Desarrollo aplicación educativa competitiva

Nombre Estudiante: Eric Gutiérrez Llopis
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Nombre Consultor/a: Roger Montserrat Ribes
Profesor/a responsable de la asignatura: Carles Garrigues Olivella

04/06/2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desarrollo aplicación educativa competitiva*
Nombre del autor:	*Eric Gutiérrez Llopis*
Nombre del consultor/a:	*Roger Montserrat Ribes*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2017
Titulación:	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Idioma del trabajo:	*Catalán, castellano o inglés*
Palabras clave	*Aprendizaje, aplicación, móvil*

Resumen del Trabajo

La finalidad de este Proyecto Final de Máster es la crear una aplicación Android que cubra la carencia de aprendizaje de idiomas a través de interacción entre usuarios. En concreto, lo que se quiere conseguir es que las personas del mismo nivel se reten en pequeños juegos para ver quién de los participantes ha consolidado mejor lo aprendido en cada temática propuesta. Estaría orientada a las personas que actualmente, con las aplicaciones existentes de aprendizaje, no consiguen motivarse y necesitan competitividad con otras personas para aprender.

Tanto la aplicación móvil cómo el servidor se ha desarrollado en Java y teniendo en cuenta los hitos impuestos en el proyecto se ha llevado a cabo una metodología en cascada para su desarrollo.

El resultado del proyecto ha sido todo un éxito. Se ha llevado todo lo planteado a cabo y se ha conseguido una aplicación 100% funcional con un diseño muy cuidado y moderno.

Gracias a este proyecto, se ha empezado el desarrollo de un producto que continuará en desarrollo una vez finalizado el Master.

Abstract:

The purpose of this Master Final Project is to create an Android application that covers the lack of language learning through interaction between users. Specifically, what we want to achieve is that people of the same level are challenged in small games to see which of the participants has better consolidated what was learned in each theme proposed. It would be oriented to the people who, with the existing applications of learning, do not get motivated and need competitiveness with other people to learn.

Both the mobile application and the server has been developed in Java and taking into account the milestones imposed on the project has been carried out

a cascade methodology for its development.

The result of the project has been a success. It has taken everything out and has achieved a 100% functional application with a very careful and modern design.

Thanks to this project, has begun the development of a product that will continue in development once the Master is finished.

Índice

Lista de figuras

1. Introducción

Este proyecto es un Trabajo Final de Máster de la especialidad Desarrollo de aplicaciones para dispositivos móviles de la Universitat Oberta de Catalunya [1].

1.1 Contexto y justificación del Trabajo

Hoy en día, gracias a internet, la comunicación a nivel mundial es posible y se usa diariamente para diversas situaciones, ya sea desde comunicarse con amistades hasta colaborar en proyectos empresariales que se están llevando a cabo en diferentes países. Este hecho genera la necesidad de aprender diversos idiomas con el fin de poder comunicarse con el resto del mundo sin restricciones.

Con el auge de los dispositivos móviles, se han desarrollado diversas aplicaciones para aprender idiomas, en concreto, destacan las aplicaciones de aprendizaje en inglés debido a que se ha establecido dicho idioma como idioma universal. Entre las diferentes aplicaciones que existen, destacan: Duolingo [2], Lingualeo [3], ABA English [4], Wlingua [5], Babbel [6] y Busuu [7]. Tal y como se puede observar en las diferentes tiendas de aplicaciones (destacan Apple Store [8] y Google Play [9]) el número de usuarios que utilizan estas aplicaciones es muy elevado, por lo que da a entender que las personas se sienten cómodas con estos métodos de aprendizaje.

Si se observan las aplicaciones mencionadas se puede observar que se pueden agrupar en dos grupos:

- **Aplicaciones de aprendizaje mediante minijuegos**: normalmente son aplicaciones en las que el usuario no tiene mucho conocimiento del idioma y quiere aprender a base de pequeños retos o juegos. Suelen ser aplicaciones en el que el objetivo principal es aprender más vocabulario que gramática y suelen ser *offline*.
- **Aplicaciones de aprendizaje mediante personas nativas al idioma**: normalmente son aplicaciones en las que el usuario debe tener un mínimo de nivel del idioma debido a que se le exige que haga pequeñas redacciones (en el grupo anterior serían frases simples), vea videos o escuche conversaciones en el idioma. La corrección de las redacciones, los videos y las escuchas suelen ser de personas nativas del idioma. Suelen ser aplicaciones en el que el objetivo principal es aprender más gramática que vocabulario.

En el grupo de aplicaciones de aprendizaje mediante personas nativas suele haber, tal y como se espera, comunicación entre diferentes usuarios.
Por ejemplo, en Busuu las correcciones las hace la comunidad. Si un usuario habla inglés y está aprendiendo castellano, escribirá redacciones en castellano y corregirá las de otros usuarios en inglés. En cambio, en las aplicaciones de aprendizaje mediante minijuegos no hay

comunicación entre usuarios (excluyendo foros de preguntas/respuestas).

Por lo tanto, se puede observar que hay un sector que no está siendo "explotado", y sería el sector de las aplicaciones de aprendizaje mediante minijuegos y cooperación/competición entre diferentes usuarios.

Según un estudio realizado por Quantic Foundry [10] a finales de 2016, una de las motivaciones para jugar sería la competitividad (esta motivación destacaría sobre todo en los hombres), a las personas les motiva ser mejores o avanzar de una manera más favorable que otras personas.

Así pues, teniendo en cuenta que existe un sector del aprendizaje en el que actualmente no existe competencia e influenciado por los resultados obtenidos en Quantic Foundry, se desea crear una aplicación de aprendizaje de idiomas, en la que el usuario puede ir aprendiendo a su ritmo, y a su vez, puede retar cuando lo desee a otras personas para ver quién sabe más dentro de un ámbito. De esta forma, el usuario tendrá una motivación de superación.

Dado la naturaleza del proyecto y el tiempo ajustado del que se dispone para la realización del proyecto, queda al margen la cooperatividad entre usuarios para llevar a cabo el aprendizaje del idioma. Este modo de aprendizaje quedará pospuesto para futuras versiones del producto.

Teniendo sólo en cuenta el modo competitivo, la aplicación, en cada reto, deberá recompensar a los usuarios de alguna manera. Por un lado, quien gane el reto conseguirá puntos que servirán para ir ascendiendo en el ranking global disponible para cada idioma (Para este proyecto sólo estará disponible el inglés). Por otro lado, el usuario que no gane, habrá conseguido identificar las palabras que más le cuesta debido a que el reto estará diseñado para que el usuario conteste rápido (sino lo hace se supondrá que no tiene esas palabras consolidadas).

1.2 Objetivos del Trabajo

El objetivo principal del proyecto se puede dividir en tres objetivos completamente relacionados entre sí:

- Ofrecer una solución software que ayude a personas de todo el mundo a aprender el idioma que desee a través de pequeñas pruebas, ya sea traducir una palabra o saber a qué tiempo pertenece un verbo.
- Ofrecer una solución software que gestione diferentes retos entre usuarios para ver qué usuario ha consolidado mejor las palabras de un idioma categorizado por temáticas.
- Ofrecer una solución software que registre qué palabras / frases / temas son las que más les cuesta a los usuarios aprender.

- Ofrecer una solución software que gestione diferentes modos cooperativos para llevar a cabo el aprendizaje de un idioma junto con otras personas (queda fuera del TFM).

Con tal de que se cumpla el objetivo principal se deberá cumplir los objetivos específicos mencionados a continuación:

- Poner en práctica todos los conocimientos adquiridos a lo largo del Máster.
- Realizar un aprendizaje sobre el funcionamiento de los servidores de notificaciones mediante *Tecnología Push* utilizadas para el envío de notificaciones a los dispositivos móviles.
- Analizar qué procesos pueden ser automatizados y qué procesos deben ser manuales, es decir, que tenga que intervenir la persona responsable del proceso.
- Analizar la información recopilada, seleccionándola y organizándola detalladamente, determinando así el alcance y las necesidades de la solución software.
- Diseñar un servidor que gestione las peticiones recibidas por el usuario y realice notificaciones a otros usuarios.
- Establecer qué plataforma móvil será la más adecuada para empezar el proyecto y llegar al máximo de usuarios posibles.
- Diseñar la aplicación móvil con toda la información, los requisitos y las especificaciones obtenidas de las partes interesadas y elaboradas por el desarrollador del proyecto.
- Elaborar las pruebas o correcciones necesarias con la finalidad de verificar que cumpla con las expectativas deseas.
- Implantar la aplicación móvil y el servidor para que pueda ser utilizada, realizando la respectiva capacitación de los usuarios finales.

Para que se lleve a cabo satisfactoriamente los objetivos mencionados, se debe cumplir con los siguientes requisitos funcionales:

Requisitos funcionales		
Id	**Descripción**	**Prioridad**
RF 1 Interacción Usuario – Dispositivo móvil		
RF 1.1	El usuario podrá registrarse mediante correo y contraseña / cuenta de Google	Alta
RF 1.2	El usuario podrá seleccionar el idioma que desea aprender	Media
RF 1.2	El usuario podrá visualizar qué ejercicios ha realizado y cuáles les falta	Alta
RF 1.3	El usuario podrá consolidar palabras según diferentes temas	Alta
RF 1.4	El usuario podrá ver sus puntuaciones y el ranking de los demás usuarios	Baja
RF 1.5	El usuario podrá retar a otro usuario en línea	Alta
RF 1.6	El usuario podrá compartir sus logros en las redes sociales	Baja

RF 2 Interacción Dispositivo móvil - Servidor		
RF 2.1	El dispositivo móvil deberá enviar las respuestas al servidor	Alta
RF 2.2	El servidor deberá indicar si la respuesta del usuario es correcta	Alta
RF 2.3	El servidor deberá registrar la respuesta en caso de fallo	Baja
RF 2.4	El servidor deberá emparejar a usuarios qué deseen retarse	Alta
RF 2.5	El servidor deberá indicar a los usuarios que se estén retando el turno del usuario	Alta
RF 3 Interacción Administrador - Servidor		
RF 3.1	El administrador podrá agregar nuevos contenidos de aprendizaje	Media

Así mismo, la aplicación deberá cumplir los siguientes requisitos no funcionales:

Requisitos no funcionales		
Id	**Descripción**	**Prioridad**
RNF 1 Requisitos de percepción		
RNF 1.1	El diseño de la aplicación tiene que ser atractiva y sencilla, invitando a hacer uso de ella	Obligatorio
RNF 1.2	El sistema ha de tener un diseño moderno y minimalista	Deseable
RNF 2 Requisitos de usabilidad y humanidad		
RNF 2.1	La aplicación móvil tiene que ser fácil de utilizar e intuitiva	Obligatorio
RNF 2.2	La aplicación se tiene que poder utilizar sin formación previa	Obligatorio
RFN 2.3	La parte lingüística de la aplicación tiene que ser impecable, sin ambigüedades. Los textos redactados los tiene que comprender todos los usuarios	Obligatorio
RNF 3 Requisitos de rendimiento		
RNF 3.1	El tiempo máximo de respuesta de la aplicación tiene que ser de 1 segundo en el 95% de las operaciones	Deseable
RNF 3.2	El sistema ha de ser capaz de dar respuesta a una petición en menos de 2 segundos	Deseable
RNF 3.3	Los usuarios han de poder acceder a la aplicación las 24 horas del día durante todo el año con un margen de error de un 10%	Deseable
RNF 3.4	La aplicación tiene que ser escalable y extensible	Obligatorio
RNF 4 Requisitos de funcionamiento y ambientales		
RNF 4.1	La aplicación a de poder utilizarse en cualquier lugar con conexión a internet	Deseable
RNF 4.2	La aplicación ha de disponer de aplicaciones	Obligatorio

	nativas para el sistema operativo Android [11] (y posteriormente para iOS [12])	
RNF 4.3	La aplicación dispone de un periodo de pruebas antes de su lanzamiento	Obligatorio
RNF 5 Requisitos de mantenimiento y soporte		
RNF 5.1	El sistema se tiene que someter a un mantenimiento mensual	Deseable
RNF 5.2	La aplicación tiene que ser compatible con versiones de Android 4.4 y posteriores	Obligatorio
RNF 6 Requisitos de seguridad		
RNF 6.1	Los usuarios no podrán acceder al aplicativo sin un previo registro	Obligatorio
RNF 6.2	El sistema no permite introducir datos en formatos incorrectos	Obligatorio
RNF 6.3	La aplicación debe proteger la información privada del usuario de acuerdo con las leyes de privacidad establecidas	Obligatorio
RNF 7 Requisitos culturales y políticos		
RNF 7.1	La aplicación no incluirá material ofensivo a religiones ni creencias de otras culturas	Obligatorio
RNF 7.2	Se debe tener los derechos de todo el material que se publique en la aplicación	Obligatorio

1.3 Enfoque y método seguido

Tal y como se ha visto anteriormente, no existe ninguna aplicación de aprendizaje de idiomas mediante minijuegos y cooperación/competición entre usuarios. Por lo que se llevará a cabo el desarrollo de un producto nuevo.

