

---

# **OpenERP Mobile Documentation**

***Release 1.0***

**OpenERP SA**

May 12, 2015

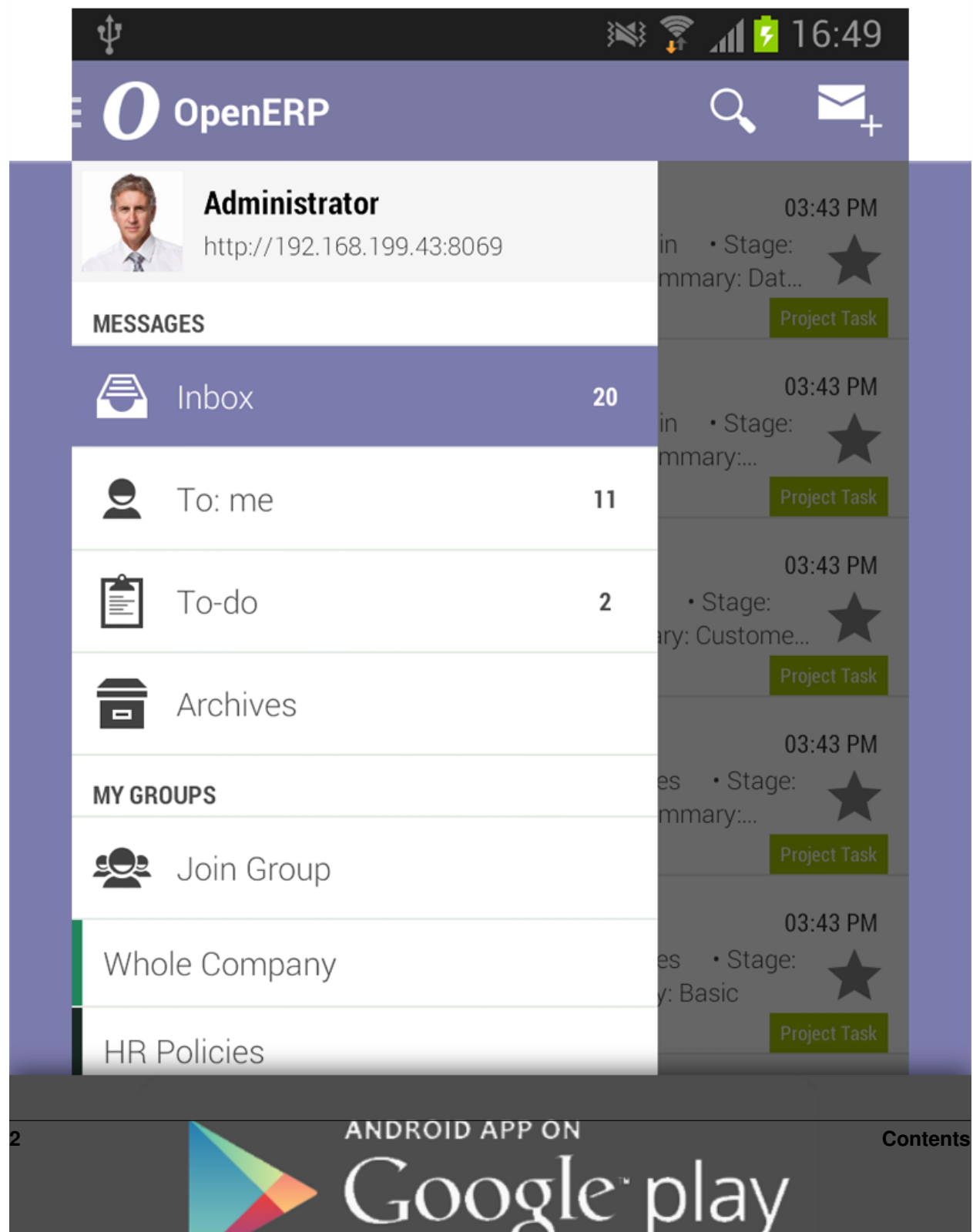


<b>1</b>	<b>Table of Contents</b>	<b>3</b>
1.1	About OpenERP Mobile Framework . . . . .	3
1.2	Setting Up Development Environment . . . . .	3
1.3	Getting started with OpenERP mobile framework . . . . .	11
1.4	OpenERP Android Framework Development . . . . .	31
<b>2</b>	<b>Indices and tables</b>	<b>49</b>





# OpenERP Mobile



---

## Table of Contents

---

### 1.1 About OpenERP Mobile Framework

OpenERP is a powerful open source framework. With help of this framework we can rapidly develop almost any application.

World is contracting with the growth of mobile phone technology. As the number of users is increasing day by day, facilities are also increasing. Now a days mobiles are not used just for making calls but they have innumerable uses and can be used as a Camera , Music player, Tablet PC, T.V. , Web browser etc. And with the new technologies, new software and operating systems are required.

One of the most widely used mobile OS these days is ANDROID. Android is a software bunch comprising not only operating system but also middleware and key applications.

**OpenERP Android framework** is an open source mobile application development framework with OpenERP integration. With the help of mobile framework we can rapidly develop almost all OpenERP supported application as faster as we can develop in OpenERP Framework.

This framework contains its own ORM to handle mobile's local database. So you do not have to worry about data coming from OpenERP Server. It has pre-developed services and providers to make your application data synchronized with OpenERP.

### 1.2 Setting Up Development Environment

#### 1.2.1 Downloading and Installing

##### Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

It is recommend to download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in ADT (Android Developer Tools) to streamline your Android app development.

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

## Installing the Eclipse Plugin

Android offers a custom plugin for the Eclipse IDE, called Android Development Tools (ADT). This plugin provides a powerful, integrated environment in which to develop Android apps. It extends the capabilities of Eclipse to let you quickly set up new Android projects, build an app UI, debug your app, and export signed (or unsigned) app packages (APKs) for distribution.

If you need to install Eclipse, you can download it from [eclipse.org/downloads/](http://eclipse.org/downloads/).

---

**Note:** If you prefer to work in a different IDE, you do not need to install Eclipse or ADT. Instead, you can directly use the SDK tools to build and debug your application.

---

## Download the ADT Plugin

- Start Eclipse, then select Help > Install New Software.
- Click Add, in the top-right corner.
- In the Add Repository dialog that appears, enter “ADT Plugin” for the Name and the following URL for the Location: <https://dl-ssl.google.com/android/eclipse/>
- Click OK. If you have trouble acquiring the plugin, try using “http” in the Location URL, instead of “https” (https is preferred for security reasons).
- In the Available Software dialog, select the checkbox next to Developer Tools and click Next.
- In the next window, you’ll see a list of the tools to be downloaded. Click Next.
- Read and accept the license agreements, then click Finish. If you get a security warning saying that the authenticity or validity of the software can’t be established, click OK.
- When the installation completes, restart Eclipse.

## Configure the ADT Plugin

- Once Eclipse restarts, you must specify the location of your Android SDK directory:
- In the “Welcome to Android Development” window that appears, select Use existing SDKs.
- Browse and select the location of the Android SDK directory you recently downloaded and unpacked.
- Click Next.

Your Eclipse IDE is now set up to develop Android apps, but you need to add the latest SDK platform tools and an Android platform to your environment.

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

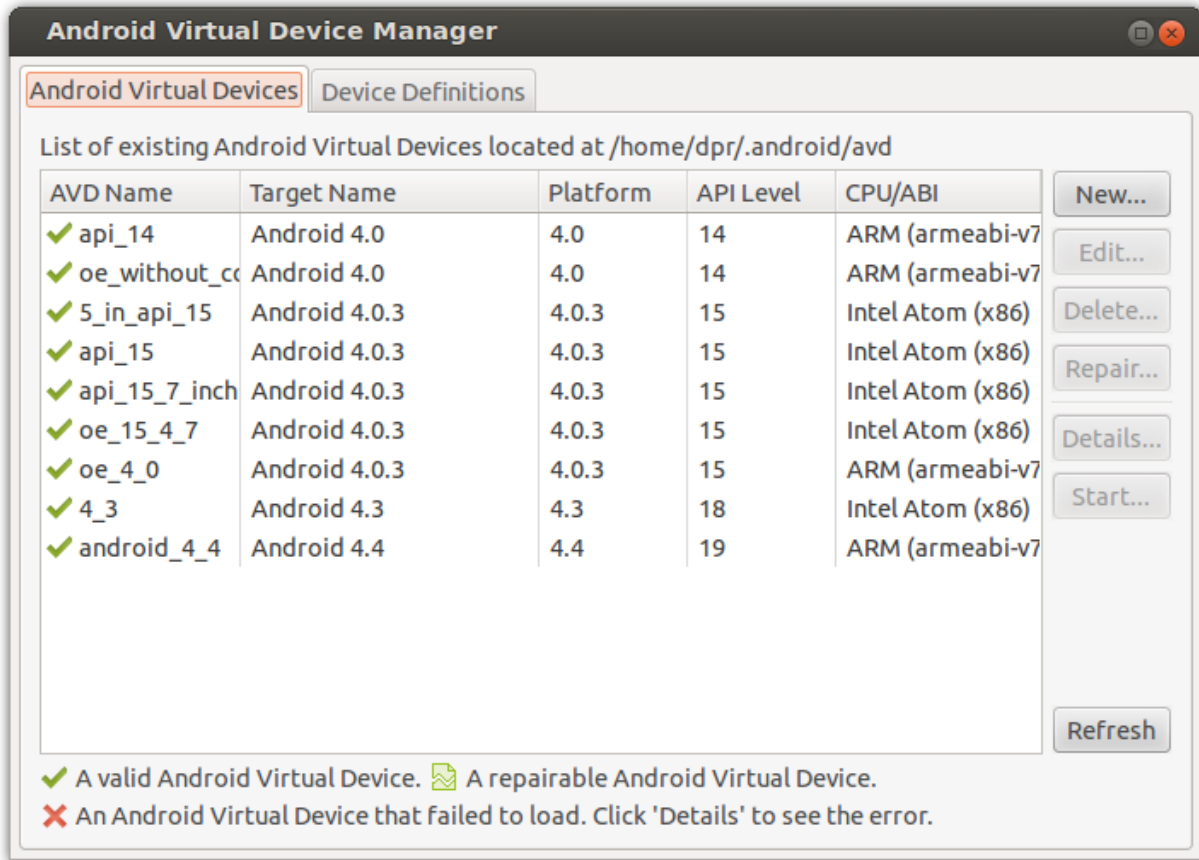
## 1.2.2 Configure virtual device

To test your application android SDK provide virtual devices with different configuration and Resolutions. Here are steps to create your own virtual device.

