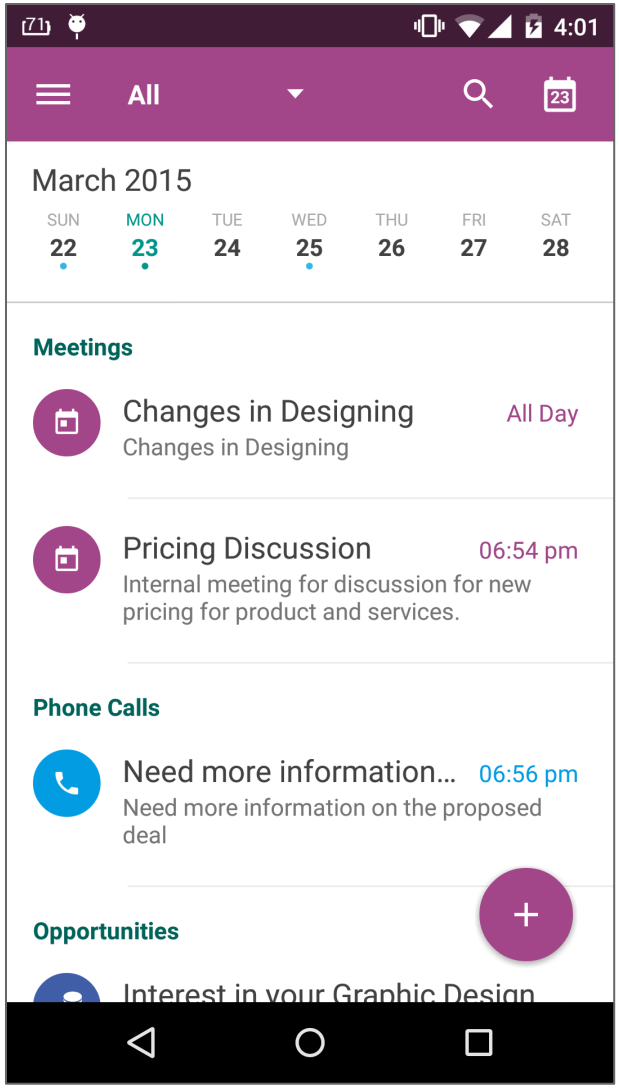
Odoo Mobile Development Framework & App build with it

