Tec. desenvolupament dispositius mòbils aula 1

Índex

Descripció de la pràctica a realitzar

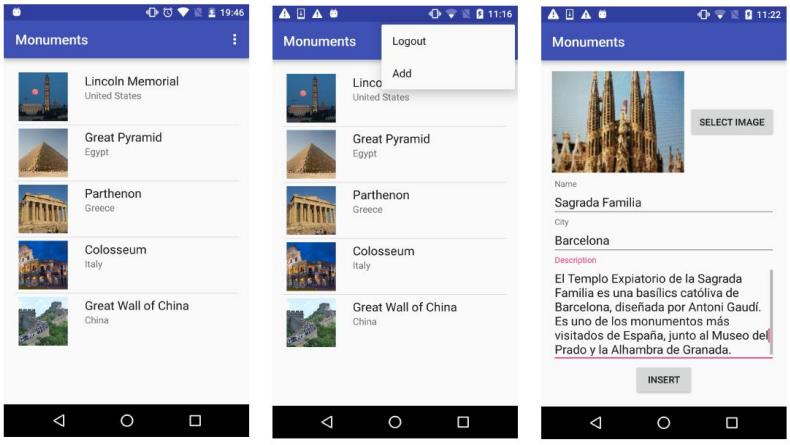
En aquesta segon lliurament de la pràctica ampliarem l'aplicació desenvolupada en el primer lliurament, afegint noves funcionalitats.

L'enunciat d'aquest lliurament consta de dos exercicis. El primer exercici és obligatori, mentre que el segon és optatiu i no afecta la nota d'aquest lliurament. No obstant això, per obtenir la nota final màxima de la pràctica (A) cal haver resolt correctament aquest exercici optatiu. En cas de no resoldre-ho, només es podrà aspirar com a màxim a una nota final de pràctiques de B (a l'hora de fer la mitjana entre les dues entregues, la nota màxima possible serà de 8 sobre 10). Per aquest motiu i per la seva utilitat pràctica, es recomana a tots els estudiants realitzar aquest exercici optatiu, i molt especialment a aquells que no cursaran altres assignatures de desenvolupament d'apps com part del seu itinerari acadèmic.

Heu treballar a partir de la solució oficial del primer lliurament (i no sobre la vostra solució) per evitar reproduir els possibles errors del primer lliurament en aquesta segona i perquè conté codi que no estava disponible a la primera pràctica. Així doncs, per començar a treballar haureu descarregar i obrir el projecte solució del primer lliurament.

Exercici 1. Menú per afegir monuments (Obligatori)

La nostra aplicació del primer lliurament tenia com funcionalitats: autenticar amb usuari i contrasenya, mostrar un llistat de monuments i mostrar la descripció d'aquests. En aquest exercici afegirem una altra funcionalitat: afegir un nou monument al llistat.



Llistat d'elements | Menú amb opció d'afegir | Vista afegir monument

Els passos per realitzar aquesta tasca són els següents:

- Afegir un menú al llistat de productes (ListMonumentActivity). Aquest menú haurà de tenir un única ítem que es dirà Add.
  Teniu més informació sobre els menús a la documentació oficial d'Android: http://developer.android.com/intl/es/guide/topics/ui/menus.html

Afegit a menu_main.xml el codi:

```
<item android:id="@+id/action_add"
   android:title="@string/menu_add"
   android:orderInCategory="10" /> //value less than the item action_logout to show it first
```

Afegit a values -> strings.xml el codi:

```
<string name="menu_add">Add</string>
```

- Crear una activity a partir d'Empty Activity que es dirà AddMonumentActivity.

```
Anar a File -> New-> Activity-> Empty Activity, utilitzar a Activity
Name: AddMonumentActivity y Package name:
edu.uoc.monuments.ui.activities
```

- Quan premin en l'ítem Add, haureu de llançar l'activitat AddMonumentActivity, tenint en compte que ha d'informar sobre si s'ha realitzat la inserció del nou monument i si s'ha de recarregar la llista de productes. Teniu més informació sobre com recuperar el resultat d'una activitat en la documentació oficial d'Android: http://developer.android.com/intl/es/training/basics/intents/result.html