Ya que los requisitos están lo suficientemente especificados y que los principales hitos del proyecto están definidos por la propia Evaluación Continua, se podrán desglosar las tareas en sub-tareas para poder abarcar todo el conjunto de manera más eficiente y efectiva.

Dado tal y como está estructurada la Evaluación Continua (Primero plan de trabajo (Requisitos), luego diseño, y por último implementación) se llevará a cabo el desarrollo en cascada debido a que promueve una metodología de trabajo efectiva: definir antes que diseñar, diseñar antes que codificar.

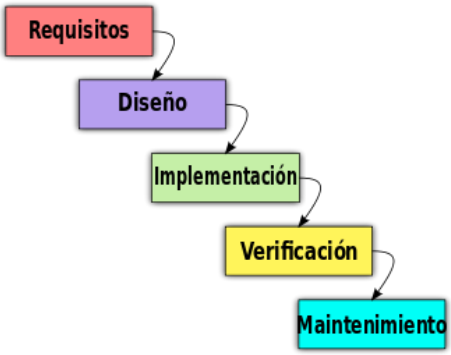


Ilustración 1: Modelo de cascada

Una vez finalizado el Trabajo Final de Máster, se continuará desarrollando el proyecto, pero cambiando la metodología por una metodología ágil.

Dentro de las metodologías ágiles se desarrollará utilizando *Scrum* [13] debido a la naturaleza del proyecto y porque cumple con los siguientes requerimientos:

- Obtención de resultados constantes.
- Los requisitos son cambiantes.
- La innovación, competitividad, flexibilidad y productividad son fundamentales.
- Riesgos de cambios durante el proceso.
- Equipos pequeños (únicamente un desarrollador).
- Realización de tareas no relacionadas simultáneamente.

1.4 Planificación del Trabajo

Para la realización del proyecto, se contará con las siguientes herramientas:

- Android Studio
- Eclipse / IntelliJ IDEA
- Dispositivo móvil con Android

El desarrollo del proyecto se llevará a cabo por el alumno Eric Gutiérrez Llopis, con la ayuda de los consultores del TFM. Debido a que el desarrollador trabaja y realiza otras asignaturas a la vez que el TFM dispondrá de 4 horas de lunes a viernes y 5 horas los días festivos, en total, 14 horas de media semanal. Las horas semanales se verán aumentadas los días laborables festivos y los días festivos personales de empresa.

A continuación, se muestra el diagrama de Gantt con las tareas y sub-tareas definidas junto con la dedicación necesaria para llevarlas a cabo satisfactoriamente, separadas por entregas.

- Primera entrega (PEC 1):

Nombre de tarea	Duración	Trabajo	Comienzo	Fin
PEC 1 - Plan de trabajo	**16 días**	**20 horas**	**mié 22/02/17**	**mié 15/03/17**
Contexto y justificación del Trabajo	16 días	5 horas	mié 22/02/17	mié 15/03/17
Objetivos del Trabajo	16 días	4 horas	mié 22/02/17	mié 15/03/17
Enfoque y método seguido	16 días	5 horas	mié 22/02/17	mié 15/03/17
Planificación del Trabajo	16 días	5 horas	mié 22/02/17	mié 15/03/17
Breve sumario de productos obtenidos	16 días	1 hora	mié 22/02/17	mié 15/03/17

Ilustración 2: Dedicación PEC1

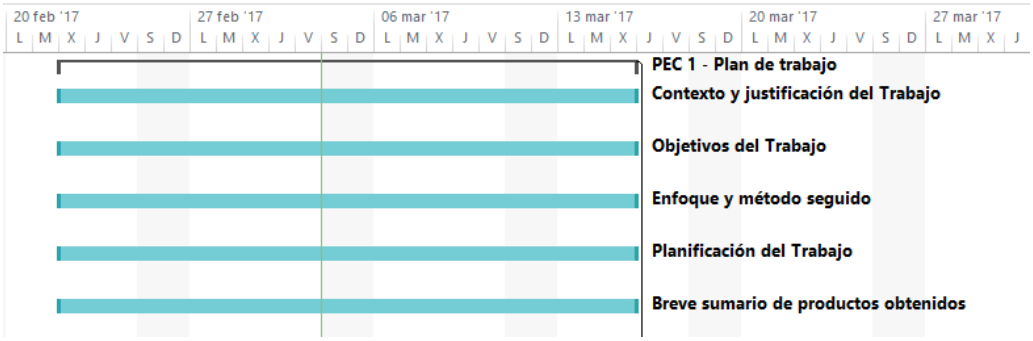


Ilustración 3: Diagrama de Gantt PEC1

- Segunda entrega (PEC 2):

Nombre de tarea	Duración	Trabajo	Comienzo	Fin
PEC 2 - Diseño	**15 días**	**56 horas**	**jue 16/03/17**	**mié 05/04/17**
Usuarios y contexto de uso	3 días	12 horas	jue 16/03/17	sáb 18/03/17
Diseño conceptual	3 días	12 horas	sáb 18/03/17	mar 21/03/17
Proptotipado	4 días	12 horas	mié 22/03/17	dom 26/03/17
Evaluación	1 día	2 horas	lun 27/03/17	lun 27/03/17
Definición de los casos de uso	5 días	12 horas	mar 28/03/17	sáb 01/04/17
Diseño de la arquitectura	3 días	6 horas	lun 03/04/17	mié 05/04/17

Ilustración 4: Dedicación PEC2

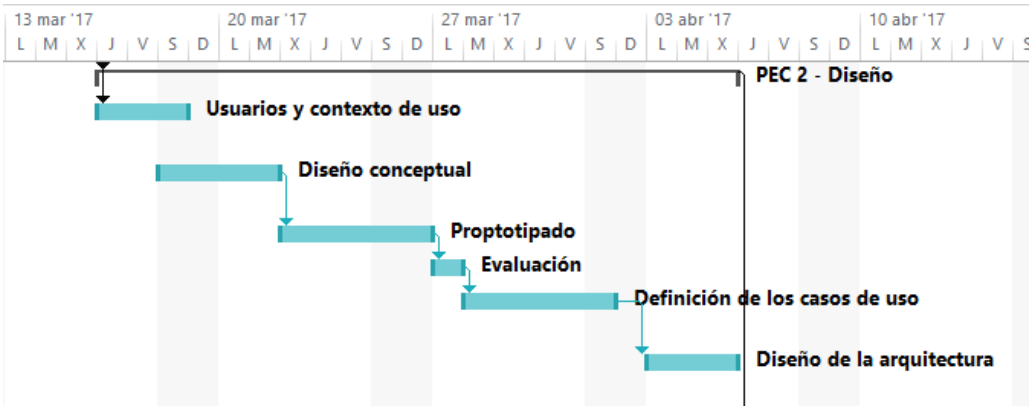


Ilustración 5: Diagrama de Gantt PEC2

- Tercera entrega (PEC 3):

Nombre de tarea	Duración	Trabajo	Comienzo	Fin
PEC 3 - Implementación	**30 días**	**130 horas**	**jue 06/04/17**	**mié 17/05/17**
Desarrollo	**27 días**	**110 horas**	**jue 06/04/17**	**vie 12/05/17**
Desarrollo del servidor	8 días	25 horas	jue 06/04/17	dom 16/04/17
Desarrollo de la base de datos	4 días	4 horas	lun 17/04/17	jue 20/04/17
Desarrollo de los servicios REST	2 días	9 horas	vie 21/04/17	dom 23/04/17
Desarrollo de la interfaz móvil	5 días	12 horas	lun 24/04/17	vie 28/04/17
Desarrollo lógica aplicación móvil	7 días	40 horas	sáb 29/04/17	dom 07/05/17
Desarrollo comunicación con servidor	2 días	10 horas	sáb 06/05/17	lun 08/05/17
Desarrollo comunicación notificaciones	4 días	10 horas	mar 09/05/17	vie 12/05/17
Pruebas	**3 días**	**20 horas**	**sáb 13/05/17**	**mié 17/05/17**
Test unitarios	4 días	20 horas	sáb 13/05/17	mié 17/05/17

Ilustración 6: Dedicación PEC3

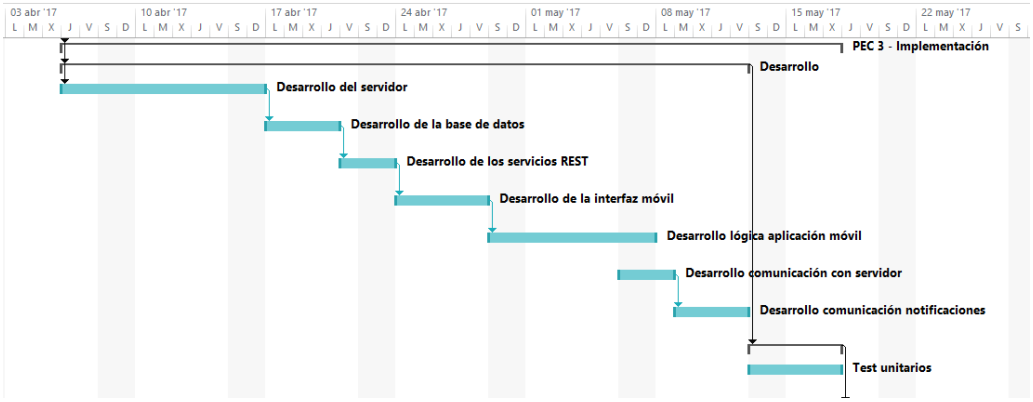


Ilustración 7: Diagrama de Gantt PEC3

- Entrega final:

Nombre de tarea	Duración	Trabajo	Comienzo	Fin
Entrega final	**15 días**	**49 horas**	**jue 18/05/17**	**mié 07/06/17**
Elaboración final de la memoria del proyecto	3 días	10 horas	jue 18/05/17	dom 21/05/17
Preparación de ejecutables a entregar	5 días	5 horas	lun 22/05/17	vie 26/05/17
Elaboración de los manuales de uso e instalación	2 días	16 horas	sáb 27/05/17	dom 28/05/17
Presentación del proyecto	8 días	18 horas	lun 29/05/17	mié 07/06/17

Ilustración 8: Dedicación Entrega final

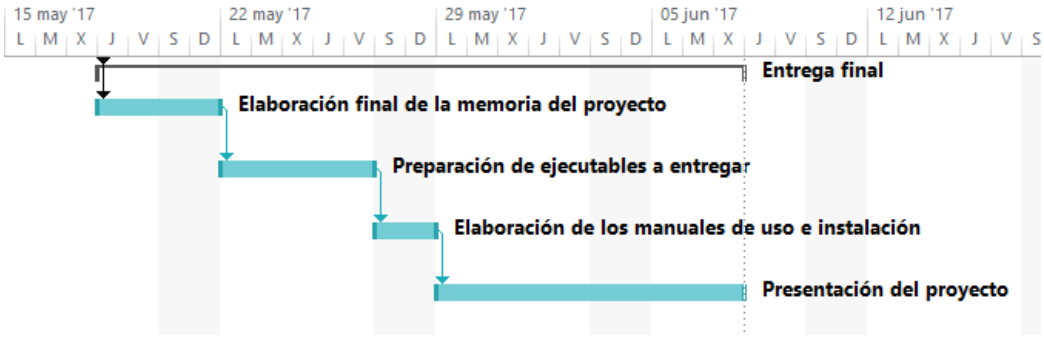


Ilustración 9: Diagrama de Gantt Entrega Final

1.5 Breve sumario de productos obtenidos

- Memoria del proyecto
- Ejecutable (.apk) de la aplicación Android
- Ejecutable (.jar) del servidor Java
- Manuales de uso e instalación

2. Diseño

El diseño que se va a llevar a cabo será un diseño centrado en el usuario, siguiendo el estándar ISO 9241-210:2010 "*Human centred design for interactive systems*"[14].

Siguiendo el estándar mencionado, la aplicación estará diseñada y probada (para obtener *feedbacks*) de los propios usuarios que la vayan a utilizar. De esta forma, la curva de aprendizaje de la aplicación será mínima, incrementando la satisfacción de uso y la comodidad del usuario al utilizarla. Dado que desde el proceso de diseño se obtiene *feedbacks* de los usuarios, es posible detectar posibles errores o problemas y poderlos solucionar cuanto antes.

2.1. Usuarios y contexto de uso

La aplicación que se quiere desarrollar está enfocada a los usuarios que quieren aprender o repasar un idioma mediante un dispositivo móvil. Para obtener cuánto tiempo, desde dónde y en qué momentos del día el usuario está interactuando con el móvil se ha llevado a cabo una encuesta online mediante Google Encuestas y distribuido por redes sociales (Facebook y Linkedin). Dicha encuesta también se ha utilizado para obtener información sobre si los usuarios utilizan aplicaciones de aprendizaje y si les gusta competir contra otros usuarios.