Mantavya Gajjar - Director, Odoo India

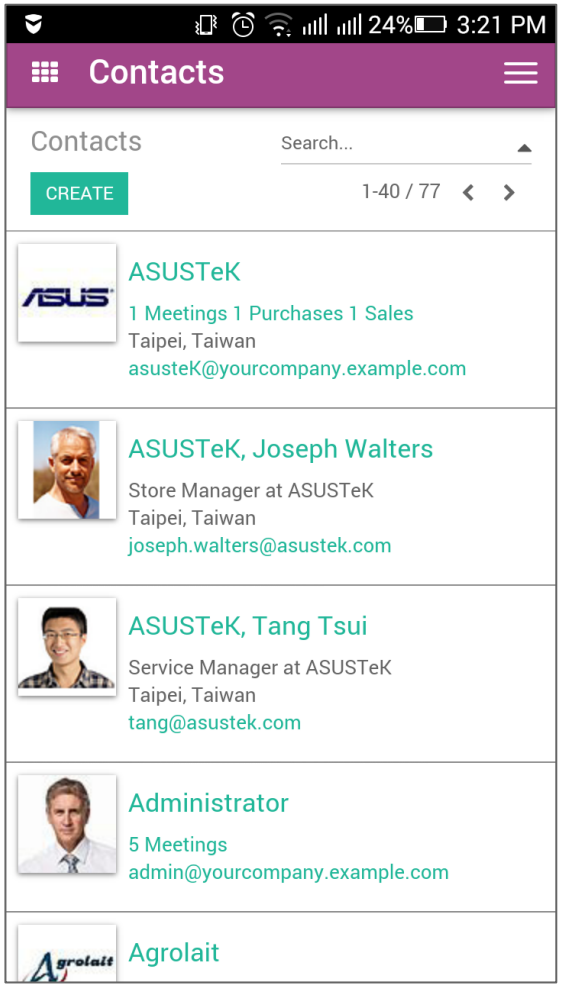
odoo

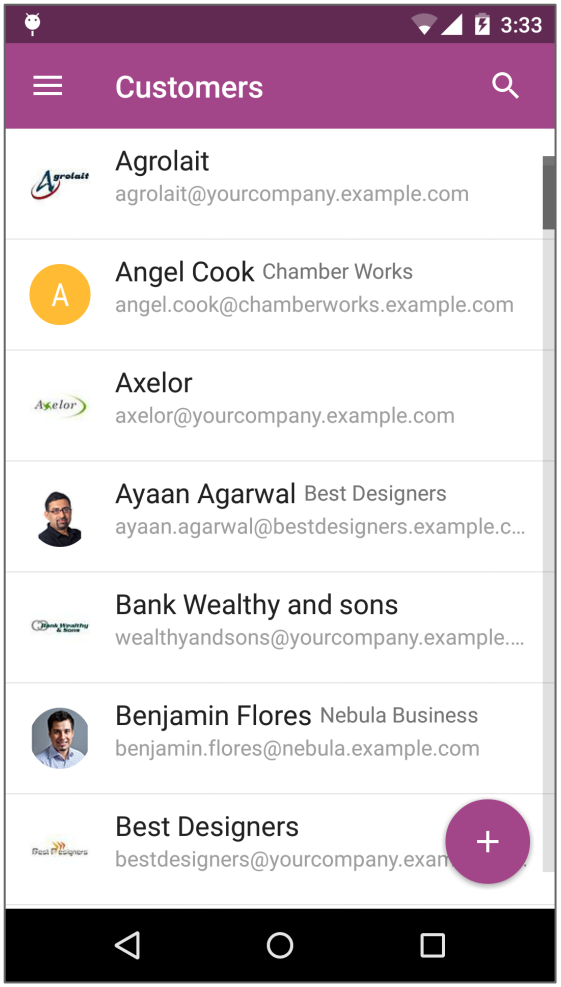
Agenda

- Introduction
- Setting Development Environment
  - Checkout Branch
  - Sample Addons
  - Creating Addons
- Framework in Details
  - model and fields
  - providers, services and utilities
  - fragments and activities
- Help and Support ?
- API Documentation



