



Tesina esame di stato 2016/2017

MEDIA IN CLOUD



A cura di Marco Pedrazzi

– Classe 5si –

Istituto Tecnico Tecnologico Marconi Rovereto

Indirizzo informatico

Sommario

Tesina esame di stato 2016/2017	1
Sigle e abbreviazioni	3
Introduzione	4
Tecnologie utilizzate	4
Panoramica ambiente Android.....	4
Funzionalità	6
Scelta accesso	7
Registrazione	7
Accesso	8
Galleria.....	8
Presentazione	9
Sincronizzazione	9
Parte Web	10
Sfide affrontate.....	11
Content Provider	11
Adapter	14
AsyncTask	15
Http.....	18
Conclusioni	22
Sitografia e fonti	22

Sigle e abbreviazioni

View= è una vista che mostra un componente grafico e gestisce eventi relativi all'interazione con l'utente.

CAB= Contextual Action Bar può essere considerata una Action Bar temporanea che riporta i comandi attivabili sotto forma di *action*.

GridView = è una View di cui dispone Android, consente di disporre in modo dinamico gli elementi in formato a griglia.

ImageView = è una View, che consente di visualizzare immagini ed interagire con esse.

Introduzione

Mentre stavo pensando a un'idea per il progetto di fine anno, ho notato che uno dei dispositivi che ha certamente rivoluzionato maggiormente le nostre abitudini è lo smartphone.

Esso ha ormai tutte le caratteristiche di un pc, ed è completo di tante funzionalità aggiuntive, come sensori di prossimità, GPS, giroscopio ed altro, ma una delle più utilizzate resta certamente la fotocamera.

Ormai si scattano centinaia di fotografie con estrema facilità, occupando velocemente la memoria del dispositivo. Per risolvere questo problema ci sono molte app, la più diffusa e conosciuta è Google photo. Nel mio progetto ho pensato di svilupparne una semplice versione.

Tecnologie utilizzate

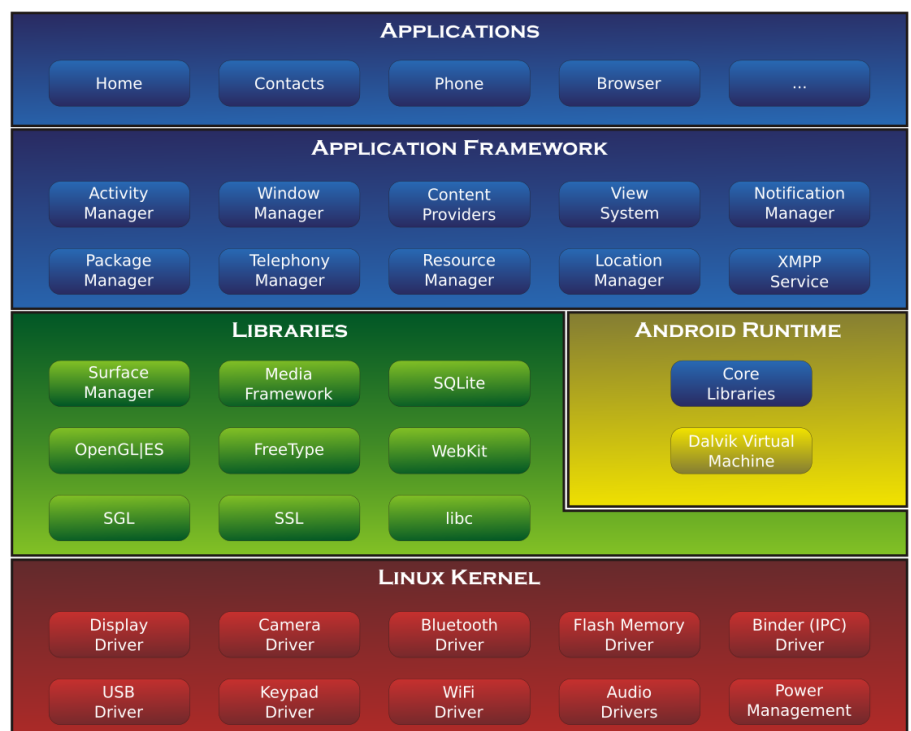
L'applicativo utilizza le seguenti tecnologie:

App	Parte web
Linguaggio: Java	Linguaggi: C#, Html,Css
Framework Android (classi che estendono java)	ADO.NET Entity Framework (mappa il database in classi)
SqlLite (database)	SQL (database)
Libreria Soap (interfaccia/connessione al Web Service)	Web Service Soap (Consente di invocare servizi da parte di applicazioni esterne)
	-Bootstrap (Grafica e Scripting)

Panoramica ambiente Android

Android è un sistema operativo per dispositivi mobili, è stato sviluppato da Google, partendo da una base open source, il kernel Linux.

Le app in assenza di ulteriori framework (Apache Cordova, AppInventor, etc) vengono sviluppate in java che attinge da un framework standard integrato nel sistema che estende le librerie classiche di java. Ogni app viene eseguita tramite un compilatore ibrido, la Dalvik virtual Machine o la più recente Android Run Time.



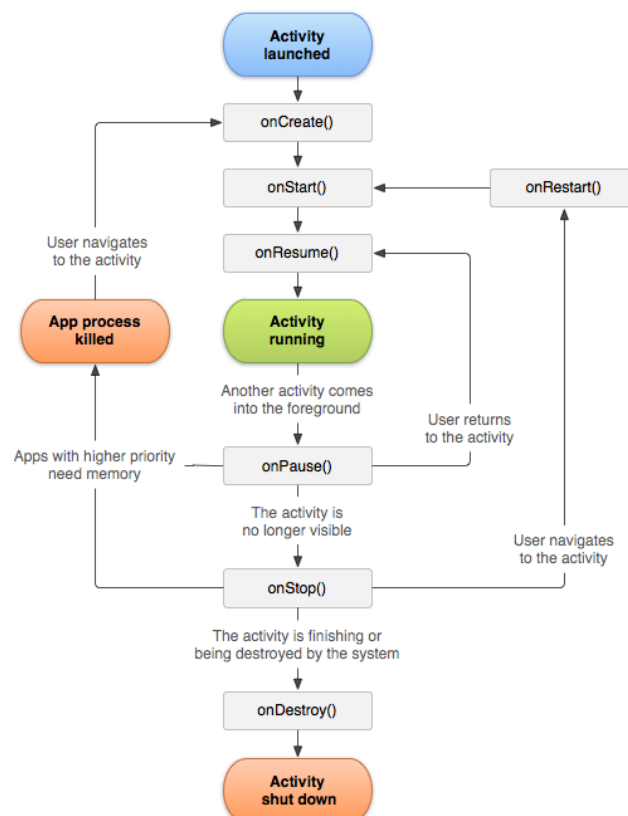
Da notare che in android tutto è un'app: la schermata home, il menu e persino l'app che effettua le chiamate telefoniche.

Con la sua flessibilità e leggerezza riesce ad adattarsi a molte categorie di dispositivi, smartphone, orologi da polso, Smart-tv, occhiali, eccetera. Ad oggi è il sistema operativo più diffuso su dispositivi mobili, con una quota di mercato pari al 62%.