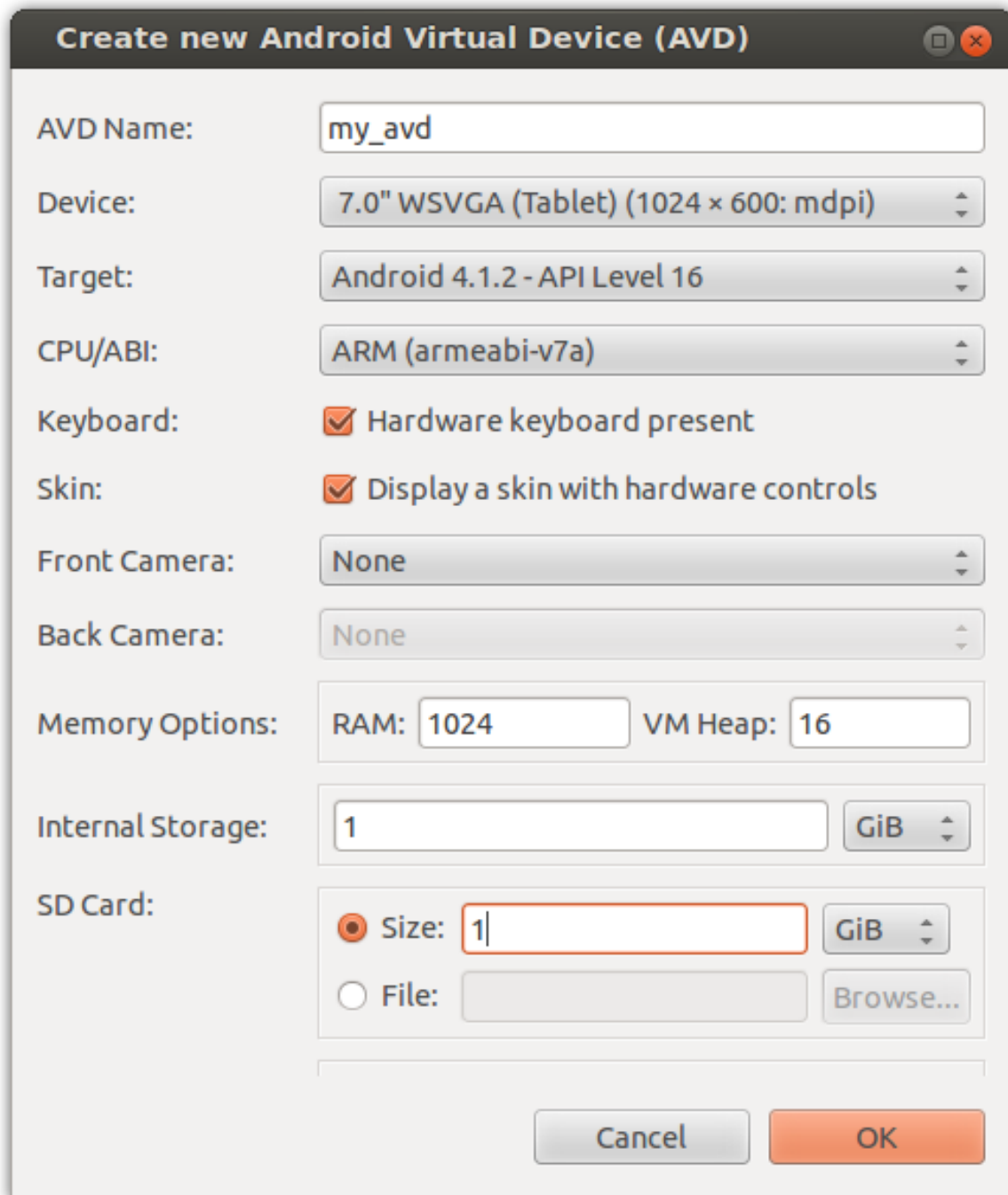
### Step 1:

Start your Virtual device manager from toolbar OR Start your Virtual device manager from menu : Window > Android Virtual Device Manager Below screen will appear with created virtual devices list (if any) and Device Definitions



**Step 2:**

Click on new button to create your new AVD (Android Virtual Device). Below dialog will appear when you click on new button.



**Create new Android Virtual Device (AVD)**

AVD Name:

Device:

Target:

CPU/ABI:

Keyboard: ☒ Hardware keyboard present

Skin: ☒ Display a skin with hardware controls

Front Camera:

Back Camera:

Memory Options: RAM:  VM Heap:

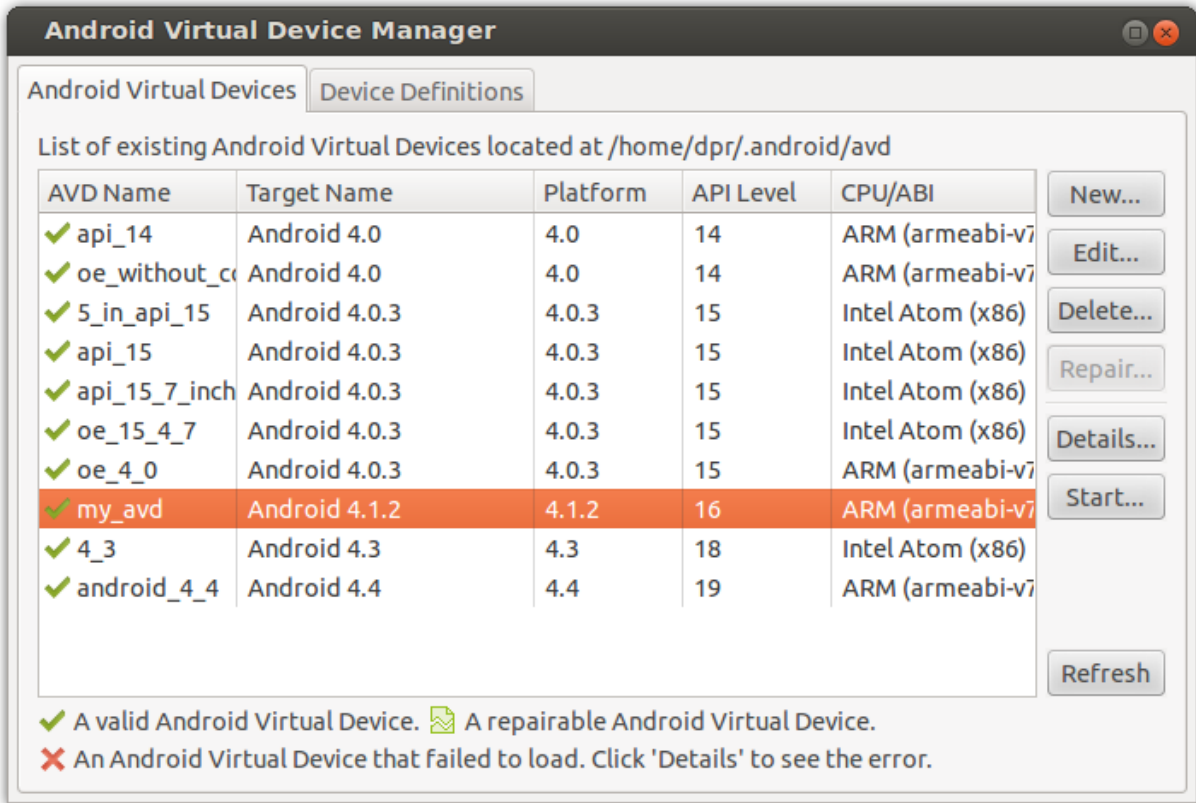
Internal Storage:

SD Card:

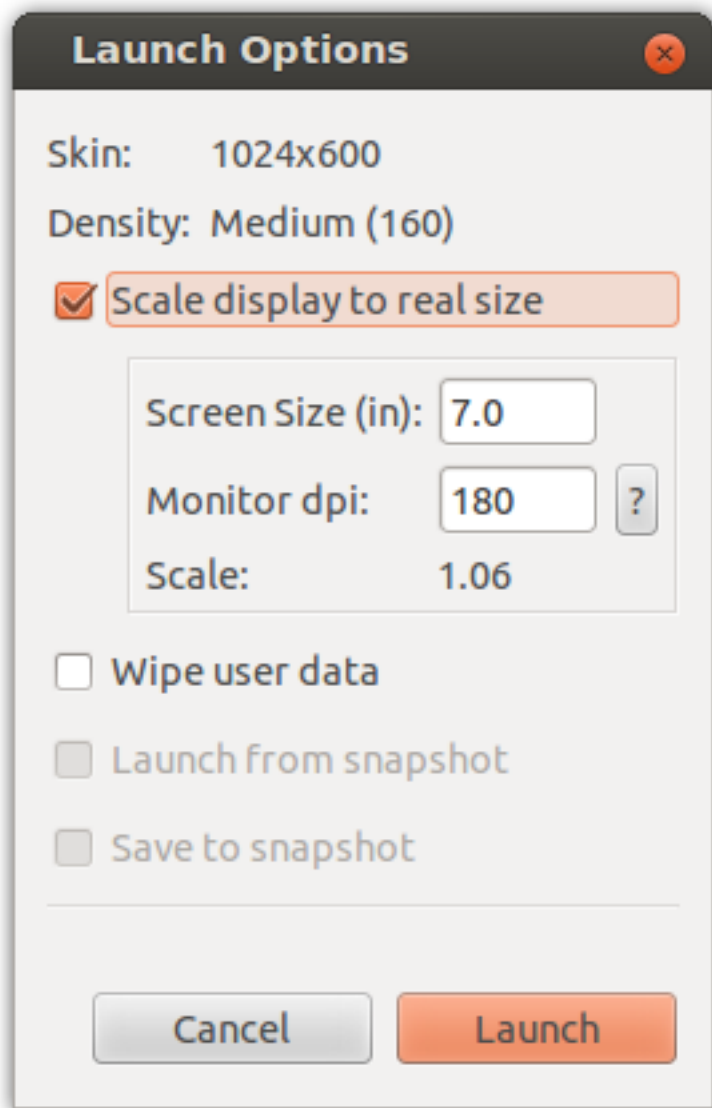
☒ Size:

☐ File:

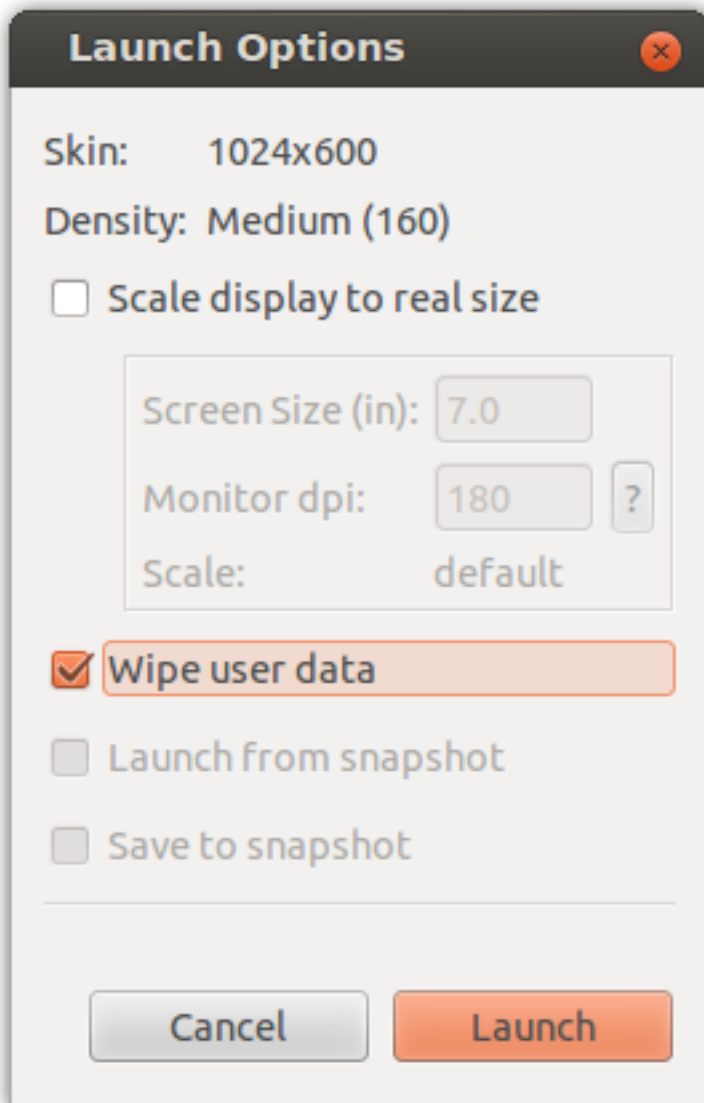
You can see your AVD in the list after successfull creation.

**Step 3:**

To launch (start) your AVD select your avd and click on Start button. Below screen will appear, if you want to resize your AVD screen than tick Scale display to real size and provide Monitor dpi as per your requirement.



You can also clean all your old data which are used at last run by checking second option Wipe user data

**Step 4:**

Click on Launch button to start your AVD



### 1.2.3 Getting OpenERP Mobile Framework

You can download latest source for framework from Launchpad branch: <https://code.launchpad.net/openerp-mobile>

You can post bugs / problems related to mobile framework at <https://bugs.launchpad.net/openerp-mobile>

Write Feedback on : [mobile@openerp.co.in](mailto:mobile@openerp.co.in)

## 1.3 Getting started with OpenERP mobile framework

### 1.3.1 Introduction

OpenERP android framework is an open source, object oriented framework works with OpenERP JSON-RPC API Connector library.

This framework provides all supporting class files required to build a module for android client.

This framework is designed to fulfill requirement of android application based on any modules (addons or app) in the OpenERP.

It provides basic menu configuration, service providers and build in ORM for managing local database (SQLite)

Model is a database part. In this framework each module has its own separate model (database helper) file to manage their related data and requested by particular module at runtime.

View is a user interface. In this framework each UI is created separately for each module.

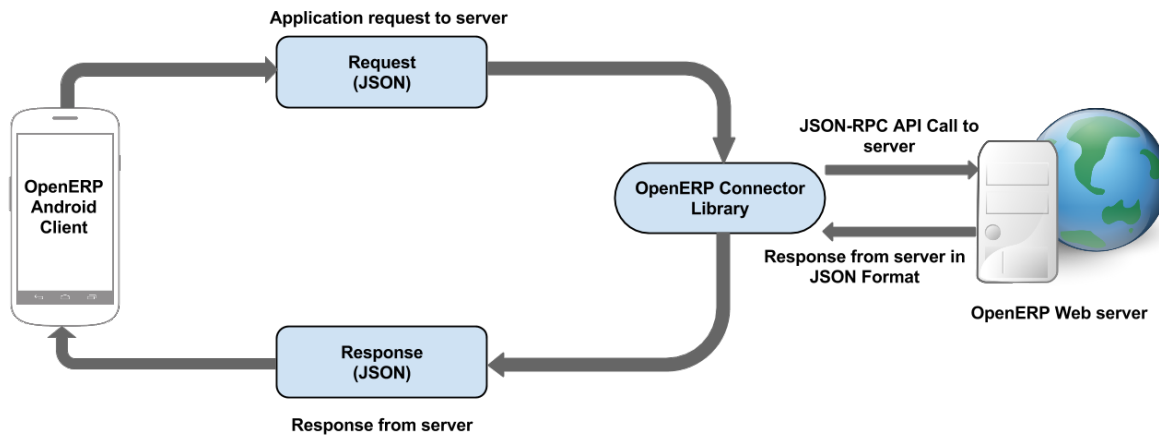
Controller is the framework itself. It will handle the user actions such as menu.

#### Basic directory structure of OpenERP Android framework:

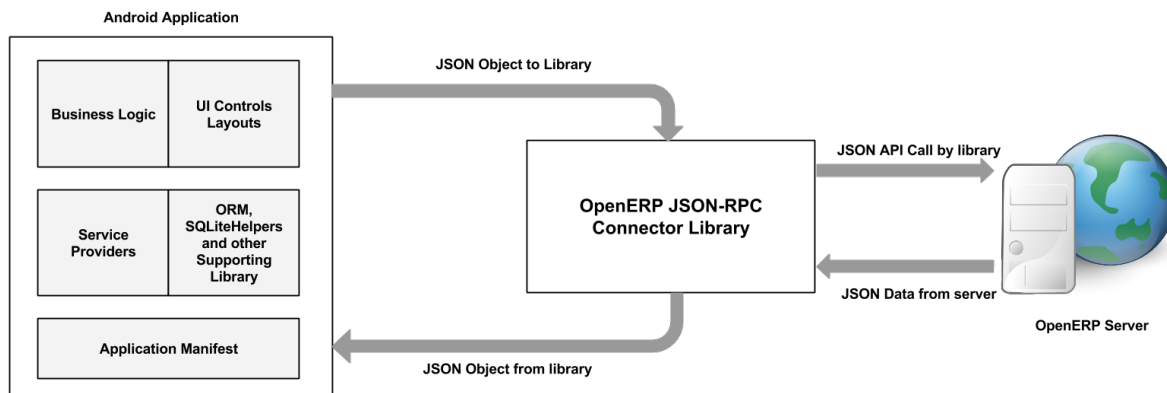
```
openerp-mobile
-- src
  -- com
    | -- openerp          // Framework loader
    |   -- base          // Base modules
    |   -- config        // Module + Sync Config
    |   -- orm           // Application ORM Package
    |   -- support       // Supporting classes
    |   -- util          // Utility classes
    |   -- auth          // Account authenticator
    |   -- addons        // All Modules (addons)
    |     -- idea        // Sample Idea Module
    |       -- services  // Idea module services (optional)
    |       -- widgets   // Idea module widgets (optional)
    |       -- providers // Idea module providers (optional)
  -- libs               // External Support libraries
  -- res               // All UI Resources
    -- drawable        // Application icons and images
    -- layout          // Application module UIs
    -- menu            // Application module menus
    -- values          // Application static String, attr, integers, styles...
    -- xml             // Application provider and preference xmls
```

## 1.3.2 OpenERP Android Architecture

### External Architecture:



### Internal Architecture:



## 1.3.3 OpenERP Android Framework Components

### Fields

#### varchar

Creates varchar (character string) type column in SQLite.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.varchar(30));
```



### integer

Creates integer type column in SQLite.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.integer(5));
```

### real

Creates real (float) type column in SQLite.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.real());
```

### booleanType

Creates boolean type column in SQLite.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.booleanType());
```

### timestamp

Creates timestamp (datetime with default current datetime value) type column in SQLite.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.timestamp(OEDate.DEFAULT_FORMAT));
```

### datetime

Creates datetime type column in SQLite.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.datetime(date_format));
```

### text

Creates text type column in SQLite.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.text());
```

### oneToMany

Creates oneToMany relation with column. Used with OpenERP Android framework ORM

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.oneToMany(new ModelObject(context));
```

### manyToOne

Creates manyToOne relation with column and also create integer type column in SQLite. Used with OpenERP Android framework ORM.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.manyToOne(new ModelObject(context));
```

### manyToMany

Creates manyToMany relation with column and create many2many related third table in SQLite based on primary column id. Used with framework ORM.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.manyToMany(new ModelObject(context));
```

### blob

Creates blob column in SQLite used to store Base64 String.

```
OEColumn column = new OEColumn("field_name", "Label", OEFIELDS.blob());
```

## Methods

Methods are based on each ModelObject which inherits OEDatabase class.

### Local Database Methods

**create** Used to create new record in Local database.

Syntax:

```
ModelObject model = new ModelObject(context);

OEValues values = new OEValues();

values.put("name", "ABC");
values.put("age", 10);

long new_id = model.create(values); // Create new record and return new created id.
```

**createORReplace** Used to create record if not exists otherwise, update record. It takes OEValues list as parameter.

Syntax:

```
ModelObject model = new ModelObject(context);

List<OEValues> records = new OEValues();

OEValues values1 = new OEValues(); // will going to create because there is no record in db.
values1.put("name", "XYZ");
values1.put("age", 15);

records.add(values1);

OEValues values2 = new OEValues(); // will going to update record because local id exists
values2.put("name", "ABC");
values2.put("age", 12);
```

```
records.add(values2);

List<Long> mAffectedIds = model.createORReplace(records); // will return affected ids.
```

**select** Used to select records from Local SQLite database.

Syntax:

```
ModelObject model = new ModelObject(context);

List<OEDataRow> records = model.select(); // Select all record from SQLite
OEDataRow record = model.select(id); // Select specific record.

List<OEDataRow> records = model.select("age > ?", new String[] { "10" }); // select records with cus
```

**update** Used to update local record.

Syntax:

```
ModelObject model = new ModelObject(context);

OEValues values = new OEValues();
values.put("name", "ABC");
values.put("age", 15);

int affected_rows = model.update(values, 10); // will return affected rows.

values = new OEValues();
values.put("age", 15);

affected_rows = model.update(values, "age = ?", new String[] { "10" });
```

**updateManyToManyRecords** Used to update many to many records of related column.

Possible operation to update Many to many records.