A la classe ListMonumentActivity, perquè al seleccionar Add del menú ens obrirà la activity AddMonumentActivity, dins el mètode onOptionsItemSelected afegir el codi:

```
if (id == R.id.action_add) {
   ApplicationUtils.setUserLoginState(getApplicationContext(), false);

   Intent AddMonumentIntent = new Intent(this, AddMonumentActivity.class);
   startActivityForResult(AddMonumentIntent,request_code);

   return true;
}
```

A la classe ListMonumentActivity, definim la variable para el paràmetre que s'enviarà

```
int request_code = 1;  // The request code
```

També afegirem un mètode per tractar el resultat que torna la activitat AddMonumentActivity quan es tanca, si OK tindrem al Logcat "Resultado devuelto OK" al Logcat i si es cancel·la "Resultado devuelto KO".

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == request_code) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            Log.d(LibraryConstants.TAG,"Resultado devuelto OK");
            monumentAdapter.notifyDataSetChanged(); // We use notifyDataSetChanged method
to inform that the list data is modified and views must be refreshed
        } else {
            Log.d(LibraryConstants.TAG,"Resultado devuelto KO");
        }
    }
}
```

La classe AddMonumentActivity haurà de retornar si el monument s'ha afegit o no i per tant cal o no actualitzar el llistat de monuments. Primer sobre la classe AddMonumentActivity farem un implements View.OnClickListener:

```
public class AddMonumentActivity extends AppCompatActivity implements View.OnClickListener{
```

Seguidament en el mètode onCreate afegim dos butons un que simularà que s'ha afegit un monument i un altre que s'ha cancel·lat la operació d'afegir. Primer els inicialitzarem i inicialitzarem listeners per a cada butó.

```
// Set button views
Button buttonSelectOk = (Button) findViewById(R.id.buttonOK);
Button buttonSelectKo = (Button) findViewById(R.id.buttonCancel);

// Set listeners als butons
buttonSelectOk.setOnClickListener(this);
buttonSelectKo.setOnClickListener(this);
```

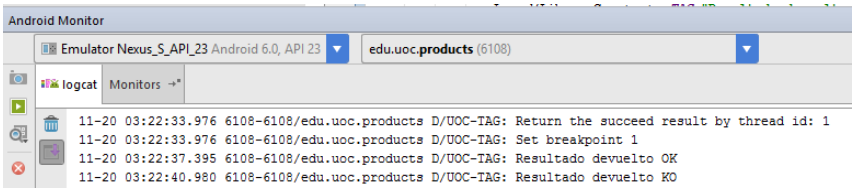
També a la mateixa classe, redefinirim el método onclick per tractar els clics dels botón, on a cada cas es farà un set del resultat a tornar i es finalitzarà l'activitat per tornar a l'activitat anterior.

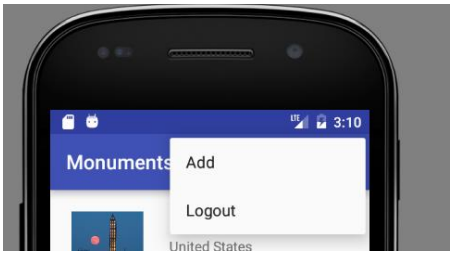
```
@Override
 public void onClick(View v) {
    switch (v.getId()) {
       case R.id.buttonOK:
          setResult(RESULT_OK);
          Toast.makeText(this,"Select OK!!! =",Toast.LENGTH_SHORT).show();
          finish();
          break;
       case R.id.buttonCancel:
          setResult(RESULT_CANCELED);
          Toast.makeText(this,"Select Cancel!!! =",Toast.LENGTH_SHORT).show();

          finish();
          break;
       case View.NO_ID:
       default:
          // TODO Auto-generated method stub

          break;
    }
}
```

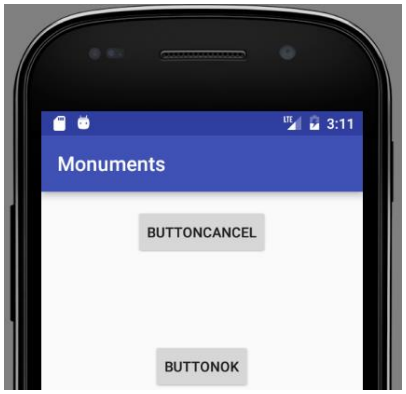
També al layout activity_add_monument.xml afegirem els dos botons.

