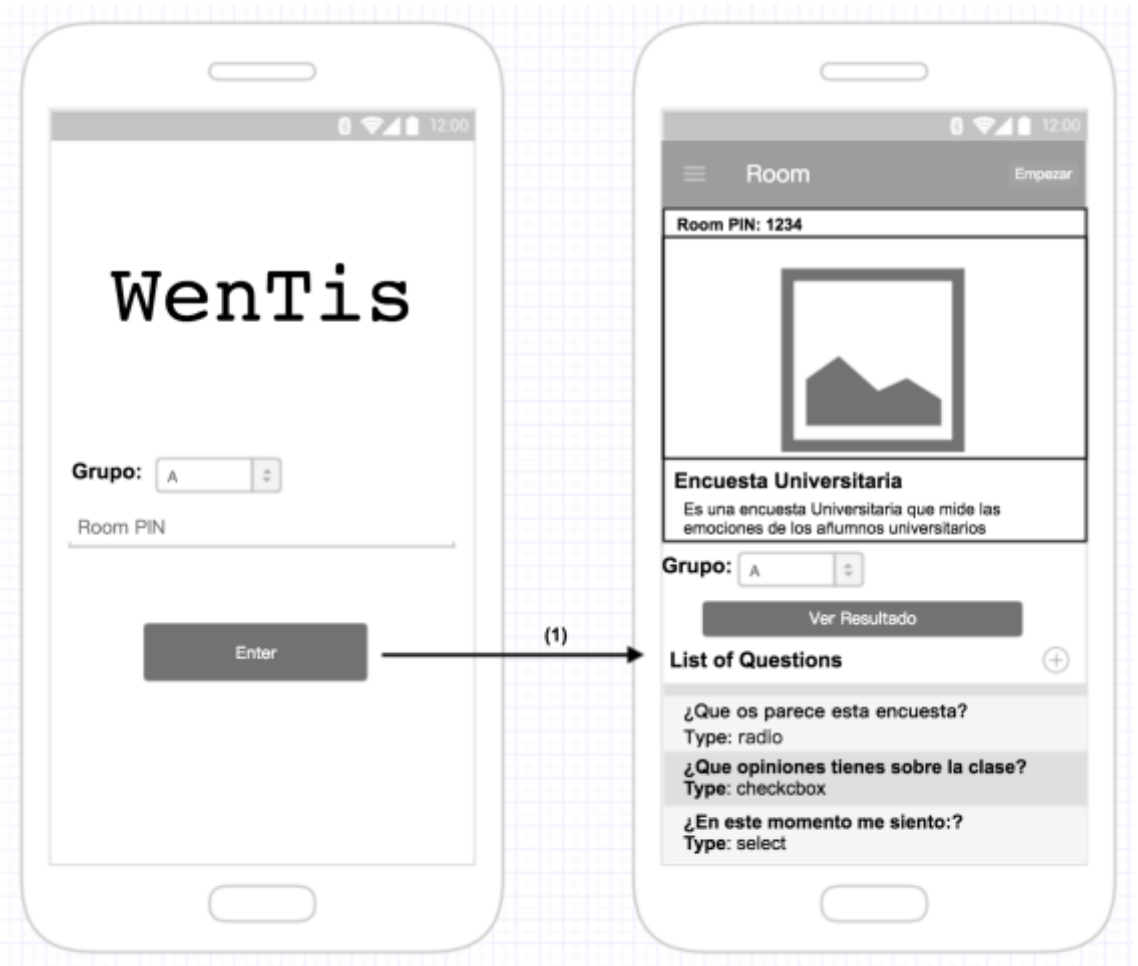

Imagen 13: MockUps de alumno para entrar una sala (room).

En el caso de usuario de tipo alumno, no es necesario registrarse o iniciar la sesión, simplemente selecciona el grupo y el room PIN del profesor y se dirige a la sala que corresponde a ese dicho PIN. El usuario de tipo alumno puede visualizar la información de la sala, pero no se puede realizar modificaciones como añadir una pregunta a la encuesta. El único permiso que tiene es hacer un clic al botón empezar para empezar a realizar la encuesta como hemos mostrado en la imagen 13 de realizar la encuesta. Cuando termina de contestar la encuesta, el resultado se almacenará directamente al Firebase y el usuario alumno ya no podrá realizar ninguna acción.

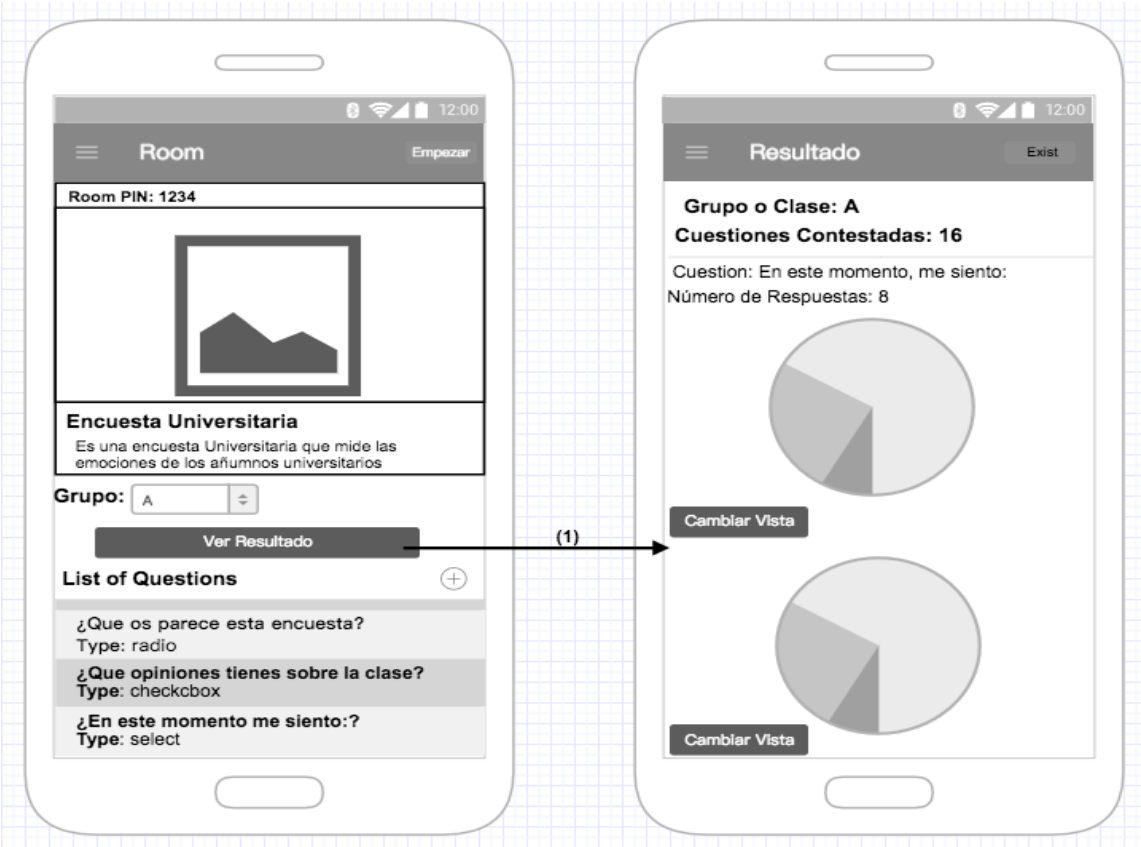

Imagen 14: Consulta de Estadísticas del Resultado.

Por último, vamos a ver la parte de consulta de estadísticas del resultado de la encuesta. Para ello, hemos diseñado un *select* para que el usuario seleccione el grupo que quiere ver y un botón para dirigir a la pantalla de estadísticas del resultado (1). En esta pantalla, hemos puesto el grupo y las cuestiones contestadas en la parte superior. A continuación, por debajo esta la gráfica de la respuesta de cada cuestión con su número de respuestas. El propósito de este diseño simplemente es para que el usuario sepa visualmente los detalles de las respuestas de cada pregunta.

4.3.2 Capa de negocio (capa lógica)

Para este apartado de diseño, vamos a detallar mediante diagramas de secuencias como son los diferentes casos de uso que expusimos en el capítulo 3 (imagen 6) de especificación de requisitos, de forma que quede detallado gráficamente su diseño.

A continuación, vamos a elaborar diferentes diagramas de secuencias para mostrar el flujo de datos entre los distintos casos de uso.

1. El usuario de tipo profesor inicia la sesión o registrarse una cuenta (imagen 15).
2. El usuario de tipo profesor gestiona la sala o crear una nueva sala (imagen 16).
3. El usuario de tipo profesor añadir una pregunta o crear una encuesta nueva (imagen 17).
4. El usuario de tipo alumno realiza la encuesta y almacenar el resultado (imagen 18).
5. El usuario de tipo profesor visualizar el resultado de la encuesta (imagen 19).

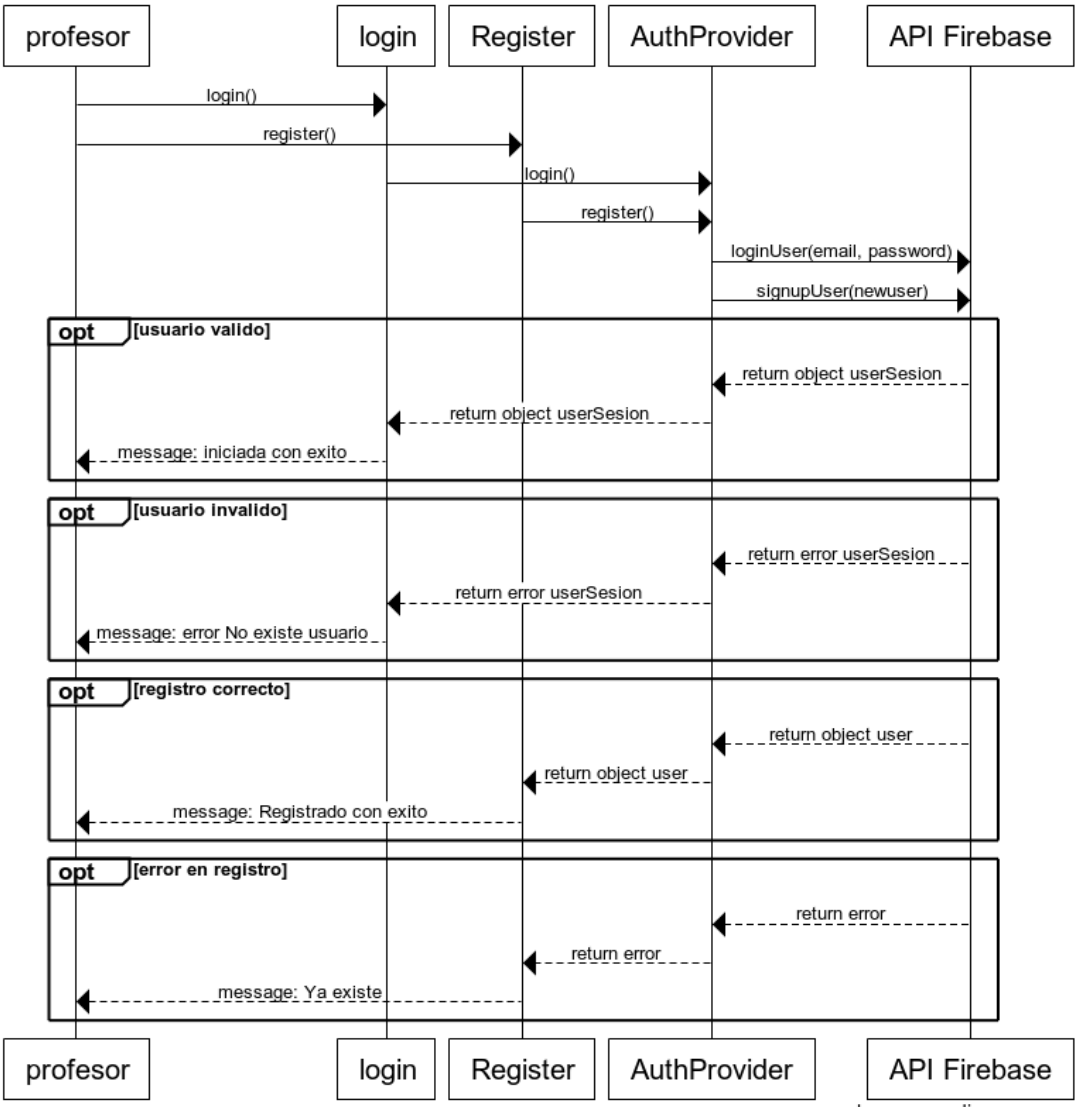

Imagen 15: esquema secuencia de iniciar sesión y registro.

