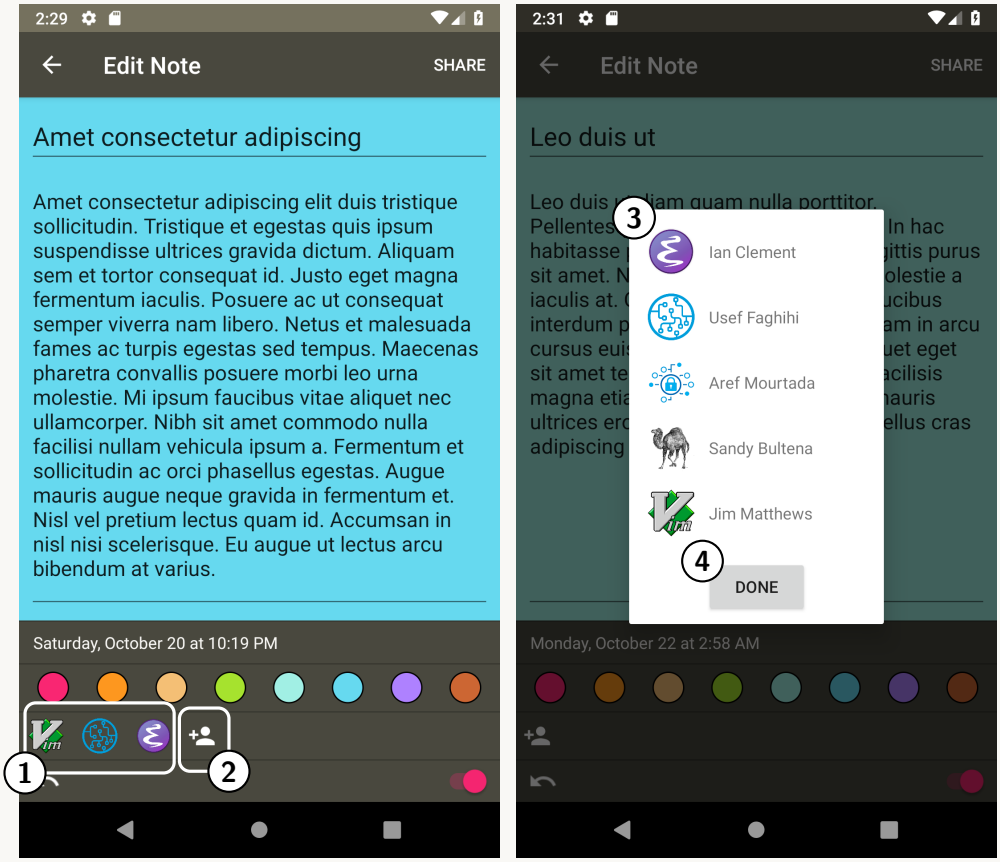
Assignment III

(Due on Check on Léa)

1 Dialog

In `NoteEditFragment` add a list of collaborators and a dialog to add new collaborators.



1. The avatars of the note's collaborators is displayed.
2. Clicking on the "add person" button brings up the dialog to add a collaborator.
3. The add collaborator dialog will display the users who are *not* currently collaborators on the current note. Clicking on the user will add them as a collaborator and remove them from the list.

④ Clicking DONE dismisses the dialog.

1.1 UI

The UI is demonstrated in the video alongside these instructions. Please watch it before continuing on. Below are the requirements and additional resources.

1.2 Database design

Requirements. Add two tables to the sqlite database: "user(s)" and "collaborator(s)". The collaborator implements a many-to-many relationship between "user(s)" and "note(s)". Model the tables based on the provided object-relational model classes `User` and `Collaborator` .

Notes.

- The `User` class contains a bitmap image. You can store this type of data in the sqlite database as a BLOB.
- You can use the `TableFactory` from the previous assignment.

1.3 Display Collaborators

Requirements. The `NoteEditFragment` should now include a list of collaborator avatars by adding a sub-fragment `DisplayUsersFragment` . The list of users to display as collaborators is sent to this fragments using it's `setUsers( .. )` method. At all times, the data displayed here should match the database.

Notes. Use the provided `DisplayUsersFragment` . If you are not using the Assignment #3 starter, use the setup video (https://youtu.be/6oaALQz_Rmc) to add this fragment to your previous assignment solution.

1.4 Add Collaborator Dialog

Requirements. Start from the provided `AddCollaboratorDialogFragment` .

