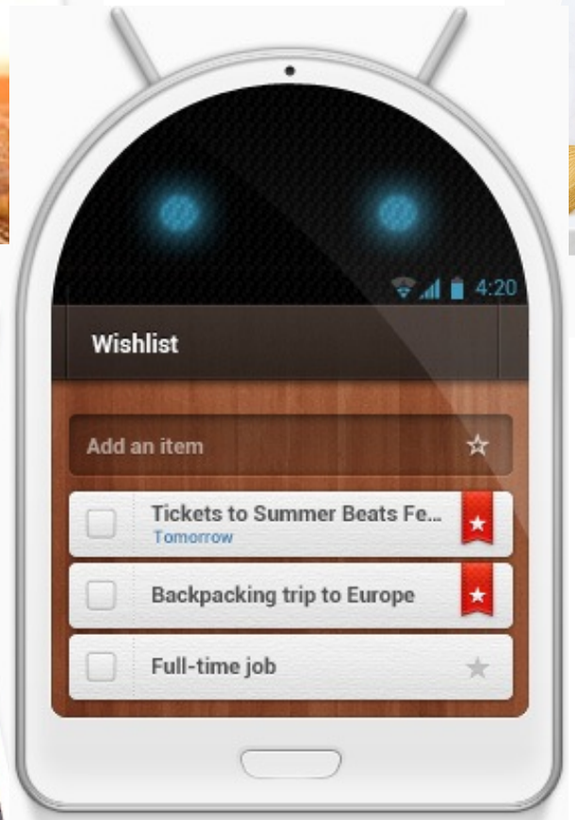
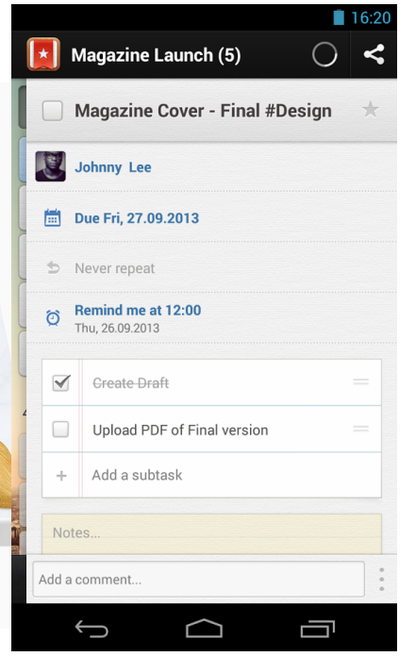
Building Wunderlist. Lessons learned.

