



UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INGENIERÍA
INFORMÁTICA

GRADO EN INFORMÁTICA

TRABAJO FIN DE GRADO

English Pursuit: una aplicación Android para aprender inglés
y competir

Ángel José Luis Argandoña

Enero, 2014



UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INGENIERÍA
INFORMÁTICA

Departamento de Sistemas Informáticos

TRABAJO FIN DE GRADO

English Pursuit: una aplicación Android para aprender inglés
y competir

Autor: Ángel José Luis Argandoña

Directores: Víctor Manuel López Jaquero
Francisco Parreño Torres

Enero, 2014

Resumen

Este documento constituye la memoria escrita del Trabajo De Fin de Grado (TFG) de la titulación de Ingeniería Informática en la tecnología específica de Tecnologías de la información para la Escuela Superior de Ingeniería Informática de la Universidad de Castilla-La Mancha sita en Albacete.

En estos momentos estamos sumidos en una situación donde los dispositivos móviles (“smartphones”, “tablets” e incluso “wearables”) tienen una importante presencia en nuestra vida cotidiana, tanto es así que en la conferencia Google I/O 2014 Google ha anunciado que se ha superado la cota de mil millones de usuarios activos con Android. Por ello es un magnífico momento para llevar a cabo un desarrollo en este sistema operativo.

La aplicación elegida para desarrollar en este TFG tiene como propósito el poner a disposición de los usuarios una forma entretenida de aprender conceptos acerca del idioma más importante del mundo y que actualmente tanta importancia tiene para posicionarse en el mercado laboral.

Así, a lo largo de este documento se repasarán tanto aspectos básicos y fundamentales de Android, como conceptos específicos de su arquitectura y de la implementación de un sistema cliente - servidor que permitirá construir una aplicación multijugador.

Agradecimientos

A mi familia y compañeros que en todo momento me han apoyado y ayudado durante el Grado. Especial agradecimiento a mis tutores, Víctor Manuel López Jaquero y Francisco Parreño Torres, por su tiempo tanto en la revisión del proyecto como en la proposición de nuevas ideas.

Dedicatorias

Agradecimiento especial a mis profesores de la UCLM por formarme con enseñanzas de calidad. Y también a mi familia por permitirme alcanzar mis metas con su apoyo incondicional.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN.....	15
1.1 CONTEXTO.....	15
1.2 MOTIVACIÓN.....	16
1.3 OBJETIVOS	17
1.4 ESTRUCTURA DE LA MEMORIA	17
CAPÍTULO 2. ESTADO DEL ARTE	21
2.1 CONTEXTO ACTUAL DEL DESARROLLO DE APLICACIONES	21
2.2 APLICACIONES SIMILARES	23
2.2.1 ATRIVIA TE.....	23
2.2.2 PREGUNTADOS.....	24
2.2.3 ENGLISH GRAMMAR TESTS	26
CAPÍTULO 3. ANDROID.....	27
3.1 INTRODUCCIÓN	27
3.2 LOS ORÍGENES	28
3.3 ARQUITECTURA	28
3.3.1 NÚCLEO LINUX	29
3.3.2 RUNTIME DE ANDROID	29
3.3.3 BIBLIOTECAS NATIVAS.....	29
3.3.4 ENTORNO DE APLICACIÓN	29
3.3.5 APLICACIONES	30
3.4 ELEMENTOS DE UN PROYECTO	30
3.5 COMPONENTES DE UNA APLICACIÓN.....	31
3.5.1 VIEW.....	31
3.5.2 LAYOUT.....	31
3.5.3 ACTIVITY	31
3.5.4 SERVICE	31
3.5.5 INTENT.....	31
3.5.6 BROADCAST RECEIVER	32
3.5.7 CONTENT PROVIDER	32
CAPÍTULO 4. “GOOGLE CLOUD MESSAGING FOR ANDROID”	33
4.1 PREPARATIVOS PARA OBTENER ACCESO AL SERVICIO GCM.....	34
4.2 FUNCIONAMIENTO DEL SERVICIO GCM.....	36

CAPÍTULO 5. ESPECIFICACIÓN DE REQUISITOS SOFTWARE.....	39
5.1 OBJETIVOS	39
5.2 REQUISITOS FUNCIONALES.....	41
5.3 REQUISITOS NO FUNCIONALES.....	66
5.4 REQUISITOS DE INFORMACIÓN.....	69
5.5 DIAGRAMAS (ANÁLISIS Y DISEÑO).....	77
5.5.1 DIAGRAMA DE CASOS DE USO.....	77
5.5.2 DIAGRAMA DE CLASES	78
5.5.3 DIAGRAMAS DE SECUENCIA	79
5.5.4 GRÁFICO DE DESPLIEGUE	83
5.6 MODELO DE DATOS.....	84
5.6.1 MODELO ENTIDAD RELACIÓN	85
5.7 BASE DE DATOS.....	85
5.7.1 TABLAS.....	86
5.8 CONCLUSIONES	89
CAPÍTULO 6. HERRAMIENTAS DE TRABAJO.....	91
6.1 HERRAMIENTAS DE DISEÑO	91
6.1.1 DIA	92
6.1.2 MICROSOFT VISIO.....	92
6.2 HERRAMIENTAS DE DESARROLLO.....	93
6.2.1 JAVA DEVELOPMENT KIT	93
6.2.2 SDK	94
6.2.3 ECLIPSE.....	97
6.2.4 ANDROID STUDIO	98
CAPÍTULO 7. INTERFAZ GRÁFICA DE USUARIO	99
7.1 PANTALLA PRINCIPAL	101
7.2 MENÚ LATERAL (NAVIGATION DRAWER)	102
7.3 PANTALLA DE CLASIFICACIÓN.....	104
7.4 PANTALLA DE INSTRUCCIONES.....	104
7.5 PANTALLA DE PERFIL DE USUARIO.....	105
7.6 PANTALLA DE PREGUNTAS.....	106
7.7 PANTALLA DE BUSCAR AMIGOS	107
7.8 PANTALLA DE AJUSTES.....	108
7.9 RECURSOS MULTIMEDIA	108

7.9.1	RECURSOS GRÁFICOS.....	109
7.9.2	RECURSOS DE AUDIO	109
CAPÍTULO 8. SERVIDOR PHP.....		111
8.1	Base de datos (MySQL).....	112
8.2	PHP como interfaz de comunicación entre BBDD del servidor y cliente (Android)	112
CAPÍTULO 9. IMPLEMENTACIÓN DE FUNCIONALIDADES EN EL CLIENTE (ANDROID) Y SERVIDOR (PHP).....		115
9.1	NAVIGATION DRAWER (MENÚ DESLIZANTE LATERAL).....	115
9.2	PREFERENCIAS DE LA APLICACIÓN	116
9.3	Registro.....	119
9.4	Login.....	124
9.5	Buscar usuario.....	126
9.6	Añadir amigo	132
9.7	Crear partida	133
9.7.1	Contra amigo	133
9.7.2	Contra rival aleatorio	136
9.8	Eliminar amigo	137
9.9	Eliminar partida (rendirse).....	138
9.10	Ver perfil	139
9.11	Ver clasificación.....	140
9.12	Ver instrucciones.....	141
9.13	Dar sugerencias y opiniones.....	141
9.14	Contestar preguntas	142
9.14.1	Animación de botones.....	146
9.14.2	Parseador XML	146
9.15	Envío de notificaciones Push con Google Cloud Messaging.....	150
9.16	Conclusiones	151
CAPÍTULO 10. EVALUACIÓN DE LA APLICACIÓN		153
10.1	Preguntas de Etnografía	153
10.1.1	Género	154
10.1.2	Nivel de experiencia en Android.....	154
10.1.3	Nivel de inglés.....	155

10.2 Preguntas SUS.....	155
10.2.1 Resultados de la pregunta “Creo que me gustará utilizar esta aplicación con frecuencia”	156
10.2.2 Resultados de la pregunta “Encontré la aplicación innecesariamente compleja”	157
10.2.3 Resultados de la pregunta “Pensé que la aplicación sería más fácil de usar”	157
10.2.4 Resultados de la pregunta “Creo que voy a necesitar el apoyo de un experto para utilizar la aplicación”	158
10.2.5 Resultados de la pregunta “Me pareció que las diversas funciones de esta aplicación se integran bien”	158
10.2.6 Resultados de la pregunta “Pienso que hay muchos aspectos mejorables en esta aplicación”	159
10.2.7 Resultados de la pregunta “La mayoría de la gente podrá utilizar esta aplicación sin problemas”	159
10.2.8 Resultados de la pregunta “Encontré la aplicación difícil de usar”	160
10.2.9 Resultados de la pregunta “Me sentí seguro utilizando la aplicación”	160
10.2.10 Resultados de la pregunta “Necesito aprender muchas cosas antes de manejar bien con la aplicación”	161
10.3 Comentarios y sugerencias	161
CAPÍTULO 11. CONCLUSIONES Y PROPUESTAS.....	163
11.1 CONCLUSIONES.....	163
11.2 TRABAJO FUTURO Y POSIBLES AMPLIACIONES.....	165
BIBLIOGRAFÍA	167
GLOSARIO DE TÉRMINOS.....	169
CONTENIDO DEL CD	170
ANEXO A. MANUAL DE INSTALACIÓN EN ANDROID.....	171
ANEXO B. MANUAL DE USO	174

ÍNDICE DE FIGURAS

Figura 1: Mercado de desarrollo de aplicaciones móviles	22
Figura 2: Aplicación "Atriviate", Inicio partida y Estadísticas	23
Figura 3: Aplicación "Atriviate", Tablero de juego y Preguntas	24
Figura 4: Aplicación "Preguntados", Inicio partida y Estadísticas	25
Figura 5: Aplicación "Preguntados", Ruleta y Preguntas	25
Figura 6: Aplicación "English Grammar Tests"	26
Figura 7: Arquitectura de Android	28
Figura 8: Instalación de Google Play Services en el SDK	34
Figura 9: Inclusión de la librería Google Play Services en nuestro proyecto	34
Figura 10: Creación de un proyecto en la consola de Google API	35
Figura 11: Obtención del número de proyecto GCM	35
Figura 12: Habilitando el servicio GCM	35
Figura 13: Obtención del API Key GCM	36
Figura 14: Arquitectura GCM	37
Figura 15: Diagrama de casos de uso	78
Figura 16: Diagrama de clases	79
Figura 17: Diagrama de secuencia "Registrar usuario"	80
Figura 18: Diagrama de secuencia "Iniciar sesión"	81
Figura 19: Diagrama de secuencia "Contestar pregunta"	82
Figura 20: Diagrama de secuencia "Agregar amigo"	82
Figura 21: Gráfico de despliegue	83
Figura 22: Modelo Entidad-Relación	85
Figura 23: Tablas BBDD	86
Figura 24: Tabla "usuarios"	86
Figura 25: Datos de la tabla "usuarios"	87
Figura 26: Tabla "amigo_de"	87
Figura 27: Tabla "estadísticas"	87
Figura 28: Tabla "partidas"	88
Figura 29: Tabla "preguntas"	88
Figura 30: Datos tabla "preguntas"	89
Figura 31: Tabla "respuestas"	89
Figura 32: Datos tabla "respuestas"	89
Figura 33: Herramienta Dia	92
Figura 34: Herramienta Microsoft Visio	92
Figura 35: Logo JDK (Java Development Kit)	93
Figura 36: Android SDK	94

Figura 37: Gestor de paquetes de Android SDK.....	95
Figura 38: Eclipse.....	97
Figura 39: Android studio	98
Figura 40: Prototipo y diseño final de la IU de la pantalla principal	101
Figura 41: Navigation Drawer.....	102
Figura 42: Ejemplos de aplicaciones con Navigation Drawer	103
Figura 43: Prototipo y diseño final de la IU del menú lateral	103
Figura 44: Prototipo y diseño final de la IU de la pantalla de clasificación.....	104
Figura 45: Diseño final de la IU de la pantalla de instrucciones.....	105
Figura 46: Prototipo y diseño final de la IU de la pantalla de perfil	105
Figura 47: Prototipo de la IU de la pantalla de preguntas	106
Figura 48: Diseño de la IU de la pantalla de preguntas.....	107
Figura 49: Prototipo y diseño final de la IU de la pantalla de buscar amigos.....	107
Figura 50: Prototipo y diseño final de la IU de la pantalla de ajustes	108
Figura 51: Recursos gráficos	109
Figura 52: Logo PHP.....	112
Figura 53: Versiones PHP	113
Figura 54: Diseño de la activity "Register"	119
Figura 55: Atributos Clase "Register"	120
Figura 56: Listener botón registro	120
Figura 57: Clase “MD5encrypter”	121
Figura 58: Implementación función "registerStatus"	121
Figura 59: Script "adduser"	122
Figura 60: Script upload	123
Figura 61: Almacenar datos de usuario en "SharedPreferences"	123
Figura 62: Diseño activity "Login"	124
Figura 63: Implementación función "loginStatus"	125
Figura 64: Script "access"	125
Figura 65: Diseño activity "SearchFriendActivity"	126
Figura 66: Diseño adapter personalizado "Amigo".....	126
Figura 67: Layout (item_friend.xml) del adapter personalizado "Amigo"	127
Figura 68: Implementación Clase "Friend"	128
Figura 69: Atributos Clase "SearchFriendActivity"	128
Figura 70: Listener botón “buscarAmigo”	129
Figura 71: Script "searchfriend"	130
Figura 72: Llenando la lista con el “adapter” personalizado.....	130
Figura 73: Atributos Clase "AdapterFriends"	131
Figura 74: Constructor de la Clase "AdapterFriends"	131

Figura 75: Obteniendo objetos "Friend" en el Adapter	131
Figura 76: Diseño "Añadir amigo"	132
Figura 77: Listener botón añadir amigo	132
Figura 78: Script "addfriend".....	133
Figura 79: Script "createMatchWithFriend".....	134
Figura 80: Función php "createMatch" Parte 1	135
Figura 81: Función php "createMatch" Parte 2	136
Figura 82: Script createMatchWithRandom.....	137
Figura 83: Diseño "eliminar usuario"	137
Figura 84: Script "deletefriend".....	138
Figura 85: Función "surrenderMatch"	138
Figura 86: Diseño "ProfileFragment"	139
Figura 87: Función "viewStats"	139
Figura 88: Diseño "Ranking Fragment"	140
Figura 89: Función "viewRanking"	140
Figura 90: Diseño interfaz "Instrucciones".....	141
Figura 91: Código para mostrar las instrucciones	141
Figura 92: Código para mostrar la aplicación de correo	142
Figura 93: Diseño de "MatchActivity"	142
Figura 94: Atributos Clase "MatchActivity"	143
Figura 95: Listener de botones respuesta	143
Figura 96: Implementación función "checkIsCorrect"	144
Figura 97: Finalización de ronda	144
Figura 98: Petición al script "playMatch"	145
Figura 99: Script "playMatch".....	145
Figura 100: Animación de botones.....	146
Figura 101: Animación Tween de traslación.....	146
Figura 102: Pasando la url del xml al parseador	147
Figura 103: Creando objetos pregunta al iniciar la lectura del xml.....	148
Figura 104: Clase Pregunta.....	148
Figura 105: Estructura documento XML de una partida.....	149
Figura 106: Obteniendo información de las etiquetas	149
Figura 107: Añadiendo la pregunta a la lista	150
Figura 108: Función "gcmSend"	151
Figura 109: Pregunta sobre el género	154
Figura 110: Pregunta sobre el nivel de experiencia en Android.....	154
Figura 111: Pregunta sobre el nivel de inglés.....	155
Figura 112: Escala de respuesta a las preguntas.....	156

Figura 113: Diagrama de barras resultados "Creo que me gustaría utilizar esta aplicación con frecuencia"	156
Figura 114: Diagrama de barras resultados “Encontré la aplicación innecesariamente compleja”	157
Figura 115: Diagrama de barras resultados “Pensé que la aplicación sería más fácil de usar”	157
Figura 116: Diagrama de barras resultados "Creo que voy a necesitar el apoyo de un experto para utilizar la aplicación"	158
Figura 117: Diagrama de barras resultados "Me pareció que las diversas funciones de esta aplicación se integran bien"	158
Figura 118: Diagrama de barras resultados "Pienso que hay muchos aspectos mejorables en esta aplicación"	159
Figura 119: Diagrama de barras resultados " La mayoría de la gente podrá utilizar esta aplicación sin problemas"	159
Figura 120: Diagrama de barras resultados "Encontré la aplicación difícil de usar"	160
Figura 121: Diagrama de barras resultados "Me sentí seguro utilizando la aplicación"	160
Figura 122: Diagrama de barras resultados "Necesito aprender muchas cosas antes de manejarme bien con la aplicación"	161
Figura 123: English Pursuit en Google Play	172
Figura 124: English Pursuit en Google Play (web)	172

ÍNDICE DE TABLAS

Tabla 1: Cuota Sistemas Operativos móviles.....	16
Tabla 2: Objetivo Número 1.....	40
Tabla 3: Objetivo Número 2.....	41
Tabla 4: Requisito Funcional Número 1 "Registrar usuario"	44
Tabla 5: Requisito Funcional Número 2 "Iniciar Sesión"	46
Tabla 6: Requisito Funcional Número 3 "Cerrar Sesión"	48
Tabla 7: Requisito Funcional Número 4 "Ver Perfil"	49
Tabla 8: Requisito Funcional Número 5 "Crear Partida"	51
Tabla 9: Requisito Funcional Número 6 "Jugar Partida"	53
Tabla 10: Requisito Funcional Número 7 "Buscar Usuario"	55
Tabla 11: Requisito Funcional Número 8 "Contestar Pregunta"	57
Tabla 12: Requisito Funcional Número 9 "Rendirse"	58
Tabla 13: Requisito Funcional Número 10 "Agregar Amigo"	60
Tabla 14: Requisito Funcional Número 11 "Eliminar Amigo"	62
Tabla 15: Requisito Funcional Número 12 "Ver Amigos"	64
Tabla 16: Requisito Funcional Número 13 "Ver Clasificación"	65
Tabla 17: Requisito No Funcional Número 1 "Entorno de explotación"	66
Tabla 18: Requisito No Funcional Número 2 "Interfaz Usable"	67
Tabla 19: Requisito No Funcional Número 3 "Posibilidad de incluir preguntas de cualquier tema"	68
Tabla 20: Requisito No Funcional Número 4 "Posibilidad de incluir una mayor cantidad de preguntas y de modificar las existentes sin necesidad de actualización"	69
Tabla 21: Requisito De Información 1 "Información acerca de los usuarios"	71
Tabla 22: Requisito De Información 2 "Información acerca de las estadísticas"	72
Tabla 23: Requisito De Información 3 "Información acerca de las partidas"	73
Tabla 24: Requisito De Información 4 "Información acerca de las preguntas"	75
Tabla 25: Requisito De Información 5 "Información acerca de las respuestas"	76
Tabla 26: Requisito De Información 6 "Información acerca de las relaciones de amistad entre usuarios"	77

CAPÍTULO 1. INTRODUCCIÓN

En este primer capítulo se pretende introducir al lector en el contexto en el cual se inicia la especificación e implementación de la aplicación, transmitir las motivaciones que han animado al planteamiento del proyecto, así como especificar los objetivos que se pretenden alcanzar con su desarrollo.

Por último se especificará la estructura del proyecto para situar al lector y permitirle una mejor comprensión de este escrito.

1.1 CONTEXTO

En cuanto a la situación tecnológica en la que nos encontramos en el momento de empezar la redacción de esta memoria cabe destacar que en un porcentaje cercano a la mitad [1] de la población española de más de 15 años dispone de un “smartphone” y lo utiliza a diario para realizar actividades de lo más variado.

Pero el dato más espectacular se encuentra en lo referente a los sistemas operativos de estos “smartphone”, ya que en España, Android es el claro dominador con una presencia de un 92,1 % en los móviles vendidos en el último trimestre de 2013 [2].

%Cuota Sistemas Operativos sobre nuevos <i>Smartphones</i>	Dic - Feb 2012	Dic - Feb 2013
Android	70,9%	92,1%
iOS	5,9%	4,4%
Symbian	10,5%	1,1%
RIM	10,2%	1,0%
Windows	2,5%	0,9%
Otros	0,0%	0,5%
Total <i>Smartphones</i>	100%	100%

Fuente: Worldpanel ComTech

Tabla 1: Cuota Sistemas Operativos móviles

1.2 MOTIVACIÓN

Con este desarrollo se busca la adquisición de conocimiento acerca del sistema operativo Android, una plataforma que desde sus inicios se presentaba tan innovadora como arriesgada ya que por aquel entonces iOS estaba dando sus primeros pasos y Symbian gozaba de bastante éxito. Sin embargo, Android aspiraba a convertirse en una gran alternativa al postularse como un proyecto de código abierto.

Otra motivación que decanta la balanza hacia la elección de este proyecto es la importancia del inglés como idioma internacional, cuyo aprendizaje es sumamente conveniente.

En lo referente a las aplicaciones disponibles para este sistema operativo, según la web de "Google Play" (la tienda virtual oficial de Android) hay más de 700.000 aplicaciones y juegos [3].

En esta tienda virtual, los usuarios pueden descargar multitud de aplicaciones de acuerdo a sus necesidades, intereses y gustos. Lo que destaca es la facilidad para encontrar, descargar e instalar una aplicación.

Además los desarrolladores pueden subir sus creaciones fácilmente a esta tienda, por no hablar de las facilidades que Google les brinda para desarrollarlas, incluyendo: tutoriales escritos y en vídeo, entornos de desarrollo, el SDK, recursos de diseño y ejemplos implementados.

1.3 OBJETIVOS

El planteamiento inicial de este TFG junto con el afán y entusiasmo de obtener un trabajo final, concluso y de calidad, engloba una serie de objetivos que se quieren alcanzar a lo largo de todo el proceso de desarrollo. Dichos objetivos son los siguientes:

- Debido a que este trabajo se centra en el desarrollo de un juego para Android, la primera meta a la que se quiere llegar y superar es obtener un conocimiento global sobre este sistema operativo. Para ello se estudiará en qué se fundamenta, cómo ejecuta sus aplicaciones, así como qué mecanismos y herramientas están disponibles para su desarrollo.
- Otro objetivo que se persigue en este trabajo es profundizar algo más en los servicios que Google nos ofrece para facilitar el desarrollo de aplicaciones o, en este caso juegos, con “Google Play Games Services” o “Google Cloud Messaging”.
- Además, al haber cursado asignaturas de la modalidad específica de Tecnologías de la información, se debe hacer hincapié en los aspectos de diseño para lograr una interfaz adecuada con el contenido del juego. Así que otro objetivo propuesto es el obtener conocimiento sobre guías de estilo y patrones de diseño e interacción en Android.
- También se busca el desarrollo de un servidor que atienda peticiones de los clientes Android y realice consultas específicas en una base de datos para obtener información útil.
- Como último objetivo a mencionar en este apartado, se encuentra la práctica de la creación completa de una aplicación desde sus inicios hasta sus etapas finales, aplicando todo el saber útil aprendido a lo largo del grado.

1.4 ESTRUCTURA DE LA MEMORIA

Capítulo 1. Introducción

Es el presente capítulo y en él se recoge tanto el contexto en el que nos encontramos al iniciar el proyecto, como las motivaciones y objetivos que se pretenden alcanzar.

Capítulo 2. Estado del arte

En el segundo capítulo se mostrará un conjunto de aplicaciones y juegos con funcionalidades comunes al producto que se quiere conseguir con el desarrollo. Además se describirá la situación en la que se encuentra el desarrollo de aplicaciones móviles en el momento de realizar el proyecto.

Capítulo 3. Android

¿Qué nos ofrece Android? Este TFG tiene como objetivo el desarrollo de una aplicación para el sistema operativo Android. En este tercer capítulo podrá encontrarse un análisis enfocado a desarrolladores sobre la estructura interna de Android, los elementos de un proyecto, el núcleo sobre el que se construye, sus elementos y las bibliotecas que utiliza entre otras cosas.

Capítulo 4. Google Cloud Messaging For Android

En el cuarto capítulo se explicará qué es Google Cloud Messaging y para qué se utiliza. Se trata de un servicio de Google que permite el envío de notificaciones push a dispositivos y que será utilizado para la notificación de eventos en esta aplicación.

Capítulo 5. Especificación De Requisitos Software

En este quinto capítulo se recogen todos los objetivos y requisitos que tendrá la aplicación a desarrollar, entre ellos se encuentran los requisitos funcionales, no funcionales y de información. Esta especificación de requisitos constituye la base sobre la que se puede comenzar la implementación de un proyecto software. Además también se especifican los diagramas UML fruto de las etapas de análisis y diseño y que nos ayudarán en el desarrollo.

Capítulo 6. Herramientas De Trabajo

El sexto capítulo muestra una serie de herramientas que formarán nuestro entorno de trabajo y nos permitirán realizar la implementación del proyecto. Ejemplo de ellas son el SDK (Software Development Kit) de Google o el entorno de desarrollo Eclipse.

Capítulo 7. Interfaz Gráfica De Usuario

En el séptimo capítulo se abordará el diseño gráfico de la interfaz de usuario de la aplicación, una parte muy importante del proyecto donde habrá que tener en cuenta principios de usabilidad y accesibilidad generales y principios de diseño específicos para esta plataforma. Además podrán encontrarse los prototipos de cada pantalla así como el resultado final de las mismas.

Capítulo 8. Servidor PHP

En el octavo capítulo se hablará del servidor PHP que atenderá a las peticiones de los clientes Android y manipulará la base de datos para obtener e insertar nueva información.

Capítulo 9. Implementación De Funcionalidades En El Cliente (Android) Y Servidor (Php)

En este extenso capítulo reside el código de la mayor parte de la funcionalidad que realiza tanto la aplicación Android como el servidor. Se muestran los recursos usados y las estructuras de datos que almacenan la información a mostrar. Además se puede ver la colaboración entre los clientes y el servidor para alcanzar los objetivos planteados.

Capítulo 10. Evaluación de la aplicación

En el décimo capítulo se exponen los resultados recogidos mediante un cuestionario de usabilidad realizado a varias personas con el objetivo de medir y evaluar el grado de satisfacción de los usuarios en la utilización de la aplicación.

Capítulo 11. Conclusiones Y Propuestas

En el décimo y último capítulo se recogen las conclusiones extraídas al finalizar el proyecto y se proponen trabajos futuros que podrían continuar y mejorar el proyecto.

Apéndices

En los apéndices se recogen otros aspectos no contemplados en ningún capítulo, entre ellos: la publicación y promoción de la aplicación en Google Play, el procedimiento para la instalación de la misma en un dispositivo Android y un pequeño manual de usuario para iniciar el manejo de English Pursuit.

CAPÍTULO 2. ESTADO DEL ARTE

En este segundo capítulo se realizará un análisis sobre el contexto actual en que nos encontramos al inicio de este proyecto, desde el punto de vista de desarrolladores. Además se expondrán aplicaciones similares existentes para Android que han servido como ejemplo para el desarrollo de la aplicación que deseamos realizar. El objetivo es ver las similitudes de las mismas con nuestra aplicación en cuestión, para así aprovechar sus bondades y solucionar sus posibles debilidades (si las tuviera).

2.1 CONTEXTO ACTUAL DEL DESARROLLO DE APLICACIONES

Antes de tomar la primera y más importante decisión en un desarrollo software, la plataforma destino de dicho desarrollo, se debe hacer un análisis de mercado que revele el éxito y la rentabilidad que puede ofrecernos cada una de las alternativas actuales.

Por ello es primordial apoyarse en estudios fiables y exhaustivos realizados por fuentes de confianza. Una de estas fuentes es [4], quien realiza informes sobre la tendencia del mercado en cuanto al interés de los desarrolladores basándose en la participación de más de 120 países.

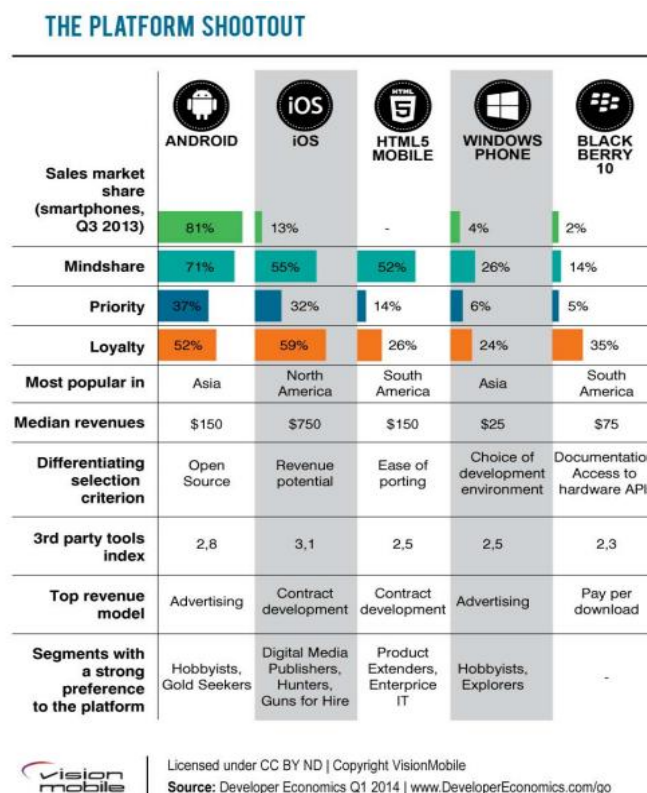


Figura 1: Mercado de desarrollo de aplicaciones móviles

Fijándonos en el primer dato, ventas de software en smartphones en el último trimestre de 2013, nos damos cuenta de que Android supera por mucho a sus competidores con un 81% de cuota, seguido por iOS con un 13%. De estos porcentajes, el 71% de los desarrolladores lo hicieron para Android y el 55% para el S.O. (Sistema Operativo) de Apple.

Como contrapartida, también puede apreciarse que iOS es el S.O. preferido en los países más desarrollados y que hay más desarrolladores que sólo se dedican a iOS (59%) que los que se dedican únicamente a Android (52%).

En conclusión, atendiendo a los datos reflejados en este estudio hemos decidido apostar por Android como plataforma destino de nuestro producto, debido a que este sistema operativo está presente en la mayoría de dispositivos móviles, por lo que podemos llegar a una mayor cantidad de usuarios.

2.2 APLICACIONES SIMILARES

En la actualidad existen múltiples juegos multijugador competitivos, cuyo objetivo es intentar demostrar una mayor posesión de conocimientos (historia, geografía, deportes, arte, ciencia, etc.) que el rival. Sin embargo no se ha encontrado ninguno cuya temática de competición sea el conocimiento sobre un idioma en particular.

