Master in Computer Engineering

Information Management Mobile Devices

Introduction to Android platform

Javier Abad (abad@decsai.ugr.es)

Dept. Of Computer Science and Artificial Intelligence

http://decsai.ugr.es

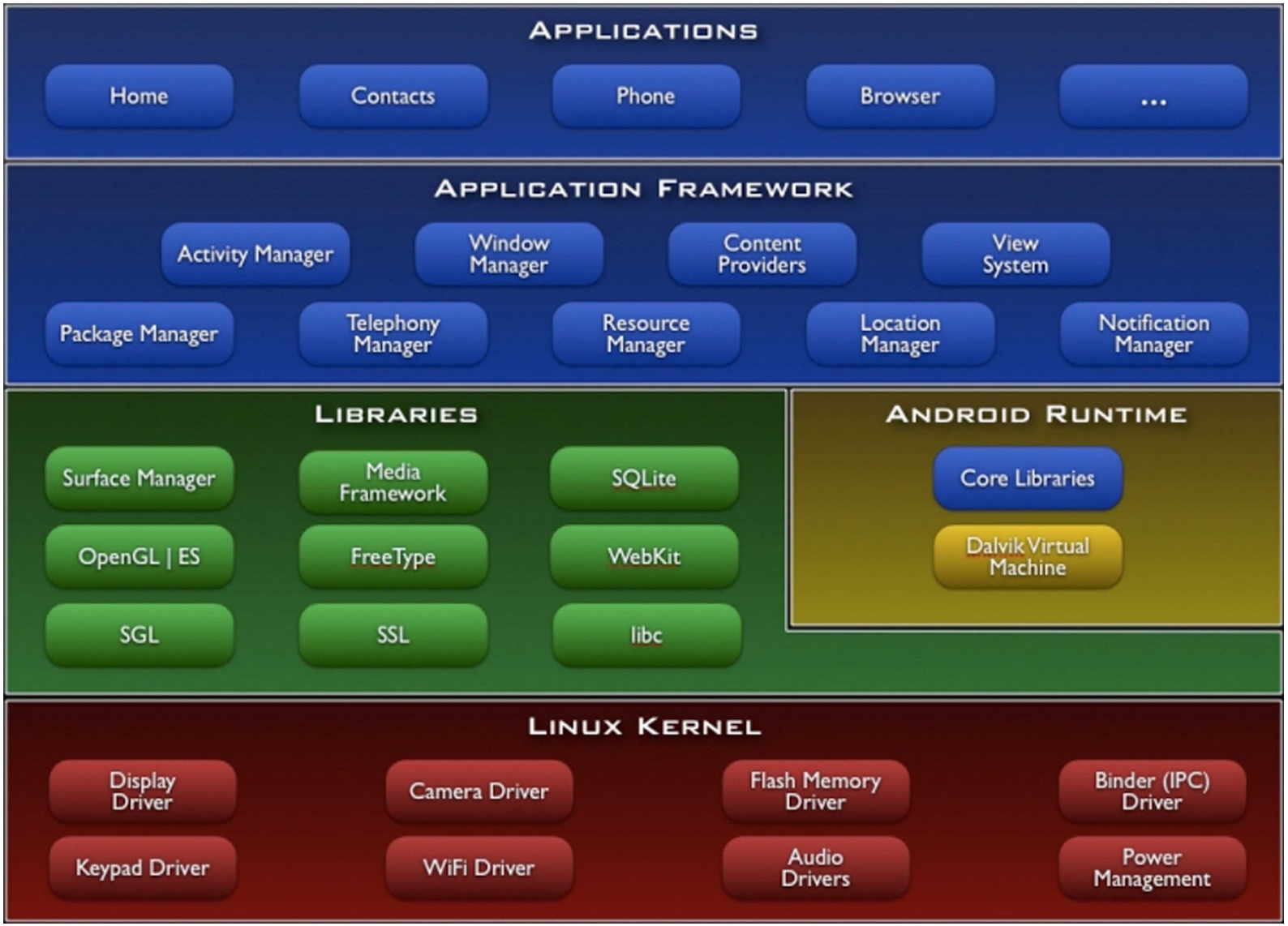
The Android platform

- Software package designed primarily, but not exclusively, to support (phones and tablets) mobile devices.

- Structured layers: OS kernel, libraries, application framework and basic applications.
- Software Development Kit (SDK) Development Tools Android applications.
- Documentation: tutorials, blogs, examples, etc.

Android Developers: https://developer.android.com

The Linux kernel



The Linux kernel

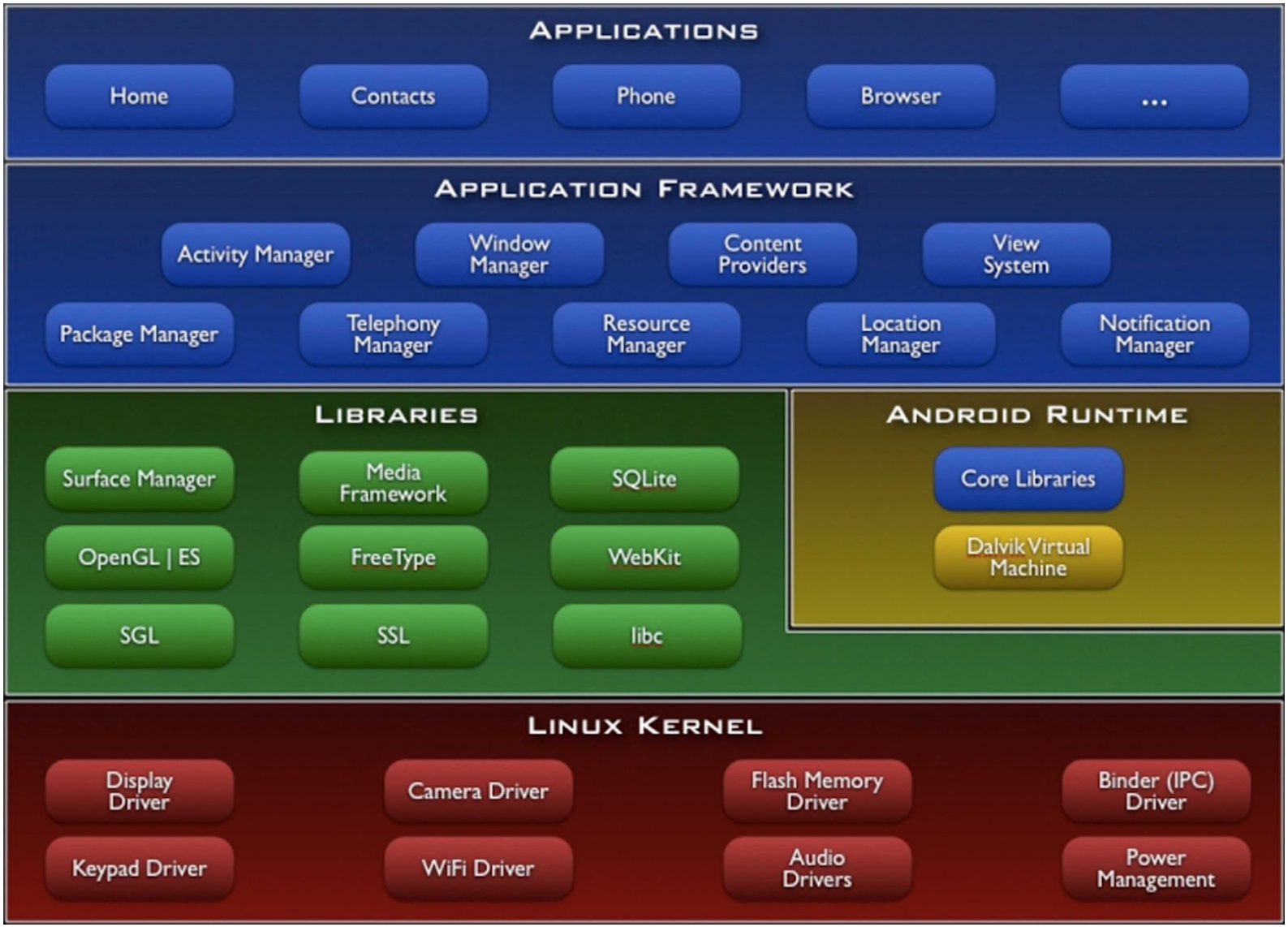
- It provides the basic services that all depends Android device.

