BARLIKE

Enrique Pérez Pisonero
ETIS

Marc Domingo Prieto
Jordi Almirall López

9 de Enero de 2015

B) GNU Free Documentation License (GNU FDL)

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	BARLIKE
Nombre del autor:	Enrique Pérez Pisonero
Nombre del consultor:	Marc Domingo Prieto / Jordi Almirall López
Fecha de entrega (mm/aaaa):	01/2015
Área del Trabajo Final:	Desarrollo de Aplicaciones para dispositivos Android
Titulación:	ETIS

Resumen del Trabajo (máximo 250 palabras):

El objetivo de este proyecto es el desarrollo de la aplicación android BARLIKE. La aplicación pretende ser un espacio social para compartir opiniones y datos sobre restaurantes y bares de tapas. El proyecto se desarrolla bajo la licencia de reconocimiento- no comercial- sin obraderivada para ser publicada en caso de que proceda en el mercado de android. En el presente documento se establece el plan de trabajo para el desarrollo del mismo.

Abstract (in English, 250 words or less):

The objective of this project is the development of barlike android application. The application aims to be a social space to share opinions and information about restaurants and tapas bars. The project is developed under license from attribution- non commercial - No derivate Works to be published if appropriate in the android market. In this document is described how the protect is plan.

Palabras clave (entre 4 y 8):

Restaurante, tapas, aperitivos

Indice

1. Introducción..... 3
    1.1. Contexto y justificación del Trabajo..... 3
    1.2. Objetivos del Trabajo..... 3
    1.3. Enfoque y método seguido..... 4
    1.4. Planificación del Trabajo..... 4
    1.5. Breve sumario de productos obtenidos..... 6
    1.6. Breve descripción de los otros capítulos de la memoria..... 7
2. Diseño Técnico..... 7
    2.1. Base de datos..... 7
    2.2. Software de Aplicación..... 9
    2.3. Arquitectura del sistema..... 10
    2.4. Análisis..... 11
3. Diseño..... 14
4. Desarrollo..... 16
    4.1. Herramientas..... 16
    4.2. Análisis del estado del Proyecto..... 17
5. Pruebas..... 19
6. Consideraciones..... 21
    6.1. Valoración económica..... 21
7. Conclusiones..... 22
    7.1. Conclusiones sobre el trabajo..... 22
    7.2. Reflexión crítica sobre logro de objetivos..... 23
    7.3. Líneas de trabajo futuro..... 23
    7.4. Análisis crítico sobre seguimiento de la planificación..... 23
8. Glosario..... 24
9. Bibliografía..... 25
10. Anexos..... 25
    10.1. Manual de instalación..... 25

1. Introducción

1.1. Contexto y justificación del Trabajo

El proyecto pretende ser una base de datos de conocimiento de lugares de restauración basado en las experiencias de los usuarios que permita compartir con otros su ubicación y datos de contacto para que puedan tener conocimiento de los mismos. La valoración de antiguos clientes permite ser una ayuda para los nuevos clientes potenciales a la hora de buscar lugares adecuados a sus intereses. En el mercado ya existen aplicaciones de tal cometido pero más complejas. El objetivo es la simplificación de su uso y un aporte más claro ayudándose de aspectos como la geolocalización.

1.2. Objetivos del Trabajo

La aplicación inicialmente se enfoca para dispositivos móviles android que ejecutan como versión mínima de software la 4.0, correspondiente a la *API 14*. Posteriormente como requisito avanzado el enfoque del desarrollo futuro se pretende ampliar para adaptarse a la utilización de tabletas y otros dispositivos android con requisitos de resolución gráfica mayores. A continuación se detallan las funcionalidades especificadas para la aplicación entre las que se categorizan las que se establecen como *inicial* o *avanzada*:

Iniciales

- Alta y edición de fichas con información de restaurante.
- Búsqueda y listado de los restaurantes registrados.
- Establecer puntuación sobre un restaurante visualizado.
- Escribir un comentario y/o sugerencia para un restaurante.
- El restaurante muestra el indice de popularidad en función de la puntuación de los usuarios.

Avanzadas

- Utilización de la funcionalidad de mapas para localizar restaurantes cercanos.
- Edición y visualización de un restaurante mediante uso de geolocalización.
- Uso de control de acceso para gestión a través de usuarios.
- Funciones de filtrado y de búsqueda avanzadas.
- Compartir fichas de restaurantes a través de medios sociales y correo electrónico.
- Añadir recomendación de producto del restaurante rápida mediante foto.

En el siguiente listado se describen los requisitos funcionales y no funcionales:
- Requisitos *Funcionales*

- El usuario quiere poder crear una nueva ficha de un restaurante.
- Que sea posible modificar la información de una ficha existente.

- Los restaurantes pueden obtener una puntuación del usuario del uno al cinco.
- La aplicación me permite realizar búsquedas de restaurantes por el nombre o sino listar todas.
- La localización permite encontrar un restaurante.
- Que se pueda compartir la ficha de un restaurante concreto con otros usuarios o cualquier contacto.
- En cualquier momento se puede realizar una foto y asignarla a un restaurante.

- Requisitos *No funcionales*

- El sistema debe cumplir las disposiciones recogidas en la ley orgánica de datos personales y en el reglamento de medidas de seguridad.
- El sistema debe ser atractivo y sencillo de utilizar.
- La ficha de cada restaurante no debe tener más de 4 fotos para el restaurante.
- Debe poder navegarse fácilmente entre pestañas.
- El sistema debe permitir trabajar localmente con la ficha en caso de que no hay conexión a internet.

1.3. Enfoque y método seguido

El proyecto pretende ser desarrollado como "nuevo" basándose en aplicaciones ya existentes pero con el objetivo de centrarse en la simplificación de uso de la herramienta sin funcionalidades extra que la conviertan en muy pesada y poco útil. Así mismo, se pretende seguir una estrategia centrada más en el diseño y menos en el desarrollo, por lo que las tareas de programación más complejas tendrán un apoyo a través de servicios externos para el almacenamiento de la base de datos.

Para ello, bajo recomendación, se ha decidido utilizar los servicios de ***cloud*** ofrecidos por el portal parse.com *[7]*. Principalmente nos permite en el caso de este proyecto concretamente liberar la carga del trabajo en cuanto a lo que entendemos por el ***backend*** necesario para su desarrollo. El servicio incluye toda la gestión a través de librerías, así como objetos y hosting remoto de los datos, por lo que conlleva una simplificación real del proyecto. De esta manera el ahorro de recursos y tiempo para el programador es elevado. Por consiguiente es una buena alternativa para alcanzar los objetivos.

1.4. Planificación del Trabajo

Recursos

A continuación se detalla el listado de recursos a utilizar para el proyecto:

Desarrollo:
- Entorno de desarrollo: Android Studio 1.0.2 (Beta)
- Sistema Operativo: MAC OS X 10.9.4
- Dispositivo: Telefono GETEK-V9. Versión Android 4.2.1 (API 17)

Diseño:

- Adobe PhotoShop CS6
- ImageOptim *[6]*

Implementación:

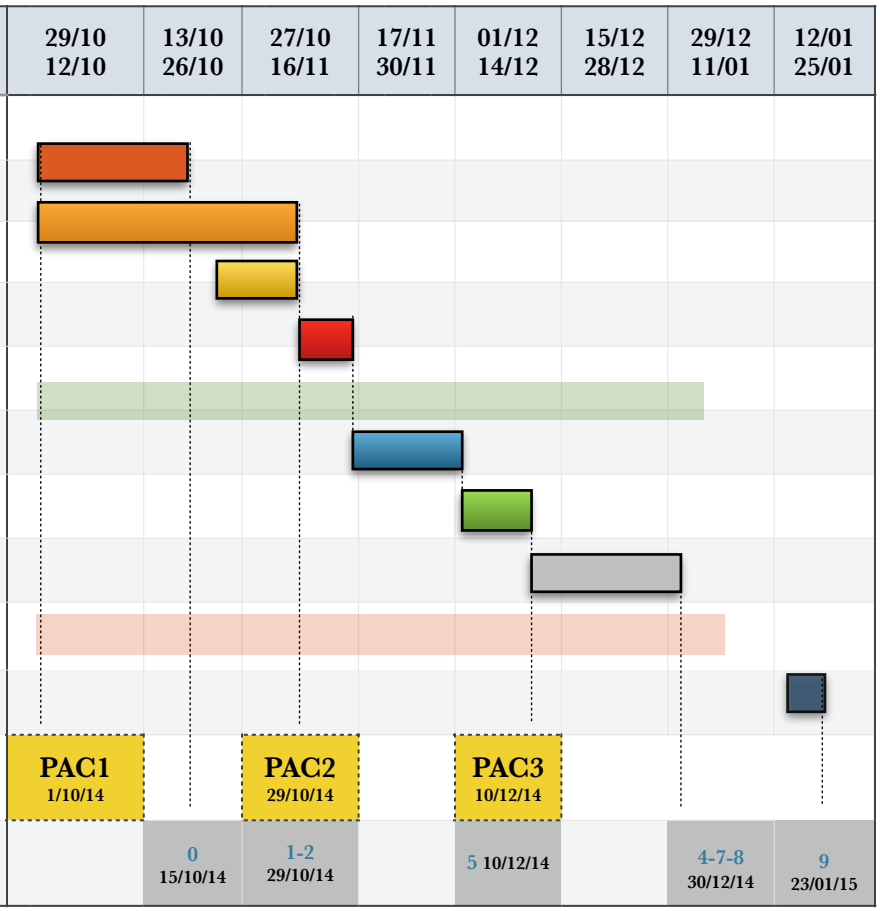
- Base de datos : Servicios Backend a través de parse.com *[7]*. Uso de servicios a través de google para claves de localización de google maps.

Planificación

Se consideran para la temporalización del proyecto las fechas límite de entrega de las pruebas de evaluación. Se añaden nuevos hitos intermedios para establecer una división y mayor control en el cumplimiento de los objetivos. De esta forma se modularizan las funcionalidades favoreciendo ajustar el desarrollo a los tiempos.