2.2.1 ATRIVIA TE

Atriviate es un popular juego de preguntas (cuenta con más 5.000.000 instalaciones en dispositivos) muy similar al conocido juego de mesa "Trivial Pursuit" en el que dos jugadores se enfrentan para conseguir todos los "quesitos". Cada quesito representa a una categoría: geografía, deportes, espectáculos, arte, ciencia, historia y "de todo".



Figura 2: Aplicación "Atriviate", Inicio partida y Estadísticas

En este juego podremos crear una partida para jugar por turnos con otro jugador, bien con un amigo o bien con un oponente aleatorio.

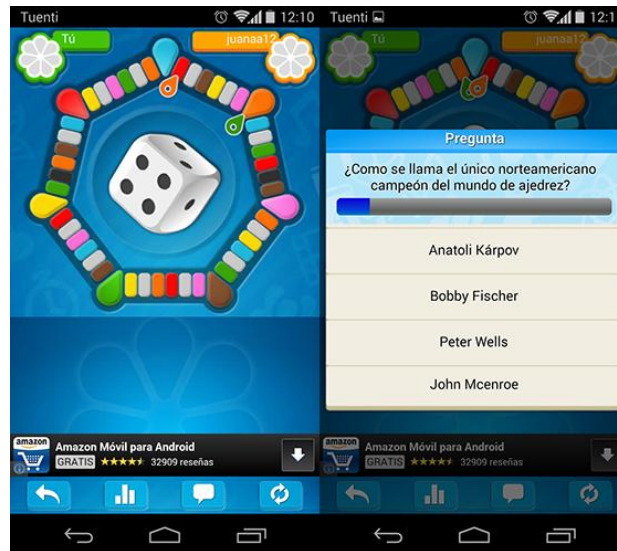


Figura 3: Aplicación "Atriviate", Tablero de juego y Preguntas

"Atriviate" cuenta con un tablero por donde ambos jugadores se irán moviendo en busca del "quesito" de cada una de las categorías.

Para conseguir este ansiado "quesito" el jugador tendrá que acertar la pregunta correspondiente al mismo. Si falla cualquier pregunta perderá el turno.

2.2.2 PREGUNTADOS

"Preguntados" es un juego muy similar al anterior (e incluso más popular con sus más de 10.000.000 de instalaciones) ya que los jugadores también deben enfrentarse y acertar preguntas para conseguir vencer al rival.

Sin embargo el modo de juego no es estilo "Trivial Pursuit", ni tampoco hay tablero.



Figura 4: Aplicación "Preguntados", Inicio partida y Estadísticas

Como en el juego anterior, también podemos crear una partida para jugar contra amigos (de Facebook o Google +) o contra rivales aleatorios.

Además, también nos ofrece la posibilidad de obtener las estadísticas respecto a nuestro porcentaje de acierto y victorias.



Figura 5: Aplicación "Preguntados", Ruleta y Preguntas

El elemento aleatorio que determinará la pregunta que el usuario debe responder es una ruleta, la cual contiene seis categorías diferentes: geografía, historia, arte, ciencia, entretenimiento y deportes.

Para conseguir la pieza de una categoría, el jugador tendrá que responder correctamente tres preguntas. Ante una respuesta incorrecta se cederá el turno al oponente

Finalmente, para ganar, un jugador tendrá que hacerse con las seis piezas que representan a cada categoría.

2.2.3 ENGLISH GRAMMAR TESTS

“English Grammar Tests” es un juego (bastante menos popular que los anteriores, con un número de instalaciones entre 100.000 y 500.000) de preguntas acerca de gramática inglesa, por lo que se acerca más al contenido de nuestra aplicación, sin embargo, carece de competitividad ya que no permite jugar contra otros jugadores, sólo contestar una batería de preguntas. Pero tiene la función de traducir cualquier palabra o frase a cualquier idioma utilizando Google Translate, aunque sabemos que este servicio no siempre realiza una buena traducción.

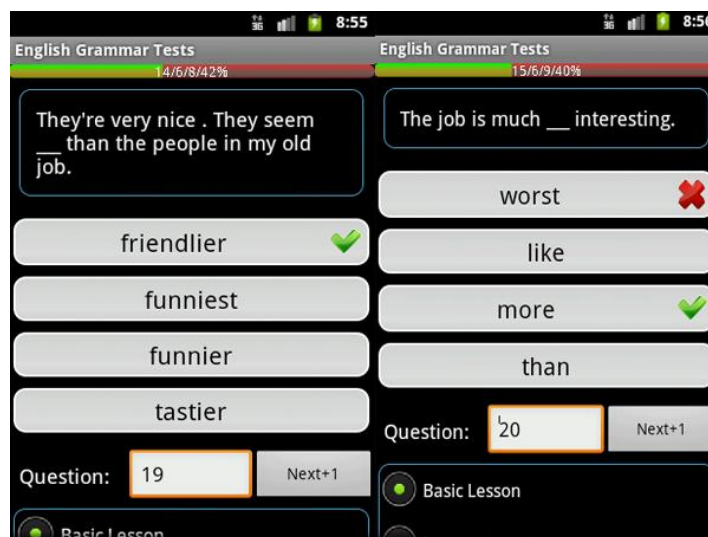


Figura 6: Aplicación "English Grammar Tests"

Habiendo visto estas tres aplicaciones, que de una forma u otra presentan aspectos y características parecidas al producto que se desea obtener, ahora es el momento de recoger ideas y construir una aplicación íntegra que combine las funcionalidades más provechosas de las mismas.

CAPÍTULO 3. ANDROID

Puesto que nuestra aplicación va a ser desarrollada para el sistema operativo Android, consideramos que debemos realizar una especial mención al mismo, describiéndolo y adentrándonos en profundidad en sus características, funcionalidades y arquitectura.

3.1 INTRODUCCIÓN

Es una realidad que los "smartphones" están suponiendo un cambio en los hábitos de la sociedad, se podría decir que supone una revolución de impacto similar al que tuvo la aparición de Internet en su época.

Cada vez podemos ver con más frecuencia la aparición de nuevos dispositivos con Android cuyas especificaciones se asemejan mucho más a los ordenadores que a los teléfonos móviles de antaño.

Existen dispositivos con gran capacidad de procesamiento (4 y 8 núcleos) y más memoria RAM (2 y 3 GB) que los ordenadores de hace apenas una década.

La principal diferencia y ventaja de estos nuevos dispositivos frente a los PC tradicionales es que siempre están disponibles y son accesibles al usuario en cualquier momento y lugar ya que éste suele llevarlos en su bolsillo.

Por último debemos destacar la amplia aceptación que este sistema operativo ha tenido entre los usuarios y la industria, ya que como veremos más adelante, Android es el S.O. móvil más utilizado en la actualidad.

3.2 LOS ORÍGENES

Como puede leerse en [5], Android, en sus inicios, fue desarrollado por Android Inc., la cual fue adquirida por Google en 2005, año en el que comienza el desarrollo de Dalvik (una máquina virtual Java optimizada para móviles), que en nuestros días va desapareciendo en favor de ART (otra máquina virtual más optimizada).

Varios años de desarrollo son los que transcurren y no es hasta finales de 2008 cuando Google lanza la primera versión comercial de Android (Android 1.1 “Apple Pie”). Desde ese momento hasta la actualidad, Google ha actualizado su S.O. una treintena de veces incluyendo nuevas funcionalidades y mejoras en cada una de ellas.

3.3 ARQUITECTURA

En el siguiente gráfico vamos a mostrar las diferentes capas de las que se compone la arquitectura. Una peculiaridad de las mismas es que están basadas íntegramente en **software libre**.

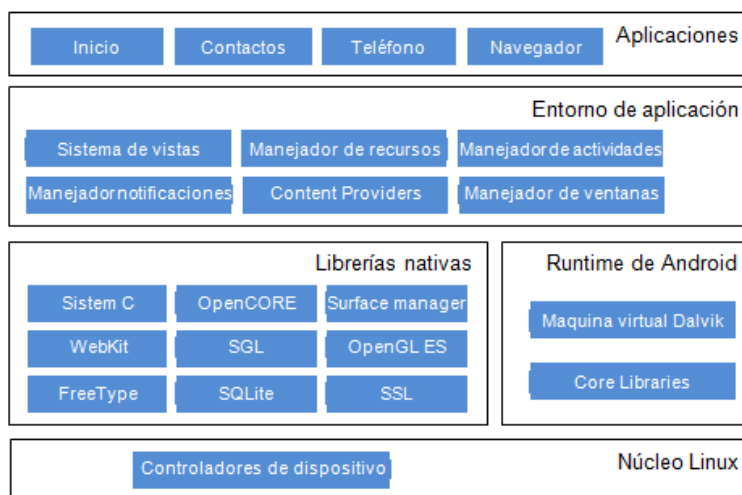


Figura 7: Arquitectura de Android

A continuación, vamos a describir brevemente cada una de las capas.

3.3.1 NÚCLEO LINUX

Android implementa Linux 3.4 en su núcleo [6], con el objetivo de proporcionar seguridad, gestión de memoria y procesos y soportar “drivers” para diferentes dispositivos.

La función de esta capa es proporcionar una abstracción entre hardware y software.

3.3.2 RUNTIME DE ANDROID

En esta capa destaca el uso de una máquina virtual Java (Dalvik) optimizada para ejecutarse en entornos con memoria y procesador limitados.

Sin embargo, como ya se ha dicho, a partir de la versión 4.4 (Kitkat) de Android existe una alternativa a Dalvik llamada ART. Además se cree que en la próxima versión de Android Dalvik se abandonará en favor a la última

En el Runtime también se incluye el “core libraries” donde se encuentran la mayoría de bibliotecas de Java.

3.3.3 BIBLIOTECAS NATIVAS

En esta capa se encuentra un conjunto de bibliotecas en C/C++ que el sistema operativo utiliza, entre las que se encuentran: codecs de audio, imágenes y vídeo, encriptación de datos, manejo de gráficos 2D y 3D y manejo de BBDD (Bases de datos) entre otras.

3.3.4 ENTORNO DE APLICACIÓN

Gracias a esta capa, los desarrolladores tienen acceso a los APIs que usan las aplicaciones base. Su diseño permite que unas aplicaciones puedan utilizar componentes de otras.

3.3.5 APLICACIONES

Esta última capa está formada por el conjunto de aplicaciones instaladas.

Para desarrollarlas podemos optar por el lenguaje Java (utilizando el SDK) o C/C++ (utilizando el NDK). En este caso se ha optado por utilizar el SDK de Android.

3.4 ELEMENTOS DE UN PROYECTO

A grandes rasgos, un proyecto Android está formado por un descriptor de aplicación (AndroidManifest.xml), el código fuente en Java y unos ficheros de recursos. Cada elemento se organiza en un directorio específico:

- src: carpeta que contiene el código fuente Java.
- gen: carpeta que contiene el código generado automáticamente por el SDK. Aquí se encuentra el fichero R.java, una clase que asocia los recursos de la aplicación con identificadores, para que dichos recursos puedan ser accesibles desde el código Java.
- Android Dependencies: librerías asociadas al proyecto.
- bin: directorio donde se compila el código y se genera el .apk listo para instalar en un dispositivo.
- res: contiene los recursos usados por la aplicación.
- drawable: almacena ficheros de imágenes y descriptores de imágenes en xml.
- layout: contiene los ficheros xml que definen la disposición de la interfaz de usuario de cada pantalla visual (activity o fragment).
- menú: contiene ficheros xml con los menús de cada activity.
- values: permite especificar valores de string, color o estilo. Además permite traducir la aplicación a otro idioma.
- anim: contiene ficheros xml con animaciones Tween.
- xml: otros ficheros xml requeridos por la aplicación.

3.5 COMPONENTES DE UNA APLICACIÓN

En este último apartado del capítulo se van a describir de forma breve los componentes de una aplicación Android, los cuales aportan diferentes funcionalidades al desarrollo:

3.5.1 VIEW

Son los elementos que componen la interfaz gráfica de una aplicación, por ejemplo: un botón, una etiqueta o una caja de texto. Pueden definirse en código Java o en un fichero XML.

3.5.2 LAYOUT

Se trata del agrupamiento de un conjunto de vistas para organizarlas y construir una interfaz clara y agradable.

3.5.3 ACTIVITY

Son lo que coloquialmente se conoce como “pantalla de una aplicación”. Constituyen la interfaz de usuario, la cual suele tener varias “Activities” independientes, pero que trabajan para un objetivo común.

3.5.4 SERVICE

Son procesos que se ejecutan en segundo plano (similar a los “demonios” de UNIX) sin necesidad de interacción con el usuario.

3.5.5 INTENT

Se refiere a la voluntad de realizar alguna acción, como por ejemplo lanzar una “Activity” o un “Service”.

3.5.6 BROADCAST RECEIVER

Es el sistema que utiliza Android para recibir y reaccionar frente a situaciones concretas, como por ejemplo cuando queda poca batería en el dispositivo o éste recibe una llamada.

3.5.7 CONTENT PROVIDER

Permite la compartición de información entre distintas aplicaciones. Un ejemplo puede verse en la popular aplicación de mensajería instantánea “Whatsapp”, desde la cual podemos acceder a los datos de nuestros contactos.

CAPÍTULO 4. “GOOGLE CLOUD MESSAGING FOR ANDROID”



Google Cloud Messaging, como su propio nombre indica es un servicio desarrollado por Google (dentro de sus APIS) que permite enviar datos mediante “la nube”. Ésto permite a los desarrolladores implementar fácilmente un método para enviar información de sus servidores a sus aplicaciones Android.

Este servicio realiza ese envío de información utilizando las llamadas notificaciones push tan utilizadas por diversas aplicaciones del ecosistema móvil para alertar al usuario de cualquier evento.

En concreto para este proyecto, las notificaciones push son un sistema muy útil y efectivo para enviar mensajes sobre acontecimientos relacionados con el juego. Por

ejemplo, cuando un usuario invita a otro para jugar una partida o cuando un usuario termina su turno y lo delega en el rival. De esta manera el usuario que deba ser notificado puede darse cuenta del aviso fácil e inmediatamente gracias al sonido, a la vibración, al LED de notificaciones o simplemente con un vistazo a la barra de notificaciones, es decir, de la misma manera que si se recibiera un correo electrónico o un SMS.

4.1 PREPARATIVOS PARA OBTENER ACCESO AL SERVICIO GCM

En este apartado vamos a ver cómo obtener los recursos y las claves necesarias para poder incluir este servicio tan útil en nuestra aplicación:

1. Lo primero de todo es instalar Google Play Services SDK, para lo cual ejecutamos el SDK Manager de nuestra máquina de desarrollo y buscamos dicho servicio.

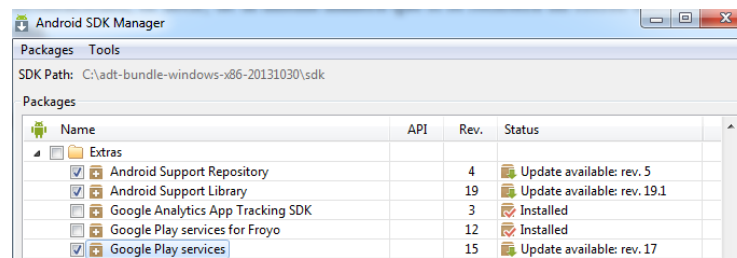


Figura 8: Instalación de Google Play Services en el SDK

2. Una vez instalado sólo tendremos que incluirlo como biblioteca en nuestro proyecto de eclipse.

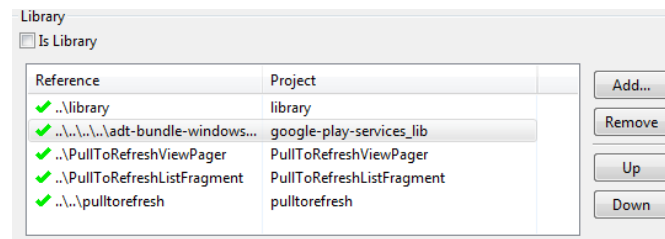


Figura 9: Inclusión de la librería Google Play Services en nuestro proyecto

3. Ahora que tenemos nuestro proyecto de eclipse configurado, es hora de crear un proyecto en la consola de Google API.

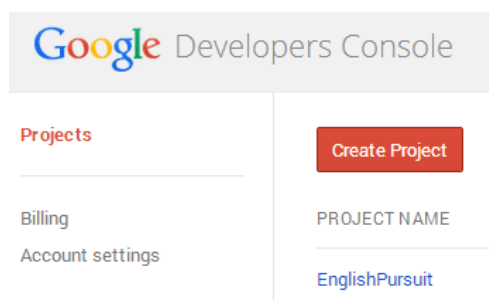


Figura 10: Creación de un proyecto en la consola de Google API

Al crear el proyecto obtendremos un número (Project Number) ,que no se muestra en esta captura (por motivos de seguridad) y que lo identificará unívocamente y nos servirá como “GCM sender ID” para identificar al servidor y permitir el envío de mensajes por parte de éste a otros dispositivos.

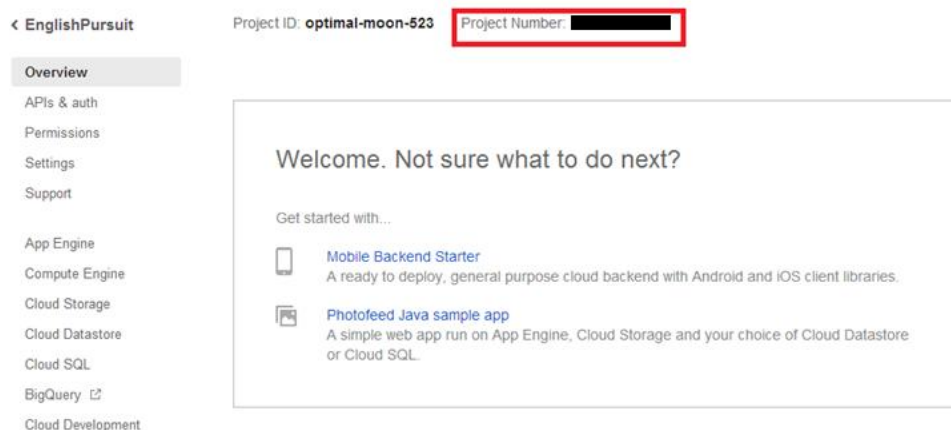


Figura 11: Obtención del número de proyecto GCM

4. El siguiente paso es habilitar el servicio GCM en nuestro proyecto. Para ello nos dirigimos a la opción “APIs” del menú lateral izquierdo.

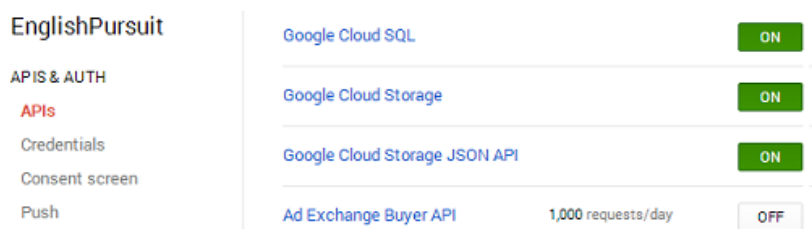


Figura 12: Habilitando el servicio GCM

5. El último paso es obtener el API Key para que nuestro servidor se pueda autenticar. Para ello seleccionamos la opción “Credentials”, después “create key” y finalmente “Server key”. Una vez obtenida, la guardamos para utilizarla cuando implementemos el servidor.

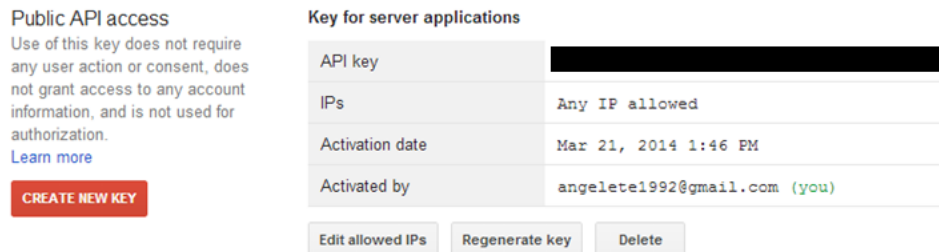
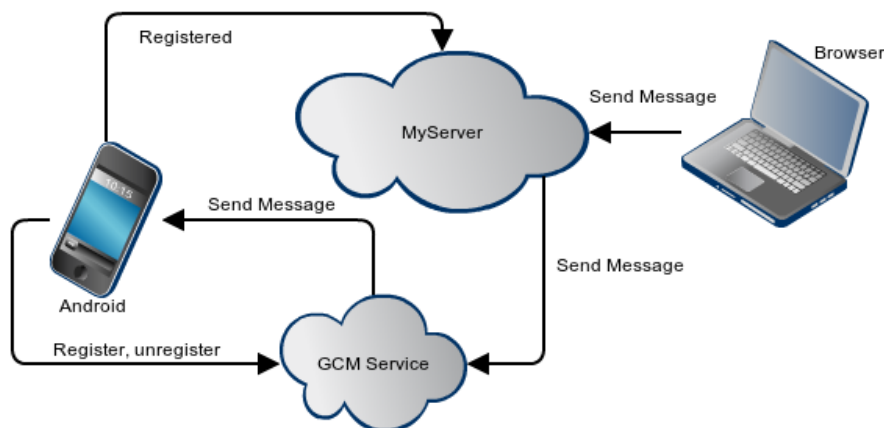


Figura 13: Obtención del API Key GCM

4.2 FUNCIONAMIENTO DEL SERVICIO GCM



Para que un dispositivo pueda recibir una notificación push, es necesario que éste se registre en el servicio, ya que debe tener una clave específica de comunicación. Para cumplir este propósito, la aplicación Android debe realizar estos pasos:

1. El dispositivo se registra en el servicio GCM (en nuestro caso se realizará cuando el usuario se registre o identifique en English Pursuit).

2. Si el dispositivo se registró con éxito en el paso anterior, GCM retorna la clave que lo identificará.
3. El dispositivo envía la clave al servidor de English Pursuit para que éste la almacene, permitiendo su posterior utilización.

Gracias a esta secuencia nuestro servidor dispondrá de las claves de los dispositivos de cada usuario, de manera que cuando quiera notificar a alguno de ellos de cualquier evento, podrá hacerlo fácilmente comunicándose con GCM y enviando como parámetro la clave del dispositivo concreto.

Finalmente el servicio GCM será el que envíe la notificación al dispositivo Android deseado.

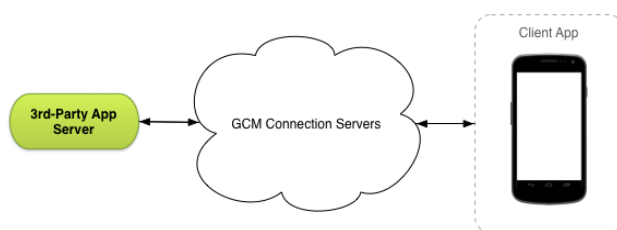


Figura 14: Arquitectura GCM

La mayor ventaja de este sistema de comunicación es que realiza un ínfimo consumo de recursos (y por tanto de energía) al ser mucho más apto para estos propósitos que otras alternativas como dejar una conexión TCP abierta permanentemente o hacer “polling” cada cierto tiempo en el servidor en busca de novedades.

En este capítulo se ha introducido y descrito el servicio GCM, su implementación se verá más adelante, (véase sección 9.15).

CAPÍTULO 5. ESPECIFICACIÓN DE REQUISITOS SOFTWARE

Como bien se recalca en las asignaturas de Ingeniería del software, la clave del éxito de un proyecto está en definir correctamente y desde un principio el conjunto de requisitos y objetivos que éste tendrá que cumplir. Por ello es de suma importancia realizar una fase de análisis y diseño previa antes de entrar en la fase de implementación, ya que de otra manera podrían ocurrir imprevistos y modificaciones que ralentizarían el desarrollo del proyecto y lo encarecerían.

Así, en este capítulo se presentan los resultados de las fases de adquisición de requisitos, análisis y diseño del proyecto.

5.1 OBJETIVOS

OBJ-01	Proporcionar un juego educativo y atractivo para los usuarios.
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero

	Francisco Parreño Torres Ángel José Luis Argandoña
Descripción	El usuario debe ser capaz de aprender conceptos de la lengua inglesa a medida que juega en un entorno cuidado y fácil de entender.
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Alta

Tabla 2: Objetivo Número 1

OBJ-02	Proporcionar un juego entretenido y competitivo.
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Descripción	El usuario debe poder ver las puntuaciones de otros jugadores y enfrentarse a ellos, así como divertirse intentando superarlos.

Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Alta

Tabla 3: Objetivo Número 2

5.2 REQUISITOS FUNCIONALES

UC-01	Registrar Usuario
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	--
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee crear una cuenta (registrarse) para jugar.
Precondición	El usuario no debe estar identificado con otra cuenta en la aplicación.

Secuencia Normal	Paso	Acción
	1	El usuario accede a la pantalla de registro (bien porque es la primera vez que ejecuta la aplicación, o bien porque ha cerrado sesión).
	2	El usuario introduce los datos que desee para su usuario y contraseña en las cajas de texto correspondiente, además de una imagen de perfil si lo desea.
	3	El usuario pulsa el botón “Registro” para enviar los datos de registro al servidor.
Postcondición	El usuario queda registrado en el sistema (se insertan las tupas necesarias en la tabla correspondiente de la BBDD). Los valores de “usuario” y “contraseña” son almacenados en las “sharedpreferences” de la aplicación. La aplicación muestra la pantalla principal del usuario.	
Excepciones	Paso	Acción
	3	Si los datos introducidos en los campos “Contraseña” y “Repetir contraseña” no coinciden, la aplicación muestra un aviso al usuario y vuelve al paso 2 de

		este caso de uso.
	3	Si el usuario deja algún campo obligatorio sin rellenar, la aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
	3	Si ya existe un usuario registrado en el sistema con el mismo “nombre de usuario”, la aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
	3	Si el dispositivo del usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	
Prioridad	Alta	

Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 4: Requisito Funcional Número 1 "Registrar usuario"

UC-02	Iniciar Sesión	
Versión	1.0	
Autores	Ángel José Luis Argandoña	
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña	
Objetivos Asociados	--	
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee identificarse en el sistema para jugar con su cuenta.	
Precondición	El usuario no debe estar identificado con ninguna cuenta en la aplicación, pero deberá tener una cuenta de usuario registrada previamente.	
Secuencia Normal	Paso	Acción

	1	El usuario accede a la pantalla de identificación.
	2	El usuario introduce sus datos personales correspondientes a su nombre de usuario y contraseña
	3	El usuario pulsa el botón “Login” para enviar los datos de identificación al servidor.
Postcondición	Los valores de “usuario” y “contraseña” son almacenados en las “sharedpreferences” de la aplicación. La aplicación muestra la pantalla principal del usuario.	
Excepciones	Paso	3
	3	Si los datos introducidos no se corresponden con los de ningún usuario registrado en el servidor, la aplicación muestra un aviso al usuario y vuelve al paso 2 de este caso de uso.
	3	Si el dispositivo del usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la

CAPÍTULO 5

		aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	
Prioridad	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	--	

Tabla 5: Requisito Funcional Número 2 "Iniciar Sesión"

UC-03	Cerrar Sesión
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	--

Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee cerrar su sesión de usuario actual.	
Precondición	El usuario debe estar identificado con alguna cuenta en la aplicación.	
Secuencia Normal	Paso	Acción
	1	El usuario accede a su perfil (UC-04).
	2	El usuario pulsa el botón “Cerrar sesión”.
Postcondición	La sesión del usuario termina. Los valores de “usuario” y “contraseña” son eliminados de las “sharedpreferences” de la aplicación. La aplicación muestra la pantalla de bienvenida.	
Excepciones	--	
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	
Prioridad	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	--	

Tabla 6: Requisito Funcional Número 3 "Cerrar Sesión"

UC-04	Ver Perfil	
Versión	1.0	
Autores	Ángel José Luis Argandoña	
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña	
Objetivos Asociados	OBJ-02 Proporcionar un juego entretenido y competitivo.	
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee ver su perfil.	
Precondición	El usuario debe estar identificado con alguna cuenta en la aplicación.	
Secuencia Normal	Paso	Acción
	1	El usuario accede a la pantalla asociada a su perfil seleccionando la opción pertinente del menú lateral izquierdo.

Postcondición	La aplicación muestra la pantalla del perfil del usuario, donde éste puede ver sus datos y estadísticas.
Excepciones	--
Rendimiento	1
Frecuencia	--
Importancia	Alta
Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 7: Requisito Funcional Número 4 "Ver Perfil"

UC-05	Crear Partida
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	OBJ-02 Proporcionar un juego entretenido

	y competitivo.	
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee crear una partida de juego.	
Precondición	El usuario debe estar identificado en la aplicación.	
Secuencia Normal	Paso	Acción
	1	El usuario selecciona la opción “Nueva Partida” de la pantalla principal y accede a la sección correspondiente.
	2	El usuario selecciona el rival con el que quiere jugar (amigo o aleatorio).
Postcondición	La partida es creada y se almacena en la tabla correspondiente de la BBDD. Una notificación push es enviada al dispositivo del rival para anunciarle el evento.	
Excepciones	Paso	Acción
	2	Si el usuario selecciona “Jugar con amigo”, se abrirá otra pantalla en la que el usuario tendrá que elegir el amigo con el que

		jugar.
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	
Prioridad	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	--	

Tabla 8: Requisito Funcional Número 5 "Crear Partida"

UC-06	Jugar Partida
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	OBJ-01 Proporcionar un juego educativo y atractivo para los usuarios. OBJ-02 Proporcionar un juego entretenido

	y competitivo.	
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee jugar una partida en curso.	
Precondición	El usuario debe estar identificado en la aplicación. El usuario debe tener en curso la partida a la que desee acceder. El usuario debe disponer del turno de la partida.	
Secuencia Normal	Paso	Acción
	1	El usuario accede a la pantalla principal de la aplicación
	2	El usuario selecciona la partida correspondiente y entra en ella.
Postcondición	El usuario accede a la partida en curso.	
Excepciones	Paso	Acción
	1	Si el usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la aplicación muestra un aviso y vuelve al paso 1 de este caso de uso.