Los usuarios que han contestado a la encuesta tienen entre 20 y 46 años, todos disponen de dispositivo móvil, sorprendentemente todos han utilizado alguna vez alguna aplicación de aprendizaje. La media de utilización del móvil es de 187 minutos al día (3 horas aprox.) y suelen utilizar el móvil en el transporte público (en momentos de descanso) y por las noches antes de irse a dormir.

Con los resultados obtenidos podemos observar que las personas que quieren aprender o repasar un idioma no disponen de mucho tiempo para dedicarse exclusivamente al estudio. Para ello, se deberá desarrollar una aplicación con un uso muy sencillo y con ejercicios rápidos y simples debido al poco tiempo del que disponen los usuarios. Dado que el usuario utilizaría la aplicación en sitios comunes cerrados, como por ejemplo el transporte público o por las noches en casa, se deberá tener en cuenta que quizás el usuario no pueda utilizar ni el micrófono para hablar ni los altavoces para escuchar los sonidos/música (ya que no está solo en el transporte público y es posible que en casa haya personas ya descansando).

Teniendo en cuenta el análisis de los datos obtenidos, se ha creado la siguiente lista, donde se resumen las principales características que deberá tener la aplicación para satisfacer a los usuarios:

- Funcionamiento en teléfonos inteligentes

- Distribución del temario organizado por temáticas
- Navegación ágil con una mano
- Permitir uso continuado e intermitente
- Lista rápida de temario deseado

2.2. Diseño conceptual

2.2.1. Escenarios de uso

Con los datos obtenidos se puede obtener dos fichas de personas y escenario bien diferenciados:

Nombre: Iker
Edad: 22 años
Profesión: Estudiante
Descripción de la persona: Iker aún no está independizado, vive en la periferia de Barcelona y cada día tiene que ir al centro de la ciudad a estudiar a la Universidad. Lo primero que hace al levantarse es mirar el móvil, mientras que va a la Universidad está jugando o hablando con los amigos por “Whats App” y hace exactamente lo mismo cuando está en casa y no tiene deberes.
Descripción del escenario: Iker necesita aprender inglés, pero no quiere ir a una academia y utiliza una aplicación móvil para estudiar inglés cada día de camino a la Universidad. Tiene unos amigos que están en la misma situación que él, y para hacerlo más divertido, se retan entre ellos para echarse unas risas mientras van aprendiendo.

Nombre: Mireia
Edad: 38 años
Profesión: Dependienta
Descripción de la persona: Mireia trabaja en pleno centro de Barcelona y dado su puesto debe de comunicarse en inglés con otros clientes, pero ella no tiene un nivel fluido del idioma y no tiene tiempo de ir a academias. Sus horarios suelen ser partidos, por lo que el único tiempo libre que tiene es un par de horas al medio día y por las noches. Dado que se pasa todo el día trabajando el poco tiempo que tiene lo dedica a tareas domésticas, a su marido y a sus hijos.
Descripción del escenario: Mireia necesita reforzar el inglés, tiene 5 minutos de descanso en su casa y decide emplear esos 5 minutos en repasar un poco de vocabulario según la temática que ella debe emplear en su trabajo (trato al cliente, nombre de los productos que vende…etc). Hay días que la monotonía del repasar palabras le cansa por lo que decide interactuar con otros usuarios para repasar lo aprendido.

2.2.2. Diagrama de flujo

A continuación se muestra la estructura de la aplicación mediante un diagrama de flujo:

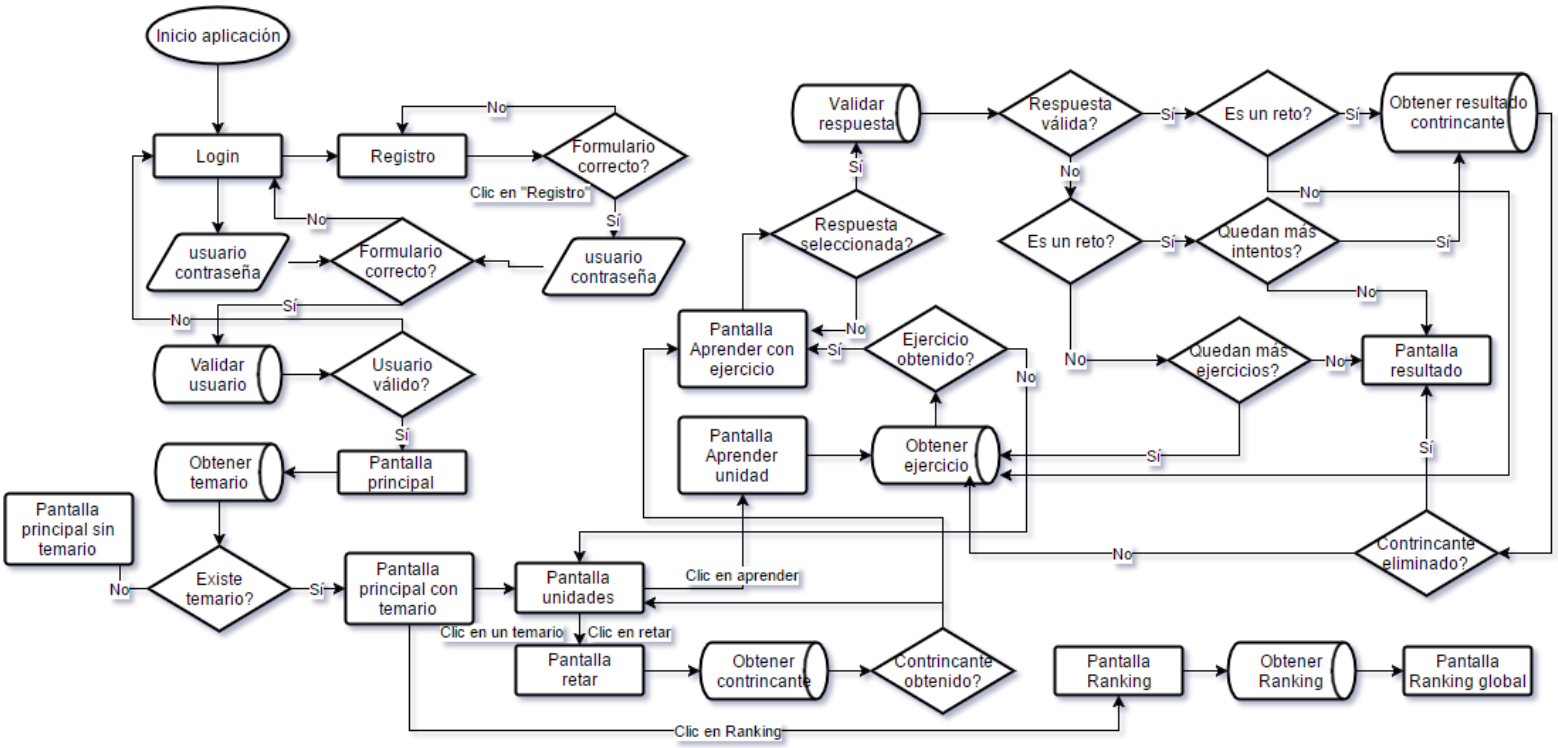


Ilustración 10: Diagrama de flujo

2.2.3. Diagrama de casos de uso

Partiendo de los escenarios de uso, el análisis de los usuarios y la evaluación del contexto de uso, se ha determinado que el aplicativo debe tener los siguientes casos de uso:

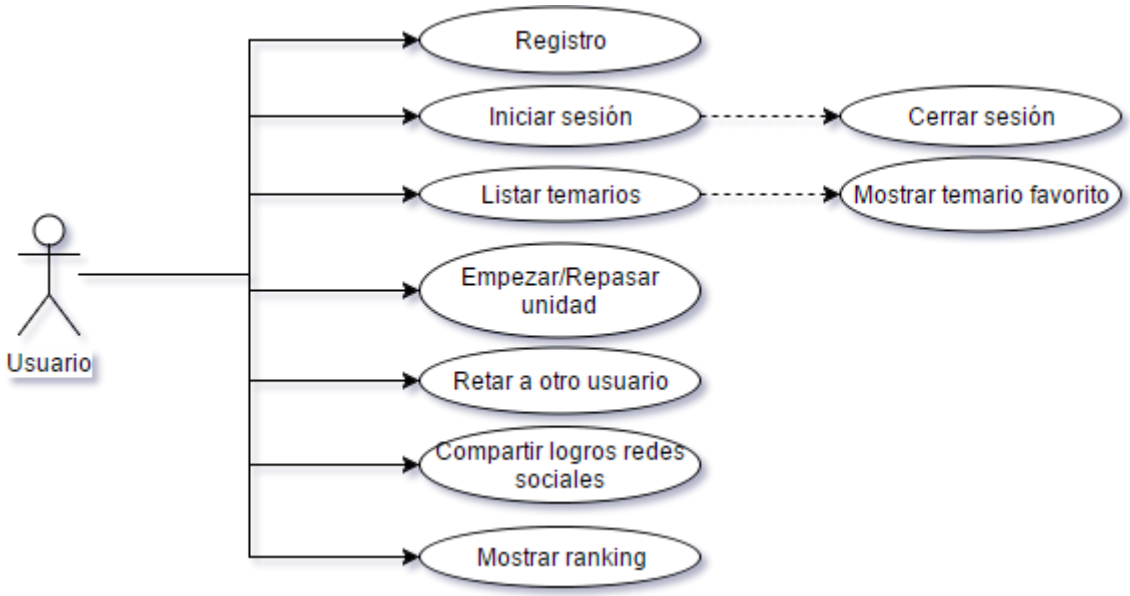


Ilustración 11: Diagrama casos de uso

A continuación, se realiza una explicación detalla de cada uno de los casos de uso representados en la ilustración anterior.

2.2.3.1. Registro

- **Actor principal:** Usuario
- **Descripción:** El usuario desea utilizar la aplicación para aprender un idioma, para ello debe realizar un registro previo para que pueda utilizar la aplicación en cualquier dispositivo y no tenga que empezar de nuevo.
- **Precondición:** El usuario tiene la aplicación instalada
- **Secuencia normal:**
  1. El usuario accede a la aplicación móvil y selecciona la opción de registrarse.
  2. El sistema muestra los datos necesarios para llevar a cabo el registro.
  3. El usuario facilita al sistema todos los datos necesarios.
  4. El sistema valida que el usuario ha introducido todos los datos.
  5. El sistema envía todos los datos al servidor.
  6. El servidor le indica al sistema que el usuario se ha registrado.
  7. El sistema notifica al usuario que se ha registrado correctamente.
- **Extensión:**
  4.A. El usuario no ha introducido todos los datos necesarios
    4.A.1. El sistema valida que el usuario no ha introducido todos los datos necesarios.
    4.A.2. El sistema notifica al usuario que faltan datos.
    4.A.3. El usuario vuelve al punto 3.
  5.A. El sistema no puede comunicar con el servidor
    5.A.1. El sistema detecta que el dispositivo no tiene conexión a internet o que no puede comunicarse con el servidor.
    5.A.2. El sistema notifica al usuario el error de comunicación.
  6.A. El usuario no se ha podido registrar
    6.A.1. El servidor notifica al sistema que el usuario no se ha podido registrar.
    6.A.2. El sistema notifica al usuario el motivo por el cual no se ha podido llevar a cabo el registro.
    6.A.3. El usuario vuelve al punto 3.

2.2.3.2. Iniciar sesión

- **Actor principal:** Usuario
- **Descripción:** Para que el usuario pueda realizar sus ejercicios de aprendizaje, primero debe iniciar sesión.
- **Precondición:** El usuario está registrado en el sistema.
- **Secuencia normal:**
  1. El usuario introduce su nombre de usuario y su contraseña.
  2. El sistema valida que exista un nombre de usuario y contraseña.
  3. El sistema envía al servidor el usuario y la contraseña.
  4. El servidor autentica al usuario y lo notifica al sistema.
  5. El sistema accede a la vista principal de la aplicación.

- **Extensión:**
  2.A. El usuario no ha introducido el nombre de usuario ni/o la contraseña
    2.A.1. El sistema detecta que no existe nombre de usuario ni/o contraseña.
    2.A.2. El sistema notifica al usuario que debe introducir un nombre de usuario y una contraseña para autenticarse.
  3.A. El sistema no puede comunicar con el servidor
    3.A.1. El sistema detecta que el dispositivo no tiene conexión a internet o que no puede comunicarse con el servidor.
    3.A.2. El sistema notifica al usuario el error de comunicación.
  4.A. El usuario o contraseña es incorrecto.
    4.A.1. El servidor notifica que el usuario o la contraseña es incorrecta.
    4.A.2. El sistema notifica al usuario el fallo de autenticación.