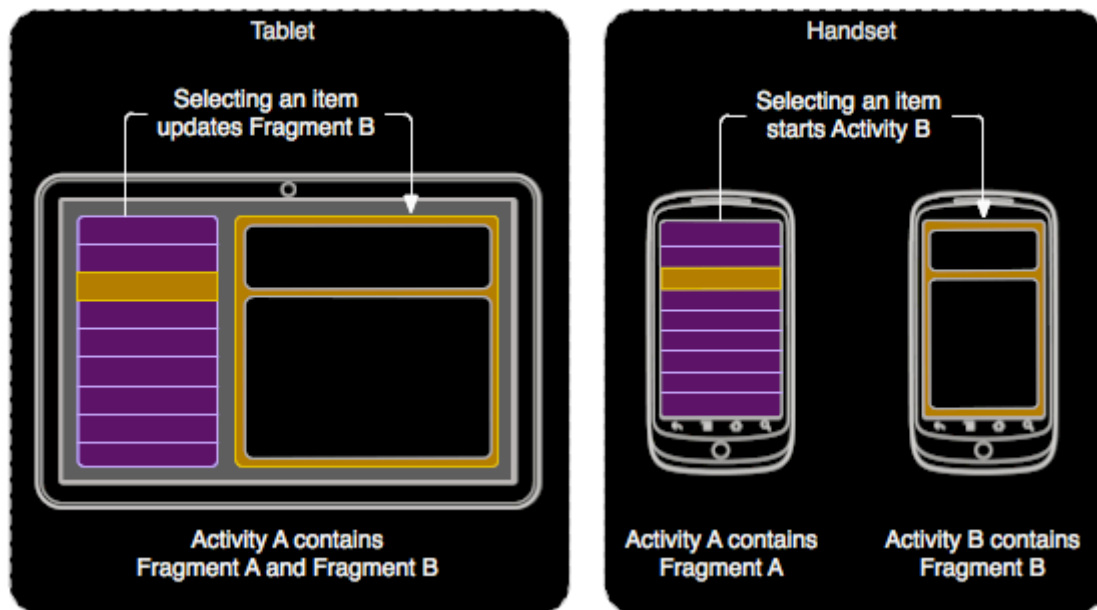
Un'arma a doppio taglio secondo me è il rapido rilascio di nuove versioni, che se da un lato introduce sempre notevoli migliorie e stabilità al sistema, dall'altra richiede un notevole impegno sia ai produttori di smartphone che agli sviluppatori di app per mantenere il software aggiornato e compatibile con le nuove release.



Entrando più in dettaglio, un'app è composta da diverse schermate chiamate "activity", che si susseguono in modalità a stack; ogni activity ha il suo ciclo di vita che a livello di codice viene gestito tramite alcune callback. A seguire qui un'immagine esplicativa.



Un'activity a sua volta, ma solo facoltativamente, può essere divisa in fragment: questo permette di migliorare l'esperienza utente, permettendo a dispositivi come i tablet di disporre il contenuto in maniera ottimale.



Funzionalità

L'applicazione offre all'utente la possibilità di sincronizzare i propri media in cloud, permettendone la visualizzazione sia su tutti i dispositivi associati all'account dell'utente, sia sull'applicazione web. In entrambi i casi, per l'accesso vengono utilizzate le medesime credenziali.

Inoltre l'applicazione fornisce la possibilità di eliminare i media dal dispositivo, mantenendo comunque la possibilità di visualizzazione grazie al download run-time implementato; questo consente di avere più spazio libero sul dispositivo.

Scelta accesso

La prima activity dell'app offre la possibilità di scegliere se creare un nuovo account oppure accedere con un account già esistente.

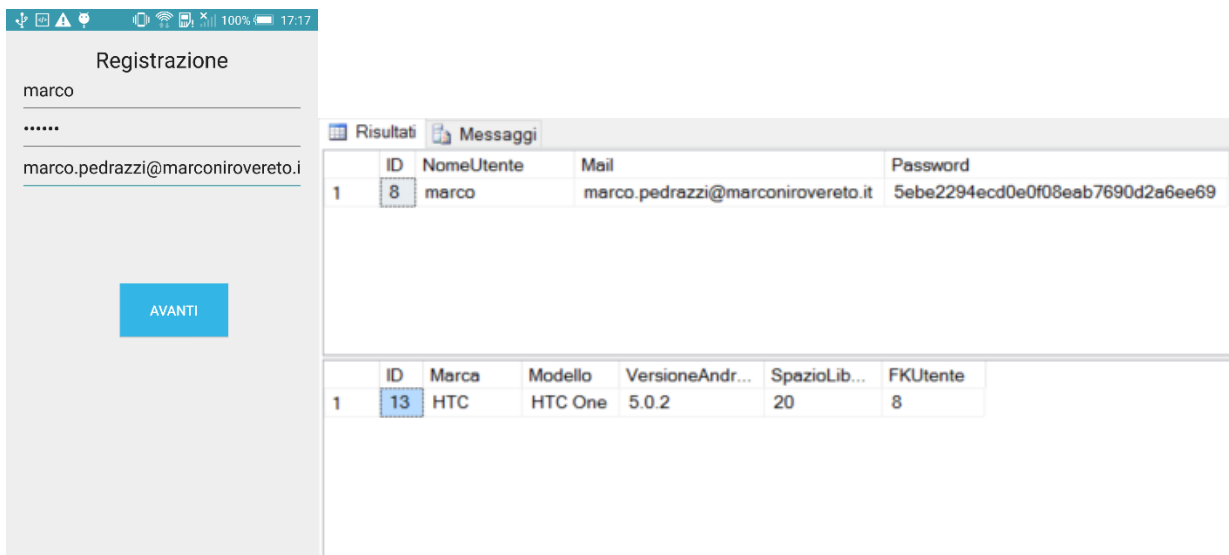


1 Activity iniziale.

Registrazione

Permette all'utente la creazione di un account, scegliendo nome utente e password.

La registrazione necessita di una connessione attiva per contattare il server, che attraverso il web service inserisce il record nel database e, oltre alle informazioni richieste, registra anche informazioni sul dispositivo.