#### Operation.ADD

Add given ids to related table

#### Operation.APPEND

Append given ids with existing rows

#### Operation.REMOVE

Remove given ids from many to many relation table

#### Operation.REPLACE

It first remove all related rows and than replace with new one

Syntax:

```
ModelObject model = new ModelObject(context);

List<Integer> ids = new ArrayList<Integer>();

ids.put(10);
ids.put(12);
```

```
ids.put(13);

// updateManyToManyRecords("relation_column_name", Operation, "record_id", "relation_record_ids OR i
model.updateManyToManyRecords("subject_ids", Operation.REPLACE, 10, ids);
```

**delete** Used to delete record from local database

Syntax:

```
ModelObject model = new ModelObject(context);

int count = model.delete(10); // returns number of record deleted.

count = model.delete("age > ? ", new String[] {"20"}); // returns number of record deleted.
```

**truncateTable** Used to clean table records.

Syntax:

```
ModelObject model = new ModelObject(context);

boolean cleaned = model.truncateTable(); // returns boolean flag
```

**count** Used to count number of records available in local SQLite database

Syntax:

```
ModelObject model = new ModelObject(context);

int count = model.count(); // returns total number of records

int mCount = model.count("age > ?", new String[] {"5"}); // returns total number of records based on
```

**lastId** Used to get table records last id.

Syntax:

```
ModelObject model = new ModelObject(context);

int last_id = model.lastId();
```

**isEmptyTable** Used to check whether table is empty or not

Syntax:

```
ModelObject model = new ModelObject(context);

if(model.isEmptyTable()){
    //TODO: next stuff
}
```

## Server related methods

**getOEInstance** Used to get OpenERP helper instance. Which allows you to interact with OpenERP server.

Syntax:

```

ModelObject model = new ModelObject(context);

OEHelper openerp = model.getOEInstance();

```

**login** Used to create session between OpenERP server and mobile app.

Syntax:

```

ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

// Create session between application and server. Returns User object
OEUser user = openerp.login("username", "password", "database", "OpenERP server URL");

```

**isModelInstalled** Used to check whether model installed on server or not.

Syntax:

```

ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

if(openerp.isModelInstalled("note.note")){
    //TODO: next stuff
}

```

**syncWithServer** Used to sync data with server. It will automatically takes all columns from ModelObject and request to server. After getting response it will store each record and its relation records to its specific tables.

Syntax:

```

ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

// Returns True, if sync finished.
if(openerp.syncWithServer()){
    //TODO: sync completed. next stuff
}

```

Possible parameters:

```

// all default parameters
syncWithServer()

// take boolean. If true, will remove local record if it not exists on server.
syncWithServer(boolean removeLocalRecordIfNotExists)

// take OEDomain, If passed, will filter requesting records from server.
syncWithServer(OEDomain domain)

// take two argumetns as said above.
syncWitServer(OEDomain domain, boolean removeLocalRecordIfNotExists)

```

Example:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

OEDomain domain = new OEDomain();
domain.put("age", ">", 10);

boolean done = openerp.syncWithServer(domain);
```

**syncWithMethod** This method will be used when you have created your model based on server's custom method result columns.

Syntax:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

OEArguments args = new OEArguments();

List<Integer> ids = new ArrayList<Integer>();
ids.put(10);
ids.put(20);

args.add(ids);
args.add(true);
args.addNull();

// OEArguments will create : [[10,20], true, null]

boolean done = openerp.syncWithMethod("get_users", args);
```

**create** Used to create record on server. After creating record on server ORM will create that record in local database by itself.

Syntax:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

OEValues values = new OEValues();
values.put("name", "PQR");
values.put("age", 14);

int new_id = openerp.create(values); // returns new created id
```

**update** Used to update server record. After updating on server it will update record in local database.

Syntax:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

OEValues values = new OEValues();
values.put("age", 15);

if(openerp.update(values, 20)){
```

```
// updated
}
```

**delete** Used to remove record from server after removing on server will remove from local.

Syntax:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

openerp.delete(20);
```

**search\_read** Used to select record from server. Will return List of OEDataRow.

Syntax:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

List<OEDataRow> records = openerp.search_read();
```

**search\_read\_remain** Used to select record that are not exists on local database. > last id.

Syntax:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

List<OEDataRow> records = openerp.search_read_remain();
```

**call\_kw** Used to call server custom method. Returns JSONObject.

Syntax:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

OEArguments args = new OEArguments();

List<Integer> ids = new ArrayList<Integer>();
ids.put(10);
ids.put(20);

args.put(ids);
args.put(true);
args.putNull();

JSONObject result = openerp.call_kw("get_users", args);
```

Possible parameters:

```
// method and arguments
call_kw(String method, OEArguments arguments)

// method, arguments and user context (if any)
call_kw(String method, OEArguments arguments, JSONObject context)
```

```
// method, arguments, user context and kwargs if any
call_kw(String method, OEArguments arguments, JSONObject context, JSONObject kwargs)

// model, method, arguments, user context and kwargs if any
call_kw(String model, String method, OEArguments arguments, JSONObject context, JSONObject kwargs)
```

Getting user context:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

JSONObject newContextValues = new JSONObject();
JSONObject context = openerp.openERP().updateContext(newContextValues);
```

**getRemovedRecords** Used after sync finish. If any record removed from local than it will return list of removed records.

Syntax:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

if(openerp.syncWithServer()){
    List<OEDataRow> removedRecords = openerp.getRemovedRecords();
}
```

**getAffectedIds** Used after sync finish. If any record affected after sync it will return list of ids.

Syntax:

```
ModelObject model = new ModelObject(context);
OEHelper openerp = model.getOEInstance();

if(openerp.syncWithServer()){
    List<Integer> ids = openerp.getAffectedIds();
}
```

## Drawer Menu

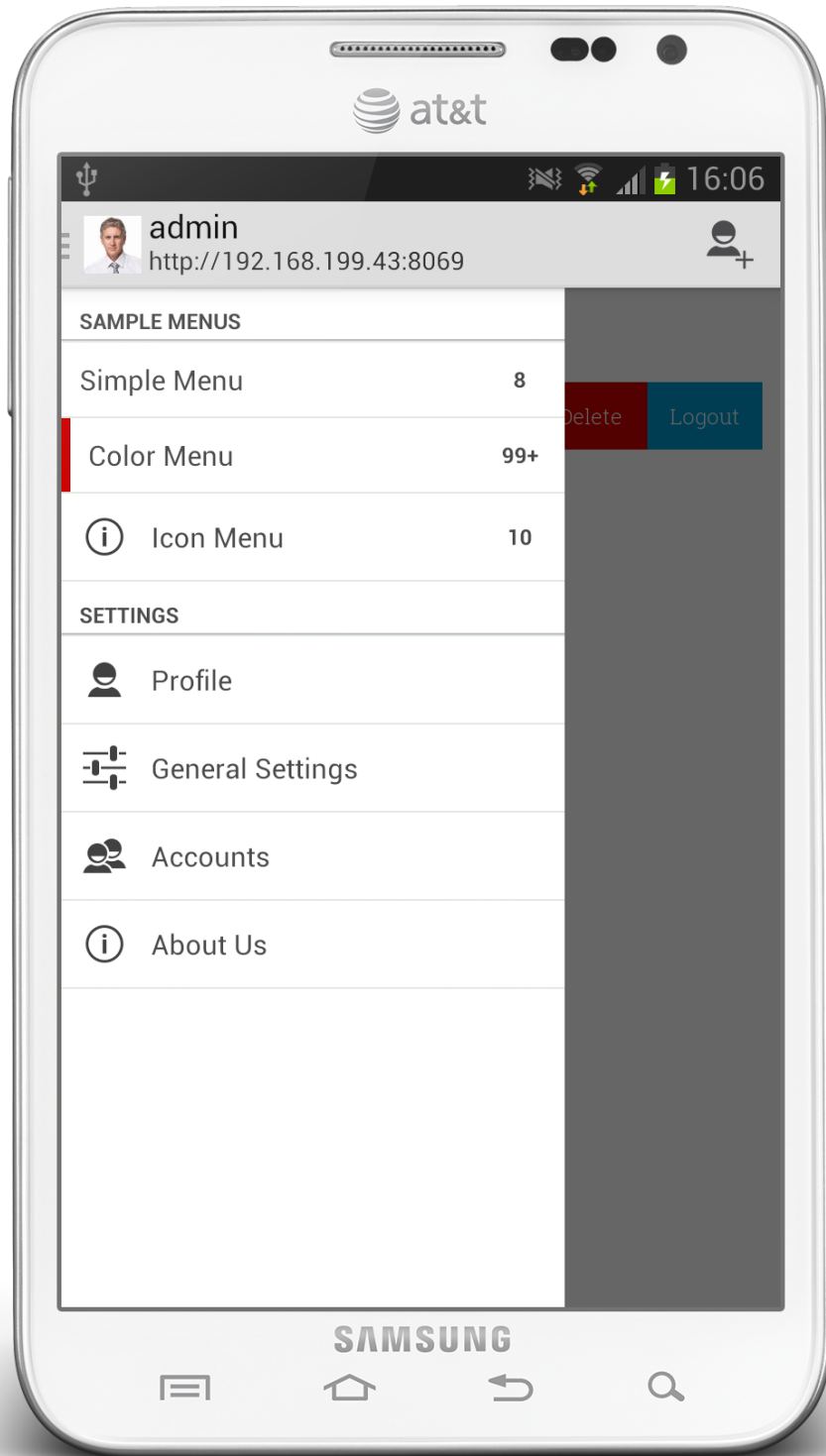
Menus are based on each module of android app. It contains different layout parameter to display different type of menu.

To generate different menu items with icon, tag color and badge we use different constructors.

```
@Override
public List<DrawerItem> drawerMenus(Context context) {
    List<DrawerItem> menu = new ArrayList<DrawerItem>();

    menu.add(new DrawerItem(TAG, "Sample Menus", true));
    menu.add(new DrawerItem(TAG, "Simple Menu", 8, 0, object("all")));
    menu.add(new DrawerItem(TAG, "Color Menu", 100, "#cc0000", object("all")));
    menu.add(new DrawerItem(TAG, "Icon Menu", 10, R.drawable.ic_action_about, object(""));
    return menu;
}
```





## OETouchListener

OETouchListener class used for providing pulling or swiping features to your listview.

## PullListener

Called when pulled down listview, gridview or scrollview.