	1	Si el turno de la partida a la que quiere acceder el usuario no le corresponde, la aplicación muestra un aviso y vuelve al paso 1 de este caso de uso.
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	
Prioridad	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	--	

Tabla 9: Requisito Funcional Número 6 "Jugar Partida"

UC-07	Buscar Usuario
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña

Objetivos Asociados	OBJ-02 Proporcionar un juego entretenido y competitivo.	
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee buscar a otro usuario en concreto.	
Precondición	El usuario debe estar identificado en la aplicación	
Secuencia Normal	Paso	Acción
	1	El usuario accede a la pantalla de “Jugar con amigo” a través del UC-05 (Crear Partida).
	2	El usuario introduce el nombre de usuario de su amigo en el campo de texto correspondiente y pulsa el botón “Buscar”.
Postcondición	La aplicación muestra los usuarios que correspondan con la búsqueda.	
Excepciones	Paso	Acción
	2	Si el usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
Rendimiento	1	

Frecuencia	--
Importancia	Alta
Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 10: Requisito Funcional Número 7 "Buscar Usuario"

UC-08	Contestar Pregunta
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	OBJ-01 Proporcionar un juego educativo y atractivo para los usuarios. OBJ-02 Proporcionar un juego entretenido y competitivo.
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee cerrar su sesión de usuario actual.
Precondición	El usuario debe estar identificado en la

	aplicación. El usuario debe tener en curso la partida con la que se corresponden las preguntas a contestar y además debe poseer el turno.	
Secuencia Normal	Paso	Acción
	1	El usuario accede a la partida deseada (UC-06).
	2	El usuario visualiza la pregunta y respuestas correspondientes, y pulsa sobre la respuesta que desee contestar.
Postcondición	La aplicación muestra al usuario si su respuesta ha sido acertada o no y registra este suceso para hacer cálculos relativos a la partida.	
Excepciones	Paso	Acción
	2	Si el usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
Rendimiento	1	

Frecuencia	--
Importancia	Alta
Prioridad	Alta
Estado	Validada
Estabilidad	Alta
Comentarios	--

Tabla 11: Requisito Funcional Número 8 "Contestar Pregunta"

UC-09	Rendirse
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee rendirse en una partida.
Precondición	El usuario debe estar identificado en la aplicación. El usuario debe tener en curso la partida en la que quiere presentar su rendición.

Secuencia Normal	Paso	Acción
	1	El usuario se rinde seleccionando la opción “Rendirse” en la partida correspondiente de la pantalla principal.
Postcondición	Las tablas correspondientes de la BBDD son actualizadas (partidas, estadísticas...).	
Excepciones	Paso	Acción
	1	Si el usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la aplicación muestra un aviso y vuelve al paso 1 de este caso de uso.
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	
Prioridad	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	--	

Tabla 12: Requisito Funcional Número 9 "Rendirse"

UC-10	Agregar Amigo	
Versión	1.0	
Autores	Ángel José Luis Argandoña	
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña	
Objetivos Asociados		
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee agregar a un usuario a su lista de amigos.	
Precondición	El usuario debe estar identificado en la aplicación.	
Secuencia Normal	Paso	Acción
	1	El usuario accede a la pantalla de “Buscar Usuario” (correspondiente al UC-07) y ejecuta la búsqueda pertinente.
	2	Cuando la aplicación haya mostrado las coincidencias de búsqueda, el usuario agregará al amigo deseado seleccionando la opción “Agregar Amigo”.

Postcondición	El amigo agregado queda guardado en la lista de amigos del usuario. La tabla correspondiente de la BBDD es actualizada.	
Excepciones	Paso	Acción
	2	Si el usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	
Prioridad	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	--	

Tabla 13: Requisito Funcional Número 10 "Agregar Amigo"

UC-11	Eliminar Amigo
Versión	1.0

Autores	Ángel José Luis Argandoña	
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña	
Objetivos Asociados		
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee eliminar a un usuario a su lista de amigos.	
Precondición	El usuario debe estar identificado en la aplicación.	
Secuencia Normal	Paso	Acción
	1	El usuario accede a la pantalla de “Amigos”.
	2	Cuando la aplicación haya mostrado el conjunto de amigos del usuario, éste eliminará al que desee seleccionando la opción “Eliminar Amigo”.
Postcondición	El amigo eliminado es borrado de la lista de amigos del usuario. La tabla correspondiente de la BBDD es actualizada.	
Excepciones	Paso	Acción

	2	Si el usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	
Prioridad	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	--	

Tabla 14: Requisito Funcional Número 11 "Eliminar Amigo"

UC-12	Ver Amigos
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña

Objetivos Asociados		
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee ver los amigos que tiene agregados.	
Precondición	El usuario debe estar identificado en la aplicación.	
Secuencia Normal	Paso	Acción
	1	El usuario accede a la pantalla de “Amigos”.
	2	La aplicación muestra el conjunto de amigos del usuario.
Postcondición	--	
Excepciones	Paso	Acción
	2	Si el usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	

Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 15: Requisito Funcional Número 12 "Ver Amigos"

UC-13	Ver Clasificación	
Versión	1.0	
Autores	Ángel José Luis Argandoña	
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña	
Objetivos Asociados		
Descripción	La aplicación deberá seguir la siguiente secuencia de pasos cuando el usuario desee ver la clasificación de los mejores jugadores.	
Precondición	El usuario debe estar identificado en la aplicación.	
Secuencia Normal	Paso	Acción
	1	El usuario accede a la pantalla de

		“Clasificación”.
	2	La aplicación muestra a los diez mejores usuarios, según el número de partidas ganadas.
Postcondición	--	
Excepciones	Paso	Acción
	2	Si el usuario no dispone conexión a internet (bien por wifi o datos) a la hora de realizar la acción, la aplicación muestra un aviso y vuelve al paso 2 de este caso de uso.
Rendimiento	1	
Frecuencia	--	
Importancia	Alta	
Prioridad	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	--	

Tabla 16: Requisito Funcional Número 13 "Ver Clasificación"

5.3 REQUISITOS NO FUNCIONALES

RNF-01	Entorno de explotación
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	
Descripción	La aplicación deberá ser capaz de ejecutarse en un dispositivo Android con características básicas, es decir, los requerimientos mínimos de la aplicación serán bajos.
Importancia	Alta
Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 17: Requisito No Funcional Número 1 "Entorno de explotación"

RNF-02	Interfaz Usable
Versión	1.0

Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	
Descripción	La interfaz de la aplicación deberá respetar los aspectos de usabilidad y los principios de diseño para dispositivos táctiles permitiendo mostrar la información de la forma más comprensible y clara posible. Para conseguir este requisito se pueden seguir las guías de estilo para Android y los principios de diseño de Google.
Importancia	Alta
Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 18: Requisito No Funcional Número 2 "Interfaz Usable"

RNF-03	Posibilidad de incluir preguntas de cualquier tema
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero

	Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	
Descripción	Aunque esta aplicación esté enfocada al aprendizaje de la lengua inglesa, debe desarrollarse con la intención de, en un futuro, incluir preguntas de más categorías de inglés o incluso de otros idiomas o parcelas del saber. Por ello se utiliza una base de datos externa que almacena todas las preguntas, pudiendo ser modificadas en cualquier momento sin tener que actualizar la aplicación Android del cliente.
Importancia	Alta
Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 19: Requisito No Funcional Número 3 "Posibilidad de incluir preguntas de cualquier tema"

RNF-04	Posibilidad de incluir una mayor cantidad de preguntas y de modificar las existentes sin necesidad de actualización.
Versión	1.0
Autores	Ángel José Luis Argandoña

Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	
Descripción	Este requisito tiene una estrecha relación con el anterior (RNF-03) e incluye la posibilidad de insertar nuevas preguntas, eliminar las antiguas o modificar las existentes de forma transparente al usuario, sin que éste tenga que instalar una actualización. La solución es la base de datos externa comentada en el requisito anterior.
Importancia	Alta
Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 20: Requisito No Funcional Número 4 "Posibilidad de incluir una mayor cantidad de preguntas y de modificar las existentes sin necesidad de actualización"

5.4 REQUISITOS DE INFORMACIÓN

RI-01	Información acerca de los usuarios
Versión	1.0
Autores	Ángel José Luis Argandoña

Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	
Requisitos Asociados	UC-01, Registrar usuario UC-02, Iniciar sesión UC-03, Ver perfil UC-07, Buscar usuario UC-10, Agregar amigo
Descripción	La aplicación deberá almacenar y mantener los datos de cada uno de los usuarios registrados.
Datos Específicos	• Identificador de usuario • Alias (nombre de usuario) • Contraseña (hash MD5) • Código GCM (para el envío de notificaciones push) • Imagen de perfil
Tiempo De Vida	--
Ocurrencias Simultáneas	--
Importancia	Alta
Prioridad	Alta
Estado	Validado

Estabilidad	Alta
Comentarios	--

Tabla 21: Requisito De Información 1 "Información acerca de los usuarios"

RI-02	Información acerca de las estadísticas
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	OBJ-02, Proporcionar un juego entretenido y competitivo.
Requisitos Asociados	UC-04, Ver perfil UC-13, Ver clasificación
Descripción	La aplicación deberá almacenar y mantener los datos de las estadísticas de los usuarios.
Datos Específicos	• Identificador de estadística • Identificador del usuario al que pertenecen la estadísticas • Número de respuestas correctas • Número de respuestas incorrectas • Número de partidas ganadas • Número de partidas perdidas
Tiempo De Vida	--
Ocurrencias Simultáneas	--

Importancia	Alta
Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 22: Requisito De Información 2 "Información acerca de las estadísticas"

RI-03	Información acerca de las partidas
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	OBJ-02, Proporcionar un juego entretenido y competitivo.
Requisitos Asociados	UC-05, Crear partida UC-06, Jugar partida UC-08, Contestar pregunta UC-09, Rendirse
Descripción	La aplicación deberá almacenar y mantener los datos de cada una de las partidas

	creadas por los usuarios.
Datos Específicos	• Identificador de partida • Fecha de creación de la partida (para, en un futuro, eliminarla cuando caduque) • Identificadores de los usuarios participantes • Respuestas correctas de ambos jugadores • Ronda en la que se encuentra la partida • Jugador que posee el turno
Tiempo De Vida	--
Ocurrencias Simultáneas	--
Importancia	Alta
Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 23: Requisito De Información 3 "Información acerca de las partidas"

RI-04	Información acerca de las preguntas
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres

	Ángel José Luis Argandoña
Objetivos Asociados	OBJ-01, Proporcionar un juego educativo y atractivo para los usuarios. OBJ-02, Proporcionar un juego entretenido y competitivo.
Requisitos Asociados	UC-08, Contestar pregunta RNF-03, Posibilidad de incluir preguntas de cualquier tema RNF-04, Posibilidad de incluir una mayor cantidad de preguntas y de modificar las existentes sin necesidad de actualización
Descripción	La aplicación deberá almacenar y mantener los datos de cada una de las preguntas que pueden ser formuladas en las partidas.
Datos Específicos	• Identificador de pregunta • Enunciado • Traducción • Categoría • Identificador de las cuatro posibles respuestas • Identificador de la respuesta correcta
Tiempo De Vida	--
Ourrencias Simultáneas	--
Importancia	Alta
Prioridad	Alta
Estado	Validado

Estabilidad	Alta
Comentarios	--

Tabla 24: Requisito De Información 4 "Información acerca de las preguntas"

RI-05	Información acerca de las respuestas
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	OBJ-01, Proporcionar un juego educativo y atractivo para los usuarios. OBJ-02, Proporcionar un juego entretenido y competitivo.
Requisitos Asociados	UC-08, Contestar pregunta
Descripción	La aplicación deberá almacenar y mantener los datos de cada una de las respuestas que pueden ser propuestas como solución a las preguntas de las partidas.
Datos Específicos	• Identificador de respuesta • Enunciado
Tiempo De Vida	--
Ocurrencias Simultáneas	--
Importancia	Alta

Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 25: Requisito De Información 5 "Información acerca de las respuestas"

RI-06	Información acerca de las relaciones de amistad entre usuarios
Versión	1.0
Autores	Ángel José Luis Argandoña
Fuentes	Víctor Manuel López Jaquero Francisco Parreño Torres Ángel José Luis Argandoña
Objetivos Asociados	OBJ-02, Proporcionar un juego entretenido y competitivo.
Requisitos Asociados	UC-10, Agregar amigo UC-11, Eliminar amigo UC-12, Ver amigos
Descripción	La aplicación deberá almacenar y mantener los datos de cada una de las relaciones de amistad entre los usuarios.
Datos Específicos	Identificador de los dos jugadores que participan en la relación

Tiempo De Vida	--
Ocurrencias Simultáneas	--
Importancia	Alta
Prioridad	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	--

Tabla 26: Requisito De Información 6 "Información acerca de las relaciones de amistad entre usuarios"

5.5 DIAGRAMAS (ANÁLISIS Y DISEÑO)

En este punto, después de haber especificado los tres tipos de requisitos más importantes y teniendo ya una idea de cómo actuará la aplicación, se van a mostrar los diagramas UML más representativos y los que más ayuda prestan en el desarrollo de este proyecto software.

De este modo, los diagramas representan gráficamente una vista de una parte del sistema, aportando diferentes perspectivas y diferentes niveles de detalle que facilitan la comprensión del sistema

5.5.1 DIAGRAMA DE CASOS DE USO

El diagrama de casos de uso engloba y representa todas las acciones que el usuario puede realizar con la aplicación.

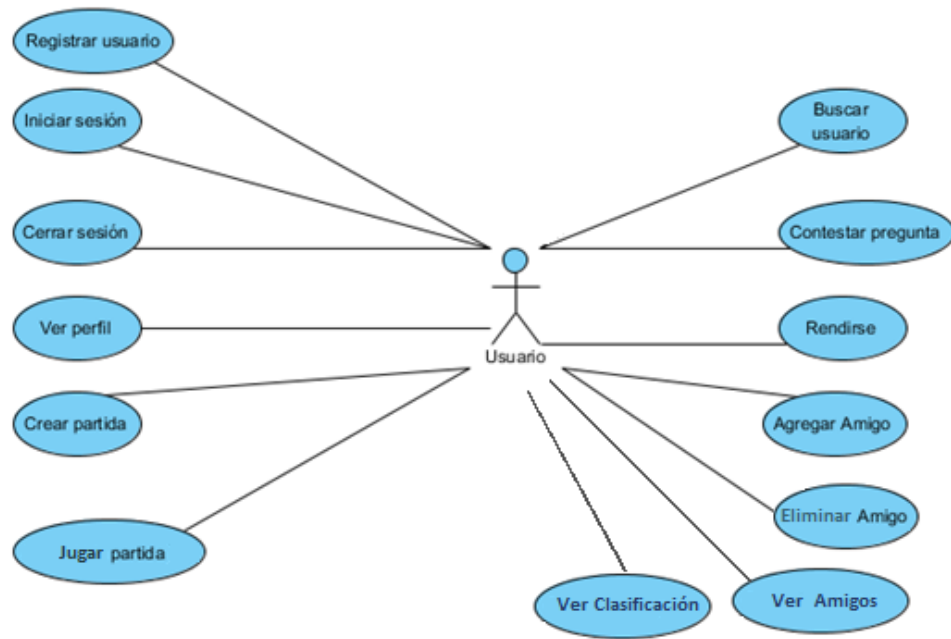


Figura 15: Diagrama de casos de uso

5.5.2 DIAGRAMA DE CLASES

Los diagramas de clases muestran las clases, interfaces y colaboraciones, así como sus relaciones. Cubren la vista de estructura estática de un sistema.

El diagrama de clases correspondiente a este proyecto software se muestra en la siguiente figura:

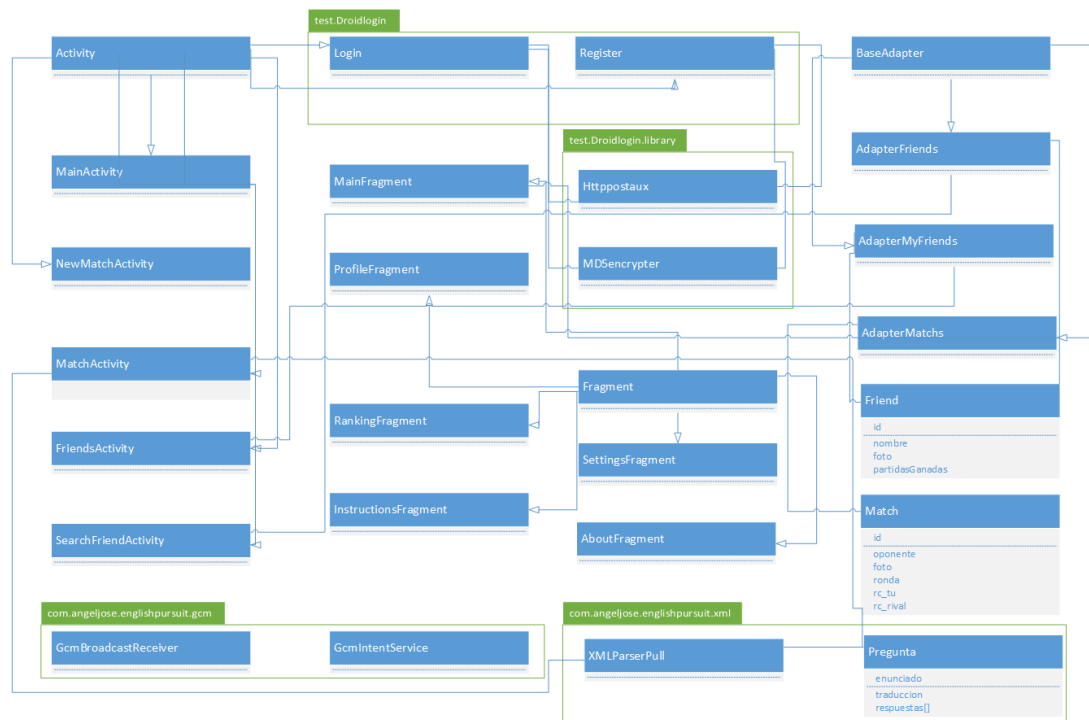


Figura 16: Diagrama de clases

5.5.3 DIAGRAMAS DE SECUENCIA

Los diagramas de secuencia, junto con los diagramas de comunicación constituyen los dos tipos de diagrama de interacción. En mi opinión, los primeros son los más representativos ya que muestran, con orden temporal, las relaciones y los mensajes intercambiados entre los elementos de la aplicación que interaccionan para llevar a cabo una tarea específica.

Debido al gran tamaño de estos diagramas, puede que sobre este documento no se aprecie la información de los mismos con el debido detalle, por ello estos mismos diagramas se encuentran con mayor calidad en el directorio “Diagramas” del CD.

Diagrama de secuencia del UC-01, “Registrar usuario”

El diagrama de secuencia mostrado en la siguiente figura representa las relaciones y el paso de mensajes entre los elementos necesarios para efectuar el registro de un usuario.

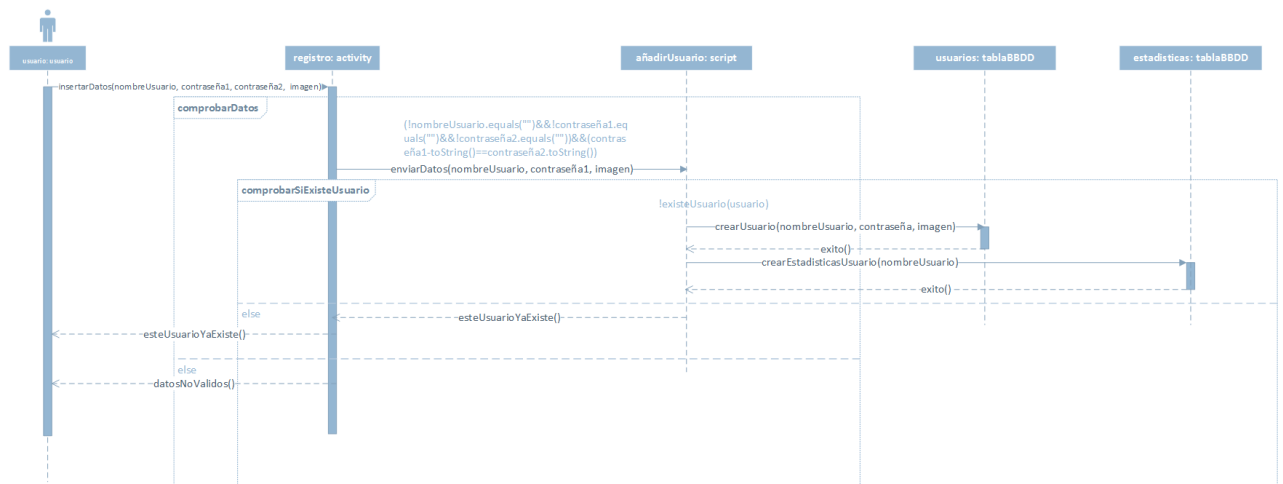


Figura 17: Diagrama de secuencia "Registrar usuario"

Puede apreciarse que es el usuario quien inicia la acción y el que rellena los datos de la activity. Posteriormente la lógica de la aplicación comprueba que los datos son correctos y de ser así llama al script del servidor encargado de hacer el registro, si los datos introducidos son incorrectos, no llega a llamarse al script y se le notifica al usuario. Por último, es el script quien comprueba que no haya un usuario registrado con el mismo nombre, si se da el caso favorable, se crean las tuplas necesarias en la tabla usuarios y estadísticas.

Diagrama de secuencia del UC-02, "Iniciar sesión"

El diagrama de secuencia mostrado en la siguiente figura representa las relaciones y el paso de mensajes que se producen entre los elementos necesarios cuando un usuario inicia sesión en la aplicación.

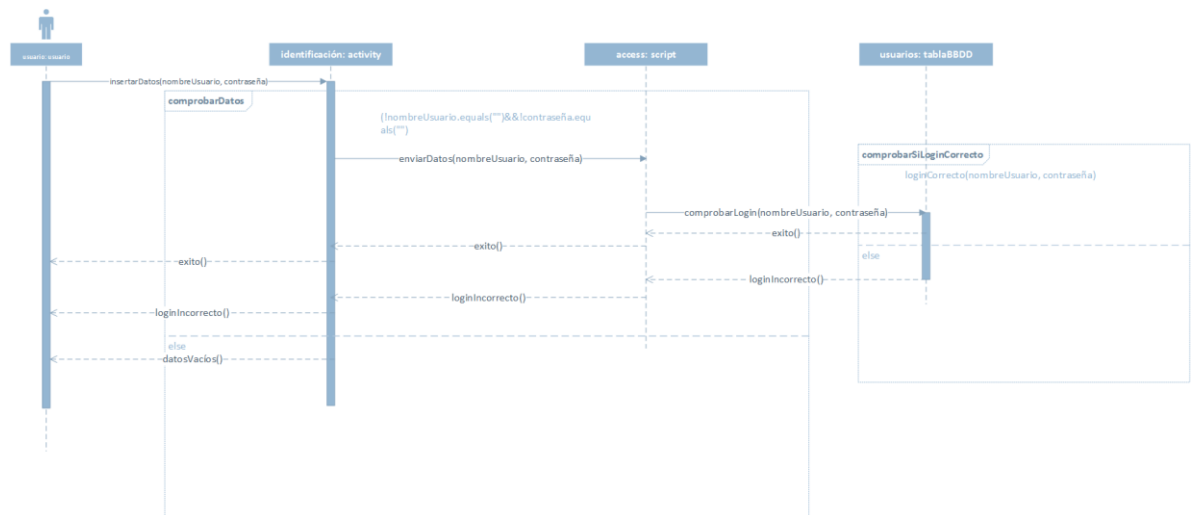


Figura 18: Diagrama de secuencia "Iniciar sesión"

Para iniciar sesión, el usuario rellena los dos campos existentes con el nombre que utilizó para registrarse y con su contraseña respectivamente. Una vez que lo ha hecho, se comprueba que ningún campo esté vacío y se procede al envío de los datos al servidor para comprobar que son correctos, de ser así los datos se almacenan en los “Shared preferences” de la aplicación y se muestra la pantalla principal con las partidas correspondientes al jugador.

Diagrama de secuencia del UC-08, “Contestar pregunta”

El diagrama de secuencia mostrado en la siguiente figura representa las relaciones y el paso de mensajes que se producen entre los elementos necesarios cuando un usuario procede a contestar las preguntas de la ronda de una partida.

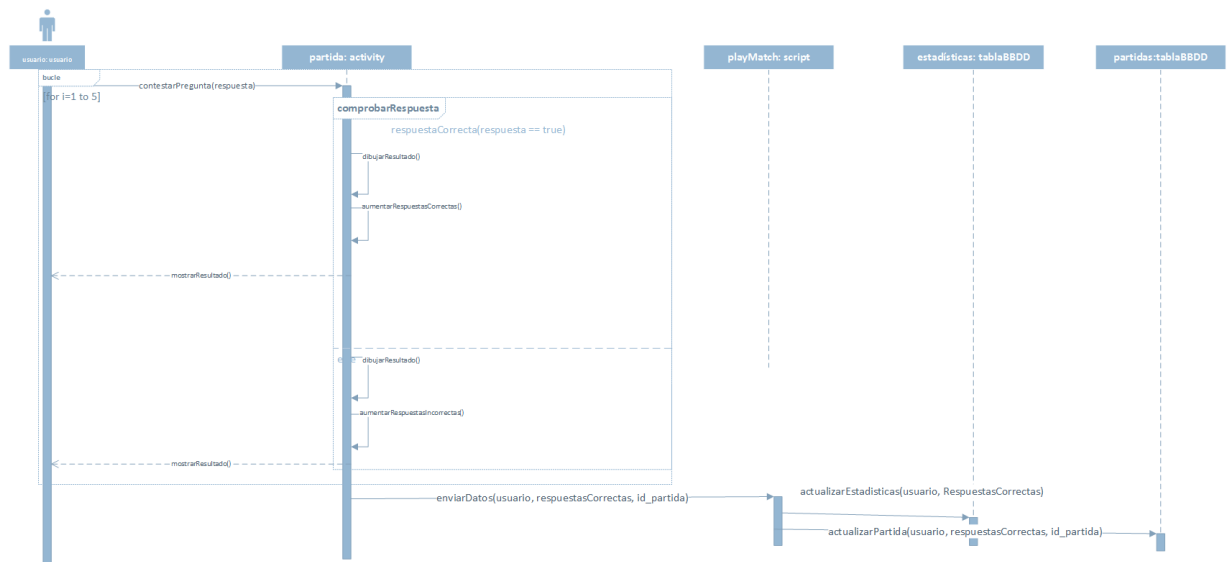


Figura 19: Diagrama de secuencia "Contestar pregunta"

Para contestar una pregunta, un usuario selecciona la respuesta que cree conveniente y la aplicación muestra la respuesta correcta. Además actualiza la puntuación de respuestas, dependiendo de si el jugador ha acertado o fallado. Finalmente, cuando acaba la ronda, se envían los resultados al servidor para que éste los almacene en la base de datos.

Diagrama de secuencia del UC-10, "Agregar amigo"

El diagrama de secuencia mostrado en la siguiente figura representa las relaciones y el paso de mensajes que se producen entre los elementos necesarios cuando un usuario agrega a otro jugador como amigo.

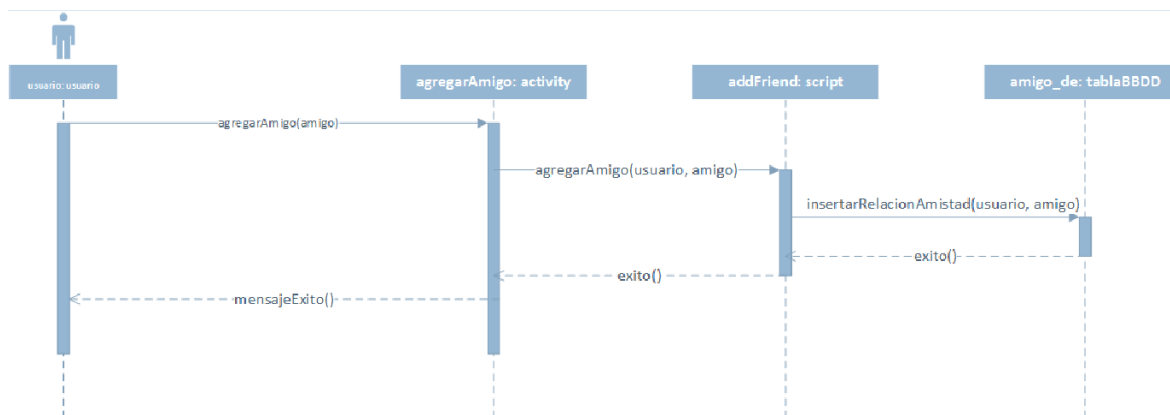


Figura 20: Diagrama de secuencia "Agregar amigo"

Para agregar a un amigo, un usuario debe encontrarse en la pantalla de buscar usuario o de clasificación, una vez ahí, lo único que debe hacer es pulsar el botón de “agregar como amigo” en el usuario que éste desee añadir. De este modo se enviarán los datos al servidor y se establecerá la relación de amistad adecuada en la base de datos.

5.5.4 GRÁFICO DE DESPLIEGUE

El siguiente gráfico muestra el despliegue de las clases representadas en el diagrama de clases:

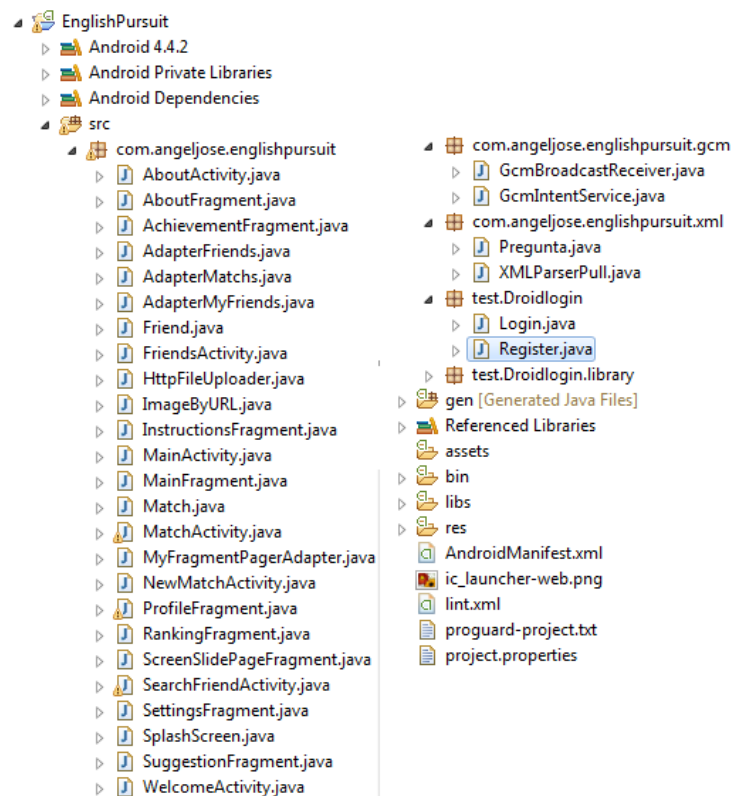


Figura 21: Gráfico de despliegue

En la raíz se encuentra el directorio del proyecto English Pursuit. A continuación y dentro de él están los cinco paquetes en los que se reparte la funcionalidad total de la aplicación:

- paquete *com.angeljose.englishpursuit*:

Contiene las clases que aportan la mayoría de la lógica de la aplicación, los “adapters” y la funcionalidad e interfaz de casi todas las “activities”.