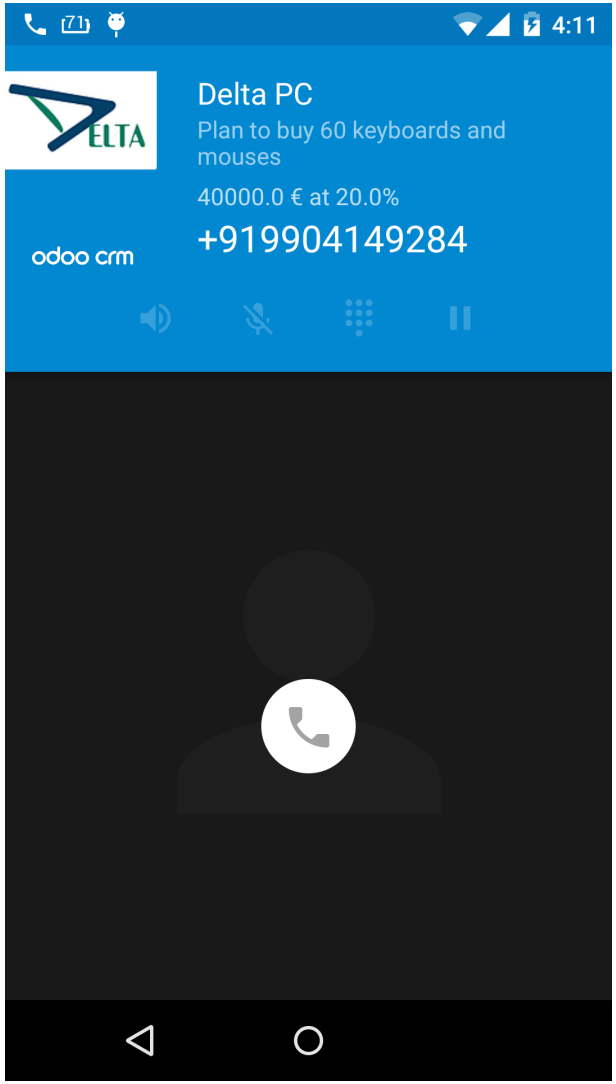
Introduction





Webview vs Native view

Introduction



Why Native Application is needed ?

- Hardware Access
- Native Views, Fast rendering
- Local Data Storage
- Offline Access to Data
- Instant Notifications

Introduction - 2 years *of development*

2012

Prototype model, developed

2013

Framework v1 development started

2015

Framework v2 with 4 official mobile apps

Real Time Example

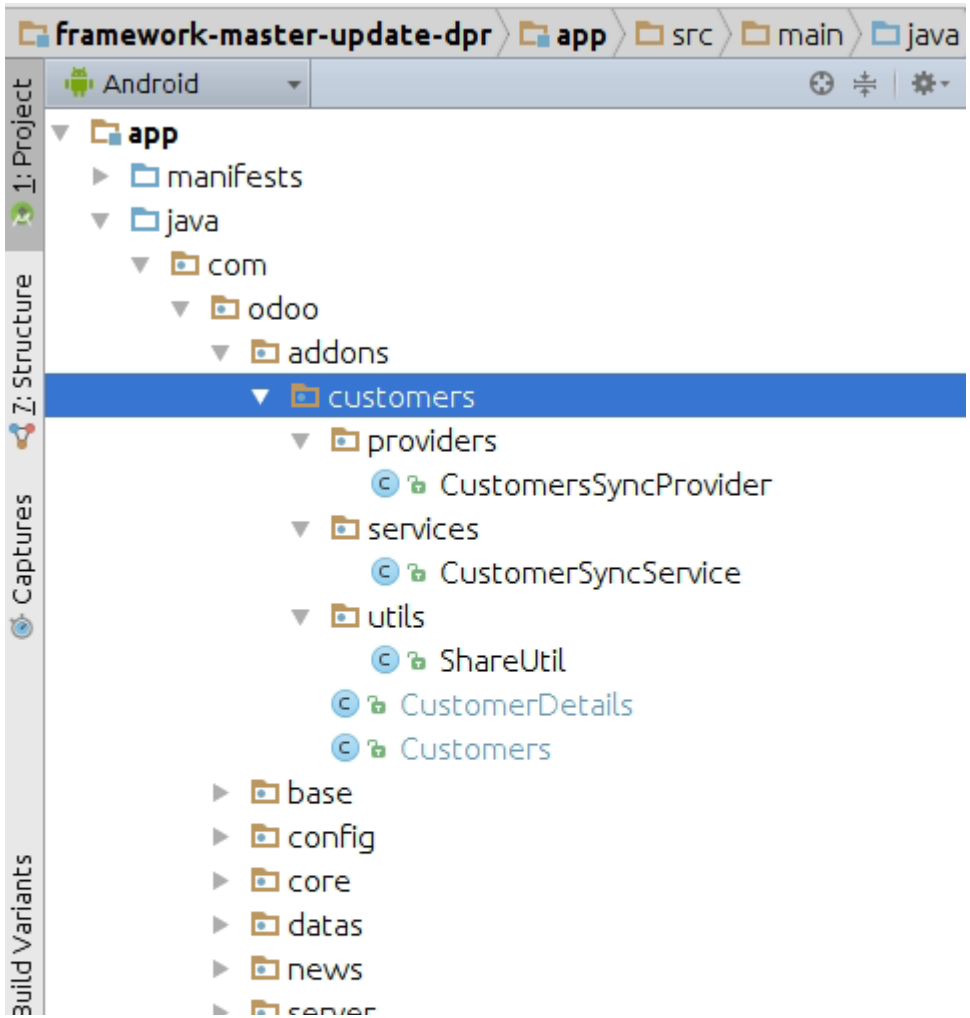
Odoo Experience App, Developed in 3 days