- Standard services:
  - Security
  - Memory Management
  - Process management
  - E / S to files and network
  - Device Drivers

The Linux kernel

- Android-specific services:
  - Power Management
  - Memory Management
    - Memory sharing
    - Process Eliminator
  - Interprocess communication mechanism
  - Others...

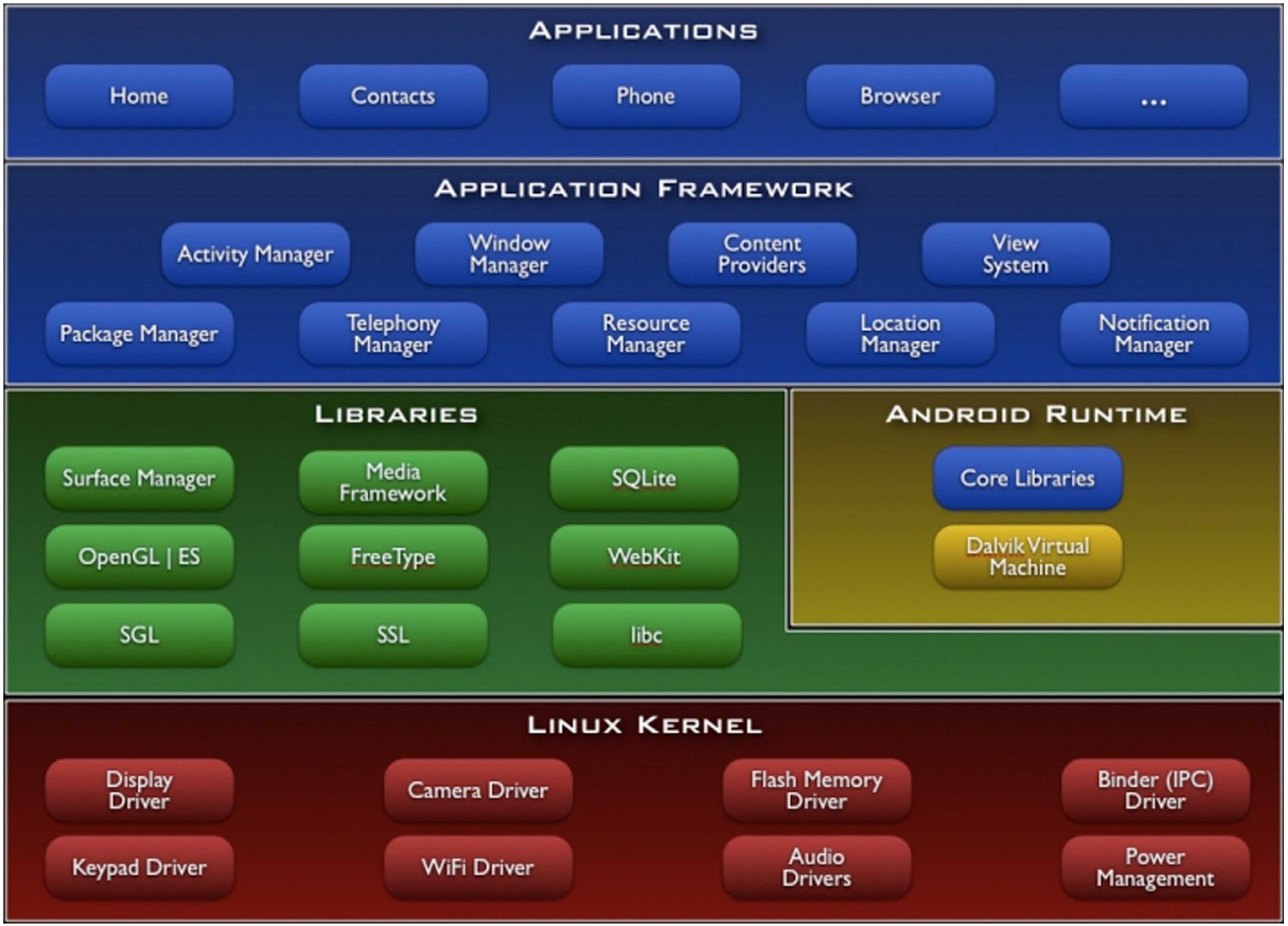
libraries


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

libraries

- System C library: Bionic libc
- Surface Manager: screen update
- Media Framework: audio / video
- Webkit: browser engine
- OpenGL graphics engine
- SQLite, manage relational databases

Android Runtime (Execution System)