A continuación se muestra un gráfico de *Gantt* con la planificación y posteriormente se describe con algo más de detalle las fases planteadas:

Fases	Fecha inicial	Fecha final
Fase 0	01/10/14	15/10/14
Fase 1	01/10/14	29/10/14
Fase 2	15/10/14	29/10/14
Fase 3	29/10/14	15/11/14
Fase 4	01/10/14	08/01/15
Fase 5	15/11/14	04/12/14
Fase 6	04/12/14	10/12/14
Fase 7	10/12/14	30/12/14
Fase 8	-	09/01/15
Fase 9	-	23/01/15
	Pacs	
	Fases	



Descripción de las fases del proyecto:

- ***Fase 0*****:** Se realiza la fase de diseño, configuración y conexión de la base de datos a utilizar, comenzando con las fichas y los datos que manejarán. Permitirá la creación de nuevos restaurantes.

- ***Fase 1****:* Finalizada esta fase se determina la linea de diseño y flujo de la aplicación.

- ***Fase 2*****:** Esta fase se dedica a la implementación de las búsquedas y manejo de los datos así como a su representación.

- ***Fase 3*****:** Implementamos el sistema de comentarios y de puntuación de los restaurantes.

- ***Fase 4*****:** Esta fase estará presente en todo el desarrollo del proyecto. Incluye la modificación y adaptación del diseño de la aplicación y su uso así como las pruebas que se irán llevando a cabo con cada nueva funcionalidad adquirida.

- ***Fase 5***: Como objetivo final se pretende finalizar las tareas iniciales de alta, búsqueda, listado, y puntuación de los restaurantes. De forma adicional se pretende dedicar la fase a la búsqueda e implementación si procede de alternativas a las utilizadas y desarrolladas hasta el momento. Incluye mejora, modificación y adición de nuevas funciones para la aplicación.

- ***Fase 6*****:** En esta fase se darán por concluidas las funcionalidades “iniciales”, estableciendo como final la referente al sistema de comentarios de los restaurantes. Acto seguido, si el recurso temporal lo permite, se hará hincapié en el apartado de la programación defensiva, realizando depuración y ordenación del código fuente.

- ***Fase 7*****:** Desde este punto, el desarrollo se centra en todas aquellas funcionalidades avanzadas especificadas en el proyecto.

- ***Fase 8*****:** Fase presente en todo el desarrollo del proyecto. Se centra en la redacción de los documentos descriptivos (Memoria, Presentación y Producto final).

- ***Fase 9***: Fase de evaluación y defensa del proyecto.

1.5. Breve sumario de productos obtenidos

Como resultado se obtiene la aplicación desarrollada para Android en formato ***apk*** sin firmar digitalmente junto con todo el código fuente y a su vez los diseños realizados.

Se entregará además la memoria completa del proyecto junto con una presentación que resume los conceptos más relevantes del producto incluyendo una demo en video.

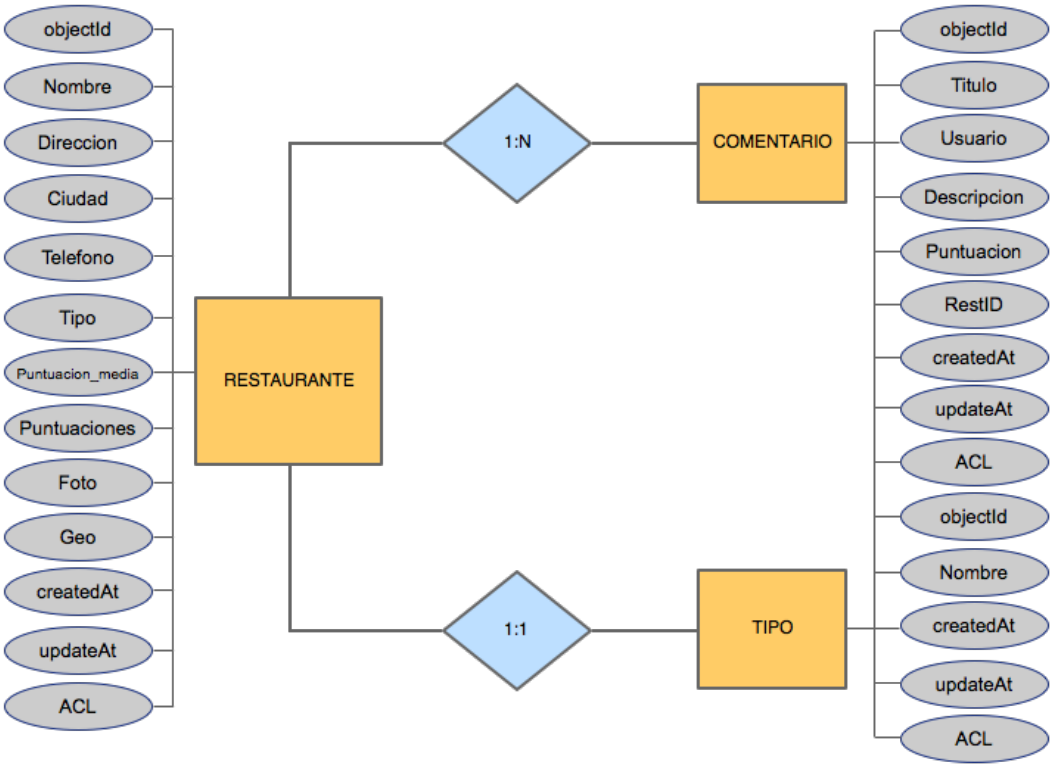
1.6. Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos se tienen en cuenta los aspectos relacionados a conclusiones finales en el desarrollo del proyecto. También se añadirán todas aquellas cuestiones relacionadas con dificultades añadidas en el desarrollo, así como cambios en la planificación y uso de otras herramientas y estrategias para el desarrollo.

2. Diseño Técnico

2.1. Base de Datos

A continuación se muestra el diagrama Entidad-Relación de la aplicación en el que podemos observar los atributos representados como elipses en color gris, las entidades, los cuadrados de color amarillo y por último las relaciones representadas por rombos de color celeste.



Entidades y Relaciones

Por lo cual tenemos tres entidades; Restaurante, Comentario y Tipo. Estos son los equivalentes a los "***classnames***" que se representan en el servicio parse y que no son más que tablas que contienen "***fields***", los cuales son los atributos en este tipo de diagrama.

En cuanto a las relaciones se establece lo siguiente:

- *“Un restaurante puede poseer ninguno, uno o más comentarios”*
- *“Un restaurante puede disponer de un solo tipo”*

Atributos

En la siguiente tabla se visualizan todos los atributos y su tipo de datos de cada una de las entidades:

Restaurante	**Comentario**	**Tipo**
objectId (String) Clave primaria)	***objectId*** (String) (Clave primaria)	***objectId*** (String) (Clave primaria)
Nombre (String)	Titulo (String)	Nombre (String)
Direccion (String)	Usuario (String)	createdAt (Date)
Ciudad (String)	Descripcion (String)	updatedAt (Date)
Telefono (String)	Puntuacion (Float)	ACL (ACL)
Tipo (String)	***RestID*** (Pointer) (Clave Foranea)	
Puntuacion_media (Float)	createdAt (Date)	
Puntuaciones (Number)	updatedAt (Date)	
Foto (ParseFile)	ACL (ACL)	
Geo (Geopoint)		
createdAt (Date)		
updatedAt (Date)		
ACL (ACL)		

Los atributos en naranja son de tipo ***String*** y representan los identificadores únicos de cada objeto o registro representado en la ***classname*** o entidad.

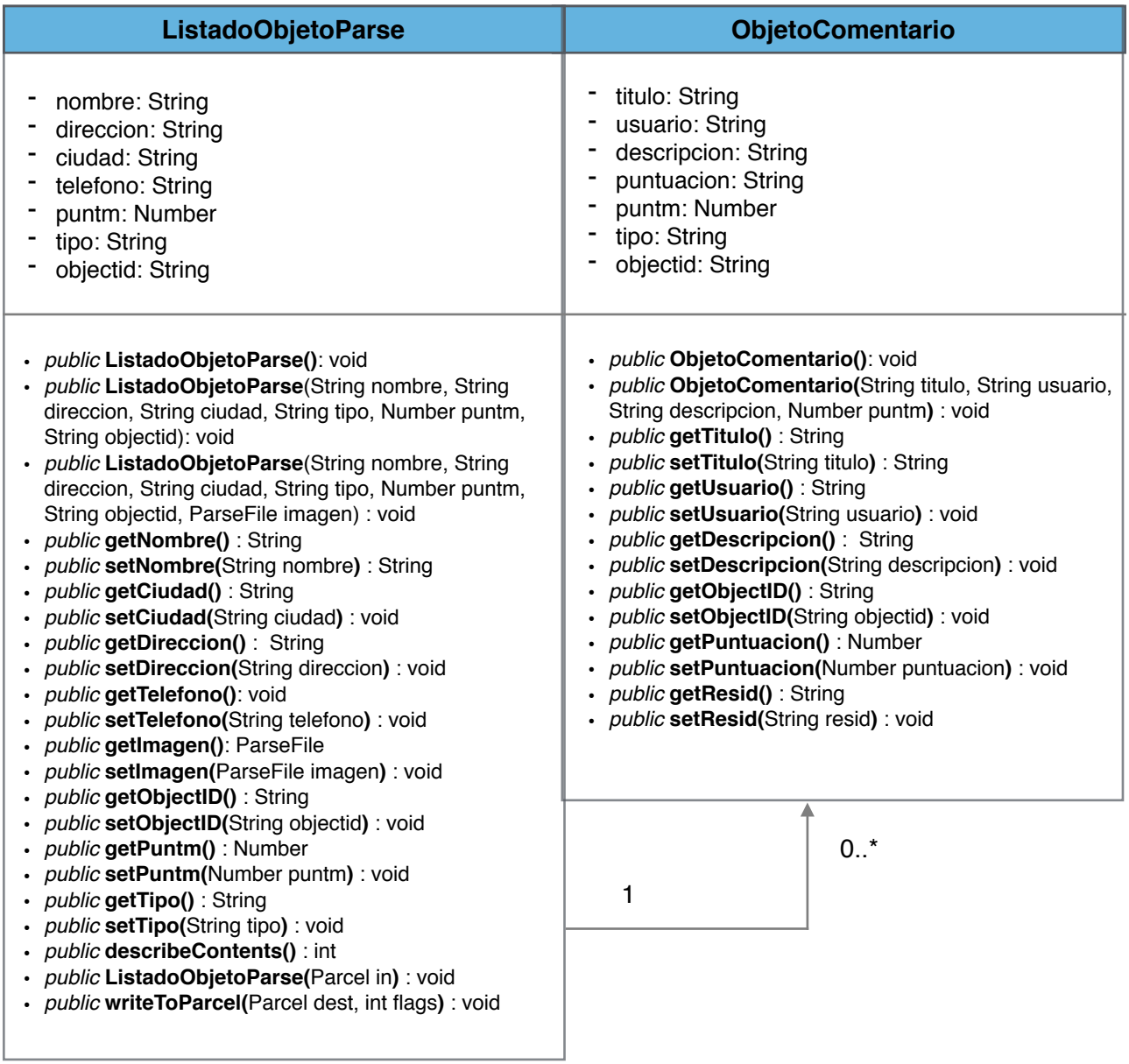
El atributo *RestID* es de tipo puntero. Es un tipo definido por el servicio *parse*[7] que permite hacer referencia a un registro concreto, en nuestro caso a un Restaurante. Todos los comentarios que pertenezcan a un restaurante concreto a través de este campo se hará referencia al identificador único u ***objectId*** de ese restaurante.

