Mobile Application Development 2022-23

A playing court reservation application

Lab3 – Managing user reservations

Learning objectives

- Using the Model/View/ViewModel architecture
- Managing date and time and their representation
- Surviving to the imperfections of real world libraries
- Improve the UX finding out how information is going to be presented and collected

Description

You will extend (or rewrite, at your choice) the app created in the previous lab to allow users managing their own reservations. By using the app, a user will be able to

- browse existing reservations and see their details
- check the availability of a playing court for a given date and time slot
- cancel an existing reservation

The app will operate exclusively on local data (suitably stored in a model of your choice): as such, playground availability, for example, will not depend on others' actions performed dynamically at run time; in spite of this simplification, you will discover that dealing with reservation and properly presenting choices to the user is complex enough, to be worth exploring.

The app will provide several different views:

- One is going to show, in a compact calendar format, the set of existing reservations for the current user
- Another will let the user choose a sport discipline and browse the availability of playgrounds in the coming days
- A third view will allow users to display and/or modify the details of their reservation (like choosing the time slots they are going to reserve, and/or enter some custom request, like renting equipments for the chosen sport)
- A last one will allow the user to confirm the cancellation of an existing reservation

These views will be linked in a suitable way, so that the user can easily navigate according to the use cases listed above.

Please note that, while the first two views may be based on a calendar representation, their semantics is quite different: the former should present in a easily understandable way the time slots that were reserved by the user and possibly the corresponding discipline; the latter should be previously filtered by the choice of a given sport, and present the union of the availability of the corresponding playing courts. Since the calendar view cannot possibly accommodate all the details of the several available time slots, an intermediate representation for the chosen date will be necessary, providing details about the various playing courts managed by the organization.

While Android does provide a CalendarView widget, it is not so flexible when it comes to present customization, so a third party library may be useful. One possible alternative is the CustomCalendarView that is described in the following page: https://stacktips.com/articles/custom-calendar-view-library-in-android.

The project will be kept on GitHub as a private repository.

Steps

1. Create a project and import the library necessary to support the Androidx architecture components and the custom Calendar.
2. Start designing the business logic component, providing methods for retrieving the existing reservations, adding a new reservation (possibly throwing an exception if the requested time slots are not available), canceling an existing reservation.
3. The business logic component will be backed by a local database, keeping the application state: create the corresponding classes and implement their behavior
4. Per each requested screen, create a corresponding view and its viewModel. The viewModel will operate on the business logic component and will make available a set of LiveData objects that can be observed by the view
5. Link the various views according to the prescribed use cases and common sense logic.

Submission rules

- Lab must be submitted by the forthcoming fourth lab
- The functionalities implemented in your code as well as the design of the user interface will be evaluated
- Before submitting, clean the project using *Build -> Clean Project*
- Create a zip file with your project and name it gXX_lab3.zip
- Upload it on the Polito web portal (only one student of the group must upload it); for multiple uploads, only the most recent file will be evaluated