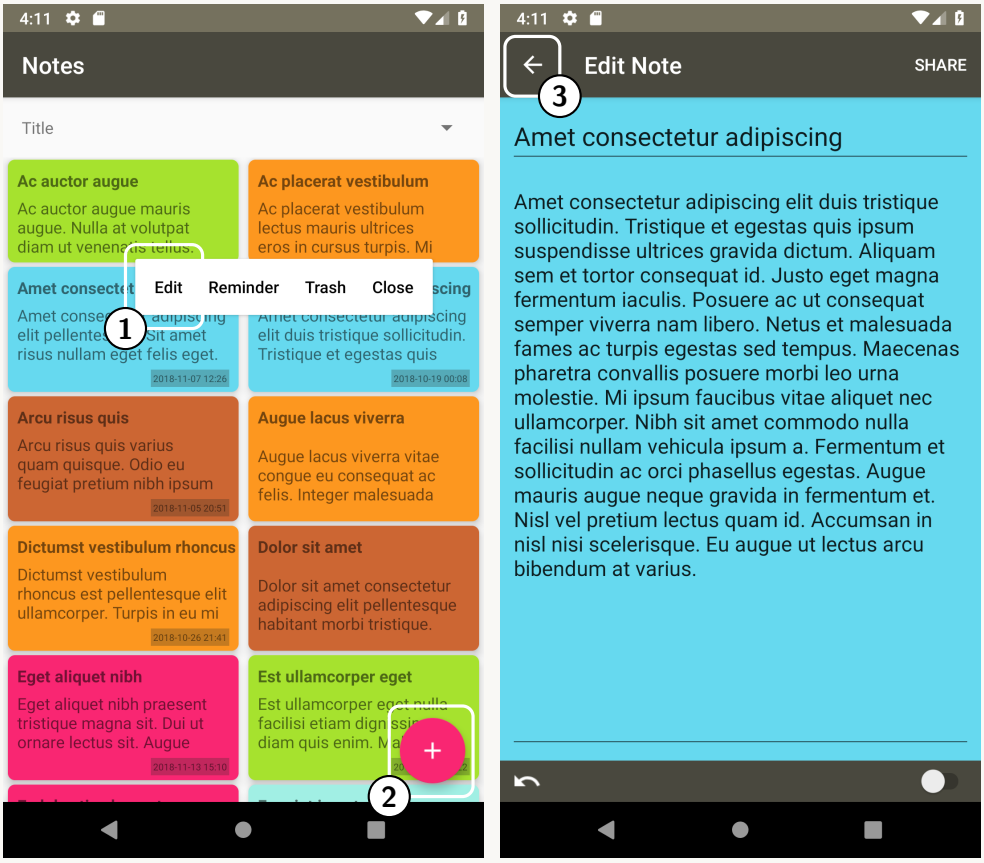
1. When the "add-person" image is clicked, display the dialog.
2. When a user is clicked in the dialog, that user is added to the list of collaborators in `NoteEditFragment` . Use a custom event to accomplish this.

3. When a user is added as a collaborator for a note, the database is updated with this information.

4. When the DONE button is clicked, the dialog is dismissed.

2 Creating and Editing a Note

You will integrate the activities and fragments from the first two assignments into a single functioning app.



The user will start by interacting with the note list activity. Now, they will be able to create and edit their notes.

1. Clicking on the EDIT menu item will allow the user to edit the note.
2. Clicking the `FloatingActionButton` allows the user to create a new note.
3. Clicking on the back arrow brings the user back to the note list and saves the current note.

The UI is demonstrated in the video alongside these instructions. Please watch it before continuing on. Below are the requirements and additional resources.

