Jose Antonio de Benito Suárez

debenito4@gmail.com

Descripción breve

Se trata de una apk creada en Android para dispositivos móviles en la cual se extrae los datos sobre los teléfonos corporativos y en la cual se distribuirán en grupos para después poder realizar llamadas a los contactos sin la necesidad de incluirlos en la agenda personal del usuario.

DOCUMENTACIón APk AGENDA

Documentación del listín telefónico

Contenido

[Introducción 2](#_Toc442359204)

[Implementaciones 2](#_Toc442359205)

[RESTRICCIONES Y ESTÁNDARES 3](#_Toc442359206)

[Aplicación 3](#_Toc442359207)

[Respuesta del servidor a la aplicación 3](#_Toc442359208)

[Restricciones de los parámetros 4](#_Toc442359209)

[Ejemplo de fichero 4](#_Toc442359210)

# Introducción

En este documento, se especifica la forma en la cual se ha realizado la apk, tanto sus restricciones como su realización a la hora de implementación.

Para centrarnos un poco en lo que se requiere de esta aplicación, la describiremos como una posible agenda o listín telefónico en la que se encuentran tanto los contactos corporativos asignados por grupos o asociaciones de personas así como sus números teléfonos. Tras ello, podremos realizar llamadas a esas personas sin la necesidad de añadirlos a nuestra agenda telefónica.

En cuanto a la apk que se pretende crear se realizaran en diferentes implementaciones, lenguajes de programación, estándares y restricciones, las cuales veremos a lo largo de este documento.

# Implementaciones

Para las implementaciones en este caso se escogieron dos tipos que son compatibles entre sí, ya que , utilizan el mismo lenguaje de programación a la hora de realizar la lógica del programa , y la diferencia estaría en las interfaces.

Las implementaciones serían:

1. **Android**: Esta parte es la encargada del desarrollo de interfaces. Su realización está programada con Android Studio y su lenguaje de programación trabajaría con las interfaces. (en la parte lógica utiliza java con sus propias restricciones.)

En nuestro caso, se utiliza solo la parte de android y la parte lógica, aunque utiliza el mismo lenguaje en este caso java, se implementara en otra plataforma Eclipse y se enlazara con las partes de interfaces creadas.

* 1. Sigue unas restricciones técnicas en cuanto al formato e implementación, las cuales nos centraremos en la parte de interfaces :
     1. Utilización de plantillas con un tipo de marcado estilo HTML propias de android mas la utilización de clases Java, las cuales implementan la interface Activity con sus métodos.

(Para más información sobre esto mirar <http://developer.android.com>).

1. **Java**: Se encargara de la parte lógica de nuestro programa.

Como anteriormente se ha comentado Android trabaja con java por lo tanto solo se encargara de enlazar la parte interfaces creadas con el programa java.

En cuanto a las restricciones que se han de seguir:

* 1. Java sigue unas restricciones en cuanto al lenguaje a utilizar y el formato de su código, por lo tanto nos restringiremos al Java-Code-Convenccions.
  2. También nos restringiremos a varios patrones de diseño para la implementación de la lógica para hacer el código más legible y fácil para su lectura.
  3. Finalmente realizaremos un archivo con extensión .jar (Extensión de java) en cual nos permitirá añadir este fichero como una librería más de nuestra apk e importarla.

1. **Protocolo HTTP, dentro de java, formando servicio REST**: Realizaremos peticiones a un documento php el cual nos devolverá los teléfonos y grupos.
   1. Las restricciones del protocolo HTTP se encontrarían en la siguiente referencia <https://es.wikipedia.org/wiki/Representational_State_Transfer>
   2. Además de las restricciones del propio protocolo tendremos las restricciones java que forman la creación de la petición HTTP, el acceso a datos, y la forma de tratarlos.
2. **PHP**: Este es el encargado de la transferencia de archivos entre la base de datos, en nuestro caso el Active directory y la respuesta a nuestra petición HTTP.
   1. Esta parte sigue un lenguaje específico y unas restricciones las cuales se muestran en la siguiente referencia <http://php.net/manual/es/index.php> .

# RESTRICCIONES Y ESTÁNDARES

Además de las restricciones y estándares vistos en el apartado anterior, proporcionaremos otros que no se basan en la forma de implementación, es decir en el formato u el lenguaje de programación sino, más centrados en la forma de modificación en cada fase del proyecto.

Nos ayudaremos de un repositorio de control de versiones en la nube llamado Github y el propio controlador de versiones Git.