2.2.3.3. Cerrar sesión

- **Actor principal:** Usuario
- **Descripción:** El usuario puede cerrar sesión para que nadie tenga acceso a su cuenta.
- **Precondición:** El usuario esta autenticado.
- **Secuencia normal:**
  1. El usuario selecciona la opción de cerrar sesión.
  2. El sistema borra la sesión abierta del usuario.
  3. El sistema muestra la vista de autenticación.

2.2.3.4. Listar temarios

- **Actor principal:** Usuario
- **Descripción:** El usuario puede ver todos los temarios o todas las temáticas de aprendizaje que hay sobre el idioma que está aprendiendo.
- **Precondición:** El usuario está autenticado.
- **Secuencia normal:**
  1. El usuario selecciona la opción de listar temario.
  2. El sistema obtiene el listado del temario del idioma.
  3. El sistema obtiene el temario que el usuario ya ha realizado y cual tiene aún bloqueado.
  4. El sistema muestra al usuario todo el temario, indicando las unidades que ya ha realizado y las unidades que aún no ha desbloqueado.
- **Extensión:**
  2.A. No se puede obtener el temario del idioma
    2.A.1. El sistema no puede obtener el listado del temario del idioma.
    2.A.2. El sistema notifica al usuario que ha habido un error al obtener el listado del temario.
  3.A. No se puede obtener los temarios realizados por el usuario

3.A.1. El sistema no puede obtener el temario que el usuario ya ha realizado y el temario que el usuario aún tiene bloqueado.
3.A.2. El sistema notifica al usuario que ha habido un error al obtener sus logros.

2.2.3.5. Mostrar temario favorito

- **Actor principal:** Usuario
- **Descripción:** El temario puede ser muy extenso, el usuario debe poder filtrar todo el temario por temas de su interés.
- **Precondición:** El usuario está autenticado.
- **Secuencia normal:**
  1. El usuario selecciona la opción de mostrar temario favorito.
  2. El sistema obtiene el listado del temario favorito.
  3. El sistema obtiene el estado del temario favorito para el usuario.
  4. El sistema muestra al usuario su temario favorito y el estado en el que lo tiene (bloques, desbloqueos, logros…etc.).
- **Extensión:**
  2.A. No se puede obtener el listado del temario favorito.
    2.A.1. El sistema no puede obtener el listado del temario favorito
    2.A.2. El sistema notifica al usuario el error producido
  2.B. El usuario no dispone de temario favorito.
    2.B.1. El sistema detecta que el usuario no tiene temario favorito
    2.B.2. El sistema notifica al usuario como puede agregar temario favorito al listado.
  3.A. No se puede obtener el estado del temario para el usuario.
    3.A.1. El sistema no puede obtener el estado del temario del usuario.
    3.A.2. El sistema notifica al usuario el error producido.

2.2.3.6. Empezar/Repasar unidad

- **Actor principal:** Usuario
- **Descripción:** El usuario puede empezar una nueva unidad del temario o repasar una que ya haya finalizado en cualquier momento.
- **Precondición:** El usuario está autenticado.
- **Secuencia normal:**
  1. El usuario ejecuta el caso de uso "*2.2.2.4. Listar temarios"* o el caso de uso "*2.2.2.5. Mostrar temario favorito*".
  2. El usuario selecciona uno de los temas disponibles.
  3. El usuario selecciona la opción de empezar/repasar unidad.
  4. El sistema muestra el ejercicio
  5. El usuario selecciona la respuesta del ejercicio
  6. El sistema valida la respuesta del usuario
  7. El sistema vuelve al punto 4 hasta que acaben todos los ejercicios de la unidad
  8. El sistema felicita al usuario por haber acabado la unidad y le da la opción de compartir su logro en las redes sociales
- **Extensión:**

6.A. La respuesta del usuario es inválida.
6.A.1. El sistema valida que la respuesta del usuario es incorrecta.
6.A.2. El sistema notifica al usuario que la respuesta es incorrecta
6.A.3. El sistema vuelve al punto 4 si el usuario tiene más intentos.
6.A.3.A. El usuario no dispone de más intentos.
6.A.3.A.1. El usuario no dispone de más intentos.
6.A.3.A.2. El sistema notifica al usuario el fin de la partida
6.A.3.A.3. El sistema muestra al usuario los errores cometidos

2.2.3.7. Retar a otro usuario

- **Actor principal:** Usuario
- **Descripción:** El usuario en vez de aprender en solitario tiene la opción de ir aprendiendo con otros usuarios retándose entre ellos para ver quien aguanta más tiempo sin fallar.
- **Precondición:** El usuario está autenticado
- **Secuencia normal:**
  1. El usuario ejecuta el caso de uso "*2.2.2.4. Listar temarios*" o el caso de uso "*2.2.2.5. Mostrar temario favorito*".
  2. El usuario selecciona uno de los temas disponibles.
  3. El usuario selecciona la opción de retar a otro usuario.
  4. El sistema notifica al servidor que desea retar a otro usuario.
  5. El sistema espera a que haya otro usuario disponible
  6. El servidor notifica al sistema que hay otro usuario disponible
  7. El servidor indica el turno del usuario al sistema.
  8. El sistema indica al usuario el turno.
  9. El sistema muestra la pregunta.
  10. El usuario selecciona la respuesta.
  11. El sistema valida la respuesta.
  12. El sistema indica al servidor que la respuesta es correcta.
  13. El sistema espera a que el otro usuario de su respuesta.
  14. El sistema notifica al usuario el turno.
  15. El caso de uso vuelve al punto 9 en el caso de que no esté eliminado el contrincante.
  16. El sistema muestra el resumen de la partida y el ganador del reto.
- **Extensión:**

  4.A. El usuario no tiene conexión a internet o no puede contactar con el servidor.
  4.A.1. El sistema detecta que no hay conexión a internet o que no puede contactar con el servidor.
  4.A.2. El sistema notifica al usuario el error producido.

  6.A. No hay otro usuario disponible
  6.A.1. El servidor notifica al sistema que no hay otro usuario disponible.
  6.A.2. El sistema notifica al usuario que no hay otro usuario disponible para retar.

  11.A. La respuesta del usuario es incorrecta
  11.A.1. El sistema valida que la respuesta es incorrecta.
  11.A.2. El sistema quita una vida al usuario

11.A.3. El sistema notifica al servidor que ha fallado la pregunta pero que aún tiene vidas.
11.A.4. El sistema vuelve al punto 13.

11.A.3. Al usuario no le quedan más vidas
11.A.1. El sistema notifica al servidor que ha fallado la pregunta y que no le quedan más vidas.
11.A.2. El sistema vuelve al punto 16.

2.2.3.8. Compartir logros redes sociales

- **Actor principal:** Usuario
- **Descripción:** Cuando el usuario acaba una unidad o un reto tiene la posibilidad de compartir el resultado con sus contactos mediante las redes sociales.
- **Precondición:** El usuario está autenticado
- **Secuencia normal:**
  1. El usuario ejecuta el caso de uso "*2.2.2.6. Empezar/Repasar unidad*" o el caso de uno "*2.2.2.7. Retar a otro usuario*".
  2. El usuario selecciona la opción de compartir el resultado en las redes sociales.
  3. El sistema le muestra al usuario las redes sociales donde puede compartir el resultado.
  4. El usuario selecciona la red social.
  5. El sistema publica en su red social el resultado obtenido.
- **Extensión:**
  5.A No se puede publicar el resultado en la red social
  5.A.1. El sistema no puede publicar el resultado en la red social.
  5.A.2. El sistema notifica al usuario el error producido.

2.2.3.9. Mostrar ranking

- **Actor principal:** Usuario
- **Descripción:** Cuando un usuario acaba un reto consigue o pierde puntos. Dichos puntos sirven para que el usuario se posicione en el ranking. Dicho ranking el usuario lo puede ver en cualquier momento.
- **Precondición:** El usuario está autenticado
- **Secuencia normal:**
  1. El usuario selecciona la opción de mostrar ranking.
  2. El sistema obtiene el ranking del idioma.
  3. El sistema muestra al usuario el ranking del idioma.
- **Extensión:**
  5.A No se puede obtener el ranking del idioma
  5.A.1. El sistema no puede obtener el ranking del idioma.
  5.A.2. El sistema notifica al usuario el error producido.

2.2.4. Diseño de la Arquitectura

La arquitectura de la aplicación estará basada en dos de los patrones de diseño de software más utilizados: el modelo-vista-presentador (MVP) y la Arquitectura Clean.

En Android, el desarrollo que impone el IDE es la de MVC o MVP, pero para proyectos cambiantes y de gran tamaño creo conveniente utilizar Arquitectura Clean debido a sus ventajas:

- Independientes del Framework.
- Testeables (las reglas de negocio son fácilmente testeables sin utilizar la interfaz de usuario).
- Independiente de la interfaz de usuario.
- Independiente de la base de datos.
- El dominio es la parte más importante pero no depende de ninguna capa (utilización del principio SOLID de Inversión de dependencia).

La Arquitectura Clean se utilizará exclusivamente en el modelo, por lo que el resultado de la fusión de ambos patrones sería el siguiente:

- Vista: todos los ficheros xml que definen el diseño de las vistas.
- Controlador: se tendrá dos paquetes, el paquete encargado de la comunicación con los ficheros xml (views) y el paquete encargado de la lógica que debe tener la vista (presenters):
    - Views: serán todos los ficheros que se comunican directamente con las Vistas. Estos ficheros no dispondrán de ningún tipo de lógica que deba realizar la vista para cada interacción del usuario.
    - Presenters: serán los encargados de darle lógica a las vistas, por ejemplo, cuando el usuario le da clic a iniciar sesión, la capa “View” sólo notificará al “Presenter” que ha hecho clic el usuario. El “Presenter” cargará un *loading*, deshabilitará temporalmente el botón de iniciar sesión, obtendrá mediante la “View” el usuario y la contraseña y se lo facilitará al modelo. Cuando reciba una respuesta del modelo, por ejemplo de error, volverá a habilitar el botón de iniciar sesión, borrará la contraseña y mostrará un error. Como se puede observar, la lógica de lo que hará la vista está en el “Presenter”. En cada cosa que se haga en la vista la notificará a la “View” para que la realice.
- Modelo: tal y como se ha comentado anteriormente, el modelo es independiente tanto a la vista como a la base de datos o a cualquier agente externo. Por lo tanto, el modelo se dividirá en dos paquetes, interactors y repositories.
    - Interactors: sería el modelo en sí de la Arquitectura Clean. Es donde se definirán los casos de uso. Los “interactors” no se comunican con los “presenters” de ninguna manera. Los “interactors” enviarán la información que quieran enviar a la vista mediante un bus de datos (y son los presenters los

encargados de obtener la información del bus). La comunicación con los agentes externos se llevará a cabo mediante una interfaz definida en los repositorios.

- Repositories: será la parte donde se definen todos los agentes externos. Estos agentes no deberían de afectar a los casos de uso. Para ello, se definirá una interfaz para cada repositorio, de esta forma en caso de que cambie se mantendrá la interfaz. En este paquete estará definida las conexiones a la base de datos y las APIs de los servidores.

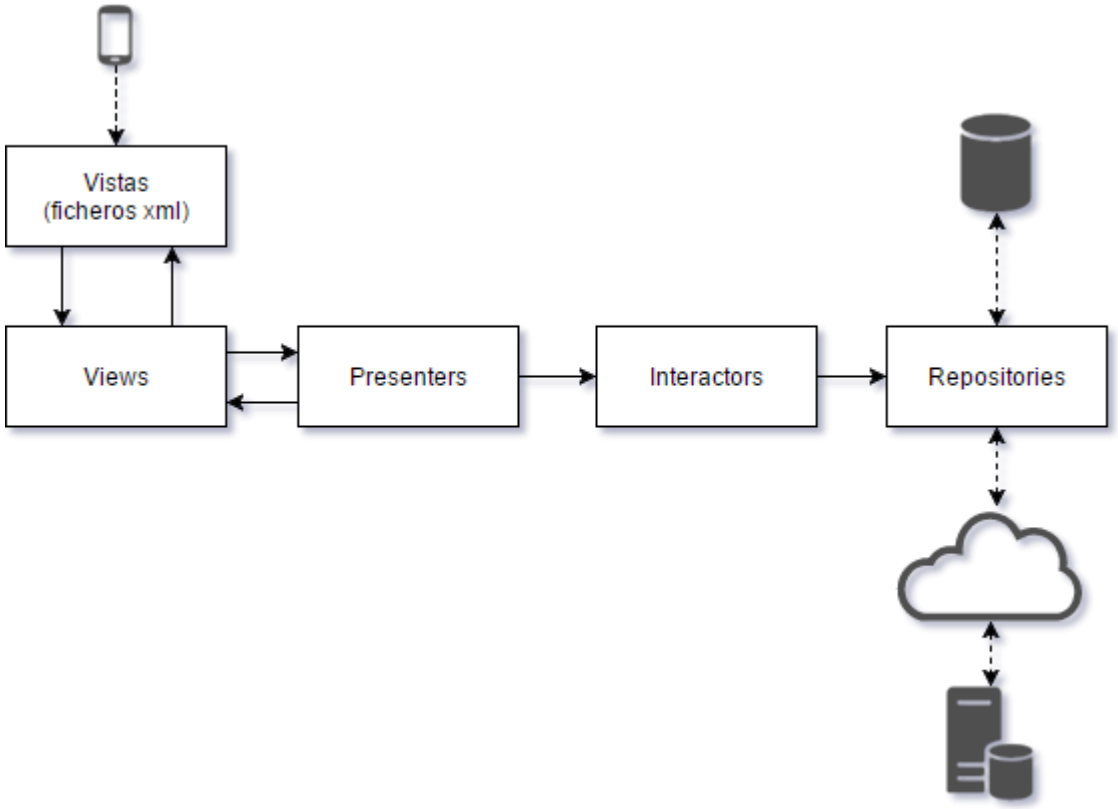


Ilustración 12: Diseño de la arquitectura móvil

2.2.5. Diagrama UML correspondiente al diseño de la base de datos

En futuras versiones del producto se quiere diferenciar los usuarios gratuitos de los usuarios Premium. Una de las ventajas que tendrá el usuario Premium es que tendrá la posibilidad de utilizar la aplicación offline, en cambio, la versión gratuita deberá comunicarse con el servidor para obtener el resultado de sus ejercicios. La base de datos se ha diseñado teniendo en cuenta este aspecto.