Como se puede apreciar en la imagen 15, el profesor mediante el page Login solicita una petición de inicio de sesión de la aplicación. La petición pasa a un provider de autenticación (auth.ts) que encarga la comunicación entre la aplicación y la base de datos (Firebase). Si existe el usuario registrado en la base de datos devolverá el objeto user al provider de autenticación y él se encarga de devolver al page Login y mostrará un mensaje de confirmación como se muestra en la imagen 14. Por otra parte, si realizamos la acción de registrarse, el usuario profesor se dirigirá al page Register en el que se enviará una petición de registro al provider de autenticación para que comunique con Firebase una petición de registro de un nuevo usuario. El Firebase se comprobará si el usuario solicitado ya estaba en la base de datos o no, si no está, enviara una confirmación de que ha registrado correctamente, sino, enviara un error al provider de autenticación y page register para avisar al usuario mediante un mensaje de error de que ya existe el usuario.

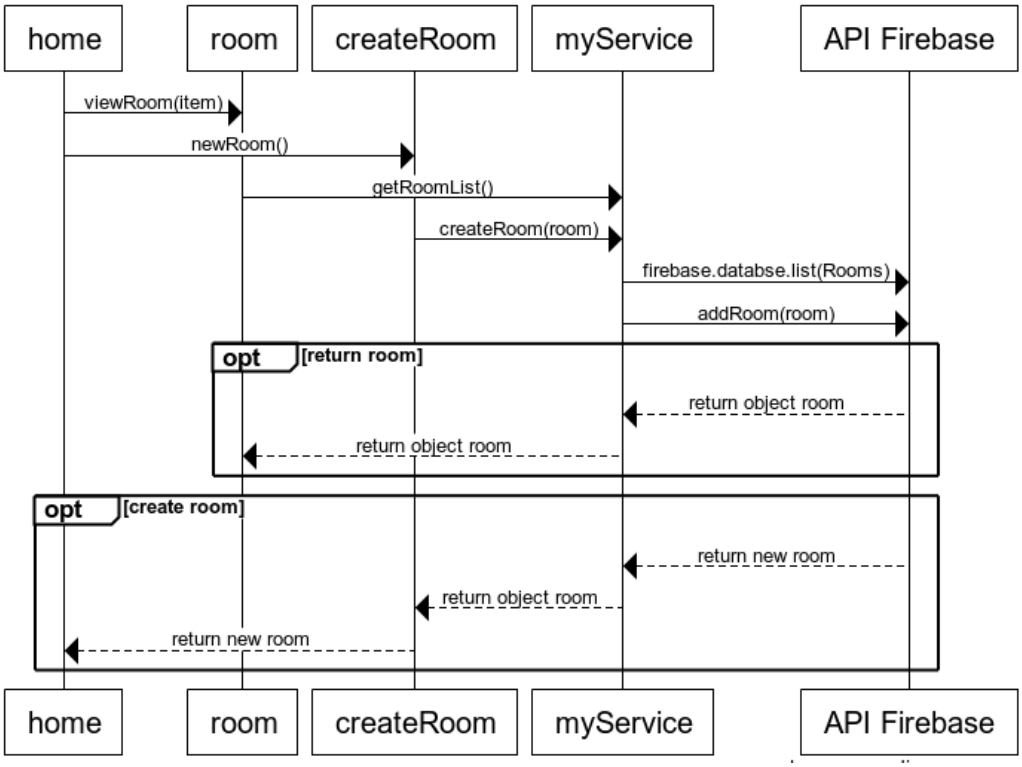

Imagen 16: gestiona la sala o crear una nueva sala

Como se detalla en el diagrama de la imagen 16 correspondiente al caso de uso de gestión y creación de una sala, el profesor puede seleccionar una room de la lista o directamente crear una nueva room. Si el profesor selecciona una room, el flujo de datos primero se dirigirá a un fichero de servicio (myservice.ts) que encargará la comunicación entre la aplicación y la base de datos (Firebase) para realizar una petición del objeto de la room seleccionado por el usuario. Por otra parte, si la intención del profesor es crear una nueva room, el proceso es más o menos similar. Primero se dirigirá al page de createRoom y desde este page se solicitará una petición de creación de room al fichero de servicio. A continuación, el fichero de servicio enviará una petición al Firebase para que realiza la acción de crear una nueva room y finalmente se devolverá la nueva room.

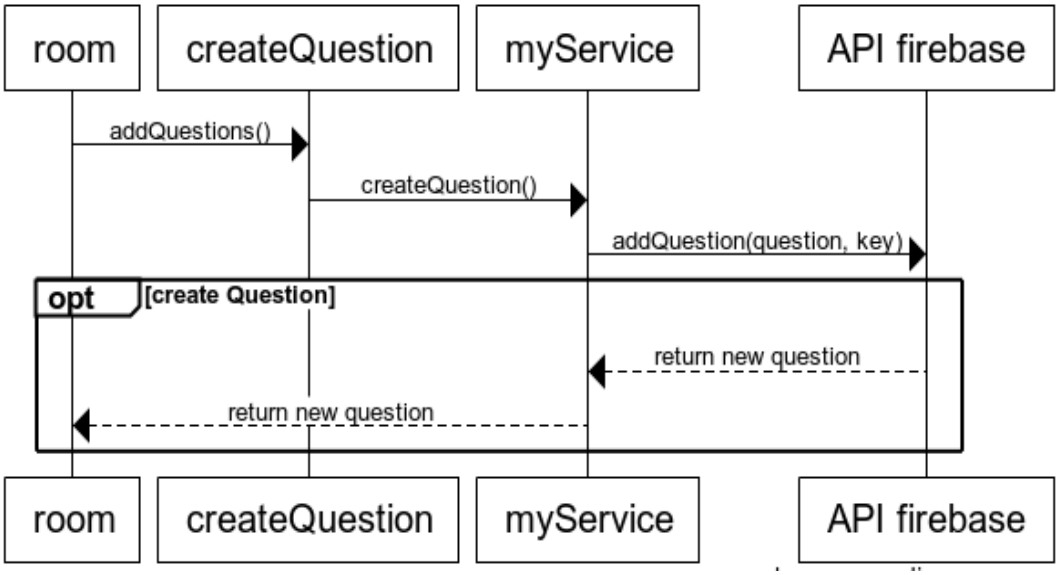

Imagen 17: añadir una pregunta o crear una encuesta nueva

Como podemos observar en la imagen 17, para la creación de cuestiones también tenemos que interactuar con el fichero de servicio situada en capa lógica que nos permitirá comunicar entre la aplicación y la base de datos. Tras pulsar el botón de crear question en el page createQuestion se enviará una petición de crear cuestión al fichero de servicio. En este caso, el fichero de servicio reenviara la petición a la API Firebase junto con dos parámetros, la cuestión que quiere crear el usuario y la clave de la room que va a situar la cuestión. La API Firebase se almacenará la nueva cuestión y devolverá una confirmación a la aplicación.

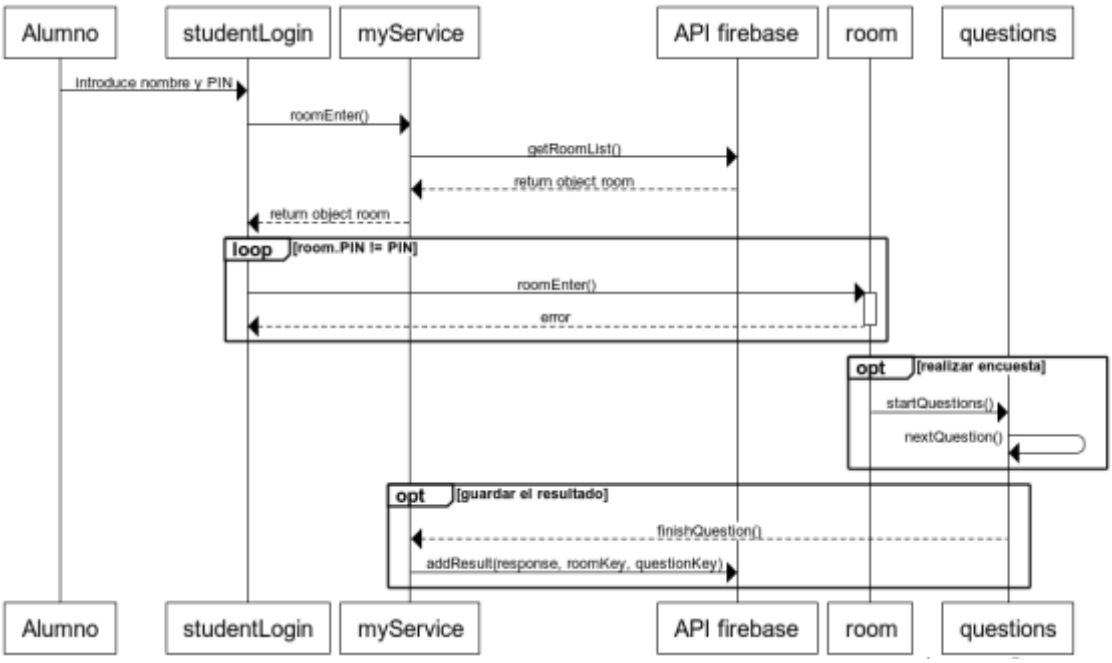

Imagen 18: realiza la encuesta y almacenar el resultado

Como podemos ver en la imagen 18 correspondiente al caso de uso de realizar la encuesta y almacenar el resultado, el alumno seleccionará directamente su nombre y el PIN que lo proporciona el profesor para entrar en la room que corresponde con el número de PIN sin la necesidad de registrarse una cuenta propia. El fichero de servicio encargará de enviar una petición de obtener la lista de room existente en la base de datos y API Firebase devolverá todas las rooms que tiene al fichero de servicio y studentLogin. En el studentLogin se encargará de recorrer mediante un bucle for para buscar aquella room en la que su PIN corresponde al PIN introducido por el alumno. Una vez encontramos la room correspondiente entrara en dicha room, sino que mostrara un mensaje de error. Cuando estamos dentro de una room, empezamos a realizar la encuesta haciendo un clic al botón empezar para enviar una petición de start question al page questions. Cada vez terminamos de responder a una cuestión pulsaremos el botón “Next” para dirigir a la siguiente cuestión. Una vez finalizamos la encuesta se enviará una confirmación al fichero de servicio y él se encargará de comunicar con la base de datos (Firebase) para almacenar el resultado de la encuesta.

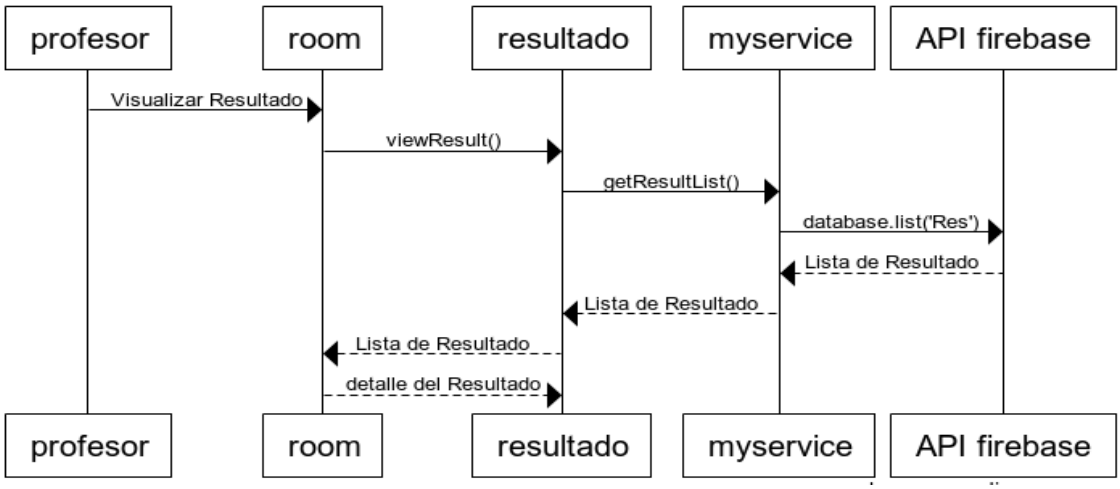

Imagen 19: Visualizar el resultado de la Encuesta.