```
<Button
   android:text="ButtonCancel"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/buttonCancel"
   android:layout_centerHorizontal="true" />

<Button
   android:text="ButtonOK"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginTop="85dp"
   android:id="@+id/buttonOK"
   android:layout_below="@+id/buttonCancel"
   android:layout_centerHorizontal="true" />
```

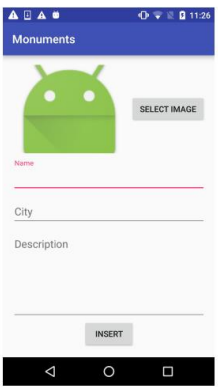
Tenim el resultat següent, tant en el Logcat com les captures de pantalla:





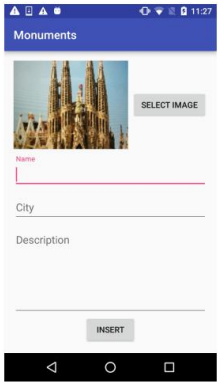


- Modificar aquest layout amb camps de tipus text que ens permetin afegir: nom, país, vista i una imatge (IMAGEVIEW). Afegir dos botons, un per capturar una imatge amb el dispositiu ("Select Image") i un altre per realitzar la inserció ("Insert"). Quan es captura la imatge, s'ha de refrescar el IMAGEVIEW perquè l'usuari la pugui veure.



Vista afegir monumento

Captura d'imagen



Es refresca l'ImageView

A l'activitat AddMonumetActivity, dins el mètode onCreate on teníem els dos botons, els canviarem per adequar-los a les noves necessitats.

```
// Set button and image views
Button buttonInsert = (Button) findViewById(R.id.buttonInsert);
Button buttonSelectImage = (Button) findViewById(R.id.buttonSelectImage);
mImage = (ImageView) findViewById(R.id.monument_image);

// Set listeners als butons
buttonInsert.setOnClickListener(this);
buttonSelectImage.setOnClickListener(this);
```

Al layout activity_add_monument també actualitzarem al nou id.

```
<Button
   android:id="@+id/buttonSelectImage"
   style="?android:textAppearanceSmall"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="SELECT IMAGE"
   android:textStyle="bold"
   android:layout_gravity="center" />

<Button
   android:id="@+id/buttonInsert"
   style="?android:textAppearanceSmall"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginTop="16dp"
   android:text="INSERT"
   android:textStyle="bold"
   android:textAlignment="center"
   android:layout_gravity="center" />
```

- Es pot capturar una imatge en Android de dues maneres, en miniatura o en mida real. Si es vol capturar a mida real s'ha de gestionar dos permisos a android, el de captura d'imatge i el d'emmagatzematge de dades a la memòria externa. És per aquest motiu que la captura que s'ha de fer en aquesta practica de la imatge han de ser en miniatura (thumbnail).

  Teniu més informació sobre com capturar una image en la documentació oficial d'Android: https://developer.android.com/training/camera/photobasics.html

Dins de la classe AddMonumentAcvtivity, fora dels mètodes existents, cal afegir.

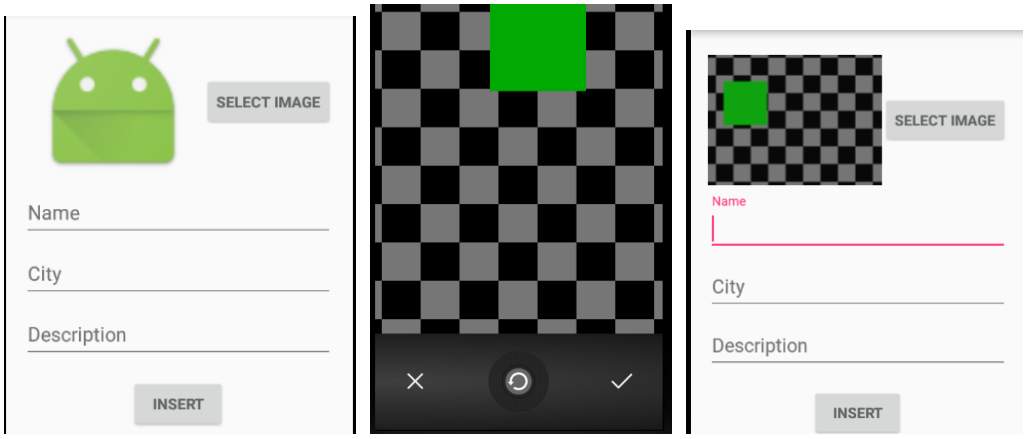
```
static final int REQUEST_IMAGE_CAPTURE = 1;