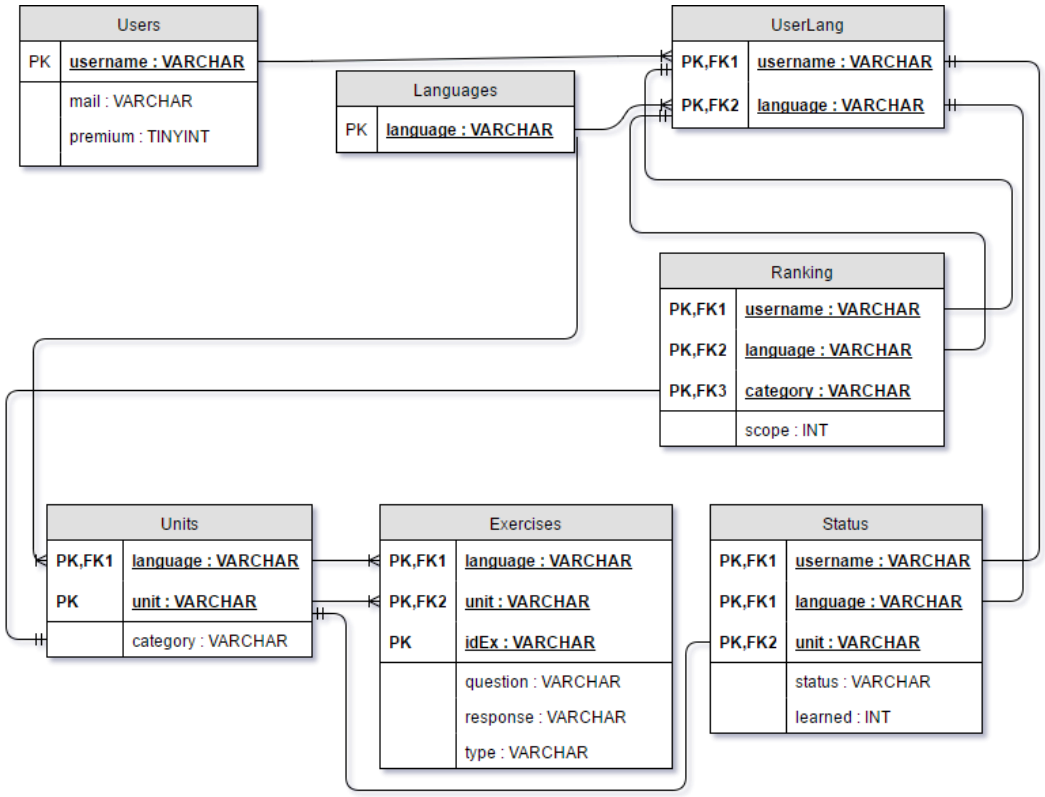


Ilustración 13: Diagrama UML de la base de datos

El diagrama definido anteriormente pertenecería a la parte del servidor. Para las versiones Premium serían las mismas tablas excepto la tabla "*Users*" y la tabla "*Ranking*" las cuales no se guardarían en el dispositivo móvil. Para el desarrollo del proyecto en el ámbito de trabajo final de Máster, se hará la versión de usuario gratuïto, es decir, la que tiene que consultar los datos a servidor.

A continuación se describe brevemente cada tabla y cada atributo de esta:

- *Users*: tabla que contiene información del usuario registrado
  - Username: usuario
  - Mail: correo electrónico del usuario
  - Premium: indica si el usuario es un usuario Premium
- *Languages*: tabla que contiene los idiomas que soporta la plataforma
  - Language: el idioma en cuestión

- *UserLang*: tabla que contiene la relación entre qué idiomas estudia cada usuario.
    - Username: identificador del usuario
    - Language: identificador del idioma que está aprendiendo
- *Units*: tabla que contiene todas las unidades de las que dispone un idioma.
    - Language: identificador del idioma
    - Unit: nombre de la unidad
    - Category: categoría a la que pertenece la unidad
- *Exercices*: tabla que contiene los diferentes ejercicios de los que dispone una unidad de un idioma.
    - Language: identificador del idioma
    - Unit: identificador de la unidad
    - idEx: identificador del ejercicio
    - question: pregunta del ejercicio
    - response: respuesta del ejercicio
    - type: tipo de pregunta del ejercio
- *Status*: tabla que contiene el estado en el que está una unidad para cada usuario
    - Username: identificador del usuario
    - Language: identificador del idioma
    - Unit: identificador de la unidad
    - Status: estado en el que está la unidad
    - Learned: porcentaje de la unidad que el usuario ha consolidado
- *Ranking*: Tabla que contiene el ranking de los usuarios en los desafíos
    - Username: identificador del usuario
    - Language: identificador del idioma
    - Category: categoría del idioma
    - Scope: puntuación del usuario

2.2.6. Diagrama UML correspondiente al diseño de las entidades y clases

El siguiente diagrama pertenece al diagrama UML correspondiente al diseño de las entidades y clases de la aplicación móvil para dispositivos Android. Tal y como se puede observar, se pueden ver las diferentes capas comentadas de la arquitectura clean:

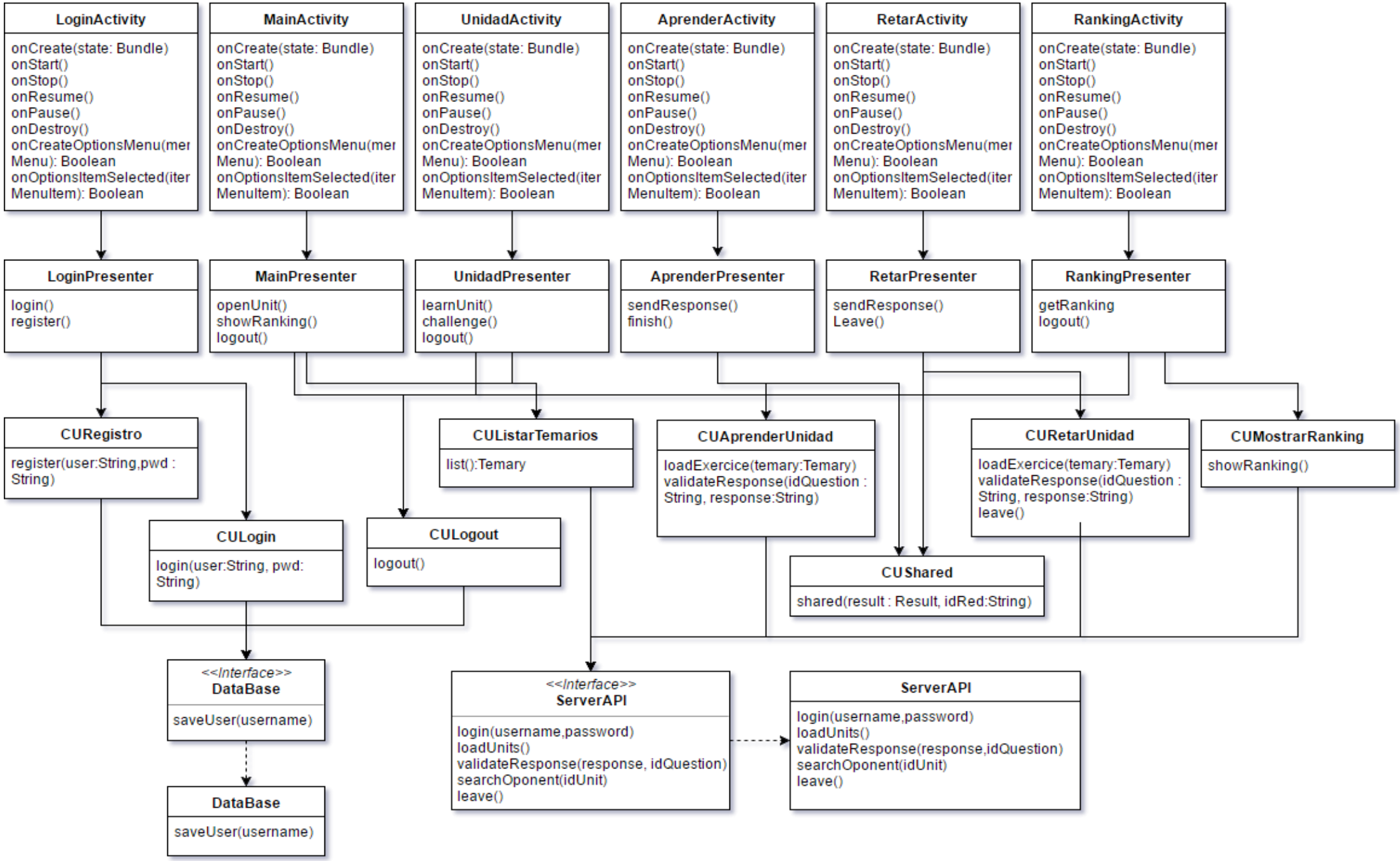


Ilustración 14: Diagrama UML diseño de las clases

2.3. Prototipado

Con el diagrama de flujo ya definido, se puede realizar el diseño de las pantallas. Por lo tanto, en este apartado se presenta un prototipo horizontal de la aplicación, mostrando cada pantalla y las funcionalidades dentro de las mismas.

A continuación, se muestra el prototipo de bajo nivel, el cual presenta las pantallas en bocetos. La ventaja que tiene realizar los bocetos es que son fáciles de crear y de modificar.

Ilustración 15: Prototipo bajo nivel

Después de haber realizado el diseño en bajo nivel, se ha realizado el diseño del prototipo en alto nivel. En comparación con el prototipo de bajo nivel, este representa una alta fidelidad en su diseño.

Si se compara ambos prototipos se podrá observar que ha habido pequeños cambios de diseño. Aun así, es posible que durante el desarrollo se vayan realizando pequeños retoques en el diseño para que se adapte perfectamente a las necesidades e inconvenientes encontrados durante dicho proceso.