Por último, como lo podemos ver en la imagen 19 que corresponde al caso de uso visualización del resultado de la encuesta. Un profesor para visualizar el resultado, como hemos mencionado en el capítulo anterior, tendría que ir a la pantalla de room y dar un clic al botón de "Ver Resultado". Según el dibujo de la imagen 18, cuando el profesor realizar la anterior acción, la clase room.ts activara el método *viewResult ()* y enviara una petición a la clase resultado (*resultado.ts*) y provider de servicio (*myservice.ts*) para pedir la información del resultado. El provider de servicio es una clase que encarga de realizar intercambio de informaciones con la base de datos Firebase, por lo que en este caso, enviara un método *getResultList ()* a la Firebase para pedir la lista de todo el resultado de la encuesta. Una vez cuando Firebase devuelve la lista del Resultado a la aplicación y llega a la clase room.ts, la clase room se encargaría de clasificar la lista y se envía la información del resultado correspondiente del grupo que ha elegido el usuario a la clase resultado. Finalmente, la clase resultado mostrara gráficamente todo el resultado del grupo que ha elegido por el usuario en la pantalla.

4.3.3 Capa de persistencia

En esta capa trataremos aspectos generales del diseño relacionados con la base de datos. Como ya hemos mencionado en las sesiones anteriores, la base de datos que estamos utilizando es la de Google Firebase. Esta base de datos permite una estructura no-relacional con almacenamiento de datos en tiempo real y los datos están en formato de tipo JSON diferente a las bases de datos relacionales tradicionalmente utilizadas.
Para entender un poco mejor el diseño de esta base de datos hemos detallado los cambios realizados sobre el modelo de diseño no-relacional del que nos hemos servido utilizando Firebase.

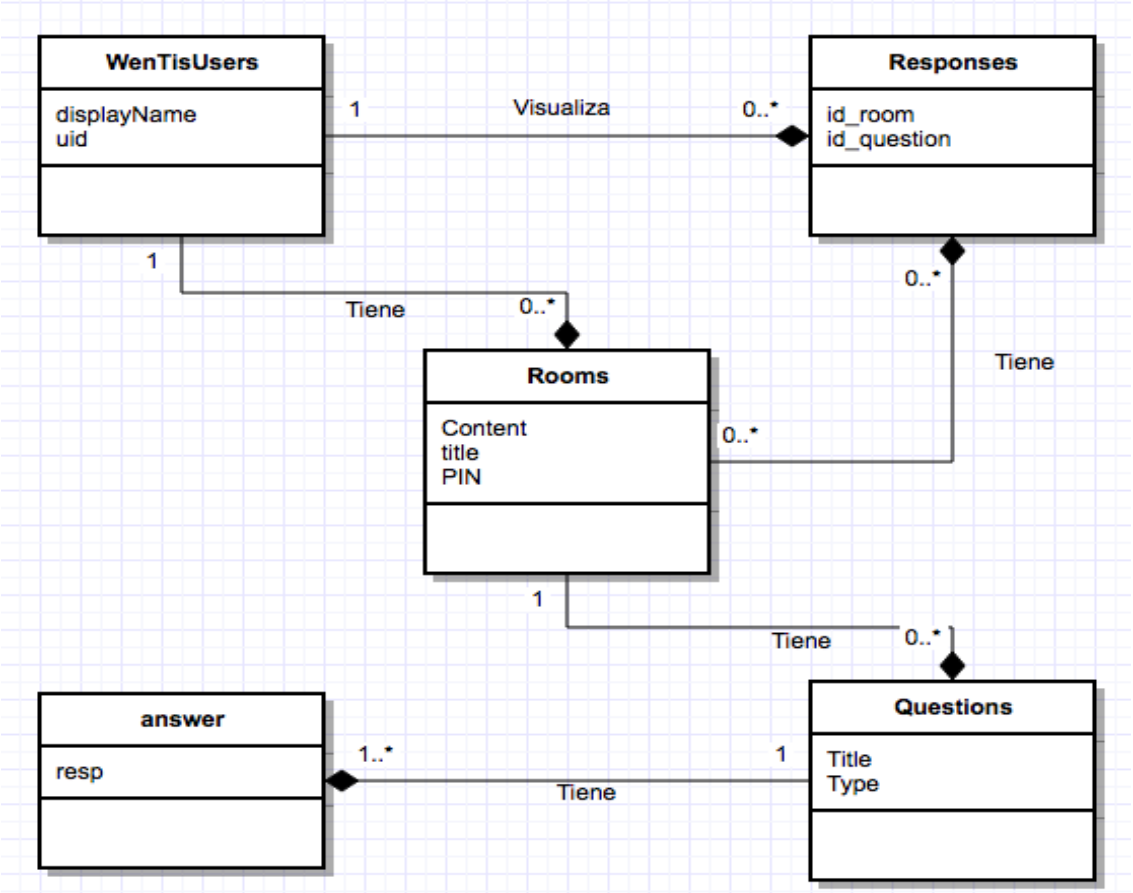

Imagen 20: diseño modelo entidad relación base de datos

la información que recoge la imagen 20 del esquema entidad relación se explica a continuación:

1. **Tabla WenTisUsers:** tabla de todos los usuarios que pertenecen a la modalidad profesor.
   - diaplayName: el nombre del usuario profesor.
   - Uid: la id único que corresponde a cada usuario.

2. **Tabla Rooms:** tabla que contiene toda la información de una encuesta.
   - Title: el título de la sala (Room).
   - Content: una breve descripción sobre la sala (Room).
   - PIN: el pin de dicha sala para luego utilizarlo por el usuario alumno.

3. **Tabla Questions:** tabla que contiene todas las preguntas de una encuesta.
   - Title: el enunciado de la pregunta.
   - Type: el tipo de la pregunta (Checkbox, radio, select…).
4. **Tabla Answer:** tabla que contiene todas las respuestas de una pregunta dentro de una encuesta.
   - Resp: es la respuesta de cada pregunta.

5. **Tabla Responses:** tabla que contiene todo el resultado de distintos rooms y distintas encuestas.
   - Id_room: es el id de la sala que esta el resultado.
   - Id_question: es el id de la pregunta que esta el resultado.

Como se trata de una base de datos orientada a documentos y está construido por objeto JSON, a continuación, en la imagen 201 vamos a ver un ejemplo de un esquema del JSON.

Imagen 21: Esquema JSON no relacional

En el ejemplo de la imagen 21 podemos observar como la información se agrupa por el uid de Firebase que corresponde a cada room existente en la base de datos y con sus atributos, y que a su vez contiene otro atributo "Questions" que contiene una lista de todas las preguntas de la encuesta de dicha room y las respuestas de cada pregunta. En este ejemplo mostrado existe una Room con una encuesta de tipo checkbox que tiene cuatro respuestas.

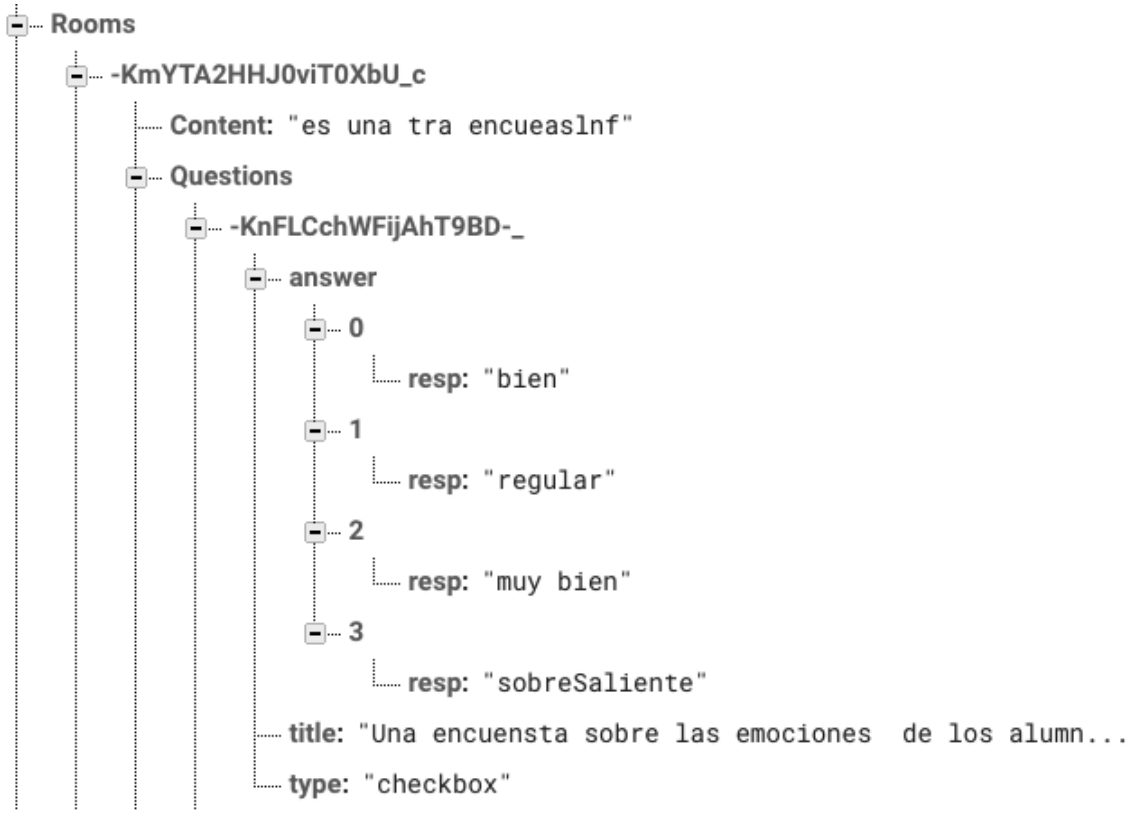

Imagen 22: Diseño esquema con Firebase

4.4 Conclusiones

En este capítulo hemos elaborado una introducción donde se detalla que es lo que se abarca en la fase de diseño de nuestro proyecto y una sección donde nos centramos en las ideas iniciales de inspiración. Sucesivamente hemos presentado el diagrama de esquema conceptual (Imagen 7) en el que presentamos los principales componentes de la aplicación y el flujo de datos que circula entre ellos. Para concluir este capítulo hemos elaborado una fase de especificación formal en la que hemos dividido por capas el diseño de la aplicación multiplataforma. En primer lugar, está la capa de presentación en la que se ha mostrado un diagrama de interfaz de la capa de presentación (imagen 8) más los diseños MockUps de la pantalla (imagen 9-14) con una explicación más detallado de las funciones que tiene cada pantalla. La capa lógica o de negocio en la que se ha establecido para cada caso de uso un diagrama de secuencia (imagen 15-19). Por último, la capa persistencia en la que hemos mostrado un diseño modelo entidad relación base de datos (imagen 20), un esquema de objeto JSON de la base de datos (imagen 21) y un ejemplo de esquema de una Room en Firebase (imagen 22).

5. Implementación

5.1 Introducción

Para el desarrollo de una aplicación multiplataforma ya hemos afirmado en los capítulos anteriores, existe una gran variedad de herramientas, tecnologías, lenguajes de programación o bibliotecas que den soporte a nuestras necesidades. Entonces es muy importante a la hora de hacer una buena elección de la herramienta que vamos a utilizar para desarrollar la aplicación. En este apartado de implementación vamos a abarcar todo lo relacionado a la codificación e implementación de la aplicación multiplataforma que hemos desarrollado. También mencionaremos la tecnología empleada mencionada en la sección 2.4 del capítulo 2, así como el entorno de desarrollo elegido para codificar nuestra aplicación.
En primer lugar, en este capítulo se estructura con un apartado de configuraciones generales previas a la codificación e implementación de la aplicación. A continuación, procederemos a visualizar la codificación de los diferentes casos de uso por separado expuestos en la imagen 8 del Capítulo 3. Para una mejor visualización y asimilación del resultado se incluirán imágenes reales de nuestra aplicación multiplataforma que se podrán compara con los ejemplos de MockUps en la sección 4.3 del capítulo 4, así como unos fragmentos de código relevante en nuestra aplicación. Por último, se concluye con una fase de testing en la que se comprueba que la aplicación proporciona garantías de funcionamiento.