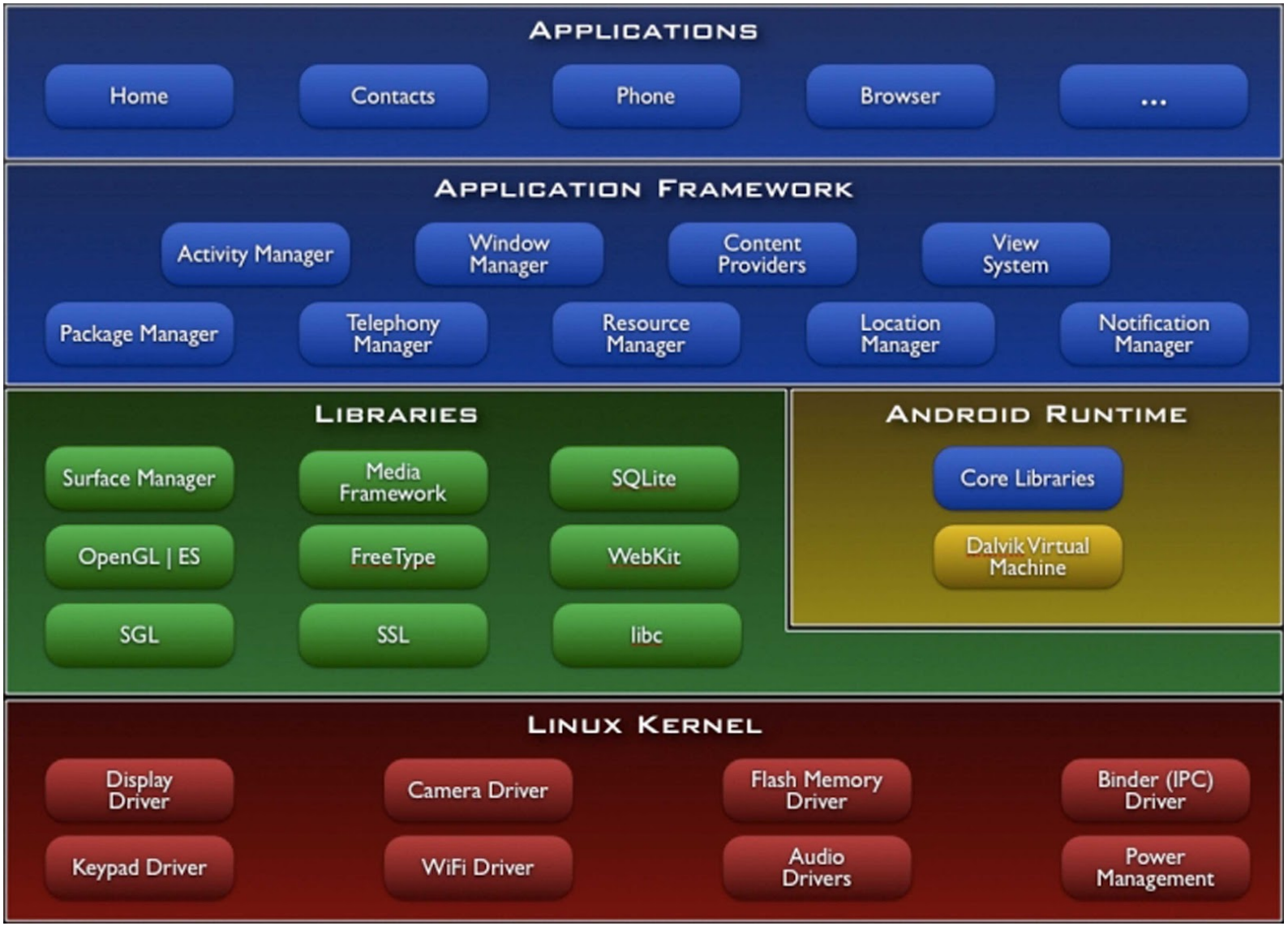
Android Runtime (Execution System)

- Java core libraries
  - Basic classes Java - java *, javax. *.
  - Lifecycle - android *.
  - Services Internet / Web - org *.
  - Unit Testing - JUnit *.
- Dalvik Virtual Machine (*Dalvik Virtual Machine - DVM)*
  - Android applications run

Android Runtime (Execution System)

- Typical workflow
  - application written in Java
  - Compilation - Java bytecode files
  - DX converts Java bytecode files into a single file dex bytecode (classes.dex)
  - MVD executes the file classes.dex
- MVD is designed to work with limited resources
  - Less powerful CPU
  - less memory
  - limited battery
- Dalvik VM Internals (Dan Bornstein) [Google I / O 2008]
  https://sites.google.com/site/io/dalvik-vm-internals

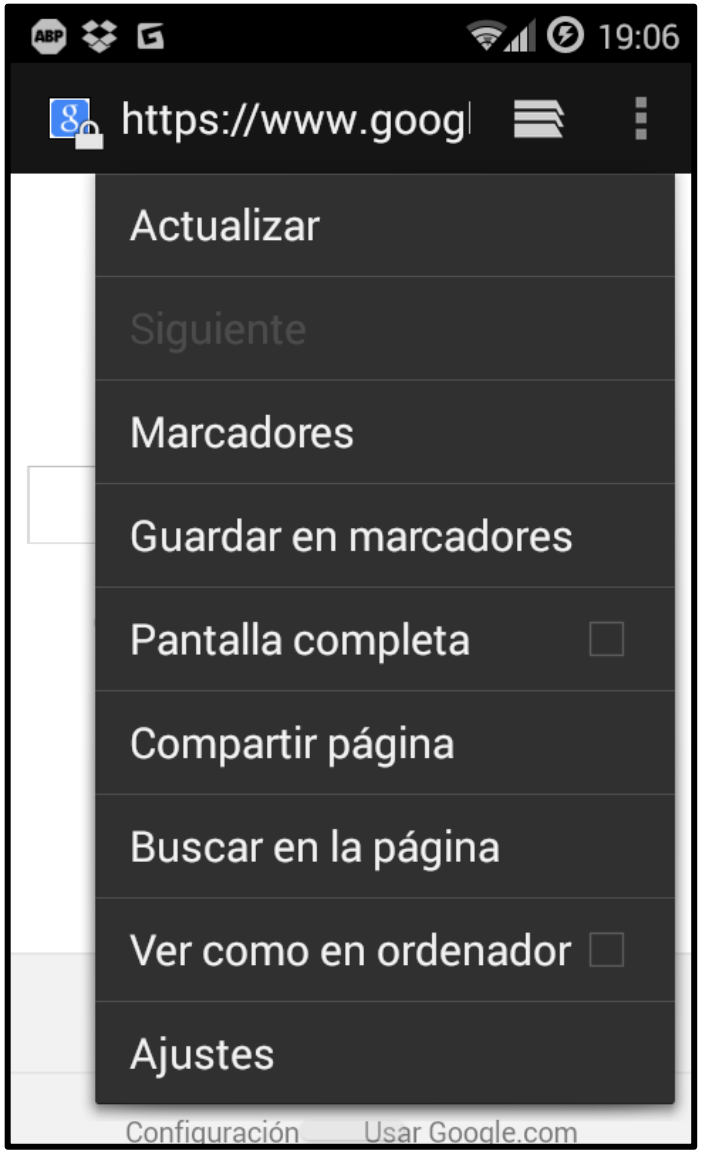
The application framework



The application framework

- Package manager (*Package Manager)*
  - It keeps track of applications installed on the device
- Window manager (*Window Manager)*
  - Manages windows forming an application

