UI design for mobile systems

by Christoffer Lundström

July 7, 2019

Development of Mobile Applications - 5DV209
Examinator: Johan Eliasson

UMEÅ UNIVERSITY

Contents

1 Introduction **2**
1.1 Purpose 2

2 Procedure & materials **3**

3 Results **4**
3.1 Requirements Analysis 4
3.1.1 Functional 4
3.1.2 Non functional 5
3.2 Design 6
3.2.1 Overview 6
3.2.2 Calendar 7
3.2.3 Event details 8
3.2.4 Add event dialog 9
3.2.5 iOS adaptation 9

4 Discussion **10**

5 References **11**

1 Introduction

This report will cover the structural design of a prototype port for Windows Calendar on PC to Android by abstracting and analysing the UI elements and functionalities of the application and converting these to appropriate entities according to the *MVC-pattern*. It will cover basic requirements of the application such as placement and design of Views, description of controllers and models.

The prototype will follow UI guidelines of Androids *Material design* and general design- principles. Furthermore a comparison will be made between *Material Design* and *iOS human interface guidelines* to show how an adaptation of the prototype could be produced for iOS.

1.1 Purpose

By examining an existing application and applying design principles for Android and iOS we will gain a practical understanding of user interface design for mobile applications, which in extension is a requirement for the course.

Another purpose of this assignment is to prioritize functionalities and reflect upon the importance and usefulness of each interaction as there generally is a limited amount of screen space in a mobile environment.

2 Procedure & materials

Windows 10 PRO and the embedded Windows Calendar(fig 1) version 16005 is used in this experiment.
The main method of analysing the Calendar application in this assignment is by visually and practically examining its use cases and determining the most suitable view or controller in an Android setting.

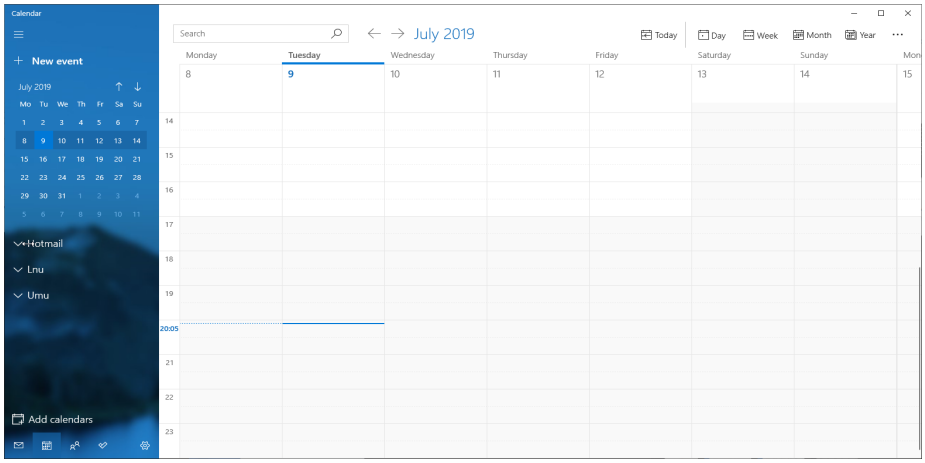

Figure 1: Week and month UI of Windows calendar.

By breaking down each component and use case the conversion to Android will be significantly easier to comprehend.

Android Studio v.3.4.1 by JetBrains is used to produce some of the interface designs.
Photoshop CC 2017 is used for an overview design and initial design template.

3 Results

3.1 Requirements Analysis

3.1.1 Functional

This section will break down individual buttons and components to a few of the most important Use Cases required by the prototype.

Use-Case - View events

Pre-condition: None.
Post-condition: None.

Main Scenario:
1. User wants to view events on a particular date.
2. User selects a date.
3. User is presented with a view of detailed information of events.

Use-Case - Add events

Pre-condition: None.
Post-condition: User is returned to Calendar-View.

Main Scenario:
1. User wants to add an event.
2. User presses Add button.
3. User is presented with an empty dialog form.
4. User enters Name, Location, Date, Time and Description of event.
5. User saves event.

Alternative Scenario:
5. User presses Back or Cancel.

Use-Case - Modify event

Pre-condition: User has selected a date with existing events.
Post-condition: User is returned to Calendar-view.