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImage.setImageBitmap(imageBitmap);
    }
}

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

Encara a la mateixa classe, pero al método onClick cridarem al mètode dispatchTakePictureIntent quan es premi el botó “SELECT IMAGE”.

```
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.buttonInsert:
            setResult(RESULT_OK);
            Toast.makeText(this,"Select Insert!!! =",Toast.LENGTH_SHORT).show();
            //return "RESULT_OK";
            finish();
            break;
        case R.id.buttonSelectImage:
            Toast.makeText(this,"Select Image!!! =",Toast.LENGTH_SHORT).show();
            //When image captura, refresh image

            dispatchTakePictureIntent();

            break;
        case View.NO_ID:
        default:
            // TODO Auto-generated method stub
```

```
            break;
        }
    }
```

Les següent captura mostren com es selecciona una imatge en miniatura “thumbnail”.



- En prémer el botó "Insert", haureu de realitzar la inserció del nou monument cridant al següent mètode

```
LibraryManager.getInstance(getApplicationContext()).saveMonumentInBackgro
und(name, country, description, bitmap, new GetCallback<Monument>() {
@Override
public void onSuccess(Monument monument) {
}
@Override
public void onFailure(Throwable e) {
}
});
```

Afegeixo l’import següent:

```
import edu.uoc.library.calback.GetCallback;
import android.graphics.Canvas;
import android.graphics.drawable.Drawable;
import android.graphics.Bitmap;
```

Abans haurem d’assignar valor als diferents paràmetres, name, country, description, bitmap.
Dins de la activitat

```
private AutoCompleteTextView monument_nameView;
private AutoCompleteTextView monument_countryView;
private AutoCompleteTextView monument_descriptionView;
```

Dins de onCreate

```
mImage = (ImageView) findViewById(R.id.monument_image);

//Set textviews
monument_nameView = (AutoCompleteTextView) findViewById(R.id.monument_name);
monument_countryView = (AutoCompleteTextView) findViewById(R.id.monument_city);
monument_descriptionView = (AutoCompleteTextView) findViewById(R.id.monument_description);
```

Dins el mètode onClick i en el cas que es faci clic en obtindrem els valors introduïts per Name,