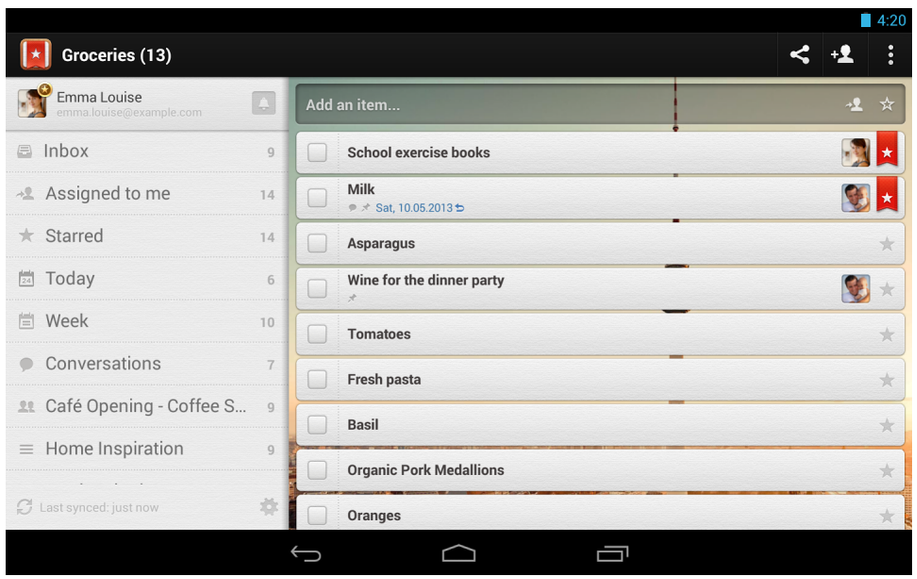
Cesar Valiente

wunderkinder

6Wunderkinder and Wunderlist

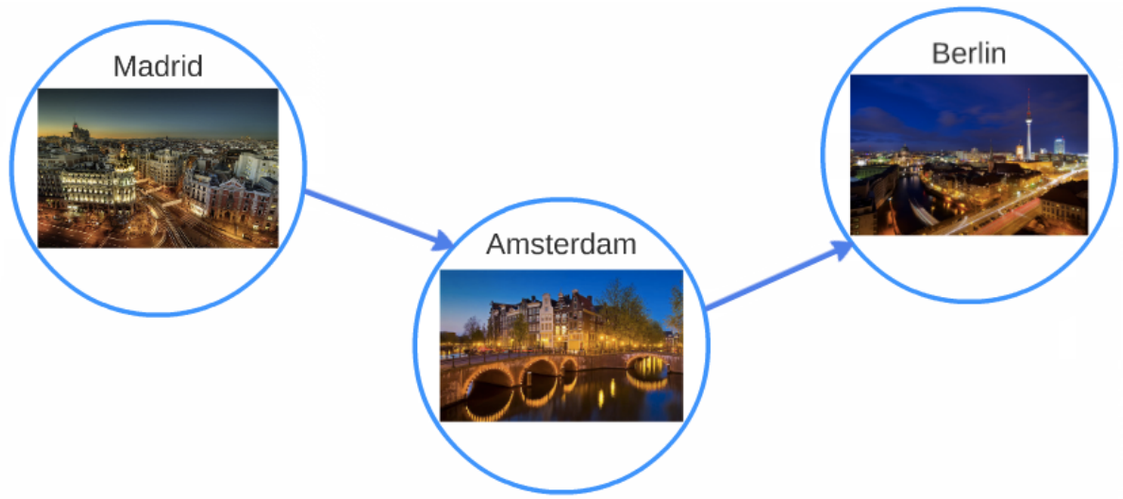






Me :-)

Cesar Valiente



https://twitter.com/CesarValiente

https://plus.google.com/u/0/+CesarValiente/

https://github.com/CesarValiente

1. Databases

1.1 Opening modes

- *getReadableDatabase* vs *getWritableDatabase*.

- Just *getWritableDatabase* mode will call onCreate, onOpen, onUpgrade, etc.

- Both methods may takes long time —> avoid main thread.

1.2 Transactions

- *onUpgrade* executes its own transaction.

- Be careful with the exceptions, if something happens, your changes won't be committed!!

1.2.1 Transactions

```
@Override
   public void onUpgrade(final SQLiteDatabase db, final int oldVersion,
           final int newVersion) {

      switch (oldVersion + 1) {
      case 2:
        //Do whatever you want to do
      case 3:
         //Here you have an exception, then your code won't be commited,
         //even the previos case, that was correct.
      }
}
```

1.2.2 Transactions

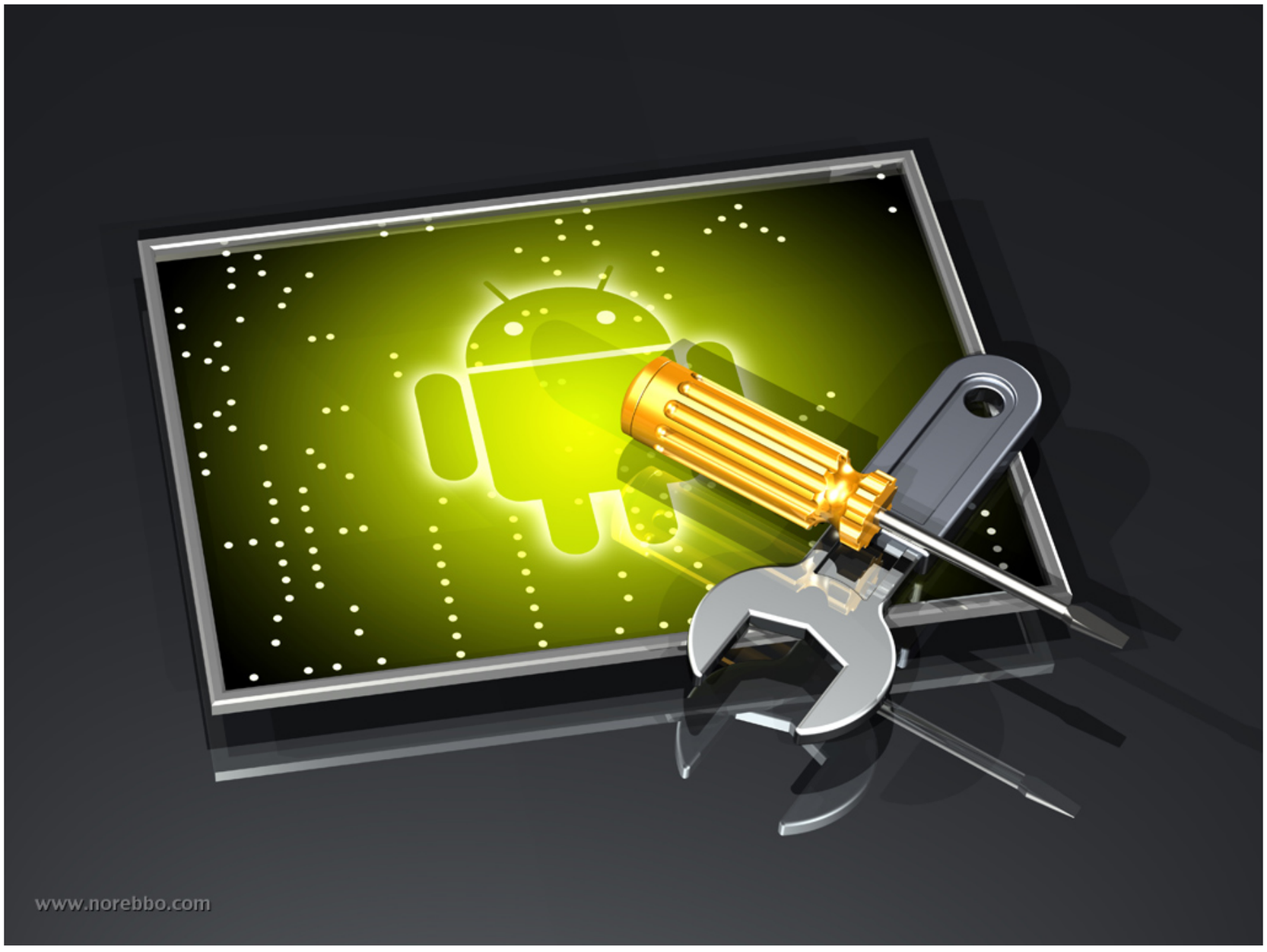
```
@Override
   public void onUpgrade(final SQLiteDatabase db, final int oldVersion,
           final int newVersion) {
       switch (oldVersion + 1) {

       case 2:
           db.beginTransaction();
           try {
              // Do whatever you have to do
               db.setTransactionSuccessful();
           }finally {
               db.endTransaction();
           }
       case 3:
           db.beginTransaction();
           try {
               //This code throws an Excpetion, then all the changes (also the changes
               //created by the previous one won't be commited)
               db.setTransactionSuccessful();
           }catch (Exception e) {
            e.printStackTrace();
           }finally {
               db.endTransaction();
           }
       }
}
```

