|  |
| --- |
| Bucharest University of Economic Studies |
| Event Planner Application |
| DAM Project |

|  |
| --- |
| Becheru Laura  Group 1077 |

Contents

[Introduction 2](#_Toc503727254)

[Solution 2](#_Toc503727255)

[Implementation 3](#_Toc503727256)

[Conclusion 6](#_Toc503727257)

[Anexe 7](#_Toc503727258)

# Introduction

The project that I have made is addressing to people that are accustomed to organizing events. Usually, people are tracking the events they are planning in their notebook but notebooks are not always handy, whilst phones are always somewhere around us. People find it easy to pick out the phone from their pocket in order to record all the details related to the event they are planning, so, the purpose of this application is to help them keep track to the most important details, such the date, the place, the fee, if there is one, and so on. This application will be mostly use in a business framework, but can also be used outside it.

# Solution

For keeping the data inputted, the user must authenticate. In order to authenticate he must create a simple account with a name, email and a password. For the account to be created he has to agree with the Terms and Conditions. After creating the account he is free to login. The login will give him access to the Conference Activity in which he will input the details of the event: name, organization, related, field, duration, fee, data, speakers, and location.

For adding the location he will open another activity which has a couple of fields related to it and a button that will lead him to google maps, in order to find the exact spot of the location and its address. Moreover, he can check the gallery for some pictures of a few places where past events where held and he can update the gallery by uploading new pictures of other locations. He also has a small option to change the color of the photo uploaded in grayscale and to save it in his phone memory.

After selecting the location he can click the add button and a list of events will appear with his already recorded events and the new one. By clicking an item of the list, he can delete a conference and by log clicking he can update it.

Furthermore he has the option to check this list whenever he pleases from the button in the main activity, where he can also rate the app and check the company partners. In this activity he also has the option to log out.

# Implementation

For developing the application I have tried to carry out the tasks from each phase of the projects.

For the input forms I have used several types of controls

*//EDITBOX***name** = (EditText) findViewById(R.id.***boxName***);  
  
*//SPINNER***field** = (Spinner) findViewById(R.id.***spField***);  
ArrayAdapter<CharSequence> genreOptions = ArrayAdapter.*createFromResource*(**this**, R.array.***field\_array***,  
 android.R.layout.***simple\_spinner\_item***);  
genreOptions.setDropDownViewResource(android.R.layout.***simple\_spinner\_dropdown\_item***);  
**field**.setAdapter(genreOptions);  
  
*//DATE PICKER***date** = (Button) findViewById(R.id.***dateBtn***);  
**date**.setOnClickListener(**dateBtnClickList**);  
  
*//AUTOCOMPLETE***organizer** = (AutoCompleteTextView) findViewById(R.id.***orgAutoText***);  
String[] organiz = getResources().getStringArray(R.array.***array\_organiz***);  
ArrayAdapter<String> organizAdapter = **new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_list\_item\_1***, organiz);  
**organizer**.setAdapter(organizAdapter);

I have used, for the lists,the ListView and GridView elements with adapters:

**lv** = (ListView) findViewById(R.id.***cListView***);  
**confList** = **new** ArrayList<Conference>();

………

**final** ArrayAdapter<Conference> confListAdapter = **new** ArrayAdapter<Conference>(**this**, android.R.layout.***simple\_list\_item\_1***, **confList**);  
**lv**.setAdapter(confListAdapter);

For GridView I have created a new adapter

PartnersAdapter gva= **new** PartnersAdapter(**parteners**,**this**);  
GridView gv = findViewById(R.id.***gridViewID***);  
gv.setAdapter(gva);

**public** PartnersAdapter(ArrayList<String> part, Context context)  
{  
 **this**.**partners** = part;  
 **this**.**context** = context;  
 **layoutInflater** = LayoutInflater.*from*(context);  
}

For the data base I have use a SQLite helper

**public class** SQLiteHelper **extends** SQLiteOpenHelper {  
  
 **private static final** String ***TAG*** = **"DatabaseHelper"**;  
 **public static** String *DATABASE\_NAME* = **"ConferenceAppDB"**;

and new data can be inserted in the data base through the ConfActivity and RegistrationActivity, each for the table they represent:

**public void** InsertDataIntoSQLiteDatabase(){  
 **if**(**editTextEmptyHolder** == **true**)  
 {  
 **SQLiteDataBaseQueryHolder** = **"INSERT INTO "**+ SQLiteHelper.***TABLE\_NAME\_CONF***+  
 **" (user\_id,name,organizer,field,duration,date,fee,speakers,location) VALUES('"** + **userID**+**"', '"**+**nameHolder**+**"', '"**+**organizerHolder**+**"', '"**+ **fieldHolder**+**"', '"**+ **durationHolder**+**"', '"**+**dateHolder**+**"', '"**+ **feeHolder**+  
 **"', '"**+**speakersHolder**+ **"', '"** + **locationHolder** + **"');"**;  
 **sqLiteDatabaseObj**.execSQL(**SQLiteDataBaseQueryHolder**);  
 **sqLiteDatabaseObj**.close();  
 Toast.*makeText*(ConfActivity.**this**,**"Conference Recorded Successfully"**, Toast.***LENGTH\_LONG***).show();  
 }  
 **else**{  
 Toast.*makeText*(ConfActivity.**this**,**"Please Fill All The Required Fields."**, Toast.***LENGTH\_LONG***).show();  
 }  
}