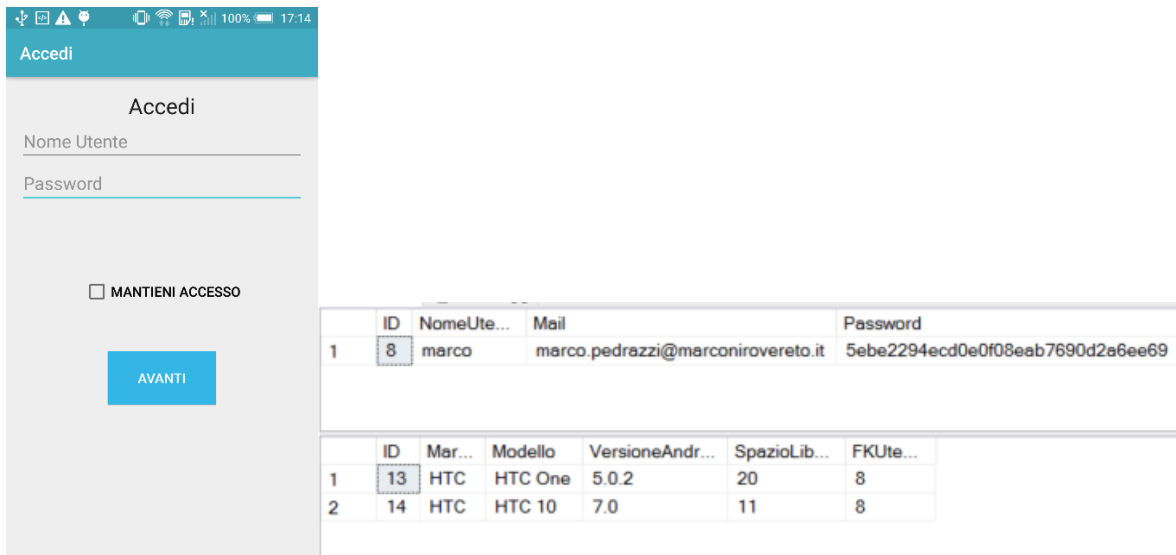
2 Activity registrazione

2.1 screenshot tabella utenti e dispositivi.

Accesso

Se l'accesso viene effettuato con un account già esistente, il nuovo dispositivo verrà associato a quell'account; questa modalità necessita di una connessione attiva.

Altrimenti l'accesso può essere eseguito in modalità offline, in quanto le credenziali sono memorizzate sul database del dispositivo; questo è stato previsto per consentire la visualizzazione dei media locali al dispositivo.



The screenshot shows the login interface with fields for 'Nome Utente' and 'Password', a checkbox for 'MANTIENI ACCESSO', and an 'AVANTI' button. Below the login form are two tables. The first table lists user accounts, and the second table lists devices.

	ID	NomeUte...	Mail	Password
1	8	marco	marco.pedrazzi@marconirovereto.it	5ebe2294ecd0e0f08eab7690d2a6ee69

	ID	Mar...	Modello	VersioneAndr...	SpazioLib...	FKUte...
1	13	HTC	HTC One	5.0.2	20	8
2	14	HTC	HTC 10	7.0	11	8

3 Activity accedi

3.1 Screenshot tabelle utenti e dispositivi.

Galleria

Dopo l'accesso vengono visualizzati:

I media che sono presenti sul dispositivo.



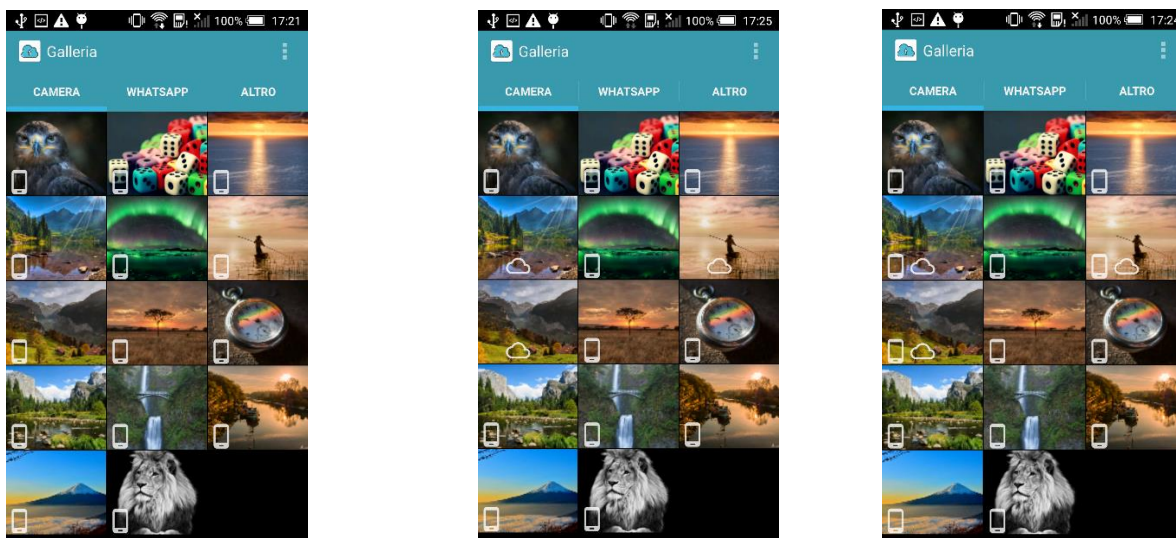
Media sincronizzati da altri dispositivi associati all'account.



Media sincronizzati dello stesso dispositivo.

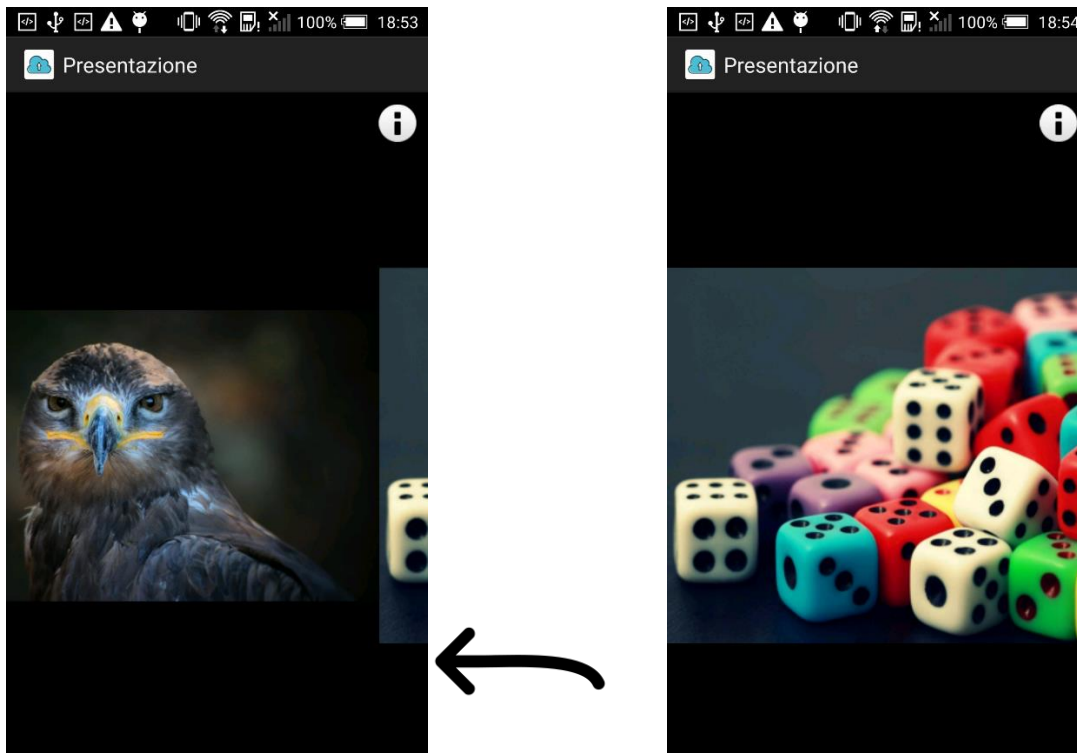


I media sono visualizzati, nel componente Gridview, con ordine decrescente in base alla data di acquisizione.



