

Cordecor

TFC – Aplicacions Mòbils (Android)

Memòria

Javier Martínez Sánchez
Coordinat per Joan Vicent Orença Serisuelo

Enginyeria Tècnica en Informàtica de Gestió
Juny 2014



Índex

1. <u>Introducció</u>	3
2. <u>Objectius</u>	4
3. <u>Funcionalitats principals</u>	5
4. <u>Planificació del projecte</u>	6
5. <u>Anàlisi Funcional</u>	7
6. <u>Anàlisi Funcional - Prototips de les Pantalles</u>	8
6.1. <u>Iniciar sessió. Menú principal</u>	8
6.2. <u>Contactes</u>	9
6.3. <u>Productes</u>	9
6.4. <u>Empreses i Noves Comandes</u>	10
6.5. <u>Històric de comandes</u>	11
6.6. <u>Perfil</u>	11
6.7. <u>Mapa</u>	12
7. <u>Disseny Tècnic</u>	13
7.1. <u>Client</u>	13
7.2. <u>Servidor</u>	13
7.2.1. <u>CodeIgniter. Framework PHP</u>	14
7.2.2. <u>Arquitectura RESTful</u>	15
7.3. <u>Disseny relacional de la Base de Dades</u>	16
8. <u>Implementació</u>	18
8.1. <u>Interfície</u>	18
8.2. <u>Entitats</u>	22
8.3. <u>Comunicació</u>	24
8.4. <u>Threads</u>	26
8.5. <u>Configuració de Google Maps v2</u>	27
8.6. <u>Exportació de l'aplicació</u>	30
9. <u>Manual d'usuari</u>	31
9.1. <u>Iniciar sessió</u>	32
9.2. <u>Menú principal</u>	32
9.3. <u>Llistat de contactes</u>	32
9.4. <u>Edició d'un contacte</u>	33
9.5. <u>Llistat de productes</u>	33
9.6. <u>Pantalla detall d'un producte</u>	34
9.7. <u>Llistat d'empreses</u>	34
9.8. <u>Edició d'una empresa</u>	35
9.9. <u>Realitzar una nova comanda</u>	35
9.10. <u>Selecció de productes</u>	36
9.11. <u>Historial de comandes</u>	36
9.12. <u>Perfil</u>	37
9.13. <u>Mapa</u>	37
10. <u>Conclusions</u>	38
10.1. <u>Fonts d'informació</u>	38

1. Introducció

Tota empresa que produeix algun tipus de producte i que necessita distribuir-los als seus clients, requereix des de ja fa anys de solucions informàtiques per agilitzar les comunicacions en tota la seva cadena de subministrament. Així mateix, any rere any, les aplicacions mòbils han anat guanyant protagonisme per facilitar una major accessibilitat a la informació i gestió de la comunicació entre empresa i clients.

Les aplicacions que permeten accedir a la informació de l'empresa en temps real, tal com la disponibilitat del producte, dades dels clients, etc. proporcionen una gran ajuda a figures clau d'una empresa com són els comercials. El continu increment de la competència, reforçada per la concentració empresarial i l'imparable avanç tecnològic fan dels comercials una imatge directe de l'empresa i de l'aplicació una eina senzilla d'administració i un instrument de màrqueting que farà augmentar les vendes.

2. Objectius

En aquest document es descriu l'aplicació Android desenvolupada per a Cordecor, empresa fictícia que crea i comercialitza productes de decoració per a la llar. L'aplicació dona suport als comercials d'aquesta empresa, dotant-los d'una sèrie d'eines que facilitin les seves tasques diàries. A més, és un instrument de màrqueting, ja que l'actualització de productes pot crear noves necessitats al client final quan el comercial se'ls hi mostri, també a noves empreses.

El treball dels comercials és visitar regularment els clients actuals per fer noves comandes i captar noves empreses que vulguin vendre els productes de Cordecor a les seves botigues. Tot això ho podran fer des de l'aplicació mòbil sense haver d'estar trucant a la central de Cordecor i així, optimitzar la capacitat de resposta i negociació davant clients.

L'app permet que el comercial pugui veure en temps real el llistat de productes i realitzar una nova comanda que especifiqui la quantitat de cada article que el client vol vendre a la seva botiga. A més, podrà gestionar una llista de contactes propis, veure les dades de les empreses clients i visualitzar un mapa que li indiqui la posició d'aquestes.

Aquesta aplicació Android s'ha desenvolupat amb codi natiu Java i el conjunt del projecte ha necessitat altres tecnologies com les referents a la part del servidor. Utilitzem Apache en un servidor Linux i desenvolupem els serveis REST amb PHP i MySQL.

3. Funcionalitats principals

Les funcionalitats més importants que ha d'implementar l'aplicació són les següents:

- **Iniciar sessió.** Pantalla d'accés per als usuaris registrats.
- **Menú principal.** Pantalla inicial amb botons que ens portin a cada secció i un botó per tancar la sessió actual.
- **Empreses.** Llistat d'empreses i pantalla d'edició per a crear noves.
- **Comandes.** Crear nova comanda i mostrar historial de les realitzades fins al moment per l'usuari actual.
- **Productes.** Llistat dels productes que té Cordecor.
- **Contactes.** Llistat de contactes de l'usuari actual i pantalla d'edició per a crear nous.
- **Perfil.** Pantalla per editar les dades personals de l'usuari actual.
- **Mapa.** Pantalla per mostrar la posició de les empreses.

4. Planificació del projecte

Cal fer la planificació del projecte tenint en compte les dates clau que marca el Pla Docent:

	Inici	Lliurament
PAC1	27/02/2014	12/03/2014
PAC2	13/03/2014	16/04/2014
PAC3	17/04/2014	21/05/2014
Lliurament Final	22/05/2014	11/06/2014

Així s'organitza el següent calendari de tasques:

Tasca	Inici	Final
PAC1		
Tasques principals i calendari	27/02/2014	12/03/2014
PAC2		
Anàlisi del sistema	13/03/2014	27/03/2014
Disseny de l'arquitectura	28/03/2014	16/04/2014
PAC3		
Disseny i programació de les pantalles	17/04/2014	21/04/2014
Programació de la interactivitat	22/04/2014	26/04/2014
Base de dades de l'app	27/04/2014	27/04/2014
Classes entitat	28/04/2014	28/04/2014
Control de la base de dades	29/04/2014	03/05/2014
Desenvolupament PHP REST	04/05/2014	08/05/2014
Comunicació http REST de l'app	09/05/2014	13/05/2014
Últims retocs	14/05/2014	18/05/2014
Proves	19/05/2014	21/05/2014
Lliurament Final		
Redacció de la memòria del projecte	22/05/2014	08/06/2014
Creació del vídeo	09/06/2014	11/06/2014

5. Anàlisi Funcional

Desglossament dels requeriments i funcionalitats que ha de tenir l'aplicació:

- **Iniciar Sessió** a la primera pantalla, introduint l'email i la contrasenya. Si la validació al servidor ha estat correcta accedim a la pantalla de menú principal.
- Pantalla **Menú Principal**. Es mostren els següents botons: Empreses, Contactes, Historial Comandes, Productes, Mapa, Perfil. A més, un últim botó per tancar la sessió de l'usuari actual.
- Consulta del llistat d'**Empreses** prement el botó del menú principal.
 - Recerca d'empreses escrivint en un camp de text.
 - Afegir una nova empresa.
 - Veure i editar les dades d'una empresa prement sobre una fila del llistat.
Des d'aquesta pantalla podem crear una **Nova Comanda**:
 - Dins la pantalla de creació de comanda, prement el botó **Productes** accedim a la pantalla del llistat de productes i prement sobre cada fila de producte s'obrirà un formulari per seleccionar la quantitat que volem d'aquest.
- Consulta del llistat de **Contactes** prement el botó del menú principal.
 - Recerca de contactes escrivint en un camp de text.
 - Afegir un nou contacte.
 - Veure i editar les dades d'un contacte prement sobre una fila del llistat.
Dins d'aquesta pantalla, a més d'actualitzar les dades, també podem eliminar aquest contacte.
- Consulta del llistat de **Productes** prement el botó del menú principal.
 - Recerca de productes escrivint en un camp de text.
 - Veure les dades d'un producte prement sobre una fila del llistat.
- Consulta del llistat de **Comandes** prement el botó del menú principal.
 - Recerca de comandes escrivint en un camp de text.
 - Veure les dades d'una comanda realitzada anteriorment prement sobre una fila del llistat.
- Pantalla **Perfil**. Pantalla per editar les dades personals de l'usuari actual.
- Pantalla **Mapa**. Localització de les empreses com a punts d'interès.

