DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Klinisches Anwendungsprojekt

Akne Informationsanwendung

Author: Robert Schauer
Advisor: Linda Tizek, PhD
Submission Date: 01.04.2021

Ich versichere, dass ich dieses Klinsche Anwendungsprojekt und diese Dokumentation selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, 01.04.2021 Robert Schauer

Inhaltsverzeichnis

1 Ziel des Projekts und Motivation

Das Ziel des Projekts war es eine Applikation für Arztpraxen und Patienten zu entwickeln, welche die Risiken und Nebenwirkungen des Akne Medikament Roaccutane darstellt und es dem Patienten erlaubt sich über die Risiken des Medikament vor oder nach der Besprechung mit dem behandelnden Arzt zu informieren. Dieses Ziel war motiviert dadurch, dass Roaccutane ein teratogener (reproduktionstoxischer) Stoff ist welcher sowohl für den Patienten selbst, aber auch bei Embryos die mit dem Medikament indirekt in Berührung kommen, schwerwiegende Nebenwirkungen haben kann. Personen die mit Roaccutane behandelt werden dürfen nicht nur bis ein Monat nach der Behandlung nicht schwanger werden, sondern auch kein Blut spenden oder Kinder stillen um zu vermeiden dass ein Embryo oder neugeborenes Kind Kontakt mit dem Stoff hat. Weil die Behandlung mit diesem Medikament außerdem über einen längeren Zeitraum und mehrfach stattfinden kann ist es umso wichtiger dass Patienten die wichtigsten Informationen präsent haben.

1.1 Aufgabenstellung

Es soll eine Applikation zu entwickelt werden, welche den Inhalt der Informationsbroschüre für Patienten [1] und der Dokumentationsbroschüre für Ärzte [2] enthält und darstellt. An dieser Stelle sei erwähnt, dass durch die COVID Situation und die dadurch verminderte Kommunikation zwischen mir und meiner Projekt Leitung hier an mir verloren ging, dass die Applikation durch eine Chat-bot Funktion gesteuert werden sollte. Als uns diese Fehlkommunikation auffiel steckte allerdings schon zu viel Zeit in dem Projekt und wir haben uns entschieden mit einer normalen Applikation Navigation über Links fortzufahren.
Der Patient sollte sich durch die Seiten navigieren können und Teile überspringen, oder zu einer gewünschten Stelle springen können. Ich sollte außerdem eine Checkliste einbauen welche erneut die wichtigsten Informationen präsentiert und dem Patienten erlaubt diese abzuhaken, wenn er den Punkt verstanden hat und offen zu lassen wenn er an einer Stelle noch Fragen hat. Offene Fragen müssen dann mit dem Arzt geklärt werden. Um sich diese eventuell offen gelassenen Fragen merken zu können sollte es außerdem einen Weg geben wie der Patient den Zustand der Checkliste extern speichern kann. Es war dabei aber wichtig dass nichts lokal gespeichert wird, um sicher zu stellen dass die Applikation keine sensiblen Informationen speichert.

2 Dokumentation des Projektablauf

18.11.2020	von - bis	10.12.2020	von - bis
Projektbesprechung	Recherche Technologien	Beginn Ausarbeitung	Implementation Navigation und Struktur
19.01.2021	29.01.2021	11.02.2021	16.03.2021
Präsentation zum Stand der Entwicklung	Besprechung unseres Missverständnis	Letzte Besprechung vor Projektende	Abschlusspräsentation beim Promotionskolleg