Los campos ***createAt*** y ***updateAt*** son creados automáticamente por *parse* al añadir un registro. Almacenan la fecha de la creación del registro y de actualización cada vez que se modifique el registro.

Por último, el campo ***ACL*** almacena un tipo de dato de *parse[7]* que consiste en el usuario. Este campo se utiliza para establecer permisos sobre el registro. Si un usuario es especificado en el campo, entonces sólo ese usuario tendrá permisos de lectura y escritura sobre el registro. Por lo contrario, si no se indica usuario, cualquiera todos poseerán permisos.

2.2. Software de Aplicación

En la siguiente imagen se muestra la relación de las clases más significativas del proyecto, las cuales son las directamente relacionadas con la estructura de datos y que establecen la conexión para manejar estos datos entre p*arse[7]* y la aplicación.



Clases

Se muestra la clase **ListadoObjetoParse** y la clase **ObjetoComentario**. La primera representa un restaurante con todos sus datos.

Atributos de Clase

Se puede observar como se indica en este modelo de *Diagrama de Clases* los atributos en la parte superior, en los que se especifica el tipo de datos de cada uno.

Métodos

En el cuadro inferior se representan todos los métodos de cada clase indicando su *ámbito, nombre de la clase*, *parámetros de entrada* si los tiene y finalmente se indica si se devuelve un tipo al finalizar el método o no devuelve nada (*void*).

Métodos Constructores

Se han definido por cada clase un método constructor vacío y uno o dos más según sea Comentario o Restaurante. La explicación es que existe la posibilidad de que se cree un nuevo restaurante que inicialmente no contenga una foto.

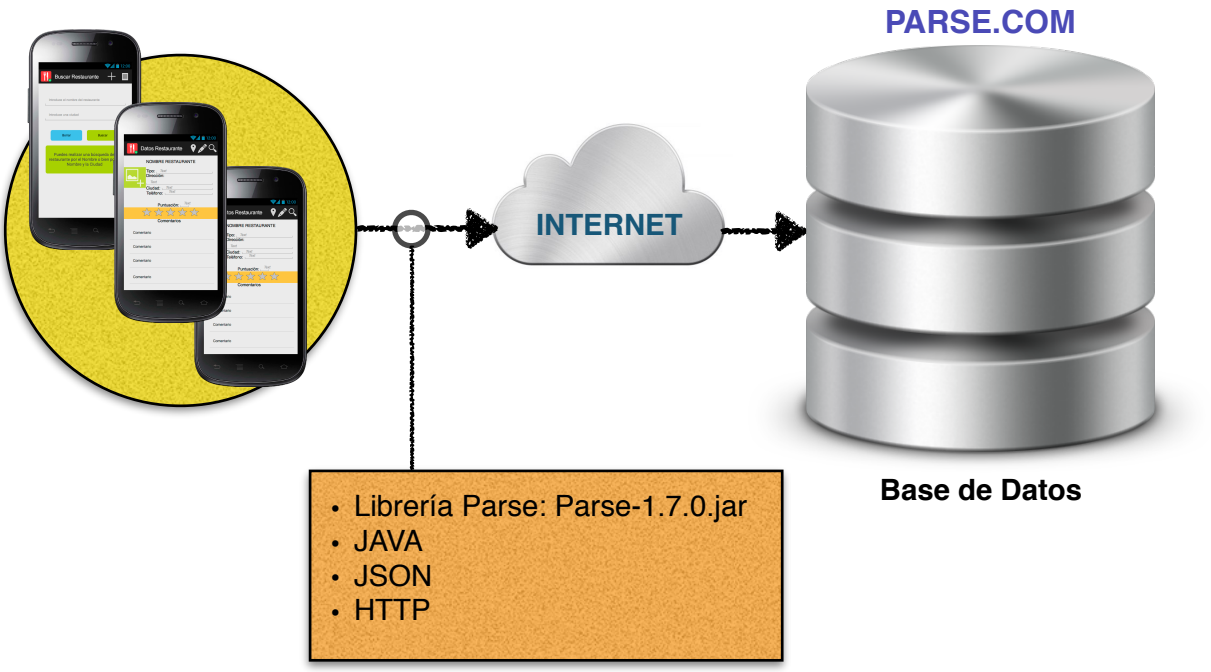
Relaciones

Observamos también la relación establecida entre ambas clases. Para cada restaurante puede existir ninguno, uno o más comentarios.

Una cosa más a añadir es referente al atributo "***creator***" de tipo Parcelable, así como los métodos ***describeContents***, ***writeToParcel*** y por ultimo el constructor de la clase ***ListadoObjetoParse***. Se trata de una implementación que no se ha llegado a implementar finalmente en este proyecto, que utiliza la clase Parcelable y que se utiliza para poder pasar objetos entre las actividades. Ya que los objetos definidos por *parse*[7] no son parcelables, es necesario definir los métodos que permitan hacerlo.

2.3. Arquitectura del Sistema

Presentamos en el siguiente diagrama, la estructura del proyecto. Representa el flujo de uso de la aplicación y las tecnologías utilizadas.



Los usuarios a través de su dispositivo móvil harán uso de la aplicación para acceder a la base de datos de *parse*[7]. La conexión a través de internet se realizará por conexión wifi inalámbrica o por conexión de 3G/4G de su operador de telefonía. La aplicación utiliza una serie de librerías proporcionadas por *parse ya empaquetadas (**Parse-1.7.0.jar**)* que constituyen parte del Backend para la conexión y manejo de datos del servicio. Internamente parse utiliza también el lenguaje ***JSON*** para el intercambio de datos entre el frontend de la aplicación y el backend en la nube así como el protocolo ***HTTP*** para el acceso a datos alojados en su web.

2.4. Análisis del Sistema

A continuación se presenta la fase de análisis como parte del diseño centrado en el usuario en donde se recoje información sobre los usuarios y los contextos de uso.

1. **Métodos de indagación**: En primer lugar se plantean los métodos de indagación para la fase de análisis. A continuación se seleccionan y se describen los que se utilizan para este proyecto.

1. *Observación*: Mediante técnica de observación (shadowing) e investigación contextual se recopila información sobre el tipo de usuarios y uso que se puede realizar. El perfil estándar es cualquier persona con necesidades alimenticias y con disponibilidad monetaria que decide desayunar,comer, cenar o tomar cualquier aperitivo en los horarios de las comidas o a lo largo del día en algún local de restauración cercano a su residencia. El usuario hace uso de la aplicación para cubrir la necesidad realizando búsquedas de forma local o en otras localidades

para su desplazamiento. Este método es considerado básico, pues el mismo desarrollador o implicado en el proyecto es un cliente que hace uso de los servicios y aporta su experiencia.

2. *Dinámicas de grupo*: Este tipo de método se aplica a grupo de personas del entorno con los que se comparten ideas y experiencias de lugares de restauración. Las ideas que aportan permiten obtener nuevas funcionalidades de diseño e implementación para este proyecto. Socialmente este método es muy común aplicarlo de las experiencias del circulo cercano a los implicados al proyecto.

3. *Análisis Competitivo o Benchmarking*: Con este método se realiza una comparativa de proyecto con otros ya existentes como son “Yelp” o “TripAdvisor” de los que sirve de referente y quiere tratarse de simplificar más el proyecto. Con el mercado actual es imprescindible tener en cuenta este método. Analizar a los competidores es una gran ayuda para definir mejor los objetivos y tareas del producto así como puede dar una visión mas general y aportar ideas de mejora.

2. **Usuarios y Contextos de Uso:** Especificamos los diferentes perfiles de usuarios que pueden hacer uso de la aplicación.

1. ***Usuario A***: Se considera perfil estándar que hará uso de la aplicación. Debido al margen elevado existente entre las características se pueden englobar todos los perfiles en uno solo indicando las pequeñas diferencias que se perciben.

- **Características**

  - Demográficas: El ámbito de la aplicación es de territorio nacional, por lo que demográficamente el perfil del usuario puede ser l*ocal*, *nacional* pero también puede ser utilizado por un perfil *extranjero* fuera del alcance para el cual se ha diseñado.