5.2 Conceptos generales previos

5.2.1 Creación de una App multiplataforma con Ionic 2 (Angular 2)

Antes de comenzar a codificar la aplicación es muy importante tener en cuenta la herramienta y el lenguaje de programación que vamos a utilizar en la aplicación. En primer lugar, tenemos que establecer el entorno de desarrollo de Ionic 2 en nuestro ordenador, entramos en la siguiente URL: https://ionicframework.com/getting-started/ y seguiremos los pasos para instalar el entorno y preparar la primera prueba de ejecución del Ionic 2. También es necesario instalar el paquete de lenguaje de programación TypeScript, ya que en Ionic 2 se utiliza dicho lenguaje para la codificación.
En los siguientes apartados de implementación se explicará con detalle como implementa la aplicación con la herramienta Ionic.

5.2.2 Bibliotecas (módulos) utilizadas en la aplicación multiplataforma

Antes de realizar el presente proyecto, necesitamos una serie de bibliotecas (en nuestro caso módulos) que nos darán soporte para poder utilizar las herramientas o las tecnologías que vamos a emplear para su desarrollo. Nuestra herramienta de entorno de desarrollo Ionic 2 por defecto ya se vincula con los módulos como *angular, ionic, ionic-native y type*, ya que Ionic 2 está basado en Angular 2 y utiliza el lenguaje de programación typeScript. Para ellos, en la parte superior de cada fichero de la aplicación es necesario importar estos módulos y módulos adicionales dependiendo de la funcionalidad que va a

emplear cada fichero de código. A continuación, vamos a mostrar una serie de tablas de los tipos de módulos adicionales que hemos importado en nuestra aplicación multiplataforma. Se especificarán en tres campos que son los siguientes:

Nombre: Nombre del módulo que vamos a importar
Código de modulo: código que hay que añadir en la aplicación para utilizarlo.
Descripción: Breve resumen de lo que hace esta biblioteca y para que la vamos a utilizar.

Nombre	Firebase
Código de módulo	import * as firebase from 'firebase'; import {AngularFireModule} from 'angularfire2';
Descripción	Módulo que permite utilizar todas las funcionalidades del Firebase.

Tabla 7: módulo de Firebase

Nombre	ChartsModule
Código de módulo	import {ChartsModule} from 'ng2-charts/charts/charts';
Descripción	Módulo que permite dibujar distintos tipos de gráficas para presentar la estadística del resultado de la encuesta.

Tabla 8: módulo de ChartsModule

5.2.3 Conexión con la base de datos (Firebase de Google)

Para utilizar el Firebase es necesario tener una cuenta de Google, para ello accedemos a la siguiente página: https://firebase.google.com/ y elegimos la opción de "crear un nuevo proyecto". Después de realizar todos los pasos de creación de un proyecto se tendrá que elegir la opción "añadir Firebase a tu aplicación web" y saldrá un fragmento de código que tenemos copiarlo y pegarlo en nuestra aplicación.

Imagen 23: fragmento de código de Firebase

Todos estos pasos explicados de forma resumida se detallan con mayor precisión en la página web mencionada anteriormente en la pestaña de documentación y también se detallarán en el capítulo implementación el uso concreto de Firebase en nuestra aplicación multiplataforma.

5.3 Implementación

En este capítulo vamos a mostrar el proceso de codificación de la aplicación multiplataforma. Para ello, hemos optado por separar los casos de uso expuestos en la imagen 6 (capitulo 3), por lo que habrá una explicación de distintos casos de uso como iniciar sesión o registrarse una cuenta, visualizar una sala (Room) o creación de una nueva sala, creación de una nueva pregunta o encuesta y realizar una encuesta y la consulta de la estadística del resultado de la encuesta.
Por último, también mostraremos la implementación de la parte del usuario alumno que corresponde los casos de uso como: introducir nombre y PIN y realizar encuesta.

5.3.1 Implementación de inicio de sesión y registrarse una cuenta

Para empezar, para poder implementar este caso de uso necesitamos tener correctamente importado el módulo Firebase (Tabla 7). En lo que respecta a la codificación de este caso de uso, necesitaremos utilizar la función de autenticación de Firebase para realizar el inicio de sesión. Para ello, implementamos un provider de autenticación (*auth.ts*) para incluir los métodos como inicio de sesión y registrarse una cuenta de Firebase. En este caso, implementamos el método *loginUser*.

```
loginUser(email: string, password: string){
  var promise = new Promise((resolve, reject) => {
    firebase.auth().signInWithEmailAndPassword(email, password).then(() => {
      resolve(true);
    }).catch((err) => {
      reject(err);
     })
  })

  return promise;

}
```

Código 1: método para llamar la función de autenticación de Firebase.

Como lo podemos ver en el Código 1, en Firebase existe una función llamada *signInWithEmailAndPassword ()* que automáticamente se encarga de realizar el inicio de sesión. Posteriormente, simplemente desde la clase *login.ts* llamamos directamente dicho método loginUser mediante la clase de provider de autenticación que hemos mencionado anteriormente. Si el usuario es válido, mostraremos un mensaje de éxito y pasamos al HomePage, si no se mostrara un mensaje de error.

```
login () {
        this.showLoading();
        this.authservice.loginUser(this.email, this.password).then((res: any) {
                if (!res.code){
        this.loading.dismiss();
        this.navCtrl.setRoot(HomePage);
        }else
    alert(res);
  })
}
```

Código 2: el método login ()

Por otro lado, si acaso el usuario no tiene cuenta, se pasará a registrar una cuenta en el page register (register.ts). Para ello, necesitamos implementar un método que realiza la función de registro, como en el caso de inicio de sesión, Firebase también existe una función que realiza automaticamente la registración del usuario:

```
signupUser(newuser){
    var promise = new Promise((resolve, reject) => {
      firebase.auth().createUserWithEmailAndPassword(newuser.email, newuser.password).then(() => {
        firebase.auth().currentUser.updateProfile({
          displayName: newuser.userName,
```

Código 3: la función de registro de un usuario en Firebase

En este caso, igual que el inicio de sesión, dicha función también esta implementada en el provider de autenticación, en el que posteriormente se llamara en la clase de register.ts para realizar correctamente la función de registración.

```
loader.present();
this.authservice.signupUser(this.newuser).then((res: any) => {
  loader.dismiss();
  if (res.success)
    this.navCtrl.setRoot(HomePage);
  else
    alert('Error' + res);
})
```

Código 4: llamada del método signupUser en la clase register.ts.

De esta forma tendremos implementado nuestro inicio de sesión y registración de usuario (imagen 26).

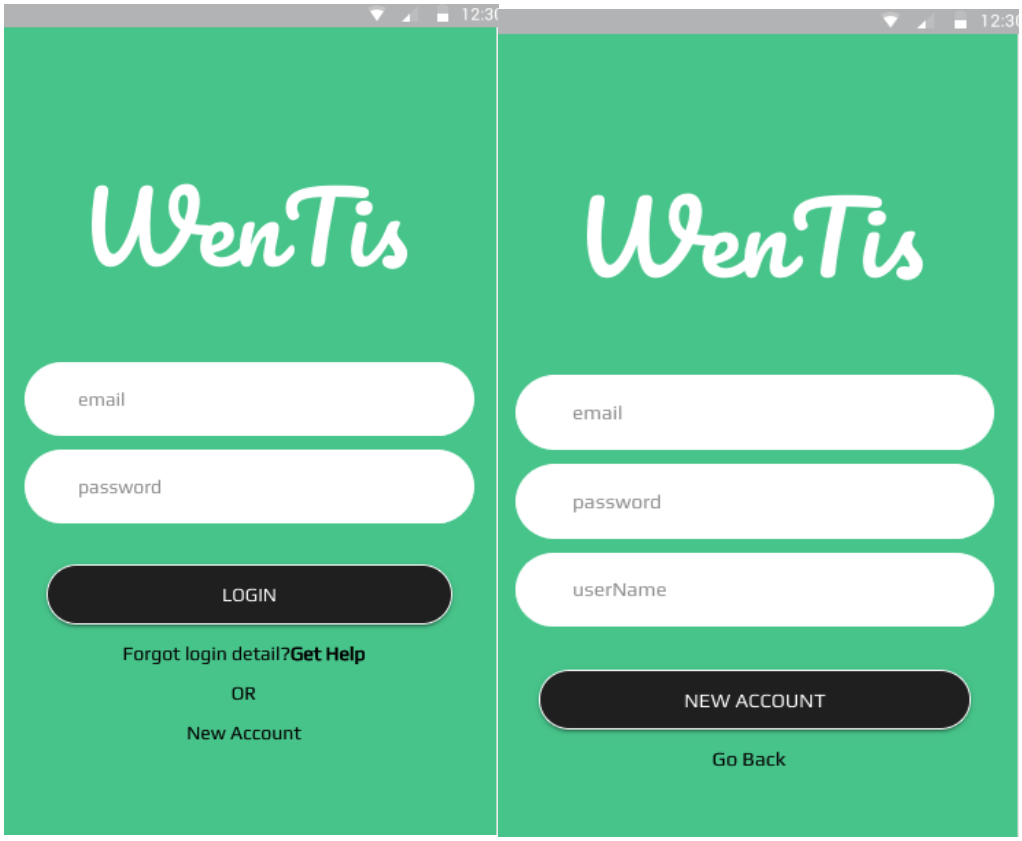

Imagen 24: Pantalla de inicio de sesión y registro codificadas

5.3.2 Implementación de visualización y creación de una sala

En primer lugar, para la sala de encuesta tendremos una lista de sala ya creadas en la base de datos de Firebase que nos permitirá a través de un método de Firebase obtener dicha lista de sala en nuestra aplicación y realizar cualquier operación sobre ella. Para ello, hemos implementado otro provider de servicio (myservice.ts) que encarga de realizar todas las comunicaciones con Firebase y por supuesto hemos incluido dicho método en ese provider. Vamos a visualizar el siguiente código:

```
getRoomList(){

    this.roomList =this.af.database.list('Rooms') as FirebaseListObservable <any>;
    return this.roomList;
}
```

Código 5: método para obtener la lista de sala en Firebase

Como lo podemos observar en el código 5, a través de Firebase podremos llamar directamente la función database.list() para obtener una lista que esta almacenada en Firebase. Una vez obtenemos la lista podemos implementar cualesquiera funciones sobre ella. En nuestro caso primero hemos llamado el método getRoomList () en la clase Home mediante el provider de servicio (Código 6) y después hemos implementado una iteración sobre ella en el fichero HTML (home.html) para mostrarla en la pantalla todas aquellas salas existentes (Código 7). Vamos a observar el siguiente código:

```
this._myservice.getRoomList().subscribe(Rooms => {
    this.datos_recuperados=Rooms
```

Codigo 6: código para obtener la lista en la clase home.ts.

```
<ion-item *ngFor="let item of datos_recuperados"  (click)='viewRoom(item.pin)'>
  <ion-thumbnail item-left>
    <img style="width:100%" class="thumbnail" src="item.image">
  </ion-thumbnail>
  <h2>{{item.Title}}</h2>
  <p>{{item.Content}}</p>
```

Código 7: código HTML5 de iteración de la lista de sala (home.html).

Como lo podemos visualizar en el código 7, hemos utilizado una iteración de ng-model (una función de Angular 2) para realizar un bucle con la lista de sala. De ahí, asignamos cada objeto de la bucle a otra variable llamada ítem que posteriormente lo utilizamos para mostrar elemento de la sala, en este caso, la imagen (ítem. image), el título (ítem.Title) y el contenido (ítem.Content).