Presentazione

Esiste la possibilità di visualizzare i media a pieno schermo, navigando tra gli stessi con un “swipe”.

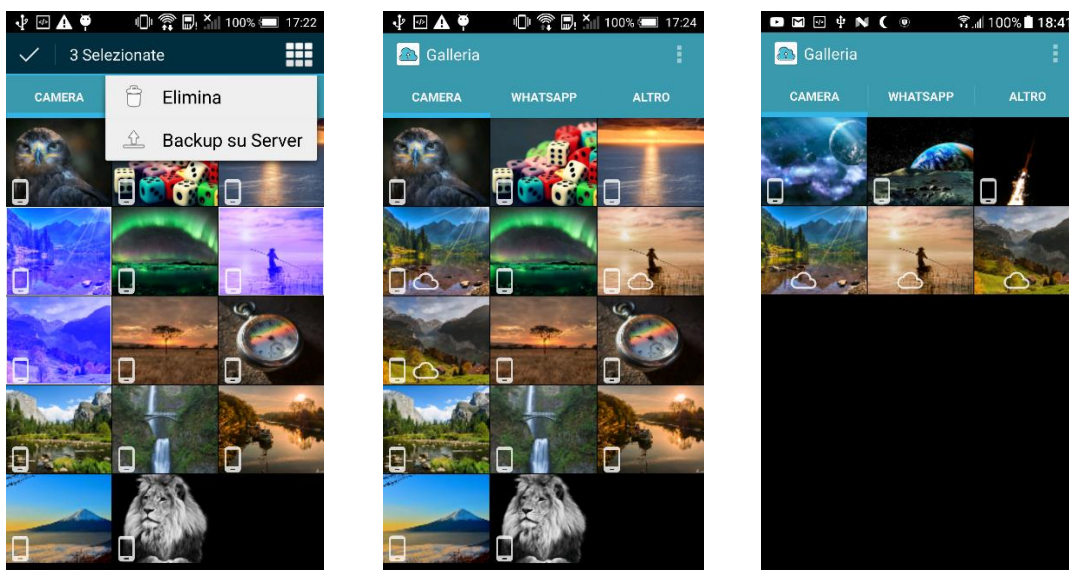


Sincronizzazione

Attivando la CAB con un touch lungo sul media, si può eseguire una selezione multipla.

Per effettuare l'upload, sarà sufficiente aprire il menu contestuale e cliccare “backup su server”.

Per quanto riguarda il download dei media già sincronizzati, viene effettuato automaticamente.



4 Action mode

Upload terminato

Altro dispositivo associato all'account

Parte Web

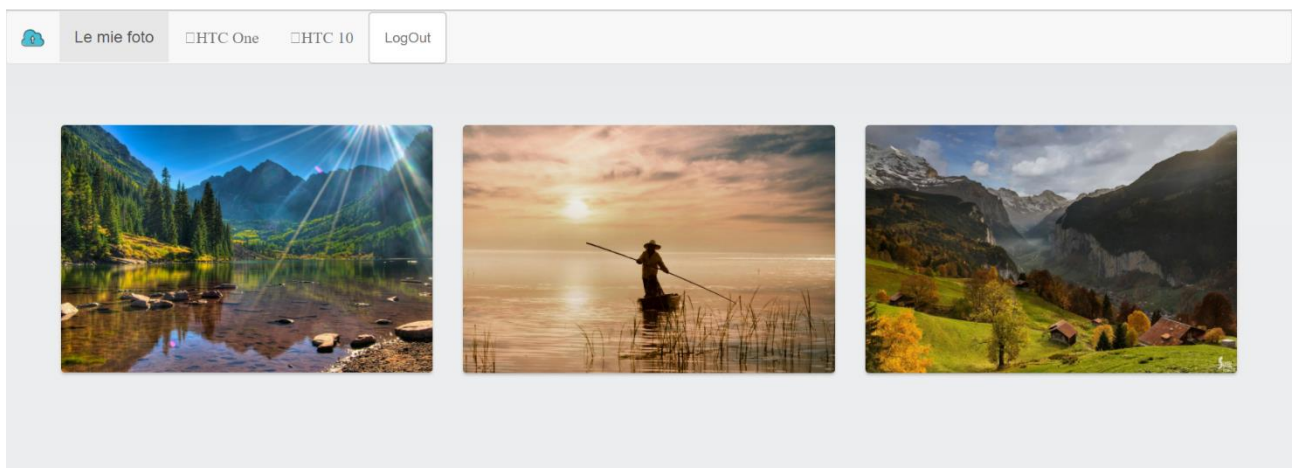
È possibile accedere tramite la pagina di Login con le credenziali utilizzate sui dispositivi.

Accedi



Accedi

Nella pagina seguente si vedranno tutte le foto sincronizzate, oppure suddivise per dispositivo.



E' possibile visualizzare a schermo intero, tramite lo slideshow implementato.



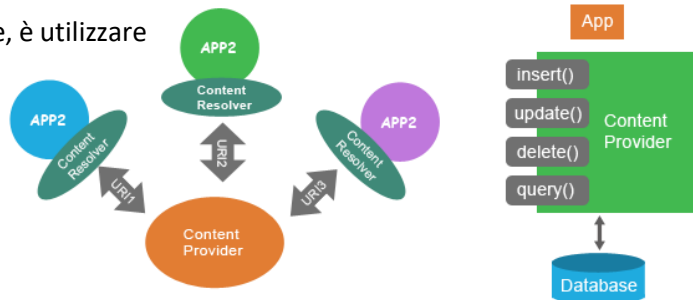
Sfide affrontate

Content Provider

Uno dei primi scogli che ho incontrato, è come recuperare i media dal dispositivo. Inizialmente ho cercato di fare una “scan” del filesystem, ma ho visto che era molto “macchinoso” ed inoltre c’è una mancanza di standard nei nomi assegnati alle cartelle.

L’app fotocamera infatti memorizza le foto sempre nella cartella “DCIM” che di fatto è uno standard, ma i vari brand utilizzano nomi diversi per le sottocartelle, oppure non fanno uso di sub-directory.

La soluzione che ho trovato dopo varie ricerche, è utilizzare un content provider; questo componente di android altro non è che un database in sqllite condiviso tra tutte le app d’ambiente.



Esso è suddiviso in settori: sotto una tabella esplicativa.

The standard content providers of Android

Provider	Since	Usage
Browser	SDK 1	Manages your web-searches, bookmarks and browsing-history.
CalendarContract	SDK 14	Manages the calendars on the user’s device.
CallLog	SDK 1	Keeps track of your call history.
Contacts	SDK 1	The old and deprecated content provider for managing contacts. You should only use this provider if you need to support an SDK prior to SDK 5!
ContactsContract	SDK 5	Deals with all aspects of contact management. Supersedes the Contacts-content provider.
MediaStore	SDK 1	The content provider responsible for all your media files like music, video and pictures.
Settings	SDK 1	Manages all global settings of your device.
UserDictionary	SDK 3	Keeps track of words you add to the default dictionary.