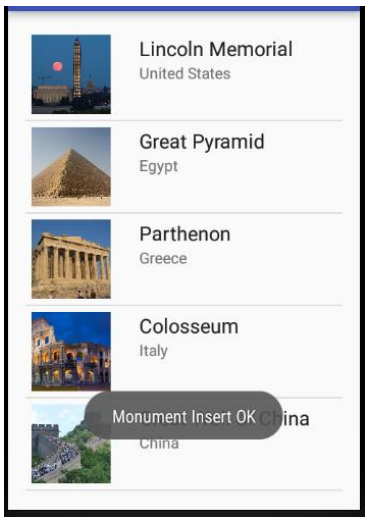
Country, Description i el Bitmap de la imatge per cridar el mètode saveMonumentInBakground. Per copiar la imatge ens ajudarem de la classe drawable i canvas per crear un nou bitmap amb les característiques de l'original.

```
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.buttonInsert:
            setResult(RESULT_OK);

            // Store values in the same type need by saveMonumentInBackground method.
            String name = monument_nameView.getText().toString();
            String country = monument_countryView.getText().toString();
            String description = monument_descriptionView.getText().toString();

            //The image must be a bitmap type
            //To copy the image, we get a Drawable from the mImage, create a Bitmap that will contain our image.
            //Initialise a canvas with the Bitmap created and Use a canvas taking bitmap (the copy of bitmap) as a
parameter
            Drawable mImageDrawable = ((ImageView) mImage).getDrawable();

            Bitmap bitmap = null;
            bitmap =
Bitmap.createBitmap(mImageDrawable.getIntrinsicWidth(),mImageDrawable.getIntrinsicHeight(),Bitmap.Config.ARGB_
8888);

            Canvas canvas = new Canvas(bitmap);
            mImageDrawable.setBounds(0,0,canvas.getWidth(),canvas.getHeight());
            mImageDrawable.draw(canvas);

            LibraryManager.getInstance(getApplicationContext()).saveMonumentInBackground(name, country, description,
bitmap, new GetCallback<Monument>() {
                @Override
                public void onSuccess(Monument monument) {
                    if (monument != null) {
                        Toast.makeText(getApplicationContext(),"Monument Insert OK",Toast.LENGTH_SHORT).show();
                    }

                }
                @Override
                public void onFailure(Throwable e) {
                    Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_SHORT).show();
                }
            });


            finish();
            break;
        case R.id.buttonSelectImage:
            //When image captured, refresh image
            dispatchTakePictureIntent();

            break;
        case View.NO_ID:
        default:
            // TODO Auto-generated method stub

            break;
    }
}
```

- En el mètode ~~onFailure()~~ onSuccess() del GetCallback haureu de realitzar les següents tasques:
    - Mostrar un missatge indicant si hi ha un error en la inserció. Podeu trobar més informació sobre com mostrar missatges a la documentació oficial Android: https://developer.android.com/guide/topics/ui/notifiers/toasts.html
    - Tancar l'activity AddMonumentActivity.
    - Realitzar la recarrega del nou document a la llista de l'activity ListMonumentActivity.

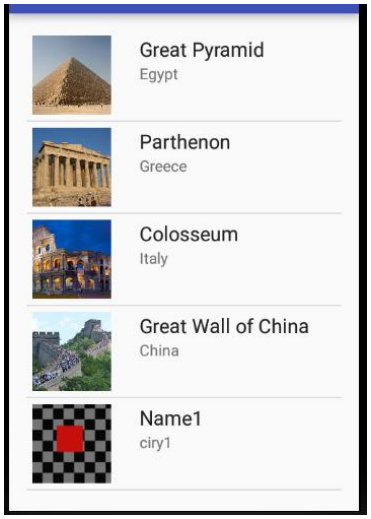
Utilitzarem la classe Toast dins del mètode onClick quan per mostrar la notificació de quan s'ha afegit un monument a la llista, amb el següent codi que donarà el resultat de la captura més abaix.

```
LibraryManager.getInstance(getApplicationContext()).saveMonumentInBackground(name, country, description, bitmap,
new GetCallback<Monument>() {
    @Override
    public void onSuccess(Monument monument) {
        if (monument != null) {
            Toast.makeText(getApplicationContext(),"Monument Insert OK",Toast.LENGTH_SHORT).show();
        }

    }
    @Override
    public void onFailure(Throwable e) {
        Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_SHORT).show();
    }
});

//Use finish to retorn the previous activity
finish();
```