- paquete *com.angeljose.englishpursuit.gcm*:

Contiene las clases encargadas de manejar y establecer las características del servicio de notificaciones de Google Cloud Messaging.

- paquete *com.angeljose.englishpursuit.xml*:

Contiene el parseador del documento xml ubicado en el servidor en el que se encuentran las preguntas de cada partida. Además contiene la clase “Pregunta”, utilizada para crear los objetos pregunta con los atributos recogidos por el parser.

- paquete *test.Droidlogin*:

Contiene las clases “Login” y “Register” encargadas de llevar a cabo el inicio de sesión y el registro respectivamente, además de especificar la interfaz de usuario de dichas “activities”.

- paquete *test.Droidlogin.library*:

Contiene clases utilizadas para el inicio de sesión y el registro. Con ellas se realiza la comunicación con los scripts del servidor (peticiones HTTP y manejo de respuestas) y también el hash MD5 de la contraseña.

5.6 MODELO DE DATOS

Al ser este proyecto software una aplicación multijugador basada en la competitividad entre usuarios, todos los datos se almacenan en el servidor para estar siempre actualizados. Además este almacenamiento externo minimiza el espacio en memoria que ocupa la aplicación (ya que si todas las preguntas se almacenasen de forma interna en la aplicación, se incrementarían muchísimo los requerimientos de espacio local), también posibilita la introducción y modificación de preguntas de forma transparente al usuario. Y otro aspecto muy importante es que evita la posibilidad de trapeo por parte de los usuarios ya que los datos estarán a buen recaudo en un lugar inaccesible para ellos.

5.6.1 MODELO ENTIDAD RELACIÓN

El MER (modelo entidad-relación) es un diagrama que representa las entidades de un sistema de información, así como sus relaciones y atributos propios. De manera que permite obtener un esquema que muestra de un vistazo la información que maneja la aplicación.

La siguiente figura muestra el MER perteneciente a la base de datos descrita anteriormente:

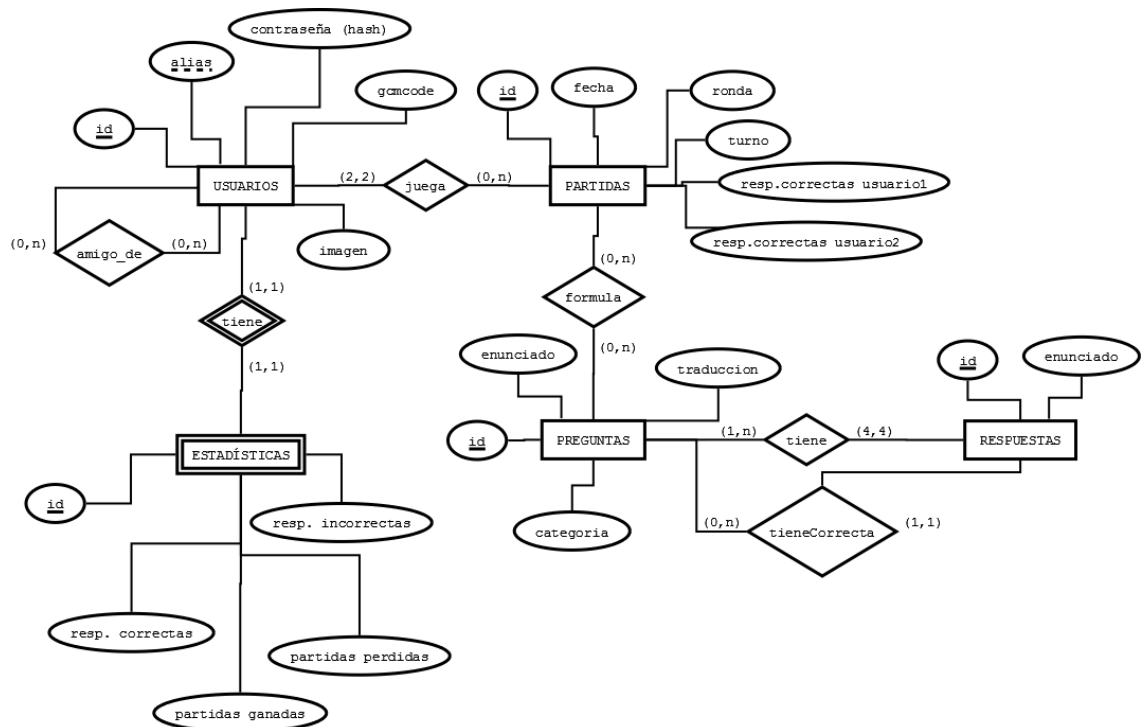


Figura 22: Modelo Entidad-Relación

5.7 BASE DE DATOS

La Base de datos mySQL ubicada en el servidor está compuesta por las siguientes tablas:

Estructura SQL Buscar Generar una consulta Exportar Operaciones											
	Tabla	Acción					Registros	Tipo	Cotejamiento	Tamaño	Residuo a depurar
☐	amigo_de						40	InnoDB	utf8_spanish_ci	32.0 KB	-
☐	estadisticas						28	InnoDB	utf8_spanish_ci	32.0 KB	-
☐	partidas						42	InnoDB	utf8_spanish_ci	64.0 KB	-
☐	preguntas						192	InnoDB	utf8_spanish_ci	160.0 KB	-
☐	respuestas						502	InnoDB	utf8_spanish_ci	16.0 KB	-
☐	usuarios						28	InnoDB	utf8_spanish_ci	16.0 KB	-
6 tabla(s)		Número de filas					832	--	latin1_spanish_ci	320.0 KB	0 Bytes
↶ Marcar todos/as / Desmarcar todos											
Para los elementos que están marcados: ▾											

Figura 23: Tablas BBDD

Cada una de las cuales presenta las siguientes estructuras:

5.7.1 TABLAS

usuarios

La tabla usuarios contiene la información de los usuarios registrados en el sistema, entre ella se encuentra: identificador único, alias (nombre de usuario), contraseña (hash MD5), código GCM (para el envío de notificaciones push al dispositivo) y ruta de la imagen de perfil.

Servidor: db532822026.db.1and1.com ▶ Base de datos: db532822026 ▶ Tabla: usuarios										
Examinar Estructura SQL Buscar Insertar Exportar Operaciones Vaciar Eliminar										
	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción		
☐	ID	int(10)			No		auto_increment			
☐	alias	varchar(10)	latin1_swedish_ci		No					
☐	password	varchar(32)	latin1_swedish_ci		No					
☐	gcmcode	varchar(500)	latin1_swedish_ci		No					
☐	imagen	varchar(120)	utf8_spanish_ci		No	http://comuandroid.com/EnglishPursuit/fotosPerfil/user.png				

Figura 24: Tabla "usuarios"

En la siguiente figura puede verse una parte de la información que contiene la tabla en un determinado momento:

	ID	alias	password	gcmcode	imagen
☐	83	angel3	e7866f0e710fca1584	_ef4pUo2mmwSOKJhKX...	http://comuandroid.com/EnglishPursuit/fotosPerfil/...
☐	85	sr388	ae1d4b431ead52e5e	ADhrFqZric3nQ24JA6...	http://comuandroid.com/EnglishPursuit/fotosPerfil/...
☐	96	i	865c0c0b4ab0e063e	_ef4pUo2mmwSOKJhKX...	http://comuandroid.com/EnglishPursuit/fotosPerfil/...
☐	98	e	e1671797c52e15f76	_ef4pUo2mmwSOKJhKX...	http://comuandroid.com/EnglishPursuit/fotosPerfil/...
☐	99	s	03c7c0ace395d8018	_ef4pUo2mmwSOKJhKX...	http://comuandroid.com/EnglishPursuit/fotosPerfil/...
☐	122	tablet	c4e0ba5422383082c	!AuK_yJwp2qYSW5V...	http://comuandroid.com/EnglishPursuit/fotosPerfil/...
☐	123	pandu	e2c3adf9bafc25757d	3yY2xV7d7DHeeWTbb...	http://comuandroid.com/EnglishPursuit/fotosPerfil/...
☐	124	angelete	b7c6ad0a2f8ab11b0	_ef4pUo2mmwSOKJhKX...	http://comuandroid.com/EnglishPursuit/fotosPerfil/...
☐	125	juan	a94652aa97c7211b0	_ef4pUo2mmwSOKJhKX...	http://comuandroid.com/EnglishPursuit/fotosPerfil/...

Figura 25: Datos de la tabla "usuarios"

amigo_de

La tabla amigo_de establece una relación de amistad entre dos usuarios utilizando sus identificadores.

Servidor: db532822026.db.1and1.com ▶ Base de datos: db532822026 ▶ Tabla: amigo_de

Examinar Estructura SQL Buscar Insertar Exportar Operaciones Vaciar Eliminar

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	ID_USUARIO1	int(10)			No			    
☐	ID_USUARIO2	int(10)			No			    

Figura 26: Tabla "amigo_de"

estadísticas

La tabla estadísticas, como su propio nombre indica, almacena las estadísticas de los usuarios, entre sus atributos se encuentra: identificador de estadística, identificador del usuario al que pertenecen dichas estadísticas, respuestas correctas del jugador, respuestas incorrectas, partidas ganadas totales y partidas perdidas.

Servidor: db532822026.db.1and1.com ▶ Base de datos: db532822026 ▶ Tabla: estadísticas

Examinar Estructura SQL Buscar Insertar Exportar Operaciones Vaciar Eliminar

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	ID	int(10)			No		auto_increment	    
☐	RC	int(10)			No	0		    
☐	RI	int(10)			No	0		    
☐	PG	int(10)			No	0		    
☐	PP	int(10)			No	0		    
☐	ID_USUARIO	int(10)			No			    

Figura 27: Tabla "estadísticas"

partidas

La tabla partidas refleja la información de cada partida creada entre dos jugadores, sus atributos son: identificador único de partida, fecha de creación, estado, identificadores de ambos participantes, sus respuestas correctas, ronda en la que se encuentra la partida y usuario al que pertenece el turno actualmente.

Servidor: db532822026.db.1and1.com ▶ Base de datos: db532822026 ▶ Tabla: partidas

Examinar Estructura SQL Buscar Insertar Exportar Operaciones Vaciar Eliminar

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	ID	int(10)			No		auto_increment	     
☐	FECHA	timestamp			No	CURRENT_TIMESTAMP		     
☐	ESTADO	int(1)			No	1		     
☐	ID_USUARIO1	int(10)			No			     
☐	ID_USUARIO2	int(10)			No			     
☐	RC_USUARIO1	int(2)			No	0		     
☐	RC_USUARIO2	int(2)			No	0		     
☐	RONDA	int(1)			No	1		     
☐	TURN0	int(10)			No			     

Figura 28: Tabla "partidas"

preguntas

La tabla preguntas almacena el banco de preguntas de la aplicación, que será utilizado para generar las cuestiones pertenecientes a una partida. Sus atributos son: identificador de pregunta, enunciado, traducción, categoría a la que pertenece, ids de respuestas posibles e id de respuesta correcta.

Servidor: db532822026.db.1and1.com ▶ Base de datos: db532822026 ▶ Tabla: preguntas

Examinar Estructura SQL Buscar Insertar Exportar Operaciones Vaciar Eliminar

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	id	int(10)			No		auto_increment	     
☐	enunciado	varchar(120)	utf8_spanish_ci		No			     
☐	traduccion	varchar(120)	utf8_spanish_ci		Sí	NULL		     
☐	id_categoria	int(10)			No			     
☐	id_respuesta1	int(10)			No			     
☐	id_respuesta2	int(10)			No			     
☐	id_respuesta3	int(10)			No			     
☐	id_respuesta4	int(10)			No			     
☐	id_respuesta_correcta	int(10)			No			     

Figura 29: Tabla "preguntas"

Puede verse un ejemplo de los datos que almacena esta tabla en un momento determinado en la siguiente figura.

	id	enunciado	traduccion	id_categoria	id_respuesta1	id_respuesta2	id_respuesta3	id_respuesta4	id_respuesta_correcta
☐	1	If you decide to do it, you'll have to ____ the...	Si decides hacerlo tendrás que acatar sus normas.	1	1	2	3	4	4
☐	2	Paul is ____ a fool today.	Paul se está comportando como un tonto hoy.	1	5	6	7	8	5
☐	3	Kevin loves comments ____ by visitors.	A Kevin le encantan los mensajes enviados por los ...	1	9	10	11	12	10
☐	4	The plane ____ at 7 am.	El avión aterrizó a las 7 de la mañana.	1	13	14	15	16	13
☐	5	If you want to join that club, you have to ____	Si quieres unírte al club, tienes que rellenar un ...	1	17	18	19	20	19

Figura 30: Datos tabla "preguntas"

rsepuestas

La tabla respuestas contiene las posibles respuestas a una pregunta formulada en una partida.

Servidor: db532822026.db.1and1.com ▶ Base de datos: db532822026 ▶ Tabla: respuestas

Examinar Estructura SQL Buscar Insertar Exportar Operaciones Vaciar Eliminar

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
☐	id	int(10)			No		auto_increment	☐ ☐ ☐ ☐ ☐ ☐
☐	enunciado	varchar(80)	utf8_spanish_ci		No			☐ ☐ ☐ ☐ ☐ ☐

Figura 31: Tabla "respuestas"

En la siguiente figura, puede verse un ejemplo de las respuestas que pueden almacenarse en esta tabla.

	id	enunciado
☐	1	apply for
☐	2	drop by
☐	3	drop off
☐	4	abide by
☐	5	act like
☐	6	act up
☐	7	be over
☐	8	rub out
☐	9	sent for

Figura 32: Datos tabla "respuestas"

5.8 CONCLUSIONES

A lo largo de este extenso capítulo se han especificado todos los requisitos (funcionales, no funcionales y de información) con los que debe contar y que debe respetar la aplicación, además de las estructuras de datos que ésta implementará para almacenar la información necesaria y lograr un funcionamiento óptimo.

CAPÍTULO 5

De esta manera y a partir de este momento, ya se tiene una base sólida, definida y fija sobre la cual comenzar la implementación de una manera segura, teniendo la certeza de que todo el trabajo que se realice de ahora en adelante no será en vano, puesto que ya se tienen descritos los fundamentos que se requieren para un buen desarrollo y se sabe que éstos no cambiarán.

En ocasiones, la afirmación de que los requisitos serán fijos y no cambiarán puede no ser totalmente cierta ya que a medida que el proyecto se desarrolla, pueden producirse algunos cambios en la especificación, sin embargo, estos cambios serán mucho más leves y generarán menos trabajo y costes que si no se hubiera realizado especificación alguna.

CAPÍTULO 6. HERRAMIENTAS DE TRABAJO

Desde el inicio del documento hasta el capítulo anterior, se han descrito numerosos aspectos acerca de la situación actual del sistema operativo, de los desarrolladores y del mercado, después se ha centrado en aspectos más específicos de este proyecto en particular, como el uso de GCM y la especificación de objetivos y requisitos, por tanto se puede decir que la base para empezar el desarrollo ya está construida.

6.1 HERRAMIENTAS DE DISEÑO

Antes de ver las herramientas utilizadas para el desarrollo e implementación, tanto de la aplicación Android como del servidor, se van a mostrar brevemente las herramientas utilizadas en la fase de diseño.

Cierto es que hay muchas herramientas disponibles gratuitamente en la red que podrían servir para diseñar los diferentes diagramas, como por ejemplo: Axiom, IBM architect, incluso la herramienta básica de diseño Paint o Photoshop (mucho más avanzada). Sin embargo, al haber probado varias alternativas, se ha decidido utilizar el software Dia y Microsoft Visio ya que los diagramas que pueden conseguirse con este software se asemejan más al estilo que se persigue.

6.1.1 DIA



Figura 33: Herramienta Dia

Dia es un software de código abierto que permite la creación de diagramas de distinto índole, entre los cuales destacan: diagramas entidad-relación, diagramas UML, diagramas de flujo, diagramas de redes y diagramas de circuitos eléctricos.

6.1.2 MICROSOFT VISIO

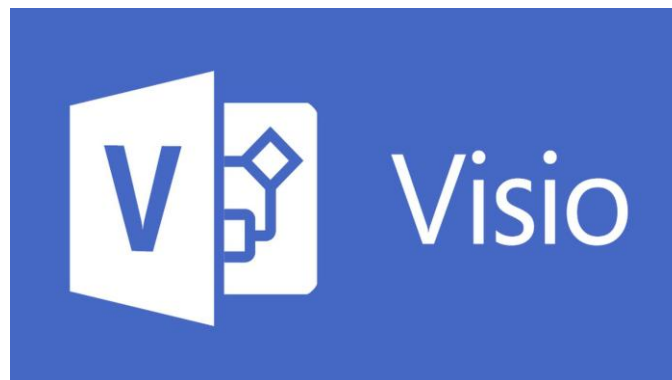


Figura 34: Herramienta Microsoft Visio

Microsoft Visio es un software de dibujo vectorial para sistemas operativos Windows.

Las herramientas que lo componen permiten realizar diagramas de oficinas, diagramas de bases de datos, diagramas de flujo de programas y diagramas UML entre otros.

El resultado de la utilización de Visio es la obtención de diagramas profesionales, claros y representativos.

6.2 HERRAMIENTAS DE DESARROLLO

A partir de este punto se encontrará la información sobre cómo proceder a la implementación de una aplicación Android, cuáles son las herramientas que nos facilitarán la tarea y cómo instalarlas. Además se verá cómo utilizar los recursos que el SDK de Android nos ofrece así como las posibilidades que PHP aporta para implementar las funcionalidades necesarias en un servidor.

Para el desarrollo de nuestra aplicación Android se van a utilizar un conjunto de utilidades basadas en software libre:

- JAVA DEVELOPMENT KIT (JDK)
- ANDROID SDK (SOFTWARE DEVELOPMENT KIT)
- ECLIPSE

6.2.1 JAVA DEVELOPMENT KIT

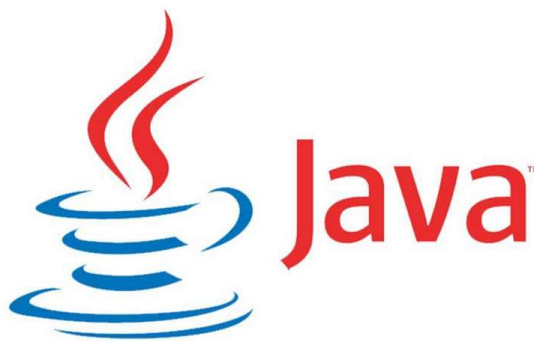


Figura 35: Logo JDK (Java Development Kit)

Se trata de un software que nos provee de herramientas (entre ellas el compilador e intérprete) para desarrollar aplicaciones en Java, el cual, como hemos dicho, es el lenguaje base en el que están escritas las aplicaciones de Android. Por ello es el componente principal que debemos instalar en nuestro entorno de desarrollo.

Para obtenerlo podemos ir a la página oficial de Oracle y descargar su última versión [7].

Su instalación es muy sencilla ya que sólo debemos seguir los pasos del asistente.

6.2.2 SDK



Figura 36: Android SDK

El SDK (Software Development Kit) proporciona un conjunto de herramientas para el desarrollo software de una aplicación Android. Entre ellas se encuentran: un depurador, las bibliotecas necesarias, documentación y ejemplos de código.

Para descargarlo debemos ir a la web de desarrolladores de Android y seleccionar la versión correspondiente a nuestro sistema operativo.

Inicialmente, al descargar el SDK, éste sólo trae las herramientas básicas de desarrollo. Obviamente para desarrollar una aplicación completa y para alguna versión específica que además pueda hacer uso de otras bibliotecas y servicios adicionales, se tendrán que obtener otras herramientas por medio de su gestor de paquetes, el cual podemos ver en la siguiente figura.

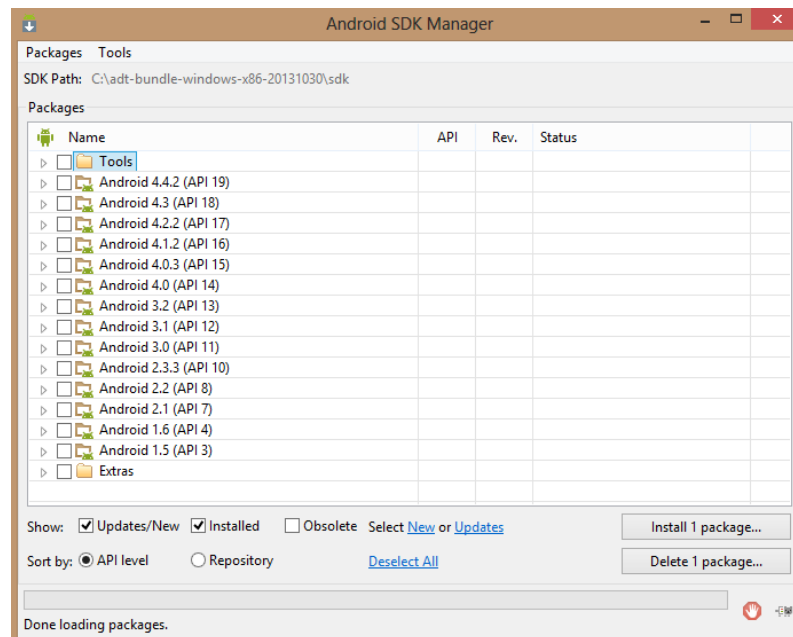


Figura 37: Gestor de paquetes de Android SDK

Entre las herramientas que se pueden incluir en el SDK destacan:

- SDK Tools
- Documentation
- SDK Platform-tools
- SDK Platform
- System Images
- Samples
- Google APIs

SDK Tools

Contiene las herramientas necesarias para depurar y testear una aplicación. Se encuentran en el directorio **<sdk>/tools/**.

Documentation

Aquí se encuentra la documentación de la API de Android de cada versión. Está situada en **<sdk>/docs/**.

SDK Platform

Contiene un SDK disponible para cada versión de Android específica. Se encuentra en el directorio `<sdk>/platforms/<Android-version>/`.

SDK Platform-Tools

Contiene herramientas específicas para el desarrollo y depuración de aplicaciones. Tiene la característica de que las herramientas de una versión posterior son totalmente compatibles con las de las versiones anteriores. Su directorio es `<sdk>/platform-tools/`.

System Images

Las imágenes del sistema son requeridas para la ejecución del emulador (el cual nos permite ejecutar las aplicaciones desarrolladas sin necesidad de disponer de un dispositivo Android físico) de Android en nuestro PC. Se encuentra en `<sdk>/platforms/<Android-version>/`.

Samples

Son ejemplos de aplicaciones ya desarrolladas que constituyen unos excelentes recursos para comprender la estructura de una aplicación y la implementación de los diferentes servicios y APIs mediante el código proporcionado. Están situados en `<sdk>/platforms/<Android-version>/samples/`.

Google APIs

Contiene multitud de APIs de Google que posibilitan la implementación de sus funcionalidades y servicios (por ejemplo Google Maps o GCM) en nuestra aplicación con facilidad. Se encuentra en `<sdk>/add-ons/`.

6.2.3 ECLIPSE



Figura 38: Eclipse

La tercera herramienta que necesitamos para llevar a cabo nuestro desarrollo es el IDE (Integrated Development Environment) o entorno de desarrollo, algo que aunque no es totalmente imprescindible sí que es muy recomendable ya que permite ahorrar una infinidad de tiempo, automatizar tareas, importar librerías, ejecutar los proyectos directamente, evitar errores...

En este caso se ha optado por “Eclipse”, aunque podría haberse optado por cualquier otro como por ejemplo “Netbeans”. Sin embargo, el primero ofrece una serie de ventajas que facilitan mucho la tarea de programación.

Sin duda, una ventaja a mencionar es que dispone de un plugin, llamado ADT (Android Development Tools), que facilita en gran medida las tareas típicas de programación. Para facilitarnos aún más las cosas, Google nos ofrece un paquete que integra Eclipse totalmente configurado con ADT (una iniciativa reciente, ya que hasta hace poco los desarrolladores debían de configurarlo de forma manual) y el SDK mencionado anteriormente.

Otro beneficio que nos aporta la integración de las múltiples herramientas en el entorno es que una vez realizada la implementación, total o parcial, de la aplicación, podemos compilarla y ejecutarla directamente en nuestro sistema operativo mediante un emulador (AVD).

La instalación de eclipse también es muy sencilla y se realiza de la misma forma que cualquier programa habitual.

6.2.4 ANDROID STUDIO

Por último, también se debe mencionar la existencia de otro entorno de desarrollo, el cual fue anunciado en el congreso “Google I/O 2013” por parte de Google y que aunque no será el utilizado para este desarrollo, muy posiblemente será empleado para desarrollos futuros.



Figura 39: Android studio

Su nombre es “Android Studio” y también está disponible para descargar desde [8], la web oficial de desarrolladores de Android.

En principio, este nuevo IDE no aporta muchas novedades significativas frente a Eclipse, aunque sí provee de pequeñas funcionalidades adicionales como: posibilidad de obtener una “preview” de la aplicación en dispositivos con diferente tamaño de pantalla, asistente para crear diseños comunes en Android y herramientas para detectar rendimiento y facilidad de uso. Además, al ser un IDE del cual Google es propietario (al igual que del Sistema Operativo), se asegura estabilidad y continuidad del entorno. Sin embargo todavía está en fase “beta” y tiene varios “bugs” por solucionar, así como otras incompatibilidades. Por ello y por la menor comunidad de soporte que dispone frente a la primera alternativa, en este proyecto se ha optado por el entorno Eclipse.

CAPÍTULO 7. INTERFAZ GRÁFICA DE USUARIO

La interfaz gráfica es la parte más importante de la aplicación a ojos del usuario ya que es lo que éste puede visualizar y manipular para interactuar con la misma. Además es uno de los factores que más pueden influir en el éxito o fracaso de un producto, ya que si la apariencia es fea y aburrida el usuario puede desecharla, sin embargo si tiene una estética clara y agradable puede animar a su uso.

Por ese motivo debemos tener especial cuidado con su diseño e implementación para conseguir una apariencia atractiva, clara y de fácil comprensión, teniendo en cuenta el concepto de captación intuitiva (“affordance”) [9].

Para diseñar una buena interfaz de usuario necesitamos tener en cuenta varios conceptos:

- Usabilidad: debemos implementar una aplicación fácil de usar y aprender.
- Accesibilidad: tenemos que asegurarnos de que las personas sean capaces de utilizar nuestro producto.
- Solidez: los usuarios tienen que poder realizar sus objetivos sin problemas.
- Consistencia: se trata de una de las características más importantes y está ligada íntimamente a la usabilidad.

Consiste en que dentro de la aplicación, los mecanismos deben ser usados de la misma manera proporcionando la misma respuesta ante una misma acción. Ofreciendo una apariencia estética común.

Además también debemos implementar consistencia entre aplicaciones, aprovechando el conocimiento que el usuario tiene en el uso de otras aplicaciones Android.

La manera de llevar estos conceptos teóricos a la práctica es mediante un análisis y estudio previo. Para darle el aspecto final debemos seguir las reglas de diseño, patrones y guías de estilo que Google nos ofrece para Android [10]. Por ello, además de atender a los conceptos descritos anteriormente, también debemos seguir los principios de diseño de Android, también disponibles en la web de desarrolladores [11]:

- Deleitar y sorprender: hay que buscar nuevas formas de enriquecer la experiencia y utilizar mecanismos que contribuyan al bienestar del usuario. Esto se puede conseguir utilizando con acierto las posibilidades que cualquier dispositivo nos ofrece, como por ejemplo, sonido, vibración y animaciones.
- Los objetos reales son más divertidos que los botones y menús: los usuarios agradecen la presencia de objetos que representen objetos de la vida real y muestre claramente su función con sólo mirarlos. De esta manera se reduce el esfuerzo cognitivo para realizar una tarea.
- Cuanto más breve mejor: hay que utilizar frases cortas con palabras sencilla, de forma que los usuarios puedan comprender el texto rápida y fácilmente.
- Imágenes mejor que palabras: “una imagen vale más que mil palabras”.
- Mostrar sólo lo que se necesita cuando se necesita: los usuarios pueden confundirse al ver muchas opciones a la vez. Por ello es mejor dividir las tareas y la información en bloques más pequeños, de forma que lo que no es esencial en ese momento quede oculto
- El usuario debe saber siempre dónde está: al igual que ocurre en la usabilidad web con las “migas de pan” (breadcrumbs), en Android también deben haber relaciones entre las distintas “activities” de una aplicación.
- Nunca perder las preferencias de un usuario: todo lo que un usuario modifique en una aplicación debe quedar guardado en las posteriores ejecuciones.
- Si algo parece que funciona de una manera, debe hacerlo: íntimamente relacionado con consistencia descrita anteriormente.
- Interrumpir sólo si es importante: las notificaciones y alertas deben utilizarse sólo cuando se necesiten.

Habiendo hecho esta breve introducción sobre la interfaz de usuario, en este capítulo se mostrará una primera aproximación del diseño con los prototipos propuestos al inicio. Junto a éstos también se expondrán los diseños definitivos.

7.1 PANTALLA PRINCIPAL

La pantalla principal es la primera vista que el usuario obtendrá nada más abrir la aplicación, por lo que es muy importante cuidar su estética y aportar sólo la información necesaria de una forma organizada y clara.

Puesto que esta aplicación, llamada English Pursuit de ahora en adelante, es un juego de preguntas multijugador por turnos, donde la importancia de su funcionalidad reside en las partidas que los usuarios pueden jugar contra otros jugadores, el contenido más útil y relevante que se puede mostrar en esta primera pantalla es un listado con las partidas ordenadas según el turno de los jugadores. De esta manera los usuarios pueden acceder rápidamente a las partidas en curso.

En la siguiente figura puede apreciarse un diseño primitivo en papel que muestra la idea inicial de esta interfaz, junto con el diseño final de esta pantalla.



Figura 40: Prototipo y diseño final de la IU de la pantalla principal

Los elementos que componen esta pantalla son: un botón para crear nueva partida y tres listas correspondientes al posible estado de una partida (posesión del turno, turno del rival o partida terminada), además para actualizar las listas de partidas se ha incluido la funcionalidad “pull to refresh” que permite realizar esta actualización deslizando la pantalla hacia abajo, al igual que ocurre en aplicaciones tan populares como Twitter y la mayoría de aplicaciones de Google.