13 Simple queries

- Use *simpleQueryFor"Xxx"* instead making a *rawQuery* to get simple stuff.

```
public long getNumItems() {
try {
      SQLiteStatement statement = db.compileStatement(YOUR_QUERY);
      return statement.simpleQueryForLong();
    } catch (SQLiteDoneException e) {
        return 0;
    }
}

public long getNumItemsNotSimple () {

    Cursor cursor = db.rawQuery(YOUR_QUERY, null);
    int count = 0;
    if (cursor != null && cursor.getCount() > 0) {
      cursor.moveToFirst();
      count = cursor.getInt(0);
      cursor.close();
    }
    return count;
}
```

2. Android internals



2.1 AsyncTasks and multithreading

- Different ways to work on different Android versions:
  - < 1.6 executed serially.
  - 1.6 < 3.0 uses a pool of threads allowing tasks in parallel.
  - Since 3.0, again, back to execution on a single thread.

2.1.1 AsyncTasks and multithreading

- Try to update the UI as soon as possible.
- Use AsyncTasks just for short time operations.
- For longer operations use java.util.concurrent package such as Executor, ThreadPoolExecutor and FutureTask.
- Concurrency management —> JavaRX, Spotify Trickle, etc.

2.2 Broadcast receivers

- *BroadcastReceiver* just valid for the duration of the call *onReceive*.

- Use *LocalBroadcastManager* (support Lib v4), or *setPackage* (ics onwards).

2.2.1 Broadcast receivers

```
private void registerBroadcast () {

        mIntentFilter intentFilter = new IntentFilter();
        mIntentFilter.addAction(YOUR_ACTION);
        mIntentFilter.addCategory(YOUR_CATEGORY);

        mReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {

                //Do whatever you want to do
                //But once this is finished no longer will live anything that was here,
                //so don't make an asynchronous operation that has to come back here

                //Use NotificationManager
            }
        };

        //---- Use one of those ---//

        //Using android.support.v4
        LocalBroadcastManager.getInstance(this).registerReceiver(mReceiver, mIntentFilter);

        //Using the usual way
        registerReceiver(mReceiver, intentFilter);
}
```

2.2.2 Broadcast receivers

```
private void sendBroadcastIntent () {

    // Create a local broadcast Intent
    Intent broadcastIntent = new Intent();
    broadcastIntent.setAction(YOUR_ACTION);
    broadcastIntent.addCategory(YOUR_CATEGORY);

    //---- Use one of those ---//

    //Using LocalBroadcastMamanger
    LocalBroadcastManager.getInstance(this).sendBroadcast(broadcastIntent);

    //Usual way
    //If you are not using LocalBroadcastManager you can define
    //the package to add security (since ICS)
    broadcastIntent.setPackage(YOUR_PACKAGE);
    sendBroadcast(broadcastIntent);
}
```

2.3 Services

- Services run in the main thread of its hosting process.