<http://blog.csdn.net/chdj>

Io ho utilizzato il MediaStore, che indicizza video, immagini, musica ed è interrogabile nel seguente modo:

```
private Cursor searchImage(String where){
    //istanzio il content resolver neccessario per interrogare il content provider (mediaStore)
    ContentResolver contentResolver = this.ctx.getContentResolver();

    //definisco le colonne che voglio estrarre
    final String[] selezione = {
        MediaStore.Images.Media.DATA,
        MediaStore.Images.Media.DATE_TAKEN,
        MediaStore.Images.Media.DISPLAY_NAME,
        MediaStore.Images.Media.BUCKET_DISPLAY_NAME,
        MediaStore.Images.Media.MIME_TYPE,
        MediaStore.Images.Media.SIZE,
        MediaStore.Images.Media.HEIGHT,
        MediaStore.Images.Media.WIDTH,
        MediaStore.Images.Media.ORIENTATION,
        MediaStore.Images.Media.LATITUDE,
        MediaStore.Images.Media.LONGITUDE};

    //definisco l'ordine del interrogazione
    final String orderBy = MediaStore.Images.Media.BUCKET_DISPLAY_NAME+" and
"+MediaStore.Images.Media.DATE_TAKEN+ " DESC";

    //eseguo query sul content provider
    Cursor cursor = contentResolver.query(
        MediaStore.Images.Media.EXTERNAL_CONTENT_URI, //URI risorsa da interrogare
        selezione, //colonne da estrarre
        where, // (WHERE in SQL)
        null,
        orderBy); //selezione (ORDER BY in SQL)

    //esempio query generata
    //SELECT _data, datetaken, bucket_display_name, mime_type
    // FROM images WHERE (bucket_display_name='100MEDIA')
    // ORDER BY bucket_display_name and datetaken DESC
    return cursor;
}
```

Quando ottengo l'oggetto cursor, ciclo le informazioni che ritorna, creo l'oggetto FileMedia e lo aggiungo alla lista.

```
public ArrayList<FileMedia> getListMedia(Album album)
{
    this.listMedia=null;
    this.listMedia=new ArrayList<FileMedia>();
    FileMedia fileMedia=null;
    String where=null;

    //controllo che album si vuole estrarre
    switch (album)
    {
        case Camera:
            where=this.camera;
            break;
        case WhatsApp:
            where=this.whatsApp;
            break;
        case All:
            where=this.all;
            break;
    }

    //recupero il cursore per le immagini
    Cursor cursorImage=searchImage(where);

    //recupero indice colonne
    int pathColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.DATA);
    int dateColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.DATE_TAKEN);
}
```



```

int nameColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.DISPLAY_NAME);
int dirColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.BUCKET_DISPLAY_NAME);
int mimeTypeColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.MIME_TYPE);
int sizeColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.SIZE);
int heightColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.HEIGHT);
int widthColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.WIDTH);
int orientationColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.ORIENTATION);
int latitudeColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.LATITUDE);
int longitudeColumnIndex = cursorImage.getColumnIndex(MediaStore.Images.Media.LONGITUDE);

//recupero il numero di righe ritornate
int countRow = cursorImage.getCount();

//ciclo per ogni riga
for (int i = 0; i < countRow; i++)
{
    fileMedia=null;
    //sposto il cursore sulla ennesima riga
    cursorImage.moveToPosition(i);
    String path = cursorImage.getString(pathColumnIndex);
    File file=new File(path);

    //se il file esiste
    if(file.exists())
    {
        String bucket = cursorImage.getString(dirColumnIndex);
        String nome = cursorImage.getString(nameColumnIndex);
        String mimeType= cursorImage.getString(mimeTypeColumnIndex);
        Integer dimensione = cursorImage.getInt(sizeColumnIndex);
        Integer altezza = cursorImage.getInt(heightColumnIndex);
        Integer larghezza = cursorImage.getInt(widthColumnIndex);
        String orientamento= cursorImage.getString(orientationColumnIndex);
        Double latitudine= cursorImage.getDouble(latitudeColumnIndex);
        Double longitudine= cursorImage.getDouble(longitudeColumnIndex);

        Long timestamp = cursorImage.getLong(dateColumnIndex);
        Date dataAquisizione=new Date();
        dataAquisizione.setTime(timestamp);

        Boolean suServer=false;
        Boolean suDispositivo=true;

        //se il media è già presente nel db locale
        if(dbGestione.CheckMedia(nome))
        {
            //significa che è già sul server
            suServer=true;
        }

        //creo ed aggiungo il media alla lista
        fileMedia = new FileMedia(dataAquisizione,path,nome,bucket,
        mimeType,dimensione,altezza,larghezza,orientamento,
        latitudine,longitudine,suServer,suDispositivo);

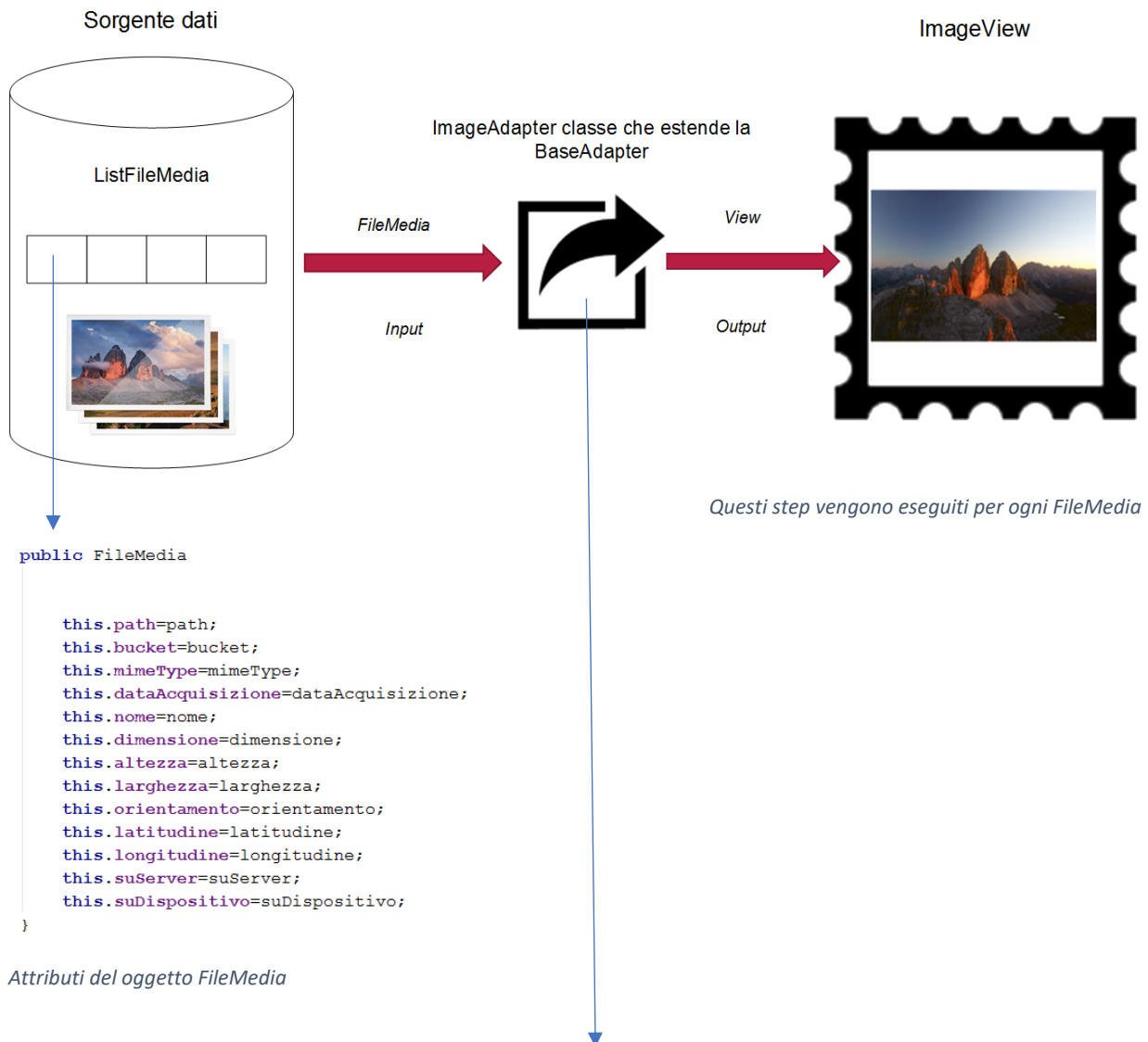
        this.listMedia.add(fileMedia);
    }
}
//chiudo il cursore
cursorImage.close();

Collections.sort(listMedia);
return this.listMedia;
}