Main Scenario:
1. User wants to modify event.
2. User presses event in detail view.
3. Dialog view is presented with details of event.
4. User modifies details.
5. User presses save.

Alternative Scenario:
5. User presses back or cancel.

Use-Case - Delete event

Pre-condition: User has selected a date with existing events.
Post-condition: None.

Main Scenario:
1. User wants to delete event.
2. User right-swipes or long-presses event item.
3. User is presented with confirmation dialog.
4. User confirms action.

Alternative Scenario:
4. User aborts confirmation.

The windows calendar consists of four overviews. Year, month, week and day. The yearly overview will be intentionally left out of the prototype due to the fact that display of dates would have to be less than 10sp which is non compliant to material design and unviable for a small screen(*Material design: Typography*).

3.1.2 Non functional

This section will cover requirements related to performance, accessibility, modifiability and scalability.

To make sure the Calendar runs smoothly one important aspect of the design is to load events dynamically based on the date. This means not loading the entire event-database into memory but rather run queries against a database

or local file. It wouldn't make sense to load passed events unless specifically asked for. A better approach would be to load the currently viewed month's events. This will not only make the application faster, but also make it more maintainable and scaleable.

Another requirement of our prototype is to keep the user interface clear, robust and specific. Meaning having a clear UI with sufficient contrast ratios and structure to nudge the user through the application. It should be clear what is asked from the user (*Material design: Accessibility*).

Modifiability of the calendar will be handled programmatically by keeping elements subdivided into fragments and adhering to an *MVC* pattern. This will ensure loose coupling of the application.

3.2 Design

3.2.1 Overview

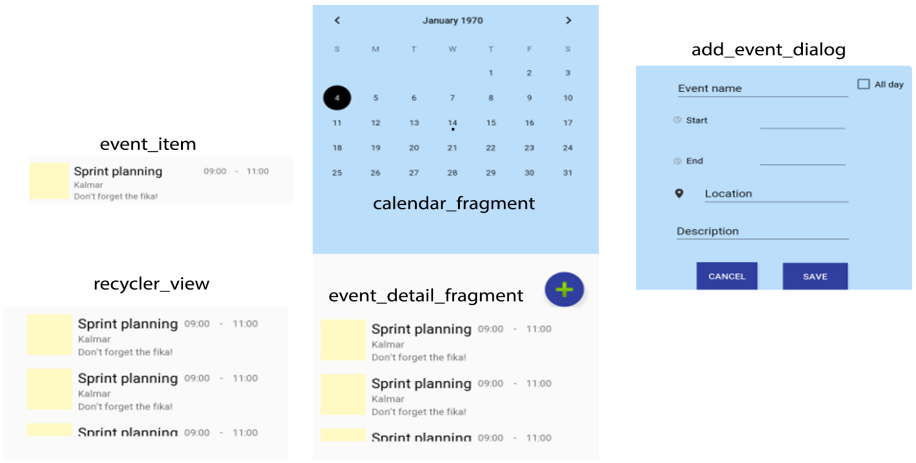

Figure 2: The different parts that make up the main calendar view.

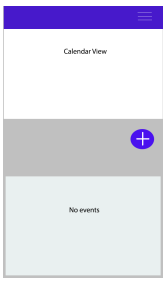

Figure 3: Initial design template.

3.2.2 Calendar

The main activity will be empty except for a single fragment container. It will be the main controller and hold a fragment-manager which is responsible for initializing the first calendar fragment.

The calendar
fragment will be responsible for date-selection and return a Date object to the main activity which in turn creates the event detail fragment and passes the data via a bundle. In general all data-transactions between fragments are done via the hosting activity (*Android API: Fragments*).

The calendar is placed at the top
of our focus hierarchy with a high contrast material background as this should be the first point of focus for the user (*Material Design: Color and Contrast*).

Following the calendar a green *Add event* button is placed within an easily reached region of the screen, both for one-hand and two-hand users. The button has a touch area of 48dp and is coloured in a stark contrast with a slight shadow and elevation to emphasize its presence. Furthermore the button is placed just above the event detail area to make the purpose clear (*Material design: Hierarchy and Placement*).