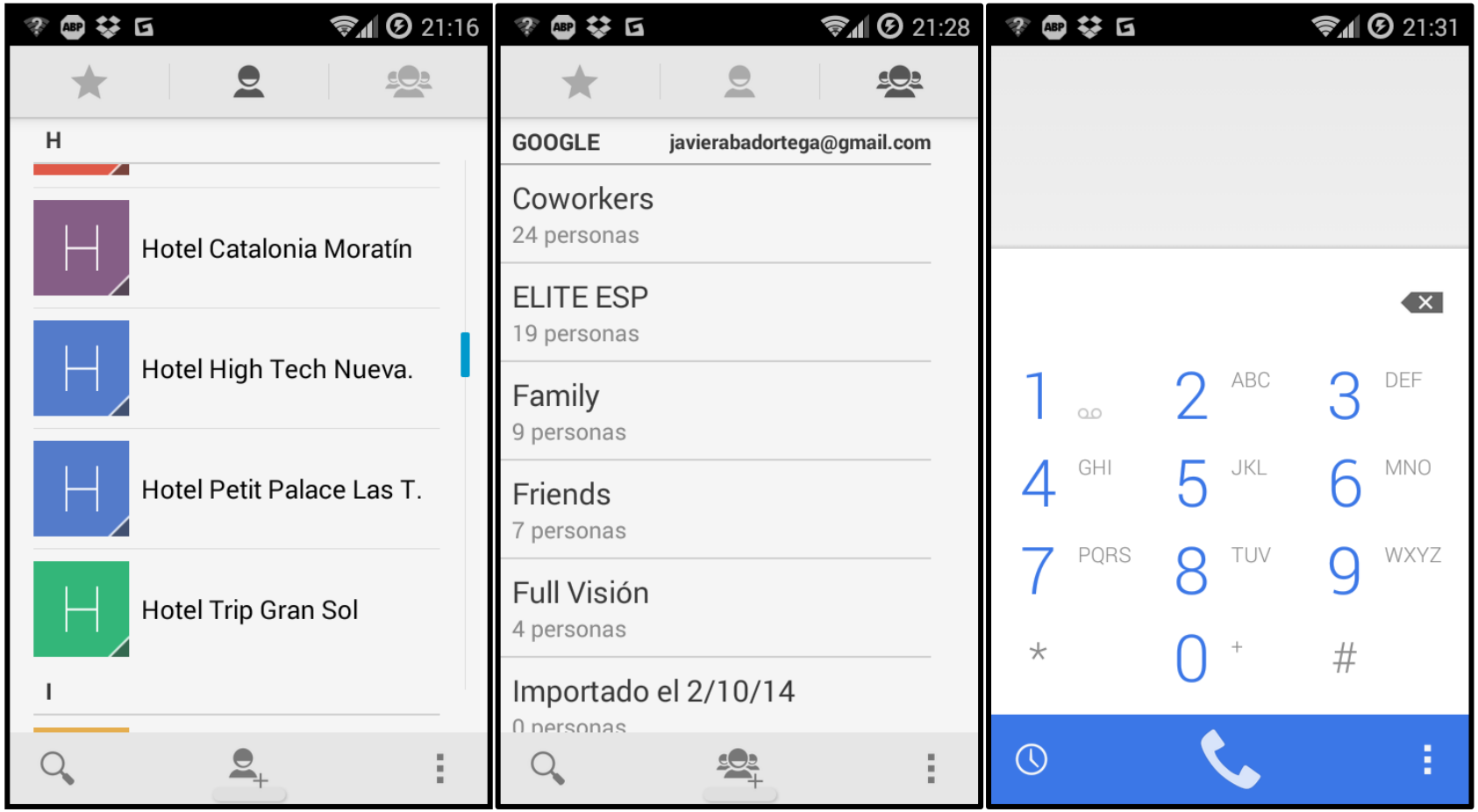
The application framework



The application framework

- System View (System View)
  - It provides common elements of the user interface

