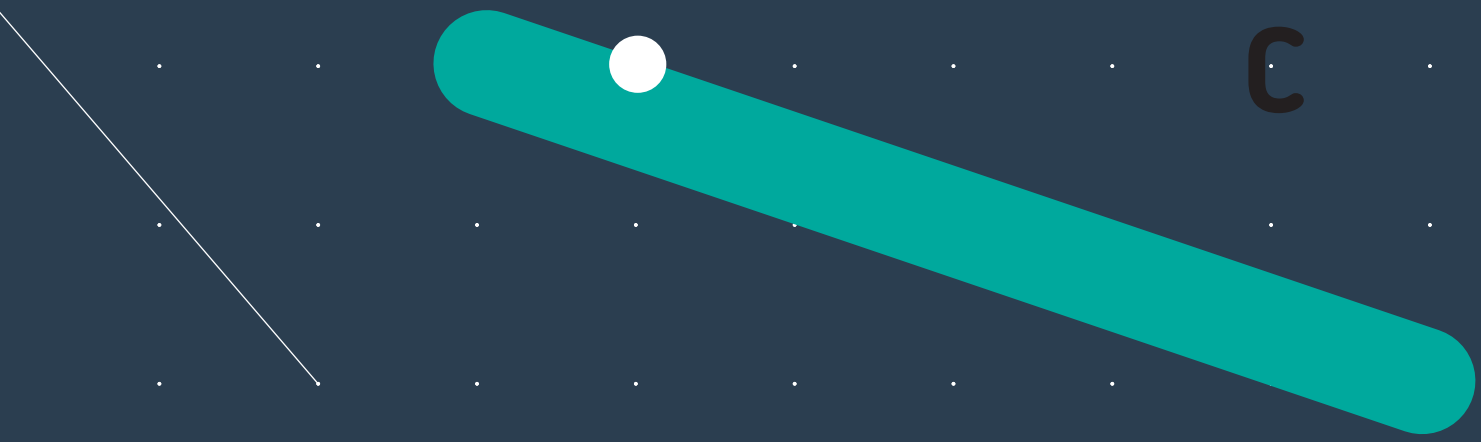
Hochschule Konstanz
Department of Computer Science

Autor
Jonas Reinwald

Immatrikulationsnummer
293320

E-Mail
jonas.reinwald@htwg-konstanz.de



Bachelorthesis

Konzipierung der Architektur einer Darstellungsschicht basierend auf aktuellen Webtechnologien im Hinblick auf eine zukünftige Anbindung an das Programm combit Relationship Manager

Konstanz, 4. Januar 2019

Hochschule Konstanz
Department of Computer Science

Bachelorthesis

Konzipierung der Architektur einer Darstellungsschicht basierend auf aktuellen Webtechnologien im Hinblick auf eine zukünftige Anbindung an das Programm combit Relationship Manager

von

Jonas Reinwald

Zur Erlangung des akademischen Grades eines

Bachelor of Science (B.Sc.)

an der Hochschule Konstanz für Technik, Wirtschaft und Gestaltung

Vorgelegt von:	Jonas Reinwald Wessenbergstr. 13 78462 Konstanz 293320
Erstbetreuer:	**Prof. Dr. Marko Boger**
Zweitbetreuer:	**Dipl.-Inf. Alexander Horak** combit GmbH Untere Laube 30 78462 Konstanz
Ausgegeben am:	01.12.2018
Eingereicht am:	28.02.2019

Inhaltsverzeichnis

Danksagung

Stefanie Frischknecht: Ideen + Korrekturlesen
Prof. Dr. Marko Boger: Betreuer
Dipl.-Inf. Alexander Horak: Betreuer

Zusammenfassung (Abstract)

Webapplikationen in Form von mobilen Apps, Single-Page-Applikationen, … FIXME: beispiel machen heute bereits einen sehr großen Teil häufig genutzter Software aus. In Zukunft wird der Anteil von webbasierter Software im Vergleich zu traditioneller Software noch weiter ansteigen (FIXME: quelle). Auch für Unternehmen ist es langfristig wichtig diesem Trend zu folgen um Kunden flexiblere Softwarelösungen anbieten zu können und nicht den Eindruck zu erwecken man könne nicht mit modernen Entwicklungen mithalten (FIXME: formulierung). Die Firma combit GmbH entwickelt mit dem combit Relationship Manager eine Software für das Management von Kundenbeziehungen (Customer Relationship Management), die zum momentanen Zeitpunkt aus einer Desktopapplikation, einer Webapplikation und einer mobile Webapplikation besteht, welche unabhängig voneinander entwickelt und benutzt werden. Im Rahmen dieser Bachelorthesis soll ein Architekturkonzept für eine einheitliche Oberfläche (ThinClient, FIXME: erklärung) basierend auf moderner Webtechnologie erarbeitet werden, mithilfe derer die drei momentanen Ausführungen des combit Relationship Managers abgelöst werden können.