  - Intereses: Estos a nivel del objetivo de la aplicación pueden ser comparados con los tipos de restaurantes existentes, los cuales se presentan como diferentes categorías en la aplicación. Pueden ser de forma inicial: *“Bar de Tapas”, “Mesón”, “Restaurante”, “Tasca”, “Cervecería”, “Cafetería”, “Japonés”, “Hindú”, “Italiano”, “Chino”, “Tailandés”.*

  - Experiencia: El perfil requiere de una experiencia mínima inherente al uso del dispositivo móvil, así como al manejo de la tienda Android para gestionar las apps, pero ninguna que presente complicación para el uso de la aplicación a desarrollar.

- **Contexto de Uso:** Se responde a las siguientes preguntas

- *¿Donde?*: Teniendo en cuenta que hablamos de dispositivos móviles y/o tabletas la localización del “donde” no tiene restricción. No existe una limitación y podrá ser en cualquier lugar.

- ¿Cuando?: Los usuarios hacen uso de la aplicación en cualquier momento del día. El porcentaje será mas elevado en momentos cercanos a las horas de las comidas, cuando su intención sea de consumir fuera del domicilio. El periodo de uso de la aplicación una vez utilizada puede ser muy relativo. En cuanto al tiempo de uso de la aplicación será muy corto por defecto. El usuario una vez encontrado los datos que necesita puede no volver a utilizarla hasta que deba buscar otro local.

- ¿En que entorno?: Se deberá disponer de una conexión de datos, ya sea a través del operador móvil o por Wifi. En cuanto a otras condiciones, no procede. Solo las básicas como condiciones que cumplan el uso del dispositivo.

- **Análisis de tareas:** Las tareas más básicas que realizará el usuario son la consulta de un local conociendo el nombre, el listado de todas utilizando datos específicos para el filtrado o la localización de locales gracias a la función de GPS del dispositivo. Como tareas más avanzadas, consta de creación y modificación de restaurantes.

- **Listado de requisitos:** Enumeramos una serie de requisitos para nuestra aplicación

  - Por medio de la aplicación podemos encontrar cualquier local de restauración en la base de datos (Siempre que se encuentre registrada).

  - Cada ficha contendrá toda la información actualizada de contacto del restaurante facilitando así la información básica al usuario que realiza la búsqueda con esa necesidad.

  - La aplicación debe utilizar la geolocalización del dispositivo para que el usuario pueda saber en todo momento donde se encuentra ubicada su búsqueda actual. Esto implica que la propia app contenga un servicio de mapa, y no tenga que utilizar uno externo. De esta forma hablamos de mayor integridad teniendo una aplicación más completa.

  - La app permitirá tanto agregar como borrar cualquier restaurante a los usuarios. Podrá ser desde la ficha del restaurante o desde la localización del mapa.

  - El restaurante podrá ser “reportado”, cuando el número de estos sea de 5 o superior, el restaurante podrá ser eliminado por cualquier usuario.

  - Los usuarios debido a demanda, utilidad y por comparativa con otras apps, solicitan que los restaurantes puedan ser guardados como “favoritos”. Para este requisito será necesario la funcionalidad avanzada de cuentas de usuario.

3. Diseño

En el siguiente apartado mostramos el diseño de las diferentes pantallas o layouts de la aplicación que se han ido desarrollando en base al prototipado original:

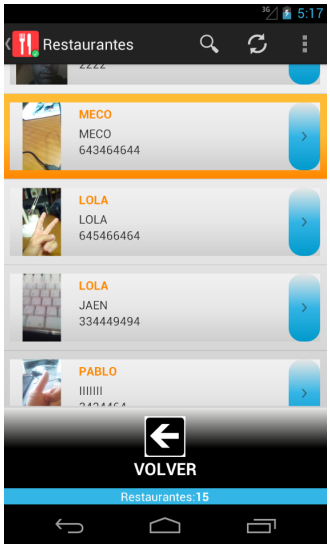
Menú principal

Búsqueda

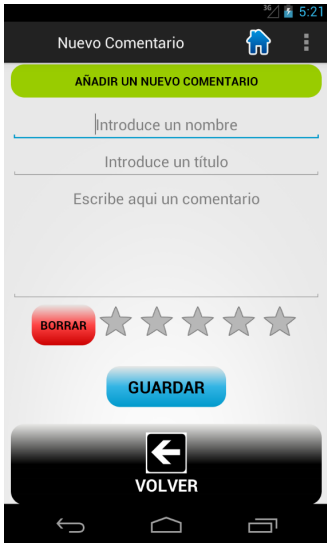
Información

Añadir Restaurante

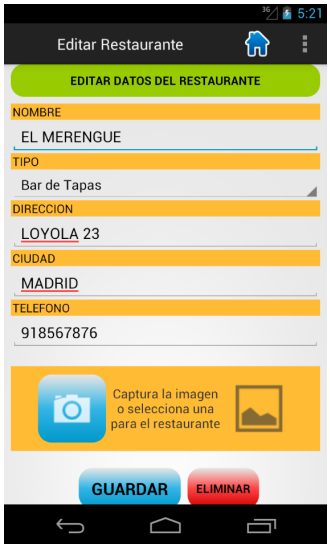
Ficha Restaurante



Listado Restaurante



Añadir Comentario



Editar Restaurante

Se incluye un pequeño apartado del diseño conceptural referente a los escenarios de uso de la la aplicación:

Escenarios de uso: Definimos en esta fase diferentes escenarios de uso de la aplicación.

Para ***Usuario A*** definido como usuario de la fase de análisis describimos los siguientes casos de uso:

- Caso A

El usuario se desplaza a una zona en donde se encuentra un restaurante del cual sólo sabe el nombre o la dirección.

- Caso B

Después de haber comido en un Bar de tapas, el usuario quiere añadir este restaurante a la base de datos, pues cuando lo buscó no estaba registrado.

- Caso C

Un grupo de personas ha cenado en un local del centro. Les ha parecido que el servicio era inadecuado y quieren dejar un comentario negativo. Para ello desde la aplicación realizan una búsqueda del local por el nombre, muestran la ficha y añaden un comentario.

- Caso D

La persona quiere ver todos los mesones que existen en la zona donde se encuentra. En la aplicación abre la localización y filtra todos los locales existentes de la zona por el tipo y obtiene una ubicación de todos con el criterio.

- Caso E

Un usuario se entera que el número de teléfono de un restaurante habitual ya no es correcto y desea actualizarlo. Realiza una búsqueda del local, muestra la ficha y selecciona la edición para actualizar el nuevo teléfono.

4. Desarrollo

4.1. Herramientas

Se detalla una descripción de las herramientas utilizadas en el desarrollo de la aplicación.

- **Android Studio (Beta) v. 1.0.2** : Entorno IDE de desarrollo de google. Se trata de una versión Beta que pretende ser la aplicación que sustituya en un futuro el resto de entornos de desarrollo para Android. Actualmente el IDE más extendido es Eclipse.

  Se ha decidido utilizar este entorno por cuestiones de comodidad, debido a que la instalación del entorno incluye un optimizado gestor de paquetes y descargas así como que incluye todo el entorno de desarrollo necesario para Android. En contraposición, eclipse requiere la instalación de paquetes adicionales. Por otro lado, se tiene en cuenta que es el nuevo entorno que pretende sustituir antiguos.

- **PhotoShop CS6** : Editor de imagenes rasterizadas de Adobe.

  Gracias a este editor es posible realizar todo el diseño de la iconografía de la aplicación así como de todas las fuentes necesarias que se utilicen. Permite adaptar las imágenes a las diferentes resoluciones de los *layouts* o plantillas de Android y de todos los recursos. De esta forma la aplicación posee imágenes que se adaptan a las diferentes resoluciones que permiten utilizar el gran abanico de dispositivos que utilizan Android. El diseño de las imagenes implica crear resoluciones para los tipos definidos en android; *mdpi*, *hdpi*, *xhdpi*,*xxhdpi.* También es posible crear plantillas personalizadas con atributos como la orientación

- **ImageOptim** : Optimizador de imágenes.*[6]*

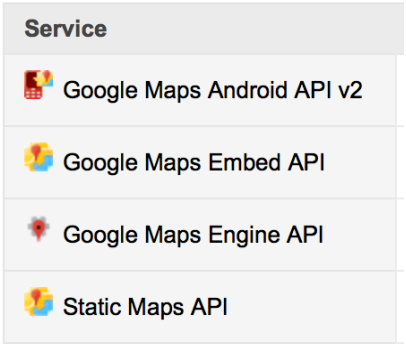
  Esta aplicación junto con las opciones de exportación de Photoshop se utiliza para optimizar las imágenes de forma que se pueda realizar una compresión y reducir el tamaño de estos recursos sin perdida de calidad. Así se optimiza la aplicación, que requiere menor espacio y se mejora la velocidad de uso con mejores resultados.

- **Parse***[7]*: Se utiliza el servicio de *cloud* proporcionado por *parse,* siendo gratuito hasta un número determinado de peticiones mensuales. El servicio se encarga de proporcionar todas las tareas de *Backend* liberando al programador del diseño e implementación enfocado en gestión de bases de datos y API remota. El servicio pone a disposición del programador una librería que se incluye en el entorno de desarrollo y en el proyecto que hace uso de esta API para gestionar el espacio *cloud* para el cliente.

  Dada la complejidad en el desarrollo de las APIs necesarias para el manejo de los datos, se opta por el uso de este servicio ahorrando el coste de recursos en tiempo y dinero, proporcionando un foco más centralizado en el diseño de la aplicación. Como aspecto negativo podemos decir que se depende en mayor grado de un servicio externo y se tratan menos aspectos del desarrollo de la aplicación. En el caso de mayores recursos, el objetivo ideal es realizar también este desarrollo creando el *backend*

necesario por medio de las tecnologías JAVA, PHP, JSON, MySQL, y HTTP junto con un servicio de *hosting* web donde alojar los datos.

- **Otros:** Para el uso del GPS del dispositivo móvil y la geolocalización hacemos uso de las API de google en su portal de servicios de desarrolladores. La app requiere el uso de unas librerías y APIs de google y de la utilización de las key que se utilizan para la app registrada en **console.developers.google.com**. Por lo que es necesario darse de alta en google y generar la clave gratuita para el uso de los mapas. Desde la consola de google a su vez es necesario activar los servicios que permiten hacer uso de estas funcionalidades.