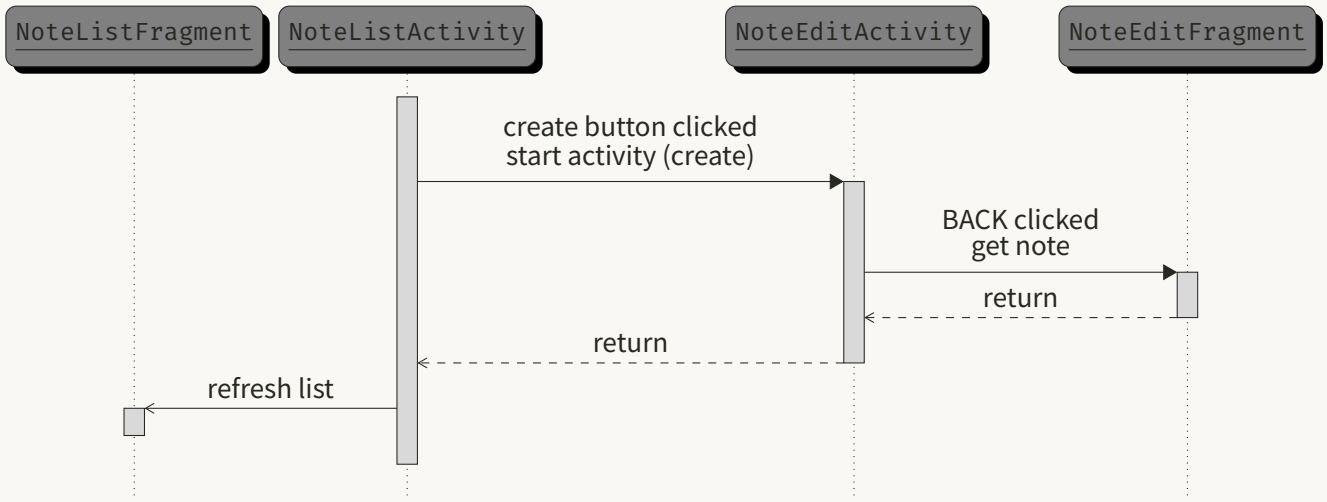
2.1 Creating a note

Requirements. A note is created in the following way:

1. In `NoteListActivity`, the user clicks the `FloatingActionButton` with the + sign.
2. `NoteEditActivity` is launched for the user to create their note. The fields are set to their default (blank) values.
3. The user writes their note, then presses the BACK button.
4. The app returns to `NoteListActivity` and the list is refreshed to include the new note.

The process creates a new `Note` in the database.

Notes. The sequence of actions is:



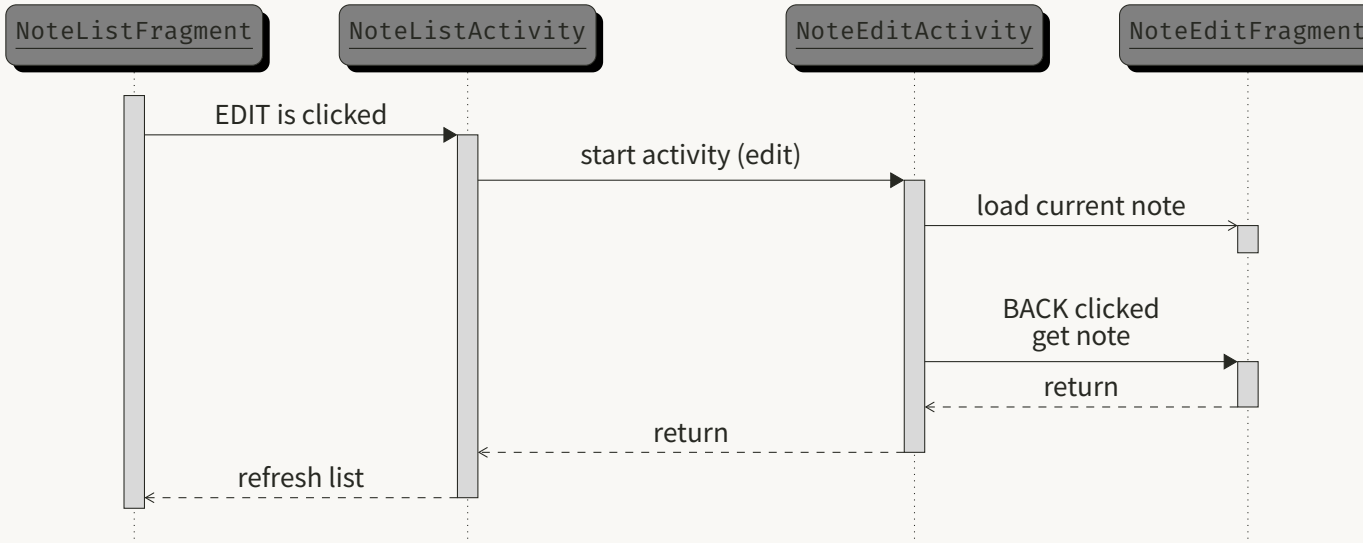
2.2 Editing a Note

Requirements. A note is edited in the following way:

1. In `NoteListActivity`, the user selects an item in the list.
2. `NoteEditActivity` is launched for the user to edit their note. The fields are set to the current values of the note.

3. The user edits the note content, then presses the BACK button.
4. The app returns to `NoteListActivity` and the list is refreshed to include the update to the note.

Notes. The sequence of actions is:

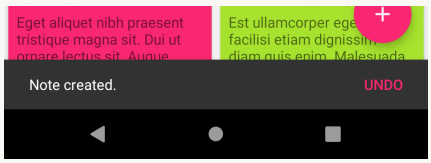


3 Notification and Undo

In `NoteListActivity`, use a `Snackbar` to inform thle user that the note was either created or edited. The snackbar should be created in `NoteListActivity` and connected to the `FloatingActionButton` (see the code sample provided when creating an Android Basic Activity).

3.1 Created Note

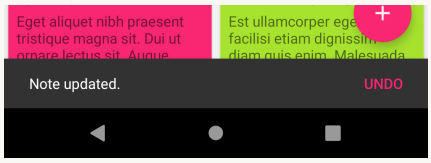
Requirements. After a note is created, show the message



Add an “undo” button to allow the user to discard the new note. This will also modify the database.