Syntax:

```
class Demo extends BaseFragment implements OnPullListener {

    OETouchListener mTouchListener = null;
    ListView mListView = null;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        mView = inflater.inflate(R.layout.demo_layout, container, false);

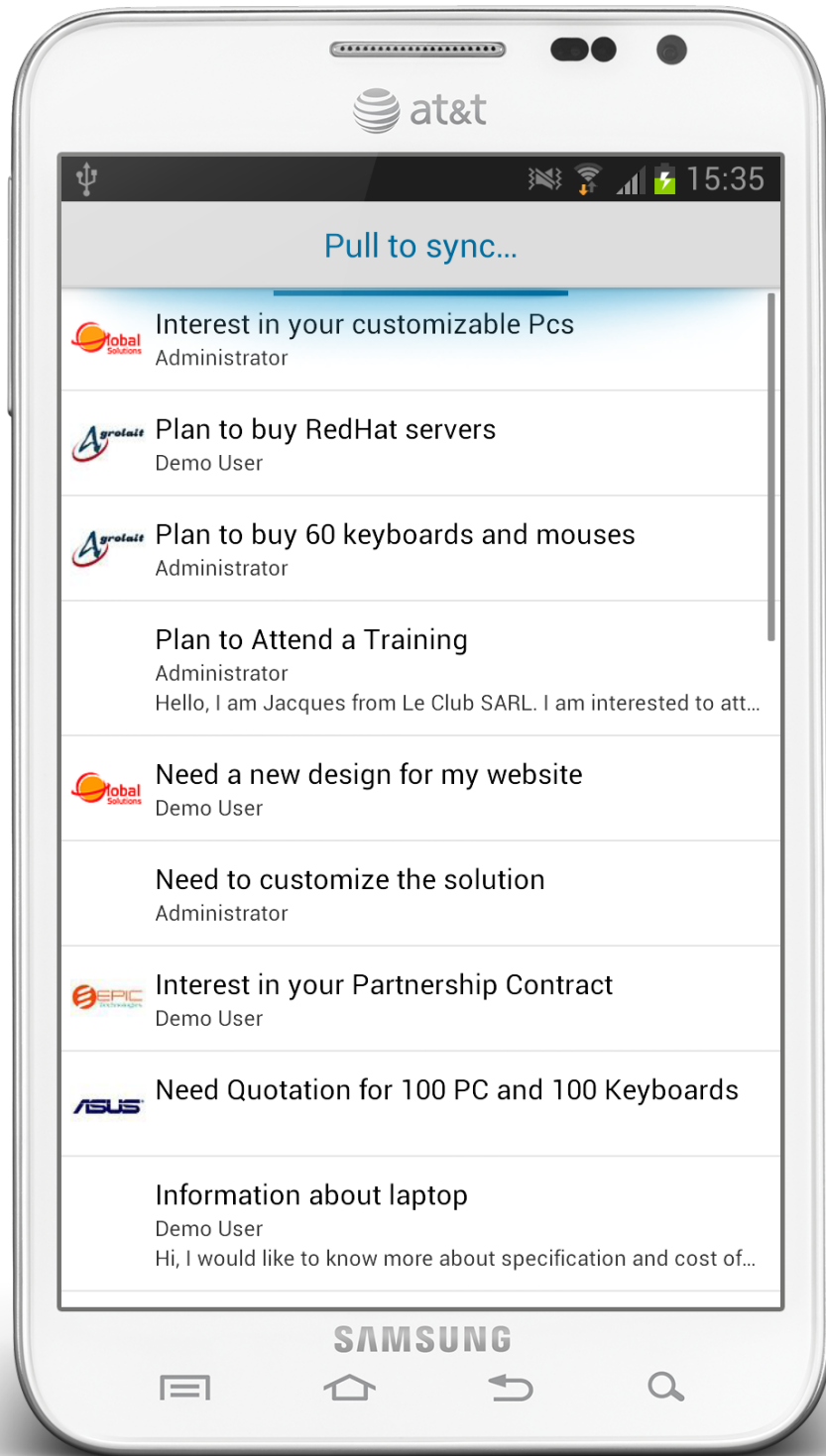
        mListView = (ListView) mView.findViewById(R.id.myListView);

        scope = new AppScope(getActivity());
        mTouchListener = scope.main().getTouchAttacher();

        mTouchListener.setPullableView(mListview, this);

        return mView;
    }

    @Override
    public void onPullStarted(View view) {
        // Task to-do
    }
}
```



## SwipeListener

Called when you swipe listview or gridview row left to right or right to left.

```
class Demo extends BaseFragment implements SwipeCallbacks {

    OETouchListener mTouchListener = null;
    ListView mListView = null;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        mView = inflater.inflate(R.layout.demo_layout, container, false);

        mListView = (ListView) mView.findViewById(R.id.myListView);

        scope = new AppScope(getActivity());
        mTouchListener = scope.main().getTouchAttacher();

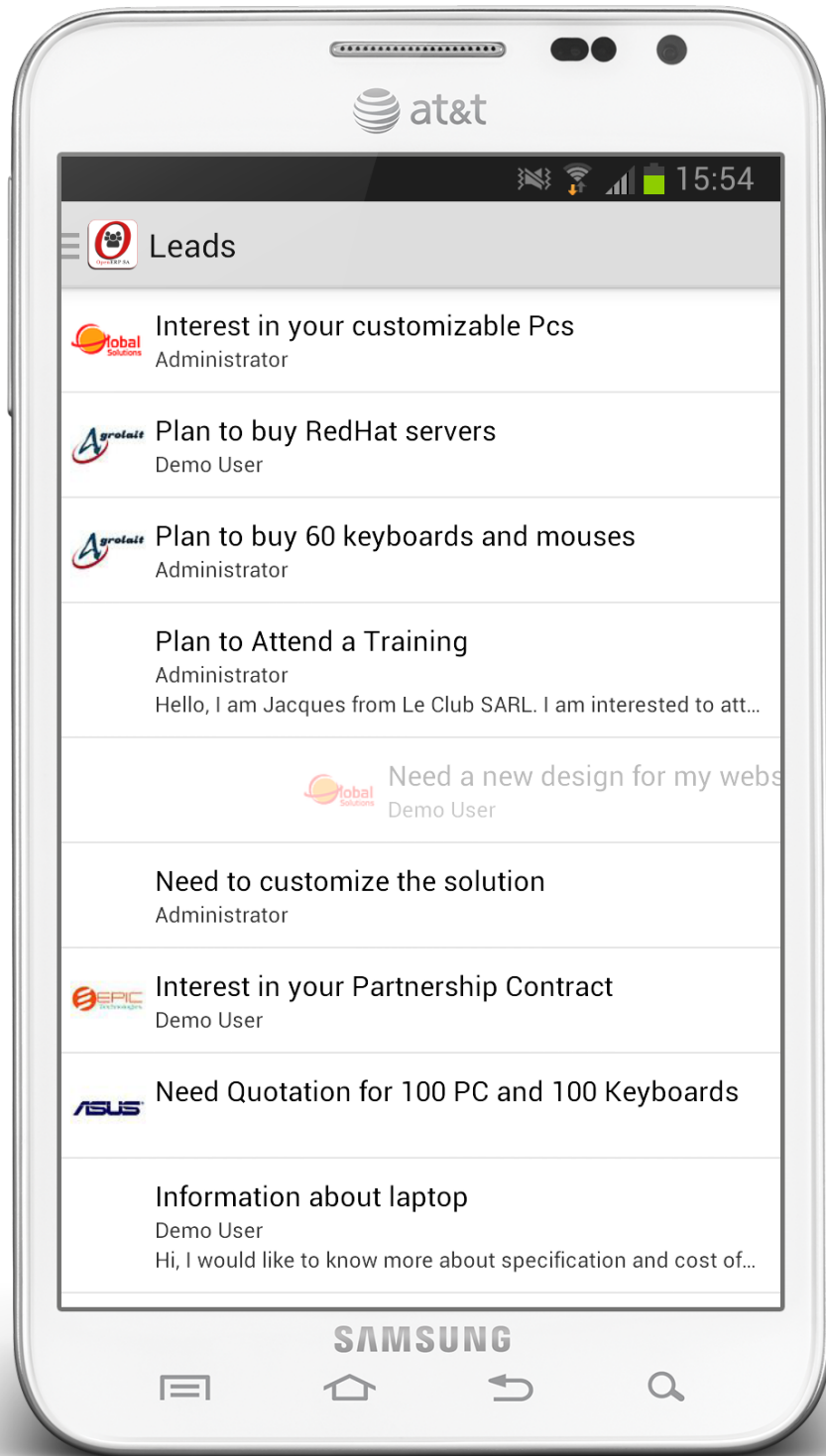
        mTouchListener.setSwipeableView(mListView, this);
        mListView.setOnScrollListener(mTouchListener.makeScrollListener());

        return mView;
    }

    @Override
    public boolean canSwipe(int position) {
        // return true if you want to allow position to swipe
        return true;
    }

    @Override
    public void onSwipe(View view, int[] ids) {

    }
}
```



## TagsView

### TagsViewDemo.java

TagsView provide AutoComplete Tags with custom UI.

```
public class TagsViewDemo extends BaseFragment implements TokenListener {

    View mView = null;
    TagsView mTagsView = null;
    OEListAdapter mAdapter = null;
    List<Object> mPartners = new ArrayList<Object>();

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        mView = inflater.inflate(R.layout.tags_view_layout, container, false);
        init();
        return mView;
    }

    private void init() {

        mPartners.clear();
        mPartners.addAll(db().select());
        mTagsView = (TagsView) mView.findViewById(R.id.tagsView);

        // generate tags (small tag)
        mTagsView.setCustomTagView(new CustomTagViewListener() {

            @Override
            public View getViewForTags(LayoutInflater inflater,
                Object object, ViewGroup tagsViewGroup) {
                View view = inflater.inflate(
                    R.layout.tags_view_custom_tag_layout, tagsViewGroup,
                    false);
                initView(object, view);

                return view;
            }

        });

        mAdapter = new OEListAdapter(getActivity(),
            R.layout.tags_view_custom_layout, mPartners) {

            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                View view = convertView;
                if (view == null)
                    view = getActivity().getLayoutInflater().inflate(
                        getResource(), parent, false);
                initView(mPartners.get(position), view);
                return view;
            }

        };
        mTagsView.setTokenListener(this);
        mTagsView.setAdapter(mAdapter);
    }
}
```

```

private void initView(Object object, View view) {
    OEDataRow row = (OEDataRow) object;
    ImageView imgPic = (ImageView) view.findViewById(R.id.tagImage);
    TextView txvName, txvEmail;
    txvName = (TextView) view.findViewById(R.id.tagName);
    txvEmail = (TextView) view.findViewById(R.id.tagEmail);

    imgPic.setImageBitmap(Base64Helper.getBitmapImage(getActivity(),
        row.getString("image_small")));
    txvName.setText(row.getString("name"));
    txvEmail.setText(row.getString("email"));
}

@Override
public Object databaseHelper(Context context) {
    return new ResPartnerDB(context);
}

@Override
public List<DrawerItem> drawerMenus(Context context) {
    List<DrawerItem> menu = new ArrayList<DrawerItem>();

    menu.add(new DrawerItem("key", "TagView demo", true));
    menu.add(new DrawerItem("key", "TagView Demo", 0, "#0099cc",
        object("all")));
    return menu;
}

private Fragment object(String value) {
    TagsViewDemo lead = new TagsViewDemo();
    Bundle bundle = new Bundle();
    bundle.putString("value", value);
    lead.setArguments(bundle);
    return lead;
}

@Override
public void onTokenAdded(Object token, View view) {
    OEDataRow row = (OEDataRow) token;
    Toast.makeText(getActivity(), row.getString("name") + " added.",
        Toast.LENGTH_LONG).show();
}

@Override
public void onTokenSelected(Object token, View view) {
}

@Override
public void onTokenRemoved(Object token) {
    OEDataRow row = (OEDataRow) token;
    Toast.makeText(getActivity(), row.getString("name") + " removed.",
        Toast.LENGTH_LONG).show();
}
}