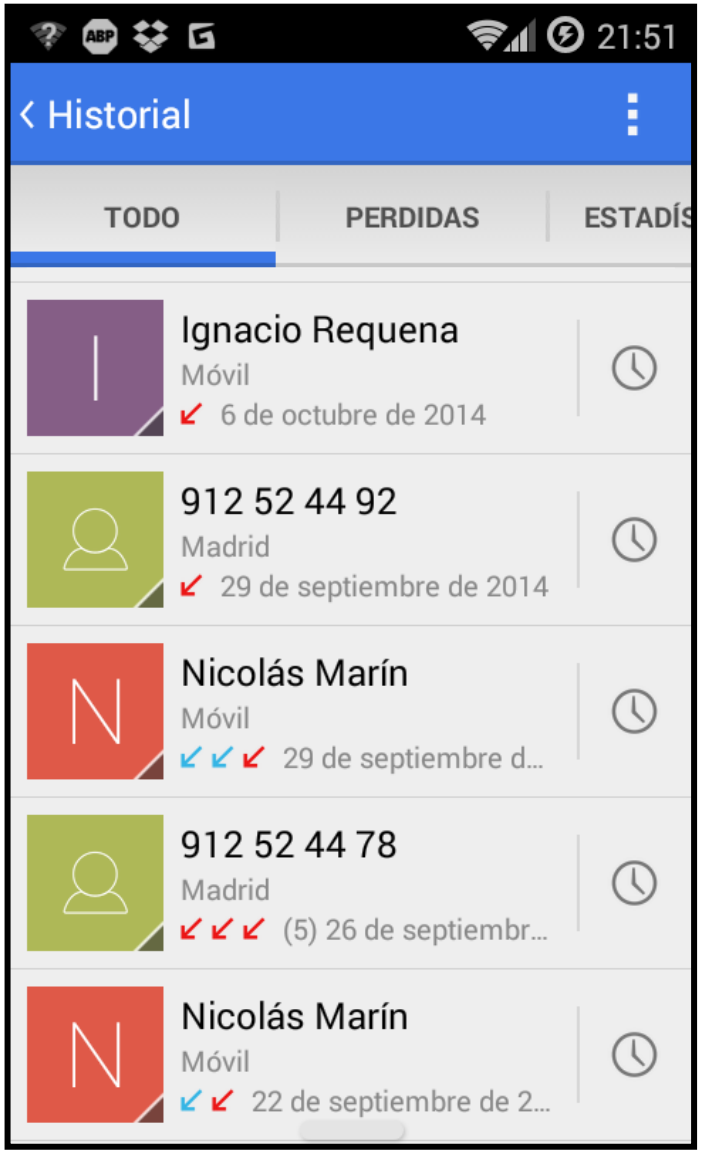
The application framework

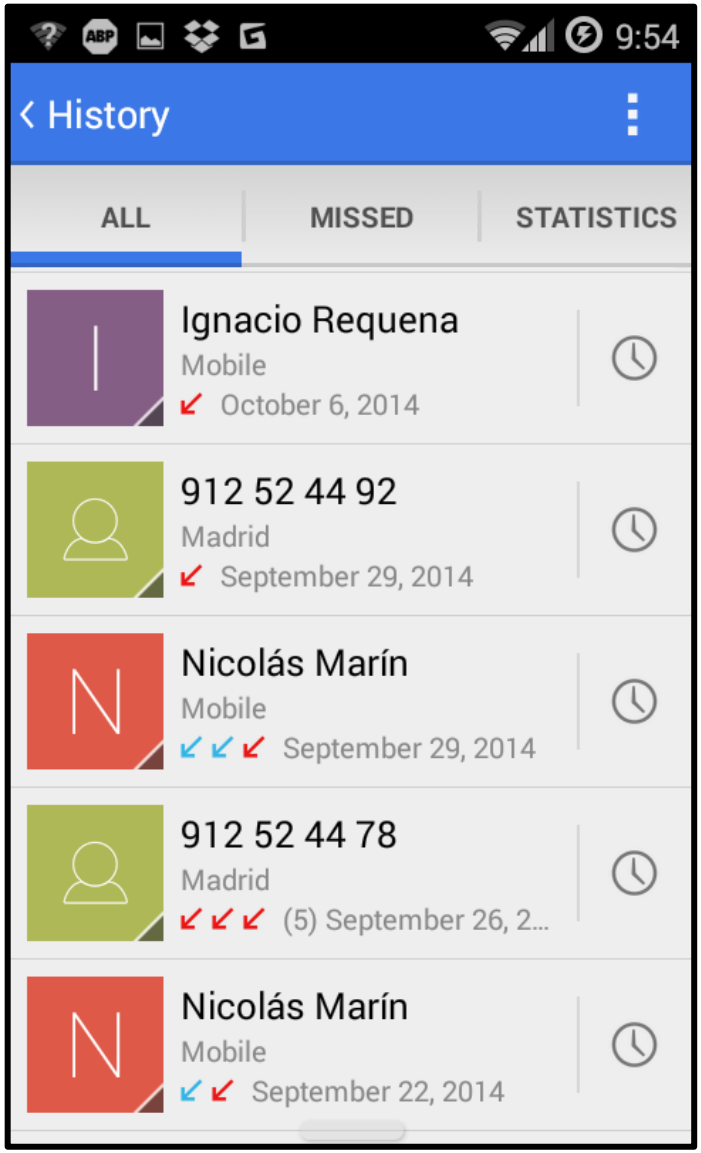


The application framework

- Resource Manager (Resource Manager)
  - Not compiled manages application resources