Tabelle 2.1: Projekt Timeline

2.1 Projektablauf

Die Bearbeitung des Projekts erfolgte unter Betreuung von Linda Tizek, Phd. Besprechungen und Organisation konnten durch die COVID Krise ausschließlich über Zoom und per Email stattfinden. Die Timeline des Projekts ist in Tabelle 3.1 abgebildet. Die geplante Bearbeitungszeit des Projekts ging vom 01.12.2020 bis zum 01.03.2021 und wurde so eingehalten.
Nach der initialen Besprechung in der wir die Ziele festgelegt hatten, habe ich mich darauf fokussiert mir zu überlegen, in welchem Format ich die das Projekt umsetzen möchte um diese Ziele zu erfüllen. Ich hatte die Wahl eine Website oder eine Applikation zu bauen und habe mich für eine Applikation entschieden. Ich habe mich in dieser Zeit außerdem in die verschiedenen Möglichkeiten und Programmiersprachen eingelesen und habe mich für die Programmiersprache Dart [3] und das Flutter UI Toolkit [4] von Google entschieden.
Am 10.12.2020 begann die Ausarbeitung des Programms. Zuerst war festzulegen welche Design Pattern dem Projekt zu Grunde liegen soll. Auch hier habe ich mich für ein Google Produkt in der BloC Pattern [5] entschieden.
In den Winter Semesterferien habe ich dann begonnen die Navigations Elemente der Benutzeroberfläche zu implementieren und ein allgemeines Konzept für den Aufbau des Inhalts der Applikation zu entwerfen. Im Zuge dessen habe ich für die Sidebar die in der Applikation zur Navigation genutzt wird ein existierendes Design Beispiel [6] umgebaut und nutzbar gemacht. Ich habe außerdem die Navigation mit dem flutter-bloc [7] Paket eingebaut. Bis zu Ende der Ferien war ich soweit meinen Betreuern ein erstes mal die Applikation mit einigen Seiten Inhalt zeigen zu können.
Es war eigentlich geplant schon am 12.01.2021 ein Gespräch zu haben, dieses musste allerdings wegen Krankheit eines Betreuers auf den 19.01.2021 verschoben werden. In dem Gespräch haben wir als nächstes Ziel die Implementation der Checkliste festgelegt. In den

folgenden 10 Tagen habe ich die Checkliste ausgearbeitet. Während ich daran gearbeitet habe sind ein paar Fragen aufgetreten wegen denen wir dann im Mail Kontakt eine Diskrepanz zwischen dem gewünschten Ergebnis und meiner Ausarbeitung gefunden haben. Es hat sich herausgestellt dass es bei der Zielsetzung eine Fehlkommunikation gab die meinen Betreuern zu diesem Zeitpunkt klar geworden ist. Gewünscht war eine Navigation der Informationen in der Applikation via Chatbot, das ist aber bei mir so nicht angekommen und deshalb auch nicht umgesetzt worden. Gemeinsam haben meine Betreuer und ich uns dann entschieden, dass es zu diesem Zeitpunkt zu spät ist die Applikation neu zu bauen und das ich stattdessen fertigstelle was ich bereits erarbeitet hatte.
Bis zum 11.02.2021 hatte ich dann die gewünschten Ziele soweit fertig dass ich meine Version meinen Betreuern erneut präsentieren konnte. Diese hatten sich hier noch einige Nachbesserungen zur Checkliste gewünscht. Wir haben uns entschieden noch eine Einführungsseite die denn Sinn der Checkliste und eine Schlussseite welche den Wiederherstellungs Code erklärt einzubauen. Das habe ich dann bis zur Abschlusspräsentation am 16.03.2021 noch umgesetzt, zusammen mit einigen Verbesserungen zur Code Qualität und generellen Lesbarkeit des Code.

3 Umsetzung und Code Dokumentation

3.1 Dart & Flutter

Zum Bau dieser Applikation wurden die Programmiersprache Dart[3] und das UI-Toolkit Flutter[4] verwendet. Der einfachste Weg diese Werkzeuge zu nutzen ist sie in Android Studio [8] als Plug-ins zu installieren. Man kann sie aber auch separat installieren und mit jedem beliebigen Texteditor nutzen.
Flutter Applikationen laufen auf Android, iOS, Linux, Mac, Windows, Google Fuchsia, und im web mit nur einer Code-Basis, sie sind also so Plattform unabhängig wie möglich. Neben einer großen Entwicklergemeinschaft die Pakete und Tutorials entwickelt, arbeitet Google außerdem weiterhin an Flutter und hat erst am 03.03.2021 Flutter 2 veröffentlicht, was viele Neuerungen gebracht hat.
Dart, die Programmiersprache in der Flutter Applikationen geschrieben werden ist außerdem der Sprache Java sehr ähnlich, was vielen Programmierern einen einfachen Einstieg ermöglicht.