FIXME: formulierung: Zur Erstellung dieses Konzepts beschäftigt sich diese Arbeit unter anderem mit der Evaluierung eines geeigneten Frontend-Frameworks bzw. einer Frontend-Bibliothek, der Kommunikation zwischen neuem UI-Layer und Backend und der Unterstützung aller bisherigen Features der Desktopapplikation. Zwei besonders spannende Probleme existieren hier zum einen in der Flexibilität der momentanen Oberfläche, welche von jedem Nutzer individuell anpassbar ist. Diese Flexibilität soll durch eine automatische Konvertierung zu einem von der neuen UI darstellbarem Format und weiterhin anpassbarem Format erhalten bleiben. Zum anderen bietet die Desktopapplikation die Möglichkeit Skripte und automatisierte Aufgaben zur Anpassung der zugrunde liegenden Daten, für Darstellungsbedingungen und weitere Zwecke direkt auf dem Client auszuführen. Das Ausführen auf dem Client ist bei Webapplikationen nicht ohne Weiteres möglich, weshalb eine andere Lösung für diese Funktionalität gefunden werden muss.

1

Einleitung

1.1. Motivation

1.2. Erwartete Schwierigkeiten

Bereits im Vorfeld soll ein kleiner Überblick über die Bereiche dieser Arbeit gegeben werden, welche im Laufe der Umsetzung zu Schwierigkeiten führen könnten. Dies dient sowohl der Kontrolle der eigenen Fähigkeit Probleme bereits im Voraus korrekt einschätzen zu können, als auch dazu Einschätzungen von Problemstellen zukünftiger Projekte zu verbessern indem am Ende der Arbeit mit den tatsächlichen problematischen Stellen verglichen wird. Weitere Informationen zu den hier aufgelisteten Punkten werden in Kapitel 2 FIXME: link aufgeführt.

- framework mit dem automatische Umsetzung von UI (+ von Kunden weiterhin möglichst einfach individualisierbar) -> dazu speichern von layout in DB anstatt bisher in einer Datei -> pro user - API die etwas taugt für Kommunikation zwischen UI und bisherigem Programm - speichern von dateibasierten Informationen (Projektdatei, registry und dli) muss in DB geschehen (- scripting, Tastenkombinationen, …)

1.3. Kapitelübersicht

In Kapitel 2 FIXME: link werden Anforderungen (auch anhand bisheriger Features) ermittelt. In Kapitel 3 FIXME: link werden die in Betracht gezogenen Technologien und deren Zusammenspiel untereinander vorgestellt. Dazu gehören die zu nutzende Programmierumgebung (Sprache, Testframework, Continuous Integration), der Vergleich verschiedener Frontend-Frameworks die anhand der in Kapitel 2 FIXME: link formulierten Anforderungen bewertet werden und die Protokolle FIXME: rest und graphql protokoll? zur Kommunikation mit dem Backend. In Kapitel FIXME: neues Kapitel: Ausarbeitung wird die Ausarbeitung des Konzepts (FIXME: API erwähnen?) mit den in Kapitel 3 FIXME: link ausgesuchten Technologien beschrieben. In Kapitel 4 FIXME: link wird der Entwicklungsprozess der beispielhaften Umsetzung des erstellten Konzepts dargestellt. In Kapitel 5 FIXME: link wird das Ergebnis kritisch hinterfragt, die tatsächlich aufgetretenen Schwierigkeiten mit den zuvor Erwarteten verglichen und ein Fazit gezogen.

2

Anforderungen

2.1. Produktüberblick

2.2.

2.3. Offline-First approach

FIXME: Übersetzung

2.4. Hauptaugenmerk dieser Arbeit

3

Technologien

3.1. Entwicklungsumgebung

3.1.1. Typescript

- OS Sprache von Microsoft - syntaktische Obermenge von zu JS - Optionale statische Typen für JavaScript -> Typsicherheit -> Codevervollständigung - wird zu JavaScript transpiliert -> es können automatisch die neusten Features von JS (ES6+) benutzt werden - mehr Sicherheit und bessere Unterstützung durch Entwicklertools beim Programmieren - viele Fehler die in JS erst zur Laufzeit auffallen und schwer nachvollziehenden Problemen führen werden bereits beim Kompilieren gefunden - Nachteil nur der zusätzliche compile Schritt bzgl. Tooling / Aufwand

3.2. Frontend-Frameworks

3.3. REST (OData) vs. GraphQL

4

Ausarbeitung

5

Implementierung

6

Fazit