Abschlussprüfung Sommer 2022

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Fußzeile als Komponente mit funktionalen Knöpfen und Darstellung von Informationen für die Abschlussstrecke eines Versicherungsproduktes

Abgabetermin: Hannover, den 27.04.2022
Prüfungsausschuss: FIAE 3

Prüfungsbewerber:
Jakob Schumann
Goethestraße 40
30169 Hannover

Ausbildungsbetrieb:
Konzept & Marketing – ihr unabhängiger Konzeptentwickler GmbH
Podbielskistraße 333
30659 Hannover

Inhaltsverzeichnis

Abkürzungsverzeichnis

K&M Konzept und Marketing GmbH

SPA Single-Page-Application

OTR Onlinetarifrechner

GUID Globally Unique Identifier

TG Tarifrechner-Gateway

TRG Tarifrechner-REST-Gateway

DTO Data Transfer Object

CI/CD Continuous Integration/Continuous Delivery

Glossar

.NET 6.0 Eine Softwareplattform für das Entwickeln und Ausführen von Anwendungen. 11

API Application Programming Interface, Programmierschnittstelle; Ein Programmteil der zur Anbindung für andere Systeme zur Verfügung gestellt wird. 11

Automapper Eine Bibliothek mit der eine Objekt-zu-Objekt Datenübertragungsstrategie konfiguriert werden kann. 12

Container Laufende Instanz eines Softwarepakets im Dockerumfeld. 8

Dependency Injection Ein Entwurfsmuster welches Abhängigkeiten von Objekten zur Laufzeit auflöst. 12

Domain-Driven Design Modellierung der Software nach umzusetzenden Fachlichkeiten der Anwendungsdomäne. 8, 12

Javascript-fetch Asynchroner Aufruf an ein Backend. 11

Helm Packetmanager und Hilfstool für Kubernetes. 8

HTML Hypertext Markup Language; textbasierte Auszeichnungssprache um Inhalte in Webbrowsern darzustellen. 11

HTTP Hypertext Transfer Protocol; Protokoll für Datenübertragung. 11

Kubernetes (Orchestrator für Containeranwendungen). 8

Linter Ein Tool zur statischen Codeanalyse. 9

npm Ein Paketmanager für Node.js Pakete. 10

Pinia-Store Eine Datenzustandsverwaltung für Vue.js welche den Localstorage des Browsers verwendet. 10, 11

Repository-Pattern Ein Designpattern, welches die Logik der Datenspeicherung- und beschaffung kapselt, so dass eine zentrale Stelle für Datenquellen besteht. 8

REST Reprensentational State Transfer; Basierend auf HTTP; ein Paradigma zur Kommunikation zwischen Systemen. 8

TypeScript Eine typsichere Erweiterung von JavaScript. 10

Vue.js Ein clientseitiges Javascript-Framework nach dem Model-View-ViewModel-Entwurfsmuster für das Erstellen von Webanwendungen. 10, 11

Vuetify Eine Vue.js Oberflächenkomponentenbibliothek. 10

1. Einleitung

1.1. Projektumfeld

Konzept und Marketing GmbH

Bei der Konzept und Marketing GmbH (K&M) handelt es sich um einen Finanzdienstleister der Versicherungskonzepte entwickelt und diese auch Komplett von der Annahme bis hin zur Schadensregulierung verwaltet.
Das Unternehmen, bestehend derzeit aus 118 Mitarbeitern, bietet Dienstleistungen sowie Portale für Versicherer, Makler und Kunden an.

Mariosoft

Bei der Mariosoft handelt es sich um eine IT-Abteilung der K&M. Die Mariosoft besteht derzeit aus 18 Mitarbeitern und beschäftigt sich mit der Konzeption, Programmierung und Wartung von internen Softwarelösungen wie Onlinetarifrechner (OTR), Kundenverwaltung und Übersichtsseiten für Makler.
Das umzusetzende Projekt ist im Umfeld der Mariosoft angesiedelt und ist eine Komponente für eine gleichzeitig entstehende Webanwendung.

1.2. Projektziel

Ziel des Gesamtprojekts ist ein neuer, modularer OTR als Webseite bzw. Single-Page-Application (SPA). Dieser soll so entworfen werden, dass die einzelnen Komponenten austauschbar sind und für weitere Versicherungsprodukte benutzt werden kann. Außerdem soll die Benutzeroberfläche einheitlich gestaltet werden. Allgemein soll die Einführung neuer und die Erweiterung bestehender Versicherungsprodukte vereinfacht werden. Der OTR wird bestehen aus einem .NET 6.0 Backend, welches unter anderem viele Schnittstellen zu umliegenden, firmeninternen Systemen, einen Cachespeicher und weitere Logik auf welche im Verlauf dieser Dokumentation eingegangen wird, besitzt. Das Frontend besteht aus Vue.js mit Designvorlagen von Vuetify sowie einer eigenen Oberflächenkomponentenbibliothek, welche Vuetifyelemente kapselt und im Corporatedesign zur Verfügung stellt. Im Frontend benutzen wir TypeScript als Programmiersprache.
Bei einem OTR hat ein Makler die Möglichkeit Angaben zu einem Versicherungstarif zu treffen und sich anhand dessen Beiträge ausrechnen zulassen. Außerdem sind hilfreiche Funktionalitäten gegeben, wie z.B. das Ausdrucken von Anträgen und Angeboten, das Speichern von Angeboten, um diese später wieder laden und weiter bearbeiten zu können und das Versenden von tarifrelevanten Dokumenten per E-Mail. Diese Funktionalitäten werden in der Fußzeile in Form von

Schaltflächen angeboten. Die Implementierung dieser Fußzeile wird von mir im Rahmen dieses Projekts umgesetzt. Die Fußzeile beinhaltet außerdem Angaben zum Copyright und einen Verweis zum Impressum.