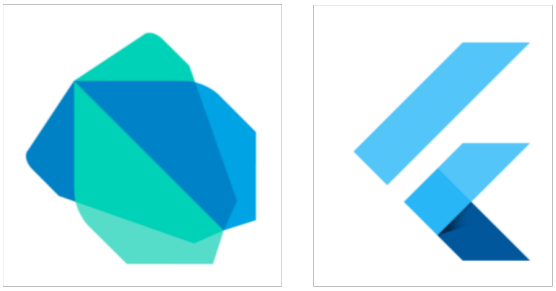

Abbildung 3.1: Dart & Flutter Logos

3.1.1 Widgets

Im folgenden werde ich immer wieder das Wort 'Widget' verwenden, deshalb möchte ich hier erklären was damit gemeint ist. Generell ist ein Widget ein Element der Benutzeroberfläche, vergleichbar mit einer Android View oder einer iOS UIView. Aber anders als in diesen Beispielen ist in Flutter alles ein Widget, sogar die App selbst. Ein Widget könnte ein Knopf, ein Text oder ein Bild sein, aber auch ein Element dass das Layout einer Seite definiert ist ein Widget und auch Elemente welche die User Interaktion steuern sind Widgets. Es gibt zwei Sorten von Widgets:

- Stateless Widgets
- Stateful Widgets

StatelessWidgets sind wie der Name vermuten lässt statisch. Das heißt sie haben nur einen Zustand der sich nicht verändern kann.
StatefulWidgets auf der anderen Seite sind dynamisch. Sie können veränderungen und inputs von der Benutzeroberfläche entgegen nehmen und darauf reagieren.
Alle Widgets sind in einer Baumstruktur dem *WidgetTree* angeordnet, in dem ein Eltern Widget beliebig beliebig viele Kinder Widgets haben kann. [9]

3.2 Architektur

```
/
├── assets ..................................................[Benötigte Ressourcen]
│   ├── fonts
│   └── images
├── lib .....................................................................[Code-Basis]
│   ├── logic
│   ├── presentation
│   │   ├── dropdown
│   │   ├── sidebar
│   │   ├── router
│   │   └── screens
│   │       └── widgets
│   ├── main.dart
│   └── checklist ..........................[Separate Code-Basis der Checkliste]
│       ├── logic
│       ├── presentation
│       │   ├── router
│       │   └── screens
│       └── checklistmain.dart
└── pubspec.yaml ....................................[Dependency Konfiguration]
```

Tabelle 3.1: Projekt Dateisystem

Der Code für die Anwendung ist auf Github zu finden in dem Repository mit dem Arbeitstitel "Prospect"[10]. Die Relevanten Ordner im Repository sind wie in Tabelle 3.1 strukturiert.

3.2.1 Assets

Assets oder auf deutsch Ressourcen ist der Ordner der alle Dateien enthält, die keinen Code enthalten, aber beim Bau der Applikation benötigt werden. Er ist unterteilt in Bild Ressourcen im 'images' Ordner und Schriftarten im 'fonts' Ordner.
Um die Ressourcen verwenden zu können müssen sie in der *pubspec.yaml* Datei deklariert werden.

3.2.2 pubspec.yaml

Die *pubspec.yaml* Datei ist eine der wichtigsten Dateien einer Flutter Anwendung. In dieser Datei müssen alle notwendigen Dependencies importiert und deklariert werden, bevor sie im restlichen Code nutzbar sind. Diese Datei, anders als der restliche Dart Code aber ähnlich wie die Programmiersprache Python, ist Einrückungs-sensitiv. Beispiele für korrekte Syntax sind in der Datei kommentiert. Neben den assets die in dieser Datei deklariert werden importiert die Applikation außerdem folgende Pakete:

- rxdart[11]: Stream Funktionalitäten
- flutter_bloc[7]: Integriert Blocs und Cubits in Flutter
- url_launcher[12]: Erlaubt das öffnen von Links im Browser

3.2.3 Lib

'lib' kurz für die Bibliothek enthält den gesamten Programm Code der die Applikation ausmacht. Weil sich im Zuge dieses Projektes herausgestellt hat dass vermutlich ein großer Teil dieser Applikation überarbeitet oder neu konzipiert werden muss wurde an dieser Stelle der Code für die Checkliste vom restlichen Code unabhängig aufgebaut. So kann die Checkliste praktisch alleine funktionieren oder von einer anderen Applikation mit der gewünschten Chatbot Navigation angesteuert werden. Die Checkliste folgt aber der allgemeinen Unterteilung der restlichen Code Basis. Die Dateien sind unterteilt in:

- logic
- presentation
  - router
  - screens

'logic' enthält alle Klassen mit einer Programm Funktionalität während 'presentation' alle Klassen enthält welche die Benutzeroberflächenelemente enthalten. In presentation findet man die Unterordner 'router' welcher den Navigationsbloc enthält und 'screens' welcher die

Seiten beziehungsweise die unterste Ebene der Benutzeroberfläche enthält.
Die Checkliste ist als eigenständiges Programm aufgebaut wird in der Applikation aber wie jeder andere Inhalt ebenfalls auf der untersten Ebene angezeigt, wie in der Abbildung 3.1 zu erkennen ist.
Im presentation folder der Applikation befinden sich außerdem auch die anderen beiden Ebenen der Benutzeroberfläche in 'sidebar' und 'dropdown' welche die gleichnamigen Widgets enthalten.

Abbildung 3.2: Rechts: Ebenen der Benutzeroberfläche markiert.
Blau: Unterste, Rot: Mittlere, Gelb: Oberste

3.2.4 main.dart

Die *main.dart* Datei enthält die main Funktion die beim Start der Applikation aufgerufen wird und nichts anderes tut als das Widget *MyApp* aufzurufen, welches sich ebenfalls in dieser Datei befindet.
MyApp ist dann dafür zuständig ein 'Theme' für die Applikation festzulegen welches Schriftarten und Farben die in der Benutzeroberfläche genutzt werden zentral definiert. Das erlaubt es diese Elemente zentral für die Applikation zu verändern ohne durch den gesamten Code gehen zu müssen.
MyApp definiert außerdem die Sidebar also das Element auf der mittleren Stufe der Benutzeroberfläche als 'home', das heißt dieses wird zuerst aufgebaut und erfragt dann vom Navigationsbloc die aktuelle Seite um diese anzuzeigen.
Die Datei *checklistmain.dart* erfüllt die gleichen Aufgaben nur für die Checkliste. Das Theme

in der Checkliste ist zum größten Teil das gleiche wie in der *main.dart* es gab nur eine kleine Anpassung der Schriftgröße.

3.3 Code Generierung

Aus rechtlichen Gründen speichert diese Applikation nichts permanent. Weil es aber wichtig ist das ein Patient über offene Fragen in der Checkliste mit dem behandelnden Arzt reden kann, musste ein Weg geschaffen werden den Zustand der Checkliste wiederherzustellen, ohne etwas zu speichern. Zu diesem Zweck hat die Checkliste eine bijektive Wiederherstellungscode Funktion die aus der Liste der abgehakten Fragen einen Code generiert den der Nutzer am Anfang der Checkliste eingeben kann um den alten Zustand wiederherzustellen. Das Prinzip des Code beruht auf der Übersetzung von binär Zahlen in hexadezimal Zahlen und wird von der Klasse checklist_state.dart im logic folder der Checkliste übernommen.
Betrachtet man die Liste von boolean Werten welche die Checkliste ausmacht als eine binär Zahl in der ein 'true' wert eine '1' und ein 'false' wert eine '0' ist, kann man diese Zahl wie in Abbildung 3.3 gezeigt schnell in eine dezimal Zahl und dann in eine Hexadezimal Zahl umwandeln.