I am adding new data to the database with the help of 4 fuctions SQLiteDataBaseBuild(), SQLiteTableBuild(),CheckEditTextStatus(), InsertDataIntoSQLiteDatabase(). The first one is opening or creating the data base if is not yet created, the second creates the table if it does not exist, the third checks if there is data in the holders( here I put the data from the edit boxes and so on) and the forth is inserting the data in the table. Furthermore, for the RegisterActivity I have put a condition on the add button so the user cannot register if he has ot agreed with the terms and conditions which can be access by clicking a button. The terms and conditions are downloaded from an external source b using an AsyncTask.

**class** TermsAndConditions **extends** AsyncTask<String,Integer,Bitmap>

In the login page I use a cursor for checking searching into the database an email similar with in email inputted and I also check to see if the password is correct. If not a proper message will be displayed.

I have created a class Session where I have used the SharedPreferences in order to keep the user logged in as long as he wants. If he wishes to logout of this account he can use the logout button from the main activity, button that is setting of the session.

**public class** Session {  
 SharedPreferences **sharedPreferences**;  
 SharedPreferences.Editor **editor**;  
 Context **context**;

The conference list I use a cursor to iterate through the data base in order to retrieve all the information recorded for the current user. Along the function which retrieves the information it has 2 events listeners: 1 for clicking and one for long clicking. The first one is for deleting he entry and the other one for updating.

In the location activity I have several implementations. One of them is the Google Map along with the Google Place Autocomplete. For implementing them I have ppick up an API key from the google developers website. I am adding a listener on the autocomplete fragment in orger to select the place searched and I set the place and its address to the text of the button on which I set an onClick function through which send the location to a text view in the LocActivity.

autocompleteFragment.setOnPlaceSelectedListener(**new** PlaceSelectionListener() {  
 @Override  
 **public void** onPlaceSelected(Place place) {  
 Log.*i*(***TAG***, **"Place: "** + place.getName());  
  
 **namePlace** = place.getName().toString();  
 **placeDetailsStr** = place.getName() + **" -> "** + place.getAddress();  
  
 **coord**=place.getLatLng();  
 **mMap**.addMarker(**new** MarkerOptions().position(**coord**).title(**namePlace**));  
 **mMap**.moveCamera(CameraUpdateFactory.*newLatLngZoom*(**coord**, (**float**) 15));  
  
 **btn**.setText(**placeDetailsStr**);  
 }  
  
 @Override  
 **public void** onError(Status status) {  
 Log.*i*(***TAG***, **"An error occurred: "** + status);  
 }  
});

In the Location Gallery I have used the Firebase data base. For retrieving the images I have used a recycler view adapter which contains a Glide object for loading the images and a holder with an image view and a text view for keeping the image for them and the text. The user can also upload the images to the firebase by importing them from the photo library of the phone with the function onActivityResult ():

**protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
  
 **super**.onActivityResult(requestCode, resultCode, data);  
  
 **if** (requestCode == **Image\_Request\_Code** && resultCode == ***RESULT\_OK*** && data != **null** && data.getData() != **null**) {  
  
 **FilePathUri** = data.getData();  
  
 **try** {  
 Bitmap bitmap = MediaStore.Images.Media.*getBitmap*(getContentResolver(), **FilePathUri**);  
 **SelectImage**.setImageBitmap(bitmap);  
  
 }  
 **catch** (IOException e) {  
  
 e.printStackTrace();  
 }  
 }  
}

And he can uploaded through the UploadImageFileToFirebaseStorage();

In this activity the user can also set a greyscale by using the ColorFilter and changing the values of the colorMatrix and can also save it, using the insertImage() function on the MediaStore.

**SelectImage**.buildDrawingCache();  
 Bitmap bm = **SelectImage**.getDrawingCache();  
 MediaStore.Images.Media.*insertImage*(getContentResolver(), bm, **"img"**, **"from app"**);

One of the best advantages of Android is that is easy to work with its libraries, they are making developers’ work less difficult. Another advantage would be the emulator that helps the developers that do not use a phone with Android.

# Conclusion

There are still many things that can be done for improving the user experience. In my opinion a good technical solution for implementing the application would be to keep the user data in the cloud database, not locally as I have implemented, but since firebase implantation was after the Sqlite, I did not have the time to change it. A cloud data base would record the user even if the application is uninstalled, while the Sqlite only keeps them while the app is in the phone. Other technical solution would be to give the user the option to delete the account or update it. In conclusion the application is not yet a final solution for the problem, but it can surely be improved to enhance user experience.

# Anexe











