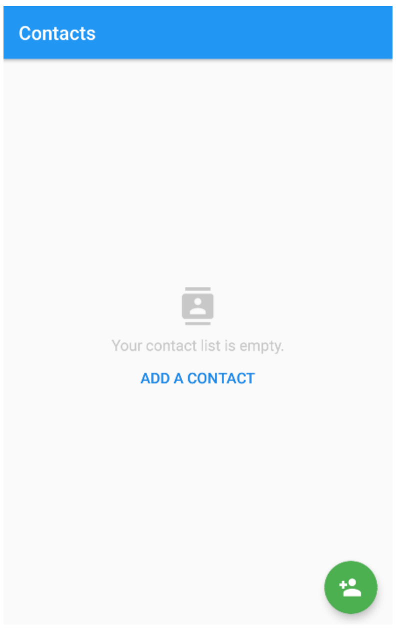
Contacts Manager

Documentation

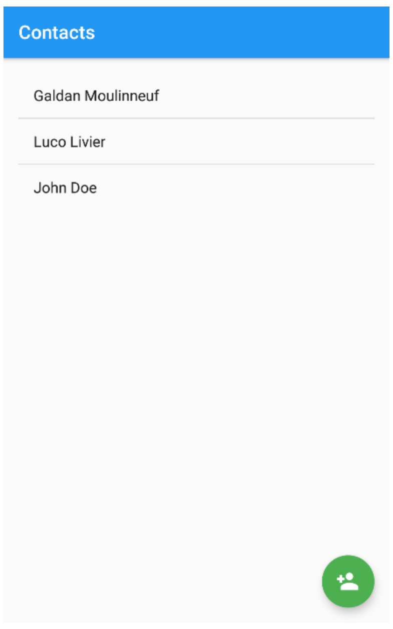
I. Design

This part will explain how and why the app is designed the way it by going through all the activities/fragments.

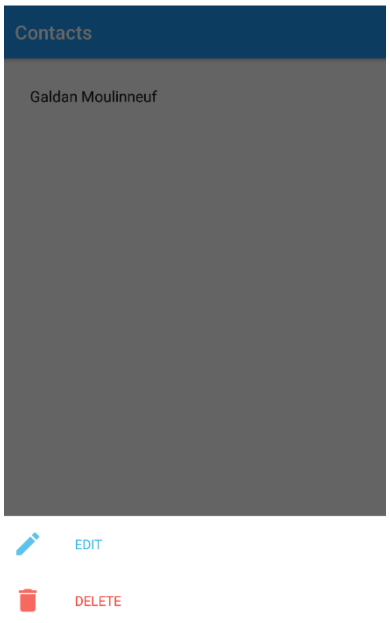
1. ListActivity:



List is empty



List is filled



Long click on a contact

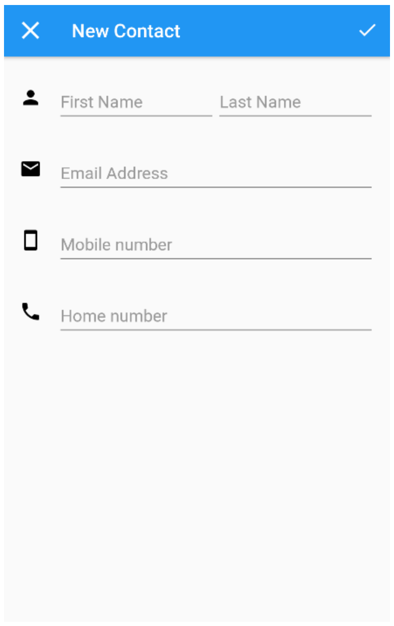
When the app starts, and having no contacts yet, you get the screen on left, to add a contact you can either click on the blue text “ADD A CONTACT” or press the floating button on the bottom right corner.

Once one or more contacts are added, you can see the list that shows the first name and the last name.

If you click on a contact, it will lead you to the contact details activity. If you maintain the touch for 2 seconds, a menu will appear asking you if you want to edit the contact or delete it. If you chose to edit, it will lead you to the contact edit activity, if you chose to delete, it will delete the row from the list.

A CoordinatorLayout is also used as the root to implement the FloatingButton.
The layout used inside is a simple ListView with a LinearLayout for the case where the list is empty.

2. EditActivity:



When you create a new contact, this screen will appear.

First name and last name can only be letters, the email address must be a valid one and mobile/home number only take numbers and some phone special characters.

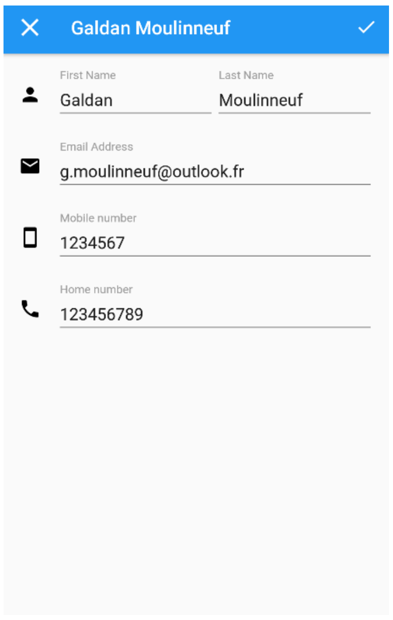
There can only be one combination of the "First name/Last name".

If you want to cancel, you can just touch the cross on the top left corner.