- When the service has finished it should stop itself.

- Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

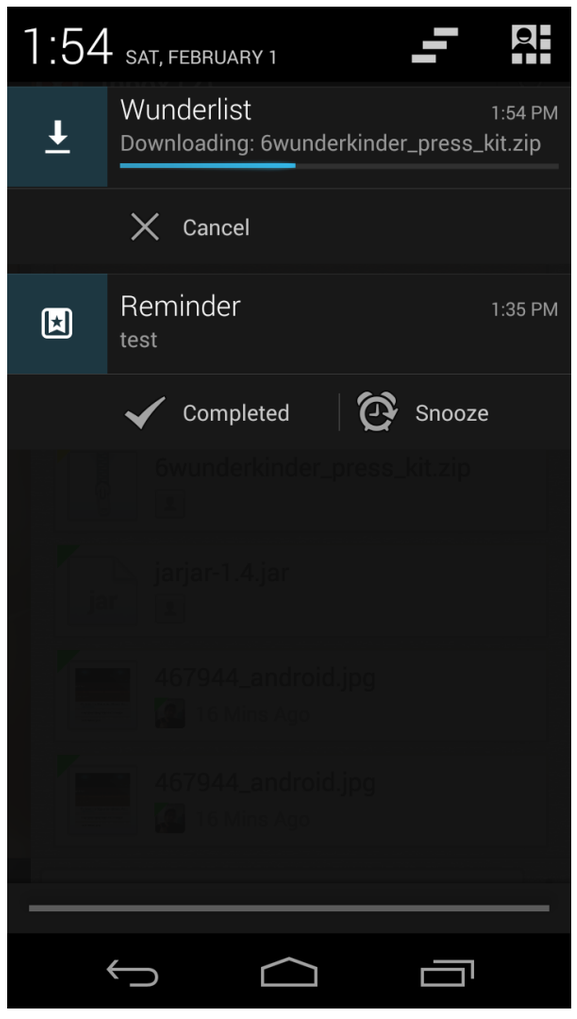
2.4 IntentServices

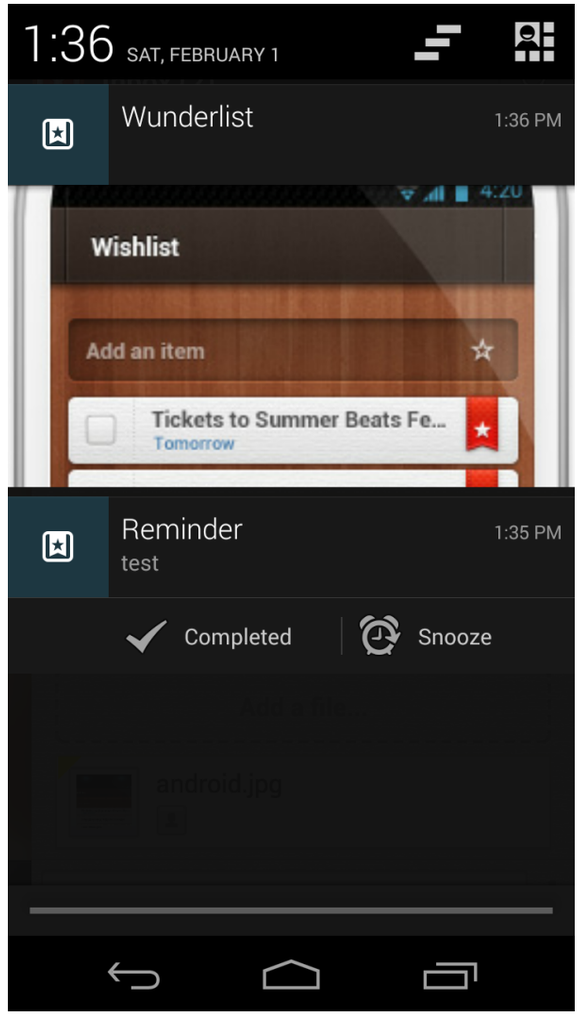
- IntentServices are handled on a single worker thread.

```
public class MyIntentService extends IntentService {

    /**
     * Creates an IntentService.  Invoked by your subclass's constructor.
     *
     * @param name Used to name the worker thread, important only for debugging.
     */
    public MyIntentService(String name) {
        super(name);
    }

    @Override
    protected void onHandleIntent(Intent intent) {

        //Do whatever you want to do, once you have finished
    }
}
```