Consola con los servicios activos

4.2. Análisis del estado del proyecto

El estado actual del desarrollo del proyecto en relación a la planificación se encuentra en la fase 6 de la descripción del mismo. Las funcionalidades iniciales han sido alcanzadas en su mayoría según el plan establecido. Se han producido retrasos debido a trabas tecnológicas, desconocimiento del lenguaje y herramientas del servicio *parse*, y se ha tenido que adaptar muchas de estas funciones a las limitaciones de la tecnología y recursos. Otro aspecto que ha tenido que dejarse más de lado es el diseño visual de la aplicación, que no ha podido lograrse más en profundidad y se ha limitado además a un formato determinado sin diseñarse para layouts diferentes y orientaciones landscape.

Entre los elementos que más retrasos han producido tenemos:

- ***Implementación y uso de objetos ListView para mostrar los restaurantes***

La complejidad en este aspecto radicaba en combinar la consulta y obtención de los objetos de parse con el manejo de los datos resultantes, realizando una conversión para representarlos a través del adaptador del listview. Se ha utilizado la clase ***AsyncTask*** *de java para poder realizar las operaciones en segundo plano de obtención de los datos de parse[7], y utilizar paralelamente un cuadro de dialogo que muestra el proceso. Los objetos daban problemas a la hora de realizar la conversión con fallos de memoria y bloqueos en gran medida por culpa de las imágenes de los restaurantes.*

- ***Implementación y uso de los objetos que almacenan los datos de los restaurantes***

Se han utilizado clases particulares dentro de las actividades que guardan los objetos para pasarlos a través del resto. Un problema encontrado ha sido el flujo de los datos almacenados entre las actividades, ya que en función del tipo de consulta y resultado se cargan diferentes actividades. Por lo cual, a la hora de mostrar la ficha de un restaurante, se debe controlar de qué actividad proviene para recuperar ese objecto de la clase adecuada. Se han tratado de utilizar los métodos del tipo Parcelable para enviar los diferentes datos de cada objeto de restaurante entre actividades. La clase parse no tiene implementación de muchos tipos y no es parcelable, por lo que se ha tratado de incluir y adaptar la clase principal del objeto de restaurante con este tipo sin éxito.

- ***Implementación y uso de los objetos imagen y Parse[7] de los restaurantes***

Las imágenes han tenido que comprimirse para poder manejarse por su peso. Inicialmente solo pudo implementarse la captura de la imagen y compresión para almacenar en parse, finalmente se ha podido añadir funcionalidad para que la foto pueda ser seleccionada internamente desde el dispositivo. Aunque siguen dando fallos cuando se trata de imágenes de gran tamaño, que colapsan la memoria del dispositivo.

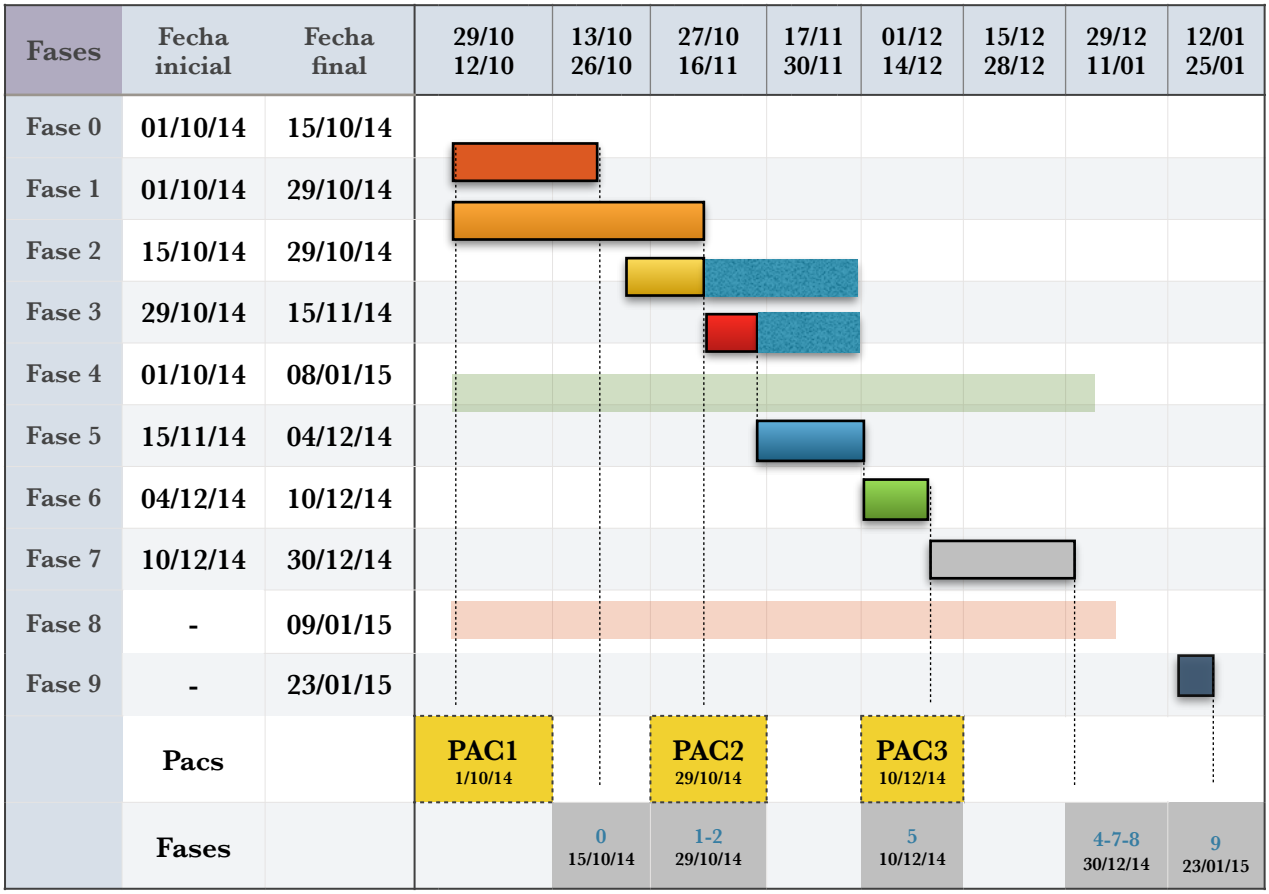
- ***Implementación y uso del flujo de la aplicación con las actividades***

Las actividades tienen una mención especial y en caso de actualización de objetos es muy importante. Se ha hecho uso de los métodos del flujo de la actividad para llamar a funciones que actualizaban objetos con los datos de los restaurantes y asi controlar las acciones del usuario cuando se refresca o se navega a través de las diferentes pantallas de la aplicación.

- ***Otros aspectos***

Existen otros aspectos que han cambiado a lo largo del desarrollo y han sido adaptados. Entre ellos destacamos las actualizaciones de librerías y de la herramienta de android studio que inicialmente se comenzó a utilizar en su versión 0.8.2 pero que actualmente es la 1.0.2. Otro aspecto detectado posteriormente fue el error en la seleccion del SDK mínimo del proyecto como la versión 3.0 o API 11, así como la 3.1 y la 3.2, los cuales se desarrollaron para cubrir tablets y no dispositivos móviles. Es por ello que el desarrollo no incluye la adaptación a estos, por lo que se ha actualizado el SDK mínimo del proyecto a la versión 4.0 (API 14).

A continuación se muestra el diagrama de ***Gantt*** de la planificación valorando de forma gráfica y aproximada la distribución del tiempo por las fases.



Recurso de tiempo extra de la fase:

5. Pruebas

La aplicación ha sido testeada a través del emulador Android ***AVD*** y desde el dispositivo utilizado a lo largo del desarrollo. Si bien se ha utilizado ambos métodos para el testeo en el desarrollo, es sobretodo el dispositivo móvil al que más uso se le ha dado. Hay que tener en cuenta que el propio emulador posee ciertas restricciones en algunas pruebas , como las indicadas a continuación:

- Uso de las cámaras frontal y trasera para añadir fotos al restaurante. AVD puede realizar una emulación de las cámaras de un dispositivo virtual pero no nos permite capturarlas.
- Uso de los mapas de la localización de los restaurantes. Existe una limitación descrita por google para el uso de los mapas *[3],* por la cual sólo es posible ejecutar los servicios de Google Play a través de un dispositivo con versión de android 2.3 o siendo por medio de una máquina virtual AVD que ejecute la versión 4.2.2 o superior.

Se incluye en este apartado una evaluación de test de la aplicación en la que se presenta varias fases para su consecución.

1. Recopilación de información sobre el usuario del test

A continuación aportamos la información sobre el usuario que participará en las pruebas.

- *Nombre: Alberto Pérez*
- *Edad: 35*
- *Experiencia con dispositivos móviles: Perfil bajo*
- *Ocupación: Estudiante y trabajador a tiempo parcial*

El usuario encaja con el perfil medio seleccionado que hará uso de la aplicación, por lo que supone que posee unas características que se encuentran en una media de todo el rango de los perfiles alcance de uso de la app. Posee un nivel de conocimientos bajo en el uso de aplicaciones, por lo que se pretende llegar a un gran número de personas con características del estilo. Cuanto más sencilla resulta la aplicación un mayor cantidad de usuarios de este perfil podrán hacer uso de ésta.

2. Tareas a realizar por el usuario

Proponemos al usuario realizar las siguientes tareas sobre la aplicación

1. Realizar una búsqueda de algún establecimiento por un nombre cualquiera.
2. Visualizar el listado de los restaurantes.
3. Añadir un nuevo restaurante con los datos y una fotografía.
4. Visualizar un local a través del mapa alrededor de la localización actual.
5. Buscar un local y editarlo cambiando algún dato de la ficha.
6. Eliminar un local.