Por otra parte, si quisiéramos crear y añadir una sala nueva ya tendremos que ir a la clase de CreateRooms.ts. para ello, necesitamos implementar un método donde podrá realizar una función "put" a la Firebse. En la plataforma de Firebase existe varios métodos que puede subir elementos a la base de datos, entre ellos, hemos elegido la función "push ()" de Firebase. Porque a través de la función push () podemos subir un objeto entero a la Firebase. Igual que en los casos anteriores, en la clase myservice.ts hemos implementado un método que realiza la función "put" mediante el método push ()

.

```
let rooms = firebase.database().ref('Rooms');
return rooms.push(room);
```

Código 8: la función push () para subir sala a la Firebase.

Una vez en la clase myservice.ts implementamos el método addRoom () que incluye la función push (), vamos a la clase CreateRooms.ts para subir correctamente la información de la nueva sala. Para ello, obtenemos todas las informaciones que ha escrito por el usuario sobre la nueva sala y las asignaremos a distintas variables y empaquetamos en un objeto JSON (ya que Firebase solo utiliza almacenamiento JSON). Finalmente, subimos el JSON mediante el método createRoom ().

```
createRoom(){
  let room = {
    Title: this.title,
    pin: this.idRoom,
    Content: this.descripcion,
    open: true,
  }


  this._myservice.addRoom(room);
```

Código 9: método *createRoom ()* de la clase createRooms.ts

De esta forma tendremos implementado la parte de visualización y creación de sala.

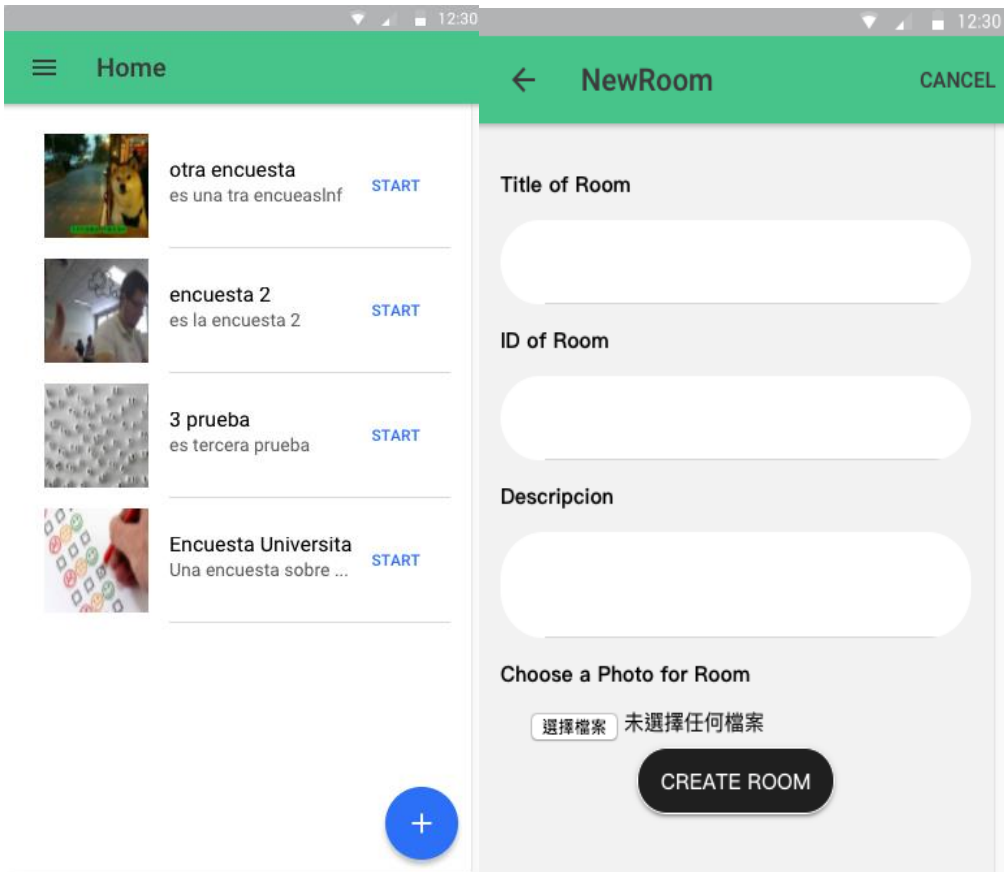

Imagen 25: pantalla de home y createRooms

5.3.3 Implementación de creación de pregunta o encuesta

Procederemos a la parte de creación de preguntas y encuestas referente a la satisfacción y al estado emocional. Para ello, hemos implementado en la pantalla de room un botón que incluye un método que realiza la función "*push ()*" en la que se sube a la Firebase una

fórmula de los datos de una pregunta escrito por el usuario. Los códigos que hemos utilizado son similares que los de crear una sala nueva, vamos a ver los siguientes códigos:

```
addQuestion(question, key: any){
    let questions = firebase.database().ref('Rooms/'+key+'/Questions');
    return questions.push(question);
}
```

Código 10: la referencia de Questions en Firebase

Como lo podemos observar en el código 10, es similar al método *addRoom* que hemos mencionado anteriormente. Pero en este caso, la referencia de Firebase es "Rooms/key/questions", es decir, hemos depositado la localidad de las preguntas mediante una estructura organizada por la key de room y un array questions donde están todas las preguntas. La finalidad de esta organización de estructura simplemente es facilitar la lectura de los datos para otros casos de uso. Una vez cuando implementamos el método *addQuestions ()*, ahora es simplemente llamarlo en la clase *createQuestion.ts* para realizar correctamente el "update" de la cuestión.

```
createQuestion(){
  //el JSON de la cuestion
  let question ={
    type: this.typeQuestion,
    title: this.question,
    answer: this.options
  }
  let questionSelect={
    type: this.typeQuestion,
    title: this.question,
    answer: this.options,
    answerOptions: this.option
  }
    //Para el caso cuando la cuestion es de tipo select
    if(this.typeQuestion=="select"){
      this._myservice.addQuestion(questionSelect, this.key);
      this.navCtrl.pop();
    }else{
    this._myservice.addQuestion(question, this.key);
    this.navCtrl.pop();
    }
```

código 11: el método para subir la pregunta a la Firebase.

por último, invocamos al método que nos permite enviar la fórmula de una cuestión a la Firebase llamando al método *addQuestion* con los parámetros como un JSON de la cuestión y la clave de la room que se sitúa la cuestión. Si todo sale bien, en la Firebase saldrá una estructura como se muestra en la imagen 24 del capítulo 4.

De esta forma obtendremos la pantalla de la imagen 28, de la cual nos basamos en el diseño del mockUps del capítulo 4 de la capa de presentación (Imagen 14).

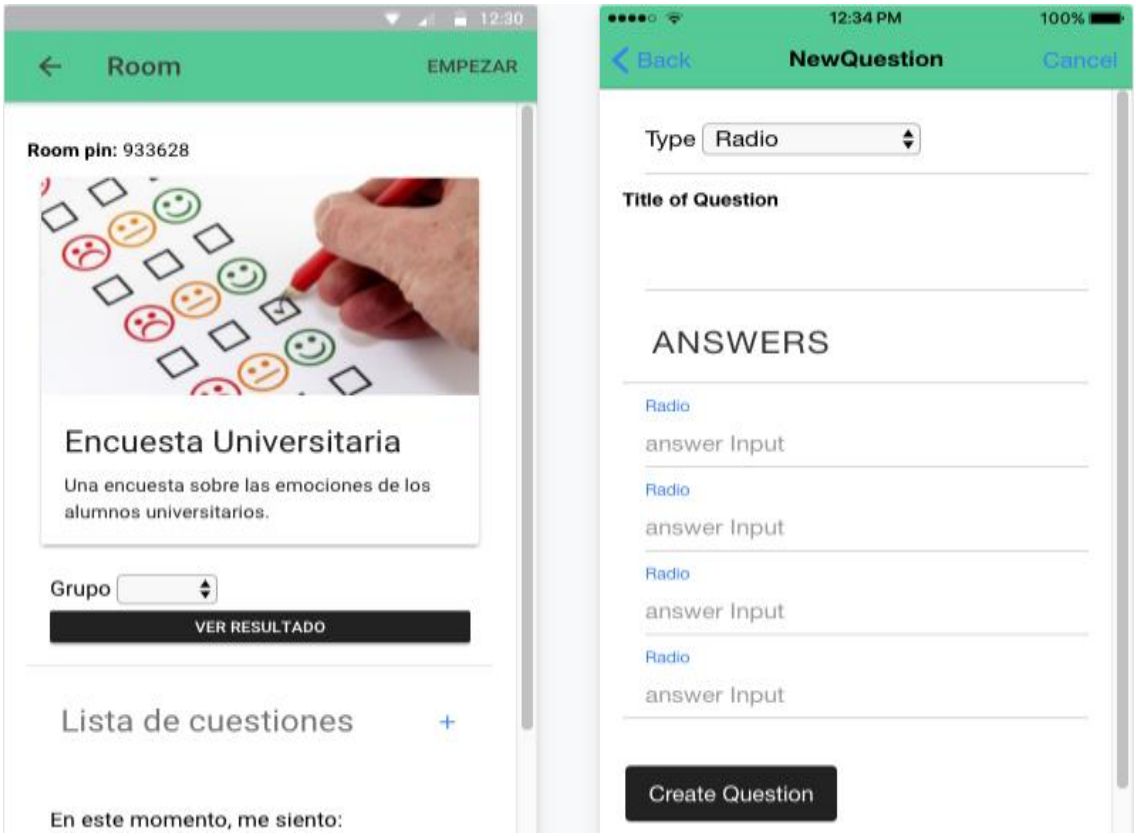

Imagen 26: pantalla de creación del cuestionario.

5.3.4 Implementación de realizar encuesta y almacenar resultado

Una vez terminamos de implementar la parte de room y la parte de cuestionario, ahora llega la parte más importante de nuestra aplicación multiplataforma. Podemos decir que es la esencia de nuestra aplicación, porque la función principal de la aplicación es medir las emociones de la clase mediante un cuestionario o una encuesta.
Siguiendo los pasos explicados en el esquema de secuencia de la sección 4.3.2, el usuario que realiza el cuestionario es el usuario de tipo alumno, así que vamos a implementar desde la vista del usuario alumno.
El usuario alumno, como ya hemos mencionado en los capítulos anteriores, es el único usuario que tiene que tener una cuenta es el usuario de tipo profesor. En el caso del usuario de tipo alumno no es necesario registrarse. Por lo que, en la interfaz del login del usuario alumno simplemente implementamos un select y un input para que los alumnos seleccionan su grupo o clase e introduzcan el número de PIN que corresponderá a una determinada habitación (room), y finalmente, un botón "entrar" para entrar en la habitación. A continuación, vamos a mostrar el código de selección de habitación mediante el número PIN.

```
roomEnter(){

  for ( var p in this.room_datos ){

    if(this.room_datos[p].pin == this.pin && this.room_datos[p].open==true){

      console.log("encontramos pin igual en elemento "+p+" y recuperamos "+this.room_datos[p].pin);
      this.nav.push(roomPage, {
        room_ this: this room_datos[p],
        key: this.room_datos[p].$key,
        grupo: this.grupo
      });
    }
```

código 12: el método *roomEnter ()* de login de alumnos.