2.5 Notifications

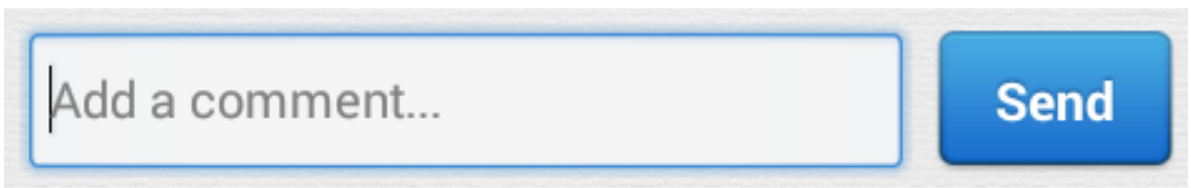
- Different notifications styles.
- Respect versions styles.
- Be careful updating the notification, it can give you problems, delays, etc.





2.6 Keyboard issues

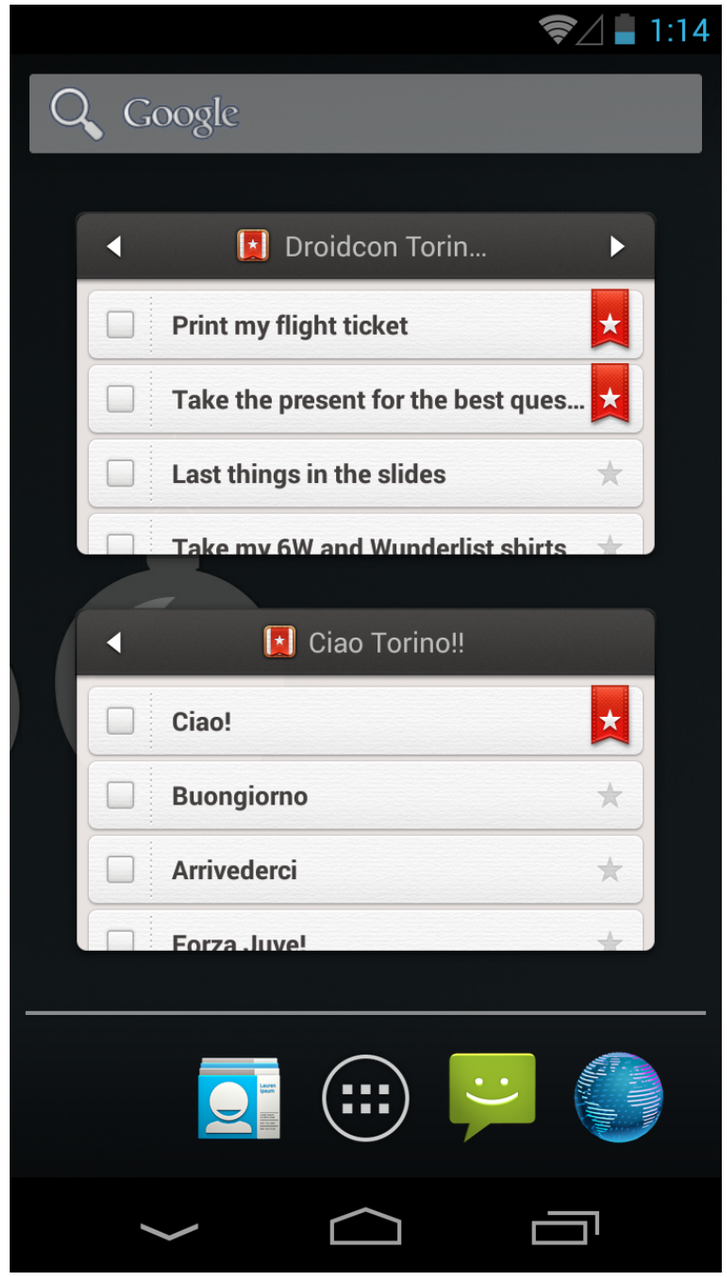
- Try to use "*send*" as *"a way to accept the text"*.
- Try to offer, always you can, a way to *accept* the text using the UI.



- Avoid to use EditText in ListViews.

2.7 Widgets

- Different widget versions.
- Try to offer the “same experience” as much as you can.
- Several widget instances is something useful.
- Use widget ids and save some data in a persistent storage.



3. Android and... Java!

3.1 Caches

- Try to cache your data as much as you can.
- Use Lists and Maps.
- Avoid “infinite sizes”, follow some strategy.
- Keep it updated.