Per un altra banda per recarregar la llista amb el nou monument, dins la classe ListMonumentActivity afegirem el següent codi, que fa que quan l'activitat es resumeix es torna a obtenir el llistat de monuments i es visualitzarà actualitzat.

```
@Override
protected void onResume() {
    super.onResume();

    LibraryManager.getInstance(getApplicationContext()).getAllMonuments(new GetCallback<List<Monument>>() {
        @Override
        public void onSuccess(List<Monument> result) {
            mProgressBar.setVisibility(View.GONE);
            Log.d(LibraryConstants.TAG, "Set breakpoint 1");
            monumentList.clear();
            monumentList.addAll(result);
            monumentAdapter.notifyDataSetChanged();
        }

        @Override
        public void onFailure(Throwable e) {
            mProgressBar.setVisibility(View.GONE);
            Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}
```



Per a aquest exercici, cal adjuntar el codi font i el fitxer .apk. A més, cal adjuntar una captura de pantalla de la vista d'afegir monument en el document a lliurar.

Segons es demana a la pregunta 1, s'adjunta per separat el codi en el fitxer TDDM_Practica2_Exercici1.zip i el fitxer .apk amb nom app-debug.apk.

Les captures de pantalla de la vista d'afegir monument estan incloses en aquest mateix document, veure punts anteriors d'aquest document.

Exercici 2. Dades JSON (Optatiu)

En moltes ocasions no disposem de llibreries com les llibreries de dades de la UOC i rebem les dades del nostre back-end en format XML o JSON. En aquest exercici anem a enfrontar-nos a aquesta situació.

Responeu a les següents preguntes:

Pregunta 2.1.
Per què serveix el format JSON? Quines característiques té?

JSON, JavaScript Object Notation, es un format per intercanviar dades. Com a característiques podem dir que utilitza convencions àmpliament conegudes per programadors, es fàcil llegir, escriu per persones i fàcilment interpretable per màquines. També es independent del llenguatge de programació, per tant es pot utilitzar per intercanviar en qualsevol llenguatge escollit.

Està constituït per dues estructures, una col·lecció de parells de nom/valor (objecte, registre, estructura) i una llista ordenada de valors (arregles, vectors, llista).

Pregunta 2.2.
El format JSON que es presenta a continuació té tres errors de format, detecta'ls i mostra com hauria de ser el format correcte:

```
{
"menu": [{
"primeros": {
"plato1": "gazpacho",
},
{
```

```
"plato2": "ensalada verde"
},
{
"plato3": "judias"
},
{
"plato4": "lentejas"
},
"segundos": [{
"plato1": "pescado de día"
}, {
"plato2": "hummus"
}, {
"plato3": "lomo a la plancha"
}, {
"plato4": "hamburguesa"
}],
"postres": [{
"postre1": "yogurt"
}, {
"postre2": "fruta"
}, {
"postre3": "helado"
}],
}]
}
```

Nota: A Internet trobareu pàgines web que validen els formats JSON.

Seguint la estructura JSON i la web http://jsonlint.com/ el código JSON següent no conté errors.

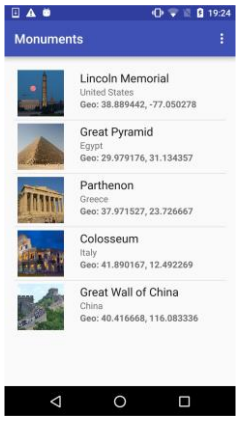
```
{
    "menu": [{
        "primeros": {
                "plato1": "gazpacho",
                "plato2": "ensalada verde",
                "plato3": "judias",
                "plato4": "lentejas"
        },
        "segundos": [{
                "plato1": "pescado de día",
                "plato2": "hummus",
                "plato3": "lomo a la plancha",
                "plato4": "hamburguesa"
        }],
        "postres": [{
                "postre1": "yogurt",
                "postre2": "fruta",
                "postre3": "helado"
        }]
    }]
}
```