The application framework

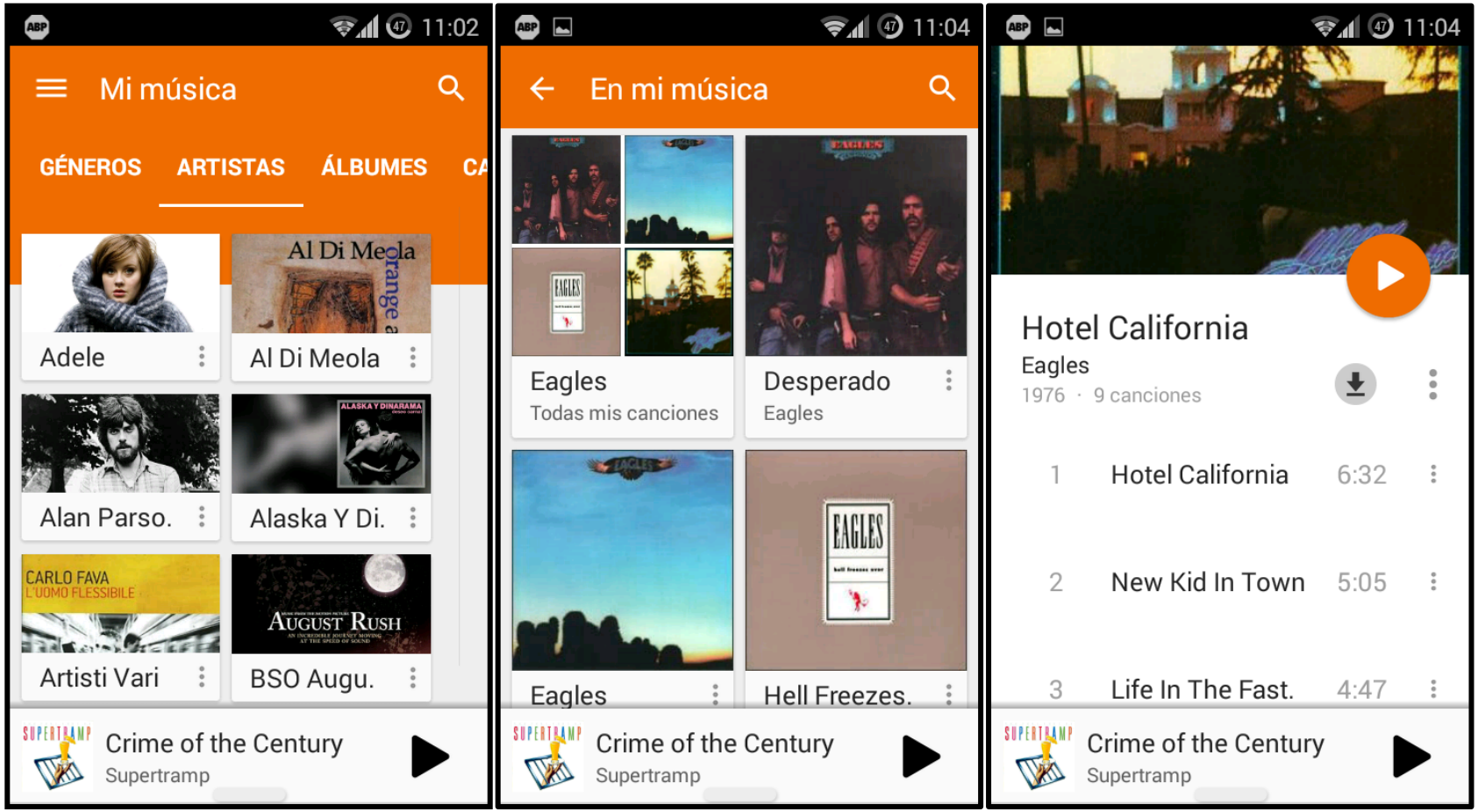




The application framework

- Activity Manager (System Activity)
  - Manages the lifecycle of applications and stack navigation

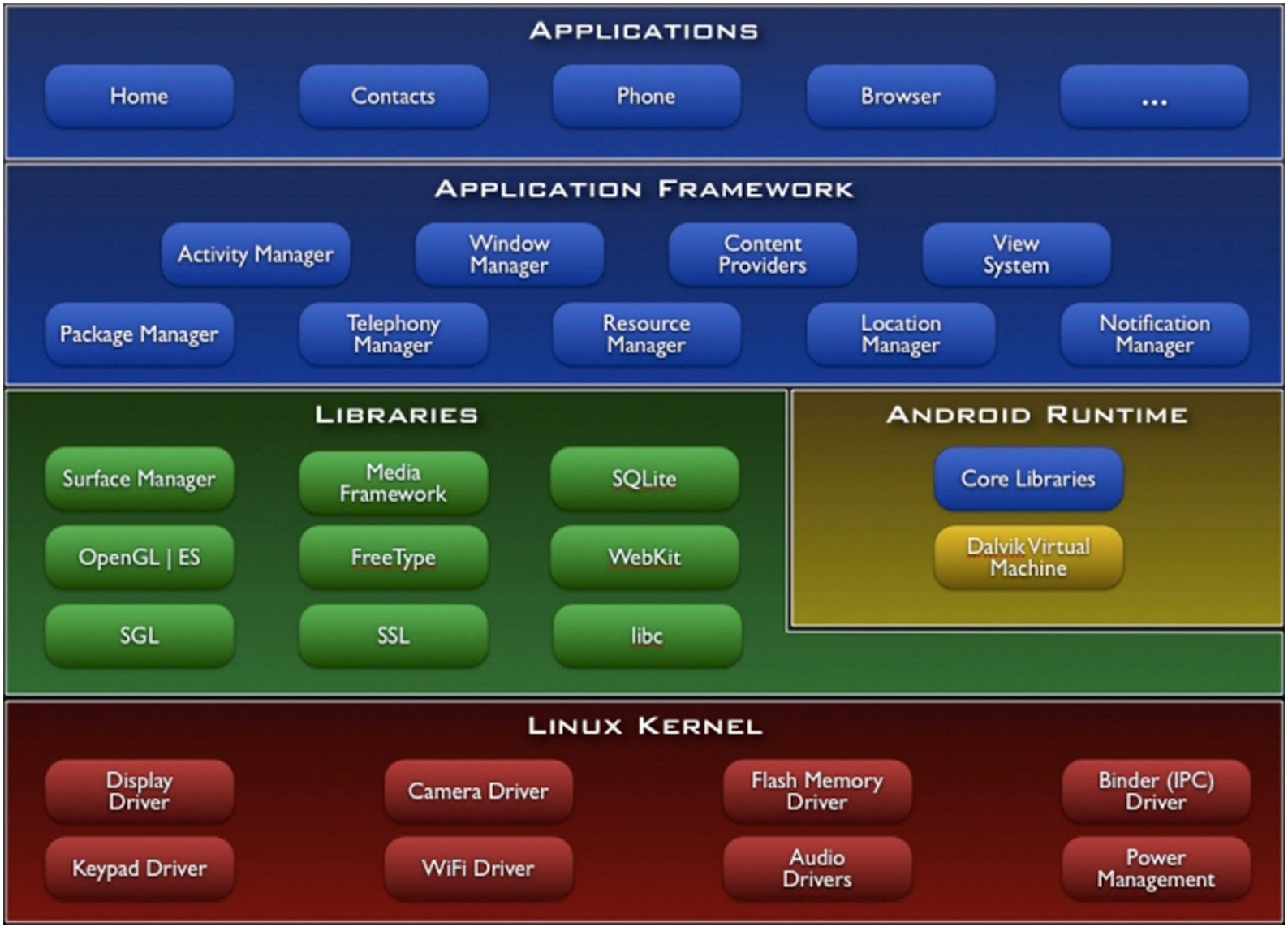
The application framework



The application framework

- Content Providers (Content Providers)
  - Sharing data between applications
- Location manager (Location Manager)
  - It provides position information and movement
- Notification Manager (Notification Manager)
  - Notification icons placed in the status bar when certain events occur

Application layer



Application layer

- Standard applications:
  - Home - Home Screen
  - Contacts - Contacts Database
  - Phone - Make phone calls
  - Browser - Displays web pages
  - Reader Mail - Read and compose e-mails
  - Etc.
- We can replace all these applications

Components of an Android application

- Activity
  - It has Graphical User Interface
- Services
  - Execution "long" (background)
- Broadcast receivers
  - Listen and respond to events
- content providers
  - Allow applications to store and share data

Applications are formed by different components Android starts (instance) and executes these components as necessary.

Each component has its own purpose and its own APIs

Activities (Activity Class)

- main class of user interaction
- Usually it implements a single task that the user can perform.

```
/ **
  *   The dialer activity That has one tab With the virtual 12key
  *   dialer, recent calls With a tab in it, a tab With the contacts and
  *   With the favorite to tab. This is the container and the tabs are
  *   embedded using intents.
  *   The Dialer tab's title is 'phone', a more common name (see strings.xml).
  * /
public class extends DialtactsActivity TransactionSafeActivity
                View.OnClickListener implements {
        private static final String TAG = "DialtactsActivity"; public static final boolean DEBUG

        = false;
```

Service (Service Activity)

- They are running in background. No interface
- Perform operations execution "long"
- Allow interaction with remote processes

```
/ **
  *  Provides "background" audio playback capabilities, Allowing the
  *  Between user to switch activities without stopping playback.
  * /
MediaPlaybackService public class extends Service {
        / ** Whether used to specify enqueue () Should start playing
          *  The new list of files right away, all the next or eleven Currently
          *  queued files Have Been Played
          * /
        public final static int NOW = 1; public final static int
        NEXT = 2; public final static int LAST = 3;

        public final static int PLAYBACKSERVICE_STATUS = 1;

        public final static int SHUFFLE_NONE = 0; public final static int
        SHUFFLE_NORMAL = 1; public final static int SHUFFLE_AUTO = 2;
```

BroadcastReceiver

- Listens and responds to events
- Plays the role of pattern subscriber at the publisher / subscriber
- The events are represented by the Intent class and then transmitted (broadcast)
- Broadcast Receiver receives and responds to the event.