6. Anàlisi Funcional - Prototips de les Pantalles

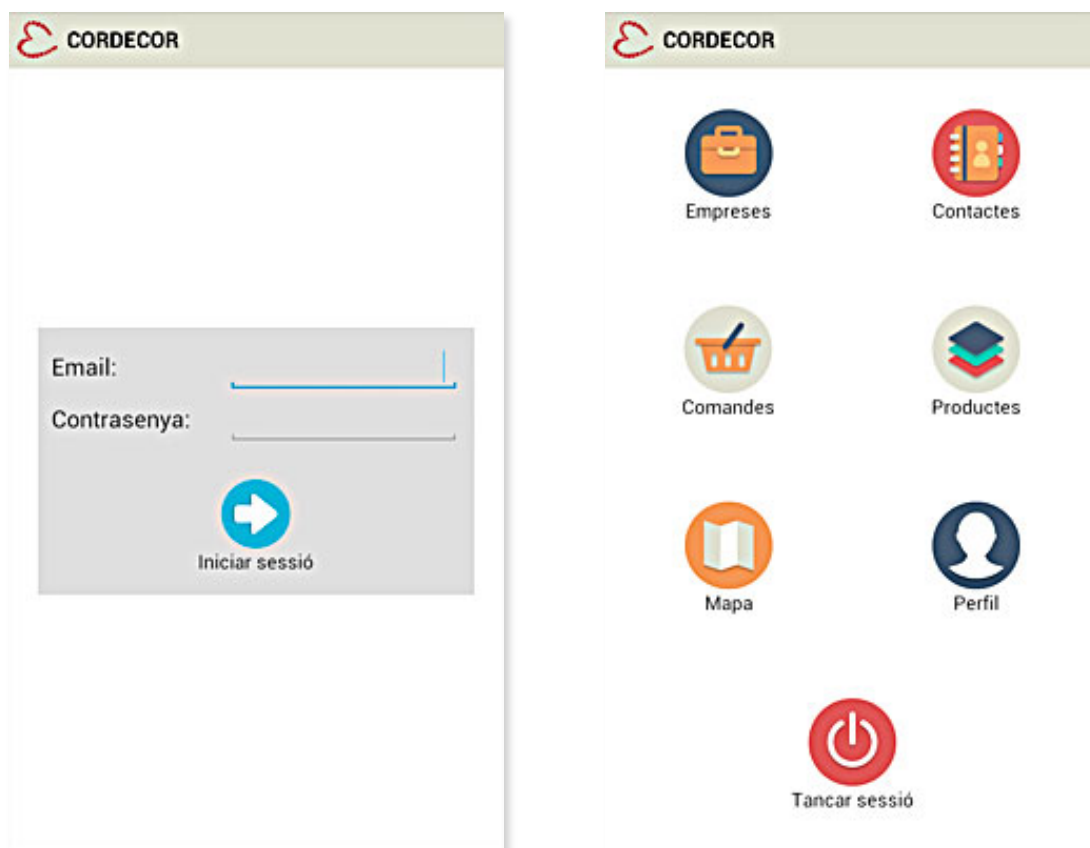
En aquest apartat passem a veure el prototip de l'aplicació. Les següents imatges mostren les diferents pantalles de la interfície d'usuari.

Després de la pantalla inicial que mostra durant uns segons el logotip de Cordecor es carrega la pantalla que ens permet iniciar la sessió i poder accedir a les diferents opcions de l'aplicació.

6.1. Iniciar sessió. Menú principal

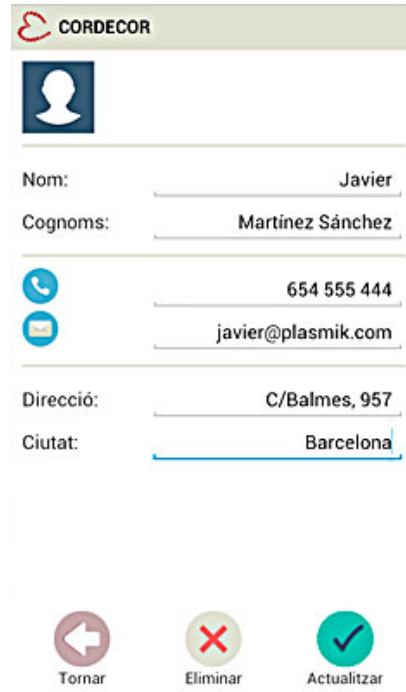
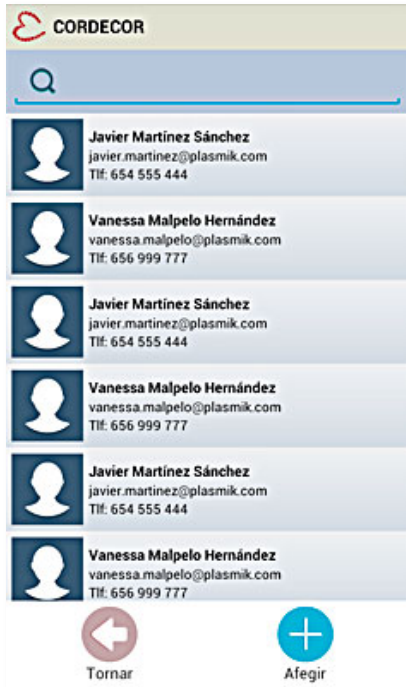
Després de realitzar la introducció de l'email i la contrasenya per iniciar sessió es mostra el menú principal. Des d'aquí l'usuari pot accedir a les seccions: Empreses, Contactes, Productes, Comandes, Mapa i Perfil.

Prement el botó Tancar sessió s'obrirà un pop-up que li preguntarà si realment vol tancar la sessió.



6.2. Contactes

Llistat de contactes i pantalla d'edició de cada contacte. L'usuari pot actualitzar les dades o eliminar el contacte amb els botons inferiors.



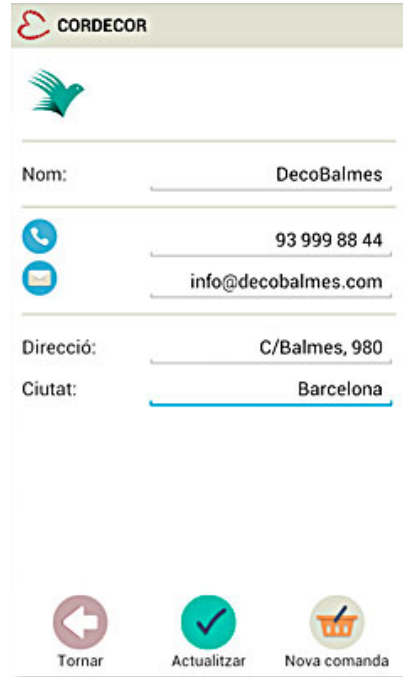
6.3. Productes

Llistat de productes i pantalla de detall de cada producte.



6.4. Empreses i Noves Comandes

Llistat d'empreses i pantalla d'edició de cada empresa. Per realitzar una nova comanda associada a aquesta empresa l'usuari ha de prémer el botó inferior Nova comanda.



En la creació d'una comanda l'usuari ha de prémer sobre el botó Productes per seleccionar les quantitats d'aquests.

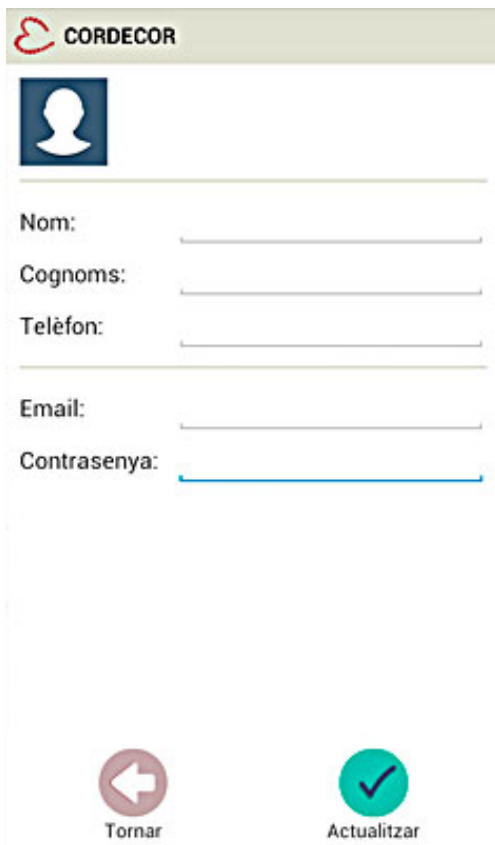




6.5. Històric de comandes

Recerca de comandes escrivint en el camp de text superior.

Si l'usuari prem una fila en concret accedirà a la pantalla de detall d'aquesta comanda que va realitzar anteriorment.



6.6. Perfil

L'usuari pot canviar el seu nom, cognoms i telèfon.



6.7. Mapa

A la pantalla de Mapa l'usuari pot veure la localització de les empreses representades com a punts d'interès sobre el mapa.

7. Disseny Tècnic

A continuació passem a tractar la part tècnica del disseny de l'aplicació.

L'arquitectura global d'aquest projecte és de tipus Client-Servidor. En aquest tipus d'estructura la xarxa informàtica de l'empresa tindrà un equip servidor, que funciona com la central de l'organització, i un grup de dispositius client que es connectaran a aquest servidor.

La part del servidor és un ordinador de gran capacitat, engloba la capa de serveis i base de dades, amb una gran quantitat de dades emmagatzemades. Els clients són ordinadors, o en aquest cas dispositius mòbils, que els empleats utilitzen per dur a terme les seves tasques diàries.

7.1. Client

Els clients d'aquesta arquitectura són els smartphone amb el sistema Android de Google que tenen instal·lada l'aplicació de Cordecor.

Android és un sistema operatiu basat en Linux dissenyat principalment per a dispositius mòbils amb pantalla tàctil com smartphones i tablets. És de codi obert i alliberat sota la llicència Apache, que permet que el programari pugui ser modificat i distribuït pels fabricants de dispositius, operadores mòbils, així com a usuaris individuals.