3. Preguntas referentes a las tareas

Se plantean las siguientes preguntas respecto a las tareas llevadas a cabo por el usuario del apartado anterior:

1. *¿Te ha resultado accesible e intuitivo buscar un restaurante? ¿Piensas que buscar un restaurante es rápido y eficaz?*

2. *¿La forma de presentar los restaurantes es la más correcta e inteligible? ¿Existe algún aspecto a mostrar que considerarías incluir o eliminar?*

3. *¿Ha resultado fácil rellenar todos los campos de la ficha del restaurante? ¿La información de la ficha es adecuada y suficiente?*

4. *¿Consideras que se puede añadir algo más en la ficha de la localización? ¿Te resulta útil la manera de presentar la localización del local?*

5. *¿Al acceder a la edición piensas que hay muchos pasos intermedios? ¿Tienes la sensación de saber lo que haces en cada momento?*

6. *¿Te ha sido fácil y rápido eliminar un local? ¿Crees que es correcto la forma de eliminar el local?*

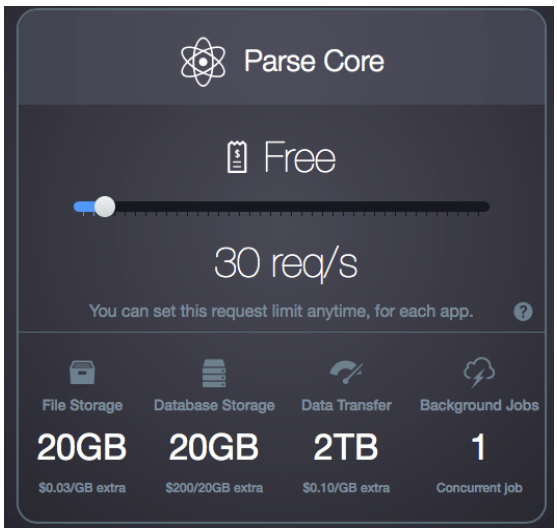
6. Consideraciones

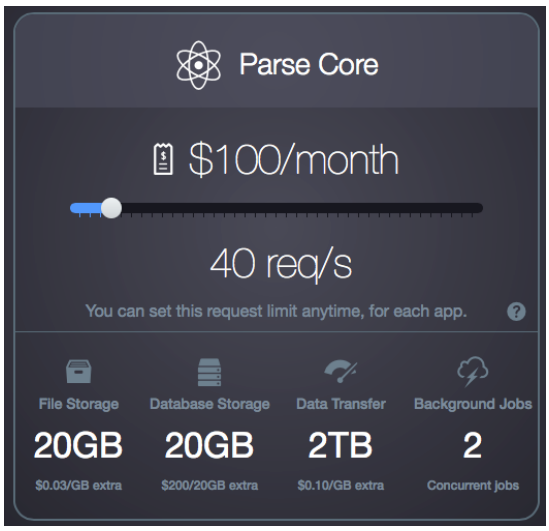
En este apartado de detallan otros datos extras relacionados con este proyecto

6.1. Valoración económica

Se debe considerar el aspecto económico extra del servicio facilitado por parse.com[7]. El servicio de pago incluye una ampliación en la limitación de numero de peticiones al servicio por segundo. Se trata de un aspecto a tener en cuenta en función del volumen de usuarios que pueda tener el producto para poder dar servicio a la demanda existente.

Las cuentas gratuitas disponen de un número de 30 peticiones por segundo. Cabe decir que el servicio permite definir más de una aplicación en su dashboard y que este número de peticiones se aplica a todas las apps al mismo tiempo, aunque el servicio nos permite limitar cada aplicación de forma individual. En las siguientes capturas podemos ver las características diferentes entre una cuenta Free y la siguiente de pago:





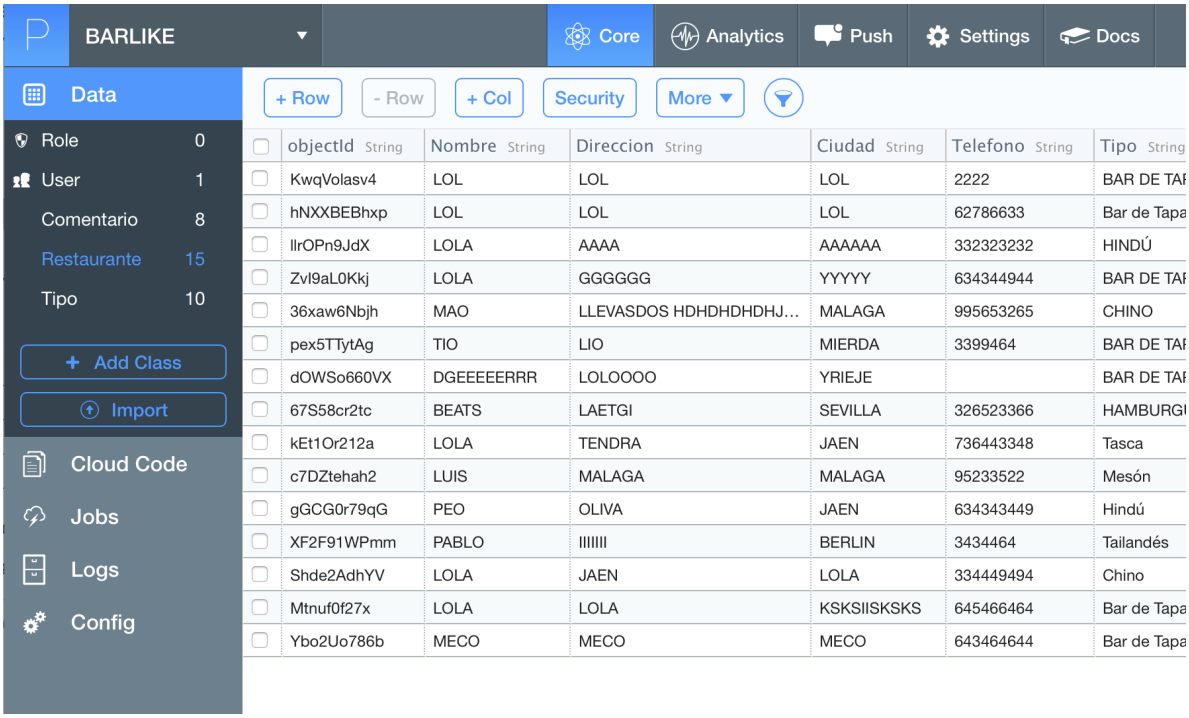
Como vemos si queremos ampliar la cuenta, a partir de 40 peticiones por segundo el servicio tendría un coste de 100$ mensuales. También el servicio incluiría una ampliación del número de operaciones concurrentes en el servidor.

El servicio incluye un número de *notificaciones push* que la aplicación puede enviar a los dispositivos. Este número es de un millón. Rebasando esa cifra el servicio tendrá un coste de 0.05$ por cada 1000 recipientes extra.

Además existe ***parse Analytics*** gratuito, herramienta que permite llevar un registro y análisis del numero de peticiones y de notificaciones de la aplicación.

Toda esta información puede ser consultada desde la página web. Es posible visualizar gráficamente estos datos y peticiones. El propio servicio informa al usuario diariamente a través del correo electrónico. Gracias a esta información y servicio de *parse* podemos hacer un seguimiento del uso de la aplicación de forma que sea posible ajustar el servicio a la demanda actual de los usuarios ya sea ampliando las características de la cuenta para mayor volumen de usuarios o disminuir cuando sea inferior.

Desde el dashboard del portal web podemos acceder a las tablas o "*classnames*", editarlas, borrarlas y modificar todos sus atributos o campos así como establecer las relaciones necesarias.



Aspecto de la pantalla principal "Core" donde podemos editar ClassNames

7. Conclusiones

7.1. Conclusiones sobre el trabajo

La conclusión referente al uso de una planificación es de que resulta totalmente necesaria y es crucial para el desarrollo del proyecto. Cuanto mayor nivel de detalle se especifique

en la planificación, mejores serán los resultados tanto en calidad del proyecto como en temporalización.

Hay que decir que es esencial la división del proyecto en diferentes etapas y se ha aprendido que cuanto mayor división, mejor control se puede tener sobre el desarrollo y conclusión de los objetivos.

La experiencia en la tecnología utilizada y en la planificación del desarrollo tiene un impacto muy grande sobre el resultado. Si esta experiencia y conocimientos sobre la tecnología son escasos la planificación es aún más necesaria para marcarse objetivos y tiempos, sobretodo porque hay que incluir en las etapas tiempo extra para aprender la tecnología.

7.2. Reflexión Critica sobre logro de objetivos

No se han logrado alcanzar todos los objetivos previstos. Inicialmente se planteó más requisitos, por lo que se estableció un ajuste más adecuado al tiempo disponible. Esa corrección ha permitido alcanzar los objetivos iniciales más básicos del proyecto, pero en cambio, los avanzados siguen en desarrollo y requieren un aprendizaje extra con más tiempo. Como autocrítica se puede considerar que la planificación debería haber sido más ajustada al nivel de complejidad de cada etapa.

7.3. Lineas de trabajo futuro

- Se pretende incluir un sistema de chat a la aplicación que permita interactuar con otros usuarios que hayan votado el restaurante objetivo
- La utilización de SQLite a nivel local para almacenar datos gestionados por el usuario de forma temporal cuando no se disponga de conexión de red para el acceso a la base de datos remota.
- Galería de imágenes para los restaurantes con una gestión de descarga de los mapas de bits y compresión según resoluciones, carga de red y capacidad del dispositivo.
- Estado actual de los restaurantes. Información en tiempo real de datos como el aforo actual del local o del horario de apertura.
- Adaptación al resto de dispositivos móviles y tabletas que ejecuten el software objetivo.
- Geolocalización junto con sistema de notificaciones por cercanía a los restaurantes.
- Incorporar las principales redes sociales a la herramienta permitiendo recomendar y compartir.

7.4. Análisis Crítico sobre seguimiento de la planificación

Durante todo el desarrollo del proyecto se ha tratado de fijar las metas en los hitos establecidos en la planificación. Inicialmente las primeras fases se han podido llevar a cabo con relativo éxito con solo varios días de diferencia. A partir de la fase 2 la lentitud en en la implementación de las funcionalidades y el arreglo de errores que han ido surgiendo han originado retrasos fuera del plan. Por consiguiente se ha tenido dejar de lado de forma temporal cualquier aspecto de diseño para centrarse en el código.

