Inhaltsverzeichnis

1 Anforderungsanalyse

1.1 Funktionale Anforderungen

Die Funktionalen Anforderungen dienen dafür, um die genaue Funktionalität und das Verhalten eines Systems zu beschreiben. Es wird behandelt, was das System können soll und muss. Um dies besser abbilden zu können, werden die einzelnen Anforderungen einer Gewichtung unterzogen. Diese Gewichtung lässt sich in Form von Muss","Soll"und "Kann"Anforderungen ausdrücken. Aufgrund des kleines Rahmens und Zeitfensters dieser Thesis wird sich im diesen Teil ausschließlich auf MussAnforderungen beschränkt, d.h. jene Funktionalität, welches das Framework erfüllen muss. Für die Übersichtlichkeit werden sämtliche Anforderungen nummeriert und mit dem Kürzel FA"versehen.

[A01] Identifizieren von "Intents"
Dem Nutzer des Frameworks muss es möglich sein, die "Intents"seiner Anwendung eindeutig zu markieren.

[A02] Trennung von Bussines und Ansicht Logik
Ähnlich wie in MVP oder MVC muss das Framework eine strikte Separierung von (Bussines) Logik und der Ansicht Logik fördern.

[A03] Überprüfung des Models als unveränderliche Datenstruktur
Es ist zwingende erforderlich, dass die vom Nutzer gewählte Struktur für das Model nicht direkt verändert werden kann. Hierfür muss das Framework verifizieren, dass es sich um eine unveränderliche Datenstruktur handelt.

[A04] Bereitstellung eines Reducers"
Für das erzeugen von einem neuem Model muss dem Entwickler eine Funktionen in Form eines Reducers bereitgestellt werden. Diese muss das derzeitige Model und Ereignis zur Verfügung stellen.

[A05] Handhabung von Seiteneffekten
Auf Grund der Tatsache, dass in den meisten Anwendungen Seiteneffekte auftreten, muss dieses vom Framework abgedeckt sein.

[A06] Verwalten des Zustands
Ein zentraler Bestandteil des Framework ist es, das Model bzw. den Zustand der Anwendung zu Verwalten.

[A07] Funktion für die Aktualisierung der Ansicht
Nachdem ein neues Model erzeugt wurde, muss dieses an die Ansicht weitergegeben werden, welches sich daraufhin aktualisiert. Hierfür muss das Framework eine dementsprechende Funktion bereitstellen.

[A08] Speichern des Zustands
Der Zustand muss im transienten und persistenten Speicher abgelegt werden.

[A09] Wiederherstellung des Zustands
Sollte es zu einem Verlust der Models kommen, muss dieses ordnungsgemäß und wiederhergestellt werden. Dies sollte ohne Eingriff des Entwicklers von statten gehen.

[A10] Asynchrone Ausführung
Viele der Zugriffe auf eine Datenbank oder einer Rest-Api finden auf unterschiedlichen Threads statt. Hierbei darf es zu keinen unerwarteten Problemen (z.B. Race.Conditions) kommen.

1.2 Nicht funktionale Anforderungen

Die nicht funktionale Anforderungen sind im Gegensatz zu den funktionalen unspezifisch für ein Produkt, d.h. sie haben meist nur einen indirekten Einfluss auf das System. So kann beispielsweise festgelegt werden, dass die Ausführung einen bestimmten Funktionalität nur ein gewisses Maß an Zeit in Anspruch nehmen darf. Dazu gehören auch Qualitätsmerkmale, die erfüllt werden müssen.Für die Übersichtlichkeit werden sämtliche Anforderungen nummeriert und mit dem Kürzel NFA"versehen.

[A11] Dogmatisches Framework
Das Framework soll den Entwickler "and die Hand nehmen"und genaue Vorgaben für die Anwendung kommunizieren. Damit sind zielgenauere Funktionen möglich und die Komplexität kann unter Umständen niedriger gehalten werden. Dies birgt allerdings die Gefahr, dass bei eigenwilliger Anwendung - und der damit einhergehenden Abweichung der Instruktionen - des Entwicklers es zu Komplikationen kommen kann.

[A12] Unidirektional
Der in MVI geforderte unidirektionale Datenfluss muss eingehalten werden.

[A13] Kotlin spezifische Implementierung
Für die Umsetzung wird Kotlin als Programmiersprache herangezogen. Sie bietet einige syntaktische Vorteile gegenüber Java.

[A14] Reaktiv
Reaktiv

[A15] Funktional
Funtkional

1.3 Ubersicht der Anforderungen

Für eine bessere Übersicht der funktionalen (fa) und nicht funktionalen (nfa) Anforderungen werden diese im Folgenden in einer Tabelle zusammengefasst. Zusätzlich wird ein Verweis auf das Design bzw. Konzept und die jeweilige Implementierung angefügt.

ID	Anforderung	Typ
.1em.1em.1em 01	Identifizieren von 'Intent', 'Action', 'State' und 'Result'	fa
02	Trennung von Business und Ansicht Logik	fa
03 Bereitstellung eines 'Reducers'	Überprüfung des Zustands als unveränderliche Datenstruktur fa	fa heigh
05	Handhabung von Seiteneffekten	fa
06	Verwalten des Zustands	fa
07	Funktion für die Aktualisierung der Ansicht	fa
08	Speichern des Zustands	fa
09	Wiederherstellung des Zustands	fa
10	Asynchrone Ausführung	fa
11	Dogmatisches Framework	fa
12	Unidirektionaler Datenfluss	fa
13	Kotlin spezifische Implementierung	fa
14 fa	Reaktiv	Funktio

1.4 Begutachtung bestehender MVI-Frameworks/Bibliotheken

In diesem Kapitel wird ein Blick auf bereits bestehende MVI-Frameworks/Bibliotheke geworfen.

1.4.1 MVICore

Hierbei handelt es sich um ein auf MVI ausgerichtet Framework dessen Entwicklung seit 2018 auf Github stattfindet. [1] Es bietet eine vollständige Abdeckung der von MVI verlangten Komponenten und stellt darüber hinaus weitere Eigenschaften wie 'Time Travel Debuggig' und 'Middlewares' zur Verfügung.

1.4.2 Roxie

Abbildungsverzeichnis

Online References

[1] badoo. *MVI framework with events, time-travel, and more*. 25. Feb. 2018. URL: https://github.com/badoo/MVICore (besucht am 23. 08. 2019).