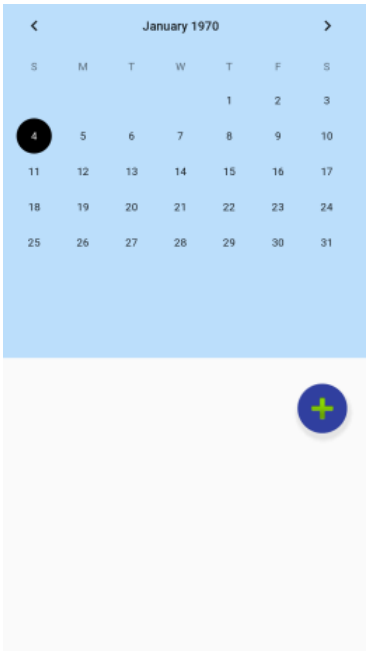

Figure 4: The first fragment includes the calendar and a button.

Figure 5: Landscape view.

3.2.3 Event details

The next part of our prototype
is the event detail fragment which will provide
information regarding the specified time, location
and description of the events on the selected date.

Each event is marked with a colour to clearly
separate its regions (*Material design: Text fields*).
The fragment fills the rest of the screen
with a scrollable
list of event items. This fragment will handle
an internal recycler view responsible for rendering
the event item model data. Before rendering a
view the fragment controller should retrieve relevant
data by querying a local or external database with
the date passed to the fragment with the bundle.

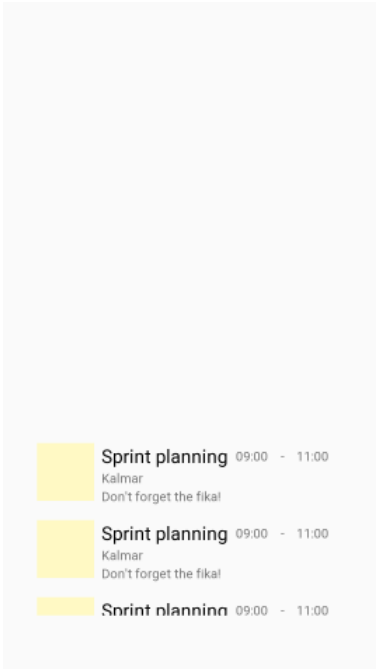

Figure 6: Event detail fragment with a Recycler view.

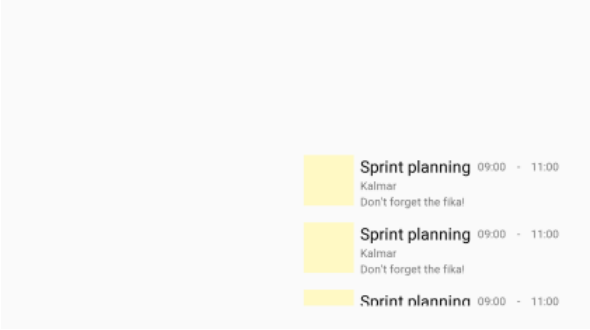

Figure 7: Recycler landscape view.

3.2.4 Add event dialog

This dialog is controlled by the Calendar fragment. By adding a click-listener to the *add event* button a new instance of the dialog will be created by the main activity fragment-manager.

The fields of this view are based on a custom event item model which is later used for storing events. It is important that this model implements a way to serialize or parse data between fragments, servers and storage locations (*Android API: Parcelable*).

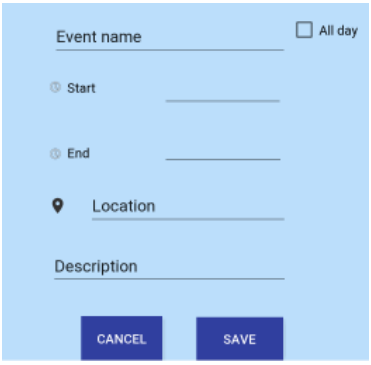

Figure 8: The add event view.

Design of this dialog is simplistic but expresses its intent clearly with icons recommended in material design (*Material design: System Icons*). It also follows the blue material theme as its parent fragment (*Material design: Hierarchy and Placement*).

3.2.5 iOS adaptation

To adapt our prototype to iOS there are some changes and additions which could be made. First of all is to have a clear hierarchical navigation. The current prototype supports this as it basically contains two different views; calendar and the event view modal/dialog. Our dialog is dismissed by navigating using back. Another way to dismiss the dialog could be to use a swiping motion (*Human Interface Guidelines: Modality*). Furthermore the dialog should be converted to a *popover*. This makes the focus point clear as the background is dimmed (*Human Interface Guidelines: Popovers*).