3.2 For loops

- Two different *"for loop types"*, conventional (old school) and enhanced.

```
int arrayInts [] = new int[ITEMS];
int length = arrayInts.length;
for (int i=0; i<length; i++) {
  System.out.print(arrayInts[i]);
}
```

```
for (int index:arrayInts) {
  System.out.print(arrayInts[index]);
}
```

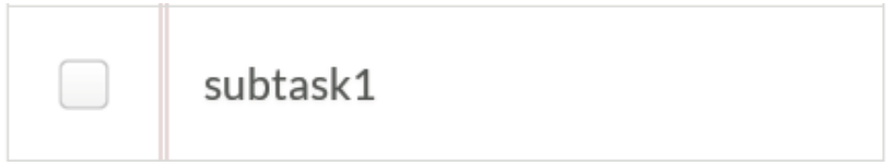
- Think carefully which one are you going to use. Performance differences.

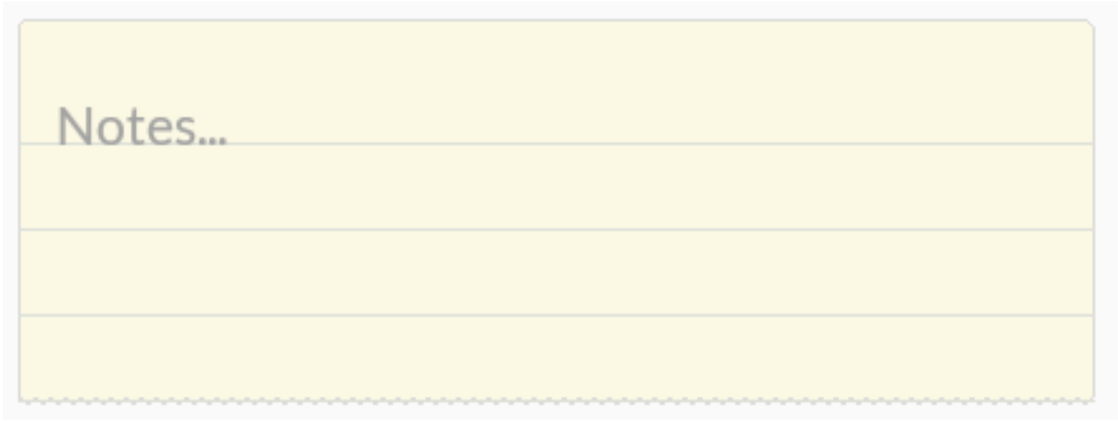
4. UI and Ux

4.1 Graphics, assets,...

- You can create a lot of different stuff by code/xml, so if you can,... do it!! :-)

- Some examples:



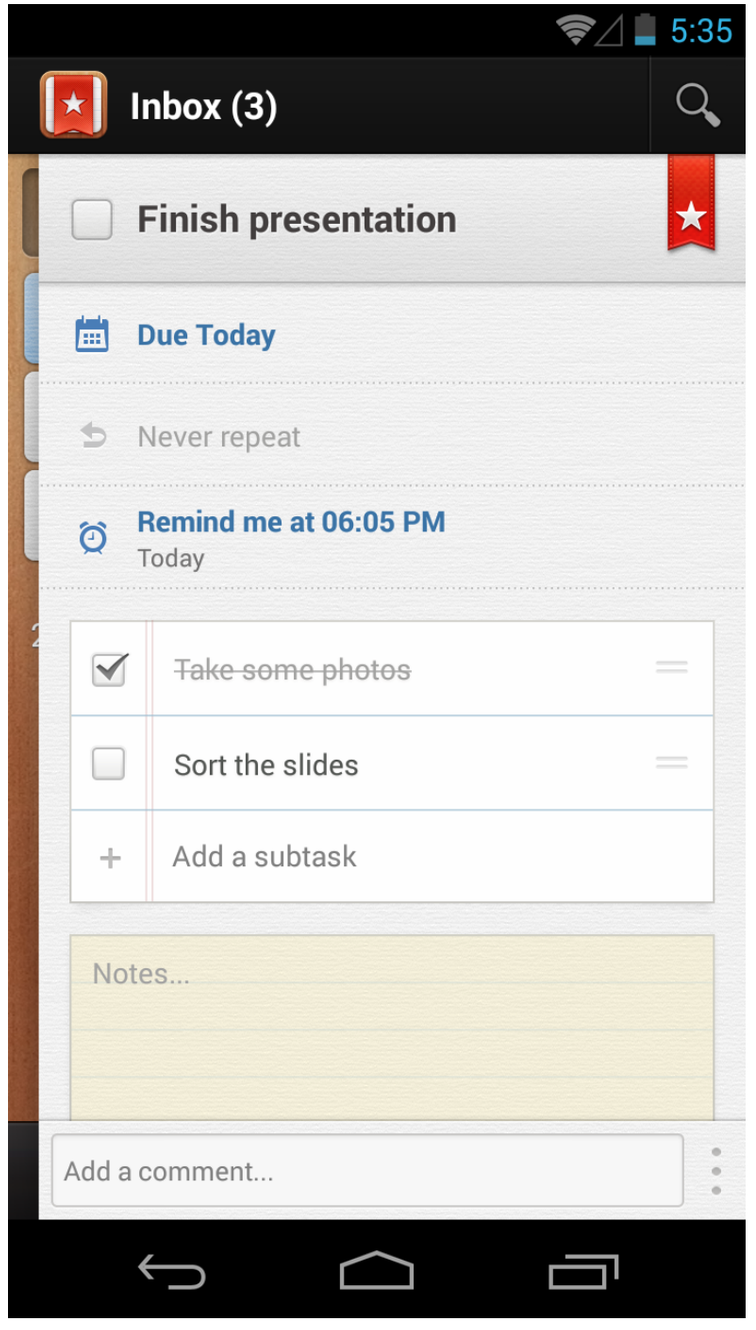


4.2 Make your own views!

- SlidingLayerLib on Github: released under Apache License v2.

- Forked and starred hundred of times.

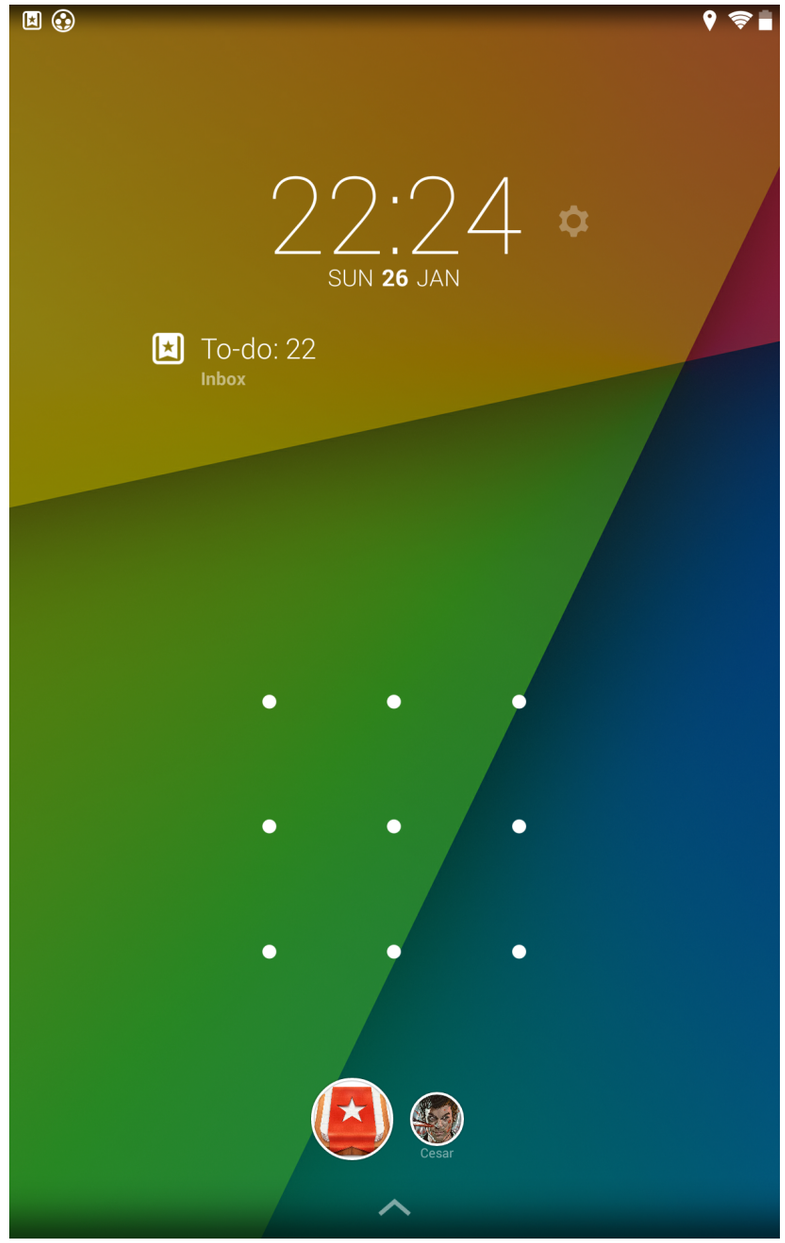
- Improved since we released it by the community.



5. Android mix

5.1 DashClock Widget extension

- Useful to show info.
- If you don’t have anything to show, then don’t show it, save space for other notifications.
- Thanks to Jake Wharton for the comment/tip :-).