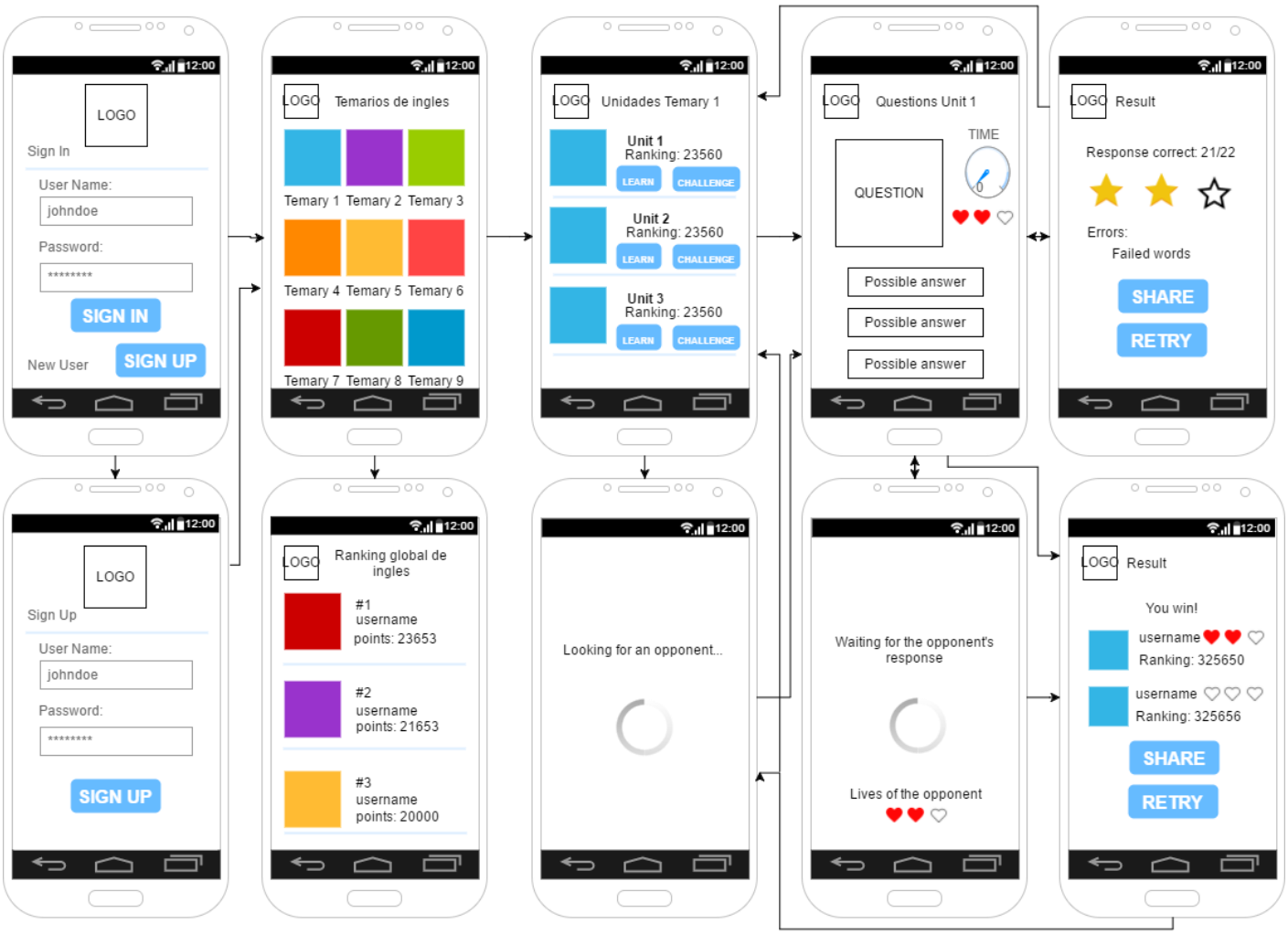


Ilustración 16: Prototipo alto nivel

2.3.1. Autenticación

Cuando el usuario abre por primera vez la aplicación, le saldrá la pantalla de autenticación.

En esta pantalla el usuario deberá autenticarse con su nombre de usuario y contraseña. En caso de que no esté registrado podrá registrarse dándole al botón de “sign up”.

En una futura versión podrá registrarse e iniciar sesión mediante la cuenta de Google o de Facebook. A parte, dispondrá de la opción de recordar contraseña.

2.3.2. Registro

En caso de que el usuario, en la pantalla anterior, seleccione la opción de registrarse, le aparecería la pantalla de registro, en la que el usuario deberá introducir un nombre de usuario y una contraseña.

Al darle al botón de registrar, el sistema validará que los datos sean correctos y una vez validado, el usuario ya estará registrado en la aplicación.

En una futura versión, se le pedirá también el correo electrónico, de esta forma se podrá contactar con el usuario mediante otro canal diferente (email en vez de mensajes push).

2.3.3. Temarios

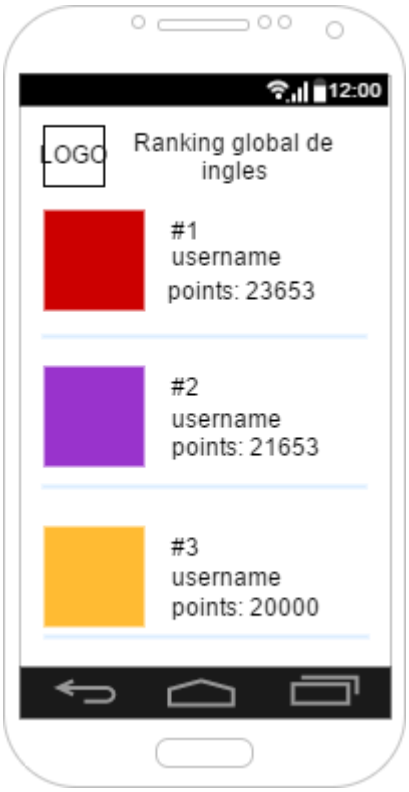
Una vez que el usuario se ha autenticado, la primera pantalla que ve será la del listado de temarios en inglés. En ella verá los diferentes temarios que ofrece la aplicación.

Para que sea más fácil y más rápido de detectar un temario, se verá una imagen representativa de cada temario, de esta forma, con un simple golpe de vista se vería el temario deseado.

En un futuro, los temarios estarán bloqueados y según vaya realizando irá desbloqueando. A parte, antes de que aparezca por primera vez el listado de los temarios en inglés, el usuario deberá de elegir el idioma que desea aprender.

Para ver los ejercicios de un temario, el usuario deberá pulsar sobre el temario deseado.

2.3.4. Ranking

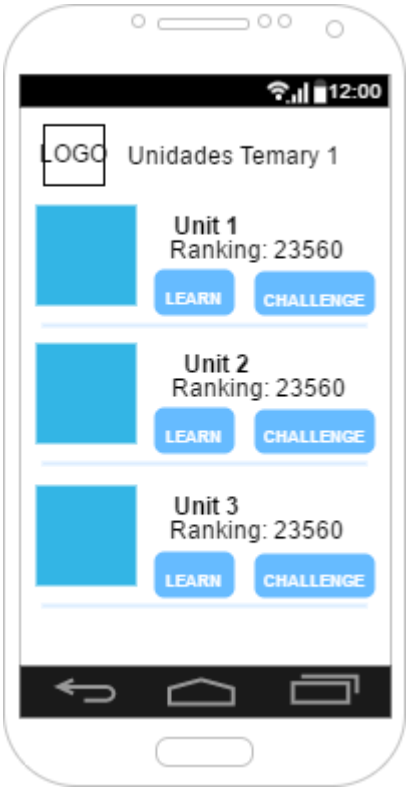


Existirá un menú lateral, en el que el usuario aparte de ver su perfil, podrá cerrar sesión, ver los temarios y ver el ranking global del idioma.

Si el usuario pulsa sobre la opción del ranking vería el ranking global, el cual, mostraría un listado con los usuarios que han obtenido más puntos en orden descendiente.

En el caso de que el usuario que está visualizando el ranking aparezca en el listado, se verá reflejado de algún color diferente para que el usuario se encuentre rápidamente en el listado.

2.3.5. Unidades

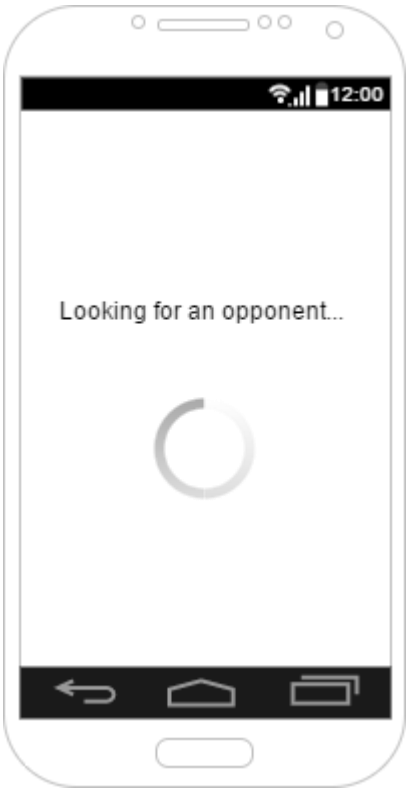


Cuando el usuario seleccione un temario, vería la pantalla de unidades, en el cual se muestran las unidades a las que pertenecen al temario seleccionado.

Al igual que con los temarios, las unidades también llevarán una imagen para detectarlas rápidamente.

A parte de mostrar la imagen y el nombre de la unidad se verá en qué posición del ranking está el usuario para ese temario en concreto.

2.3.6. Buscando oponente



Cuando el usuario seleccione la opción de retar a una persona, le aparecerá una pantalla en la que le indicará que está buscando un oponente.

Se verá una pequeña animación para que el usuario sea consciente de que la aplicación no está bloqueada.

En futuras versiones, aparte de buscar oponentes, se podrá retar directamente a conocidos.

2.3.7. Realizando ejercicio

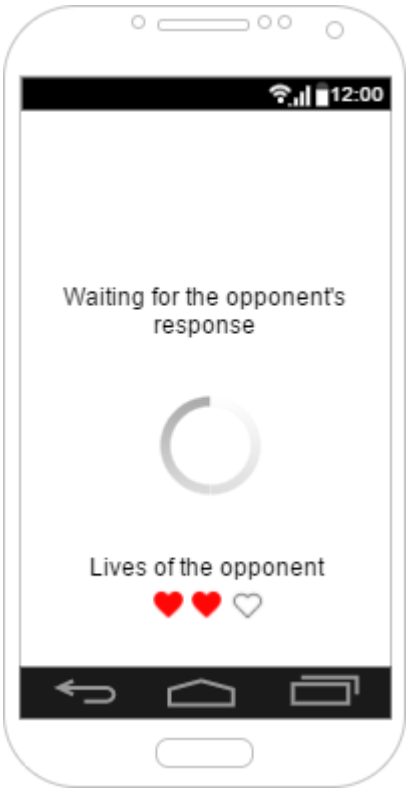
Esta es la pantalla más importante de la aplicación. Es en la que se pretende que el usuario pase más tiempo debido a que es donde el usuario estará practicando o retando a otros usuarios.

En ella aparecerá un ejercicio, en un principio una palabra, y en la parte inferior, aparecerán varías respuestas (una de ellas será la correcta).

Para realizar el ejercicio el usuario dispondrá de un tiempo limitado para resolverlo, el cual se verá reflejado en pantalla en todo momento.

Aparte, para darle un punto de *gaming*, el usuario tendrá unas vidas, las cuales perderá en cada fallo. En caso de que pierda todas las vidas, se dará por acabada la unidad y el usuario deberá empezar de nuevo.

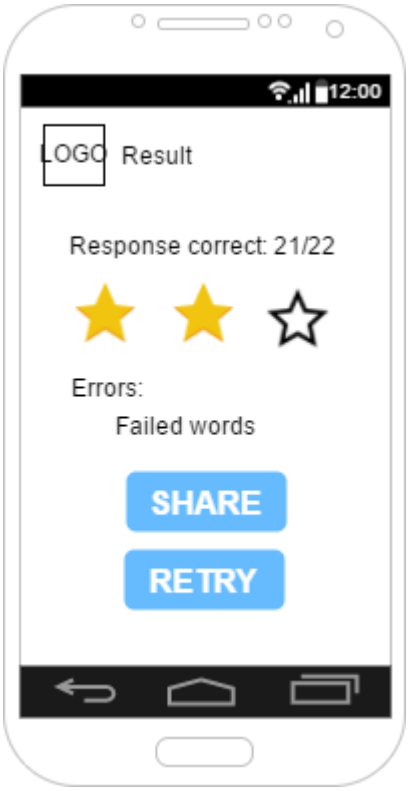
2.3.8. Esperando respuesta del oponente



En el caso de que el usuario este retando a otro usuario, y sea el turno de este, el usuario verá una pantalla de espera, en la que se le indicará que el oponente está respondiendo.

A parte también verá las vidas de las que dispone el oponente.

2.3.9. Resultado del entrenamiento



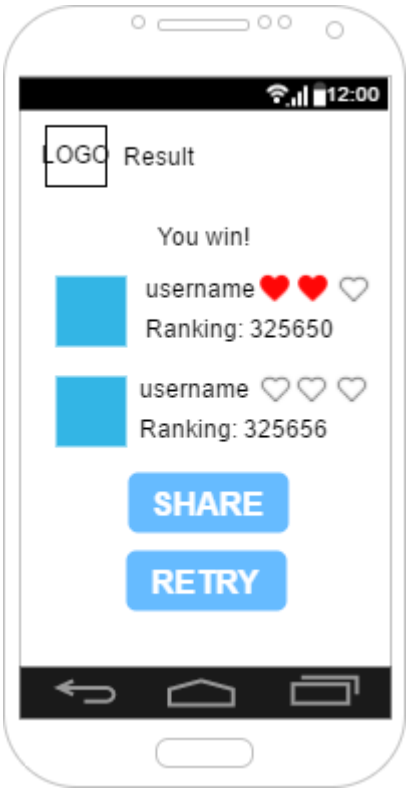
Una vez finalizado el entrenamiento, ya sea porque ha perdido todas las vidas o porque la ha finalizado satisfactoriamente, el usuario verá el resultado obtenido.

Verá el total de preguntas y respuestas realizadas, el nivel de madurez (1 estrella entre el 50% y el 74%, 2 estrellas entre un 75% y un 99% y 3 estrellas cuando responde correctamente el 100% de las preguntas).

En caso de que haya fallado alguna pregunta lo verá reflejado en la parte de errores (también verá la respuesta correcta).