When a new event has been added or an existing has been modified a feedback notification could be added at the bottom of the screen similar to a *toast* in Android(*Human Interface Guidelines: Feedback*).

Animation is also a big part of interfacing with a user and thus a slider which fills the event detail row could be added for additional clarity when pressing down on a event. This could be an indication for modifying or deleting the event (*Human Interface Guidelines: Animation*).

Additionally a two-color system for the current theme could be added to represent a dark and light mode (*Human Interface Guidelines: Color*).

Like *material design* the *Human Interface Guidelines: Materials* uses elevation of elements to separate content from backgrounds or foregrounds. This is already present in our prototype and doesn't need further adaptation.

As user feedback and immediate response is an important aspect of HIG a loading bar or indicator could be added when the event detail fragment queries the database for event content asynchronously (*Human Interface Guidelines: Progress Indicators*).

4 Discussion

Material design and iOS human interface are two very similar principles of design which overlap in many areas such as content elevation, touch controls and especially navigation. Other differences are more subtle such as dimming of foreground with popovers or navigating with tabs. Because both of these guidelines have a friendly user-experience as the end goal and follow similar design patterns prototyping is easier, especially for tiny projects like this one. I suspect that the differences become more apparent and substantial as projects scale in size, perhaps there are more limiting factors to UI designs from a technical perspective when comparing activities, fragments etc. to the iOS framework.

This design may seem trivial compared to the fully fledged Windows calendar but the key functionalities of the calender are covered and designed with the mobile user in mind. As mentioned in *Seven deadly myths of Mobile* by Josh Clark:

> "The point is not to arbitrarily strip out features and content, but a matter more of organizing and prioritizing."

To summarize I think that this project has been quite insightful as to current methods of UI development and I look forward to pursuing these schools of design further.

5 References

Online sources

Android. *Material design: Accessibility*. URL: https://material.io/design/usability/accessibility.html. (accessed: 20.07.2019).

— *Material Design: Color and Contrast*. URL: https://material.io/design/components/buttons.html#hierarchy-placement. (accessed: 20.07.2019).

— *Material design: Hierarchy and Placement*. URL: https://material.io/design/components/buttons.html#hierarchy-placement. (accessed: 20.07.2019).

— *Material design: System Icons*. URL: https://material.io/design/iconography/system-icons.html#. (accessed: 20.07.2019).

— *Material design: Text fields*. URL: https://material.io/design/components/text-fields.html#usage. (accessed: 20.07.2019).

— *Material design: Typography*. URL: https://material.io/design/typography/the-type-system.html. (accessed: 20.07.2019).

Apple. *Human Interface Guidelines: Animation*. URL: https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/animation/. (accessed: 20.07.2019).

— *Human Interface Guidelines: Color*. URL: https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/color/. (accessed: 20.07.2019).

— *Human Interface Guidelines: Feedback*. URL: https://developer.apple.com/design/human-interface-guidelines/ios/user-interaction/feedback/. (accessed: 20.07.2019).

— *Human Interface Guidelines: Materials*. URL: https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/materials/. (accessed: 20.07.2019).

— *Human Interface Guidelines: Modality*. URL: https://developer.apple.com/design/human-interface-guidelines/ios/app-architecture/modality/. (accessed: 20.07.2019).

— *Human Interface Guidelines: Popovers*. URL: https://developer.apple.com/design/human-interface-guidelines/ios/views/popovers/. (accessed: 20.07.2019).

— *Human Interface Guidelines: Progress Indicators*. URL: https://developer.apple.com/design/human-interface-

guidelines/ios/controls/progress-indicators/. (accessed: 20.07.2019).
Clark, Josh. *Seven deadly myths of Mobile*. URL: https://www.youtube.com/watch?v=_F85eutt82Y. (accessed: 20.07.2019).
Google. *Android API: Fragments*. URL: https://developer.android.com/guide/components/fragments. (accessed: 20.07.2019).
— *Android API: Parcelable*. URL: https://developer.android.com/reference/android/os/Parcelable. (accessed: 20.07.2019).