7.2 MENÚ LATERAL (NAVIGATION DRAWER)

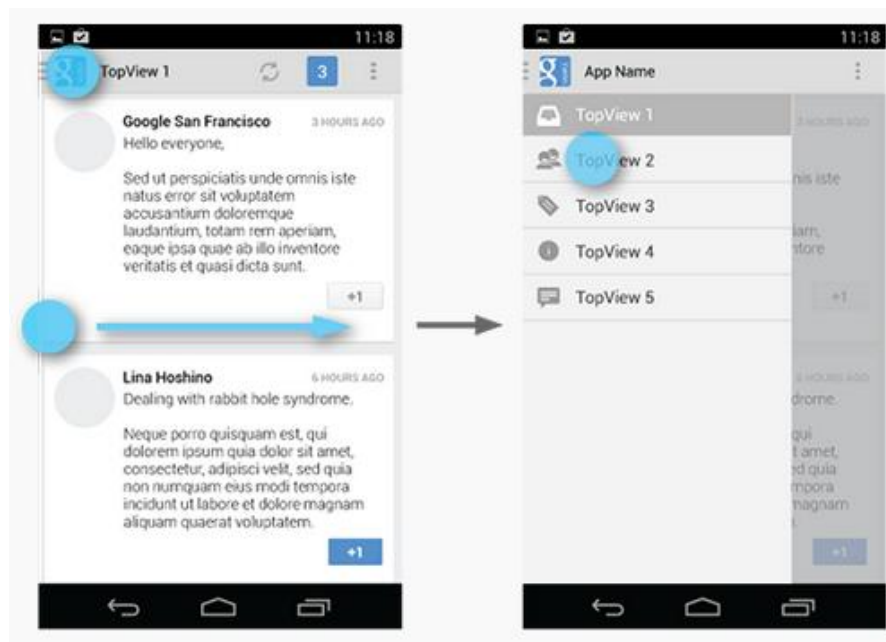


Figura 41: Navigation Drawer

“Navigation Drawer” [12], es el nombre dado a un panel deslizante situado en un lateral de la interfaz, el cual puede extenderse o esconderse con un simple gesto táctil.

Se trata de un estándar que Google ha adoptado en todas sus aplicaciones recientemente para facilitar el uso y la navegación por las distintas opciones. Actualmente, no sólo Google está renovando sus productos con esta nueva interfaz, sino que otros muchos desarrolladores importantes ya han hecho la implementación de sus aplicaciones con esta nueva filosofía gráfica. En la siguiente ilustración tenemos varios ejemplos:

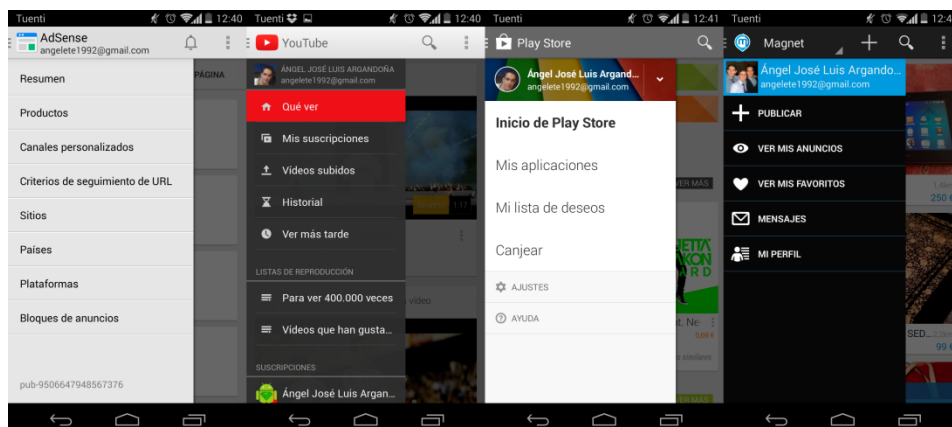


Figura 42: Ejemplos de aplicaciones con Navigation Drawer

La función de este panel es actuar como un contenedor de opciones para navegar por las distintas ventanas (activities o fragments) de la aplicación.

Para mejorar la accesibilidad a este panel, además de poder abrirse utilizando un gesto deslizante, también se puede acceder al mismo tocando el icono de la aplicación.

Al comprobar que este menú encaja perfectamente en la interfaz de English Pursuit se ha decidido implementarlo y recoger en él las secciones más importantes de la aplicación.

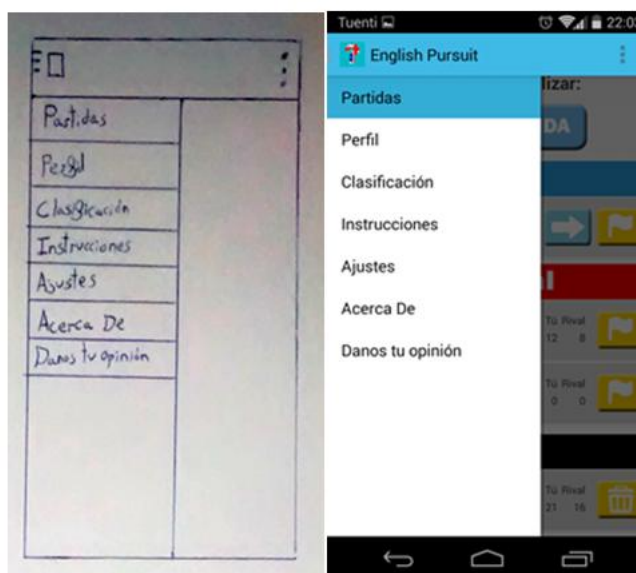


Figura 43: Prototipo y diseño final de la IU del menú lateral

7.3 PANTALLA DE CLASIFICACIÓN

En la pantalla de clasificación aparece una lista con los diez mejores jugadores de English Pursuit, es decir, los que más partidas ganadas tienen. Cada elemento de la lista muestra la imagen de perfil del usuario, su nombre y unos botones que permiten agregarlo como amigo e incluso crear una partida contra ellos.



Figura 44: Prototipo y diseño final de la IU de la pantalla de clasificación

7.4 PANTALLA DE INSTRUCCIONES

Las instrucciones de English Pursuit están escritas en un archivo .html almacenado en el servidor, por lo que la visualización del mismo se realizará mediante el navegador predeterminado del dispositivo Android.

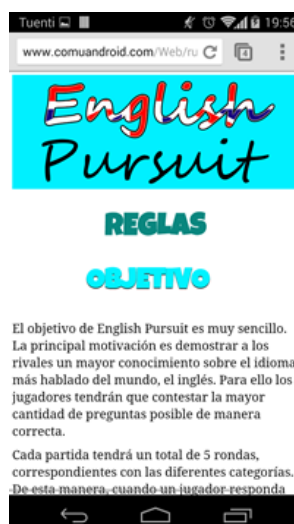


Figura 45: Diseño final de la IU de la pantalla de instrucciones

7.5 PANTALLA DE PERFIL DE USUARIO

En la pantalla de perfil aparece el nombre del usuario identificado y sus estadísticas (tanto del total de respuestas contestadas como de todas las partidas jugadas). Además el usuario tiene la posibilidad de cerrar sesión, para poder identificarse con las credenciales de otro jugador.



Figura 46: Prototipo y diseño final de la IU de la pantalla de perfil

7.6 PANTALLA DE PREGUNTAS

Cuando el usuario decida jugar una partida (cuyo turno le corresponda) y pulse el botón correspondiente, accederá a la pantalla de preguntas, en donde aparecerá la categoría de las mismas, el enunciado, la traducción y las cuatro posibles respuestas, además de un contador de tiempo que nos obligará a terminar la ronda antes de que llegue a cero. En la parte baja también se especifica el número de pregunta actual y el número de aciertos o fallos acumulados hasta el momento.

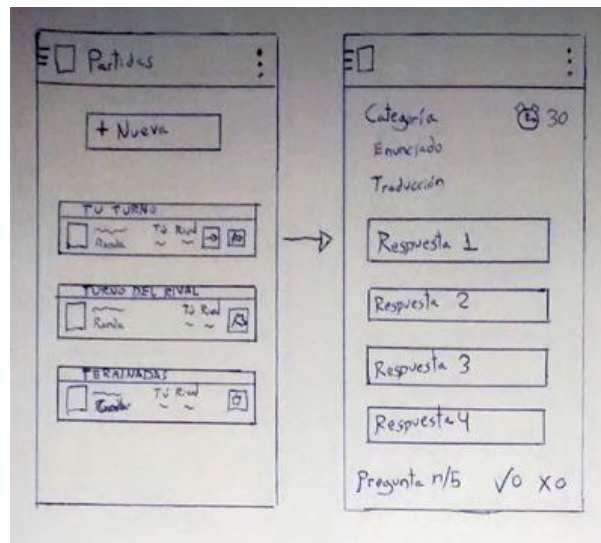


Figura 47: Prototipo de la IU de la pantalla de preguntas

En la siguiente figura se muestra el diseño final de la pantalla de preguntas y del aspecto visual en caso de acierto o fallo.

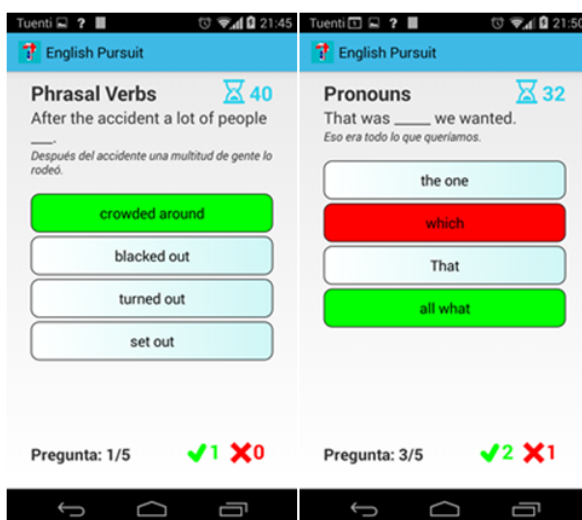


Figura 48: Diseño de la IU de la pantalla de preguntas

7.7 PANTALLA DE BUSCAR AMIGOS

En la pantalla de “buscar amigos” un usuario puede introducir el Nick de un amigo para buscarlo y agregarlo como tal o crear una partida contra él directamente. Está compuesta por un campo de texto y un botón que lanza la acción de buscar el nombre de usuario en la base de datos para devolver las posibles coincidencias.



Figura 49: Prototipo y diseño final de la IU de la pantalla de buscar amigos

7.8 PANTALLA DE AJUSTES

En la pantalla de ajustes se muestran las preferencias relativas a las notificaciones que el dispositivo puede recibir frente a varios eventos (como la invitación a una partida o la finalización del turno del rival).

Está compuesta por un interruptor que habilita o deshabilita todas las funciones referentes a las notificaciones y de una serie de “checkbox” que activarán o desactivarán el sonido, la vibración y la iluminación del led del dispositivo.

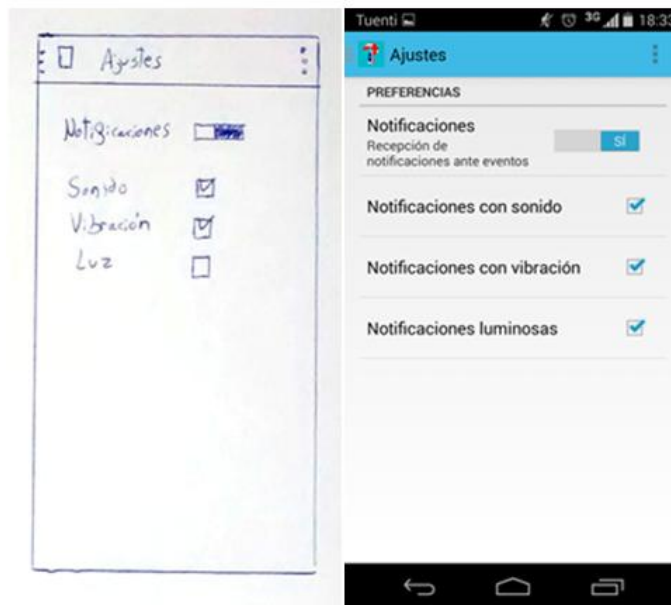


Figura 50: Prototipo y diseño final de la IU de la pantalla de ajustes

7.9 RECURSOS MULTIMEDIA

English Pursuit utiliza una serie de recursos multimedia con el objetivo de mejorar la interfaz y la interacción con el usuario. Los tipos de recursos utilizados son gráficos y audio.

SoundPool utiliza el servicio de la clase MediaPlayer para decodificar el audio en el formato PCM de 16 bits, lo que permite reproducirlo rápidamente por el hardware sin tener que decodificarlas cada vez que se utilice.

El constructor es el siguiente:

```
SoundPool(int maxStreams, int streamType, int srcQuality)
```

-El primer parámetro, maxStreams, especifica cuantos efectos de sonido pueden reproducirse simultáneamente.

-El segundo, streamType, especifica a qué stream de audio se va a enviar el sonido, AudioManager.STREAM_MUSIC es el que Android utiliza por defecto para reproducir música.

-El tercero, srcQuality, ajusta la calidad, debemos dejarlo a 0 para obtener la calidad original.

El método load se utiliza para cargar un recurso de sonido:

```
public int load (Context context, int resId, int priority)
```

-El primer parámetro es el contexto de la aplicación.

-El segundo contiene el ID del recurso de sonido almacenado en el directorio /raw.

-El tercero especifica la prioridad y aunque todavía no tiene efecto, debe dejarse a 1 para compatibilidad futura.

Una vez que tenemos el SoundPool y el flujo de sonido definidos, sólo queda reproducirlo con el método play, en el cual se especifica dicho flujo de sonido y otros parámetros:

```
public final int play (int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)
```

En este caso, el significado de los parámetros no va a explicarse debido a su obviedad.

CAPÍTULO 8. SERVIDOR PHP



Puesto que la idea de esta aplicación es que brinde una experiencia multijugador, es necesario disponer de un servidor que permita almacenar datos referentes a todos los aspectos del juego, como jugadores, partidas, preguntas, etc.

Para abordar esta tarea nos hemos decantado por utilizar un servidor propio en un pc normal de escritorio. Hay muchos programas para Windows (sistema operativo donde se montará el servidor) que nos permiten llevar instalar y poner en funcionamiento el servidor. Sin embargo hemos decidido utilizar “WAMP”, una plataforma para Windows que contiene herramientas imprescindibles para nuestro desarrollo, entre ellas PHP y MySQL que nos ayudarán a montar nuestra aplicación web y comunicarnos con nuestros clientes Android.

Linux y Mac también disponen de programas que ofrecen estas mismas funcionalidades, éstos son LAMP y MAMP respectivamente.

Si queremos que nuestro propio servidor atienda a las peticiones externas deberíamos conectarnos a su ip pública (para pruebas he utilizado la IP privada con el dispositivo conectado a la misma red que el servidor), la cual debería ser fija ya que debe quedar especificada en el código de la aplicación cliente. De este modo necesitamos un mecanismo para poder hacer referencia a nuestro servidor siempre con el mismo nombre. Esta tarea puede realizarse asociando un nombre de dominio y que éste se apunte a nuestra

IP dinámica. Un servicio que puede ayudarnos en este cometido es un DNS dinámico como “No-IP” [14].

Sin embargo, lo más adecuado es contar con un servidor dedicado de alguna empresa de hosting que nos ofrezca un buen servicio. Por eso, finalmente se ha decidido contratar un paquete básico en 1&1 [15] y alojar ahí tanto los archivos y scripts del servidor como la base de datos.

8.1 Base de datos (MySQL)

Como ya se dijo (véase la sección 5.7), la base de datos será mySQL y estará ubicada en el servidor con el objetivo de mantener los datos centralizados en un único lugar.

8.2 PHP como interfaz de comunicación entre BBDD del servidor y cliente (Android)



Figura 52: Logo PHP

Para implementar la interacción entre el cliente y el servidor y permitir la modificación de la Base de datos desde dispositivos Android se han implementado varios scripts que realizan funciones específicas con distintas consultas mysql.

El funcionamiento general de esta comunicación es el siguiente:

1. La aplicación Android envía una petición POST con unos parámetros específicos a un script en php alojado en el servidor.
2. El script recoge los parámetros y realiza las consultas necesarias en la BBDD (inserción, actualización, borrado o consulta).

3. Una vez finalizadas todas las consultas, el script devuelve los datos necesarios al cliente Android mediante el protocolo JSON, el cual es un formato ligero para intercambiar datos.
4. Finalmente la aplicación Android recibe el objeto JSON y obtiene los datos enviados por el servidor, pudiendo mostrarlos o utilizarlos para otros fines.

Originalmente se comenzó a trabajar con la versión 5.4 de PHP en el servidor, sin embargo, ésta no era compatible con la función “array_column” que simplifica la extracción de valores pertenecientes a una clave y que se decidió utilizar en el script “createMatchWithRandom”.

PHP 5.5	PHP 5.4
Versión utilizada	Versión PHP recomendada
Versión estable actual - No es compatible todavía con todas las aplicaciones.	Versión estándar actual - ofrece prestaciones superiores con una carga de memoria menor.
Documentación PHP55	Documentación PHP54

Figura 53: Versiones PHP

Todos los scripts PHP están ubicados en la ruta absoluta “C:\wamp\www\EnglishPursuit”. Entre ellos se encuentran:

- config.php

Contiene las variables de configuración de la base de datos: nombre del servidor, usuario y contraseña con privilegios y nombre de la base de datos donde se alojan las tablas.

- connectbd.php

Realiza la conexión a la base da datos utilizando los parámetros especificados en “config.php”.

- funciones_bd.php

Contiene todas las funciones encargadas de manipular la base de datos.

- Numerosos scripts encargados de realizar tareas específicas llamando a las funciones de “funciones_bd.php”.

CAPÍTULO 8

En vez de exponer aquí cada uno de los scripts, a fin de mejorar la comprensión será en el siguiente capítulo donde se vea el código fuente y la funcionalidad de los mismos, junto a la parte de cliente encargada de realizar las peticiones. Así se podrá relacionar cada función y podrá verse el comportamiento complementario de cliente y servidor.

CAPÍTULO 9. IMPLEMENTACIÓN DE FUNCIONALIDADES EN EL CLIENTE (ANDROID) Y SERVIDOR (PHP)

En capítulos anteriores se ha realizado una introducción y una descripción de las funcionalidades que nuestra aplicación va a llevar a cabo. Pero no es hasta este capítulo donde se va a profundizar en ver el funcionamiento interno y el código programado para satisfacerlo.

9.1 NAVIGATION DRAWER (MENÚ DESLIZANTE LATERAL)

Como se dijo en la sección anterior, la interfaz gráfica de la pantalla principal de nuestra aplicación dispondrá del menú deslizante lateral denominado “Navigation Drawer”.

Para llevar a cabo su implementación se ha consultado la documentación pertinente en la página oficial de desarrolladores Android [16].

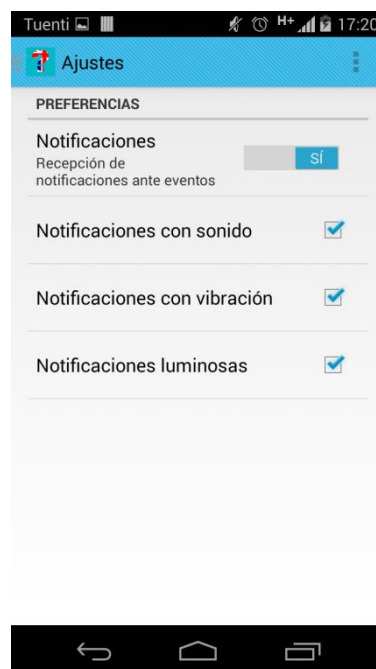
Debido a la implementación de este menú y según las recomendaciones de Google, además de representar las pantallas de la aplicación con “activities”, también se utilizan “fragments”. Los fragments constituyen una porción de la interfaz y la muestran como si de una “activity” se tratara. La única diferencia es que un “fragment” está asociado a una “activity” (y una “activity” puede contener varios “fragments”) y también a su ciclo de vida. El beneficio que aporta esta técnica es que al cargarse la aplicación se cargan todos los “fragment” y al seleccionar un elemento del menú, la aplicación no tiene que realizar el esfuerzo de memoria que llevaría a cabo al cambiar de “activity”.

9.2 PREFERENCIAS DE LA APLICACIÓN

Las preferencias de una aplicación son una parte fundamental de la misma ya que permite que el usuario pueda ejercer un control sobre las distintas funcionalidades que presenta.

Se ha procedido a su implementación utilizando los recursos que Google nos ofrece en su documentación [17].

En este caso debido a que se utiliza el ya mencionado peculiar “Navigation Drawer” y en algunas ocasiones sustituimos las “activities” por “fragments”, las preferencias también van a ser mostradas por un “fragment” que extenderá de “`PreferenceFragment`” para ofrecernos las funcionalidades necesarias. Para acceder a estas preferencias desde la aplicación, el usuario podrá seleccionar la opción “Ajustes” en el menú lateral o en el menú general de la aplicación. Desde esta ventana se podrán controlar los ajustes referentes a las notificaciones que el dispositivo recibe ante varios eventos.



El código de la clase (**`SettingsFrament.java`**) encargada de mostrar las preferencias es muy sencillo ya que lo único que realiza es mostrar la interfaz definida en el layout xml correspondiente (`res/xml/preferences.xml`).

```

package com.angeljose.englishpursuit;

import Android.os.Bundle;
import Android.preference.PreferenceFragment;

public class SettingsFragment extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Load the preferences from an XML resource
        addPreferencesFromResource(R.xml.preferences);
    }
}

```

A su vez, el archivo **preferences.xml**, donde se define el tipo de preferencias y su interfaz, contiene el siguiente código:

```

<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:Android="http://schemas.Android.com/apk/res/Android">

    <PreferenceCategory
        Android:title="@string/pref_title"
        Android:key="pref_settings">

        <SwitchPreference
            Android:key="pref_notif"
            Android:title="Notificaciones"
            Android:summary="Recepción de notificaciones ante eventos"
            Android:defaultValue="true" />

        <CheckBoxPreference
            Android:dependency="pref_notif"
            Android:key="pref_notif_sound"
            Android:title="Notificaciones con sonido"
            Android:defaultValue="true" />

        <CheckBoxPreference
            Android:dependency="pref_notif"
            Android:key="pref_notif_vibrate"
            Android:title="Notificaciones con vibración"
            Android:defaultValue="true" />

        <CheckBoxPreference
            Android:dependency="pref_notif"
            Android:key="pref_notif_light"
            Android:title="Notificaciones luminosas"
            Android:defaultValue="true" />

    </PreferenceCategory>
</PreferenceScreen>

```

De esta manera tan sencilla podemos construir los elementos de preferencias más habituales como pueden ser: checkbox, listas o cajas de texto.

El funcionamiento interno de las preferencias es muy sencillo. Cada una de ellas se registra en las preferencias compartidas (SharedPreferences) identificándose con una clave (key) y almacenando como valor su estado actual. De este modo podemos

consultar el estado de las mismas desde cualquier lugar de nuestra aplicación para ejecutar las acciones pertinentes.

En este caso vamos a utilizar el valor de estas preferencias en la clase (**GcmIntentService.java**), encargada de manejar las notificaciones push de nuestro dispositivo.

```
private void sendNotification(String msg) {
    mNotificationManager = (NotificationManager)
        this.getSystemService(Context.NOTIFICATION_SERVICE);

    PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
        new Intent(this, MainActivity.class), 0);

    SharedPreferences sharedPref =
    PreferenceManager.getDefaultSharedPreferences(this);
    Boolean receiveNotifications = sharedPref.getBoolean("pref_notif", true);
    Boolean soundNotifications = sharedPref.getBoolean("pref_notif_sound",
true);
    Boolean vibrateNotifications =
sharedPref.getBoolean("pref_notif_vibrate", true);
    Boolean lightNotifications = sharedPref.getBoolean("pref_notif_light",
true);

    //Creamos la notificación asignándole los atributos que deseemos (título,
mensaje, sonido, habilitar LED...)

    if(receiveNotifications){
        int sound = 0;
        long vibrate = 0;
        int light = 0;

        if(soundNotifications) sound = RingtoneManager.TYPE_NOTIFICATION;
        if(vibrateNotifications) vibrate = 10;
        if(lightNotifications) light = NOTIFICATION_ID;

        long vibration[] = { vibrate, vibrate, vibrate, vibrate, vibrate };

        NotificationCompat.Builder mBuilder =
            new NotificationCompat.Builder(this)
                .setContentTitle("EnglishPursuit")
                .setSmallIcon(R.drawable.ic_launcher)
                .setSound(RingtoneManager.getDefaultUri(sound))
                .setVibrate(vibration)
                .setLights(light, light, light)
                .setAutoCancel(true)
                .setStyle(new NotificationCompat.BigTextStyle()
                    .bigText(msg))
                .setContentText(msg);

        mBuilder.setContentIntent(contentIntent);
        mNotificationManager.notify(NOTIFICATION_ID, mBuilder.build());
    }
}
```

Como puede apreciarse, en el método “sendNotification” se leen los valores de las preferencias para, dependiendo de los mismos, cambiar los valores de unas variables que se utilizarán para construir las notificaciones.

9.3 Registro

Lo primero que debe hacer un usuario, antes de poder utilizar la aplicación es registrarse en la misma eligiendo un nombre de usuario, contraseña e imagen de perfil.

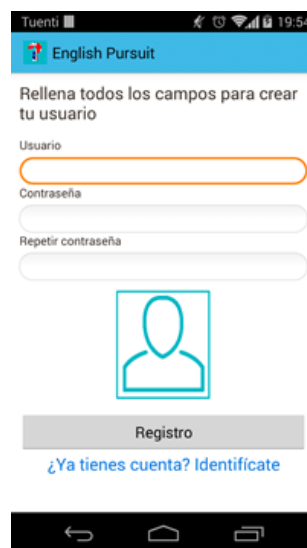


Figura 54: Diseño de la actividad "Register"

Una vez terminado el registro se insertan las tuplas necesarias en la base de datos.

Como ya se ha comentado en el capítulo anterior, la interacción con la base de datos se realiza siempre de la misma manera. Llamando a un script php del servidor y enviando los parámetros necesarios mediante petición post. Esto debe realizarse en una tarea asíncrona ya que de otra manera si la ejecutásemos en el mismo proceso que la interfaz de usuario ésta se bloquearía y daría la sensación de que la aplicación se hubiera parado (de hecho desde Android 3.0 la implementación de cualquier conexión a internet debe realizarse en un proceso diferente).

Sin más, los fragmentos de código más relevantes encargados de realizar la tarea de registro desde el cliente (Android) son los siguientes:

Lo primero de todo es definir e inicializar los atributos necesarios:

```

@Override
public void onCreate(Bundle savedInstanceState) {

    rutaFotoPerfil = Environment.getExternalStorageDirectory() + "/imagen"+" .jpg";

    super.onCreate(savedInstanceState);
    setContentView(R.layout.register);
    post=new HttpPostaux();

    user= (EditText) findViewById(R.id.edusuario);
    pass= (EditText) findViewById(R.id.edpassword);
    pass2= (EditText) findViewById(R.id.edpassword2);
    bregister= (Button) findViewById(R.id.BRegister);
    bfotoPerfil= (ImageButton) findViewById(R.id.fotoPerfilButton);
    login = (TextView) findViewById(R.id.Link_to_login);
}

```

Figura 55: Atributos Clase "Register"

Después se define el listener del botón de registro encargado el cuál ejecutará una serie de comprobaciones para ver que los datos son correctos: ningún campo obligatorio vacío, contraseña de verificación equivalente, nombre de usuario no existente en la base de datos...

```

bregister.setOnClickListener(new View.OnClickListener(){

    public void onClick(View view){

        //Extraemos datos de los EditText
        String usuario=user.getText().toString();
        String passw=pass.getText().toString();
        String passw2=pass2.getText().toString();

        //verificamos si están en blanco
        if( checklogindata( usuario , passw, passw2)==true){

            if(usuario.length()<=12){

                //si no hay ningún campo de entrada en blanco y el nombre de usuario tiene menos de 12 caracteres
                //pasamos a realizar la segunda comprobación, que las dos passwords coincidan

                if(passEquals(passw, passw2)==true){
                    //lanzamos la tarea asíncrona que realiza el registro
                    new asynclogin().execute(usuario,passw);
                }
            }
            else{
                //si las contraseñas no coinciden se muestra un aviso al usuario
                Vibrator vibrator1 =(Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
                vibrator1.vibrate(200);
                Toast toast = Toast.makeText(getApplicationContext(),"Error:Las contraseñas no coinciden", Toast.LENGTH_SHORT);
                toast.show();
            }
        }
    }
}

```

Figura 56: Listener botón registro

Si todo es correcto se procede al envío de datos al servidor, no sin antes encriptar la contraseña realizando el hash MD5 para no guardarla en texto plano en la base de datos. El algoritmo encargado de realizar este hash se encuentra en la clase **MD5encrypter** del paquete **test.Droidlogin.library**:

```

public class MD5encrypter {

    public static String md5(String clear) throws Exception {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] b = md.digest(clear.getBytes());
        int size = b.length;
        StringBuffer h = new StringBuffer(size);
        //algoritmo y arreglo md5
        for (int i = 0; i < size; i++) {
            int u = b[i] & 255;
            if (u < 16) {
                h.append("0" + Integer.toHexString(u));
            }
            else {
                h.append(Integer.toHexString(u));
            }
        }
        //clave encriptada
        return h.toString();
    }
}

```

Figura 57: Clase “MD5encrypter”

Finalmente la petición post al script del servidor y el envío de datos se realiza en una tarea asíncrona:

```

public boolean registerStatus(String username ,String password) {
    uploadFile(rutaFotoPerfil);
    /*Creamos un ArrayList del tipo nombre valor para agregar los datos
    y enviarlos mediante POST a nuestro sistema para relizar el registro*/
    ArrayList<NameValuePair> postparameters2send= new ArrayList<NameValuePair>();

    postparameters2send.add(new BasicNameValuePair("usuario",username));
    postparameters2send.add(new BasicNameValuePair("password",password));
    postparameters2send.add(new BasicNameValuePair("gcmcode",regid));
    //si el usuario ha elegido una imagen, enviamos su nombre para almacenarla en la bbdd, sino no lo enviamos
    if(nombreFotoPerfil!=null) postparameters2send.add(new BasicNameValuePair("imagen",rutaFotoPerfil));

    //realizamos una peticion y como respuesta obtenes un array JSON
    JSONArray jdata=post.getServerdata(postparameters2send, URL_connect);

    //si lo que obtuvimos no es null
    if (jdata!=null && jdata.length() > 0){

        JSONObject json_data; //creamos un objeto JSON
        try {
            json_data = jdata.getJSONObject(0); //leemos el primer segmento en nuestro caso el unico
            regstatus=json_data.getInt("regstatus");//accedemos al valor
            Log.e("loginstatus","regstatus= "+regstatus);//muestro por log que obtuvimos
        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        /*validamos el valor obtenido (si se obtiene un 1 es que el registro ha ido bien,
        por el contrario, si se obtiene cualquier otro valor significa que algo ha fallado)*/

        if(regstatus==1)return true;
        else return false;
    }
}

```

Figura 58: Implementación función "registerStatus"

Como puede apreciarse, los parámetros se envían con un array clave-valor, para así, en los scripts recuperar cada valor mediante su clave.