```

**tags\_view\_layout.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clipToPadding="false"
    android:orientation="vertical"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:paddingTop="?android:attr/actionBarSize" >

    <com.openerp.util.tags.TagsView
        android:id="@+id/tagsView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

**tags\_view\_custom\_tag\_layout.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#f5f5f5"
    android:orientation="horizontal" >

    <ImageView
        android:id="@+id/tagImage"
        android:layout_width="26dp"
        android:layout_height="26dp"
        android:layout_gravity="center_vertical"
        android:src="@drawable/ic_launcher" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="5dp" >

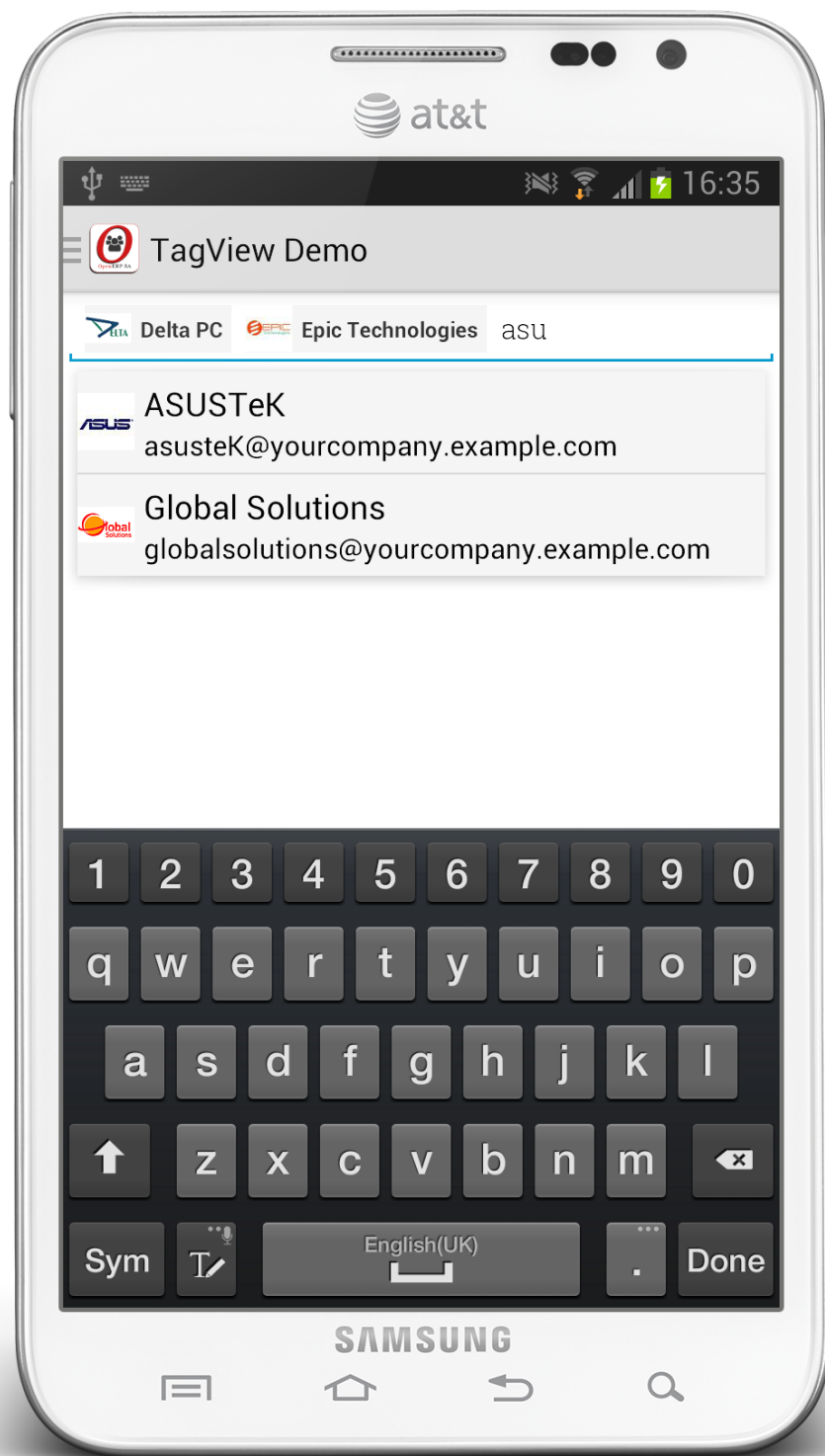
        <TextView
            android:id="@+id/tagName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Name"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/tagEmail"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Email"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:visibility="gone" />

    </LinearLayout>
```



```
</LinearLayout>
```



**Download Demo application source** `file/openerp-mobile-crm.zip`

## 1.4 OpenERP Android Framework Development

### 1.4.1 Building an OpenERP Mobile module

#### CRM Lead Module Development

A very basic module structure will be our starting point:

---

**Note:** As a sample, we are going to develop mobile app for **CRM Lead**.

---

For developing any module you need to add a folder in your `addons` directory. So, we are creating a folder for `crm` in `addons`. Create two files under `crm`:

- `Lead.java` (*Android code behind file.*)
- `LeadDB.java` (*Used for creating local database table and fetching data from sever*)

For creating sync service and module wise sync provider add two folder in `crm` and add `LeadService` file into `services` and `LeadProvider` into `providers`:

```
-- crm
-- services
|   -- LeadService.java
-- providers
--   lead
--     -- LeadProvider.java
```

See: [Basic directory structure of OpenERP Android framework](#):

**After adding files your directory structure looks like this:**

```
-- addons
--   crm
--     -- LeadDB.java
--     -- Lead.java
--     -- services
--     |   -- LeadService.java
--     -- providers
--     --   lead
--     --     -- LeadProvider.java
```

#### LeadDB.java

In this file, you need to inherit `OEDatabase` which implements two methods:

- `getModelName()` - The name of your server model. Here, `crm.lead`
- `getModelColumns()` - Returns the list of database table columns

Content of file:

```
package com.openerp.addons.crm;

import java.util.List;
```

```
import android.content.Context;

import com.openerp.orm.OEColumn;
import com.openerp.orm.OEDatabase;

public class LeadDB extends OEDatabase {

    public LeadDB(Context context) {
        super(context);
    }

    @Override
    public String getModelName() {
        return "crm.lead";
    }

    @Override
    public List<OEColumn> getModelColumns() {
        return null;
    }

}
```

Now we move to add columns in LeadDB.

```
package com.openerp.addons.crm;

import java.util.ArrayList;
import java.util.List;

import android.content.Context;

import com.openerp.base.res.ResPartnerDB;
import com.openerp.orm.OEColumn;
import com.openerp.orm.OEDatabase;
import com.openerp.orm.OEFields;

public class LeadDB extends OEDatabase {

    Context mContext = null;
    public LeadDB(Context context) {
        super(context);
        mContext = context;
    }

    @Override
    public String getModelName() {
        return "crm.lead";
    }

    @Override
    public List<OEColumn> getModelColumns() {
        List<OEColumn> cols = new ArrayList<OEColumn>();

        cols.add(new OEColumn("name", "name", OEFields varchar(64)));
        cols.add(new OEColumn("partner_id", "Customer", OEFields manyToOne(new ResPartnerDB(mContext)));
        cols.add(new OEColumn("user_id", "Sales Person", OEFields manyToOne(new ResUsers(mContext)));
        cols.add(new OEColumn("categ_ids", "Category", OEFields manyToMany(new CRMCaseCateg(mContext)));
    }

}
```

```

        cols.add(new OEColumn("description", "Description", OEFIELDS.text()));

        return cols;
    }
    class ResUsers extends OEDatabase {

        public ResUsers(Context context) {
            super(context);
        }

        @Override
        public String getModelName() {
            return "res.users";
        }

        @Override
        public List<OEColumn> getModelColumns() {
            List<OEColumn> cols = new ArrayList<OEColumn>();
            cols.add(new OEColumn("name", "name", OEFIELDS.varchar(64)));
            return cols;
        }
    }
    class CRMCaseCateg extends OEDatabase {

        public CRMCaseCateg(Context context) {
            super(context);
        }

        @Override
        public String getModelName() {
            return "crm.case.categ";
        }

        @Override
        public List<OEColumn> getModelColumns() {
            List<OEColumn> cols = new ArrayList<OEColumn>();
            cols.add(new OEColumn("name", "name", OEFIELDS.varchar(64)));
            return cols;
        }
    }
}

```

Reference: [Fields](#)

Here, we have created two other classes inside LeadDB:

- ResUsers
- CRMCaseCateg

We required relation model to provide column relation. Here, ResPartnerDB is already in base database models but we required two other models for our relation in columns (many2many and many2one).

Now we are ready with our database structure for CRM Lead.

### Lead.java

In this file, you need to inherit abstract BaseFragment which implements two methods:

- databaseHelper() - Returns LeadDB instance.
- drawerMenus() - Returns list of Drawer menu

Reference: [Drawer Menu](#)

Content of file:

```
package com.openerp.addons.crm;

import java.util.ArrayList;
import java.util.List;

import android.content.Context;
import android.os.Bundle;
import android.support.v4.app.Fragment;

import com.openerp.support.BaseFragment;
import com.openerp.util.drawer.DrawerItem;

public class Lead extends BaseFragment {

    public static final String TAG = Lead.class.getSimpleName();

    @Override
    public Object databaseHelper(Context context) {
        return new LeadDB(context);
    }

    @Override
    public List<DrawerItem> drawerMenus(Context context) {
        List<DrawerItem> menu = new ArrayList<DrawerItem>();

        menu.add(new DrawerItem(TAG, "Leads", true));
        menu.add(new DrawerItem(TAG, "Leads", 0, "#cc0000", object("all")));
        return menu;
    }

    private Fragment object(String value) {
        Lead lead = new Lead();
        Bundle bundle = new Bundle();
        bundle.putString("lead_value", value);
        lead.setArguments(bundle);
        return lead;
    }
}
```

### **LeadService.java**

This file used to fetch data from server when user start sync service. It inherit OEService class which implement two methods:

- getService() - returns self object i.e., this
- performSync() - called when user start sync.

Content of file:

```

package com.openerp.addons.crm.services;

import android.accounts.Account;
import android.app.Service;
import android.content.ContentProviderClient;
import android.content.Context;
import android.content.SyncResult;
import android.os.Bundle;

import com.openerp.support.service.OEService;

public class LeadService extends OEService {

    @Override
    public Service getService() {
        return this;
    }

    @Override
    public void performSync(Context context, Account account, Bundle extras,
        String authority, ContentProviderClient provider,
        SyncResult syncResult) {

    }

}

```

Here, `performSync()` method will be called when user starts sync. We just need to call our database model and start sync for our model.