<http://developer.android.com/sdk/index.html>

7.2. Servidor

L'altra part de l'arquitectura, el servidor, engloba la capa de serveis i base de dades. El servidor d'aquest projecte és una plataforma de tipus LAMP (Linux Apache MySQL PHP). L'ús d'LAMP ofereix un sistema més barat al client ja que els seus diferents components utilitzen la llicència "GNU General Public License": http://ca.wikipedia.org/wiki/GNU_General_Public_License

Linux:

Linux és la capa de nivell més baix, el sistema operatiu en el qual s'executen cada un dels altres components.

Apache:

La següent capa és Apache, el servidor web. Apache proporciona la mecànica que ens permet oferir pàgines web als usuaris. Apache és un servidor estable i dels més utilitzats en els llocs web a internet.

MySQL:

MySQL proporciona la capa d'emmagatzematge de dades del sistema LAMP. Amb MySQL tenim accés a una base de dades molt adequada per al tipus de projecte que s'ha desenvolupat. Les dades de l'aplicació, productes, comptes i resta d'informació resideix en aquesta base de dades en un format que es pot consultar amb facilitat amb el llenguatge SQL.

PHP:

PHP és un llenguatge de programació simple i eficaç que proporciona la unió de la resta de parts del sistema LAMP. El component de PHP en realitat es troba dins d'Apache. S'utilitza Apache i PHP per crear pàgines dinàmiques que accedeixen a les dades de la base de dades MySQL i els mostra a l'usuari.

7.2.1. CodeIgniter. Framework PHP

En el desenvolupament de programari és comú l'ús de frameworks. Un framework és una estructura tecnològica formada per diferents mòduls de programari, que serveix de base per a l'organització i desenvolupament de nou programari. És una manera de crear aplicacions sense haver de començar des de zero i utilitzant parts de codi ja testejat. Ens permeten enfocar-nos en el nostre projecte, reduint al mínim la quantitat de codi necessari per a una tasca determinada.

Per al desenvolupament de l'aplicació PHP allotjada al servidor s'ha utilitzat el framework CodeIgniter. Aquest és un dels frameworks més lleugers i amb execució més ràpida, proporciona un ric conjunt de llibreries per a tasques comunament necessàries, així com una interfície senzilla i l'estructura lògica per accedir a aquestes llibreries.

CodeIgniter implementa el patró d'arquitectura programari MVC (Model Vista Controlador), que separa la lògica de negoci de les dades i de la interfície d'usuari.

<http://ellislab.com/codeigniter>

En aquest projecte s'ha utilitzat en concret la versió de Phil Sturgeon que implementa una arquitectura RESTful.

<https://github.com/philsturgeon/codeigniter-restserver>

7.2.2. Arquitectura RESTful

L'arquitectura implementada en la part del servidor és de tipus RESTful, nom que se li dóna als sistemes que segueixen els principis REST.

REST (REpresentational State Transfer) és un estil d'arquitectura que utilitza la tecnologia i els protocols existents de la web, incloent HTTP (Hypertext Transfer Protocol) i XML/JSON, per proporcionar la interacció entre els clients i els serveis que ofereix el servidor.

És d'ús freqüent en les aplicacions mòbils, ja que ofereix senzillesa i flexibilitat. A la comunicació client-servidor no requereix que el servidor hagi de guardar informació del client entre peticions consecutives. Cada missatge HTTP conté la informació necessària per satisfer la petició.

Reuneix un conjunt d'operacions que s'apliquen a tots els recursos d'informació, les més importants són GET, POST, PUT i DELETE. Cada recurs és direccionable únicament a través del seu URI (Uniform Resource Identifier).

Exemple d'operació GET realitzada per l'aplicació Android per obtenir els productes que tenim guardats a la base de dades del servidor:

<http://www.javierms.com/cordecor/index.php/rest/productes>

Si executéssim aquesta petició en un navegador web rebríem la següent cadena de text en format JSON:

```
{"status":401,"message":"Unauthorized","result":""}
```

JSON (JavaScript Object Notation) és un format lleuger que s'utilitza per a l'intercanvi de dades, és més fàcil d'analitzar i generar que l'XML. Abans de la seva aparició el format utilitzat pels serveis web era l'XML, però JSON ha passat a ser el preferit ja que és molt més lleuger.

Tornant a l'exemple veiem que realment no hem rebut el llistat de productes, observem que l'estat és "Unauthorized", això es deu al fet que la petició HTTP que hem fet no té assignat en la capçalera el username i password de l'usuari, per tant l'accés a les dades del servidor s'ha denegat.

En el cas de l'aplicació Android necessitem utilitzar la següent sentència Java a la nostra classe encarregada de realitzar les peticions HTTP:

```
request.setHeader("Authorization", "Basic " + Base64.encodeToString(  
(username+":"+password).getBytes(), Base64.NO_WRAP));
```

A continuació es mostren exemples de peticions GET que utilitza l'aplicació Android i el JSON que rep:

Usuari: <http://www.javierms.com/cordecor/index.php/rest/usuari>

```
{"status":200,"message":"OK","result":{"id":"2","nom":"Javier","cognoms":"Mart\u00ednez","telefon":"222124896","email":"javier@cordecor.cat","contrasenya":"password"}}
```

Empreses: <http://www.javierms.com/cordecor/index.php/rest/empreses>

```
{"status":200,"message":"OK","result":[{"id":"1","logo":"empresa_1.jpg","nom":"Cent100","direccio":"Consell de Cent, 20","ciutat":"Barcelona","telefon":"935153468","email":"info@cent100.com","latitud":"41.37603741398848","longitud":"2.14483180000002"}, {"id":"2","logo":"empresa_2.jpg","nom":"ViaDeco","direccio":"Via Laietana, 12","ciutat":"Barcelona","telefon":"934527891","email":"info@viadeco.com","latitud":"41.38300151399194","longitud":"2.180185300000062"}]}
```

Contactes: <http://www.javierms.com/cordecor/index.php/rest/contactes>

```
{"status":200,"message":"OK","result":[{"id":"1","usuari":"2","foto":"contacte_u2_1.jpg","nom":"Marc","cognoms":"Puig","direccio":"Via Laietana, 2, 1-2","ciutat":"Barcelona","telefon":"555000231","email":"marcpuig@testemail.com"}]}
```

Productes: <http://www.javierms.com/cordecor/index.php/rest/productes>

```
{"status":200,"message":"OK","result":[{"id":"1","foto":"producte_1.jpg","nom":"Gerro","codi":"p0001","descripcio":"Gerro de cer\u00e0mica","preu":"59.99"}, {"id":"2","foto":"producte_2.jpg","nom":"Kitllar","codi":"p0002","descripcio":"Kit b\u00e0sic de decoraci\u00f3","preu":"40.00"}]}
```

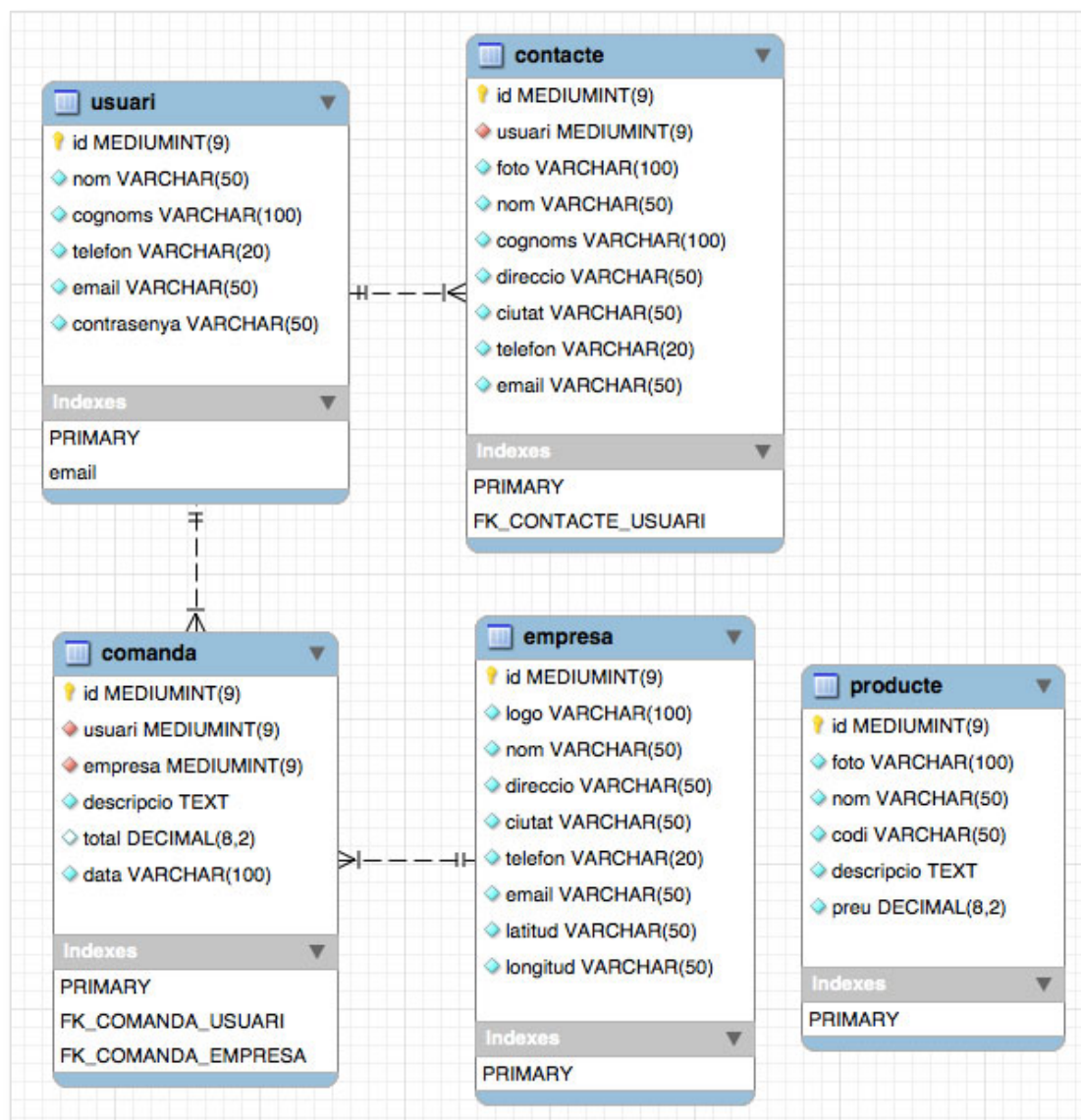
Comandes: <http://www.javierms.com/cordecor/index.php/rest/comandes>

```
{"status":200,"message":"OK","result":{"id":"1","usuari":"2","empresa":"2","descripcio":"- (2) Kitllar [p0002]\r\n- (1) Catifa [p0005]\r\n- (3) Coful [p0006]\r\n","total":"234.90","data":"20-05-2014 23:52:08"}}
```