odoo

Setting Development Environment

- Download Android Development Studio from http://developer.android.com/sdk/index.html
- Checkout Framework Branch form https://github.com/Odoo-mobile/framework
- Create a new Project and Import an existing code http://mobile.odoo.co.in/v2/getting-started/downloading/downloading.html

Sample Addons



New addon will go to addons directory

- Models
- Fragments & Activities
- Providers
- Services
- Utilities

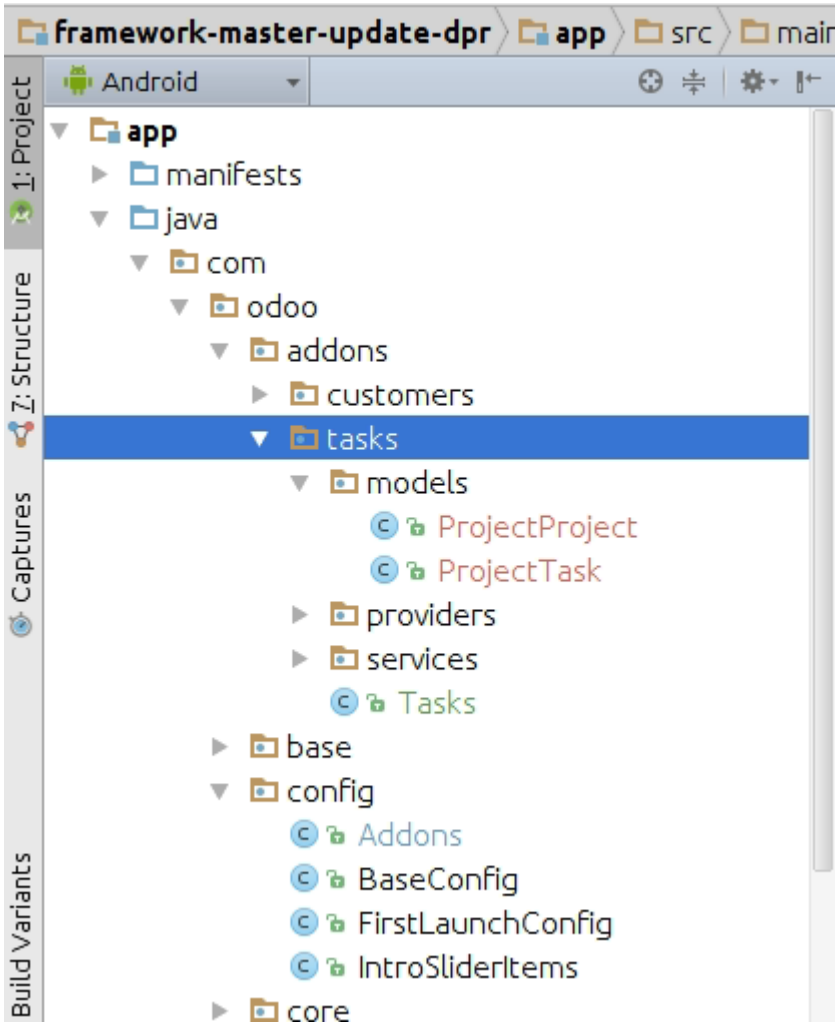
odoo

Framework in details

- Manage the Sync Service between Odoo Server and Mobile App
- Manage Database to provide the Offline support
- Standard Views and Controls
  - Form
  - List View
  - Chatter View
  - Parallax Effect on view
  - App Introduction view
- And they all are based on the Material Design