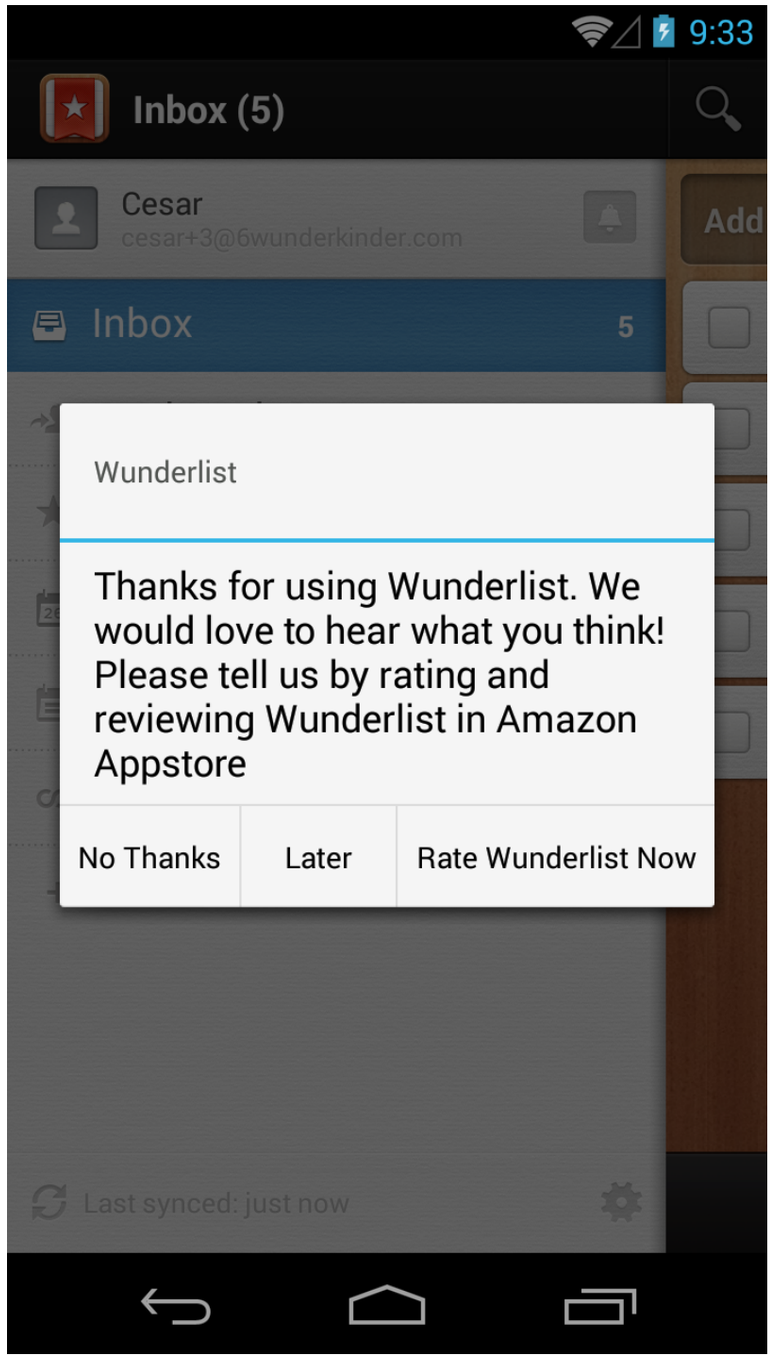


5.2 Rating dialog

- Fact: people usually just rate to give you a bad one.

- Much more good ratings with a rating dialog than without it.

- Don’t force the user to vote.



5.3 Open Source

- Try to keep up to date about the “Android Open Source world”.
- Don’t reinvent the wheel.
- Collaborate if you want/can.

5.3.1 Open Source

- Be careful with licenses —> Really important! —> Law problems!
- *Privative Licenses.*
- *Open Source Licenses:*
    - Reciprocal licenses (strong copyleft):
        - GNU GPL, Eclipse Public License, Affero License, etc.
    - Partially closable licenses (weak copyleft):
        - LGPL, Mozilla License, etc.
    - Permissive licenses:
        - BSD, MIT, Apache License v2, etc.
- *Public domain.*

5.4 Managing several apps

- Several apps to manage, Google Play, Google For Education, Amazon App store, Samsung, etc.
- *Gradle* is your friend! ;-)
- Use *flavors*.

5.4.1 Managing several apps

```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
     //your stuff
    }
}
apply plugin: 'android'

repositories {
    mavenLocal()
    mavenCentral()
}
android {
 //your stuff
}
signingConfigs {
 release {
  //Your stuff
  }
}
```

```
buildTypes {
 release {
  //Your stuff
  }
}
productFlavors {
        beta {
            //Your stuff
        }
        production {
            //Your stuff
        }
        edu {
         //Your stuff
        }
}
dependencies {
 //Your stuff
}
```

5.5 Testing and Continuous Integration

- UI testing: Robotium, UIAutomator, Espresso, Calabash, etc.
- Others: JUnit, Robolectric, etc.
- Really important/useful —> C.I. software like Jenkins, Hudson, Travis, etc.

5.6 Tools we use

Questions?

Thank you!! :-)

License