7.3. Disseny relacional de la Base de Dades

A continuació veurem un esquema de la base de dades MySQL del servidor mostrant les taules que s'utilitzen per donar servei a les peticions REST que utilitza l'aplicació Android.

La base de dades SQLite d'Android seguirà la mateixa estructura, tot i que tenim una taula extra anomenada `loggedUser`, que utilitzem únicament per guardar el id de l'últim usuari que va iniciar sessió al dispositiu.



En el següent arxiu de l'aplicació web es configura l'accés a la base de dades del servidor: `cordecor/application/config/database.php`

```

45 $db['default']['hostname'] = 'localhost';
46 $db['default']['username'] = 'username';
47 $db['default']['password'] = 'password';
48 $db['default']['database'] = 'cordecor';
49 $db['default']['dbdriver'] = 'mysql';
50 $db['default']['dbprefix'] = '';
51 $db['default']['pconnect'] = TRUE;
52 $db['default']['db_debug'] = TRUE;
53 $db['default']['cache_on'] = FALSE;
54 $db['default']['cachedir'] = '';
55 $db['default']['char_set'] = 'utf8';
56 $db['default']['dbcollat'] = 'utf8_general_ci';
57 $db['default']['swap_pre'] = '';
58 $db['default']['autoinit'] = TRUE;
59 $db['default']['stricton'] = FALSE;

```

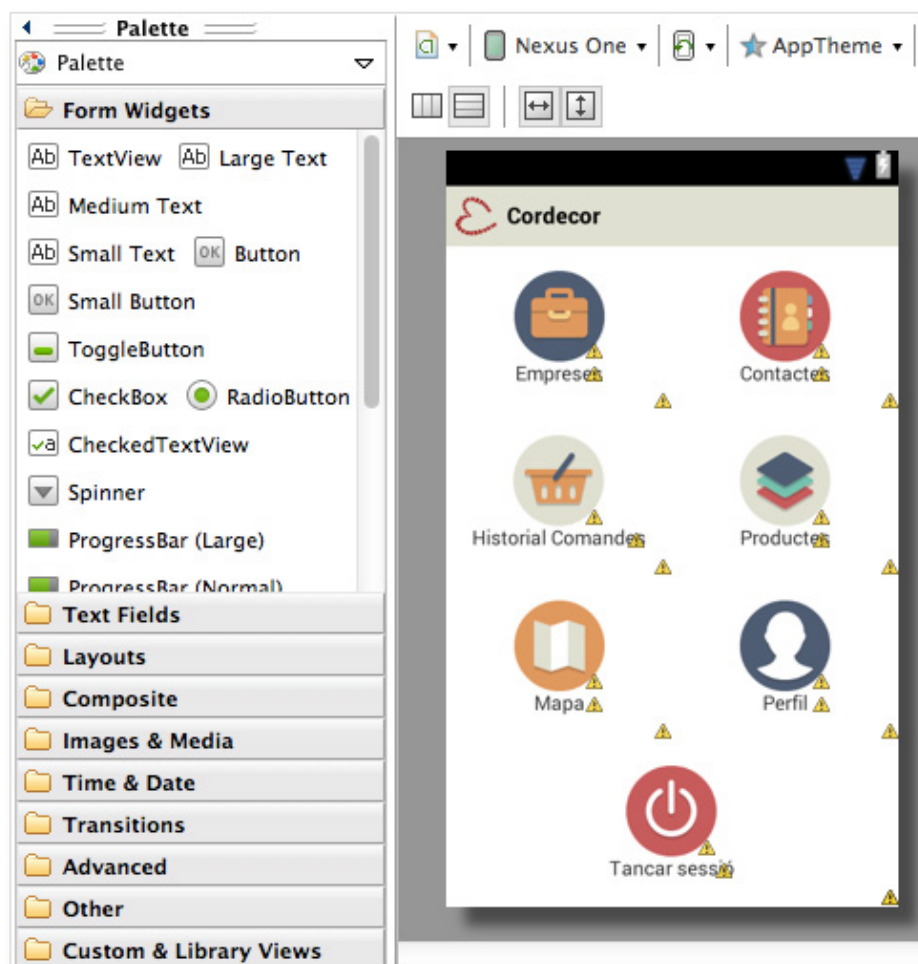
8. Implementació

S'ha intentat fer una aplicació Android amb varietat de serveis que ajudin a l'usuari en el seu treball diari, però sempre buscant una navegació senzilla i una aparença visualment atractiva que millori l'experiència d'usuari.

8.1. Interfície

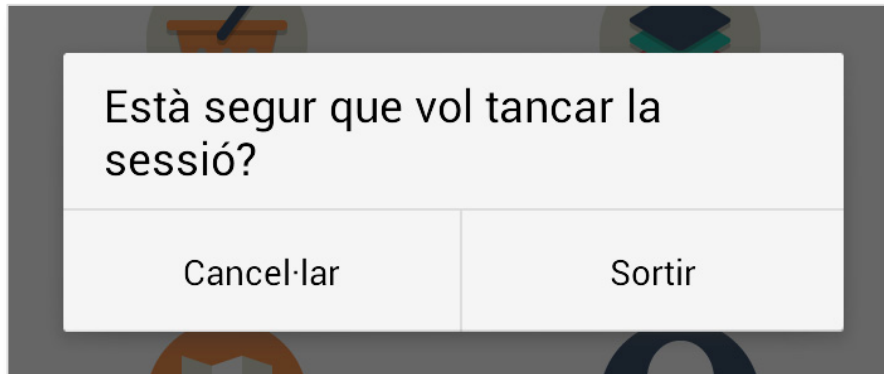
Per desenvolupar els layouts de cada Activity s'ha utilitzat el propi editor d'Android en Eclipse, alternant entre l'eina gràfica que podem veure en la captura i l'edició directa de codi xml.

<http://www.eclipse.org/>



Dialog

Durant l'execució de l'aplicació hi ha moments en els quals es mostra a l'usuari dialogs que ajudin en la presa de decisions per part d'aquest:

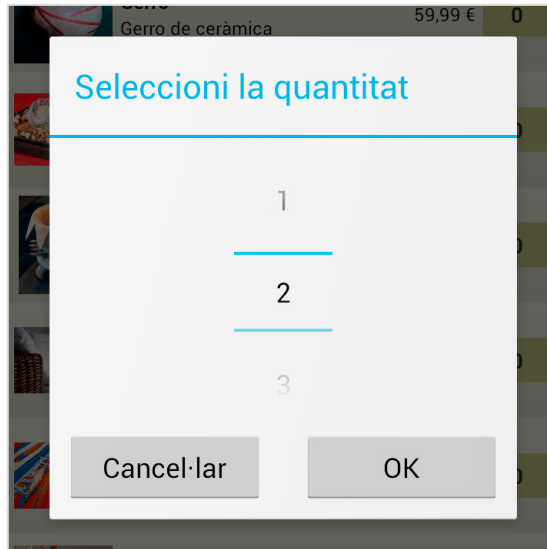


El següent mètode de la classe Utils.java crea i configura el dialog que acabem de veure:

```
public static void exitDialog(Context context, String message) {
    Log.i("Utils.exitDialog()");
    try{
        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setCancelable(false)
            .setMessage(message)
            .setNegativeButton(context.getString(R.string.btn_sortir),
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        exit();
                    }
                });
        AlertDialog dialog = builder.create();
        dialog.setCancelable(false);
        dialog.show();
    }
    catch(Exception e){
        Log.e("/error:" + e.toString());
    }
}
```

NumberPicker

En un d'aquests dialogs inserim el widget NumberPicker que ajuda l'usuari en la selecció de la quantitat de cada producte que volem assignar a una nova comanda:



```
private void showDialogQuantity(final EntityHolder holder, final int position){
    final Dialog d = new Dialog(this);
    d.setTitle(getString(R.string.producte_dialog_title));
    d setContentView(R.layout.dialog_quantity);

    Button btnCancelar = (Button) d.findViewById(R.id.btnCancelar);
    Button btnOK = (Button) d.findViewById(R.id.btnOK);

    final NumberPicker np = (NumberPicker) d.findViewById(R.id.numberPicker);
    np.setMaxValue(100); // valor màxim 100
    np.setMinValue(0); // valor mínim 0
    np.setWrapSelectorWheel(false);

    btnCancelar.setOnClickListener(new Button.OnClickListener() {
        @Override
        public void onClick(View v) {
            d.dismiss();
        }
    });
    btnOK.setOnClickListener(new Button.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(adapter != null){
                holder.quantitat = np.getValue();
                adapter.setItem(holder, position);

                // Associem l'id del producte amb la quantitat seleccionada
                quantitats.put(holder.producte.getId(), holder.quantitat);
            }

            d.dismiss();
        }
    });
    d.show();
}
```

ListView

Un dels components més utilitzats en aquesta aplicació és ListView. Aquesta classe ens permet mostrar un llistat d'elements, a cada fila de la llista mostrarem les dades més representatives de cada entitat, per exemple en la pantalla de productes cada fila mostra la foto del producte, el seu nom, descripció i el preu d'aquest.