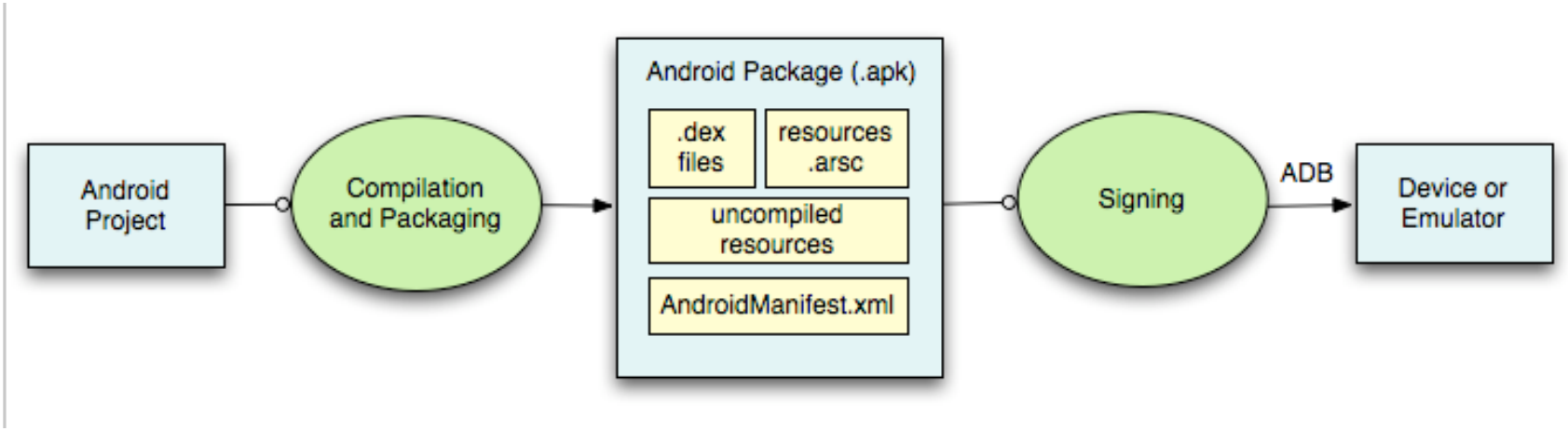
```
/ **
  *   Handle incoming SMSes. Just dispatches the work off to a Service.
  * /
public class SmsReceiver extends BroadcastReceiver {
        static final Object mStartingServiceSync = new Object (); static
        PowerManager.WakeLock mStartingService; private static SmsReceiver sInstance;


        public static SmsReceiver getInstance () {
                if (sInstance == null) {
                        SmsReceiver sInstance = new (); } sInstance

            return; }
```

content Providers

- Allow applications to store and share data
- Use a standard database interface
- Manages communication between processes

```
public class BrowserProvider extends ContentProvider {

        private SQLiteOpenHelper mOpenHelper; private BackupManager mBackupManager; static final
        String sDatabaseName = "browser.db"; private static final String TAG = "BrowserProvider"; private
        static final String order_by = "visits DESC, date DESC";



        private static final String PICASA_URL = "http://picasaweb.google.com/ m /" +

                        "Source = androidclient viewer?";

    static final String [] = new String TABLE_NAMES [] {
        "Bookmarks", "searches"};
```

Building applications



one. • Setting Resources

two. • Implementation of application classes

3. • Application Packaging

Four. • Installing and running the application

Building applications

one. • Setting Resources

- not compilable entities (not code) layout files,
  images, strings, menus ...
- Separate code
- The application can be adapted to different devices and users
- http://developer.android.com/guide/topics/resources
- strings:
  - string, string array and plurals
  - Are stored in res / values / *. Xml
  - Specified in XML

<String name = "hello"> Hello World! </ String>

  - You can include formatting
  - Access as @ string / string_name
  - Java access as R.string.string_name

Resources - Strings

- res / values / strings.xml

<? Xml version = "1.0" encoding = "utf-8"?> <Resources>

<String name = "show_map_string"> Show Map </ string> <string name =
"location_string"> Enter Location </ string> </ resources>

- res / values-en / strings.xml

<? Xml version = "1.0" encoding = "utf-8"?> <Resources>

<String name = "show_map_string"> Show Map </ string> <string name = "location_string">
Enter the </ string> </ resources>

Resources - File Layout

- Design (Layout) user interfaces are specified in XML files

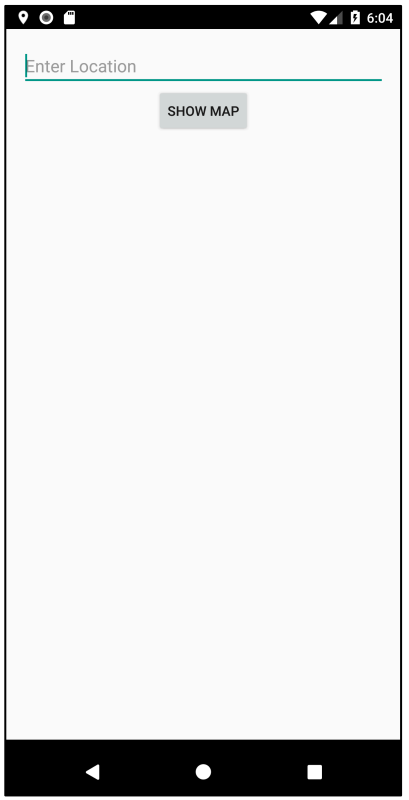
- We have tools to create designs visually. The tool will generate the XML file.
- XML files are normally stored in res / layout / * .xml
- Java access as R.layout.layout_name
- Access other resources such as files @ layout / layout_name

- We can use multiple layout files. Android runtime choose the layout depending on the device characteristics.

