Inhaltsverzeichnis

1 Einleitung

Bis zum Jahre 1973 und der Entwicklung des ersten Eingabegerätes mit Graphischer Oberfläche (GUI) (Xerox Alto [1]) erfolgte die Interaktion mit einem Computer im Wesentlichen über eine Konsole. Dies reduzierte die Ein- und Ausgabe eines Programms auf rein textuelle Elemente. Seither hat sich viel getan: Die Erschaffung des Internets leitet den Beginn von Webseiten ein, welche von einer anfänglich statischen Ausprägung zu der heutigen dynamisch und komplexen wuchsen. Dazu gesellten sich im Laufe der Zeit mobile Endgeräte - zu Beginn bestückt mit Tasten für die Eingabe, sowie einem primitiven Bildschirm für die Anzeige finden sich heutzutage vorwiegend leistungsfähige, auf einem kapazitiven Touchscreen basierende Smartphones wieder. Hierbei ist über die Jahre der Funktionsumfang von Betriebssystem und Applikationen im Allgemeinen gestiegen.

1.1 Problemfeld

Die Herausforderung für einen Entwickler ist es, die Nutzerschnittstelle (auch: Presenation Layer [2]) , innerhalb einer "Drei-Schichtenarchitektur"[3] sinnvoll aufzubauen und dabei die Modellierung und Verwaltung des Zustandes innerhalb einer Applikation zu berücksichtigen. Hierfür müssen auch externe Vorkommnisse wie bswp. die Aktualisierung der zugrundeliegenden Datenbank miteinbezogen werden.
Da diese Problematik nicht erst seit kurzem sondern seid vielen Jahren besteht, wurden hierfür

1.2 Ziel der Arbeit

1.3 Aufbau der Arbeit

2 Grundlagen

In diesem Kapitel gilt es zu klären, auf welchen Grundlagen, Ideen und Konzepten Model-View-Intent beruht, wie diese miteinander fungieren und weshalb sie als Inspiration dienten.

2.1 Unidirektionaler Datenfluss: Flux, Redux und React

In einer Applikation existieren grundsätzlich zwei Komponenten: Eine, die der Nutzer wahrnehmen kann und eine, die für ihn unsichtbar bleibt. Bei ersterer handelt es sich meist um das, was der Nutzer(auf dem Bildschirm) sieht - die sogenannte „View“. Die zweite Komponente beschreibt die Ebene, welche das Geschehen observiert, darauf reagiert und den weiteren Verlauft (zum größten Teil) kontrolliert. Sie kann unter anderem als „Controller“ betitelt werden.
Der Zustand in dem sich eine Applikation befindet kann hierbei von beiden Seiten modifiziert und beobachtet werden. Ist dies der Fall, so handelt es sich um einen bidirektionalen Datenfluss. Bei dieser Variante entsteht die eventuelle Gefahr von kaskadierenden Updates als auch in einen unvorhersehbaren Datenfluss zu geraten. Des weiteren muss immer überprüft und sichergestellt werden, dass „View“ und „Controller“ synchronisiert sind, da beide den globalen Zustand darstellen. Schlussendlich verliert man zusätzlich die Fähigkeit zu entscheiden, wann der Zustand manipuliert wird.
Ein anderer Ansatz ist, den Datenfluss in eine Richtung zu beschränken und ihn damit unidirektional [4, 5] operieren zu lassen. Diese Variante erfreut sich an zunehmender Popularität seit der Bekanntmachung der „Flux“ [6] Architektur im Jahre 2015 von Facebook. [7]

2.1.1 Flux

Für die Einhaltung und Umsetzung eines unidirektionalen Datenfluss. Die bedient „Flux“ bei zwei fundamentalen Konzepten: Der Zustand innerhalb einer Applikation wird als „single source of truth (SSOT)“ angesehen und darf keine direkte Änderung erfahren. Um dies zu Gewährleisten finden sich mehrere Komponenten in „Flux“ wieder:

Action: Eine Aktion beschreibt ein Ereignis, welches zum Beispiel vom Nutzer ausgelöst werden kann. Jeder dieser Aktionen wird dabei ein Typ zugewiesen. Des weiteren können zusätzliche Attribute an eine Aktion gebunden werden.

```
{
 type: ActionTypes.INCREMENT,
 by: 2
}
```

Dispatcher: Dieser ist für die Entgegennahme und Verteilung einer Aktion an sogenannte „Stores“ zuständig. Er besitzt die wichtige Eigenschaft der sequentiellen Verarbeitung, d.h., dass er zu jedem Zeitpunkt nur je eine „Action“ weiterreicht.

Abbildungsverzeichnis

Book References

[2] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 13. Jan. 2013, S. 19–22.

[3] Donald Wolfe. *3-Tier Architecture in ASP.NET with C sharp tutorial*. SitePros2000.com, 13. Jan. 2013.

[4] Adam Boduch. *Flux Architecture*. Packt Publishing, 30. Jan. 2017, S. 27, 198, 312.

[5] Ilya Gelman und Boris Dinkevich. *The Complete Redux Book*. Leanpub, 30. Jan. 2017, S. 6–7.

[6] Adam Boduch. *Flux Architecture*. Packt Publishing, 30. Jan. 2017.

Artikel Referenzen

[1] Thomas A Wadlow. „The Xerox Alto Computer“. In: (Sep. 1982). URL: https://tech-insider.org/personal-computers/research/acrobat/8109-e.pdf.

Online References

[7] Facebook Developers. *Hacker Way: Rethinking Web App Development at Facebook*. 4. Mai 2014. URL: https://www.youtube.com/watch?v=nYkdrAPrdcw (besucht am 08. 07. 2019).