Esto nos permitirá seguir la evolución de cada proyecto en cada fase y realizar los cambios especificados en el momento que se desee, sin el problema de pérdida del proyecto.

También nos ayudaremos de documentación asociada, como puede ser este documento o comentarios que se realizaran a lo largo del código para saber que realiza cada parte.

Los proyectos repositorios Github creados para este proyecto son los siguientes:

* Lógica de la agenda que se encuentran en el siguiente link : <https://github.com/debenito/AgendaLogic>
* Interfaces y producto final : <https://github.com/debenito/Agenda-4.4.2>

Por último se ha de tener en cuanta una restricción muy importante y es que solo funcionara en móviles con versiones igual o superior a android 4.4.2. En cualquier otro caso esta aplicación podría dar fallos en cuanto interfaces.

# Aplicación

## Respuesta del servidor a la aplicación

Como bien hemos explicado en apartados anteriores, nuestra aplicación extraerá los datos a través de un servicio REST implementado gracias a un fichero php, al cual nos conectaremos a través de una URL [[1]](#endnote-1)[[2]](#endnote-2) y tras el protocolo HTTP (realizando una petición GET) que nos devolverá una cadena de caracteres. En esta cadena de caracteres se separa cada pertenencia a grupo y agrupación o persona física o entidad por un **“;”** tras ello sacaremos los parámetros necesarios, que serán divididos en dos partes:

1. Grupos y Agrupación : Una persona o entidad puede pertenecer a una agrupación y dentro de ese agrupación a un grupo, es decir , por ejemplo el grupo sería Grupo parlamentario en este caso la agrupación seria Grupo Reins y la entidad Sede de Reins, para ello se separan por **“:”** y de este se obtendrán los siguientes parámetros en este orden:
   1. Imagen: Imagen del grupo al que pertenecen, esta será volcada directamente en formato texto en base64.(nota la especificación del formato base64 se encuentra en la siguiente URL[[3]](#endnote-3) y además se ha de tener en cuenta que por ello se ha escogido “:” y el “;” para separar ya que no existen en este)
   2. Nombre del Grupo: Será el nombre del grupo, recibida en formato texto.
   3. Nombre de la Agrupación: Será el nombre de la agrupación a la que pertenecen dentro de ese grupo, también en un campo de texto.
2. Personas física o entidad: Dentro de este grupo, también separados por **“:”** , obtendremos los siguientes datos o parámetros para esta aplicación en el siguiente orden:
   1. Imagen: Imagen del grupo al que pertenecen, esta será, volcada directamente en formato texto en base 64.
   2. Nombre de la persona o la entidad: Nombre y apellidos que identifica a la persona o nombre que identifica a la entidad como campo de texto. En el caso del nombre y apellidos de una persona este vendrá separado por **“:”**.
   3. Teléfono: Número de teléfono del contacto como un campo texto.

### Restricciones de los parámetros

Dentro de los parámetros descritos con anterioridad se describirán las siguientes restricciones, las cuales pueden surgir.

1. Si cualquier parámetro es nulo o no existe se dejara vacio pero con los “:” para la separación clara de los elementos. En el caso de que sea una imagen, se le pondrá encontrada en la aplicación, por el contrario si es un nombre se le pondrá el texto “No existe” y en el caso de teléfonos se dejara con ese valor.
2. Nuestra aplicación no va a tener nombres iguales dentro de los mismos grupos por lo tanto solo se le pasara el nombre.
3. También no tendrá números de teléfono iguales dentro de un grupo y además estos han de tener 9 dígitos ya que no se llamara fuera del país.
4. Se ha de cumplir el formato explicado anteriormente, en cualquier caso si el formato anterior no se siguiera el orden, es decir no se encontraran ni **“:”** y **“;”** la aplicación debería mostrar un mensaje de error al cliente especificando que se está teniendo problemas son del servidor.
5. Las imágenes como se ha indicado anteriormente han de venir en el estándar base64, debido a problemas que puedan surgir a la hora de sacar la cadena con caracteres como los “:”.
6. El formato de salida de los datos seguirá el siguiente ejemplo en el orden indicado:
   1. Imagen:Nombre\_grupo:Nombre\_agrupación;
   2. Imagen: Nombre\_agupación:Nombre\_persona:Telfono;
7. Según el formato anterior la imagen pertenecerá al último parámetro en caso de que una agrupación y sino al tercer parámetro en caso de que sea una persona o entidad.
8. La relación de dependencia entre agrupaciones viene dada por el último parámetro en el caso (a) y por el segundo parámetro en el caso (b).
9. En la primera línea del fichero se ha de encontrar el siguiente formato:
   1. Imagen::Nombre\_grupo;
10. Finalmente se ha de tener en cuenta que un grupo o agrupación no puede estar repetido dentro de el mismo.