Como lo podemos observar en el código 12, mediante el bucle *for* se iterará la lista de la room y a través de un *if* se compara el PIN de habitación de cada iteración con el PIN introducido por el usuario alumno, si son iguales entrara a dicha habitación. De esta forma, tendremos las siguientes pantallas:

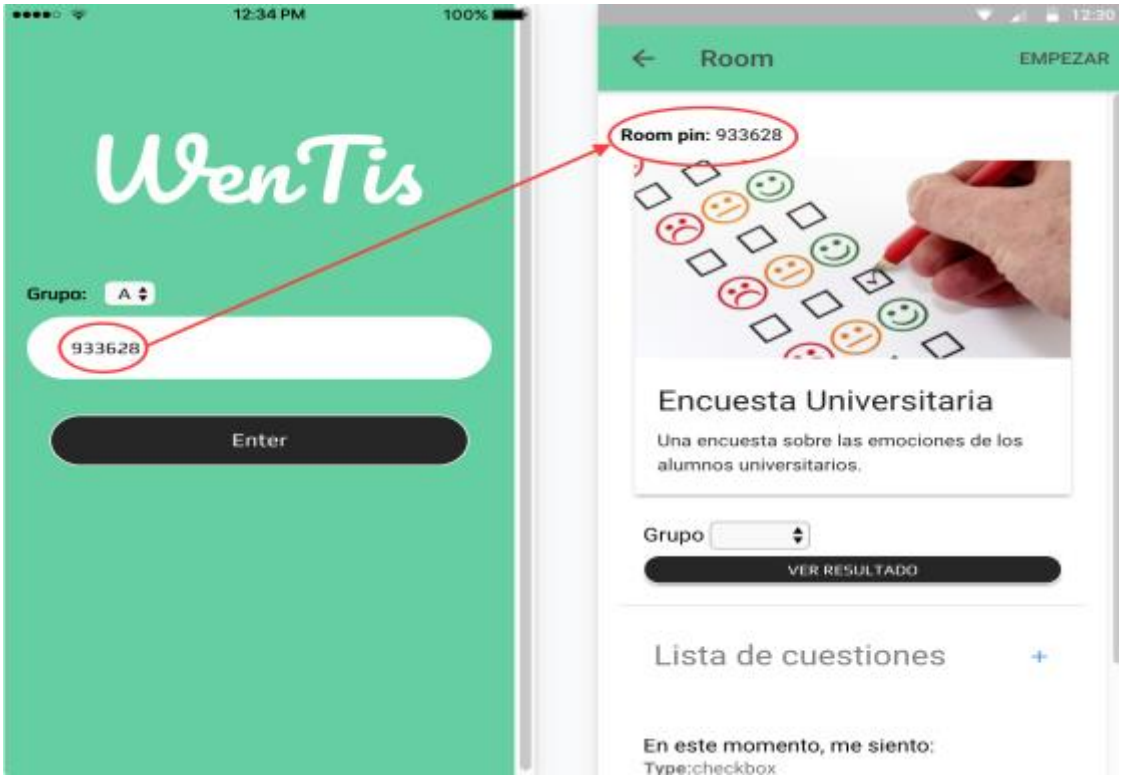

Imagen 27: Login del usuario de tipo alumnos.

Una vez estamos dentro de la habitación, pulsamos el botón "Empezar" que está situado en la parte superior derecha y nos dirigirá hacia a la pantalla de la encuesta para comenzar a realizar el cuestionario. Para ello, hemos utilizado la misma metodología que los casos de uso anteriormente mencionado, llamando al método *getQuestionList()* de *myservice.ts* para recuperar la lista de todas las questiones y a continuación, a través del método *startQuestion()* se dirige hacia la pantalla de cuestionario pasándola los parámetros como el room_datos (la lista de todas las habitaciones), questions (la lista de todas las cuestiones), la key (la clave de la habitaciones que pertenece el cuestionario), el id (una variable entera que se utiliza para avanzar la pregunta) y por último, el grupo (el grupo del usuario alumno).

```
//obtenemos la lista de la question con el key actual
this._myservice.getQuestionList(this.key).subscribe(pregunta => { this.questions = pregunta });
//empieza a realizar la encuesta
startQuestions() {
   var id = 0;
   this.checkbox = this.questions[id];
   this.nav.push(questions, {
      room_datos: this.room_datos,
      checkboxQuestion: this.questions,
      key: this.key,
      index: id,
      grupo: this.grupo

   });
```

código 13: el método *getQuestionList ()* y *startQuestions()*

Posteriormente, para implementar correctamente según el diagrama de secuencia, es necesario iterar una a una las cuestiones, porque hemos decidido que cada vez en la pantalla solo puede mostrar la información de solo una pregunta. Entonces, hemos utilizado una variable temporal id que ya hemos mencionado anteriormente, en la que cada vez un usuario alumno se finaliza de contestar una pregunta, la variable id se incrementa en una y se pasa como parámetro juntos con los demás parámetros a la siguiente pantalla. Para ello, hemos implementado un botón "Siguiente" enlazado con un método llamado "*nextQuestion ()*" que incluye todas las funcionalidades que hemos mencionado anteriormente. A parte del incremento de la variable id, dicho método también se encarga de subir el resultado de la pregunta a la Firebase. Para conseguir eso, igual que los casos anteriores, también hemos implementado un método llamado *addResult ()* para subir el resultado cada vez que el usuario pulsa el botón "Siguiente" (código 14).

```
if(this.checkbox[this.index].type=="checkbox" || this.checkbox[this.index].type=="radio"){
this._myservice.addResult(response, this.key, this.grupo);
}else
this._myservice.addResult(res_input, this.key, this.grupo);
// pasamos a la siguiente cuestion
let j = this.index +1;
    this.nav.push(questions,{
    room_datos: this.roomDatos,
    checkboxQuestion: this.checkbox,
    key: this.key,
    index: j,
    grupo: this.grupo

  });
```

Código 14: el método *nextQuestions ()* para avanzar a la siguiente cuestión

Por último, una vez cuando llegamos a la última cuestión de toda la encuesta, el botón "siguiente" se convertirá en otro botón llamado "Finalizar" para finalizar la encuesta. Para conseguir eso, hemos implementado los siguientes códigos (código 15 y 16).

```
<button *ngIf="index < checkbox.length-1" ion-item icon-right color="danger" (click)="nextQuestion($event)" >
Siguiente
</button>
<button *ngIf="index == checkbox.length-1" ion-item icon-right color="danger" (click)="finishQuestion($event)" >
Finalizar
</button>
```

Código 15: código HTML de botón para comprueba el length de la encuesta.

Como lo podemos observar en el código 15, hemos implementado un *ng-If* para comprueba cuando llega la última pregunta de la encuesta. Si el índex es inferior al checbox.length-1 aparecerá el botón "Siguiente", al contrario, aparecerá el botón "Finalizar".

```
// para finalizar la encuesta.
this.showAlert();
if(this.checkbox[this.index].type=="checkbox" || this.checkbox[this.index].type=="radio"){
this._myservice.addResult(response, this.key, this.grupo);
}else
this._myservice.addResult(res_input, this.key, this.grupo);
   this.nav.setRoot(studentLoginPage,{
        room_datos: this.roomDatos,
        key: this.key,
        grupo: this.grupo
    });
    console.log("finalizamos");
```

Código 16: código para finalizar la encuesta.

Como lo podemos ver en el código 16, la mayoría implementación es igual que la de *nextQuestion ()* y también se realiza la operación de subir el resultado a la Firebase. Pero en este caso, como es para finalizar la encuesta, el método *setRoot* se dirigirá al login de los alumnos y se acaba con un mensaje de aviso diciendo que has finalizado la encuesta (*this.showAlert()*).

De esta forma, con todos los que hemos mencionado anteriormente, obtendremos la pantalla de la imagen 30 que corresponde al diseño de mockUps del capítulo 4.

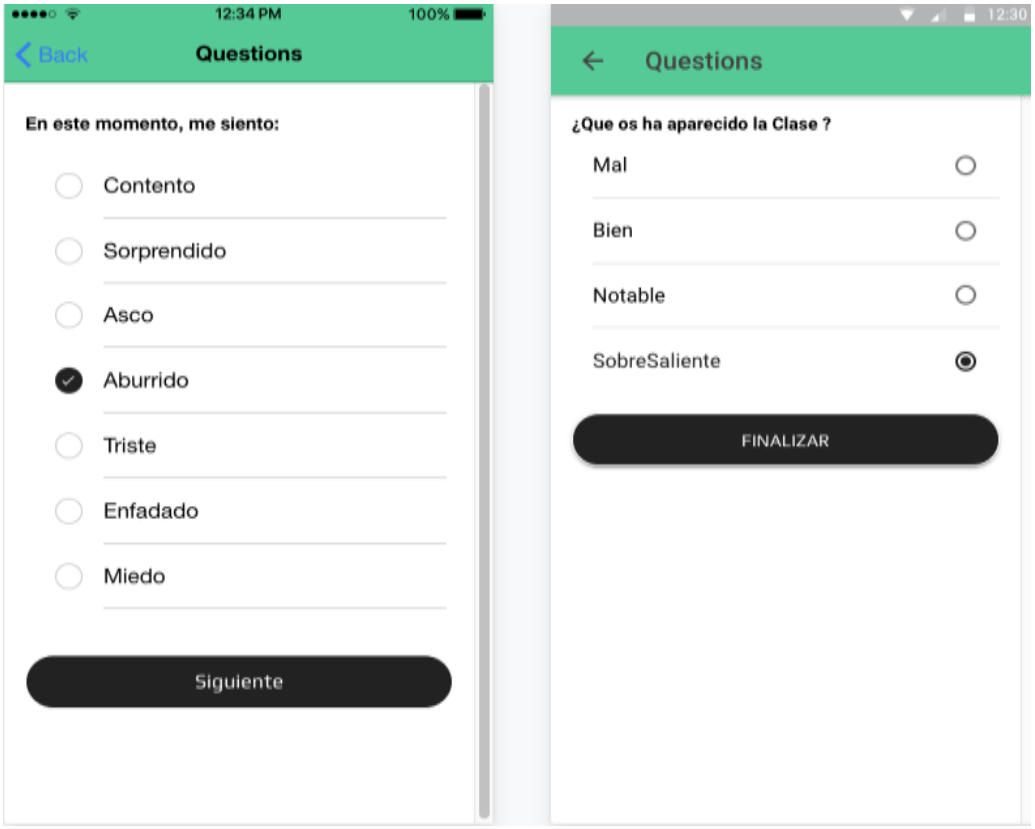

Imagen 28: la pantalla de la encuesta.

5.3.5 Implementación de consultas y mostrar estadísticas

Para implementar este diagrama de secuencia es necesario previamente estar conectado a la base de datos de Firebase y tener conexión a Internet para poder acceder a ella.

Siguiendo los pasos explicados en la sección 5.2.3, conectamos la base de datos de una forma igual que otros casos anteriores, mediante codificación crearemos una referencia que será el grupo seleccionado por el usuario en la pantalla de habitación. Entonces, obtendremos una lista de todas aquellas respuestas respondidas por el usuario que pertenece a ese grupo (código 17).

```
this._myservice.getResultList(this.key, this.grupoClase).subscribe(result => {
  this.resultado = result
```

Código 17: crear referencia para obtener la lista del grupo.

Posteriormente se implementará la consulta a la base de datos para obtener las respuestas disponibles que hay para un grupo determinado (código 18).

```
var resulta = [];
for (var p in this.resultado) {
    resulta.push(this.resultado[p].result);
}
```

Código18: código para recoger todas las respuestas de un grupo.

En esta aplicación, para la cuestión de medir emociones hemos propuesto siete respuestas posibles que son: Contento, Sorprendido, Aburrido, Triste, Asco, Enfadado y Miedo. Entonces para recoger todas las respuestas de un grupo, hemos implementado una variable array "*resulta []*" para guardar todas las respuestas. A continuación, iteraremos el array "*resulta []*" para obtener el número total de cada respuesta mediante condiciones *if*.

```
for(var p in resulta){
  if (resulta[p] == "Contento") cont++;
  if (resulta[p] == "Sorprendido") sor++;
  if (resulta[p] == "Aburrido") abur++;
  if (resulta[p] == "Triste") tri++;
  if (resulta[p] == "Asco") asc++;
  if (resulta[p] == "Enfadado") enf++;
  if (resulta[p] == "Miedo") mie++;
  if (resulta[p] == "Mal") mal++;
  if (resulta[p] == "Bien") bien++;
  if (resulta[p] == "Notable") notable++;
  if (resulta[p] == "SobreSaliente") sobre++;
}
let totalQuestion1 = cont + sor + abur + tri + asc + enf + mie;
let totalQuestion2 = mal + bien + notable + sobre;
```

Código 19: código para obtener el número de respuestas.

Una vez obtendremos todos los datos necesarios pasamos a mostrar gráficamente las estadísticas de las respuestas. Para ello, primero con la función "push" de la navegación pasamos a la pantalla de resultado con todos los parámetros que hemos obtenido en el código 18 y 19, y a través de estos parámetros construimos la gráfica.