When you are done, you simply touch the check on the top right corner.

Creating a new contact

The layout used is a TableLayout with 4 TableRow to make the information displayed in an ergonomic way.


Galdan Moulinneuf

First Name: Galdan

Last Name: Moulinneuf

Email Address: g.moulinneuf@outlook.fr

Mobile number: 1234567

Home number: 123456789

When you edit a contact, it's first and last name are displayed in the upper Toolbar and the fields are filled with the contact's information.

Same rules apply.

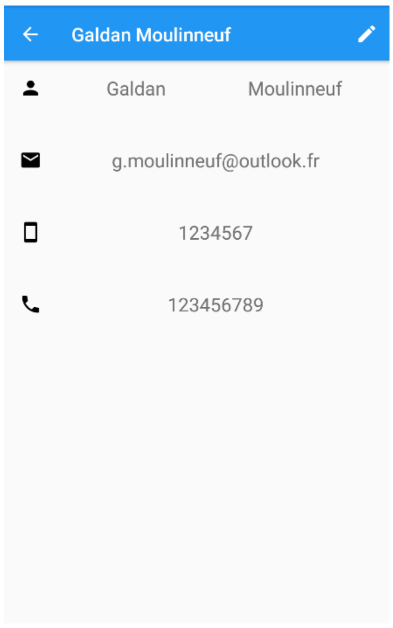
Editing an existing contact

3. DetailsActivity:

When you click on a contact in the list, this screen will appear summarizing all contact’s information.

You can go to the edit activity by clicking on the pen at the top right corner.

The layout is pretty much the same as the edit activity but instead of having editable fields, they are just TextViews.



Contact details

II. Code

This part will explain the code.

The first class to be called is ContactListActivity, in this class there are 5 custom methods.

> initComponents will be in charge to initialize all the component the activity needs for example setting the name of the Toolbar and linking all the components with findViewById.
>
> In setListeners, the components that need to be listened will take their function here like what happen when you click on the floatingButton or when you longPress on an item in the list.
>
> To store the contacts, I used a simple List that takes objects "Contact" which represent what's in my database.
>
> loadDatabase will be in charge to keep the list up to date with the database by requesting it all the contacts. It's also its work to link the List to the ListView. This function will be called will creating the activity AND when resuming it.
>
> goEdit is just a function that create an intent to launch ContactEditActivity and pass it the contact that needs to be edited through intent's extras.
>
> goDetails is the same as goEdit bu it will launch ContactDetailsActivity and just provides the ID of the contact.

ContactListDialog is not an activity but a Fragment (DialogFragment to be precise). It's launched by ContactListActivity when there is a longPress on a contact in the list.

> On the "onCreateView" method, first it will initialize the components by inflating the rootView and then with "findViewById".
>
> Next it will indicate the two buttons (Edit and Delete) what to do when they're clicked.
>
> Then, it will place the fragment at the bottom of the screen and finally set the animations when appearing/disappearing.
>
> Now, the setListeners method takes an interface called IListeners which is an inner class interface. It's used to pass listeners to this class from another class. For example, in ContactListActivity it will create an instance of this interface by overriding the two methods which are actually the buttons (Edit and Delete) listeners. Then ContactListActivity will call the setListeners method with the instance of the interface that's been overridden.

ContactEditActivity is the activity that will be used when you want to add OR edit a contact thanks to a Boolean, the activity knows what case it is.

> initComponents is the same as before and will initialize the components and doing the “findViewById work”.
>
> setListeners is also same as before and set the listener for the confirm button on the ToolBar menu.
>
> onCreateOptionsMenu will inflate the button that needs to be in the Toolbar menu.
>
> onOptionsItemSelected will manage what to do when the button in the Toolbar is pressed, here, it will make some checks on the fields and update the database if all the fields are corrects.
>
> setErrorEditXXX: those functions are used in onOptionsItemSelect and are checking the content of the corresponding field (first and last names have only letters, home and mobile phone have only digits, email is in a correct email format). They return true if everything is ok, false otherwise.
>
> setErrorDuplicate is a function also used in onOptionsItemSelect and is likely to others “setErrorEditXXX” functions except that this one will check in the database if the combination of last and first name isn’t already existing.
>
> saveChanges is in charge to save/update the contact in the database.

ContactDetailsActivity is the activity that’s launch when you press a contact in ContactListActivity.

> > This class is pretty much the same as ContactEditActivity without the editable things.
> >
> > Also, there is a loadContact method that’s being called while creating the activity and when resuming it since we can choose to edit the contact by pressing the pen on the Toolbar menu (which will call ContactEditActivity by the way).

Now let’s talk about how the database is being handled. There are two classes for this task: MyDatabase and DatabaseHandler.

> MyDatabase is the object that the project will use if it need to access the database. It has the basic methods in a CRUD way such as “add, update and delete” and also two methods to get the database in a different state: openToRead and openToWrite.
>
> In every method listed above, DatabaseHandler is used which is the focus of the next paragraph.

DatabaseHandler is a class that extends from the SQLiteOpenHelper which is basically doing all the work to handle the database. It just need a few configurations to match how we want it to work.

In first place, the onCreate method needs a String that is in fact a SQL query to create the tables we will be needed.

Then, there's the onUpdate method which will just drop the table and create a new one with the most recent content.

So basically, MyDatabase is mainly using the SQLiteOpenHelper methods for updating the database like "add, update or delete".