```

Adapter

Popolata la lista di media, devo mostrarla nel componente gridView: per fare questo sono necessari alcuni step.



Qui il metodo della classe ImageAdapter che genera le views; nel mio caso carico nel imageView il bitmap in maniera asincrona grazie alla classe AsyncTask che descriverò nel paragrafo successivo.

```
// crea un imageView per ogni view della gridView
public View getView(int position, View convertView, ViewGroup parent)
{
    FileMedia media=listMedia.get(position);
    ImageViewOverlay imageViewOverlay;
    //contollo se esiste già la view e posso riutilizzarla
    if (convertView != null)
    {
        imageViewOverlay = (ImageViewOverlay) convertView;
        //reimposto gli attributi corretti
        imageViewOverlay.suServer=media.getSuServer();
        imageViewOverlay.suDipositivo=media.getSuDispositivo();
    }
    else
    {
        imageViewOverlay = new ImageViewOverlay(ctx,media.getSuServer(),media.getSuDispositivo());
    }
}
```

```

        imageViewOverlay.setLayoutParams(new GridView.LayoutParams(250, 250));
    }

    Bitmap bitmap=null;

    bitmap = cachePhoto.get(media.getDataAcquisizione().getTime());

    if (bitmap != null)
    {
        imageViewOverlay.setImageBitmap(cachePhoto.get(media.getDataAcquisizione().getTime()));
        Log.d("Load image: ", "Cache");
    }
    else //se non presente in cache la carico in modo asincrono
    {
        try
        {
            CaricaImmagine(this.ctx,media,imageViewOverlay,position); ←Carico in maniera asincrona
            Log.d("Load image: ", "Async");
        }
        catch (Exception e)
        {
            Log.d(this.getClass().getSimpleName(),e.getMessage());
        }
    }

    //l'immagine è selezionata?
    if(media.getSelezionata())
    {
        return SelezionaImmagine(imageViewOverlay,true);
    }
    else
    {
        return SelezionaImmagine(imageViewOverlay,false);
    }
}

```

AsyncTask

Come visto nel capitolo precedente, ogni volta che l'adapter genera un ImageView carico all'interno dello stesso un oggetto Bitmap.

Fare questo direttamente nella classe ImageAdapter non è possibile, perché impegna troppo il thread principale, che deve occuparsi anche di tenere la grafica fluida ed aggiornata. Per questo carico i media in maniera asincrona estendendo la classe AsyncTask.

I metodi principali di AsyncTask:

1. onPreExecute: viene eseguito sul thread principale, contiene il codice di inizializzazione dell'interfaccia grafica.
2. doInBackground: eseguito in un thread in background si occupa di eseguire il task vero e proprio.
3. onPostExecute: eseguito nel thread principale e si occupa di aggiornare l'interfaccia dopo l'esecuzione per mostrare i dati scaricati.

La classe *AsyncTask* definisce tre parametri in ingresso:

1° parametro: input di doInBackground()

2° parametro: On progress update (opzionale)

3° parametro: output di doInBackground()

```

//OnPreExecute e OnPostExecute vengono eseguiti sul thread principale

//1° parametro input doInBackground -- 3° parametro output doInBackground
public class LoadPhotoBackground extends AsyncTask<FileMedia,Void,Bitmap> {
    private int larghezzaMaxMedia = 100;
    private int altezzaMaxMedia = 100;
    WeakReference<ImageViewOverlay> imageViewReferences;
    private MemoryCachePhoto cachePhoto;
    private int posizione;
    private Context ctx;
    FileMedia media;

    public LoadPhotoBackground(Context ctx, ImageViewOverlay imageViewOverlay, MemoryCachePhoto
    cachePhoto, int posizione)
    {
        //riferimento debole dell'imageView per l'adapter, serve per consentire al garbage collector
        l'eliminazione
        imageViewReferences = new WeakReference<ImageViewOverlay>(imageViewOverlay);
        this.cachePhoto=cachePhoto;
        this.posizione=posizione;
        this.ctx=ctx;
    }
    @Override
    protected void onPreExecute()
    {
        super.onPreExecute();
        Log.w("OnPreExecute", "");
    }

    //metodo eseguito in un thread separato
    @Override
    protected Bitmap doInBackground(FileMedia... params) {
        this.media=params[0];
        String mimeType = this.media.getMimeType();
        String percorsoFile = this.media.getPath();
        Bitmap anteprima = Bitmap.createBitmap(200, 200, Bitmap.Config.RGB_565);
        BitmapFactory.Options opzioniBitmap = new BitmapFactory.Options();
        //codifica più leggera ogni pixel 2 byte
        opzioniBitmap.inPreferredConfig = Bitmap.Config.RGB_565;
        String log="";

        try {
            switch (mimeType)
            {
                case "video/mp4":
                    anteprima = ThumbnailUtils.createVideoThumbnail(percorsoFile,
                    MediaStore.Video.Thumbnails.MICRO_KIND); //96x96 dp
                    log+="Video";
                    Log.d("Video: ", "H: " + anteprima.getHeight() + " W: " + anteprima.getWidth()+" "+log);
                    break;
                case "image/jpeg":
                case "image/jpg":
                case "image/png":
                    //cerco se esiste già un anteprima nei metadati exif
                    ExifInterface exifInterface = new ExifInterface(percorsoFile);
                    Log.i(this.getClass().getSimpleName(),exifInterface.toString());
                    if (exifInterface.hasThumbnail())
                    {
                        byte[] thumbnail = exifInterface.getThumbnail();
                        opzioniBitmap.inJustDecodeBounds = true;
                        anteprima = BitmapFactory.decodeByteArray(thumbnail, 0, thumbnail.length, opzioniBitmap);

                        //Calcolo il corretto fattore di ridimensionamento
                        opzioniBitmap.inSampleSize = calculateInSampleSize(opzioniBitmap);
                        //decodifico
                        opzioniBitmap.inJustDecodeBounds = false;
                        anteprima = BitmapFactory.decodeByteArray(thumbnail, 0, thumbnail.length, opzioniBitmap);
                        log+="DatiExif";
                    }
                else { //Altrimenti la creà scalando l'immagine

                    opzioniBitmap.inJustDecodeBounds = true;
                    //estraggo solamente le dimensioni
                    anteprima = BitmapFactory.decodeFile(percorsoFile, opzioniBitmap);
                    //Calcolo il corretto fattore di ridimensionamento
                    opzioniBitmap.inSampleSize = calculateInSampleSize(opzioniBitmap);