Desde la misma pantalla podrá volver a realizar la unidad o compartir el resultado en las redes sociales.

2.3.10. Resultado del reto

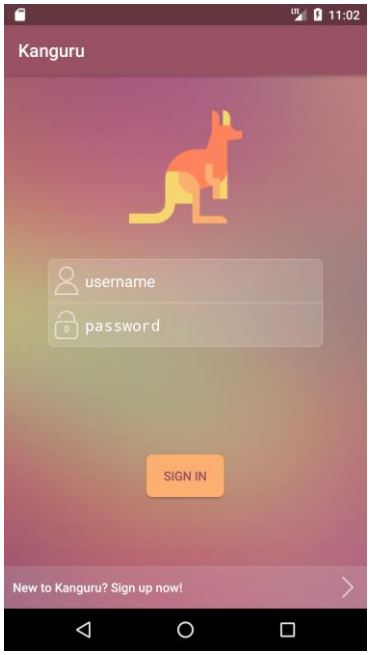


Una vez finalizado un reto, el usuario verá un mensaje indicándole si ha ganado o ha perdido. A parte, verá información de él y del oponente (usuario, vidas que le han quedado y posición del ranking). El ganador ganará 3 puntos en caso de que no tenga todas las vidas. En caso de que tenga todas las vidas obtendrá 5 puntos. El perdedor perderá 2 puntos.

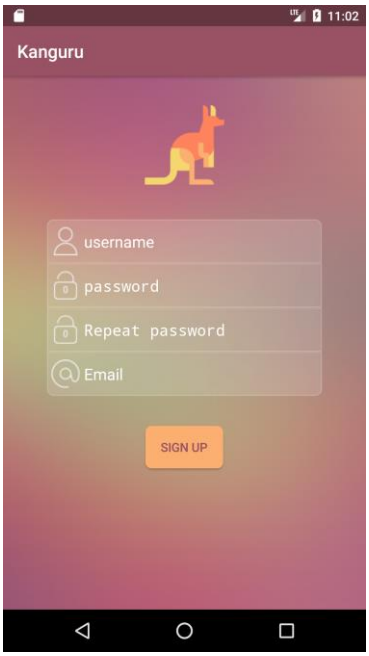
Al igual que con el entrenamiento, los usuarios podrán compartir el resultado en las redes sociales y podrán volver a retar desde la misma pantalla.

2.4. Diseño final de las pantallas

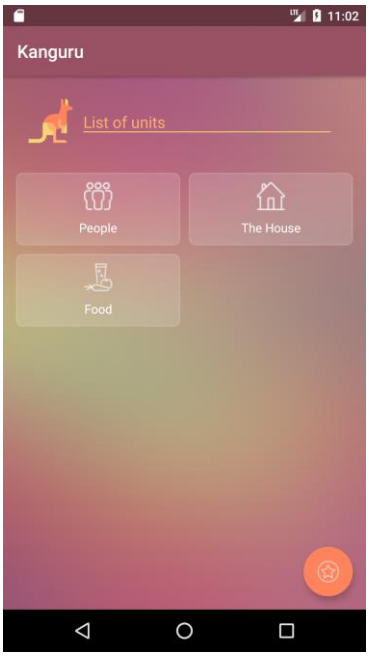
A continuación se puede ver el diseño final que tendrá la aplicación. De esta forma se podrá ver la evolución entre el diseño de alto nivel visto en el punto anterior y el diseño final.



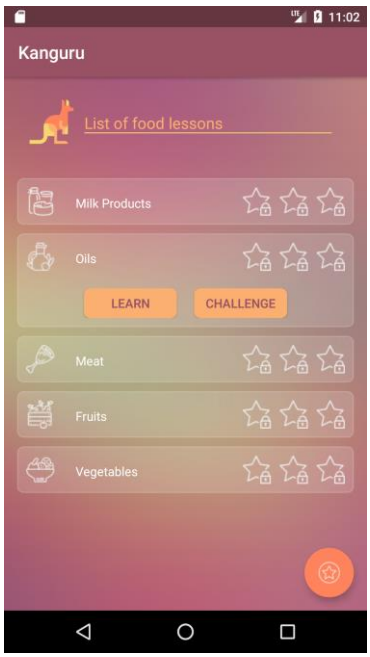
Login



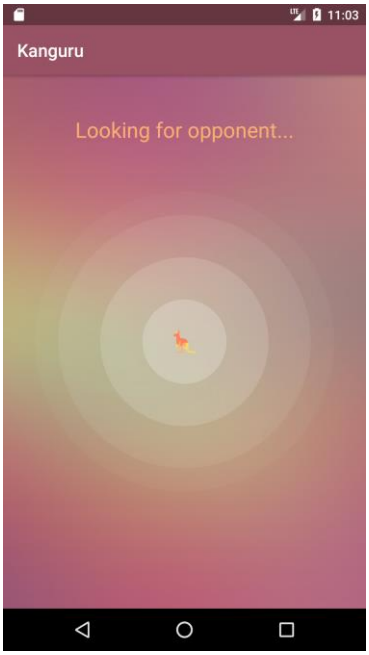
Registro



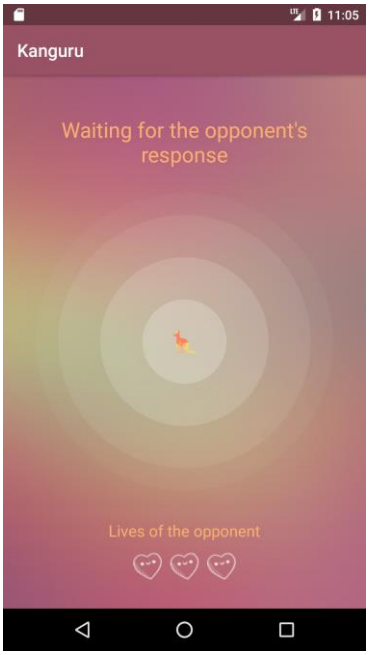
Unidaes



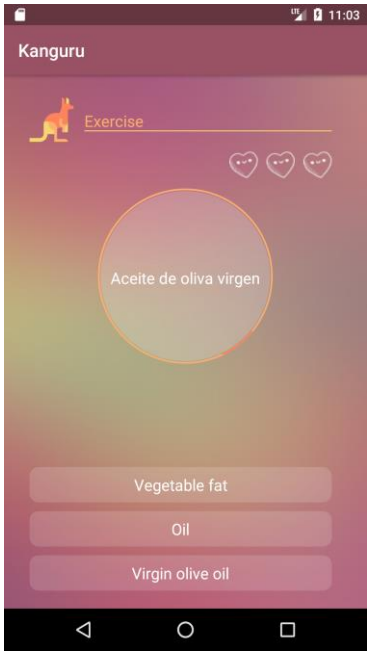
Lecciones



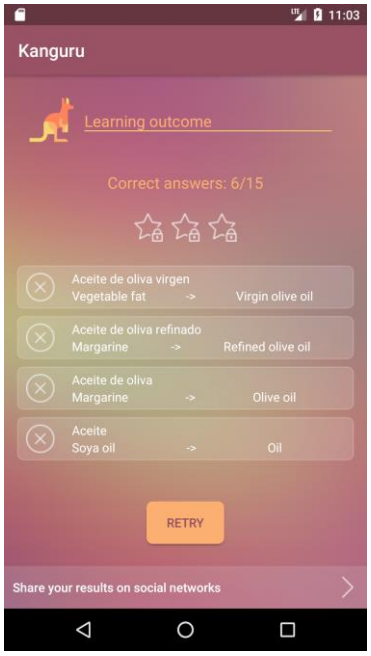
Buscando oponente



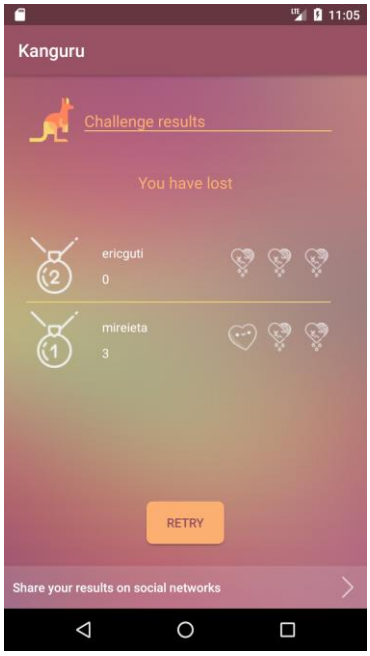
Esperando respuesta



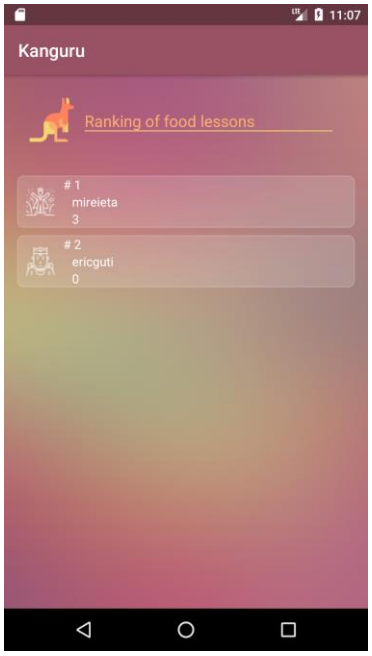
Ejercicio



Resultado aprendizaje



Resultado reto



Ranking

2.5. Evaluación

La tarea de evaluación se realiza para identificar posibles problemas de usabilidad y asegurar la calidad del producto. Para llevar a cabo la evaluación, el usuario final deberá interactuar intuitivamente con la aplicación sin ningún tipo de ayuda.

La tarea de evaluación es un proceso iterativo por lo que se debe realizar la evaluación de las interfaces de manera periódica hasta alcanzar un alto grado de madurez, dónde se hayan corregido las deficiencias encontradas en cada iteración.

El usuario final evaluador de la aplicación tendrá acceso al diseño horizontal de la aplicación y se le pedirá que intente realizar las siguientes tareas:

- Registro
- Inicio de sesión/Cierre de sesión
- Realizar el aprendizaje del temario
- Retar a otro usuario
- Compartir el resultado en las redes sociales

Una vez realizado todas las tareas deberán notificar:

- Si han tenido algún problema para la realización de las tareas.
- Si los colores utilizados son adecuados.
- Si se ha sentido desorientado en algún momento.
- Si detectan carencias en el producto.
- Si la distribución les parece la correcta.

Una vez finalizada la primera versión del producto, el usuario final evaluador, evaluará la aplicación comercial, así como cada versión posterior que se vaya a lanzar.

3. Implementación

3.1. Implementación de la parte de servidor

Como entorno de desarrollo de la parte de servidor se ha utilizado el IDE IntelliJ IDEA[15]. El lenguaje utilizado para su desarrollo ha sido Java utilizando el framework Spring[16], debido a su baja curva de aprendizaje y las múltiples facilidades que ofrece para crear una API REST.

Tal y como se ha mencionado anteriormente, se ha utilizado la arquitectura Clean para el desarrollo del servidor. En él, podemos encontrar la aplicación que levanta el servicio, el servicio, las entidades y los repositorios.

En los repositorios se encuentra Firebase[17], para la gestión de envío de notificaciones a los dispositivos móviles. La base de datos ha sido sacrificada en esta primera versión del proyecto (a nivel académico) debido a los diferentes inconvenientes encontrados (comentados más adelante). Por lo tanto, las unidades y lecciones disponibles actualmente se cargan en memoria cuando se arranca el servicio. Los usuarios se guardan en un fichero JSON externo cuando el usuario realiza el registro.

Para la utilización de librerías de terceros se ha utilizado Maven, herramienta de software para la gestión y construcción de proyectos Java.

3.2. Implementación de la parte móvil

Como entorno de desarrollo de la parte móvil se ha utilizado el IDE Android Studio (también de IntelliJ IDEA). El lenguaje utilizado ha sido Java, pero para futuras versiones, se está planteando migrar el código a Kotlin[18] dada las ventajas que ofrece.

La mínima versión de SDK de Android para que funcione la aplicación es la API 19: Android 4.4 (KitKat).

Tal y como se mencionó anteriormente, se ha utilizado la arquitectura clean para el desarrollo de la aplicación móvil. En ella, podemos encontrar las vistas, los presentadores, los interactuadores, los repositorios, los DTO (Data Transfer Object) y la aplicación principal. Para el guardado de información se ha utilizado las Shared Preferences[19] debido que en esta versión se guarda únicamente información de configuración (token de autenticación e identificador de dispositivo en Firebase).

Para la utilización de librerías de terceros se ha utilizado Gradle, herramienta de software para la gestión y construcción de proyectos Java.