3.2 Edited Note

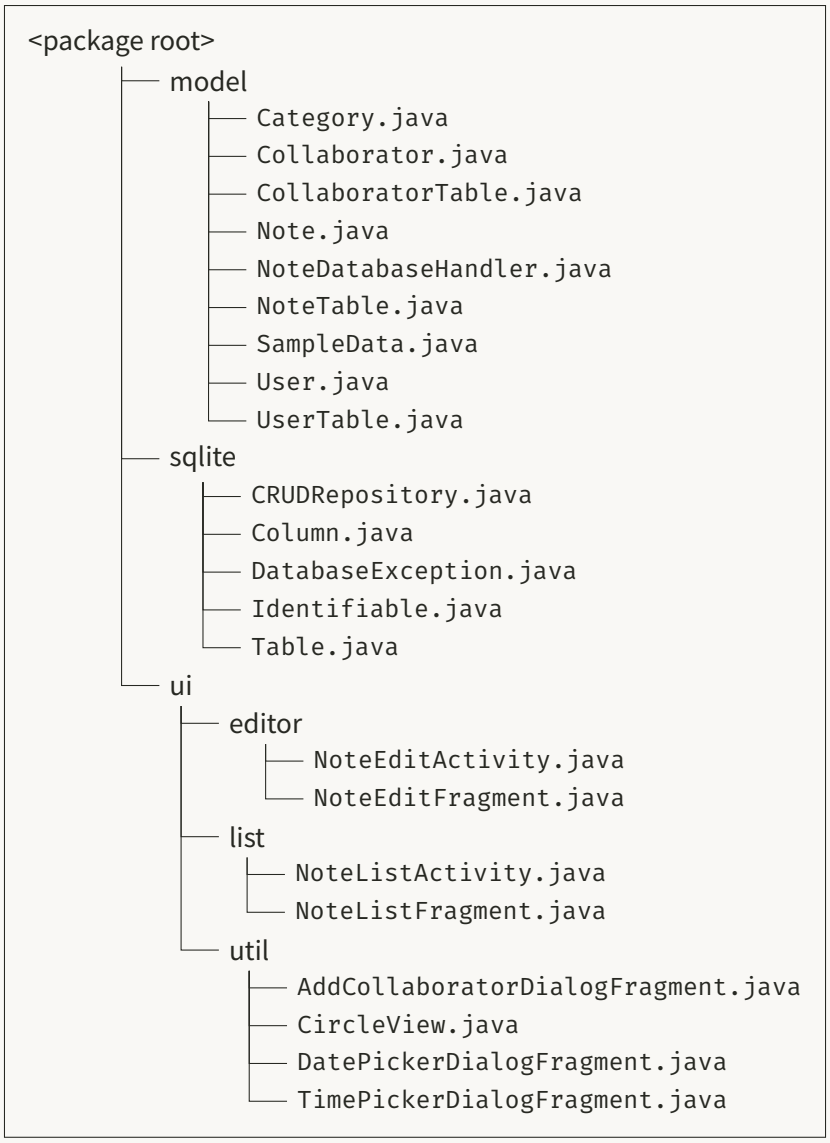
Requirements. After a note is edited, show the message



Add an “undo” button to allow the user to discard their edits, returning the note to the state is was in before launching the edit activity. This will also modify the database.

4 Project Structure

Use the following directory structure for your project’s source code, separating the app into model, ui, activities and utilities:


<package root>
model
Category.java
Collaborator.java
CollaboratorTable.java
Note.java
NoteDatabaseHandler.java
NoteTable.java
SampleData.java
User.java
UserTable.java
sqlite
CRUDRepository.java
Column.java
DatabaseException.java
Identifiable.java
Table.java
ui
editor
NoteEditActivity.java
NoteEditFragment.java
list
NoteListActivity.java
NoteListFragment.java
util
AddCollaboratorDialogFragment.java
CircleView.java
DatePickerDialogFragment.java
TimePickerDialogFragment.java

5 Requirements

- Your program should be clear and well commented. It must follow the "420-616 Style Guidelines" (on Léa).
- Create an Android project with minimum SDK 23 - Android 6.0 (Marshmallow) or later.
- The UI meets all the requirements above.
- All `Intent` objects are only created and used in `NoteListActivity` and `NoteEditActivity`, never in any Fragment.
- Fragments (both dialogs and regular) communicate information with Activities by defining their own events.
- The keys for the Intent's extras are all constants.
- The `Note` is passed as an extra and implements `Parcelable`.
- All notes, users and collaborators are stored and modified in the `sqlite` database using the provided classes.
- You can use the provided Assignment #3 starter.
- Submit using Git by following the instructions (on Léa).