Para el caso de gráfica, hemos decidido utilizar la herramienta ng-Charts [3] debido a su buena compatibilidad con la herramienta Ionic 2 y Angular 2. Es una herramienta muy bonita y facilidad de uso, y está basado en Chart.js. para implementarla, simplemente hemos implementado tres variables e inicializarlas en el constructor con todos los datos necesarios, y luego implementar una referencia a estas tres variables en el código HTML (código 20 y 21).

```
//grafica doughnutChart de emociones
this.doughnutChartLabels = ['Contento', 'Sorprendido', 'Aburrido', 'Triste','Asco','Enfadado','Miedo'];
this.doughnutChartData = [this.cont, this.sor, this.abur, this.tri, this.asc, this.enf, this.mie];
this.doughnutChartType = 'doughnut';

//grafica de donutchart de question 2
this.doughnutChartLabels2 = ['Mal', 'Bien', 'Notable', 'SobreSaliente'];
this.doughnutChartData2 = [this.mal, this.bien, this.notable,this.sobre];
this.doughnutChartType2 = 'doughnut';
```

Código 20: código ng-Charts para la presentación gráfica.

```
<div style="display: block">
    <canvas baseChart
            [data]="doughnutChartData"
            [labels]="doughnutChartLabels"
            [chartType]="doughnutChartType"
            (chartHover)="chartHovered($event)"
            (chartClick)="chartClicked($event)"></canvas>
</div>
```

Código 21: código HTML de ng-Charts.

También hemos implementado un botón que tiene la función de cambiar la vista de gráfica. Es decir, si pulsamos el botón se cambiará desde la gráfica de tipo donut Chart a grafica de tipo pie Chart. La finalidad simplemente es para ver el resultado en otra forma distinta y además, pensamos que es interesante implementarla (código 22).

```
public randomizeType():void {
  this.doughnutChartType = this.doughnutChartType === 'doughnut' ? 'pie' : 'doughnut';
}
```

Código 22: método para cambiar la vista de gráfica.

Para finalizar, también hemos implementado la presentación gráficamente de los resultados a nivel global. Es decir, anteriormente solo hemos implementado la presentación del resultado de un solo grupo, ahora vamos a implementar la visualización del resultado de todos los grupos. Para ello, es casi igual que la implementación de solo un grupo, la diferencia simplemente es, hemos implementado otra vista diferente al de solo un grupo llamada radar Chart, en la que nos facilita visualizar la diferencia de numero entre distintas respuestas. La implementación es la siguiente:

```
//grafica de RadarChart de emociones
this.radarChartLabels = ['Contento', 'Sorprendido', 'Aburrido', 'Triste','Asco','Enfadado','Miedo'];
 this.radarChartData = [
    {data: [this.contA, this.sorA, this.aburA, this.triA, this.ascA, this.enfA, this.mieA], label: 'Grupo A'},
    {data: [this.contB, this.sorB, this.aburB, this.triB, this.ascB, this.enfB, this.mieB], label: 'Grupo B'},
    {data: [this.contC, this.sorC, this.aburC, this.triC, this.ascC, this.enfC, this.mieC], label: 'Grupo C'},
    {data: [this.contD, this.sorD, this.aburD, this.triD, this.ascD, this.enfD, this.mieD], label: 'Grupo D'},
    {data: [this.contE, this.sorE, this.aburE, this.triE, this.ascE, this.enfE, this.mieE], label: 'Grupo E'},
    {data: [this.contF, this.sorF, this.aburF, this.triF, this.ascF, this.enfF, this.mieF], label: 'Grupo F'},
 ];
this.radarChartType = 'radar';
```

Código 23: código para la presentación de rada Charts.

Como lo podemos ver en el código 23, a cada grupo tiene un array de distintas respuestas. Entre ellos se construye la gráfica radar Charts y las demás gráficas.

Así podemos dar por finalizada esta funcionalidad en la que la mayor complicación es trabajar con la base de datos, y procederemos a visualizar la interfaz gráfica real. Como en las anteriores pantallas, este diseño de la interfaz también se corresponde con los mockUps (Imagen 9-14) presentados en la sección 4.3.1.

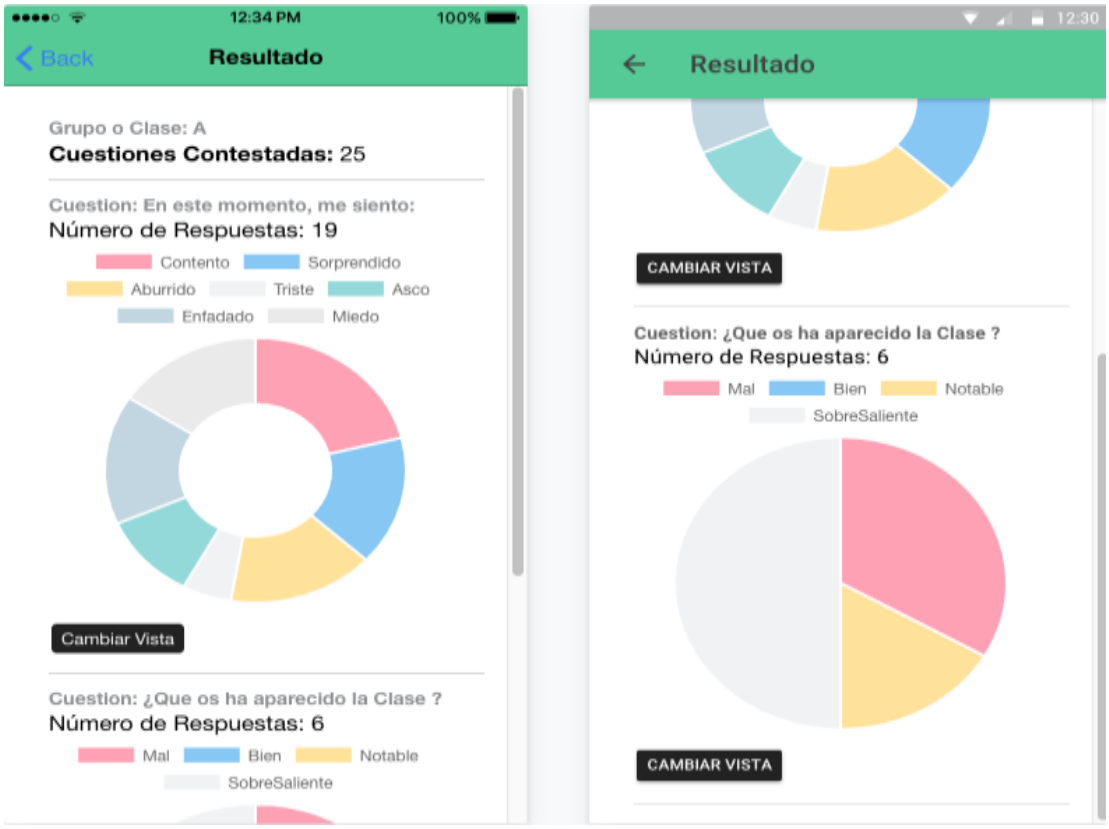

Imagen 29: gráfica real de solo un grupo.

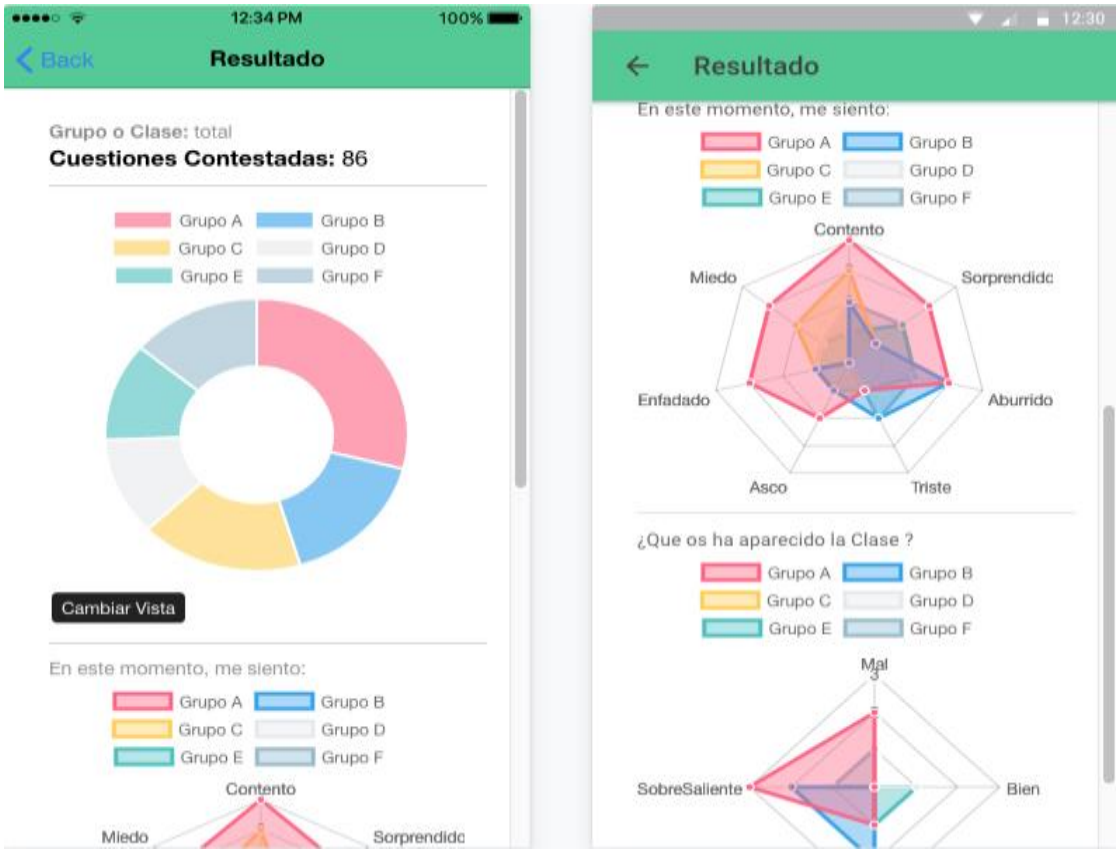

Imagen 30: gráfica real de todos los grupos.

5.3 Implantación

En este apartado vamos a explicar los requisitos necesarios para la puesta en marcha de nuestra aplicación multiplataforma tanto para los usuarios de tipo profesor y de tipo alumno.

Como la aplicación es multiplataforma, para poder utilizarla, los usuarios pueden utilizar cualquier tipo de dispositivos, teléfono móvil con cualquier sistema operativos, ordenador o Tablet, y luego instalar la aplicación en estos dispositivos. Gracias a la herramienta Ionic 2, podemos generar paquete de instalación como APK de Android (Aplication Package File) para Android y IPA para IOS (iphone e Ipad). Posteriormente, para instalarla de forma correcta, tendremos que activar en el menú de configuración del teléfono móvil la opción: "Instalar aplicaciones de origen desconocido" en el caso de Andriod y en el caso de IOS tenemos que ir al "Ajuste-General-Perfil" y encontrar el perfil de nuestra aplicación y verificarla.

Una vez efectuadas todas estas operaciones, la aplicación quedará instalada en el dispositivo móvil y se podrá empezar a utilizar de inmediato.

5.5 Resultado

Para verificar el correcto funcionamiento de la aplicación, en esta parte vamos a realizar unas series de pruebas de forma manual y testeadas por el desarrollador de la aplicación. Las pruebas han validado los casos de uso mencionados en la sección 3.2.1 (Imagen 8). Se recopilan las pruebas en las siguientes (tablas 13- 16).

Prueba	Caso de uso	Resultado
Test 01	CU01	El usuario inicia sesión correctamente.
Test 02	CU01	El usuario registra la sesión correctamente
Test 03	CU01	La sesión siempre se mantiene activa a pesar de salir de la aplicación y volver a entrar.
Test 04	CU01	La aplicación siempre muestra un mensaje de error cuando no es válido el usuario y la contraseña.

Tabla 9: Pruebas correspondiente al Inicio y Registro de sesión.