Let’s develop and run mobile application

Creating Addons



New addon will go to addons directory

- Models
- Fragments & Activities
- Providers
- Services
- Utilities

odoo

Models

```
1  public class ProjectProject extends OModel {
2      public static final String TAG = ProjectProject.class.getSimpleName();
3
4      OColumn name = new OColumn("Name", OVarchar.class).setSize(100);
5
6      public ProjectProject(Context context, OUser user) {
7          super(context, "project.project", user);
8      }
9  }
10 public class ProjectTask extends OModel {
11     public static final String TAG = ProjectTask.class.getSimpleName();
12     public static final String AUTHORITY = "com.odoo.addons.projects.project_tasks";
13
14     OColumn name = new OColumn("Name", OVarchar.class).setSize(100);
15     OColumn project_id = new OColumn("Project", ProjectProject.class, OColumn.RelationType.ManyToOne);
16     OColumn description = new OColumn("Description", OText.class);
17
18     public ProjectTask(Context context, OUser user) {
19         super(context, "project.task", user);
20     }
21
22     @Override
23     public Uri uri() {
24         return buildURI(AUTHORITY);
25     }
26 }
27
28
```

odoo

Fields

OVarchar

A string of limited length. Default length : 64

```
OColumn name = new OColumn("Name", OVarchar.class).setSize(100).setRequired(
);
```

OInteger

An integer

```
OColumn counter = new OColumn("Counter", OInteger.class);
```

OBoolean

A boolean (true, false). Default false

```
OColumn is_active = new OColumn("Active", OBoolean.class);
```

OFloat

A floating point number.

```
OColumn weight = new OColumn("Weight", OFloat.class);
```

odoo

Fields

OText

A text field with no limit in length.

```
OColumn body = new OColumn("Message body", OText.class);
```

OHtml

A html (actual text) field with no limit in length.

```
OColumn body = new OColumn("Message body", OHtml.class);
```

ODate

A date. Stores `yyyy-MM-dd` formatted date or `false` if value not set

```
OColumn date = new OColumn("Date", ODate.class);
```

ODateTime

Allows to store a date and the time of day in the same field. Stores `yyyy-MM-dd HH:mm:ss` formatted date or `false` if value not set

```
OColumn date = new OColumn("Date", ODateTime.class);
```

odoo

Fields

OBlob

Allows to store a base64 data in database. Generally used by ir.attachment

```
OColumn image = new OColumn("Avatar", OBlob.class);
```

OSelection

Allows to store a string value (i.e., key for selection). Used selection for parsing Label for stored key.

```
OColumn state = new OColumn("State", OSelection.class)
        .addSelection("draft","Draft")
        .addSelection("confirm","Confirmed")
        .addSelection("close","Canceled")
        .addSelection("done","Done");
```

OTimestamp

Stores current date time to column.

```
OColumn order_date = new OColumn("Order date", OTimestamp.class);
```

odoo

Relational Fields

OneToMany

One2many field; the value of such a field is the recordset of all the records in comodel_name such that the field inverse_name is equal to the current record.

It required Type as another model's class type and also required realted column (as ManyToOne in related model)

```
OColumn parent_id = new OColumn("Company", ResPartner.class, RelationType.Many
ToOne);
OColumn child_ids = new OColumn("Contacts", ResPartner.class, RelationType.One
ToMany).
        setRelatedColumn("parent_id");
```

ManyToOne

The value of such a field is a recordset of size 0 (no record) or 1 (a single record).

```
OColumn parent_id = new OColumn("Company", ResPartner.class, RelationType.Many
ToOne);
```