Resources - layout file

res / layout / activity_main.xml

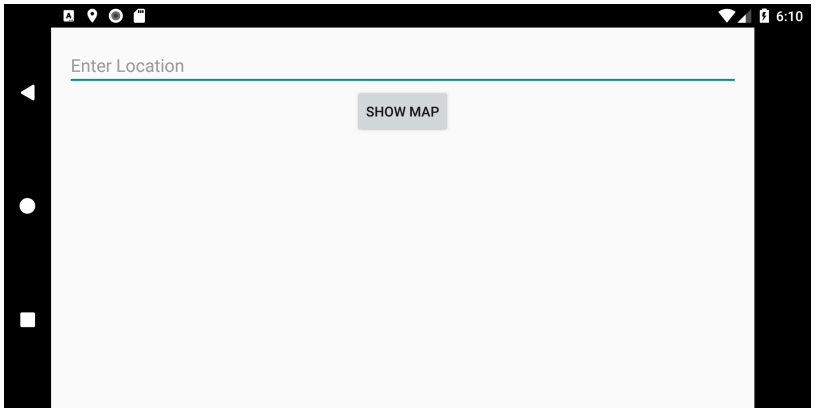
```
<RelativeLayout xmlns: android = "http://schemas.android.com/apk/res/android" xmlns: tools = "http://schemas.android.com/tools"
    android: layout_width = "match_parent" android: layout_height = "match_parent" android: paddingLeft = "@ Dimen /
    activity_horizontal_margin" android: paddingRight = "@ Dimen / activity_horizontal_margin" android: paddingTop = "@ Dimen /
    activity_vertical_margin"

    android: paddingBottom = "@ Dimen / activity_vertical_margin" tools: context = ". MainActivity">

    <EditText
        android: layout_width = "match_parent" android:
        layout_height = "wrap_content" android: inputType =
        "textPostalAddress" android: ems = "10" android: id =
        "@ + id / location" android: layout_alignParentTop =
        "true" android: layout_alignParentLeft = "true" android:
        layout_alignParentStart = "true" android: hint = "@
        string / location_string" />



    <Button
         android: layout_width = "wrap_content" android:
       layout_height = "wrap_content" android: text = "@ string /
       show_map_string" android: id = "@ + id / map_button"
       android: layout_below = "@ + id / location" android:
       layout_centerHorizontal = "true" /> </ RelativeLayout>
```



Resources - layout file

- res / layout-land / activity_main.xml

```
<? Xml version = "1.0" encoding = "utf-8"?>
<RelativeLayout xmlns: android = "http://schemas.android.com/apk/res/android"
        android: id = "@ + id / RelativeLayout1" android:
       layout_width = "match_parent" android: layout_height =
       "match_parent" android: orientation = "vertical"> <EditText

            android: id = "@ + id / location" android: layout_width =
            "match_parent" android: layout_height = "wrap_content"
            android: layout_alignParentTop = "true" android: layout_toLeftOf
            = "@ + id / mapButton" android: ems = "10"

            android: hint = "@ string / location_string" android: inputType =
         "textPostalAddress"> </ EditText> <Button

            android: id = "@ + id / mapButton" android: layout_width =
            "wrap_content" android: layout_height = "wrap_content"
            android: layout_alignParentRight = "true" android:
            layout_alignTop = "@ + id / location" android: text = "@ string /
            show_map_string "android: textsize =" 20SP "/>

</ RelativeLayout>
```



R.java

- In compiling, resources are used to generate the class R.java
- The Java code uses the R class to access resources

```
/ * AUTO-GENERATED FILE. DO NOT MODIFY.
  *
  *   This class was generated by the Automatically
  *   AAPT tool from the resource data it found. Item
  *   Should not be modified by hand.
  * /
course.examples.MapLocation package; public final class
R {
        public final static class attr {} public final static class

        drawable {
                public final static int ic_launcher = 0x7f020000; } public static final class id {


                public final static int RelativeLayout1 = 0x7f050000; public final static int location
        = 0x7f050001; public final static int mapButton = 0x7f050002; } public final static
        class layout {


                public static final int main = 0x7f030000; } public final static class

        string {
                public final static int location_string = 0x7f040001; public final static int
        show_map_string = 0x7f040000; }}
```

Building applications

two. • Implementation of application classes

} • It involves creating at least one activity

} • onCreate method: initialization activity

one. • Restore been stored

two. • Set the Content View (interface to show how the activity)

3. • Initialize GUI elements

Four. • Linking code elements of the GUI

Building applications

```
public class MapLocation extends Activity {
    private final String TAG = "MapLocation"; @Override

    protected void onCreate (Bundle savedInstanceState) {
        // Restore any saved state super.onCreate
        (savedInstanceState); // set content view

        setContentView (R.layout.main); // Initialize UI
        elements
        EditText end addrText = (EditText) findViewById (R.id.location); end = Button button (Button)
        findViewById (R.id.mapButton);
. . .
```

Building applications

```
Link // UI elements to actions in code
button.setOnClickListener (new Button.OnClickListener () {
    @Override
    public void onClick (View v) {
        try {
            String address = addrText.getText (). ToString (); address.replace address
            = ( ',' + '); Intent geoIntent = new Intent (

                    android.content.Intent.ACTION_VIEW, Uri
                            . parse ( "geo: 0.0 q =" + address)); startActivity
          (geoIntent); } Catch (Exception e) {

            Log.e (TAG e.ToString ()); }}}); }
```

. . .

Building applications

3. Application Packaging
   - The system packs the components and resources
     a file application APK
   - The developer provides the necessary information on the
     file AndroidManifest.xml
     - Application Name
     - Component List
     - Other information: permissions, hardware requirements, minimum API level

Building applications

```
<? Xml version = "1.0" encoding = "utf-8"?>
<Manifest xmlns: android = "http://schemas.android.com/apk/res/android"
        package = "course.examples.MapLocation" android:
        versionCode = "1" android: versionName = "1.0">
        <uses-sdk

                android: minSdkVersion = "10" android:
          targetSdkVersion = "17"> </ uses-sdk> <application


                android: allowBackup = "false" android: icon = "@
                drawable / ic_launcher" android: label = "MapLocation">
                <activity

                        android: name = "course.examples.MapLocation.MapLocation"> <intent-filter>

                                <Action android: name = "android.intent.action.MAIN" /> <category android: name =
"android.intent.category.LAUNCHER" /> </ intent-filter> </ activity> </ application> </ MANIFEST>
```

Building applications

Four. • Installing and running the application

- Since Android Studio, run in the emulator or device
- From the command line:
  - Enable USB debugging
  - adb install <path of the application>