Pregunta 2.3.
Per a aquest exercici heu de continuar amb la solució de l'exercici 1. El fitxer monuments.json conté informació que la llibreria de l'aplicació no està processant. La informació que no es processa és la relativa a les característiques del monument (any, localització i nombre de visitants);

```
"id": 1,
"name": "Lincoln Memorial",
"country": "United States",
"description": "The Lincoln Memorial is an American national monument
built to honor the 16th President of the United States, Abraham
Lincoln. It is located on the western end of the National Mall in
Washington, D.C., across from the Washington Monument. The architect
was Henry Bacon; the designer of the primary statue – Abraham
Lincoln, 1920 – was Daniel Chester French; the Lincoln statue was
```

carved by the Piccirilli Brothers; and the painter of the interior murals was Jules Guerin. Dedicated in 1922, it is one of several monuments built to honor an American president. It has always been a major tourist attraction and since the 1930s has been a symbolic center focused on race relations.",
"image": "washington.jpg",
"characteristics": {
"year": "1922",
"location": {
"latitude": 38.889444,
"longitude": -77.050278
},
"visitors": 3638806
}

Heu de modificar la llibreria perquè aquesta informació es recuperi en l'aplicació i es mostri la latitud i la longitud en la llista dels monuments com mostra la captura de pantalla de a continuació:



Es recomana seguir els següents passos:

- Afegir les classes amb els models de dades de les característiques del momument.
- Modificar el layout adapter_monument del llistat per mostrar la latitud i longitud del monument.
- Afegir la informació de la localització en el camp de text

Per aquesta pregunta és precís adjuntar el codi font. A més a més, s'ha d'ajuntar una captura de pantalla de la vista al document a lliurar.

Tenint en ment la classe monument i l'estructura json de les dades, a part de la classe Monument he creat dos classes noves, Characteristics i Location que coincideixen amb valors de l'estructura JSON. Dins de la classe Characteristics definirem un valor tipus Location, que estarà definit a la classe Location. Dins la classe Location definirem el mètode getLocation per recuperar les dades longitude i latitude com a string.

```
public class Characteristics {
   @SerializedName("year")
   private String year;

   @SerializedName("location")
   private Location location;

   @SerializedName("visitors")
   private int visitors;

   public String getYear() {
      return year ;
   }
```

```
    public Integer getvisitors() {
        return visitors ;
    }

    public Location getLocation(){return location;}

}

public class Location {

    @SerializedName("latitude")
    private double latitude;

    @SerializedName("longitude")
    private double longitude;

    public double getLatitude() { return latitude ; }
    public double getLongitude() { return longitude ;}

    public String getLocation(){
        String  result = Double.toString(latitude) + ", " + Double.toString(longitude);
        return result;}
}
```

A la classe Monument.java afegirem i definim el mètode getCharacteristics que ens permetrà obtenir les característiques de cada monument.

```
@SerializedName("characteristics")
private Characteristics characteristics;

public Characteristics getCharacteristics(){return characteristics;}
```

A la classe MonumentAdapter.java afegim dins del Holder un textview per a la nova dada que volem mostrar.

```
static class ViewHolder {
    protected ImageView mThumbnail;
    protected TextView mName;
    protected TextView mCountry;
    protected TextView mLocation;
}
```

També afegim a la mateixa classe, en el mètode getView, associem la localització al layout i també fem un get primer del objecte Characteristics que inclourà l'objecte Location que es el que ens interesa. Com tenim creat el mètode getLocation que ens retorna en un valor string l'utilitzarem per recuperar la posició Geolocalització.

```
public View getView(int position, View convertView, ViewGroup parent) {
    View view;
    final ViewHolder viewHolder;

    if (convertView == null || convertView.getTag() == null) {
        viewHolder = new ViewHolder();
        view = LayoutInflater.from(context).inflate(R.layout.adapter_monument, parent, false);

        viewHolder.mName = (TextView) view.findViewById(R.id.name);
        viewHolder.mCountry = (TextView) view.findViewById(R.id.country);
        viewHolder.mThumbnail = (ImageView) view.findViewById(R.id.thumbnail);
        viewHolder.mLocation = (TextView) view.findViewById(R.id.location);
        view.setTag(viewHolder);

    } else {
        viewHolder = (ViewHolder) convertView.getTag();
        view = convertView;
    }
    // Get monument object
    Monument monument = monumentList.get(position);
    // Set text with the item name
    viewHolder.mName.setText(monument.getName());
    // Set country
    viewHolder.mCountry.setText(monument.getCountry());

    Log.d(LibraryConstants.TAG, "Pre-characteristics Extract OK");
    //Get the JSON characteristics object
```

