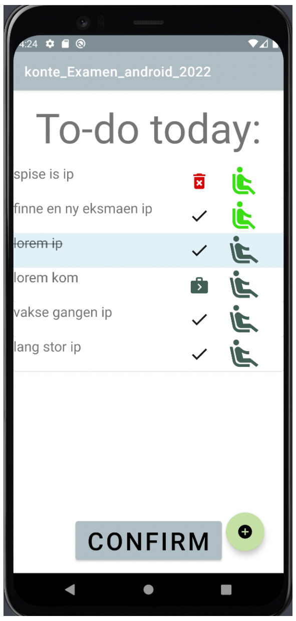
Documentation kNr: 2041

PGR208 1 V21 HEL 2022

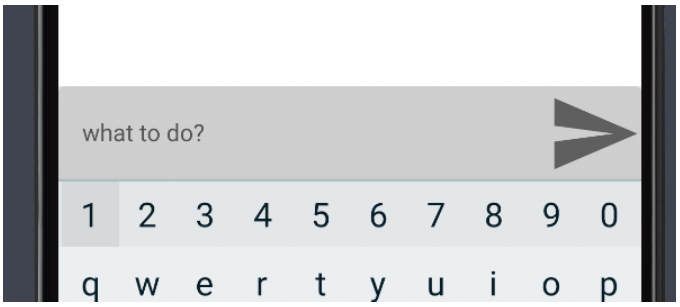
Introduction

The task given we were given for this resit exam, was to make a todo-list with a connection to an external api, whereas the tasks could be categorized into three different categories. With the possibility to move the todos to a new day or delete them.

Overview of the app

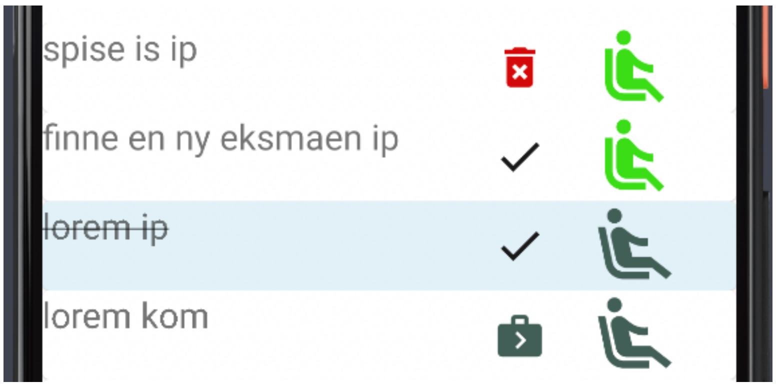


There is two buttons on the bottom of the main activity, where one is to add another ToDo, and the other one is to confirm the list manipulations you've applied above. When you click the hoverbutton, a textview is shown and you can input a new todo and add it. Its both an event listener where enter is pressed or the "endIcon" is clicked to add the toDo.



List manipulation

I was trying to be fancy with the design and implement gestures to toggle the handling of list items. Both selecting them, and moving them. Now you can toggle items as done by clicking the item. The button on the right is a toggle between delete and checkmark. Where theres four different states the items can be in. starting from the top in the embedded screenshot. The trashcan = delete tagged item. The checkmark is a normal item. Blue-background and line trough is a finished todo. While the suitcase symbol represents that the selected item is tagged to move to the next day.



Code Structure

- MainActivity

 I have used the main activity to build my main view. If i had more time, i would have implemented multiple fragments where I could show the toDos for Tomorrow as well. I did not get the time to implement the fragment, but most of the backend functionality is available nevertheless. I probably should have been using intents to pass the data when changing the states of the different toDoItems, but i thought of that in retrospect and did not have the time to make the changes needed.

- ToDoListDatabaseManager

I chose sqlite as a database for saving all the different todos, and to keep data between saves, I have a bit too many database actions for my liking, so i should probably have implemented a saveStoredState with bundles, using onRestoreState and other state-changes to save data in memory to improve the performance.

- DataManager

The Datamanager has two jobs, to create the objects of the todos for today, make them into a list so that the ToDoItemAdapter can inflate the recycleview with views. It also uses the WebRequestHandler to check the api wether the item should be done or not. While writing the documentation, I just realized that the items do not get a new priority when moved to the next day, so therefore the items will stay in the same neutral state forever. This will be implemented in beta 0.2 😉

- ToDoItemAdapter

The recycleviewAdapter has the purpose of creating the views for the recyclerView, and inflating the layouts with the correct data. Recyclerviews only creates as many views as needed to be viewed at the same time, and reuses the same views to show the data the user sees. Therefore we need an adapter to inflate the correct view with the data for the shown item.

- ToDoItem

This is a simple dataClass which helps us with the implementation of the adapter.

- WebRequestHandler

Uses Okhttp3 to send a simple webRequest to the api with the question as a payload. Here I used a when condition to turn the three different “priorities” into Integers to save some storage in the sql database and make other queries and deps easier to write.

- XML
    - **Activity_Main**

Constraintlayout which holds main components.

I have used material components to make sure they are closer to the guidelines given by google for good design language.

It has a textview for the heading, the recycleriew, the confirmbutton and the floating button. It also holds a hidden textinput field, which activates on the click of a floatingactionbutton.

- **to_do_item**

  This is the layout we inflate into the todoItems. It contains a TextView, and two imageViews. Organized neatly in a horizontal linearlayout.

COLORS

For colors I tried to use material guidelines and used the generator at materialtheme.io to generate the shades (https://material.io/resources/color/#!/?view.left=0&view.right=0&primary.color=B0BEC5&secondary.color=C5E1A5)

HTTP-Request

For the http request to the external REST-api, I looked around for one that had sufficient documentation and had an easy and clean way to implement the functions I needed. After consideration I ended up using OkHttp3 since it was light weight and strong enough for this project. I have used a try/ catch to make sure that the JSON payload recieved is handled properly. And if there is an error there the app wont crash.

Conclusion

Overall i’m happy with parts of the app. But the design ended up being too complicated and not intuitive enough. If i were to design it again I would have gone with a more straight forward design, with clear checkmarks and a separate button to move the items to next day. Some of the dataflow in the code is a bit overcomplicated. where a more straight forward approach would have been better for the overall process of designing and especially if the app were to be developed further with more features.