Por su parte, el servidor recogerá los datos con el script “adduser” y subirá la imagen de perfil con el script “upload”:

```

<?php
$usuario = $_POST['usuario'];
$password = $_POST['password'];
$gcmcode = $_POST['gcmcode'];
/*$imagen = "http://192.168.1.15/EnglishPursuit/fotosPerfil/".$_POST['imagen'];*/

//basename devuelve el último componente de nombre de una ruta, es decir, el nombre del archivo
$imagen = "http://comuandroid.com/EnglishPursuit/fotosPerfil/".$_POST['imagen'];

require_once 'funciones_bd.php';
$db = new funciones_bd();

/**Según lo que ocurra, devolvemos un json con el resultado codificado de la siguiente manera:
0 = El usuario que se quiere registrar YA existe en la BBDD
1= Registro correcto
2= Error
*/
//si el usuario no sube ninguna imagen al servidor se utiliza la imagen por defecto
if(!isset($_POST['imagen'])) $imagen="http://comuandroid.com/EnglishPursuit/fotosPerfil/user.png";

if($db->isuserexist($usuario)){
    $resultado[]=array("estatus"=>"0");
    //El usuario YA existe en la BBDD
}else{
    if($db->adduser($usuario,$password, $gcmcode, $imagen))
    {
        //El usuario fue agregado a la Base de Datos correctamente
        $resultado[]=array("estatus"=>"1");
    }else{
        //cualquier otro error
        $resultado[]=array("estatus"=>"2");
    }
}

/**
 * agregar nuevo usuario
 */
public function adduser($username, $password, $gcmcode, $imagen) {

    $result = mysql_query("INSERT INTO usuarios(alias,password, gcmcode, imagen) VALUES('$username', '$password', '$gcmcode', '$imagen')");
    $last_id = mysql_insert_id();
    //creo las estadísticas asociadas al nuevo usuario
    $crearEstadisticas = mysql_query("INSERT INTO estadisticas(ID_USUARIO) VALUES('$last_id')");

    // check for successful store

    if ($result && $crearEstadisticas) {

        return true;

    } else {

        return false;

    }

}

```

Figura 59: Script "adduser"

```

<?php
$target_path = basename( $_FILES['uploadedfile']['name']);
//obtenemos la extensión del archivo para conocer el formato de imagen
$ext = substr(strtolower(strrchr($_FILES['uploadedfile']['name'], ".")),1);
// Creamos la imagen de un tipo u otro dependiendo de la extensión
switch($ext)
{
    case 'jpg':
        $image = imagecreatefromjpeg($_FILES['uploadedfile']['tmp_name']);
        break;
    case 'png':
        $image = imagecreatefrompng($_FILES['uploadedfile']['tmp_name']);
        break;
    case 'gif':
        $image = imagecreatefromgif($_FILES['uploadedfile']['tmp_name']);
        break;
    default:
        exit('Unsupported type: '.$_FILES['uploadedfile']['type']);
}
//Redimensionamos la imagen para que ocupe mucho menos espacio en el servidor,
//ya que de todas maneras la aplicación android mostrará una imagen de reducidas dimensiones.
// Target dimensions
$max_width = 100;
$max_height = 100;
// Get current dimensions
$old_width = imagesx($image);
$old_height = imagesy($image);
$new_width = 100;
$new_height = 100;
// Create new empty image
$new = imagecreatetruecolor($new_width, $new_height);
// Resize old image into new
imagecopyresampled($new, $image,
    0, 0, 0, 0,
    $new_width, $new_height, $old_width, $old_height);
ob_start();
imagejpeg($new, "fotosPerfil/".$target_path, 70);
$data = ob_get_clean();

```

Figura 60: Script upload

Además, los datos del usuario se almacenarán en los “SharedPreferences” de la aplicación, para poder consultarlos cuando sea necesario de forma transparente.

```

//Si el registro es correcto almacenamos el mismo en las preferencias
SharedPreferences prefs =
    getSharedPreferences("Preferencias",Context.MODE_PRIVATE);

SharedPreferences.Editor editor = prefs.edit();
editor.putString("user", user);
editor.putString("password", pass);
editor.putString("gcmCode", regid);
editor.commit();
Intent i=new Intent(Register.this, MainActivity.class);
i.putExtra("user",user);
startActivity(i);
finish();

```

Figura 61: Almacenar datos de usuario en "SharedPreferences"

Y eso es todo, una vez realizado todo el proceso, los datos del usuario estarán en la BBDD y éste quedará registrado en la aplicación.

9.4 Login

La clase que maneja la identificación de un usuario es muy parecida a la anterior (registro). Aunque más sencilla ya que no realiza ninguna inserción en la base de datos y sólo comprueba dos campos: el nombre de usuario y la contraseña.

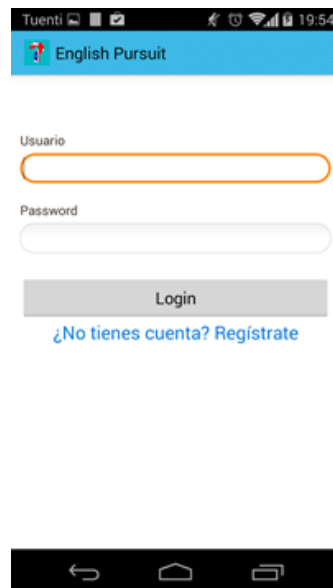


Figura 62: Diseño activity "Login"

Con motivo de no alargar la explicación y no repetir conceptos ya aclarados, de ahora en adelante no se va a entrar en el detalle del código de las tareas asíncronas que realizan las peticiones a los scripts del servidor, ya que todas se comportan de la misma manera. De modo que sólo se nombrará la tarea y se especificarán los parámetros a enviar.

Como ocurría en el registro, en vez de enviar la contraseña escrita por el usuario se obtendrá su hash ya que es lo que se comprará con el campo de la base de datos. Además de la contraseña, se enviará el nombre de usuario y el código GCM (cada dispositivo tendrá un código de Google Cloud Messaging asociado y si nos identificamos como otro usuario, debemos actualizar el campo de la BBDD con dicho código para que las notificaciones enviadas a ese usuario nos lleguen a el dispositivo que ha realizado el inicio de sesión).

```

/*Valida el estado del logueo solamente necesita como parametros el usuario y passw*/
public boolean loginStatus(String username ,String password ) {
    int logstatus=-1;

    /*Creamos un ArrayList del tipo nombre valor para agregar los datos recibidos por los parametros anteriores
    * y enviarlo mediante POST a nuestro sistema para realizar la validacion*/
    ArrayList<NameValuePair> postparameters2send= new ArrayList<NameValuePair>();

    postparameters2send.add(new BasicNameValuePair("usuario",username));
    postparameters2send.add(new BasicNameValuePair("password",(password)));
    postparameters2send.add(new BasicNameValuePair("gcmcode",regid));

    //realizamos una peticion y como respuesta obtenes un array JSON
    JSONArray jdata=post.getServerdata(postparameters2send, URL_connect);

    //validamos el valor obtenido
    if (logstatus==0){// [{"logstatus":"0"}]
        Log.e("loginStatus ", "invalido");
        return false;
    }
    else{// [{"logstatus":"1"}]
        Log.e("loginStatus ", "valido");

        return true;
    }
}

```

Figura 63: Implementación función "loginStatus"

Esta función recoge la respuesta emitida por el servidor, la cual está codificada en un json e indica si el login ha sido correcto (logStatus==1) o no (logStatus==0). Dependiendo de ello, accederá a la pantalla de partidas del usuario o notificará al usuario con un error.

Por su parte, el código del script del servidor es el siguiente:

```

<?php
/*LOGIN*/

$usuario = $_POST['usuario'];
$password = $_POST['password'];
$gcmcode = $_POST['gcmcode'];

require_once 'funciones_bd.php';
$db = new funciones_bd();

if($db->login($usuario,$password)){
    $resultado[]=array("logstatus"=>"0");
}else{
    $db->updateGcmcode($usuario, $gcmcode);
    $resultado[]=array("logstatus"=>"1");
}

echo json_encode($resultado);

```

```

public function login($usuario,$password){

$result=mysql_query("SELECT COUNT(*) FROM usuarios WHERE alias='$usuario' AND password='$password' ");
$count = mysql_fetch_row($result);

/*como el usuario debe ser unico cuenta el numero de ocurrencias con esos datos y si es mayor que cero, entonces existe*/

    if ($count[0]==0){
        return true;
    }else{
        return false;
    }
}

```

Figura 64: Script "access"

9.5 Buscar usuario



Figura 65: Diseño activity "SearchFriendActivity"

Las pantallas que muestran listas de elementos recogidos del servidor: “SearchFriendActivity”, “FriendsActivity”, “RankingFragment” y “MainFragment”. Tienen en común la utilización de lo que se llaman “ListView personalizadas”, las cuales hacen uso de “Adapters” para mostrar la información.

Estos “Adapters” se rellenan con “ArrayList” de objetos y a su vez, estos “ArrayList” se alimentan de los datos recogidos por el servidor.

Además como queremos que el adapter contenga una serie de elementos a nuestra conveniencia, debemos crear un “Adapter personalizado” y también un archivo “Layout” xml donde se defina la apariencia del mismo. Esto puede parecer complejo, así que, vayamos por partes:

1. Teniendo en cuenta que en los adaptadores se persigue el siguiente diseño:

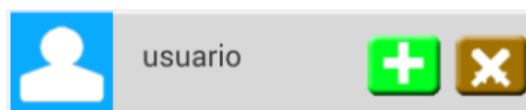


Figura 66: Diseño adapter personalizado "Amigo"

Necesitamos crear un layout xml con: un “ImageView” para la imagen de perfil, un “TextView” para el nombre de usuario y dos “Button”, los cuales permitan agregar al usuario como amigo o jugar contra él respectivamente.

Para satisfacer estas necesidades podemos utilizar el siguiente xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/border1"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/imageUser"
        android:layout_width="70dp"
        android:layout_height="70dp" />
    <TextView
        android:id="@+id/nameUser"
        android:textSize="20sp"
        android:layout_marginTop="15dp"
        android:layout_marginLeft="20dp"
        android:layout_toRightOf="@id/imageUser"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/fightButton"
        android:background="@drawable/button_fight"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:layout_width="60dp"
        android:layout_height="60dp" /> />
    <Button
        android:id="@+id/addUserButton"
        android:background="@drawable/button_add"
        android:layout_toLeftOf="@id/fightButton"
        android:layout_centerVertical="true"
        android:layout_width="60dp"
        android:layout_height="60dp" /> />
</RelativeLayout>
```

Figura 67: Layout (item_friend.xml) del adapter personalizado "Amigo"

2. El siguiente paso es definir la clase “Friend”, que nos permitirá crear objetos con los atributos de nuestros amigos. Como elementos tendrá: un objeto “Drawable” con la imagen de perfil, un “String” con el nombre de usuario, un “int” con las partidas ganadas y un “long” con su id. Además de las funciones para establecer y recuperar esos atributos:

```

public class Friend {
    protected Drawable foto;
    protected String nombre;
    protected int partidasGanadas = -1;
    protected long id;

    public Friend(Drawable foto, String nombre, long id) {
        super();
        this.foto = foto;
        this.nombre = nombre;
        this.id = id;
    }

    public Drawable getFoto() {
        return foto;
    }

    public void setFoto(Drawable foto) {
        this.foto = foto;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }
}

```

Figura 68: Implementación Clase "Friend"

3. Es el momento de ver la implementación de la clase “SearchFriend”, donde se introduce el nombre del usuario a buscar (o el comienzo del mismo) y se reciben los datos coincidentes del servidor.

Los atributos son los siguientes:

```

public class SearchFriendActivity extends Activity{

    EditText friendEditText;
    Button bSearch;
    String friend, user;
    TextView matchesText;
    RelativeLayout relative;
    int matches=-1;
    String[] friendsReturn, imagesReturn; //Array de String que almacenará los nombres de usuario y url de las imagenes
                                         //devueltos por el servidor

    Handler mHandler;
    static ArrayList<Friend> arraydir;
    Drawable[] imagenes;
    HttpPostaux post;
    String IP_Server=MainActivity.IP_Server;//IP DE NUESTRO PC
    //String IP_Server="172.19.177.130";
    String URL_connect="http://"+IP_Server+"/EnglishPursuit/searchfriend.php";//ruta en donde estan nuestros archivos
    private ProgressDialog pDialog;
}

```

Figura 69: Atributos Clase "SearchFriendActivity"

Al hacer click en el botón de búsqueda, se ejecuta la tarea en segundo plano encargada de realizar la petición post al script “searchfriend” y de enviarle los datos.

```

bSearch.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        friend = friendEditText.getText().toString();

        new asyncSearchFriend().execute(friend);
    }
});

//Este método envía por post el nombre del alias del amigo a buscar y recibe las posibles
//coincidencias mostrándolas en el listView del layout de la activity
public void sendQuery(String friend){

    ArrayList<NameValuePair> postparameters2send= new ArrayList<NameValuePair>();
    postparameters2send.add(new BasicNameValuePair("usuario",user));
    postparameters2send.add(new BasicNameValuePair("friend",friend));
    JSONArray jdata=post.getServerdata(postparameters2send, URL_connect);

    JSONObject json_data;
    try {
        //matches contendrá el número de coincidencias de usuarios que devolverá la petición
        matches=(int) jdata.getJSONObject(0).getInt("coincidencias");
    } catch (JSONException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    friendsReturn= new String[matches];
    imagesReturn= new String[matches];
    imagenes= new Drawable[matches];
    //Creamos las views necesarias para mostrar todos los usuarios devueltos
    //inicializamos i a 1 porque el valor 0 corresponde al número de coincidencias,
    //los siguientes ya corresponden al nombre de los usuarios
    for (int i =1; i<matches+1; i++){
        try {
            friendsReturn[i-1]= jdata.getJSONObject(((i-1)*2)+1).getString("usuario");
            imagesReturn[i-1]= jdata.getJSONObject(((i-1)*2)+2).getString("imagen");

        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    ImageByURL image = new ImageByURL();

    for(int i =0; i<matches; i++){
        Bitmap b = image.downloadImage(imagesReturn[i]);
        imagenes[i]=new BitmapDrawable(getResources(),b);
    }
}

```

Figura 70: Listener botón “buscarAmigo”

Los datos de respuesta del servidor se recogen y se almacenan en los array de String (“friendsReturn” para el nombre de usuario e “imagesReturn” para las url de sus imágenes de perfil).

Para ver cuáles son estos datos de respuestas y cómo se obtienen, a continuación se muestra el script del servidor complementario a esta comunicación:

```

<?php
//usuario es el alias del usuario que busca a un amigo y se utiliza para que no se muestre a si mismo
// (y así no pueda crear una partida consigo mismo ni añadirse como amigo)
//friend es el alias del amigo que se quiere buscar

$usuario = $_POST['usuario'];
$friend = $_POST['friend'];

require_once 'funciones_bd.php';
$db = new funciones_bd();

/*Este script utiliza la variable $friend pasada como parámetro para buscar en la tabla "usuarios" los usuarios cuyo
alias empiece o corresponda con dicha variable

*/
//result devuelve las filas de los alias coincidentes
$result = ($db->searchfriend($usuario, $friend));

echo json_encode($result);
?>

public function searchfriend($usuario, $friend){
    $id_usuario = $this->getIdByAlias($usuario);

    //obtenemos como mucho 30 usuarios va que si hay muchos y se obtienen todos, la carga podría ser demasiada

    //además sólo vamos a devolver los usuarios que aún no son amigo del usuario que busca
    $result=mysql_query("SELECT * FROM usuarios WHERE (alias LIKE '$friend%' AND alias!='$usuario'
    AND ID NOT IN (SELECT ID_USUARIO2 FROM amigo_de WHERE ID_USUARIO1 ='$id_usuario')) LIMIT 30");

    /*el primer valor que almacenaremos en el json array devuelto será el número de coincidencias, para que la aplicación android sepa
    cuantos elementos debe añadir a la lista*/

    $num_rows = mysql_num_rows($result);

    $resultado[]=array("coincidencias"=>$num_rows);
    while ($row = mysql_fetch_array($result)){
        $resultado[]=array("usuario"=>$row['alias']);
        $resultado[]=array("imagen"=>$row['imagen']);
    }
    return $resultado;
}

```

Figura 71: Script "searchfriend"

Al recibir estos datos, se descargan las imágenes de perfil de los usuarios y se convierten en objetos “Drawables” que se almacenarán en el array apropiado.

Por último y una vez que se tienen rellenas las estructuras de datos anteriores, se van a crear los objetos “Friend” y se va a rellenar un “ArrayList” con ellos para crear los “Adapter” y finalmente asignárselos al “ListView” de la “activity”:

```

ListView lista = (ListView) findViewById(R.id.listaFriends);

arraydir = new ArrayList<Friend>();

Friend[] friendsT = new Friend[matches];

for(int i=0; i<matches; i++){

    friendsT[i] = new Friend(imagenes[i], friendsReturn[i], i);
    arraydir.add(friendsT[i]);

}

// Creo el adapter personalizado
AdapterFriends adapter = new AdapterFriends(SearchFriendActivity.this, arraydir);

// Lo aplico
lista.setAdapter(adapter);

```

Figura 72: Llenando la lista con el “adapter” personalizado

4. Es el turno de ver la clase del “Adapter” personalizado y los atributos definidos en ella:

```
public class AdapterFriends extends BaseAdapter{

    protected Activity activity;
    protected ArrayList<Friend> items;
    public static String[] friendsReturn;
    String usuario;
    Handler mHandler;
    HttpPostaux post;
    String IP_Server=MainActivity.IP_Server;//IP DE NUESTRO SERVIDOR
    ProgressDialog pDialog;
    static int userToAdd=-1;
}
```

Figura 73: Atributos Clase "AdapterFriends"

“items” es el elemento más relevante ya que es la lista que contendrá a los usuarios devueltos a la hora de realizar la consulta al servidor en la clase “SearchFriend”.

Dicha lista, coincide con el segundo parámetro del constructor y se corresponde con la lista de amigos creada en la clase “SearchFriend”, por lo que cuando se cree el “Adapter” ya tendrá la lista devuelta.

```
public AdapterFriends(Activity activity, ArrayList<Friend> items) {
    this.activity = activity;
    this.items = items;
    post=new HttpPostaux();
    mHandler=new Handler();

    SharedPreferences prefs =
        activity.getSharedPreferences("Preferencias",Context.MODE_PRIVATE);

    usuario = prefs.getString("user", "");
}
}
```

Figura 74: Constructor de la Clase "AdapterFriends"

Ahora, obtenemos los objetos “Friend” de la lista y recogemos sus atributos:

```
// Creamos un objeto Friend
final Friend dir = items.get(position);
//Rellenamos la fotografía
ImageView foto = (ImageView) v.findViewById(R.id.imageUser);
foto.setImageDrawable(dir.getFoto());
//Rellenamos el nombre
TextView nombre = (TextView) v.findViewById(R.id.nameUser);
nombre.setText(dir.getNombre());
```

Figura 75: Obteniendo objetos "Friend" en el Adapter

Dichos atributos son utilizados, tanto para rellenar las “Views” de la interfaz, como para enviar los datos correctos al servidor cuando se haga clic en un botón del “Adapter”.

9.6 Añadir amigo

La tarea “añadir amigo” está íntimamente relacionada con los “Adapters” explicados en el punto anterior y es que los “Listeners” de los botones que realizan esta función, se encuentran dentro de éstos.



Figura 76: Diseño "Añadir amigo"

Como hemos dicho, el “Adapter” tiene los atributos del “Friend” al que representa. Estos atributos (en concreto el nombre) serán usados para enviarlos al servidor a la hora de añadir a un usuario como amigo. Sin más, el código del “Listener” del botón de “agregar amigo” es el siguiente:

```
Button bAddUser = (Button) v.findViewById(R.id.addUserButton);

bAddUser.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        new asyncAddFriend().execute ((String)dir.nombre);
    }

});
```

Figura 77: Listener botón añadir amigo

Puede verse, que el “Listener” ejecuta la tarea asíncrona para añadir a un amigo y le pasa como parámetro a ésta el nombre de dicho amigo. Este nombre será utilizado por el script del servidor, para realizar la inserción pertinente en la tabla “partidas” de la BBDD:

```

<?php

$user = $_POST['usuario'];
//necesitamos codificar la variable "friend" enviada por POST
//en el juego de caracteres UTF-8 para poder trabajar con ella
$friend = utf8_encode($_POST['friend']);

require_once 'funciones_bd.php';
$db = new funciones_bd();

/*Este script utiliza las variables $user y $friend pasadas como parámetros
para insertar los registros necesarios en las tablas
estableciendo una relación de "amistad".
*/
//result true or false dependiendo de si la inserción ha sido exitosa o no
$result = ($db->addfriend($user,$friend));

$resultado[]=array("exitoso"=>$result);

echo json_encode($resultado);
?>

```

```

public function addfriend($usuario, $friend){

//Obtenemos el id del usuario que desea añadir un amigo
$id_usuario =mysql_query("SELECT * FROM usuarios WHERE alias = '$usuario'");
$id_usuario = mysql_fetch_array($id_usuario);
$id_usuario=$id_usuario['ID'];

//Obtenemos el id del amigo a añadir
$id_friend =mysql_query("SELECT * FROM usuarios WHERE alias = '$friend'");
$id_friend = mysql_fetch_array($id_friend);
$id_friend=$id_friend['ID'];

$result=mysql_query("INSERT INTO amigo_de (id_usuario1, id_usuario2) VALUES('$id_usuario','$id_friend')");

if($result) return true;
else return false;

}

```

Figura 78: Script "addfriend"

9.7 Crear partida

Cuando el usuario decide crear una partida, puede hacerlo de dos modos: seleccionando a un amigo como contrincante o dejando la elección del rival en manos de la suerte y jugar contra un oponente aleatorio. Por ello debemos distinguir entre las dos situaciones.

9.7.1 Contra amigo

Nos encontramos en la misma situación del apartado 9.6.

La implementación es exactamente la misma, lo único que cambia es el script del servidor ya que las consultas para realizar esta acción (crear una partida) serán distintas a las de añadir a un amigo:

```

<?php

//Este script se lanza cuando el usuario desea iniciar una partida contra un oponente amigo (conocido)

$susuario = $_POST['usuario'];
$friend = utf8_encode($_POST['friend']);

require_once 'funciones_bd.php';
$db = new funciones_bd();

$id_opponent = $db->getIdByAlias($friend);
$id_user = $db->getIdByAlias($susuario);

//ahora se procede a la inserción de los datos correspondientes en la tabla "partidas"

$result = ($db->createMatch($id_user, $id_opponent));

$resultado[] = array("exit0" => $result);
$resultado[] = array("opponente" => $friend);

//Una vez creada la partida, enviamos por gcm una notificación push al rival, para que sepa de nuestra invitación
$db->gcmSend($susuario, $friend, "te ha invitado a una partida");
echo json_encode($resultado);
?>

```

Figura 79: Script "createMatchWithFriend"

La función “createMatch” del script “funciones_db” es la encargada de realizar la inserción de datos y de crear el documento xml que encapsula todas las preguntas de una partida y que posteriormente será parseado para mostrarlas en la aplicación Android. En esta primera parte de la función se insertan los datos en las tablas y se crean algunos los nombres de las etiquetas que contendrá el xml. Además, para evitar que las preguntas se repitan en una misma partida, se generará un aleatorio que no podrá repetirse y con él se realizará la extracción:

```

public function createMatch($id_user, $id_opponent){

    //el turno lo empieza teniendo el oponente
    $consulta="INSERT INTO partida(estado, id_usuario1, id_usuario2, turno) VALUES(1, '$id_user', '$id_opponent', '$id_opponent')";
    $result = mysql_query($consulta);
    //Con mysql_insert_id() obtenemos el id de la nueva partida insertada, el cual nos hace falta para crear el xml con las preguntas
    //y respuestas de dicha partida
    $last_id = mysql_insert_id();

    //si la inserción se hace bien, pasamos a generar el xml
    if($result){

        //muy importante, mysql_query ("SET NAMES 'utf8'") nos permite establecer el conjunto de caracteres de las consultas a UTF-8,
        //si no hacemos esto aparecerán caracteres extraños no reconocibles
        mysql_query ("SET NAMES 'utf8'");

        $xml=new DomDocument("1.0","UTF-8");

        $partida=$xml->createElement("partida");
        $partida=$xml->appendChild($partida);
        //bucle para encapsular las preguntas de las 5 categorías
        for ($i=1; $i<=5; $i++){

            $categoria=$xml->createElement("categoria","uno");
            $categoria=$partida->appendChild($categoria);
            $indiceDePregunta = array();
            //bucle para encapsular las 5 preguntas de una categoría
            for ($j=1; $j<=5; $j++){
                //recorrido las 5 preguntas aleatorias de una misma categoría
                $preguntaBD = mysql_query("SELECT * FROM preguntas WHERE ID_CATEGORIA= '$i'");
                //obtenemos el número de preguntas que hay de una categoría
                $num_rows = mysql_num_rows($preguntaBD);

                //generamos un array de enteros donde insertamos los índices de las preguntas aleatorias
                //para comparar los números aleatorios que se van generando con los generados previamente
                //y generar siempre distintos. De esta manera se evita que las partidas tengan preguntas repetidas

                do{
                    //calculamos un random entre 0 y el número de preguntas -1
                    $randomRow = rand(0,$num_rows-1);

                }while(in_array($randomRow, $indiceDePregunta));

                array_push($indiceDePregunta,$randomRow);
                for ($k=0; $k<=$randomRow; $k++){
                    $preguntaAleatoria = mysql_fetch_array($preguntaBD);
                }
                $enunciado=$preguntaAleatoria['enunciado'];
                $traduccion=$preguntaAleatoria['traduccion'];
                $id_respuesta1=$preguntaAleatoria['id_respuesta1'];
                $id_respuesta2=$preguntaAleatoria['id_respuesta2'];
                $id_respuesta3=$preguntaAleatoria['id_respuesta3'];
                $id_respuesta4=$preguntaAleatoria['id_respuesta4'];
                $id_respuesta_correcta=$preguntaAleatoria['id_respuesta_correcta'];
            }
        }
    }
}

```

Figura 80: Función php “createMatch” Parte 1

El texto entre las etiquetas se obtiene de las consultas en la base de datos. Por ejemplo, el texto que irá entre las etiquetas <enunciado> </enunciado>, se obtiene de la columna “enunciado” de la tabla “preguntas”.

Finalmente, cuando se haya recogido toda la información, el xml se almacenará en el servidor con el nombre del identificador de partida (para que la aplicación Android sepa qué documento parsear según la partida en cuestión).

```

$respuesta1 = mysql_query("SELECT * FROM respuestas WHERE ID='$id_respuesta1'");
$respuesta2 = mysql_query("SELECT * FROM respuestas WHERE ID='$id_respuesta2'");
$respuesta3 = mysql_query("SELECT * FROM respuestas WHERE ID='$id_respuesta3'");
$respuesta4 = mysql_query("SELECT * FROM respuestas WHERE ID='$id_respuesta4'");
$respuesta_correcta = mysql_query("SELECT * FROM respuestas WHERE ID='$id_respuesta_correcta'");

$respuesta1 = mysql_fetch_array($respuesta1);
$respuesta2 = mysql_fetch_array($respuesta2);
$respuesta3 = mysql_fetch_array($respuesta3);
$respuesta4 = mysql_fetch_array($respuesta4);
$respuesta_correcta = mysql_fetch_array($respuesta_correcta);

$enun_respuesta1 = $respuesta1['enunciado'];
$enun_respuesta2 = $respuesta2['enunciado'];
$enun_respuesta3 = $respuesta3['enunciado'];
$enun_respuesta4 = $respuesta4['enunciado'];
$enun_respuesta_correcta = $respuesta_correcta['enunciado'];

$pregunta1=$xml->createElement("pregunta");
$pregunta1=$categoria->appendChild($pregunta1);

$enunciado1=$xml->createElement("enunciado", $enunciado);
$enunciado1=$pregunta1->appendChild($enunciado1);

$traduccion1=$xml->createElement("traduccion", $traduccion);
$traduccion1=$pregunta1->appendChild($traduccion1);

$respuesta1=$xml->createElement("respuesta1", $enun_respuesta1);
$respuesta1=$pregunta1->appendChild($respuesta1);
$respuesta2=$xml->createElement("respuesta2", $enun_respuesta2);
$respuesta2=$pregunta1->appendChild($respuesta2);
$respuesta3=$xml->createElement("respuesta3", $enun_respuesta3);
$respuesta3=$pregunta1->appendChild($respuesta3);
$respuesta4=$xml->createElement("respuesta4", $enun_respuesta4);
$respuesta4=$pregunta1->appendChild($respuesta4);

$valida=$xml->createElement("respuesta_correcta", $enun_respuesta_correcta);
$valida=$pregunta1->appendChild($valida);
}
}

/*guardar xml-----*/
// $xml->preserveWhiteSpace = false;
$xml->encoding = 'utf-8';
header('Content-type: text/html; charset=utf-8');
$xml->formatOut=true;

$string_xml=$xml->saveXML();

$xml->save("partidas/$last_id". ".xml");

return true;
}

```

Figura 81: Función php “createMatch” Parte 2

9.7.2 Contra rival aleatorio

Cuando un usuario crea una partida contra un rival aleatorio se ejecutan casi los mismos pasos que en la situación anterior. La única diferencia es que en vez de enviar como parámetro al script, el nombre de nuestro amigo, será el propio script quien seleccione un rival aleatoriamente, con la única restricción de no seleccionar un usuario con el que ya estamos jugando. De esta forma, el script es el siguiente:

```

<?php
//Este script se lanza cuando el usuario desea iniciar una partida contra un oponente aleatorio

//para devolver el rival con el que se va a entablar la partida se aplica un rand desde 1 hasta el número de usuarios en la bbdd
//int rand ( int $min , int $max )
$susuario = $_POST['usuario'];

require_once 'funciones_bd.php';
$db = new funciones_bd();

//también debemos verificar que el oponente con el que crearemos la partida no esté involucrado como rival en
//una partida creada previamente, para ello obtenemos las partidas en las que participemos
$id_user = $db->getIdByAlias($susuario);
$partidasYaCreadas = ($db->viewMatches($id_user));
//obtenemos un array cuyos valores son los nombres de los oponentes de las partidas creadas
$oponentesDePartidasYaCreadas =(array_column($partidasYaCreadas, 'opponent'));

//obtenemos el alias del oponente aleatorio verificando que este oponente devuelto no seamos nosotros mismos
//ni sea un usuario con el que ya estamos jugando una partida. Si una de las dos cosas ocurre, se repetirá la consulta
do {
    $opponent = ($db->randomUser());
} while (($opponent == $susuario) || (in_array($opponent, $oponentesDePartidasYaCreadas)));

//obtenemos el id del alias obtenido anteriormente (ya que será el id el que utilizaremos para la
//inserción en la BBDD "partidas"

$id_opponent = $db->getIdByAlias($opponent);
//ahora se procede a la inserción de los datos correspondientes en la tabla "partidas"

$result = ($db->createMatch($id_user, $id_opponent));
$resultado[]=array("exito"=>$result);
$resultado[]=array("opponent"=>$opponent);
//Una vez creada la partida, enviamos por gcm una notificación push al rival, para que sepa de nuestra invitación
$db->gcmSend($susuario, $opponent, "se ha invitado a una partida");

echo json_encode($resultado);
?>

```

Figura 82: Script createMatchWithRandom

9.8 Eliminar amigo



Figura 83: Diseño "eliminar usuario"

Es muy similar a la tarea de “Añadir amigo”, lo único que cambia, de nuevo, es el script del servidor, que en vez de hacer una inserción en la tabla “amigo_de”, realiza un borrado de la tupla objetivo.