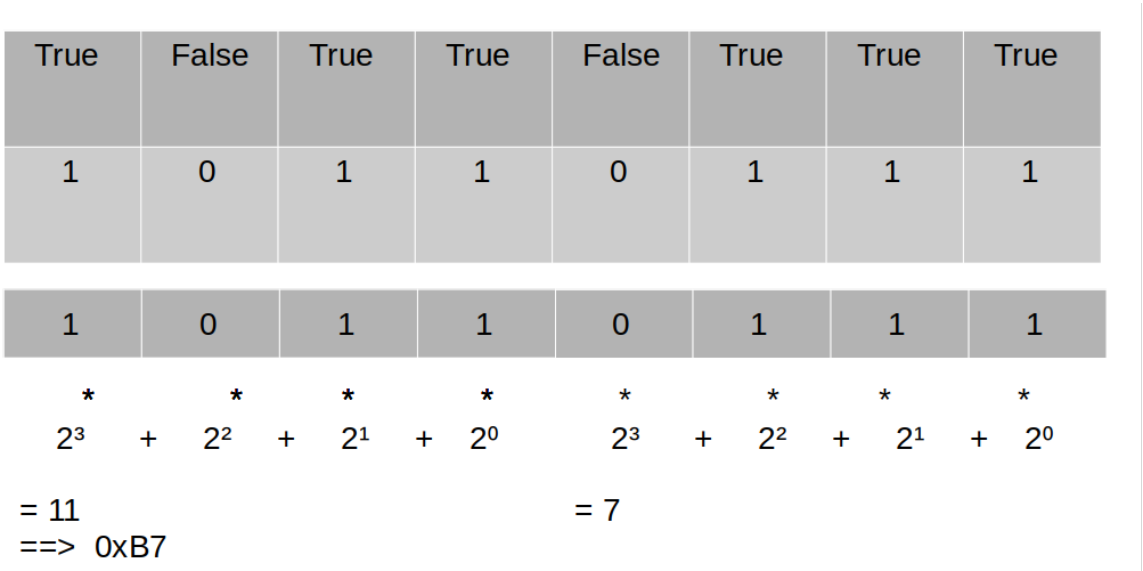

Abbildung 3.3: Beispiel Code Generierung

Um in die umgekehrte Richtung aus einem eingegebenen String (dem Code) wieder eine Liste aus boolean Werten zu generieren muss man den String erst in eine dezimal Zahl auslesen. Dafür gibt es in Dart die Bibliotheksfunktion $int.tryparse(String, Radix)$ in der der String Parameter der auszulesende String ist und der Radix Parameter das zu verwendende Stellenwertsystem. Man geht dann so vor dass man die Zahl solange mit Modulo 2 teil bis man 0 erreicht und für jeden Rest '1' fügt man ein 'true' und jeden Rest '0' ein 'false' in die Liste ein.

4 Fazit & Ausblick

Dieses Praktikum war insofern erfolgreich, dass diese Ziele die wir gut kommuniziert haben erreicht wurden und die anderen einen Lerneffekt hatten. Mir persönlich hat es gezeigt wie wichtig gute Kommunikation ist und wie wichtig es ist ausgemachte Ziele schriftlich festzuhalten.
Die Applikation funktioniert, ist aber ausbaufähig. Vor allem in Sachen Design lässt sich hier noch einiges schöner gestalten. Auch in Sachen Konzipierung gilt es einen Weg zu finden den Inhalt der Broschüren noch effektiver darzustellen um große Blöcke von Text zu vermeiden. Die verwendeten Werkzeuge Flutter und Dart scheinen aber die wohl besten Entscheidungen in diesem Projekt gewesen zu sein, man kann ein wachsendes Interesse in diesen Technologien beobachten, was für die Ausbaufähigkeit der Applikation hoffen lässt.

Literatur

[1] *Informationsbroschüre für Patienten behandelt mit Roaccutane*. 2003. URL: https://www.afmps.be/sites/default/files/downloads/Roaccutane%5C%20patient%5C%20DE%5C%20information.pdf.

[2] *Dokumentationsbroschüre mit Checkliste für die Verordnung von Isotretinoin*. URL: http://www.bfarm.de/SharedDocs/Downloads/DE/Arzneimittel/Pharmakovigilanz/Service/aktuelles/isotret/documbrosch-isotretinoin.pdf?__blob=publicationFile&v=2.

[3] *Dart*. URL: https://dart.dev/.

[4] *Flutter*. URL: https://flutter.dev/.

[5] *BloC Pattern*. URL: https://bloclibrary.dev/#/.

[6] P. S. "TechieBlossom". *sidebar-animation-flutter*. URL: https://github.com/TechieBlossom/sidebar_animation_flutter.

[7] *flutter_bloc*. URL: https://pub.dev/documentation/flutter_bloc/latest/.

[8] *Android Studio*. URL: https://developer.android.com/studio.

[9] *Introduction to Widgets*. URL: https://flutter.dev/docs/development/ui/widgets-intro.

[10] R. Schauer. *Code Repository des KAP Akne Informationsanwendung*. URL: https://github.com/rs-maker/Prospect.

[11] *rxdart*. URL: https://pub.dev/packages/rxdart.

[12] *url_launcher*. URL: https://pub.dev/packages/url_launcher.