A la Activity l'arxiu `ProducteLlistatActivity.java` tenim el mètode `configListView()`, en el qual vam crear un `ArrayList` que contindrà cadascuna de les entitats que es mostraran. Aquesta llista s'assigna a una instància de la classe `RowAdapter` i al seu torn aquest adaptador s'assigna al nostre `ListView`.

La classe `RowAdapter`, que es defineix en aquest mateix arxiu `java`, és una subclasse de `ArrayAdapter`, classe que converteix els elements de l'array en objectes visuals per a ser carregats en el contenidor `ListView`.

`RowAdapter` conté la personalització que necessitem per mostrar les files com desitgem, utilitza el layout `R.layout.activity_producto_llistat_fila` que conté els elements visuals que veurem a cada fila.

Aquest layout conté un `LinearLayout` que organitza els seus elements interns horitzontalment. A l'esquerra tenim un `ImageView` (la foto del producte), i a la dreta un altre `LinearLayout` que conté a la vegada els tres `TextView`, nom del producte, descripció i preu:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:orientation="horizontal" android:padding="5dp"
    android:background="@drawable/llistat_fila_background2" >
    <ImageView android:id="@+id/image"
        android:layout_width="64dp"
        android:layout_height="64dp" >
    </ImageView>
    <LinearLayout
        android:background="@color/transparent"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_gravity="center_vertical"
        android:orientation="vertical"
        android:padding="5dp">
        <TextView android:id="@+id/textTop"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/text"
            android:textStyle="bold"
            android:textSize="13sp" />
        <TextView android:id="@+id/textDown"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/text"
            android:textSize="12sp" />
        <TextView android:id="@+id/textRight"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/text"
            android:textSize="12sp" />
    </LinearLayout>
</LinearLayout>
```

8.2. Entitats

Cadascuna de les taules de la base de dades que vam veure a l'apartat 7.3 té un equivalent com a objecte en el codi del projecte. Les classes que representen cada entitat són: EntityComanda, EntityContacte, EntityEmpresa, EntityProducte i EntityUsuari.

En cadascuna d'aquestes Entity agrupem tot una sèrie d'atributs i mètodes que ens ajudaran en la lògica de l'aplicació. Aquestes classes implementen la interfície Parcelable, això ens permetrà passar instàncies d'aquestes classes entre activitats.

També tenen mètodes per parsejar els JSON que utilitzarem en la comunicació amb el servidor.

Finalment contenen atributs i mètodes per a les consultes SQL de la base de dades:

```

EntityEmpresa.java
244 public static String sqlCreateTable(){
245     String sql = "CREATE TABLE "+TABLE_NAME+" ("
246         + COLUMN_ID + " INTEGER NOT NULL, "
247         + COLUMN_NOM + " TEXT, "
248         + COLUMN_DIRECCIO + " TEXT, "
249         + COLUMN_CIUTAT + " TEXT, "
250         + COLUMN_TELEFON + " TEXT, "
251         + COLUMN_EMAIL + " TEXT, "
252         + COLUMN_LOGO + " TEXT, "
253         + COLUMN_LATITUD + " REAL, "
254         + COLUMN_LONGITUD + " REAL, "
255         + "PRIMARY KEY (" + COLUMN_ID + ")");
256     return sql;
257 }
258
259 public static String sqlDropTable(){
260     return "DROP TABLE IF EXISTS "+TABLE_NAME;
261 }
262

```

Els mètodes `sqlCreateTable()` i `sqlDropTable()` de cada Entity els executarem a la classe `DataSqlite`, classe encarregada de configurar la base de dades SQLite:

```

DataSqlite.java
23 @Override
24 public void onCreate(SQLiteDatabase db) {
25     Log.i("DataSqlite.onCreate()");
26
27     // create all tables
28     db.execSQL(EntityComanda.sqlCreateTable());
29     db.execSQL(EntityContacte.sqlCreateTable());
30     db.execSQL(EntityEmpresa.sqlCreateTable());
31     db.execSQL(EntityProducte.sqlCreateTable());
32     db.execSQL(EntityUsuari.sqlCreateTable());
33     db.execSQL("CREATE TABLE "+LOGGEDUSER_TABLE_NAME
34         +"("+LOGGEDUSER_COLUMN_ID+" INTEGER)");
35 }

```

La classe `DataManager` és l'encarregada d'agrupar els mètodes que utilitzarem des de les activitats per recuperar les dades de la base de dades o modificar-les:

```

DataManager.java
86 public SparseArray<EntityEmpresa> getEmpreses(){
87     EntityUsuari user = getUsuari();
88     if(DataRuntime.getEmpreses()==null && user!=null){
89         SQLiteDatabase database = openDatabase();
90         DataRuntime.setEmpreses( EntityEmpresa.databaseGetAll(database) );
91         closeDatabase();
92     }
93     return DataRuntime.getEmpreses();
94 }

```

8.3. Comunicació

La classe `HttpManager` és l'encarregada d'agrupar els mètodes que utilitzarem des de les activitats per comunicar-nos amb el servidor. Cada mètode configura les URL que utilitzarem en les peticions REST que vam veure en l'apartat 7.2.2:

```

HttpManager.java
83 public static SparseArray<EntityEmpresa> getEmpreses() {
84     Log.i("HttpManager.getEmpreses()");
85     EntityUsuari user = DataManager.getInstance().getUsuari();
86     SparseArray<EntityEmpresa> data = null;
87
88     if(user != null){
89         SparseArray<EntityEmpresa> recover = DataManager.getInstance().getEmpreses();
90         boolean syncMode = (recover!=null && recover.size(>0) ? true : false;
91
92         String url = (syncMode) ? REST_URL + "empresesSync" : REST_URL + "empreses";
93         String responseStr = httpGet(url);
94         Log.i("/response: "+responseStr);
95
96         RestResponse response = RestResponse.parseJSON(responseStr);
97         if(response!=null && !response.hasError()){
98             if(response.isOkEmpty()){
99                 data = new SparseArray<EntityEmpresa>();
100             }
101             else{
102                 if(syncMode){
103                     data = EntityEmpresa.parseHttpGetSync(responseStr);
104                 }
105                 else{
106                     data = EntityEmpresa.parseHttpGetAll(responseStr);
107                 }
108             }
109         }
110     }
111     if(data == null) Log.i("/error");
112
113     return data;
114 }

```

`HttpManager` utilitza internament els mètodes de la classe `HttpMethods`. Aquesta classe conté els mètodes que realitzen les peticions HTTP:

```

public static String httpGet(String url) {
    EntityUsuari user = DataManager.getInstance().getUsuari();
    String email = (user != null) ? user.getEmail() : "";
    String pwd = (user != null) ? user.getContrasenya() : "";

    return HttpMethods.doGet(url, email, pwd);
}

```

En l'inici d'aquesta classe configurem la instància de `DefaultHttpClient` que utilitzem per realitzar cada petició HTTP:


```

HttpMethods.java
40     private static final DefaultHttpClient httpClient;
41     static {
42         final HttpParams params = new BasicHttpParams();
43         HttpProtocolParams.setVersion(params, HttpVersion.HTTP_1_1);
44
45         ConnManagerParams.setTimeout(params, HTTP_TIMEOUT);
46         HttpConnectionParams.setConnectionTimeout(params, HTTP_TIMEOUT);
47         HttpConnectionParams.setSoTimeout(params, HTTP_TIMEOUT);
48         HttpConnectionParams.setStaleCheckingEnabled(params, false);
49         HttpConnectionParams.setSocketBufferSize(params, 8192);
50
51         HttpClientParams.setRedirecting(params, false);
52
53         SchemeRegistry schemeRegistry = new SchemeRegistry();
54         schemeRegistry.register(new Scheme("http",
55             PlainSocketFactory.getSocketFactory(), 80));
56         schemeRegistry.register(new Scheme("https",
57             SSLSocketFactory.getSocketFactory(), 443));
58
59         ClientConnectionManager manager =
60             new ThreadSafeClientConnManager(params, schemeRegistry);
61         httpClient = new DefaultHttpClient(manager, params);
62     }

```

En el mètode doRequest() configurem la capçalera de cada petició HTTP afegint el username i password de l'usuari, així se'ns permetrà l'accés a les dades del servidor:

```

HttpMethods.java
134     private static String doRequest(HttpUriRequest request, String username, String password){
135         try {
136             request.setHeader("Authorization", "Basic " +
137                 Base64.encodeToString((username+":"+password).getBytes(), Base64.NO_WRAP));
138
139             HttpEntity entity = executeRequest(httpClient, request, 0);
140             if(entity != null) {
141                 return EntityUtils.toString(entity, UTF_8);
142             }
143         }
144         catch(Exception e) {
145             request.abort();
146             Log.e("HttpMethods.doPostRequest() /Exception: "+e.toString());
147         }
148         return null;
149     }

```

Sense aquesta capçalera la resposta que rebríem del servidor és la que vam veure en l'apartat 7.2.2:

```
{"status":401,"message":"Unauthorized","result":""}
```

8.4. Threads

Perquè una aplicació sigui àgil i dinàmica, de vegades hem de realitzar tasques en segon pla, com pot ser la descàrrega d'informació del servidor o l'accés a la base de dades local. De vegades, també hem de realitzar processos que ens demanen si o si que ho fem en un fil secundari al principal. Per a tot això ens podem ajudar de la classe AsyncTask:

```

Utils.java
183 public static void executeAsync(final AsyncListener listener){
184     if(listener != null){
185         new AsyncTask<Void, Void, Boolean>(){
186             @Override
187             protected Boolean doInBackground(Void... params) {
188                 boolean ok;
189                 try{ ok = listener.runBackground(); }
190                 catch(Exception e){ ok = false; }
191                 return ok;
192             }
193             @Override
194             protected void onPostExecute(Boolean param){
195                 boolean ok = param.booleanValue();
196                 try{ listener.runPost(ok); }
197                 catch(Exception e){
198                     Log.e("Utils.executeAsync() /runPost error: "
199                         +e.toString());
200                 }
201             }
202         }.execute();
203     }
204 }
205
206 public interface AsyncListener {
207     public boolean runBackground();
208     public void runPost(boolean ok);
209 }
210 }

```

En aquest projecte hem creat un mètode genèric a la classe Utils.java per utilitzar AsyncTask. Li passarem com a paràmetre la interfície Utils.AsyncListener que podem veure com s'utilitza en el següent exemple:

```

EmpresaListatActivity.java
74 // Recuperem les dades per omplir el listview
75 Utils.executeAsync(new Utils.AsyncListener() {
76     @Override
77     public boolean runBackground() {
78         empreses = DataManager.getInstance().getEmpreses();
79         return (empreses != null) ? true : false;
80     }
81
82     @Override
83     public void runPost(boolean ok) {
84         configListView();
85         configInputSearch();
86     }
87 });
88 }
89

```

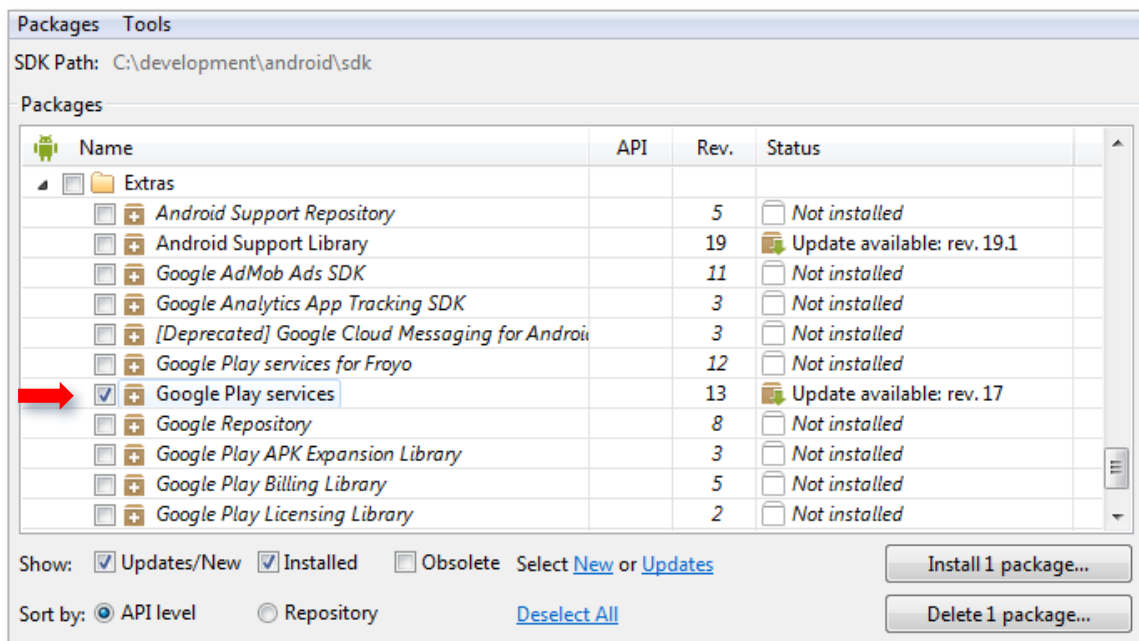
8.5. Configuració de Google Maps v2

A l'aplicació tenim una secció en la que es visualitza un mapa amb punts d'interès, punts que representen la posició de les nostres empreses clients.

A continuació es detallen els passos que s'han dut a terme per a poder configurar la versió 2 de Google Maps en l'aplicació Android.

1. Descàrrega de *Google Play Services*

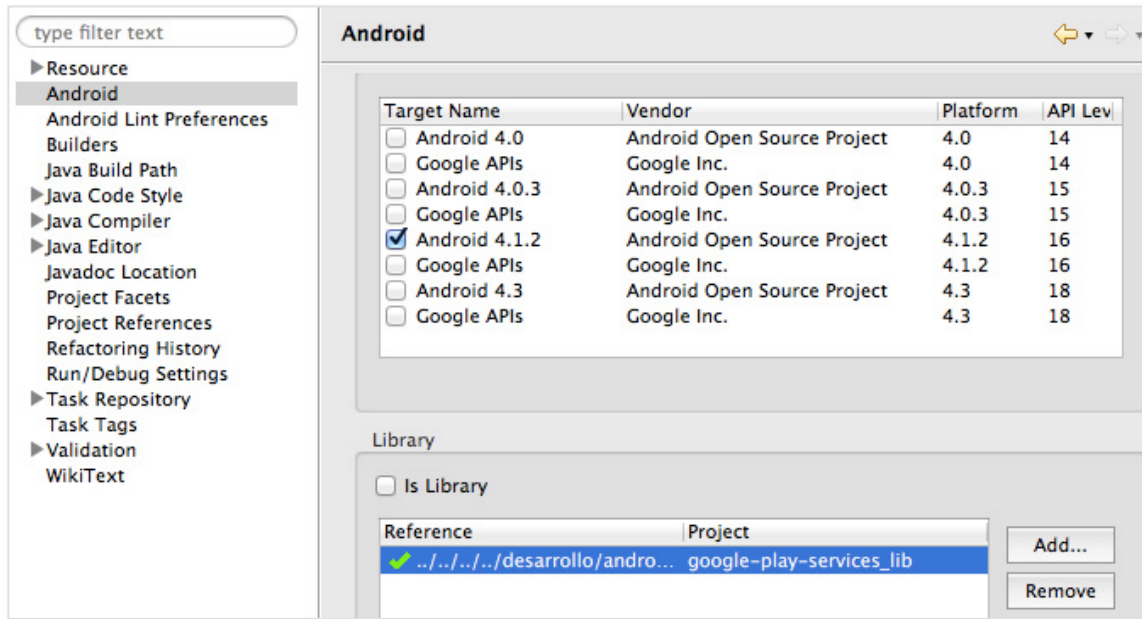
Per poder treballar amb l'API de Google Maps instal·lem Google Play Services accedint a l'Android SDK Manager des Eclipse, seleccionem la casella que veiem en la captura:



2. Importació de *Google Play Services* en Eclipse

Un cop descarregat importem en Eclipse el projecte que hem de tenir a la carpeta: android-sdk/extras/google/google_play_services/libproject/google-play-services_lib

Ara ja es pot accedir a les propietats del projecte Cordecor i afegir la referència a google-play-services_lib:



3. Obtenir la clau de la API de Google Maps

Necessitem generar el valor SHA-1 que utilitzarem posteriorment. Per a això utilitzem l'eina keytool. Obrim el terminal i anem a la carpeta on tenim l'arxiu cordecor.keystore. Aquest fitxer és a la carpeta client del lliurament del projecte i ho tornarem a utilitzar posteriorment per exportar l'aplicació a un arxiu APK.

Executem la següent ordre en el terminal:

```
keytool -list -v -keystore cordecor.keystore -alias cordecor -storepass cordecor -keypass cordecor
```

I obtenim les següents línies:

```
Nombre de alias: cordecor
Fecha de creaci?n: 21-may-2014
Tipo de entrada: PrivateKeyEntry
Longitud de la cadena de certificado: 1
Certificado[1]:
Propietario: CN=Cordecor, O=Cordecor, L=Barcelona
Emisor: CN=Cordecor, O=Cordecor, L=Barcelona
N?mero de serie: 537d0997
V?lido desde: Wed May 21 22:16:23 CEST 2014 hasta: Sun May 15 22:16:23
CEST 2039
Huellas digitales del certificado:
MD5: 4B:6E:17:71:0D:77:0E:94:34:64:D8:A0:8F:59:32:C1
SHA1: 5B:FC:85:14:74:75:90:ED:E2:14:7D:DE:54:39:CF:B0:5D:0B:8F:BB
Nombre del algoritmo de firma: SHA1withRSA
Versi?n: 3
```

Un cop tenim aquest valor ja podem accedir a la consola de Google API:
<https://console.developers.google.com>

Després de seleccionar el servei de Google Maps i crear el projecte de cordecor, accedim a APIs & auth ⇒ credentials i creem una nova Android key accedint a Public API Access.

En aquesta pantalla utilitzarem l'anterior valor SHA1 i el package de l'aplicació: cat.cordecor.crm

Finalment s'ha obtingut l'API key que utilitzem en la nostra aplicació:

API key	AlzaSyAcIDrIXatThGiWLhiP9ZoOPYcxfD9YtKM
Android applications	5B:FC:85:14:74:75:90:ED:E2:14:7D:DE:54:39:CF:B0:5D:0B:8F:BB ;cat.cordecor.crm
Activation date	May 21, 2014 1:22 PM
Activated by	jmarsan78@gmail.com (you)

4. Configuració de AndroidManifest.xml

Afegim l'API key que hem obtingut anteriorment a l'arxiu AndroidManifest.xml del projecte, també els permissions i features que necessita Google Maps. Finalment tindrem a AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />

<application
    android:name="cat.cordecor.crm.App"
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <!-- Google Maps API Key -->
    <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyAcLDrlXatThGiWLhiP9ZoOPYcxFD9YtKM" />
    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
```

8.6. Exportació de l'aplicació

Finalment s'ha exportat l'aplicació a un arxiu APK des d'Eclipse. L'aplicació s'ha de signar amb el seu propi certificat, utilitzem l'arxiu cordecor.keystore que ja tenim generat a la carpeta client, del lliurament del projecte.

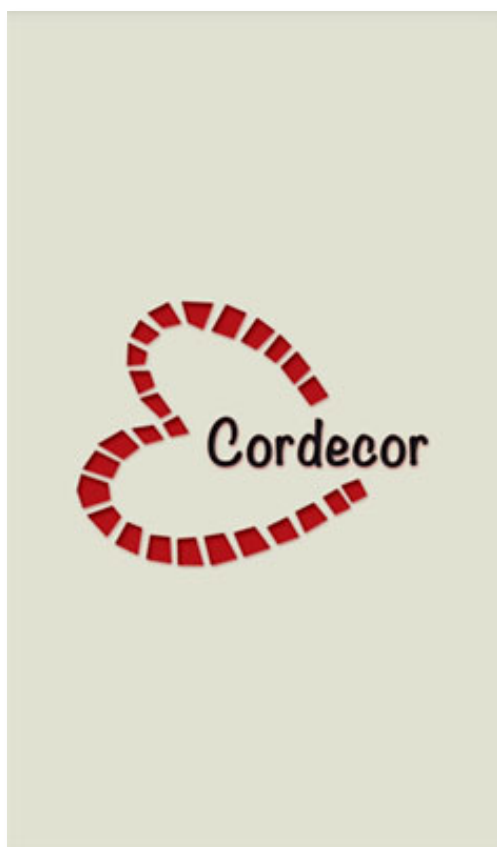
1. Premem amb el botó dret del ratolí sobre el projecte Android i seleccionem l'opció Export... ⇒ Export Android Application.
2. Seleccionem el projecte a exportar: cordecor
3. Seleccionem "Use existing keystore" i anem a la ubicació del nostre arxiu cordecor.keystore
4. Escrivim la contrasenya: cordecor
5. Seleccionem l'àlies cordecor i escrivim la contrasenya: cordecor
6. Finalment escollim la ubicació del fitxer apk que vá a generar i premem sobre Finalitzar.

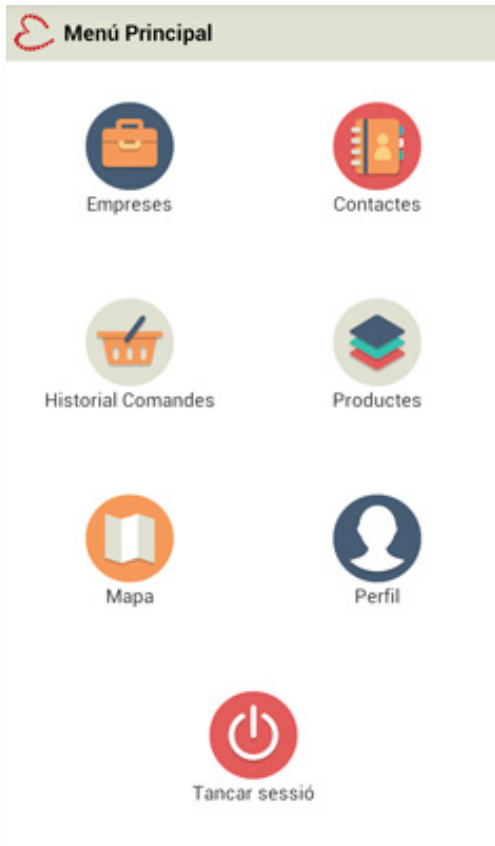
9. Manual d'usuari

Manual d'usuari de l'aplicació per als comercials de Cordecor.

9.1. Iniciar sessió

Introducció de l'email i la contrasenya per iniciar sessió com a usuari de Cordecor.

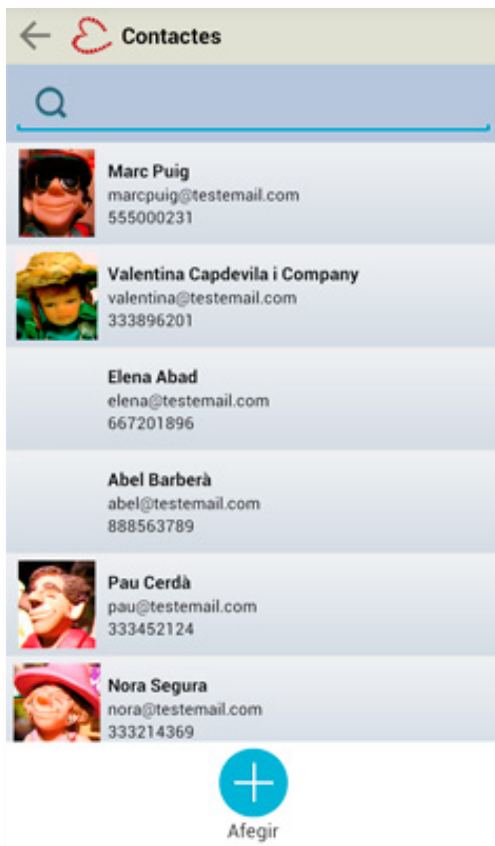
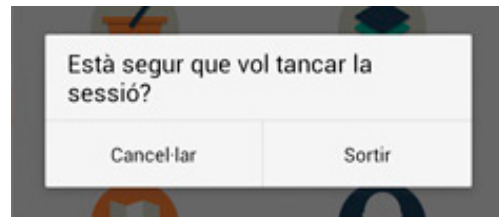




9.2. Menú principal

Des del menú principal l'usuari pot accedir a les seccions: Empreses, Contactes, Productes, Comandes, Mapa i Perfil.

Prement el botó Tancar sessió s'obrirà un dialog que li preguntarà si realment vol tancar la sessió.



9.3. Llistat de contactes

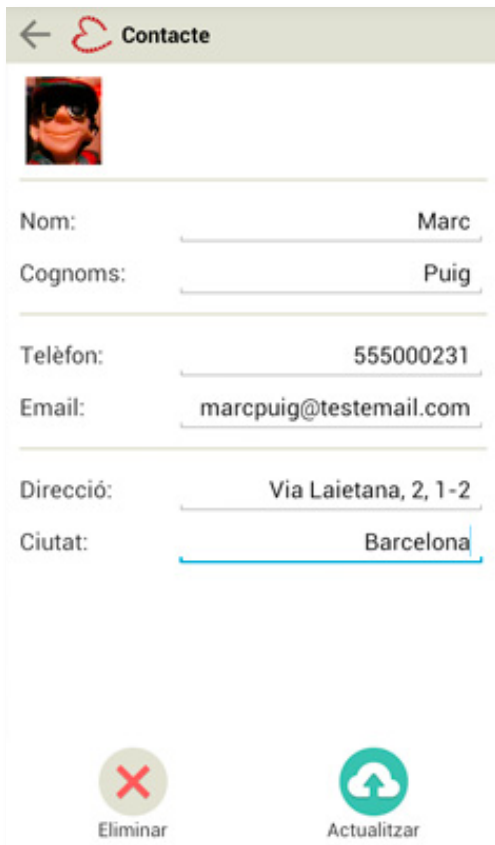
Recerca de contactes escrivint en el camp de text superior.

L'usuari pot afegir un nou contacte prement el botó inferior Afegir.

Si l'usuari prem una fila en concret accedirà a la pantalla d'edició d'aquest contacte.

Navegació

Sempre que estem en una pantalla que no sigui el menú principal ens apareixerà una fletxa, a la part superior esquerra de la pantalla, que ens permetrà tornar a la pantalla anterior.



9.4. Edició d'un contacte

L'usuari pot editar el nom, cognoms, telèfon, direcció i ciutat del contacte.

Pot actualitzar les dades o eliminar el contacte amb els botons inferiors.