```

public class LeadService extends OEService {

    @Override
    public Service getService() {
        return this;
    }

    @Override
    public void performSync(Context context, Account account, Bundle extras,
        String authority, ContentProviderClient provider,
        SyncResult syncResult) {
+
+         LeadDB lead = new LeadDB(context);
+         OEHelper oe = lead.getOEInstance();
+         if (oe != null) {
+             if (oe.syncWithServer()) {
+                 // TODO: Sync finished
+             }
+         }
    }

}

```

Now, add service entry into `AndroidManifest.xml` file: Add service tag under `<application>` tag

```

<service
    android:name="com.openerp.addons.crm.services.LeadService"
    android:exported="true" >
</service>

```

We are ready with our service but; we need service provider and its syn-adapter xml file to display it under Android Accounts.

### LeadProvider.java

This file provide authority and sync adapter to our sync service. It inherit `OEContentProvider` which implement two methods:

- `authority()` - returns provider authority string
- `contentUri()` - returns provider content uri string

Content of file:

```
package com.openerp.addons.crm.providers.lead;

import com.openerp.support.provider.OEContentProvider;

public class LeadProvider extends OEContentProvider {
    public static String CONTENTURI = "com.openerp.addons.crm.providers.lead.LeadProvider";

    public static String AUTHORITY = "com.openerp.addons.crm.providers.lead";

    @Override
    public String authority() {
        return AUTHORITY;
    }

    @Override
    public String contentUri() {
        return CONTENTURI;
    }
}
```

Content Provider is ready now, but we must link this provider with sync-adapter xml file.

Create new `sync_crm_adapter.xml` file under `/res/xml/`

```
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.openerp.auth"
    android:contentAuthority="com.openerp.addons.crm.providers.lead"
    android:supportsUploading="true"
    android:userVisible="true" />
```

Here, we uses `com.openerp.auth` type for creating our custom OpenERP account under Android Accounts.

Now, we have to add this providers entry into `AndroidManifest.xml` file using `<provider>` tag.

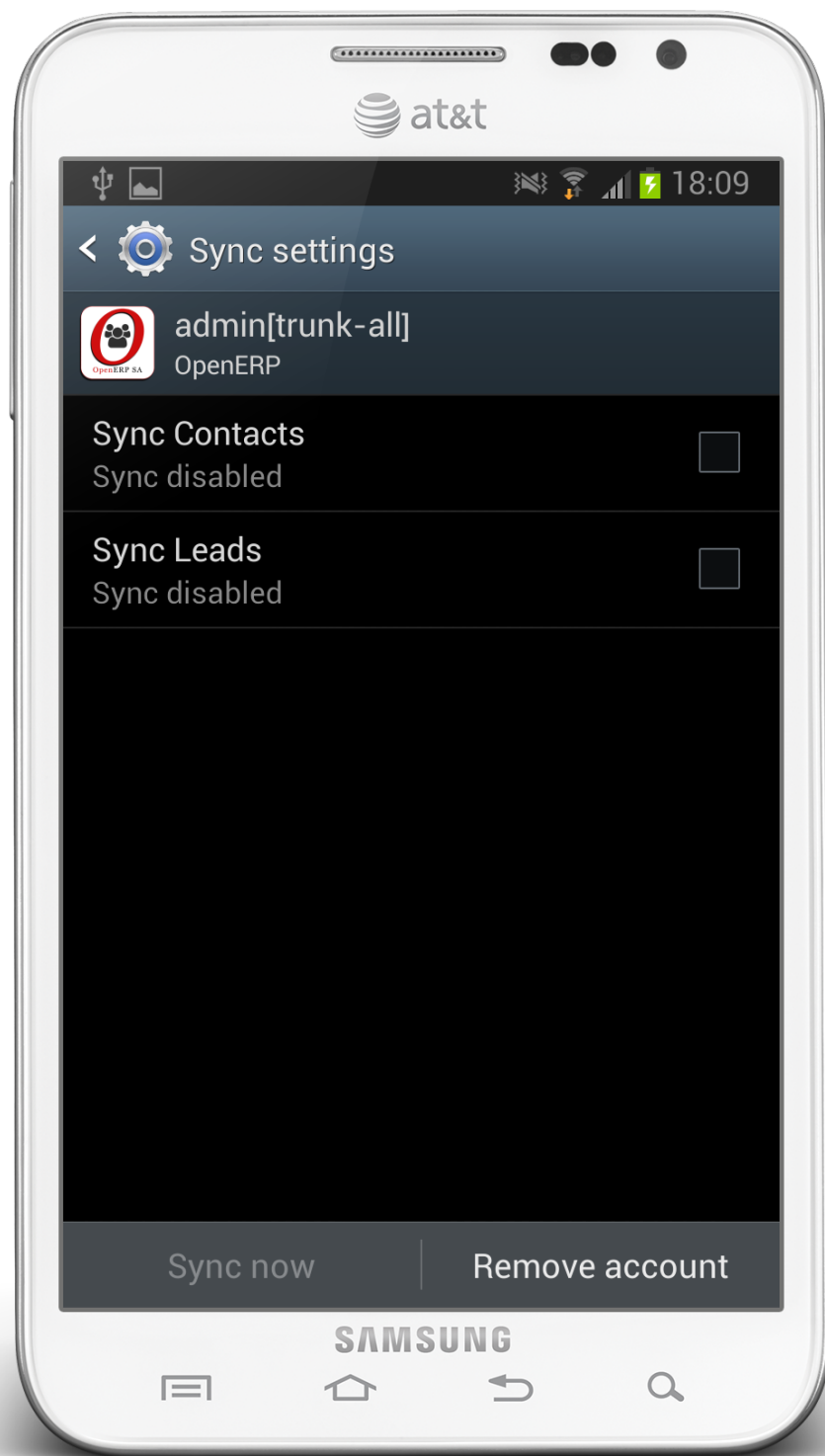
```
+ <provider
+     android:name="com.openerp.addons.crm.providers.lead.LeadProvider"
+     android:authorities="com.openerp.addons.crm.providers.lead"
+     android:enabled="true"
+     android:exported="true"
+     android:label="Leads"
+     android:syncable="true" />

<service
    android:name="com.openerp.addons.crm.services.LeadService"
    android:exported="true" >
```



```
+      <intent-filter>
+          <action android:name="android.content.SyncAdapter" />
+      </intent-filter>
+
+      <meta-data
+          android:name="android.content.SyncAdapter"
+          android:resource="@xml/sync_crm_adapter" />
+  </service>
```

To test that sync service is added under Android account run your application and create account. You can find your sync service provider as below screenshot:



Now, We are ready with our core setup. Registering our module in ModuleConfig.java under com.openerp.config package.

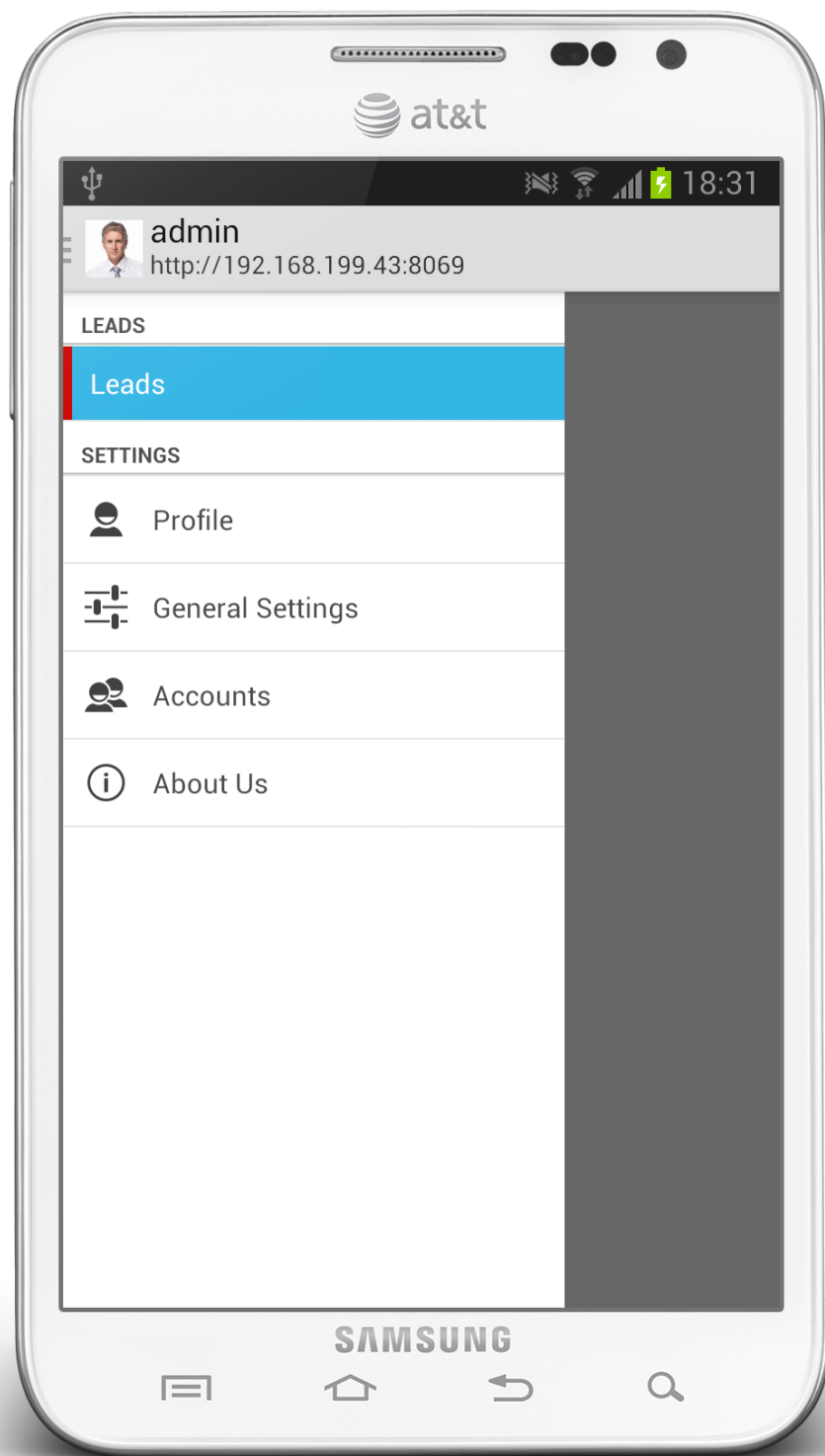
```
package com.openerp.config;

import com.openerp.addons.crm.Lead;
import com.openerp.support.Module;
import com.openerp.support.ModulesConfigHelper;

public class ModulesConfig extends ModulesConfigHelper {

    public ModulesConfig() {
        add(new Module("module_crm", "CRM Lead", new Lead()), true);
    }
}
```

Now, First of all uninstall app from your device or emulator because it had created some base database tables. After uninstalling run your application. It will show you your menu for CRM Lead as below screenshot:



Now, we create one `crm_lead_layout.xml` file for viewing all leads. Create file under `res/layout/`

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/crmLeadListView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:clipToPadding="false"
        android:paddingTop="?android:attr/actionBarSize" >
    </ListView>

</LinearLayout>
```