En cuanto a la planificación, se ha tenido que modificar en varias ocasiones al principio debido a que las etapas eran demasiado largas para todo el tiempo de desarrollo. Se han creado mas etapas para que sean más cortas y el control sobre el tiempo sea más preciso.

Igualmente no ha sido muy acertado el calculo de tiempo dedicado para las funcionalidades avanzadas del proyecto. El grado de complejidad y la poca experiencia hacen que consuma más tiempo.

8. Glosario

- **ACL** (*Access Control List*): Atributo de las classnames de parse que especifica permisos de lectura y escritura de usuarios de parse.
- **API** (*Application Programming Interface*) : Interfaz de procedimientos y funciones utilizados para realizar llamadas y permitir comunicación entre componentes de software.
- **APK** (*Application Package File*) : Es un tipo de paquete generado en el desarrollo final de una aplicación de Android, contiene a su vez las fuentes y permite ser instalado en el dispositivo destino.
- **AVD** (*Android Virtual Devices*) : Emulador de dispositivos Android utilizado por los principales entornos de desarrollo.
- **Backend:** Tipo de abstracción que hace referencia a la parte de software que procesa los datos que interactúan con los usuarios.
- **ClassName:** Nombre de clase. Comúnmente utilizado en parse para llamar una tabla de la base de datos.
- **Cloud:** Paradigma que permite servicios de programación de forma remota en un servidor.
- **Frontend:** Tipo de abstracción que hace referencia a la parte de software con la que el usuario interactúa.
- **Gantt:** Famoso ingeniero industrial que da nombre a los diagramas de Gantt, los cuales sirven para representar el progreso de etapas en función del tiempo en forma de barras.
- **hdpi** (*High density per inch*): Densidad de resolución alta de dpi para las fuentes de android.
- **Hosting:** Termino para identificar el servicio de alojamiento web en servidores para su acceso remoto.
- **HTTP** (Hypertext Transport Protocol): Protocolo de internet para mostrar las paginas web en exploradores.
- **landscape**: Se refiere a la orientación del dispositivo a modo panorámico.
- **ListView**: Objeto utilizado en el desarrollo de android que permite listar elementos de a través del uso de una estructura array.
- **JSON** (*JavaScript Object Notation*): Consiste en una notación utilizada para sustituir la de xml por su simplicidad y que es utilizada para el intercambio de datos.
- **layout:** Fichero xml que define la estructura visual de una pagina.
- **mdpi** (*Medium density per inch*): Densidad de resolución media para las fuentes de android.

- **MySQL:** Sistema de gestión de base de datos relacional.
- **PHP** (*Hypertext pre-processor*): Es un lenguaje de programación ejecutado en el lado del servidor generalmente diseñado para el desarrollo web.
- **push:** Es un tipo de comunicación establecida por el servidor para actualizar información sobre el cliente.
- **Rasterizadas:** Las imágenes de este tipo son creadas por mapa de bits en lugar de imágenes originadas con mapas vectoriales.
- **SDK** (*Software Development Kit*) : Corresponde con una paquete de desarrollo utilizado para Android.
- **SQLITE:** Es un sistema de gestión de base de datos relacional muy ligero escrito en C y utilizado por dispositivos Android.
- **xhdpi** (*Extra-high density per inch*): Densidad de resolución extra para las fuentes de android.
- **xxhdpi** (*Extra-extra-high density per inch*): Densidad de resolución doblemente extra para las fuentes de android.

9. Bibliografía

- **Libro**: Donn Felker with Joshua Dobbs, "*Android Application Development for Dummies",* Wiley Publishing Inc, Indianapolis, 2011. *[1]*
- **Libro**: Jesús Tomás, "El gran libro de Android Avanzado", Primera edición Marcombo ediciones técnicas, Barcelona, 2013. *[2]*

- **Web**: developer.android.com. *[3]*
- **Web**: stackoverflow.com. *[4]*
- **Web**: github.com. *[5]*
- **Web:** www.imageoptim.es *[6]*
- **Web:** www.parse.com *[7]*

10. Anexos

10.1. Manual de instalación

Método A: Instalación directa a través del paquete final android APK sobre el dispositivo.

El proyecto final será entregado de forma empaquetada en formato APK. Este tipo de paquete es ejecutado e instalado directamente desde la memoria interna o externa del dispositivo. Los siguientes pasos describen de qué forma puede ser instalado el producto final sobre un dispositivo Android:

1. ***Activación de Orígenes Desconocidos***: Para evitar problemas de instalación del paquete activaremos esta opción para que no exista restricción de paquetes no firmados o desconocidos.

La ubicación de la opción puede variar según dispositivo. Se encuentra en *Ajustes -> Seguridad -> Orígenes desconocidos* bajo la sección "Administración de Dispositivos"

2. ***Guardar el paquete en la memoria***: A continuación conectaremos el dispositivo a través del cable de datos al equipo que contenga el paquete para guardar el este en el dispositivo.

Al conectar el dispositivo debemos especificar que la conexión se establezca en el modo de *Almacenamiento USB masivo* para poder acceder a la memoria interna del mismo. Una vez realizado copiaremos el paquete a cualquier ubicación de la estructura de directorios del dispositivo, siempre que podamos posteriormente localizarlo y borrarlo finalmente cuando ya hayamos instalado la app.

Pantalla de aviso de reconocimiento y activación del modo USB

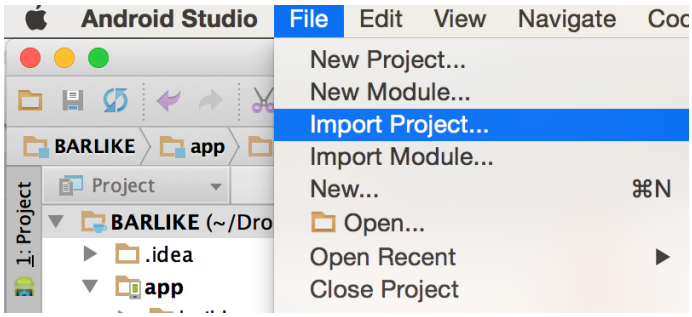
En caso de no acceder a la opción por defecto, igualmente podremos activarla desde los ajustes del dispositivo

3. ***Ejecutar el paquete****:* Abriremos el paquete apk desde cualquier gestor de archivos de android. Algunos dispositivos disponen de una aplicación cuya funcionalidad es la de buscar en la ubicación indicada el paquete y realizar la instalación.

Método B: Compilación del código fuente y emulación sobre el ***AVD*** de android.

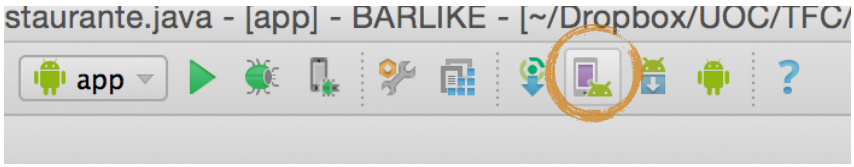
Este método requiere las fuentes de la aplicación completas y un entorno de desarrollo. Podrá ser compilado a través de Android Studio o Eclipse. Se recomienda utilizar el primero.

1. **Importar el proyecto**: Desde Android Studio debemos importar el proyecto situado en la carpeta BARLIKE. Para ello desde el menú Android Studio seleccionamos *File -> Import Project*



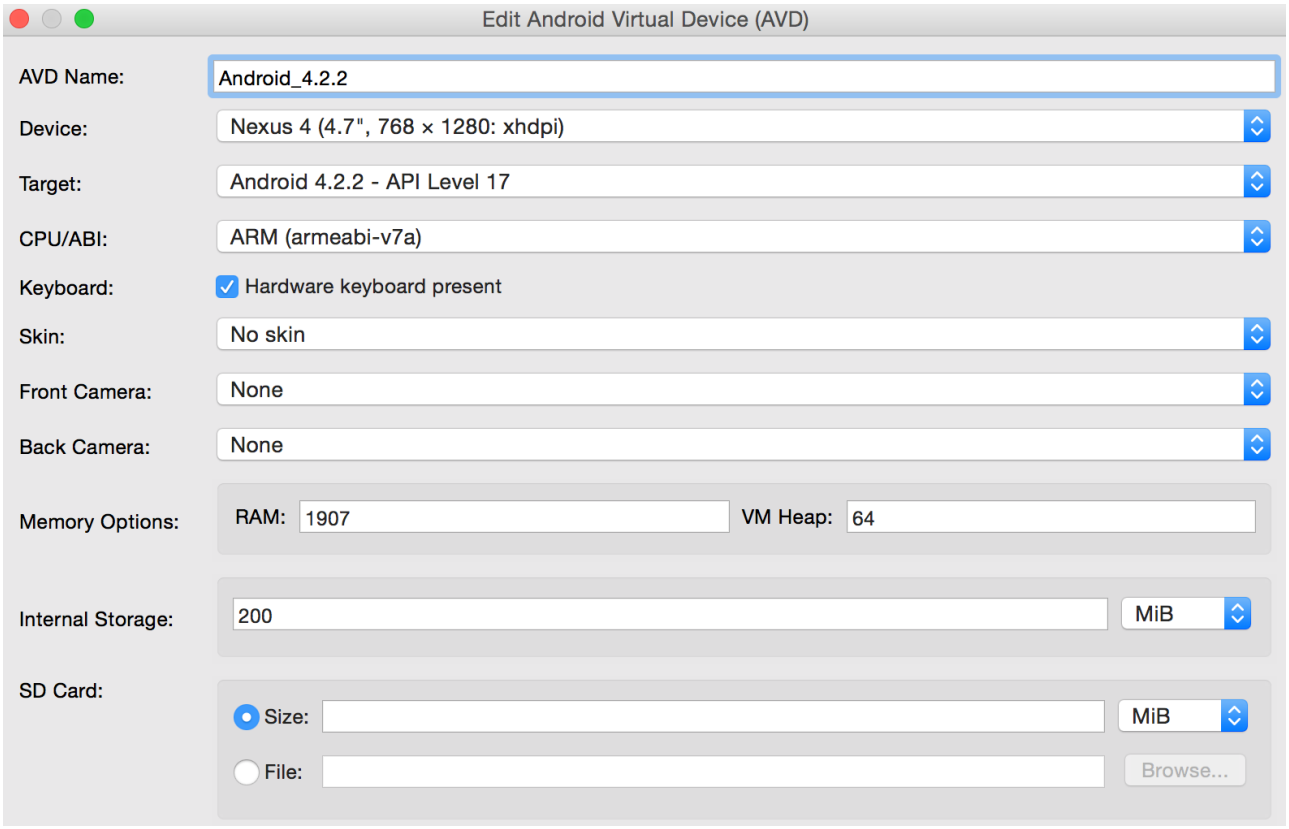
Importar el proyecto existente

2. **Crear un dispositivo virtual:** Lo siguiente necesario será configurar una máquina virtual para un dispositivo adecuado al del proyecto. Ejecutamos el *AVD manager* desde la barra de iconos de la aplicación.