9.5. Llistat de productes

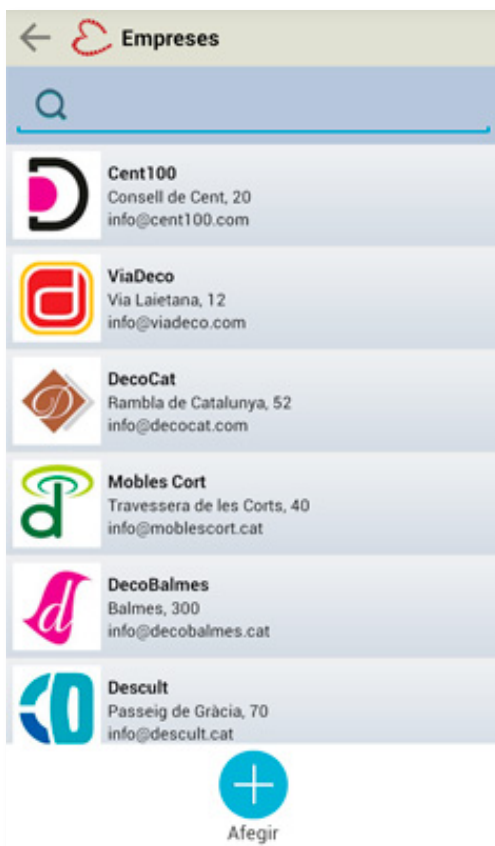
Recerca de productes escrivint en el camp de text superior.

Si l'usuari prem una fila en concret accedirà a la pantalla de detall d'aquest producte.



9.6. Pantalla detall d'un producte

A la pantalla de detall d'un producte concret podem veure la imatge d'aquest, el nom, el codi, la descripció i el preu.

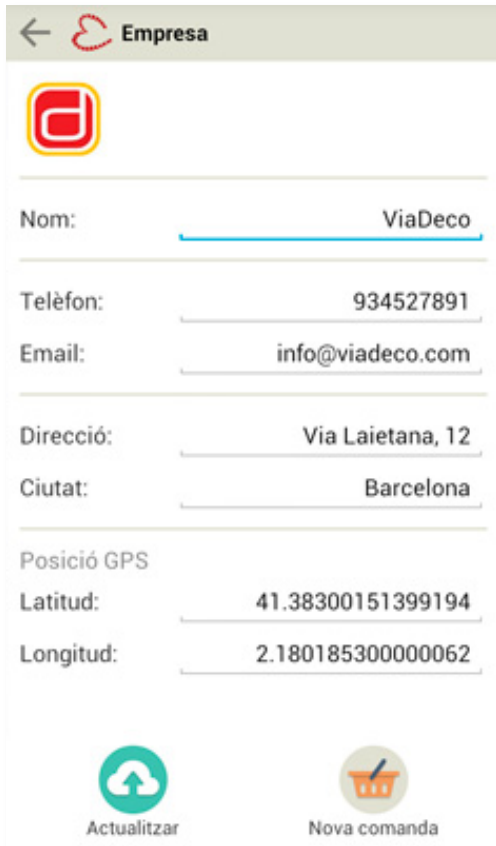


9.7. Llistat d'empreses


Recerca d'empreses escrivint en el camp de text superior. Segons introdueix lletres buscarà les coincidències amb nom o adreça de totes les empreses.

L'usuari pot afegir una nova empresa prement el botó inferior Afegir.

Cada fila del llistat mostra un resum de dades de cada empresa. Si l'usuari prem una en concret accedirà a la pantalla d'edició d'aquesta empresa.



← Empresa



Nom:

Telèfon:

Email:



Direcció:

Ciutat:

Posició GPS

Latitud:

Longitud:

 Actualitzar  Nova comanda

9.8. Edició d'una empresa

L'usuari pot editar el nom d'empresa, el telèfon, l'email, la direcció i la ciutat.

A més podem afegir la latitud i longitud, valors que serviran per posicionar l'empresa en el mapa.

Nova comanda

Per realitzar una nova comanda associada a aquesta empresa l'usuari ha de prémer el botó inferior Nova comanda.



← Comanda

 ViaDeco

 Productes

Descripció:

- (2) Kitllar [p0002]
- (1) Catifa [p0005]
- (4) Coful [p0006]
- (1) Butaca [p0007]

Total: 374,90 €

 Enviar Comanda

9.9. Realitzar una nova comanda

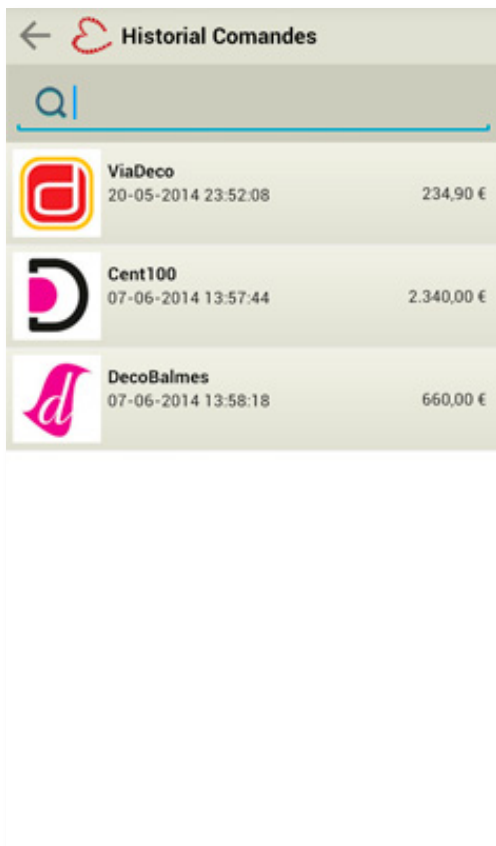
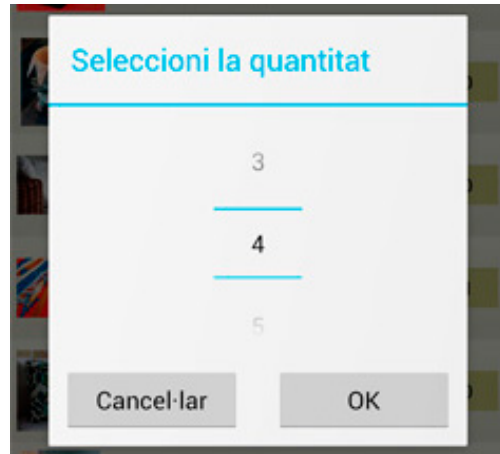
Per realitzar una comanda l'usuari ha de prémer sobre el botó Productes per seleccionar les quantitats d'aquests.

Un cop seleccionats l'usuari pot veure el preu total de la comanda i prémer sobre el botó Actualitzar per enviar les dades al servidor.



9.10. Selecció de productes

Aquesta pantalla és com l'anterior del llistat de productes, però al prémer sobre una fila s'obrirà un dialog en el qual podrà seleccionar la quantitat d'aquest producte que es vol afegir a la comanda.



9.11. Historial de comandes

Recerca de comandes escrivint en el camp de text superior. Podem buscar per nombre d'empresa.

Si l'usuari prem una fila en concret accedirà a la pantalla de detall d'aquesta comanda que va realitzar anteriorment.

Veurem una pantalla com la de l'apartat 9.9 però amb el botó productes ocult i sense la possibilitat de canviar els valors.



← Perfil

Nom: Javier

Cognoms: Martínez

Telèfon: 222124896

Email: javier@cordecor.cat

Contrasenya: *****

Actualitzar

9.12. Perfil

L'usuari pot canviar el seu nom, cognoms i telèfon.

L'usuari pot veure l'email que utilitza per accedir per iniciar sessió però no pot canviar-lo, ja que és l'identificador del compte.

Prement el botó inferior Actualitzar, s'enviaran les dades al servidor per actualitzar les dades, així com a la base de dades del mòbil.



9.13. Mapa

A la pantalla de Mapa l'usuari pot veure la localització de les empreses representades com a punts d'interès sobre el mapa.

10. Conclusions

Durant el desenvolupament d'aquest projecte, s'ha creat amb èxit una aplicació per a smartphones Android amb codi natiu Java i una arquitectura RESTful al servidor amb PHP i MySQL. Aquestes fites han permès aconseguir els objectius que es van exposar al començament d'aquesta memòria.

Atès que és la primera versió de l'aplicació, a priori no es perceben millores. No obstant això, amb l'ús diari de l'aplicació no hi ha dubte que apareixeran nous camps i dades a introduir que cobreixin totes les necessitats d'empresa-comercial-client.

Personalment estic content pel que ha suposat realitzar un projecte relacionat amb les tecnologies mòbils, ja que m'ha servit per seguir aprenent aspectes importants del desenvolupament de programari que podré aplicar a la meua carrera professional.

10.1. Fonts d'informació

Per al desenvolupament d'aquest projecte, s'han consultat les següents fonts d'informació:

Developer Android

<http://developer.android.com/index.html>

<http://developer.android.com/reference/packages.html>

<http://developer.android.com/guide/practices/index.html>

Altres enllaços d'interès

<http://stackoverflow.com/questions/tagged/android>

<http://code.tutsplus.com/tutorials/android-user-interface-design-building-a-listview-application--mobile-5195>

<http://www.vogella.com/tutorials/AndroidBackgroundProcessing/article.html>

<http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/>

<http://code.tutsplus.com/tutorials/working-with-restful-services-in-codeigniter--net-8814>