### Ejemplo de fichero

Muestra de un fichero de entrada con 6 grupos o agrupaciones y 8 personas o entidades.

RXN0YSBlcyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:: Grupo Parlamentario;

RXN0YSBlcyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Grupo Parlamentario:Grupo Reins;

RXN0YSBlcyBsYSBub3RhIGRlIGVOFnaW5h:Grupo Reins:Asociacion Asede;

RXN0YSBlcyBsYSIGRlIGVzdGEgcOFnaW5h:Grupo Reins:Leo:Batistao:523123879;

SBlcyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Grupo Reins:Marta:Sanchez:123432567;

YSBlcyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Asociacion Asede:Representantes;

YSBlcyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Asociacion Asede:Juanma:Castaño:345462967;

XN0YSBlcyBsYSRhIGRlIGVzdGEgcOFnaW5h:Representante:Presidente:123432476;

RBlcyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Representantes:Vicepresidente:123452158

BsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Grupo Parlamentario:Favoritos;

0YSBlcyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Favoritos:Asociacion Reiquinats;

N0YSBlcyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Favoritos:Presidente Parlamento:124322345;

cyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Asociacion Reiquinats:Ramon:Sanjurjo:789080809;

lcyBsYSBub3RhIGRlIGVzdGEgcOFnaW5h:Asociacion Reiquinats:Lourdes:Requina:689087809;

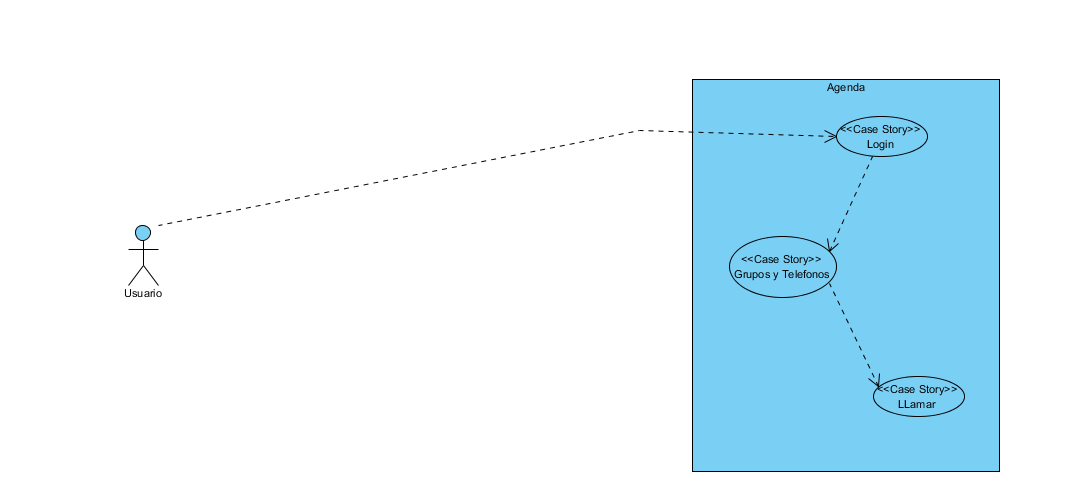
# ANÁLISIS

En este apartado describiremos la forma en que el cliente trabajara con nuestra aplicación además nos ayudaremos de un esquema para ver las funciones principales.

En lo que respecta al análisis, describiremos tres funciones principales, la primera seria el logueo del usuario. A partir del logueo tendremos dos pasos fundamentales:

1. Verificación correcta: El usuario pasará a una segunda pantalla en donde se encuentran los teléfonos y los grupos, a partir de ahí podrá realizar las funciones de búsqueda de teléfonos a través de los grupos, es decir, por ejemplo clicando en el grupo donde se encuentre el teléfono, y finalmente podrá cliquear en el teléfono de esa para llamar.
2. Verificación incorrecta: El usuario se quedaría en la misma pantalla de logueo.

Para ver mejor este trabajo se muestra un esquema, describiendo las acciones antes comentadas por el usuario sobre nuestra aplicación.



***Figura 1: UML Análisis***

1. [↑](#endnote-ref-1)
2. [1] (<http://eparlamento.jgpa.es/ws/WSListin.php>) [↑](#endnote-ref-2)
3. <https://www.ietf.org/rfc/rfc2045.txt> [↑](#endnote-ref-3)