ManyToMany

Many2many field; the value of such a field is the recordset.

```
OColumn tag_ids = new OColumn("Tags", NoteTag.class, RelationType.ManyToMany);
```

odoo

Sync Service

```
1  public class TasksSyncService extends OSyncService {
2      public static final String TAG = TasksSyncService.class.getSimpleName();
3
4      @Override
5      public OSyncAdapter getSyncAdapter(OSyncService service, Context context) {
6          return new OSyncAdapter(getApplicationContext(), ProjectTask.class, this, true);
7      }
8
9      @Override
10     public void performDataSync(OSyncAdapter adapter, Bundle extras, OUser user) {
11                 adapter.syncDataLimit(80);
12     }
13 }
```

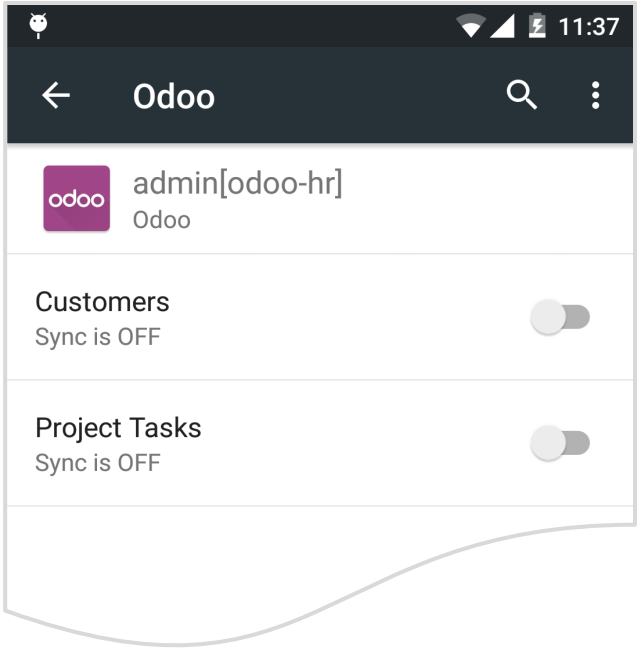
Just create sync service and register it in application manifest file.

Model Provider

```java
public class ProjectTaskProvider extends BaseModelProvider {
    public static final String TAG = ProjectTaskProvider.class.getSimpleName();

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

You just need to extend the class and provide some normal configuration and your sync service will ready to get data from server.



odoo

Fragments

```java
public class Tasks extends BaseFragment {
    public static final String TAG = Tasks.class.getSimpleName();

    @Override
    public List<ODrawerItem> drawerMenus(Context context) {
        List<ODrawerItem> menu = new ArrayList<>();
        menu.add(new ODrawerItem(TAG).setTitle("Tasks").setInstance(new Tasks()));
        return menu;
    }

    @Override
    public Class<ProjectTask> database() {
        return ProjectTask.class;
    }
}
```

Fragment with drawer menus and default database model class.

odoo

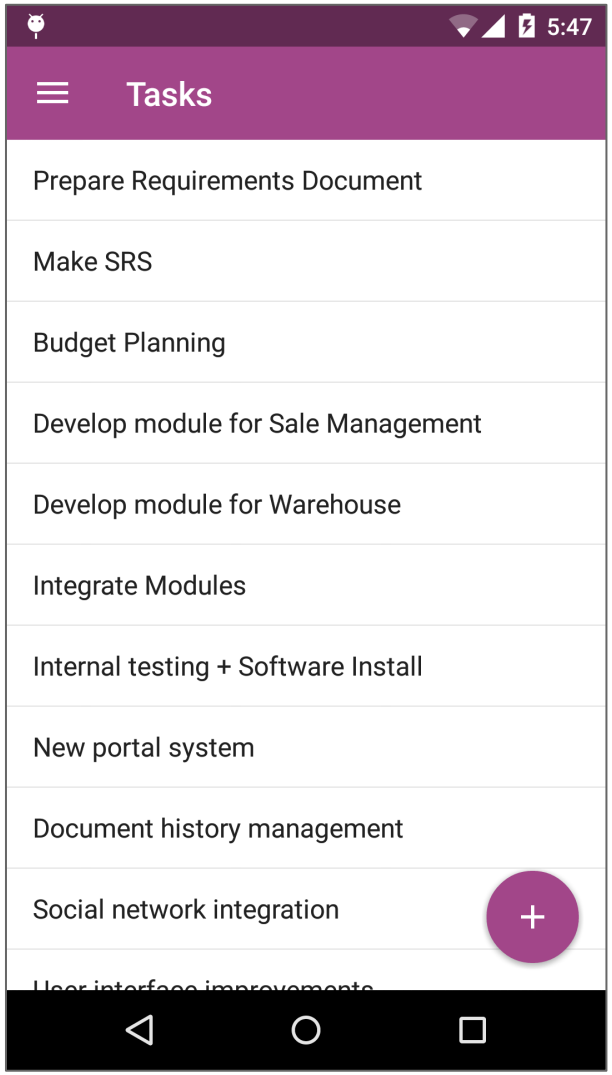
Registry