```
public function deletefriend($usuario, $friend){
    //Obtenemos el id del usuario que desea eliminar al amigo
    $id_usuario =mysql_query("SELECT * FROM usuarios WHERE alias = '$usuario'");
    $id_usuario = mysql_fetch_array($id_usuario);
    $id_usuario=$id_usuario['ID'];

    //Obtenemos el id del amigo a eliminar
    $id_friend =mysql_query("SELECT * FROM usuarios WHERE alias = '$friend'");
    $id_friend = mysql_fetch_array($id_friend);
    $id_friend=$id_friend['ID'];

    $result=mysql_query("DELETE FROM amigo_de WHERE (ID_USUARIO1='$id_usuario' AND ID_USUARIO2='$id_friend')");

    if($result) return true;
    else return false;
}
```

Figura 84: Script "deletefriend"

9.9 Eliminar partida (rendirse)

Al igual que en el caso anterior el script realiza un borrado, aunque esta vez sobre la tabla “partidas”. Además de esto, también elimina el xml asociado a la partida, para así liberar espacio en el servidor:

```
public function surrenderMatch($id_match, $id_usuario, $id_oponente){
    //borramos la partida de la base de datos
    $result=mysql_query("DELETE FROM partidas WHERE ID='$id_match'");

    //actualizamos las estadísticas de ambos jugadores
    $actualizarPartidasGanadas= mysql_query("UPDATE estadisticas SET PG=PG+1 WHERE ID_USUARIO='$id_oponente'");
    $actualizarPartidasPerdidas= mysql_query("UPDATE estadisticas SET PP=PP+1 WHERE ID_USUARIO='$id_usuario'");
    //además borramos el fichero .xml asociado a dicha partida para liberar espacio del servidor
    unlink("partidas/$id_match"."xml");
    return $result;
}
```

Figura 85: Función "surrenderMatch"

9.10 Ver perfil



Figura 86: Diseño "ProfileFragment"

En la pantalla de perfil se muestran las estadísticas del usuario, para ello se realiza una consulta en la tabla “estadísticas” de la BBDD:

```
//Esta función devuelve las estadísticas de un usuario
public function viewStats($id_usuario){

    $estadisticas=mysql_query("SELECT * FROM estadisticas WHERE ID_USUARIO = '$id_usuario'");
    $estadisticas = mysql_fetch_array($estadisticas);
    $resp_correctas = $estadisticas['RC'];
    $resp_incorrectas = $estadisticas['RI'];
    $par_ganandas = $estadisticas['PG'];
    $par_perdidas = $estadisticas['PP'];

    $resultado[]=array("respuestas_correctas"=>$resp_correctas);
    $resultado[]=array("respuestas_incorrectas"=>$resp_incorrectas);
    $resultado[]=array("partidas_ganadas"=>$par_ganandas);
    $resultado[]=array("partidas_perdidas"=>$par_perdidas);

    return $resultado;
}
```

Figura 87: Función "viewStats"

9.11 Ver clasificación

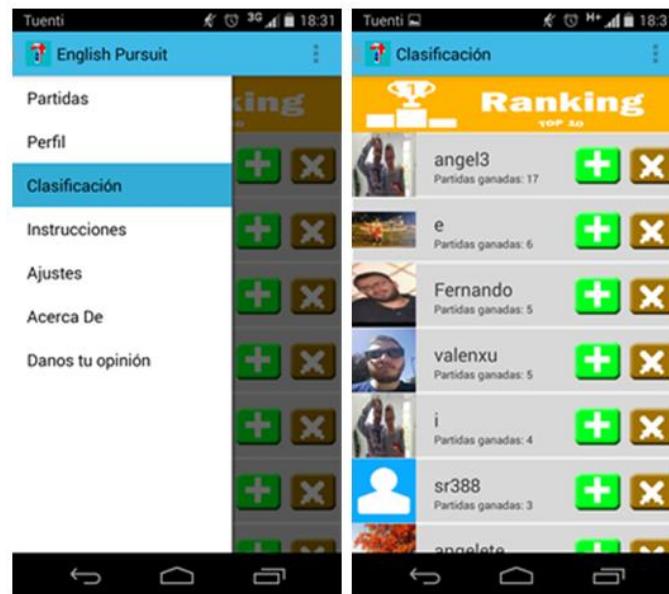


Figura 88: Diseño "Ranking Fragment"

Para mostrar la clasificación, la aplicación Android realiza una consulta a la BBDD para obtener los diez jugadores que hayan conseguido ganar más partidas:

```
//Esta función devuelve el ranking con los mejores jugadores de "EnglishPursuit", es decir,
//devuelve los nombres de los jugadores ordenados de mayor a menor partidas ganadas
public function viewRanking(){

    $result=mysql_query("SELECT * FROM estadisticas ORDER BY PG DESC LIMIT 10");

    while ($row = mysql_fetch_array($result)){

        $id_usuario = $row['ID_USUARIO'];
        $usuario = $this->getAliasById($id_usuario);
        $imagen = $this->getImageById($id_usuario);
        $par_ganandas = $row['PG'];

        $resultado[]=array("usuario"=>$usuario);
        $resultado[]=array("partidas_ganadas"=>$par_ganandas);
        $resultado[]=array("imagen"=>$imagen);

    }
    return $resultado;
}
```

Figura 89: Función "viewRanking"

9.12 Ver instrucciones



Figura 90: Diseño interfaz "Instrucciones"

Las instrucciones son un recurso que pueden consultar los usuarios para conocer cuál es el objetivo de English Pursuit, cómo se desarrollan las partidas, el contenido de las mismas y cómo se determina el vencedor. Esta vez, para construir su interfaz, en vez de utilizar una “activity” se ha escrito un documento .html (con una hoja de estilos asociada) almacenado en el servidor que será visualizado por medio del navegador del dispositivo. El código para realizar esta sencilla tarea es el siguiente:

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://" + MainActivity.IP_Server + "/Web/rules.html"));
this.startActivity(intent);
```

Figura 91: Código para mostrar las instrucciones

Lo único que hay que hacer es crear y ejecutar un “intent” del tipo “ACTION_VIEW” con la url del .html en cuestión.

9.13 Dar sugerencias y opiniones

Un usuario puede querer informarnos de su opinión o hacer alguna sugerencia acerca de English Pursuit, para ello se abrirá la aplicación de correo con los campos de destinatario y asunto rellenos, facilitando la redacción. El código que implementa esto es:

```

Intent itSend = new Intent(android.content.Intent.ACTION_SEND);
itSend.setType("plain/text");
//colocamos los datos para el envío
itSend.putExtra(android.content.Intent.EXTRA_EMAIL, new String[]{ "movilkapp@gmail.com"});
itSend.putExtra(android.content.Intent.EXTRA_SUBJECT, "EnglishPursuit");

//iniciamos la actividad
startActivity(itSend);

```

Figura 92: Código para mostrar la aplicación de correo

Es tan sencillo como crear un “intent” del tipo “*ACTION_SEND*” con el destinatario del correo y el asunto.

9.14 Contestar preguntas



Figura 93: Diseño de "MatchActivity"

Para contestar las preguntas pertenecientes a una partida, el usuario pulsará el botón “Jugar partida” de la partida correspondiente, entonces se ejecutará la clase **MatchActivity**, la cual se encargará de leer las preguntas de un documento xml alojado en el servidor (el objetivo de tener un único xml con las preguntas de una partida es que ambos jugadores accedan a él para leer las mismas preguntas y de esta manera jugar una partida equilibrada) y mostrar estas preguntas, para que el usuario las conteste. Dependiendo de si el usuario acierta o falla se ejecutará distinto código, como veremos más adelante. Finalmente se enviarán los resultados al servidor.

Los atributos definidos en la clase **MatchActivity** pueden verse en la siguiente figura:

```

public class MatchActivity extends Activity{

    Activity activity = this;
    TextView categoryTitleTextView, questionTextView, countdownTextView, respCorrectTextView, respIncorrectTextView,
    Button answersButtons[] = new Button[4];
    SoundPool sp;
    String categorias[] = {"Phrasal Verbs", "Prepositions", "Pronouns", "Vocabulary", "Irregular Verbs"};
    int flujoSonidoCorrect = -1;
    int flujoSonidoIncorrect = -1;
    int flujoSonidoClock = -1;
    int respuestasCorrectas = 0;
    int respuestasIncorrectas = 0;
    int numeroPregunta = 1;
    int idPartida, ronda;
    String oponente, usuario;
    private List<Pregunta> preguntas;
    HttpPostaux post;
    //URL donde se encuentra el xml a parsear
    String URL_connect;
    //contador de tiempo
    CountdownTimer ct;
    //animacion
    Animation animacion;

```

Figura 94: Atributos Clase "MatchActivity"

Se necesitan una serie de “TextView” para mostrar la información, los cuatro botones de las respuestas, la lista de preguntas recogidas por el parseador (que se verán más adelante), la cuenta atrás y otros atributos para el correcto funcionamiento y la inserción en la base de datos externa.

Una vez que se haya recogido y mostrado la pregunta y sus respectivas respuestas, el usuario tendrá que pulsar sobre el botón de una respuesta para contestar. Además deberá actuar con rapidez ya que hay un contador de 50 segundos que cuando llegue a cero no dejará contestar más preguntas. Para ello, cada botón de respuesta tiene un listener que llamará a una función para comprobar si la respuesta elegida por el usuario es correcta o no:

```

answersButtons[0].setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        checkIsCorrect(answersButtons[0]);
    }
});

```

Figura 95: Listener de botones respuesta

La función “checkIsCorrect” dirá si la respuesta es correcta y ejecutará las acciones necesarias para actualizar variables y mostrar la siguiente pregunta (o terminar la ronda en caso de que fuera la última):

CAPÍTULO 9

```
//función que se ejecuta cuando el usuario selecciona una respuesta, aquí se comprueba si es o no correcta
public void checkIsCorrect(Button boton){
    //si el texto del botón es igual que el texto de la respuesta correcta, hemos acertado
    if(boton.getText().toString().equals(preguntas.get(numeroPregunta-1+((ronda-1)*5)).getRespuestas(4).toString())){
        boton.setBackgroundResource(R.drawable.button_correct);
        sP.play(flujoSonidoCorrect, 1.0f, 1.0f, 1, 0, 1);
        respuestasCorrectas++;
        respCorrectTextView.setText(""+respuestasCorrectas);
        siguientePregunta();
    }

    //si no es correcta
    else{

        sP.setVolume(sP.play(flujoSonidoIncorrect, 1.0f, 1.0f, 1, 0, 1), 0.5f, 0.5f);
        boton.setBackgroundResource(R.drawable.button_incorrect);

        //la pregunta actual será numeroPregunta-1 ya que la lista empieza con el índice 0
        //y + (ronda-1)*5 ya que en cada ronda hay 5 preguntas
        for(int i =0; i<answersButtons.length; i++){
            if(answersButtons[i].getText().toString().equals(preguntas.get(numeroPregunta-1+((ronda-1)*5)).getRespuestas(4).toString())){
                answersButtons[i].setBackgroundResource(R.drawable.button_correct);
            }
        }
        respuestasIncorrectas++;
        respIncorrectTextView.setText(""+respuestasIncorrectas);
        siguientePregunta();
    }
}
```

Figura 96: Implementación función "checkIsCorrect"

Si la pregunta contestada es la última (lo mismo ocurre si se acaba el tiempo), la ronda terminará y se mostrará un “AlertDialog” mostrando el resumen al usuario. Además se procederá al envío de datos al servidor, para actualizar la tabla “partidas”.

```
if(numeroPregunta >5){
    //si numeroPregunta es mayor que 5 entonces el usuario ha contestado a las 5 preguntas de la ronda
    //y ésta se da por terminada

    AlertDialog.Builder builder =
        new AlertDialog.Builder(activity);

    builder.setMessage("Has terminado la ronda de " +categorias[ronda-1]+ " con "+
        respuestasCorrectas+" aciertos y "+respuestasIncorrectas+" fallos.")
        .setTitle("Ronda terminada")
        .setPositiveButton("OK", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                new setTurnAndRound().execute();
                activity.finish();
            }
        });

    AlertDialog dialogoConfirmacion = builder.create();
    dialogoConfirmacion.setCancelable(false);
    dialogoConfirmacion.show();

    ct.cancel();
}
```

Figura 97: Finalización de ronda

La tarea asíncrona llama al script “playMatch.php” y le envía los parámetros mostrados en la siguiente figura:

```

public void playMatch(){

    String URL = "http://"+MainActivity.IP_Server+"/EnglishPursuit/playMatch.php";
    ArrayList<NameValuePair> postparameters2send= new ArrayList<NameValuePair>();

    postparameters2send.add(new BasicNameValuePair("usuario",usuario));
    postparameters2send.add(new BasicNameValuePair("oponente",oponente));
    postparameters2send.add(new BasicNameValuePair("resp_cor", Integer.toString(respuestasCorrectas)));
    postparameters2send.add(new BasicNameValuePair("id_partida",Integer.toString(idPartida)));

    JSONArray jdata=post.getServerdata(postparameters2send, URL);

```

Figura 98: Petición al script "playMatch"

Dichos parámetros son recogidos por el script del servidor y utilizados para la inserción de datos en la BBDD, como se muestra en la siguiente figura:

```

<?php
//Este script se ejecutará cuando el jugador finalice la ronda de una partida

$usuario = $_POST['usuario'];
$oponente = $_POST['oponente'];
$resp_cor = $_POST['resp_cor'];
$id_partida = $_POST['id_partida'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

$id_usuario = ($db->getIdByAlias($usuario));
$id_oponente = ($db->getIdByAlias($oponente));

$result = ($db->playMatch($id_partida, $id_usuario, $id_oponente, $resp_cor));
$db->gcmSend($usuario, $oponente, "ha terminado la ronda, te toca a ti");
?>

```

```

//Esta función cambia el turno de una partida al oponente y en caso de que el jugador sea el creador de la partida, ésta aumenta de ronda
//además actualiza la información de la partida con las respuestas correctas de cada jugador
public function playMatch($id_partida, $id_usuario, $id_oponente, $resp_cor){

//Actualiza el turno para dárselo al oponente
$result = mysql_query("UPDATE partidas SET TURNO='$id_oponente' WHERE ID='$id_partida'");

//Si el que va a jugar es el creador (es decir, el usuario 1), la ronda aumenta al igual que el número de sus respuestas correctas
$result=mysql_query("SELECT * FROM partidas WHERE ID='$id_partida'");

$partida = mysql_fetch_array($partida);
$id_creador=$partida['ID_USUARIO1'];
$ronda=$partida['RONDA'];

if($id_creador==$id_usuario){

$result = mysql_query("UPDATE partidas SET RONDA=RONDA+1 WHERE ID='$id_partida'");
$result = mysql_query("UPDATE partidas SET RC_USUARIO1=RC_USUARIO1+'$resp_cor' WHERE ID='$id_partida'");
}

else{
$result = mysql_query("UPDATE partidas SET RC_USUARIO2=RC_USUARIO2+'$resp_cor' WHERE ID='$id_partida'");
}

//Si la ronda es la última, cuando un jugador acaba de contestar las preguntas, se harán inserciones en la BBDD
if($ronda==5){

$numPreg = 25;
$respuestas = mysql_query("SELECT * FROM partidas WHERE ID='$id_partida'");
$respuestas = mysql_fetch_array($respuestas);
$rc_usuario1 = $respuestas['RC_USUARIO1'];
$rc_usuario2 = $respuestas['RC_USUARIO2'];

```

Figura 99: Script "playMatch"

9.14.1 Animación de botones

Para aportar dinamismo a las partidas, se ha incluido una animación en los botones, de manera que éstos se deslizan por la pantalla cada vez que el usuario contesta una pregunta. Esto es posible gracias a las llamadas “Animaciones Tween”, las cuales permiten realizar transformaciones simples (posición, tamaño, rotación y transparencia) sobre un objeto View.

Para crear la animación se ha de crear un objeto del tipo “Animation” y cargar el recurso de animación especificado en un xml. Posteriormente cuando se quiera ejecutar, se llamará al método “startAnimation” del objeto “View” a animar pasando como parámetro la animación creada con anterioridad.

```
Animation animacion;

animacion = AnimationUtils.LoadAnimation(this,
    R.anim.animation_button_answer);

public void animarBotones(){
    for(int i=0; i<answersButtons.length;i++){
        answersButtons[i].startAnimation(animacion);
    }
}
```

Figura 100: Animación de botones

El .xml que contiene la animación contiene el siguiente código:

```
<translate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="1000"
    android:fromXDelta="700"
    android:toXDelta="0" />
```

Figura 101: Animación Tween de traslación

Lo único que hace esta animación es trasladar el objeto desde la distancia x=700 hasta x=0 en un segundo.

9.14.2 Parseador XML

Una de las partes más importantes del proyecto recae en el parseador XML que será utilizado por la aplicación para leer las preguntas correspondientes a una partida y

mostrarlas al usuario. Esta tarea se realiza en la clase *XMLParserPull* ubicada en el paquete *com.angeljose.englishpursuit.xml*.

Existen varios métodos que permiten parsear y extraer los datos de las etiquetas de un documento xml, entre ellos se encuentran: SAX, DOM y XmlPULL. Habiendo analizado los tres, finalmente se decidió utilizar el último por su fácil implementación y por la [recomendación](#) de Google, los pasos a seguir para realizar el paseo son los siguientes:

1. Lo primero que se debe hacer es pasarle al parseador la url donde se encuentra el documento xml perteneciente a una partida. Cada partida tiene un documento xml asociado ubicado en el servidor y cuyo nombre es el identificador de esa partida.

```
URL_connect="http://"+MainActivity.IP_Server+"/EnglishPursuit/partidas/"+idPartida+".xml";

//lanzamos la tarea asíncrona de cargarXml en segundo plano pasándole como único parámetro la url del xml
CargarXmlTask tarea = new CargarXmlTask();
tarea.execute(URL_connect);

//Tarea Asíncrona para cargar un XML en segundo plano
private class CargarXmlTask extends AsyncTask<String,Integer,Boolean> {

    protected Boolean doInBackground(String... params) {

        //al parseador le pasamos la url del xml
        XMLParserPull saxparser =
            new XMLParserPull(params[0]);

        preguntas = saxparser.parse();

        return true;
    }
}
```

Figura 102: Pasando la url del xml al parseador

2. Cuando el parser obtiene el documento, se procede a su lectura. A medida que se va leyendo el documento, ocurren una serie de eventos que nos permitirán extraer la información, entre ellos destacan:

- *XmlPullParser.START_DOCUMENT*: indica que comienza el documento XML.
- *XmlPullParser.END_DOCUMENT*: indica que termina el documento XML.
- *XmlPullParser.START_TAG*: indica que comienza una etiqueta. Podemos saber el nombre de dicha etiqueta con el método *getName()* y el texto a continuación con *nextText()*.
- *XmlPullParser.END_TAG*: indica que termina una etiqueta.

Gracias a estos eventos se puede recoger la información y crear las estructuras de datos necesarias para jugar la partida.

3. Cuando se detecta el inicio de documento **START_DOCUMENT**, se crea una lista (de objeto Preguntas) con las preguntas que van a aparecer en una partida:

```
public List<Pregunta> parse()
{
    List<Pregunta> preguntas = null;
    XmlPullParser parser = Xml.newPullParser();

    try
    {
        parser.setInput(this.getInputStream(), null);

        int evento = parser.getEventType();

        Pregunta preguntaActual = null;

        while (evento != XmlPullParser.END_DOCUMENT)
        {
            String etiqueta = null;

            switch (evento)
            {
                case XmlPullParser.START_DOCUMENT:
                    preguntas = new ArrayList<Pregunta>();
                    break;
```

Figura 103: Creando objetos pregunta al iniciar la lectura del xml

Antes de ver el siguiente paso, donde se asignan valores a los atributos de los objetos *Pregunta*, debe verse la propia clase pregunta:

```
public class Pregunta {
    private String enunciado;
    private String traduccion;
    //Array de respuestas donde 0-3 son las posibles y la 4 es la correcta
    private String respuestas[] = new String[5];

    public String getEnunciado() {
        return enunciado;
    }

    public String getTraduccion() {
        return traduccion;
    }

    public String getRespuestas(int i) {
        return respuestas[i];
    }

    public void setEnunciado(String t) {
        enunciado = t;
    }

    public void setTraduccion(String l) {
        traduccion = l;
    }

    public void setRespuestas(int i, String r) {
        respuestas[i] = r;
    }
}
```

Figura 104: Clase Pregunta

- Es el momento de leer las etiquetas del xml y obtener la información para asignársela a los atributos de los objetos *Pregunta*:

La estructura de un documento xml perteneciente a una partida y generado cuando ésta se creó, es la siguiente:

```

▼<partida>
  ▼<categoria>
    uno
    ▼<pregunta>
      <enunciado>My daughter ____ her mother.</enunciado>
      <traduccion>Mi hija se parece a su madre.</traduccion>
      <respuesta1>seems like</respuesta1>
      <respuesta2>takes after</respuesta2>
      <respuesta3>looks at</respuesta3>
      <respuesta4>leaves in</respuesta4>
      <respuestacorrecta>takes after</respuestacorrecta>
    </pregunta>
    ▼<pregunta>
      <enunciado>I'll always ____ ____ you.</enunciado>
      <traduccion>Siempre estaré a tu lado.</traduccion>
      <respuesta1>stand for</respuesta1>
      <respuesta2>stand by</respuesta2>
      <respuesta3>stand from</respuesta3>
      <respuesta4>stand to</respuesta4>
      <respuestacorrecta>stand by</respuestacorrecta>
    </pregunta>
  </categoria>
</partida>

```

Figura 105: Estructura documento XML de una partida

De esta manera y teniendo en cuenta el nombre de las etiquetas, se procede a su lectura por parte del parseador:

```

case XmlPullParser.START_TAG:
    etiqueta = parser.getName();

    if (etiqueta.equals("pregunta"))
    {
        preguntaActual = new Pregunta();
    }
    else if (preguntaActual != null)
    {
        if (etiqueta.equals("enunciado"))
        {
            preguntaActual.setEnunciado(parser.nextText());
        }
        else if (etiqueta.equals("traduccion"))
        {
            preguntaActual.setTraduccion(parser.nextText());
        }
        else if (etiqueta.equals("respuesta1"))
        {
            preguntaActual.setRespuestas(0, parser.nextText());
        }
        else if (etiqueta.equals("respuesta2"))
        {
            preguntaActual.setRespuestas(1, parser.nextText());
        }
    }
}

```

Figura 106: Obteniendo información de las etiquetas

- Como último paso, cuando se detecte el evento de cierre de etiqueta *XmlPullParser.END_TAG* y el nombre de dicha etiqueta sea “pregunta”, querrá decir que ya se ha terminado de leer una pregunta, entonces el objeto

“preguntaActual” (que es al que se le ha asignado la información recién recogida) se añade a la lista de preguntas. Después de esto, vuelve a repetirse la secuencia desde el paso 4 hasta que no queden preguntas que leer:

```
case XmlPullParser.END_TAG:
    etiqueta = parser.getName();
    if (etiqueta.equals("pregunta") && preguntaActual != null)
    {
        preguntas.add(preguntaActual);
    }
    break;
```

Figura 107: Añadiendo la pregunta a la lista

9.15 Envío de notificaciones Push con Google Cloud Messaging

Como ya se comentó (véase CAPÍTULO 4), English Pursuit implementa un sistema de notificaciones para avisar al usuario de ciertos eventos y permitir así un juego más inmediato y entretenido (de otra manera el usuario tendría que ejecutar la aplicación cada cierto tiempo para ver si tiene alguna partida en la que le pertenece el turno).

Para su implementación se utilizarán las keys ya mencionadas y los códigos GCM de cada usuario, almacenados en la tabla “usuarios” e insertados cuando el usuario se registró. La función php que permite realizar el envío de estas notificaciones es el siguiente:

```

//esta función enviará una notificación push al dispositivo del usuario "receiver"
public function gcmSend($sender, $receiver, $textToSend){

    $registatoin_ids = array();
    $result = mysql_query("SELECT *FROM usuarios WHERE alias = '$receiver'");
    while($row = mysql_fetch_assoc($result)){
        array_push($registatoin_ids, $row['gcmcode']);
    }
    // Set POST variables
    $url = 'https://android.googleapis.com/gcm/send';

    // $message = array("Notice" => $_POST['message']);
    $message = array("Notice" => $sender . " " . $textToSend);
    $fields = array(
        'registration_ids' => $registatoin_ids,
        'data' => $message,
    );
    $headers = array(
        'Authorization: key=AIzaSyDTP2T0gxEjuStvcC4hc2bfvOp36o-x1lk',
        'Content-Type: application/json'
    );
    // Open connection
    $ch = curl_init();

    // Set the url, number of POST vars, POST data
    curl_setopt($ch, CURLOPT_URL, $url);

    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    // Disabling SSL Certificate support temporarily
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($fields));

    // Execute post
    $result = curl_exec($ch);
    if ($result === FALSE) {
        die('Curl failed: ' . curl_error($ch));
    }

    // Close connection
    curl_close($ch);
}

```

Figura 108: Función "gcmSend"

9.16 Conclusiones

En este capítulo se han podido ver los fragmentos de código más importantes que permiten dotar de funcionalidad a la aplicación English Pursuit. Además se ha visto tanto la parte Android del cliente como los scripts PHP del servidor, los cuales colaboran para intercambiar información y sustentar un sistema en línea al que pueden acceder todos los usuarios sin problema alguno.

CAPÍTULO 10. EVALUACIÓN DE LA APLICACIÓN

Con motivo de obtener una valoración de la satisfacción de la aplicación se ha decidido planificar una prueba de satisfacción con un cuestionario SUS (System Usability Scale) que nos permite recoger opiniones anónimas acerca de varios aspectos de la aplicación.

El propósito de este cuestionario es proporcionar un test fácil de responder por los encuestados y que permita obtener unos resultados más o menos objetivos. Tiene una cantidad pequeña de preguntas y éstas se contestan puntuando en un rango del 1-5.

Para llevarlo a cabo se ha utilizado un servicio de Google que permite la creación de formularios de forma rápida y sencilla que además facilita la difusión y recopila los resultados en una hoja de cálculo para su posterior manipulación.

En total han participado 19 personas que han jugado a English Pursuit por lo menos en tres partidas diferentes. Los resultados obtenidos han sido los siguientes:

10.1 Preguntas de Etnografía

Antes de formular las preguntas específicas correspondientes a la usabilidad, se realizaron algunas preguntas etnográficas con motivo de determinar las características básicas de la población que participaba en el experimento.

10.1.1 Género

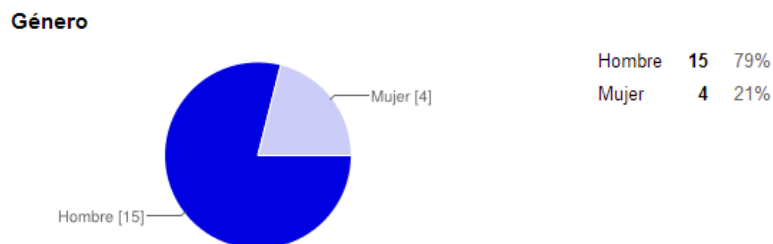


Figura 109: Pregunta sobre el género

En este diagrama de sectores se ha recogido el género de las personas encuestadas. De un vistazo puede verse como el género predominante es el masculino con un 79%.

10.1.2 Nivel de experiencia en Android

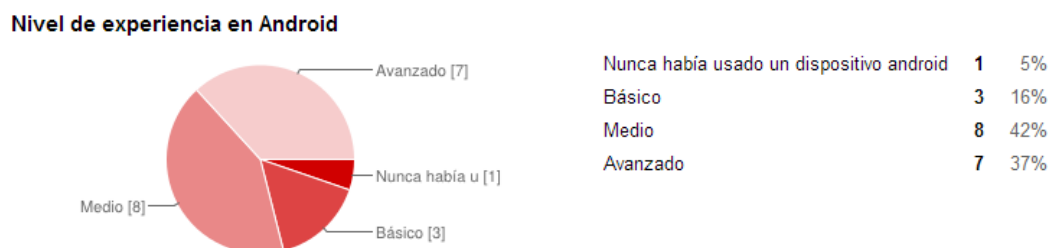


Figura 110: Pregunta sobre el nivel de experiencia en Android

En este diagrama de sectores se han obtenido los niveles de experiencia en Android de los participantes. En él puede apreciarse que la mayoría de usuarios tienen un nivel medio-avanzado, por lo que podrán jugar a la aplicación sin problemas.

10.1.3 Nivel de inglés

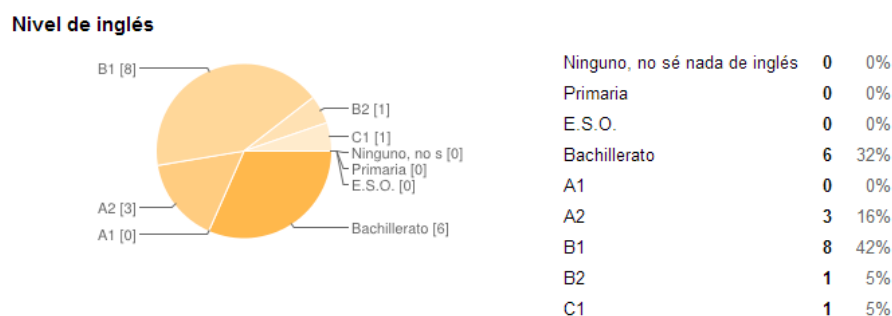


Figura 111: Pregunta sobre el nivel de inglés

Por su parte, este diagrama representa los niveles de inglés de los encuestados. La mayor parte de ellos disponen del nivel adquirido en bachillerato o el B1, así que podrán contestar muchas de las preguntas de English Pursuit sin problema.