Prueba	Caso de uso	Resultado
Test 01	CU02	El usuario crear una sala correctamente con toda la información escrito por el usuario.
Test 02	CU02	El usuario entra una sala creada por el propio usuario y con todos los datos escrito por el usuario.
Test 03	CU03	El usuario crea una encuesta correctamente dentro de la sala creada por el propio usuario.
Test 04	CU03	La sala permite añadir más cuestiones a la encuesta.

Tabla 10: Pruebas correspondiente al Crear Salas y Encuestas

Prueba	Caso de uso	Resultado
Test 01	CU05	El usuario alumno entra correctamente a la sala introduciendo el grupo y el PIN de dicha sala.
Test 02	CU05	La aplicación siempre muestra un mensaje de error si el PIN introducido no corresponde a ninguna sala creada en la base de datos.
Test 03	CU06	El usuario alumno responde con éxito a una cuestión y avanza a la siguiente cuestión pulsando el botón "Siguiente".
Test 04	CU06	El usuario alumno finaliza la encuesta y la aplicación envía todas las respuestas a la base de datos.
Test 05	CU06	El usuario alumno finaliza la encuesta y vuelve a la pantalla de inicio sesión de alumno por si el alumno quiere realiza otra encuesta diferente.

Tabla 11: Prueba de inicio sesión del alumno y realizar la encuesta

Prueba	Caso de uso	Resultado
Test 01	CU04	El usuario tiene la opción de escoger entre todos los grupos de forma correcta.
Test 02	CU04	La base de datos permanece siempre conectada a la aplicación de forma satisfactoria.
Test 03	CU04	Las gráficas se generan de forma correcta y representan los datos de forma precisa almacenados en la base de datos.
Test 04	CU03	Las gráficas no se generarán si la encuesta no tiene ninguna respuesta.
Test 05	CU04	La aplicación muestra de forma correcta los resultados de un grupo y también todos los grupos

Tabla 12: prueba corresponde a la visualización del resultado de la encuesta.

5.5.1 Pruebas en escenario real

Por otra parte, una aplicación también es importante realizar pruebas en escenario real para ver si esta lista en pasarse a utilizar por el público. Para ello, hemos realizado una serie de pruebas en la Escuela de estudio durante el mes de Julio, participaron alumnos de curso 7º y 8º (13-14años). Puesto que eran niños y no todos tenían acceso al teléfono móvil para poder descargar e instalar la app, lo que hicimos fue generarla para versión web y colgarla en un servidor de AWS. La prueba consistía en realizar una encuesta sencilla que tenían que marcar las emociones que sentían en ese momento y enviar la respuesta. Las pruebas se realizaron en los laboratorios del DSIC y se registraron 89 usuarios de 7º y 89 usuarios de 8º, en total 178 usuarios.

Las pruebas se realizaron estos días:

Sesión	Día	Curso	Hora
Sesión 1	Jueves 6 de Julio	8º	11:30 a 12:30
Sesión 2	Viernes 7 de Julio	7º	11:30 a 12:30
Sesión 3	Lunes 10 de Julio	8º	11:30 a 12:30
Sesión 4	Lunes 17 de Julio	7º	11:30 a 12:30
Sesión 5	Martes 25 de Julio	7º	9:00 a 10:30
Sesión 6	Martes 25 de Julio	8º	11:30 a 12:30

Tabla 13: Tiempo que realizaron las pruebas

Todos estos participantes tenían que acceder al inicio de la sesión a rellenar la encuesta y otra vez al final de la sesión. Podrían acceder por dos medios, por el ordenador o por su teléfono.

Sesión	N º Enc al inicio	N º Enc al final
Sesión 1	79	60
Sesión 2	75	53
Sesión 3	62	57
Sesión 4	30	51
Sesión 5	43	40
Sesión 6	36	37

Tabla 14: Número de encuestas que se rellenaron en cada sesión.

En ningún momento la aplicación dejo de funcionar y dio siempre una respuesta rápida. La interacción con la base de datos fue buena y por tanto, podemos asumir que la aplicación estaría lista para poderla pasar a producción.

5.7 Conclusiones

En este capítulo se detalla la fase de implementación del presente proyecto. Se ha elaborado una sección de configuraciones previas para tener en cuenta antes de la fase de implementación. En ella, se ha especificado como configurar el entorno de desarrollo de la herramienta Ionic 2, como descargar e instalar las bibliotecas o módulos que vamos a emplear en la aplicación y se ha detallado como configurar la base de datos con Firebase para poder conectarla a nuestra aplicación. Posteriormente hemos detallado como codificar la aplicación por los casos de uso. Para cada caso de uso se han mostrado fragmentos de código relevante (código 1- 23) y la pantalla final después de su codificación (Imagen 24-30). Además, se ha especificado una fase de implantación en la que se detalla cómo poner en funcionamiento la aplicación una vez codificada. Para finalizar, se ha realizado unas series de pruebas en las que se comprueba el correcto funcionamiento de la aplicación (Tabla 9 -12) y una prueba en producción real de la aplicación (Tabla 13 y 14).
Con todo lo que hemos realizado durante todo el proyecto se procederá al capítulo final de nuestro proyecto: **las conclusiones**.

6. Conclusiones

En este proyecto hemos desarrollado una aplicación multiplataforma para medir las emociones en clase. Para llevar a cabo el trabajo hemos elaborado un estudio previo a aquellas herramientas similares a nuestra aplicación como Google Form, kahoot, y survey Nuts para compararlas con nuestras necesidades y obtener las características principales que tiene la aplicación.

Como resultado de este desarrollo, se han implementado las siguientes funcionalidades:

- Gestión adecuadamente los usuarios.
- La aplicación permite crear encuestas y gestionarlas.
- El usuario alumno realiza la encuesta sin tener que registrarse como usuario de la aplicación.
- Debe gestionar adecuadamente el resultado de las encuestas.
- Por cada repuesta de la encuesta los alumnos reflejan el estado emocional y la satisfacción en diferentes grupos de Clase.
- La aplicación recupera los datos de una base de datos y los visualiza mediante gráficas.
- La aplicación tiene que ser escalable y multiplataforma que pueda ser ejecutada en múltiples dispositivos.

Algunas de estas funcionalidades se vieron reflejadas en los casos de uso (Imagen 6). En el apartad de diseño hemos detallado un esquema conceptual general del funcionamiento del sistema (Imagen 7) y un esquema de interfaz de la capa de presentación (Imagen 8) que muestra las relaciones que tienen entre las distintas pantallas de la aplicación. Además, se ha detallado el diseño especifico de la capa de presentación mediante la realización de mockUps (Imagen 9 -14), la funcionalidad de la capa lógica o de negocio mediante diagramas de secuencias (Imagen 15-19). Por otro lado, para la capa de persistencia hemos detallado mediante un esquema entidad sobre la relación que tiene la aplicación con la base de datos, y para completar, una captura de la estructura JSON que esta almacenada en la base de datos (Imagen 21) y la estructura de organización de datos (Imagen 22).

Por último, hemos detallado la implementación de la aplicación, primero hemos especificado las configuraciones previas como:

- Configurar el entorno de desarrollo (Ionic 2) de la aplicación.
- Conectar la aplicación con Firebase.
- Descargar e instalar los módulos.

A continuación, hemos mostrado y detallado los fragmentos de código relevante de cada funcionalidad como:

- El inicio y registro de sesión (Código 1-4).
- Crear sala de encuesta (Código 5-9).
- Crear pregunta o encuesta (Código 10-11).
- El alumno realiza encuesta y almacenar respuestas (Código 12-16).
- El profesor consulta y muestra el resultado (Código 17-23).

Finalmente hemos explicado cómo se implanta la aplicación en distintos dispositivos o plataformas y también hemos elaborado unas series de pruebas para comprobar el correcto funcionamiento de la aplicación.

6.2 Dificultades y soluciones

En este apartado se detallarán las dificultades surgidas y los errores que hemos cometido a lo largo de todo el desarrollo del proyecto. A continuación, las soluciones que hemos aportado para resolver estas dificultades y errores.

En primer lugar, en la fase de análisis de requisitos se tuvo dificultades para seleccionar cuales serían las funcionalidades principales o casos de uso que mejor se adaptarían a la aplicación y al objetivo planteado inicialmente. Para resolver este problema, se ha realizado unas tutorías con mis tutores para consulta sus opiniones y elaborando diagramas con precisión junto con la fase de diseño.

Por otra parte, en la fase de diseño es una de las partes donde más dificultades han surgido, puesto que la interfaz de una aplicación es lo que ven los usuarios y puede variar constantemente si no es preciso. Hay que ser muy preciso tanto la posición de cada componente de la interfaz hasta el estilo y el color de cada pantalla. Para ello, hemos realizado una serie de diagramas y mockUps para determinar el diseño, y se modificaron progresivamente en el tiempo hasta dar con el diseño idóneo que facilitara la implementación de la aplicación. Muchos de los errores en el diseño pueden generar innumerables cambios en la implementación.

Por último, en la fase implementación también surgieron dificultades, ya que tanto la herramienta Ionic2 hasta el lenguaje de programación TypeScript son conocimientos nuevos para nosotros. Estos problemas se pueden resolver con los videos tutoriales y documentación de la página oficial de Ionic2. La parte más costosa fue la “upload” y “Down load” de datos de la base de datos Firebase, hubo varios atascos que estuvimos días y días intentando resolverlos. Finalmente, gracias a mi tutor y las personas del foro de Firebase se resolvieron todas las dificultades y errores.

6.2 Relación trabajo con estudios cursados

Aunque todas las herramientas y lenguajes de programación necesarios para realizar este proyecto son cosas nuevas para nosotros, pero lo podemos aprender rápidamente gracias a los conocimientos básicos de programación que hemos aprendido durante el grado de la ingeniería informática.

Por otra parte, gracias al grado, hemos aprendido la estructura básica de cualquier aplicación de distintos plataforma, y sobre todo, la manera correcta para llevar al cabo una aplicación desde cero. Estos nos ayudan bastante a la hora de organizar las tareas y realizar el proyecto paso por paso.

Por último, ya podemos contestar perfectamente a la pregunta: ¿Habría sido capaz de realizar el proyecto si no hubiera cursado el grado de ingeniería informática? La respuesta

es que NO, como ya hemos mencionado anteriormente, para realizar este proyecto requiere bastantes conocimientos previos tanto el conocimiento de programación y el método o estrategia para ponerse en marcha un proyecto y terminarlo de forma satisfactoriamente.

6.3 Trabajos futuros

Para este proyecto se puede ampliar en varios partes y en diferentes ámbitos, con la finalidad de mejorar la calidad de la aplicación. Actualmente, la aplicación tiene que ir renovándose y añadiendo funcionalidades para que los usuarios cubran todas sus necesidades en todo el momento. Los trabajos futuros a partir de este proyecto podrían ser añadir más opciones de tipo de encuestas para que el usuario tenga más opciones para elegir. También podría mejorar el código para aumentar la velocidad y pasos de consultar la estadística de los resultados de forma gráfica. No obstante, los datos también pueden sufrir un gran cambio en trabajos futuros, puesto que se pueden verse ampliados considerablemente no solo a la clase de la universidad sino a cualquier lugar de la sociedad. Por último, esta aplicación también puede incluir funcionalidad como redes sociales: foro, comentarios y Chat, con el objetivo de establecer una comunicación más cercana entre los profesores y los alumnos.

6.4 Referencias

[1] Ionic 2 documentation https://ionicframework.com/docs/
Ultimo acceso: 24/08/2017

[2] Firebase documentation https://firebase.google.com/docs/
Ultimo acceso: 21/08/2017

foro de stack overflow https://stackoverflow.com/questions
Ultimo acceso: 19/08/2017

[3] ng-Chart https://valor-software.com/ng2-charts/
Ultimo acceso: 22/08/2017

Estándar IEEE 830.
https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf
Ultimo acceso: 12/08/2017

Chris Griffith, M (2017) Mobile App Development with Ionic 2: Cross-Platform Apps with Ionic 2, Angular 2, and Cordova (Ingles) O'REILLY

[5] TypeScript: https://www.typescriptlang.org/docs/home.html
Ultimo acceso: 15/08/2017

[6] Angular 2: https://angular.io/tutorial
Ultimo acceso: 10/08/2017