```
public class Addons extends AddonsHelper {

    OAddon customers = new OAddon(Customers.class).setDefault();
    OAddon tasks = new OAddon(Tasks.class);
}
```

- Entry point registry for each addons (i.e. customers, tasks)
- It will also init, upgrades the data structure (db) on first app launch and when app upgrades
- A default entry point for the app to be execute by calling setDefault() method

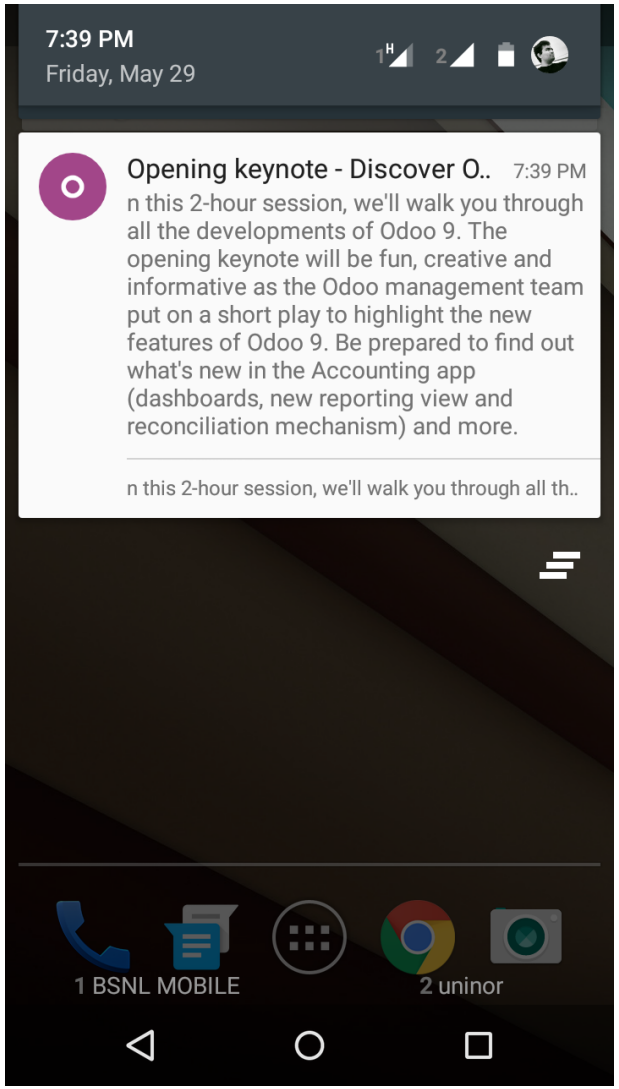
Let’s jump to the Demo



This is how it looks, supports

- Native Views, Fast rendering
- Local Data Storage
- Offline data access
- Auto / manual sync

Quick Notification



Apps Developed with framework

25+ more apps ...

Thank you

Odoo
sales@odoo.com
www.odoo.com

R&D and Services office
Chaussée de Namur 40 B-1367 Grand-Rosière

Sales office
Avenue Van Nieuwenhuyse 5 B-1160 Brussels