10.2 Preguntas SUS

Las preguntas SUS propuestas en este cuestionario son las siguientes:

1. Creo que me gustará utilizar esta aplicación con frecuencia
2. Encontré la aplicación innecesariamente compleja
3. Pensé que la aplicación sería más fácil de usar
4. Creo que voy a necesitar el apoyo de un experto para utilizar la aplicación
5. Me pareció que las diversas funciones de esta aplicación se integran bien
6. Pienso que hay muchos aspectos mejorables en esta aplicación
7. La mayoría de la gente podrá utilizar esta aplicación sin problemas
8. Encontré la aplicación difícil de usar
9. Me sentí seguro utilizando la aplicación
10. Necesito aprender muchas cosas antes de manejarla bien con la aplicación

Teniendo en cuenta que la escala de respuesta a las preguntas oscila entre 1-5 (mostrando un nivel de desacuerdo en los valores bajos y un nivel mayor de acuerdo en los altos):

1	2	3	4	5
En desacuerdo ☐ ☐ ☐ ☐ ☐ De acuerdo				

Figura 112: Escala de respuesta a las preguntas

Estos son los resultados obtenidos para cada una de ellas:

10.2.1 Resultados de la pregunta “Creo que me gustará utilizar esta aplicación con frecuencia”

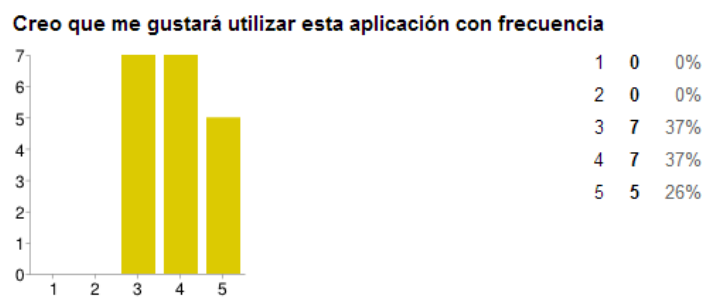


Figura 113: Diagrama de barras resultados "Creo que me gustaría utilizar esta aplicación con frecuencia"

Al analizar el Diagrama de barras anterior nos damos cuenta de que los valores 3 y 4 son los que más se repiten, seguido por el 5, por lo que podemos concluir que a los usuarios les gustará utilizar la aplicación.

10.2.2 Resultados de la pregunta “Encontré la aplicación innecesariamente compleja”

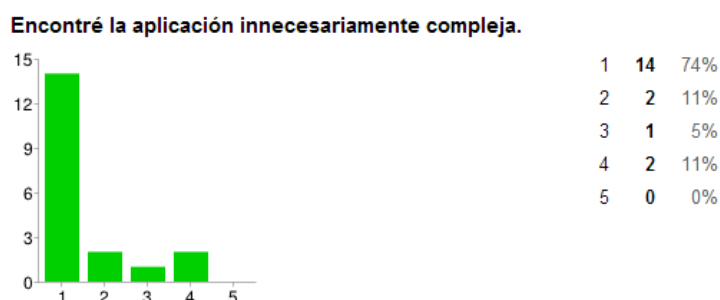


Figura 114: Diagrama de barras resultados “Encontré la aplicación innecesariamente compleja”

A la pregunta “Encontré la aplicación innecesariamente compleja”, la mayoría de los usuarios (74%) han contestado con una valoración negativa, es decir, que el uso de la aplicación les ha resultado sencillo.

10.2.3 Resultados de la pregunta “Pensé que la aplicación sería más fácil de usar”

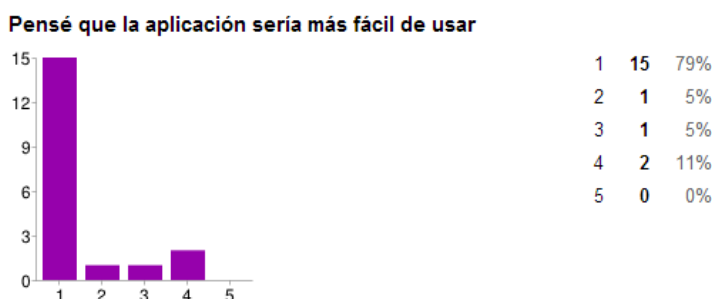


Figura 115: Diagrama de barras resultados “Pensé que la aplicación sería más fácil de usar”

El Diagrama de barras correspondiente a esta pregunta revela que la moda (el valor que se repite con más frecuencia) es el 1 con un 79%, por lo que podemos decir y corroborar apoyándonos en la pregunta anterior, que la aplicación posee un fácil uso.

10.2.4 Resultados de la pregunta “Creo que voy a necesitar el apoyo de un experto para utilizar la aplicación”

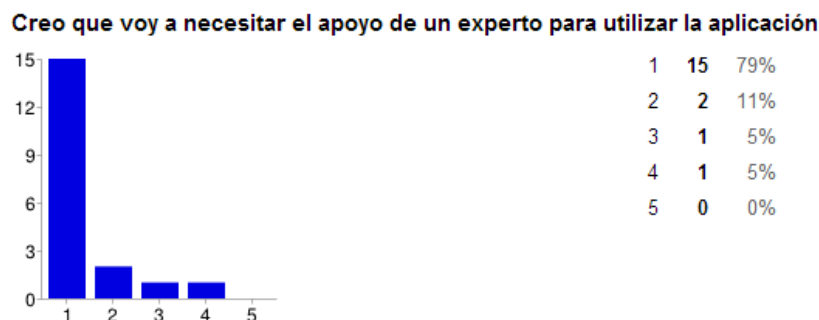


Figura 116: Diagrama de barras resultados "Creo que voy a necesitar el apoyo de un experto para utilizar la aplicación"

Al igual que ocurría en el caso anterior, la mayoría de usuarios (79%) están en desacuerdo con la pregunta, afirmando que no necesitan la ayuda de ningún usuario experto para utilizar English Pursuit.

10.2.5 Resultados de la pregunta “Me pareció que las diversas funciones de esta aplicación se integran bien”



Figura 117: Diagrama de barras resultados "Me pareció que las diversas funciones de esta aplicación se integran bien"

En los resultados de esta pregunta se produce una mayor controversia que en las anteriores, al haber sido más variados. Sin embargo, no hay ningún usuario que esté en desacuerdo y más de la mitad opinan que las funciones de la aplicación están bien integradas. En base a los resultados puede hacerse un estudio para integrar las distintas funciones de la manera más óptima.

10.2.6 Resultados de la pregunta “Pienso que hay muchos aspectos mejorables en esta aplicación”

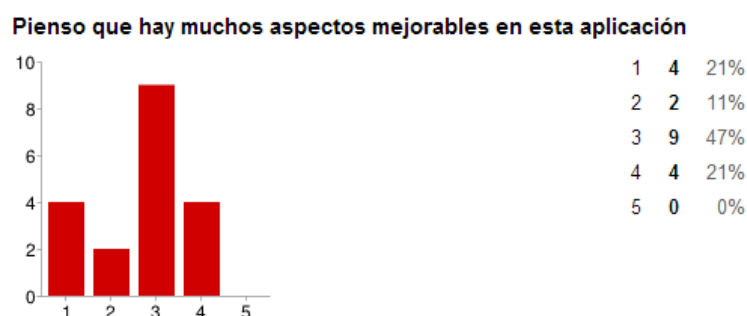


Figura 118: Diagrama de barras resultados "Pienso que hay muchos aspectos mejorables en esta aplicación"

En esta pregunta, como ocurría en la anterior, se han obtenido resultados muy dispares, este hecho, junto con el de que la respuesta que se repite con mayor frecuencia (47%) es la número tres, demuestra que los usuarios piensan que hay varios aspectos a mejorar en la aplicación. Por supuesto se es consciente de ello y muchas de estas mejoras se recogen en el apartado dedicado a ello (véase sección 11.2), donde se especifican varias de las mejoras en mente.

10.2.7 Resultados de la pregunta “La mayoría de la gente podrá utilizar esta aplicación sin problemas”

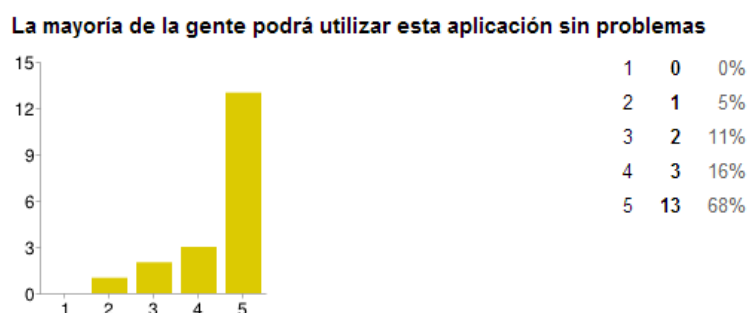


Figura 119: Diagrama de barras resultados " La mayoría de la gente podrá utilizar esta aplicación sin problemas"

El aspecto de este Diagrama de barras muestra los resultados de forma escalonada ascendente hacia resultados más positivos, con una clara tendencia (68%) al último valor, lo que demuestra que la mayor parte de usuarios están de acuerdo con que la gente podrá utilizar la aplicación sin problemas.

10.2.8 Resultados de la pregunta “Encontré la aplicación difícil de usar”

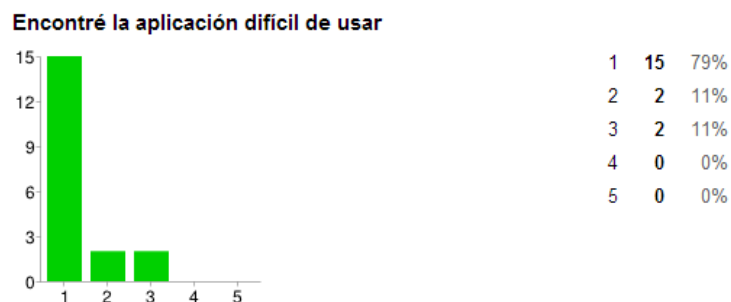


Figura 120: Diagrama de barras resultados "Encontré la aplicación difícil de usar"

Esta pregunta tiene una estrecha relación con la segunda (“**Encontré la aplicación innecesariamente compleja**”) y tercera (“**Pensé que la aplicación sería más fácil de usar**”) y casualmente demuestra el fácil uso de la aplicación con un porcentaje muy similar (79%).

10.2.9 Resultados de la pregunta “Me sentí seguro utilizando la aplicación”



Figura 121: Diagrama de barras resultados "Me sentí seguro utilizando la aplicación"

En esta pregunta, la mayoría de usuarios (63%) han manifestado su seguridad en cuanto al uso de la aplicación, sin embargo, hay varios que no lo tienen tan claro, pero representan una minoría.

10.2.10 Resultados de la pregunta “Necesito aprender muchas cosas antes de manejarme bien con la aplicación”

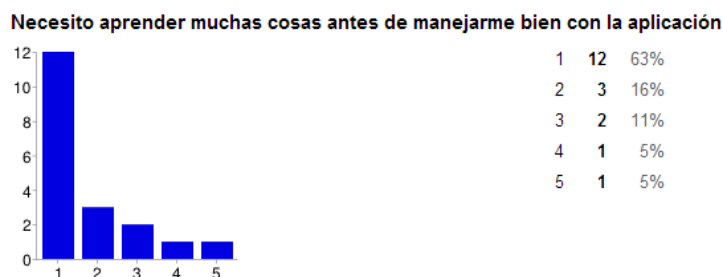


Figura 122: Diagrama de barras resultados "Necesito aprender muchas cosas antes de manejarme bien con la aplicación"

Este último Diagrama de barras representa un aspecto escalonado descendente, lo que demuestra que hay algunos usuarios que confiesan tener que aprender cosas antes de poder manejarse con soltura en la aplicación. Sin embargo la mayoría (63%) cree que no necesita aprender nada nuevo para obtener un buen manejo.

10.3 Comentarios y sugerencias

Los comentarios y sugerencias recogidos para mejorar la aplicación son los siguientes:

- “Aun hay que añadir más variedad de preguntas ya que de momento está un poco reducido, pero por lo demás: Una excelente aplicación para practicar inglés mientras te diviertes y compites con tus amigos o gente on-line.”
- “Mayor variedad de preguntas”
- “Un pequeño tutorial o guía de iniciación para el manejo de la aplicación, y una breve descripción de su funcionamiento y dificultad de contenido (nivel de idioma necesario)”
- “Bueno, hay que adaptar la imagen o tamaño de letra para todas las pantallas, porque en móviles con pantalla pequeña hay cosas que no se ven porque se superponen a otras y es imposible tocar el botón. También añadiría un poco más de tiempo, un minuto estaría bien, porque ha veces que no llegas a la quinta pregunta.”

CAPÍTULO 11. CONCLUSIONES Y PROPUESTAS

En este último capítulo se exponen las conclusiones extraídas después de todo el trabajo llevado a cabo para desarrollar la versión presente de la aplicación. Además, en el segundo apartado también se mencionan las posibles nuevas funcionalidades que pueden implementarse como trabajos futuros, para dotar al proyecto de más funciones y obtener una aplicación más completa.

11.1 CONCLUSIONES

Para terminar y una vez acabada tanto la redacción de este documento escrito como el desarrollo de la aplicación objetivo de este trabajo, se pueden extraer varias conclusiones interesantes que ponen de manifiesto el estudio llevado a cabo para alcanzar los objetivos propuestos al inicio y que sin duda ayudarán a construir una perspectiva más precisa a la hora de comenzar otro proyecto de complejidad similar.

El primer propósito que se ha perseguido ha sido conocer en profundidad el sistema operativo sobre el que se ejecuta la aplicación desarrollada. Para ello se ha realizado un análisis de su arquitectura interna, de los elementos que constituyen un proyecto y de los componentes que construyen las aplicaciones.

Debido a que el sistema implementado debe ofrecer una experiencia multijugador, donde varios usuarios compiten y juegan entre ellos para ver quien tiene mayor conocimiento sobre la lengua inglesa, ha sido necesario implementar una infraestructura de servidor centralizado con soporte al juego colaborativo y encargado de atender las peticiones de todos los usuarios, así como de almacenar la información de los mismos y la del juego en general. Por ello se ha llevado a cabo un estudio del lenguaje PHP, el cual ha

permitido realizar esta implementación y llevar a cabo la comunicación entre los clientes Android y el propio servidor (integrando ambos lenguajes de programación), además de la interacción con la BBDD, posibilitando la manipulación de la información según la necesidad.

Se ha estudiado el sistema de gestión de bases de datos MySQL, que en combinación con PHP permite llevar a cabo esa manipulación de la información explicada en el anterior párrafo.

Otro objetivo relevante perseguido y alcanzado durante este desarrollo ha sido la construcción de una interfaz siguiendo la filosofía de diseño centrado en el usuario, donde lo importante es crear un producto visual intuitivo que aporte una experiencia satisfactoria a los usuarios, logrando que éstos realicen las tareas con el mínimo esfuerzo posible. Para conseguir esto se han consultado fuentes de información variadas [11] [18].

Para hacer un juego más interactivo se ha agregado la funcionalidad del envío y recepción de notificaciones Push. Su implementación se ha llevado a cabo utilizando la API de un servicio de Google llamado Google Cloud Messaging.

También se han de destacar las técnicas de ingeniería del software utilizadas en el análisis y diseño previos a la implementación, las cuales, indudablemente, ayudan mucho a la fijación de los objetivos y requisitos que debe respetar el proyecto, lo que permite abordar las fases de desarrollo con unas metas específicas y planteadas previamente.

Para finalizar el ciclo vida RUP se han realizado una serie de pruebas que han permitido detectar algunos errores de implementación (leyendo el Log y utilizando el modo DEBUG), los cuales han sido solucionados correctamente.

Por último, se ha realizado un cuestionario SUS donde han participado varios usuarios valorando múltiples aspectos de la aplicación. De esta forma se han podido obtener una serie de resultados que permiten conocer el grado de satisfacción de los usuarios cuando utilizan la aplicación.

Una vez concluido el desarrollo de la presente versión de la aplicación se ha procedido a su inclusión y promoción en Google Play Store (para ello he adquirido una cuenta de desarrollador), donde una vez subida se pone a disposición de millones de usuarios, facilitando enormemente su difusión y aumentando sus probabilidades de éxito.

Desde mi punto de vista personal he de decir que me ha sido muy placentero el poder desarrollar un proyecto software de estas características ya que me ha permitido poner en práctica muchos de los conceptos adquiridos a lo largo de los diferentes cursos de este

grado. Es en estos momentos cuando uno realmente se da cuenta de la importancia de cada una de las experiencias vividas en la universidad y de las dificultades superadas a lo largo de la elaborada formación de la que se dispone en la UCLM.

11.2 TRABAJO FUTURO Y POSIBLES AMPLIACIONES

Para finalizar con este último capítulo, comentaremos algunas de las propuestas que se podrían añadir para continuar y mejorar el trabajo actual, dotándolo de mayor funcionalidad:

- Logros de juego

La primera funcionalidad adicional interesante que podría incluirse en la aplicación serían los típicos logros que los jugadores podrían ir consiguiendo a medida que juegan y van superando retos. Por ejemplo, cuando un usuario gana una cantidad de partidas, responde correctamente un número de preguntas o consigue una buena racha sin perder.

- Preferencias de sonido y privacidad

Al igual que se han implementado las preferencias referentes a las notificaciones push que reciben los usuarios, también podrían implementarse ajustes de sonido (permitiendo deshabilitar el mismo en una partida) y de privacidad, para dejar a elección del usuario si desea o no aparecer entre los usuarios públicos

- Nivel de experiencia de usuario

Una de las funcionalidades que más aportaría a la experiencia y entretenimiento de “English Pursuit” sería incluir niveles de experiencia de usuarios, para que éstos pudieran ascender según sus logros y de alguna manera, presumir de tener mayor conocimiento que otros jugadores.

Además, si lo deciden podrían retarse con usuarios de su mismo nivel o similar.

- Chat in-game

Como en otros juegos similares, sería interesante incluir algún medio de comunicación entre los jugadores participantes de una partida para que así puedan intercambiar opiniones y experiencias de juego.

- Caducidad de partidas

Con el fin de evitar trampas, cuando un usuario disponga del turno tendrá un tiempo límite para jugar, ya que si esto no se controla y un usuario va perdiendo pero quiere evitar que se le adjudique la derrota, podría no jugar nunca más la partida para no terminarla.

Por último, decir que al igual que ocurre en todo desarrollo software actual, “English Pursuit” podrá ir evolucionado, incluyendo nuevas características y mejorando su eficiencia en futuras actualizaciones.

BIBLIOGRAFÍA

- [1] «Un 41,5% de la población española de 15 años o más dispone de 'smartphone', según el Ontsi,» 14 11 2013. [En línea]. Available: <http://www.eleconomista.es/interstitial/volver/nuezene14/tecnologia/noticias/5311577/11/13/Un-415-de-la-poblacion-espanola-de-15-anos-o-mas-dispone-de-smartphone-segun-el-Ontsi.html>. [Último acceso: 05 04 2014].
- [2] «Android alcanza un 92,1% de las ventas de smartphones en España en el último trimestre,» 18 4 2013. [En línea]. Available: <http://www.xatakamovil.com/sistemas-operativos/Android-alcanza-un-92-1-de-las-ventas-de-smartphones-en-espana-en-el-ultimo-trimestre>. [Último acceso: 07 04 2014].
- [3] «Google Play Apps,» [En línea]. Available: http://play.google.com/intl/ALL_es/about/apps/. [Último acceso: 02 03 2014].
- [4] «Developer Economics,» [En línea]. Available: <http://www.developereconomics.com/reports/q1-2014/>. [Último acceso: 04 06 2014].
- [5] J. T. Gironés, El gran libro de Android, Barcelona: Marcombo, 2013.
- [6] «Android (operating system),» [En línea]. Available: http://en.wikipedia.org/wiki/Android_%28operating_system%29#Linux_kernel. [Último acceso: 07 05 2014].
- [7] «Oracle,» [En línea]. Available: <http://www.oracle.com/technetwork/es/java/javasebusiness/downloads/index.html>. [Último acceso: 09 02 2014].
- [8] «Android Developer,» [En línea]. Available: <http://developer.Android.com/sdk/installing/studio.html>. [Último acceso: 18 03 2014].
- [9] D. A. Norman, Usability of Everyday things: Revised and Expanded Edition, New York: Basic Books, 2013.
- [10] «Android Developer,» [En línea]. Available: <http://developer.Android.com/design/index.html>. [Último acceso: 26 03 2014].
- [11] «Android Developer,» [En línea]. Available: <http://developer.Android.com/design/get-started/principles.html>. [Último acceso: 26 05 2014].
- [12] «Android Developer,» [En línea]. Available:

<https://developer.Android.com/design/patterns/navigation-drawer.html>.

[Último acceso: 28 05 2014].

- [13] «Flaticon,» [En línea]. Available: <http://www.flaticon.com/>. [Último acceso: 14 01 2014].
- [14] «No-IP,» [En línea]. Available: <http://www.noip.com/>. [Último acceso: 19 03 2014].
- [15] «1and1,» [En línea]. Available: <https://www.1and1.es/>. [Último acceso: 19 03 2014].
- [16] «Android Developer,» [En línea]. Available: <http://developer.Android.com/training/implementing-navigation/navigation-drawer.html>. [Último acceso: 04 05 2014].
- [17] «Android Developer,» [En línea]. Available: <http://developer.Android.com/guide/topics/ui/settings.html>. [Último acceso: 11 05 2014].
- [18] «Android Patterns,» [En línea]. Available: <http://www.Androidpatterns.com/>. [Último acceso: 21 03 2014].
- [19] «Wikipedia,» [En línea]. Available: http://es.wikipedia.org/wiki/Sistema_operativo. [Último acceso: 05 06 2014].
- [20] «Wikipedia,» [En línea]. Available: <http://es.wikipedia.org/wiki/BBDD>. [Último acceso: 05 06 2014].
- [21] «PHP Development,» [En línea]. Available: <https://php.net/>. [Último acceso: 14 03 2014].

GLOSARIO DE TÉRMINOS

S.O.

Siglas de Sistema Operativo. Se trata de un programa o conjunto de programas que en un sistema informático gestiona los recursos de hardware y provee servicios a los programas de aplicación [19].

BBDD

Siglas de Base de datos. Es el conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso [20].

PHP

Acrónimo de “Hypertext Pre-processor”. Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. En este caso se ha utilizado para la implementación de funcionalidades en el lado del servidor, el cual atenderá a las peticiones de los clientes Android [21].

API

Acrónimo de “Application Programming Interface”. Es el conjunto de funciones y procedimientos que ofrece una biblioteca para utilizarse en otro software.

RUP

Acrónimo de “Rational Unified Process”. Se trata de un proceso de desarrollo de software que constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de software.

Hosting

El “Hosting” o alojamiento web es un servicio en internet que permite a los usuarios almacenar archivos e información, en servidores remotos, para disponer de ella en cualquier momento y lugar.

CONTENIDO DEL CD

En el contenido del CD que acompaña a la memoria podemos encontrar los siguientes recursos:

- Memoria del trabajo en los formatos PDF, DOCX y DOC dentro del directorio Memoria.
- Código fuente del trabajo dentro del directorio Código fuente.
- Diagramas de diseño dentro del directorio Diagramas.

ANEXO A. MANUAL DE INSTALACIÓN EN ANDROID

A.1 INSTALACIÓN DESDE GOOGLE PLAY



La mayoría de dispositivos Android llevan instalada de fábrica una versión de la aplicación Google Play, la tienda oficial de aplicaciones Android de Google. Esta aplicación permite descargar e instalar un millar de aplicaciones para este sistema operativo de una forma sencilla y cómoda.

Además el usuario puede instalar directamente la aplicación que desee en su dispositivo Android bien buscándola en la aplicación Google Play o bien en la web oficial de este servicio.

El caso de “English Pursuit” no es diferente, pudiendo proceder a su descarga mediante la aplicación de Google Play en Android, como puede apreciarse en la siguiente ilustración:

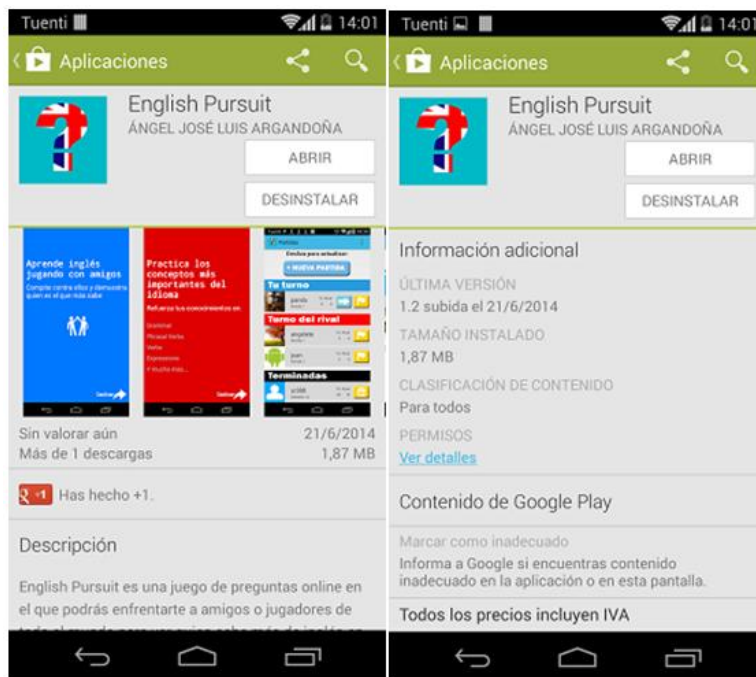


Figura 123: English Pursuit en Google Play

Y también desde la web de “Google Play” accediendo a la aplicación en cuestión:

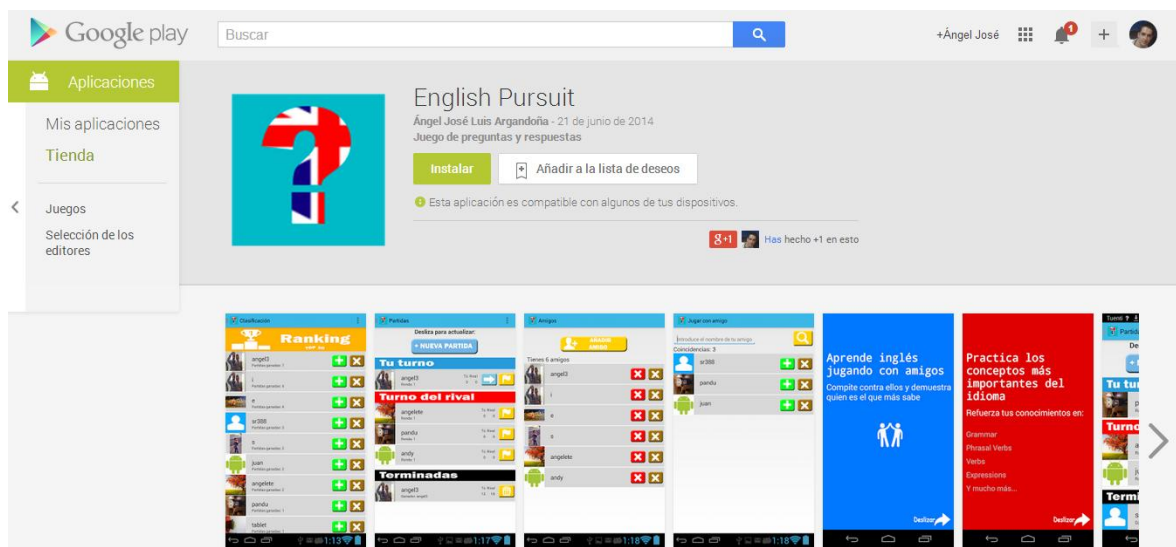


Figura 124: English Pursuit en Google Play (web)

A.2 INSTALACIÓN MEDIANTE APK (Application PacKage File)

Un APK es un archivo que constituye un paquete que se utiliza para distribuir e instalar aplicaciones y componentes para el sistema operativo Android.

Puede obtenerse directamente del directorio /bin de nuestro proyecto una vez que es compilado y este paquete contiene todo lo necesario para poder instalar y ejecutar la aplicación (clases, recursos, AndroidManifest.xml, etc).

Este modo de instalación es una alternativa a lo ofrecido por Google Play para quien no tenga acceso a la tienda de Google o tenga algún tipo de problema con ella.

De esta forma, al disponer del archivo .apk en nuestro smarphone o tablet Android sólo hay que seleccionarlo para ejecutar el instalador y esperar a que el proceso termine.

ANEXO B. MANUAL DE USO

B.1 INTRODUCCIÓN

El uso de esta aplicación resulta muy sencillo para cualquier usuario ya que es similar a la mayoría de aplicaciones de la misma temática, sin embargo, es conveniente dictar algunas pautas y pasos a seguir para que todo usuario pueda adaptarse con facilidad al conjunto de opciones que ofrece.

B.2 CREAR PARTIDA

1. El primer paso a seguir consiste en acceder a la pantalla principal de la aplicación donde el usuario puede ver y acceder tanto a las partidas en curso como terminadas.
2. El siguiente paso es seleccionar el botón “Nueva partida”.
3. Ahora llega el momento de seleccionar el rival con el que jugar: amigo o aleatorio. Si queremos jugar con un contrincante al azar habrá que seleccionar aleatorio y si queremos jugar contra un amigo se elegirá el amigo deseado en la pantalla “Amigos”. Ya está todo listo, la partida se habrá creado y sólo habrá que esperar a que el otro jugador conteste su ronda para que el turno sea nuestro.



B.3 JUGAR PARTIDA

Para poder jugar una partida hay que tener en cuenta que el turno de la misma debe pertenecernos.

1. Para acceder a una partida, el usuario tendrá que entrar en la pantalla principal donde se muestran todas sus partidas disponibles.
2. El siguiente paso es seleccionar el botón "Jugar" (representado por una flecha) de la partida deseada.
3. Para contestar las preguntas pertenecientes a una ronda de la partida, el jugador tendrá que pulsar sobre la respuesta que le parezca correcta. Si la selección es correcta, se añadirá un punto positivo a la puntuación del usuario, en caso contrario será el punto negativo el que suba al marcador.
4. Cuando el jugador conteste las cinco preguntas de una ronda (o bien se acabe el tiempo) el turno pasará al jugador rival y la puntuación será actualizada.



B.4 VER CLASIFICACIÓN

La posibilidad de ver la clasificación, donde se encuentran los mejores jugadores de English Pursuit aporta un grado más de competitividad al dar a los usuarios la oportunidad de aparecer en las posiciones más altas del ranking.

Acceder a este apartado es muy sencillo ya que el usuario sólo tiene que seleccionar la opción pertinente del menú lateral.