Also create one `crm_lead_custom_row.xml` that we will use for our `ListView` custom layout.

```
<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" >

    <ImageView
        android:id="@+id/imgCustomerPic"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center_vertical"
        android:src="@drawable/ic_launcher" />

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="vertical"
        android:padding="5dp" >

        <TextView
            android:id="@+id/txvLeadName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Lead Name"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <TextView
            android:id="@+id/txvLeadSalesPerson"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Sales Person Name"
            android:textAppearance="?android:attr/textAppearanceSmall" />

        <TextView
            android:id="@+id/txvLeadDescription"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ellipsize="end"
            android:singleLine="true"
```

```
        android:text="Lead Description"
        android:textAppearance="?android:attr/textAppearanceSmall" />
    </LinearLayout>

</LinearLayout>
```

We have created our Layout files. Now open `Lead.java` file and override `onCreateView()` method.

```
public class Lead extends BaseFragment {

    public static final String TAG = Lead.class.getSimpleName();

+   View mView = null;
+   ListView mListView = null;
+   OEListAdapter mListAdapter = null;
+   List<Object> mLeadItems = new ArrayList<Object>();

+   @Override
+   public View onCreateView(LayoutInflater inflater, ViewGroup container,
+                           Bundle savedInstanceState) {
+       mView = inflater.inflate(R.layout.crm_layout, container, false);
+       return mView;
+   }

    . . . . .
    . . . . .
```

Create `init()` method to for initiate objects.

```
public class Lead extends BaseFragment {

    public static final String TAG = Lead.class.getSimpleName();

    View mView = null;
    ListView mListView = null;
    OEListAdapter mListAdapter = null;
    List<Object> mLeadItems = new ArrayList<Object>();
+   LeadLoader mLeadLoader = null;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
        scope = new AppScope(getActivity());
        mView = inflater.inflate(R.layout.crm_lead_layout, container, false);
+       init();
        return mView;
    }

+   private void init() {
+       mListView = (ListView) mView.findViewById(R.id.crmLeadListView);
+       mListAdapter = new OEListAdapter(getActivity(),
+                                       R.layout.crm_lead_custom_row, mLeadItems) {
+           @Override
+           public View getView(int position, View convertView, ViewGroup parent) {
+               View view = convertView;
+               if (view == null)
+                   view = getActivity().getLayoutInflater().inflate(getResource
+                               parent, false);
+               return view;
+           }
+       };
+   }
```

```

+         }
+     };
+     mListView.setAdapter(mListAdapter);
+     mLeadLoader = new LeadLoader();
+     mLeadLoader.execute();
+ }
+
+ private void checkStatus() {
+     if (db().isEmptyTable()) {
+         scope.main().requestSync(LeadProvider.AUTHORITY);
+     }
+ }
+
+ class LeadLoader extends AsyncTask<Void, Void, Void> {
+
+     @Override
+     protected Void doInBackground(Void... params) {
+         mLeadItems.clear();
+         LeadDB db = new LeadDB(getActivity());
+         mLeadItems.addAll(db.select());
+         return null;
+     }
+
+     @Override
+     protected void onPostExecute(Void result) {
+         mListAdapter.notifyDataChange(mLeadItems);
+         checkStatus();
+     }
+ }

```

Here, we have created **asynchronous task** LeadLoader for loading row from data table in background process. After finishing data loading we have checked for record availability. If we got empty table than system will start sync as written in checkStatus() method.

Now, we have to fill each data in our custom row of ListView as below code:

```

private void init() {
    mListView = (ListView) mView.findViewById(R.id.crmLeadListView);
    mListAdapter = new OEListAdapter(getActivity(),
        R.layout.crm_lead_custom_row, mLeadItems) {
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            View view = convertView;
            if (view == null)
                view = getActivity().getLayoutInflater().inflate(getResource(),
                    parent, false);

+            OEDataRow row = (OEDataRow) mLeadItems.get(position);

+            ImageView imgCustomerPic = (ImageView) view.findViewById(R.id.imgCustomerPic);
+            Bitmap bitmap = null;
+            OEDataRow partner = row.getM20Record("partner_id").browse();
+            if (partner != null) {
+                String base64Image = partner.getString("image_small");
+                bitmap = Base64Helper.getBitmapImage(getActivity(), base64Image);
+            }
+            imgCustomerPic.setImageBitmap(bitmap);

+            TextView txvName, txvSalesPerson, txvDescription;

```

```
+         txvName = (TextView) view.findViewById(R.id.txvLeadName);
+         txvSalesPerson = (TextView) view.findViewById(R.id.txvLeadSalesPerson);
+         txvDescription = (TextView) view.findViewById(R.id.txvLeadDescription);

+         txvName.setText(row.getString("name"));
+         OEDataRow user = row.getM2ORecord("user_id").browse();
+         String salesPerson = "";
+         if (user != null)
+             salesPerson = user.getString("name");
+         txvSalesPerson.setText(salesPerson);

+         if(salesPerson.equals("false"))
+             txvSalesPerson.setVisibility(View.GONE);

+         txvDescription.setText(row.getString("description"));
+         if(row.getString("description").equals("false"))
+             txvDescription.setVisibility(View.GONE);
+         return view;
    }

    };
    mListView.setAdapter(mListAdapter);
}
```

Now we are ready with our app. But when there are no any data in data table app will start sync, on sync finish we have to reload listview. For that we just register SyncFinishReceiver receiver in your fragment.

```
@Override
public void onResume() {
    super.onResume();
    getActivity().registerReceiver(mSyncFinish,
        new IntentFilter(SyncFinishReceiver.SYNC_FINISH));
}

@Override
public void onPause() {
    super.onPause();
    getActivity().unregisterReceiver(mSyncFinish);
}

SyncFinishReceiver mSyncFinish = new SyncFinishReceiver() {
    public void onReceive(Context context, android.content.Intent intent) {
        mLeadLoader = new LeadLoader();
        mLeadLoader.execute();
    }
};
```

Broadcast Syncfinish intent from your LeadService when sync finished:

```
public class LeadService extends OEService {

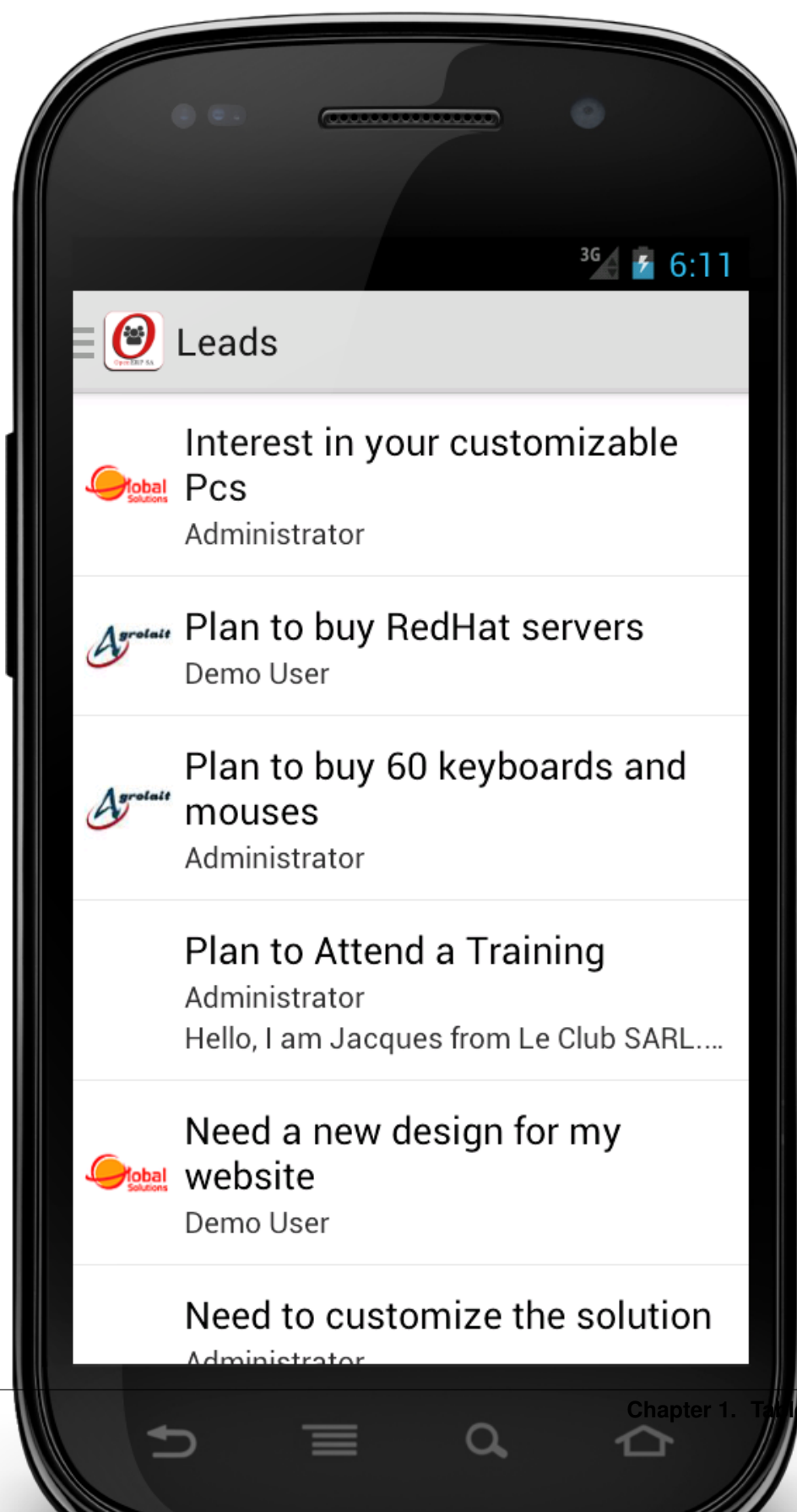
    @Override
    public Service getService() {
        return this;
    }

    @Override
    public void performSync(Context context, Account account, Bundle extras,
        String authority, ContentProviderClient provider,
        SyncResult syncResult) {
```



```
        LeadDB lead = new LeadDB(context);
        OEHelper oe = lead.getOEInstance();
        if (oe != null) {
            if (oe.syncWithServer()) {
-                // TODO: Sync finished
+                Intent intent = new Intent(SyncFinishReceiver.SYNC_FINISH);
+                sendBroadcast(intent);
            }
        }
    }
}
```

Now, Run your application. It will first sync lead from server and when sync finish. Receiver got call and your listview will be reloaded.



### Download Demo application source

`file/openerp-mobile-crm.zip`



---

## Indices and tables

---

- `genindex`
- `search`