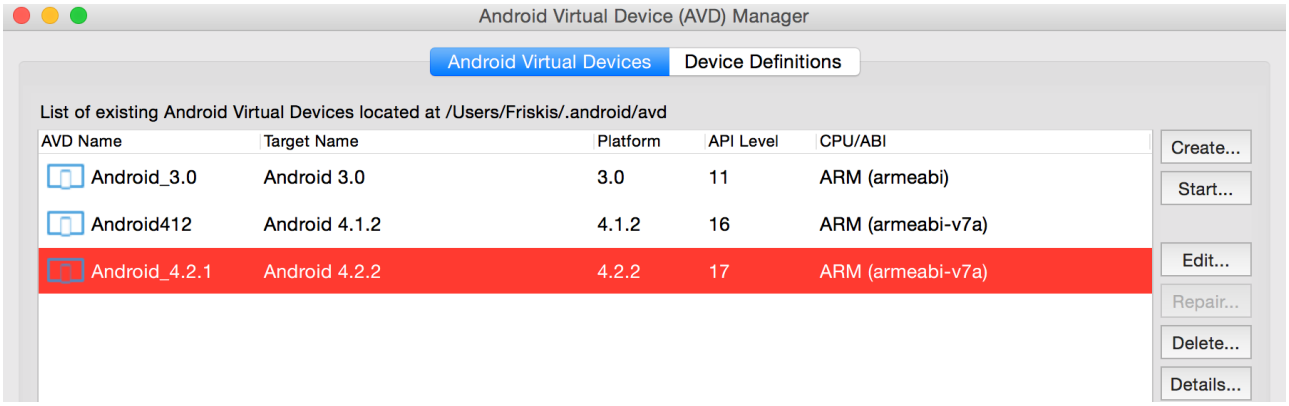


Panel AVD Manager

Desde la ventana principal resultante debemos crear un nuevo dispositivo virtual que cumpla como requisito el ***SDK*** mínimo para el que ha sido desarrollado el proyecto, es decir API 11, correspondiente a la versión 3.0 de Android. Por razones de optimización, se puede utilizar una maquina virtual para la versión de nuestro dispositivo de prueba, la 4.2.1 (API 17). Aunque los datos de configuración pueden ser modificados y personalizados, se recomienda en primera instancia especificar los siguientes mostrados:



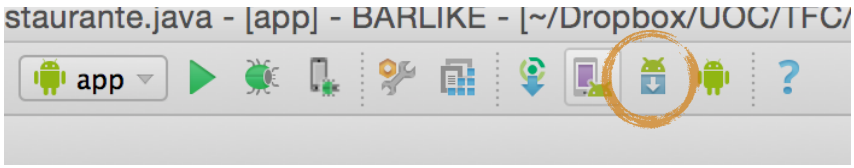
Panel de Creación de un nuevo dispositivo virtual



Panel de listado de máquinas virtuales

Al guardar la configuración del nuevo dispositivo podremos listar todas las máquinas virtuales existentes:

3. **Descargar los paquetes necesarios**: Ahora bien, antes de compilar nuestro código y emularlo en la máquina virtual será necesario que tengamos todas las librerías y paquetes de compilación necesarios para el nivel de la API utilizada. Para ello iniciaremos el SDK Manager para realizar la descarga y actualización de paquetes y librerías requeridos.



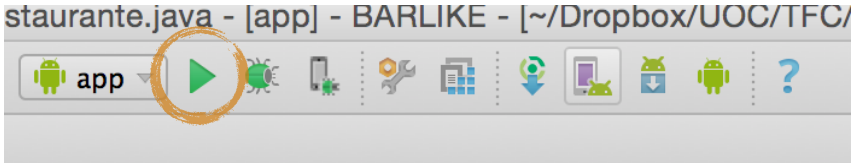
Panel AVD Manager

Una vez cargado el panel de descargas debemos procurar instalar las últimas actualizaciones de los siguientes paquetes para asegurar la ejecución de nuestro proyecto:

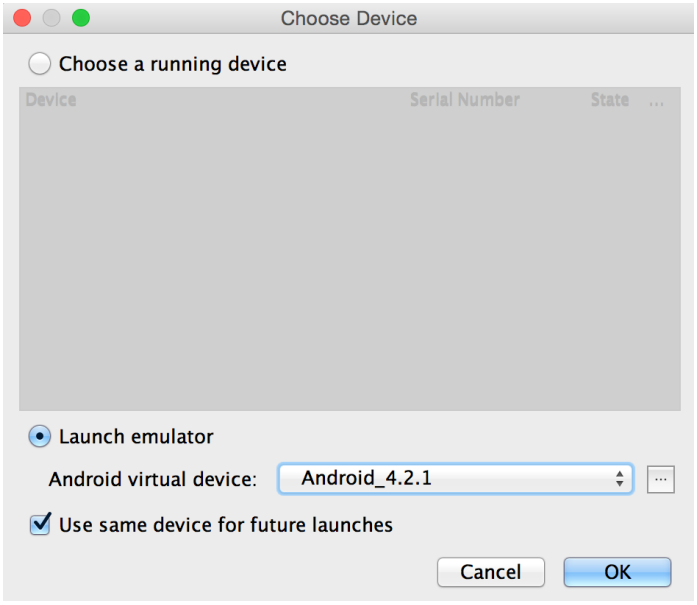
- *Android 4.2.2 (API 17)*
- *Android Support Repository*
- *Android Support Library*
- *Google Play Services*
- *Google Repository*

Si quisiéramos utilizar otro nivel de API, evidentemente necesitaríamos descargar los paquetes correspondientes. Cabe decir, que dentro de cada paquete es sólo necesario la descarga de la plataforma (*SDK Platform*), la imagen del procesador utilizado para el emulador y las APIs de google para los mapas.

4. **Ejecución de la aplicación:** Ejecutaremos la aplicación sobre la máquina virtual. Una vez terminado los pasos anteriores ejecutaremos la aplicación sobre la máquina virtual. Desde el menú de iconos de la barra de menú ejecutamos la aplicación y en la ventana que aparecerá posteriormente seleccionamos el emulador como destino.

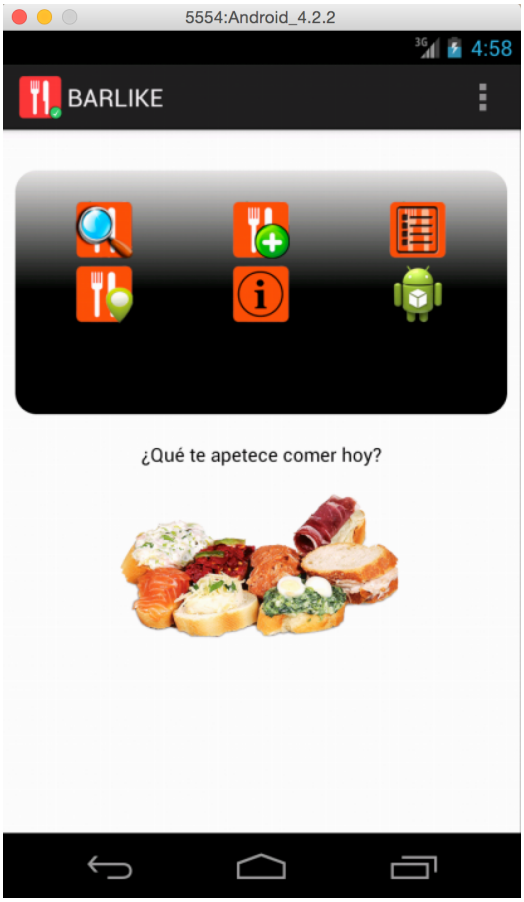


Panel de ejecución de la app

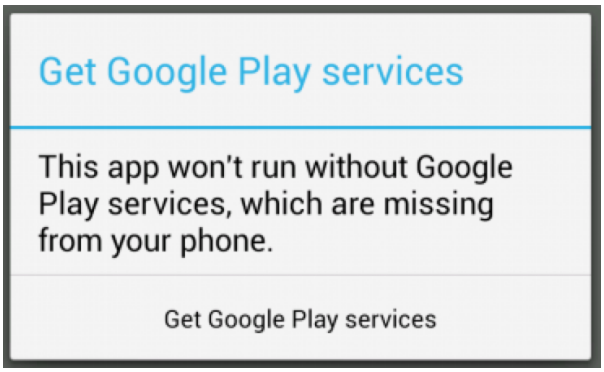


Díalogo de selección del destino

Finalmente, después de un rato iniciando el emulador se cargará ejecutando automáticamente el proyecto sobre la máquina virtual.



Pantalla de la máquina virtual



Pantalla de aviso de Google Play services: La limitación de google de los servicios en la máquina virtual cuando se muestran los mapas. En este caso recomendamos hacer uso de un dispositivo con versión superior a la 2.3.3 o una máquina virtual con una versión superior a la misma utilizada