```



```

        opzioniBitmap.inJustDecodeBounds = false;
        //ora decodifico
        anteprima = BitmapFactory.decodeFile(percorsoFile, opzioniBitmap);
        log+="NoDatiExif";
    }
    break;
case "image/web":

    Http http =new Http();
    // richiesta get
    byte[] arrayByte=http.Ricevi(this.media.getPath());
    Log.i(this.getClass().getSimpleName(),""+arrayByte.length);

    if(arrayByte!=null)
    {
        opzioniBitmap.inJustDecodeBounds = true;
        //estraggo solamente le dimensioni
        anteprima = BitmapFactory.decodeByteArray(arrayByte, 0, arrayByte.length,opzioniBitmap);
        //Calcolo il corretto fattore di ridimensionamento
        opzioniBitmap.inSampleSize = calculateInSampleSize(opzioniBitmap);
        opzioniBitmap.inJustDecodeBounds = false;
        //ora decodifico
        anteprima = BitmapFactory.decodeByteArray(arrayByte, 0, arrayByte.length,opzioniBitmap);
    }
    break;
}

} catch (IOException e)
{
    e.printStackTrace();
}

return anteprima;
}

@Override
protected void onPostExecute(Bitmap bitmap) {

    cachePhoto.put(this.media.getDataAcquisizione().getTime(),bitmap);
    imageViewOverlay.setImageBitmap(bitmap);
}

//calcola il fattore di scala ottimale per il media
private int calculateInSampleSize(BitmapFactory.Options bmOptions)
{
    Log.d("immagine originale: ", "H: " + bmOptions.outHeight + " W: " + bmOptions.outWidth);
    final int photoWidth = bmOptions.outWidth;
    final int photoHeight = bmOptions.outHeight;
    int scaleFactor = 1;
    if (photoWidth > larghezzaMaxMedia || photoHeight > altezzaMaxMedia) {
        final int halfPhotoWidth = photoWidth / 2;
        final int halfPhotoHeight = photoHeight / 2;while (halfPhotoWidth / scaleFactor >
larghezzaMaxMedia || halfPhotoHeight / scaleFactor > altezzaMaxMedia)
        {
            scaleFactor *= 2;
        }
    }
    return scaleFactor;
}

```

Http

Come accennato nel capitolo **presentazione** sia nel upload che nel download dei media faccio uso del protocollo http.

L'http (*HyperText Transfer Protocol*) è un protocollo di livello 7_(applicazione) ed è basato sul testo; è usato come sistema principale nella trasmissione delle informazioni sul web ed è basato su un'architettura Client-Server.

È stateless, ed espone dei metodi per le richieste; i principali e più utilizzati sono:

GET

Metodo che utilizzo quando la classe LoadPhotoBackground incontra un media che è sul server e procede al download.

```
public byte[] Ricevi(String linkFoto)
{
    //Creo l'oggetto url
    URL url = null;
    //Istanzio la connessione/client
    HttpURLConnection client = null;

    try
    {
        url = new URL("http://savemyphoto.gear.host/"+linkFoto);
        //istanzio il client
        client = (HttpURLConnection) url.openConnection();
        //imposto il metodo http da utilizzare
        client.setRequestMethod("GET");
        //user agent fake
        client.setRequestProperty("User-Agent", "CodeJava Agent");
        //imposto la codifica
        client.setRequestProperty("charset", "UTF-8");
        //non utilizzo il meccanismo di caching
        client.setUseCaches(false);
        //apro la connessione (da qui non posso più impostare nessuna proprietà di connessione)
        client.connect();
        Log.i(this.getClass().getSimpleName(), "Connessione aperta...");
    }
    catch (MalformedURLException e)
    {
        e.printStackTrace();
        Log.i(this.getClass().getSimpleName(), "Url nn corretto!");
    }
    catch (IOException e)
    {
        e.printStackTrace();
        Log.i(this.getClass().getSimpleName(), "Problemi nel contattare il server!");
    }
}
```

Quando leggo la risposta dal server il protocollo http risponde con dei codici “predefiniti” qui elenco i principali:

200 OK. Il server ha fornito correttamente il contenuto nella sezione body.

400 Bad Request. La risorsa richiesta non è comprensibile al server.

404 Not Found. La risorsa richiesta non è stata trovata e non se ne conosce l'ubicazione. Di solito avviene quando l'URI è stato indicato in modo incorretto, oppure è stato rimosso il contenuto dal server.

```

//Leggo la risposta dal server
switch (LetturaRispostaServerDownload(client))
{
    case 200:
    {
        Log.i(this.getClass().getSimpleName(), "Download ok! Risposta server: 200 OK!");
        try
        {
            InputStream inputStream=client.getInputStream();
            ByteArrayOutputStream buffer = new ByteArrayOutputStream();
            int nRead;
            byte[] data = new byte[1024];
            while ((nRead = inputStream.read(data, 0, data.length)) != -1) {
                buffer.write(data, 0, nRead);
            }
            client.disconnect();
            return buffer.toByteArray();
        }
        catch (IOException e)
        {
            e.printStackTrace();
            client.disconnect();
        }
    }
}

//DISCONNESSIONE
client.disconnect();

return null;

```

POST (multipart)

Metodo che utilizzo quando faccio l'upload di uno o più media sul server cliccando "backup su server".

```

public boolean Invia(String nomeUtente, Integer idDispositivo, ArrayList<FileMedia> listMedia)
{
    //Creo l'oggetto url
    URL url = null;
    try
    {
        url = new URL(urlServer);
    }
    catch (MalformedURLException e)
    {
        e.printStackTrace();
        Log.i(this.getClass().getSimpleName(), "Url nn corretto!");
    }
    //Istanzio la connessione/client
    HttpURLConnection client = null;
    try
    {
        client = (HttpURLConnection) url.openConnection();
    }
    catch (IOException e)
    {
        e.printStackTrace();
        Log.i(this.getClass().getSimpleName(), "Problemi nel contattare il server!");
    }
    //imposto che la connessione è in uscita(abilita la scrittura sul oggetto) e in entrata
    client.setDoOutput(true);
    client.setDoInput(true);
    client.setRequestProperty("charset", "UTF-8");

    //non utilizzo il meccanismo di caching
    client.setUseCaches(false);