Dado que la parte más importante del proyecto es la aplicación móvil voy a mencionar y explicar brevemente las diferentes librerías de terceros utilizadas en el proyecto:

- ***Gson***[20]: librería de Google para transformar ficheros JSON a objetos Java y viceversa.
- ***Retrofit***[21]: librería para realizar peticiones a servidor de manera simple.
- ***Butterknife***[22]: librería para referenciar elementos visuales en clases Java mediante anotaciones en lugar de utilizar el findViewById().
- ***Eventbus***[23]: librería para crear buses de información. Se encarga de poner "escuchas" en las clases deseadas y cuando otra clase escribe por el bus suscrito obtiene los datos para su tratamiento. Esta librería es ideal para la arquitectura clean debido a que aunque haya funciones asíncronas no nos debemos de preocupar de quien está esperando la información.
- ***CircularMusicProgressBar***[24]: librería pensada para mostrar un tiempo en forma circular (pensado para la música). En este proyecto se ha utilizado para el cronometro de cuenta atrás en la realización de los ejercicios.
- ***Pulsator***[25]: librería para realizar la animación de ondas mientras se está buscando oponentes o se está esperando la respuesta del oponente.
- ***FoldingCell***[26]: librería para realizar la animación de desplegable en las lecciones.
- ***HanksTextView***[27]: librería para realizar la animación de los textos en la realización de los ejercicios.
- ***Firebase***: librería para el control y la gestión de las notificaciones recibidas del servidor.

3.3. Inconvenientes durante el desarrollo

En un principio, durante la primera fase del desarrollo del servidor no hubo ningún problema. Pero cuando se empezó a realizar el desarrollo de la aplicación móvil surgió el problema de intentar enviar peticiones al servidor, debido a que no dispongo de una IP pública de mi ordenador.

Para solventar el problema se buscó un servidor gratuito en el que se pudiese arrancar proyectos Java. El servidor escogido fue Heroku[28], el cual cumplía casi todas las necesidades de las que necesitaba.

En la primera versión de servidor se utilizaba una base de datos no relacional llamada MongoDB[29]. Se utilizaba para guardar los datos de las unidades y lecciones, ranking y usuarios. Heroku no acepta mongoDB, solo acepta base de datos relacional. Para solventar este problema se eliminó la base de datos y se ha ficheros JSON debido a la falta de tiempo.

Otros inconvenientes de utilizar Heroku en su versión gratuita es que cada media hora de inactividad se apaga el servicio (Se vuelve arrancar

cuando se realiza una nueva petición). Otro inconveniente es que los ficheros generados en Heroku se borran cada vez que se apagan, por lo que los usuarios registrados se borran cuando se apaga (es como si estuviesen en memoria). Una solución sería guardar los usuarios en Firebase, pero por falta de tiempo no se ha realizado dicha mejora.

3.4. Pruebas de la aplicación

Durante el desarrollo se han realizado diferentes pruebas para comprobar el correcto funcionamiento de la aplicación:

- En la primera fase se ha utilizado Postman[30] para comprobar que todas las funcionalidades de las que dispone el servidor eran correctas.

- En la segunda fase se ha realizado la navegación por todas las pantallas de la aplicación móvil, para comprobar que la navegación entre ellas era correcta. Se trataba de validar las vistas y los presentadores de la aplicación.

- En una tercera fase se ha realizado todos los casos funcionales con el servidor configurado en la aplicación móvil. Se trataba de validar que los interactors y los repositorios (data y server) funcionaban correctamente.

- En una cuarta fase se ha realizado pruebas tanto desde Posman como desde el servidor para comprobar que la aplicación móvil obtenía y trataba correctamente todas las notificaciones recibidas.

- Por último, la quinta fase fue con múltiples dispositivos (Honor 7, Samsung Galaxy S7 y emulador de Android Nexus 5X) realizar retos y ver que funcionaba correctamente, tanto los retos como el sistema de puntuación/ranking.

4. Conclusiones

A nivel personal, puedo concluir que este trabajo de fin de Master ha sido una gran experiencia y todo un éxito, dado que una de las motivaciones que tenía para realizar el Master era llevar a cabo este proyecto.

Con la realización del proyecto, he podido adentrarme más en la gran comunidad de desarrolladores que hay, he podido conocer nuevos componentes desconocidos para mí hasta la actualidad y he podido indagar en muchas de las librerías de terceros que ayudan muchísimo tanto en el desarrollo como en la experiencia final del usuario.

La planificación se ha ajustado a la realidad en todas las fases excepto en las subtareas de la fase de desarrollo. El desarrollo del servidor ha tenido variaciones, como por ejemplo la base de datos, que no se ha llevado a cabo debido a incompatibilidades con Heroku. Por otro lado, el tiempo que he ahorrado en el desarrollo del servidor lo he consumido en el desarrollo de la aplicación móvil. Tuve un bug con una de las librerías utilizadas y tarde varios días en encontrarlo.

A quedado fuera del proyecto la integración completa de las redes sociales (con sus respectivas librerías) debido a la falta de tiempo. Como solución temporal se puede compartir con redes sociales mediante un Intent de Android.

Dado que este proyecto ya lo tenía en mente desde hace mucho tiempo, lo que se muestra en este proyecto sólo es la punta del iceberg. Por eso, aún queda muchas ampliaciones por hacer que se llevarán a cabo a lo largo del año:

- Desarrollo de una base de datos no relacional (MongoDB).
- Introducir más material al producto.
- Añadir control de errores y de uso (Crashlytics, Google Analytics).
- Introducir la API de Facebook y Twitter.
- Añadir potenciadores para el modo competitivo/cooperativo (con opción de compra).
- Cuenta premium de usuario, el cual podrá aprender en modo offline.
- Envío de notificaciones cuando el usuario lleve varios días sin utilizar la app.
- Creación de una web para el proyecto e incluir la versión web de la app.
- Traducir la aplicación móvil de Java a Kotlin

Como conclusión final, considero que la aplicación reúne todas las características de programación móvil que hemos estudiado en este Master (uso de Activities, Intents, Listas personalizadas, Firebase, Listeners…etc), demostrando en su conjunto el total aprovechamiento del mismo.

5. Glosario

- ***Android***: Sistema operativo basado en el núcleo Linux
- ***Android SDK***: Kit de desarrollo para Android, que contiene las herramientas básicas necesarias para desarrollar en Android, con independencia del IDE.
- ***Android Studio***: Entorno de desarrollo integrado para la plataforma Android.
- ***IDE***: Entorno de programación
- ***MVC***: Modelo-Vista-Controlador, es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la inferfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.
- ***Arquitectura Clean***: es un patrón de arquitectura de software que se focaliza en los casos de usos y tiene completamente separada la parte gráfica, el modelo y el repositorio. La vista sólo conoce una interfaz del modelo y el modelo solo conoce una interfaz del repositorio.
- ***MongoDB***: Es un sistema de base de datos NoSQL orientado a documentos, desarrollo bajo el concepto de código abierto.
- ***Servidor Backend***: Es el servidor que se dedica a la parte lógica de la aplicación, principalmente de la persistencia de los datos y la comunicación entre dispositivos móviles.
- ***Bug***: Termino inglés utilizado para referirse a un error de software
- ***Framework***: Término inglés utilizado para referirse a un conjunto de herramientas de software, destinadas a un uso genérico, empaquetadas y distribuidas para poder ser adaptadas en un desarrollo.

6. Bibliografía

[1] UOC (2017). *Universitat Oberta de Catalunya* [en línea]. [Consultada: 06 marzo de 2017]. Disponible en internet < http://www.uoc.edu >

[2] Duolingo (2017). *Duolingo: Aprende inglés, francés y otros idiomas gratis* [en línea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < https://es.duolingo.com >

[3] Lingualeo (2017). *Lingualeo – Inglés en línea* [en línea]. [Consultada: 06 marzo de 2017]. Disponible en internet < https://lingualeo.com/es >

[4] ABA English (2017). *ABA English – El método definitivo* [en línea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < www.abaenglish.com >

[5] Wlingua (2017). *Wlingua – Aprende inglés online, rápido y fácil* [en linea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < www.wlingua.com/es/ >

[6] Babbel (2017). *Babbel te hace hablar* [en línea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < https://es.babbel,com >

[7] Busuu (2017). *Busuu: aprende idiomas online* [en línea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < https://www.busuu.com/es >

[8] App Store (2017). *App Store Descargas en iTunes* [en línea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < https://itunes.apple.com/es/genre/ios/id36?mt=8 >

[9] Google Play (2017). *Google Play* [en línea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < https://play.google.com/store?hl=es >

[10] Quantic Foundry (2016). *7 Things We Learned About Primary Gaming Motivations From Over 250,000 Gamers* [en línea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < http://quanticfoundry.com/2016/12/15/primary-motivations/ >

[11] Android (2017). *Android* [en línea].[Consultada: 06 de marzo de 2017]. Disponible en internet < https://android.com/intl/es_es >

[12] iOS (2017). *iOS – iOS 10 – Apple* [en línea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < www.apple.com/es/ios/ios-10/ >

[13] Scrum (2017). *Scrum (desarrollo de software)* [en línea]. [Consultada: 06 de marzo de 2017]. Disponible en internet < https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software) >

[14] ISO (2015). *International Organization for Standardization* [en línea]. [Consultada: 04 de abril de 2017]. Disponible en internet < https://www.iso.org/standard/52075.html >

[15] IntelliJ IDEA (2017). *IntelliJ IDEA the Java IDE* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://www.jetbrains.com/idea/ >

[16] Spring (2017) *Spring Framework* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://spring.io >

[17] Firebase (2017) *Firebase* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://firebase.google.com/?hl=es-419 >
[18] Kotlin (2017) *Kotlin Programming Language* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://kotlinlang.org/ >
[19] Shared Preferences (2017) *Shared Preferences Android* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://developer.android.com/reference/android/content/SharedPreferences.html >
[20] GSON (2017) *GSON* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://github.com/google/gson >
[21] Retrofit (2017) *Retrofit – Square Open Source* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < http://square.github.io/retrofit/ >
[22] ButterKnife (2017) *Butter Knife* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < http://jakewharton.github.io/butterknife/ >
[23] EventBus (2017) *Greenrobot EventBus* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://github.com/greenrobot/EventBus >
[24] CiruclarMusicProgressBar (2017) *Circular music Progressbar* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet <https://github.com/aliab/circular-music-progressbar >
[25] Pulsator (2017) *Pulse animation for Andoird* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://github.com/booncol/Pulsator4Droid >
[26] FoldingCel (2017) *FoldingCell expanding content cell* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://github.com/Ramotion/folding-cell >
[27] HTextView (2017) *Animation effect to text* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://github.com/hanks-zyh/HTextView >
[28] Heroku (2017) *Heroku: Cloud Application Platform* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://www.heroku.com/ >
[29] MongoDB (2017) *MongoDB* [en línea]. [Consultada: 14 de mayo de 2017]. Disponible en internet < https://www.mongodb.com/es >

7. Anexos

7.1. Instrucciones de compilación

1. La compilación de la aplicación Android se realiza mediante Android Studio. Cuando se ejecuta Build > Build APK, se genera el fichero apk sin firma. La ubicación del fichero generado es la siguiente ubicación_del_ proyecto\app\build\outputs\apk\app-debug.apk
2. La compilación del servidor se realiza mediante Maven. En mi caso, cuando se ejecuta desde IntelliJ IDEA -> Maven Projects > package, se genera un fichero jar. La ubicación del fichero generado es la siguiente:

   ubicación_del_proyecto\target\kanguru-0.0.1-SNAPSHOT.jar
3. Para ejecutar la aplicación móvil sólo se debe instalar el fichero apk generado en el dispositivo móvil. Por defecto la aplicación está apuntando al servidor que hay en Heroku. Para cambiar el servidor se debe ir al fichero ServerRepository.java de la carpeta del proyecto app\src\main\java\es\wolfis\kanguru\repositories\server\ y se debe modificar la baseUrl.
4. Para ejecutar el servidor se debe ejecutar el siguiente comando “java -jar .\kanguru-0.0.1-SNAPSHOT.jar” en el servidor. Si no se modifica, el puerto por el que se pública el servicio es por el 8080. Hay una versión ya desplegada en Heroku:

   https://kanguru-uoc.herokuapp.com/

   (si se accede por navegador no hará nada ya que en la base ‘/’ no hay nada publicado, está publicado con el puerto 443).

Consideraciones importantes

Si se utiliza la versión desplegada en Heroku hay que tener las siguientes consideraciones:

- Cada 30 minutos de inactividad el servidor se apaga. Por lo que si el servidor se apaga cuando se haga una petición desde la aplicación móvil dará error (time out). En la segunda petición ya funcionará, ya que con la primera el servicio se vuelve arrancar.
- Cada vez que se apaga, los ficheros que se han ido generando/actualizando se borran (vuelven al estado original), por lo que todos los usuarios registrados y sus avances se pierden (limitación del servidor, en futuras versiones fuera del marco académico no pasará). Por defecto hay siempre un usuario registrado con nombre ericguti y contraseña 123456. Si se desea se puede registrar todos los que se desee teniendo en cuenta el apagado tras 30 minutos de inactividad.