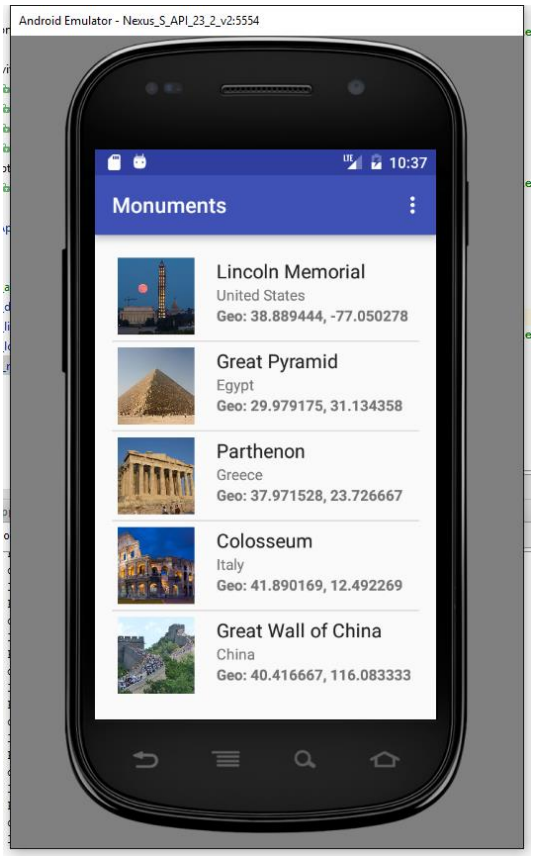
```
    Characteristics characteristicsObject = monument.getCharacteristics();
    Log.d(LibraryConstants.TAG, "characteristics Extract OK");

    Location locationObject = characteristicsObject.getLocation();
    Log.d(LibraryConstants.TAG, "location Extract OK");

    String location = "Geo: " + locationObject.getLocation();

    //Set location
    viewHolder.mLocation.setText(location);

    // Set image
    Bitmap image = LibraryManager.getInstance(context).getImage(monument.getImagePath());
    viewHolder.mThumbnail.setImageBitmap(image);

    return view;
}
```

Per últim al layout adaptar_monument.xml afegim el següent codi per afegir la geolocalització.

```
<TextView
    android:id="@+id/location"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:gravity="center"
    android:layout_marginLeft="10dp"
    android:textAppearance="@android:style/TextAppearance.Holo.Small"
    android:textStyle="normal|bold" />
```

Segons es demana a la pregunta 2 s'adjunta el codi en el fitxer TDDM_Practica2_Exercici2.zip.

La captura de pantalla de la vista es la següent:



Bibliografía

Android.(2016). Menús | Android Developers. US:Android. http://developer.android.com/intl/es/guide/topics/ui/menus.html

Android.(2016). Cómo obtener un resultado de una actividad | Android Developers. US:Android. http://developer.android.com/intl/es/training/basics/intents/result.html

Android.(2016). Taking Photos Simply | Android Developers. US:Android. https://developer.android.com/training/camera/photobasics.html

Android.(2016). Toasts | Android Developers. US:Android. https://developer.android.com/guide/topics/ui/notifiers/toasts.html

SekthDroid. (31/01/2013).Retornar valor desde una Actividad Secundaria en Android. ES:SekthDroid. https://sekthdroid.wordpress.com/2013/01/31/retornar-valor-desde-una-actividad-secundaria-en-android/

Workassis. (2016). Android three ways to set click listener – wiki.US: Workassis. http://wiki.workassis.com/android-three-ways-to-set-click-listener/

Arif Nadeem. (11/02/2013). How to correctly implement BaseAdapter.notifyDataSetChanged() in Android – Atom. http://androidadapternotifiydatasetchanged.blogspot.com.es/

JSON.ORG. ES:JSON. www.json.org/json-es.html

JSONLint - The JSON Validator. . jsonlint.com.