```

```

//imposto il tipo di richiesta utilizzato
try
{
    client.setRequestMethod("POST");
}
catch (ProtocolException e)
{
    e.printStackTrace();
    Log.i(this.getClass().getSimpleName(), "Metodo http impostato è errato!");
}

//imposto che in caso di disconnessione ritenti
client.setRequestProperty("Connection", "Keep-Alive");

//imposto il timeout della connessione ad infinito
client.setConnectTimeout(0);

//imposto il content type multipart + il delimitatore/separatore della form
client.setRequestProperty("Content-Type", "multipart/form-data; boundary=" + separatore);

//user agent fake
client.setRequestProperty("User-Agent", "CodeJava Agent");

//apro la connessione (da qui non posso più impostare nessuna proprietà di connessione)
try {
    client.connect();
    Log.i(this.getClass().getSimpleName(), "Connessione aperta...");

    //STREAM
    //Faccio lo stream di quello che voglio inviare sulla connessione

    //Comincio a scrivere sulla connessione
    DataOutputStream dos = new DataOutputStream(client.getOutputStream());

    //Parametri post standard
    dos.writeBytes(CreaMsgPost(campoUno, nomeUtente));
    dos.writeBytes(CreaMsgPost(campoDue, idDispositivo.toString()));
    //fine parametri

    //invio i media
    for (FileMedia media:listMedia)
    {
        Log.i(this.getClass().getSimpleName(), "---\nUpload di "+media.getNome()+"\n DimensioneOriginale: "+media.getDimensione());

        dos.writeBytes(CreaHeadMsgMedia(media.getNome(), media.getMimeType()));

        //comprimo il media
        byte [] tmp=CreaBodyMsgMedia(media.getPath(),qualitaJpeg);
        //setto la nuova dimensione per inviarla al server in quanto la compressione fa perdere i metadati
        media.setDimensione(tmp.length);
        dos.write(tmp);
        dos.writeBytes(CreaTailMsgMedia());
    }

    //chiudo la richiesta
    dos.writeBytes(FineRichiesta());
    Log.i(this.getClass().getSimpleName(), "Richiesta http conclusa.");
} catch (IOException e) {
    e.printStackTrace();
    Log.i(this.getClass().getSimpleName(), "Errore durante l'upload");
}

//Leggo la risposta dal server
switch (LetturaRispostaServer(client))
{
    case 200:
        Log.i(this.getClass().getSimpleName(), "Upload ok! Risposta server: 200 OK!");
        //aggiungo i record sul db remoto
        if(AggiungiMediaDbRemoto(listMedia,nomeUtente,idDispositivo))
        {
            //aggiungo il media nel db locale
            if(this.dBgestione.SyncListMedia(listMedia))
            {
                //imposto che il media è sul server
            }
        }
    }
}

```

```

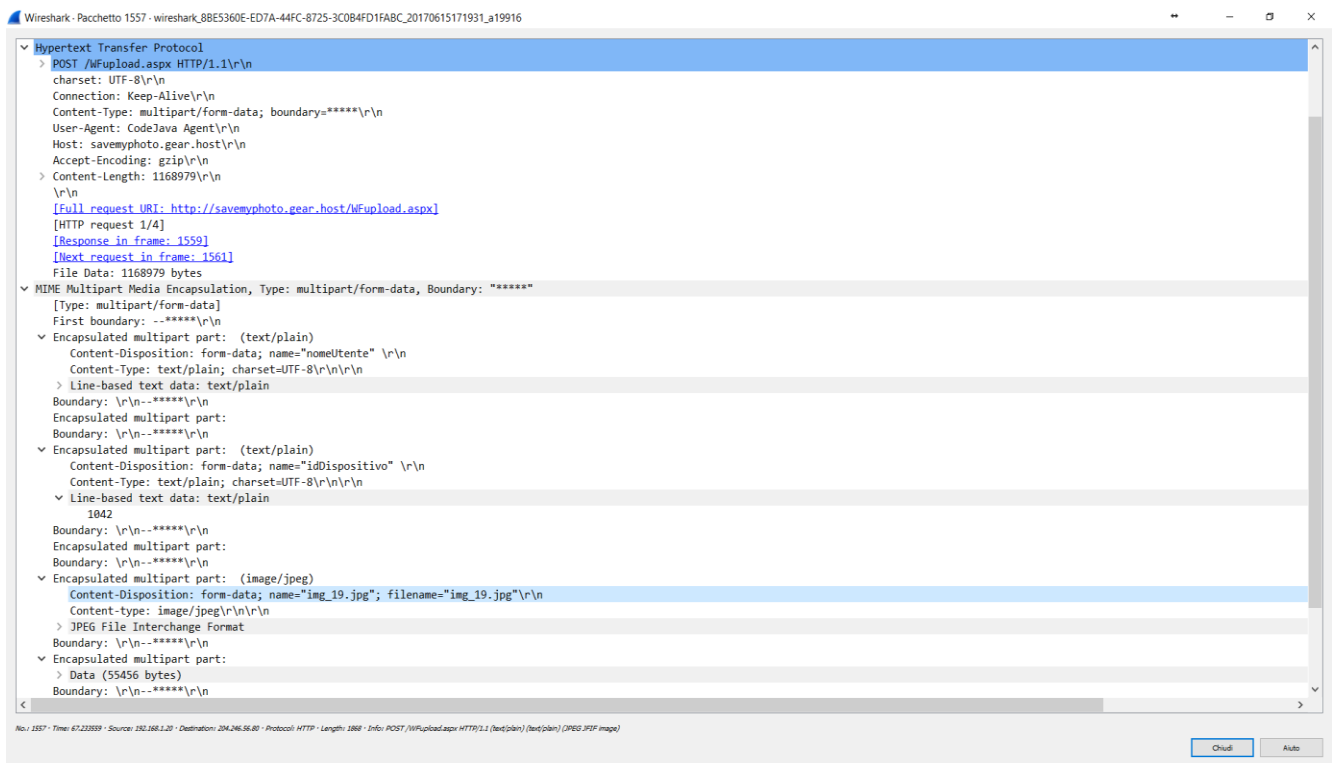
        for (FileMedia media:listMedia)
        {
            media.setSuServer(true);
        }
        //DISCONNESSIONE
        client.disconnect();
        return true;
    }
}
break;
}
//DISCONNESSIONE
client.disconnect();
return false;
}

```

La richiesta POST, a differenza di GET, scrive i parametri all'interno del body, e non nella Url; io utilizzo una richiesta POST non standard, bensì multipart.

Quest'ultima consente di "allegare" più di un file all'interno del body, specificando per ognuno il mimeType e un boundary/separatore per delimitarne il contenuto.

Qui un esempio della richiesta (catturato con wireshark):



Conclusioni

Il progetto è ancora in fase di sviluppo; mancano ancora delle funzionalità da sviluppare e ci sono delle criticità di sicurezza da risolvere. Nel complesso grazie al lavoro che ho svolto e alle conoscenze acquisite a scuola, sono riuscito ad affrontare tutte le problematiche che si sono presentate, acquisendo dimestichezza con l'ide "Android Studio" e con i retroscena del sistema operativo android.

Sitografia e fonti

www.codejava.net

www.nigeapptuts.com

www.easywsdl.com

www.developer.android.com

www.stackoverflow.com