Beim Speichern eines Angebots sollen alle im OTR eingegebenen Daten im sogenannten Maklerportal für den angemeldeten Makler gespeichert werden. Diese können über das Maklerportal wieder geladen werden. Außerdem ist es vorgesehen, dass Makler Angebote kopieren können. Die K&M bietet auch OTR außerhalb des Maklerportals an. Für diese Rechner ist dann keine Angebots- oder Kopiespeicherung vorgesehen, dementsprechend sind die Schaltflächen dort nicht vorhanden.

Die Druckenfunktionalität wird durch zwei Schaltflächen abgebildet, einen um ein Angebot zu drucken, der zweite für den Antrag. Ein Angebot wird z.B. dem Endkunden vom Makler unterbreitet und ist, genau wie der Antrag, ein verbindliches Dokument mit welchem ein Versicherungsvertrag abgeschlossen werden kann. Die Funktionalität wird häufig genutzt und ist damit wichtig. Ein Dokument soll dann gedruckt werden können, wenn Beiträge erfolgreich berechnet wurden. Dafür müssen alle berechnungsrelevanten Angaben getroffen worden sein. Die Schaltflächen sind bis das geschehen ist deaktiviert und werden reaktiv aktiviert. Außerdem kann es sein, dass für einen Tarif der Angebot- oder Antragsdruck deaktiviert ist. Dementsprechend sind die Schaltflächen aktiviert, oder nicht aktiviert. Das Drucken ist dabei eigentlich das Herunterladen eines PDF-Dokumentes, damit der Makler dieses auch per E-Mail verschicken oder selber ausdrucken kann. Das Dokument wird von einem angebundenem System anhand der Daten in XML-Format generiert.

Eine weitere Funktion ist die des Dokumentenversands. Bei Drücken dieser Schaltfläche öffnet sich ein Dialog, wo der Nutzer die gewünschten Dokumente aus allen relevanten Dokumenten auswählen kann um sie zu versenden. Dazu muss noch eine oder mehrere E-Mails angegeben werden und dann werden die Dokumente per E-Mail in einem ZIP-Archiv versendet. Die Dokumente beinhalten rechtliche Informationen und den Antrag sowie das Angebot. Die komplette Dokumentenversand-Komponente gibt es als eigenständige Bibliothek, da sie auch für andere OTR genutzt wird, so dass diese für das Frontend eingebunden werden musste.

1.3. Projektbegründung

Die Motivation hinter diesem Projekt begründet sich in der einfachen Erweiterbarkeit für neue Versicherungsprodukte sowie ein Redesign des Aussehens des OTR. Der OTR ist durch den modularen und komponentenbasierten Aufbau des Projekts besser wartbar.
Die Fußzeile ist eine Komponente die in allen bestehenden Produkten und zukünftig kommenden Produkten vorhanden sein muss. Die Platzierung der gegebenen Funktionalitäten in die Fußzeile

bietet sich dafür gut an, da sie unabhängig von dem Kontext der Rest der Webseite funktionieren soll.

1.4. Projektschnittstellen

Das Backend hat Schnittstellen zu einigen Systemen. Es gibt das Tarifrechner-Gateway (TG), welches den Einstiegspunkt von Extern zu den Online Tarifrechnern darstellt. Dieses sowie der OTR benutzt das Tarifrechner-REST-Gateway (TRG), welches als REST-Schnittstelle funktioniert um unter anderem Konfigurationen für Tarife oder das Speichern von Angeboten anzubieten. Eine weitere Schnittstelle die von der Fußzeilenkomponente genutzt werden ist der PDFToolsservice, welcher die Generierung von PDF-Dokumenten übernimmt. Die Fußzeile sowie alle Komponenten im Frontend bekommen ihre Daten über den Localstorage des Browsers (Vue.js Pinia Store) geliefert.
Das Maklerportal ist eine Webseite an der sich Makler anmelden können, wo sie den OTR aufrufen können, Übersichten über ihr Kundenbestand haben und noch vieles mehr. Zu dem können dort auch gespeicherte Angebote eingesehen, bearbeitet und mit einem OTR aufgerufen werden.
Der OTR wird vor allem von Maklern benutzt um Tarife für ihre Endkunden auszurechnen.

1.5. Projektabgrenzung

Das bearbeitete Projekt ist hierbei nur die Fußzeile mit den Schaltflächen einschließlich ihrer Funktionalitäten. Eine Benutzeroberflächenkomponentenbibliothek ist gegeben, aus welcher die Schaltflächen eingebunden werden. Genauso ist eine Synchronisation zwischen Front und Backend durch HTTP-JSON-PATCH schon vorhanden. Weiterhin ist ein Backend mit mehreren Schnittstellen welche nur für die einzelnen Funktionalitäten erweitert werden müssen verfügbar. Im Frontend ist ein Gridlayout mit Elementvorlagen gegeben, welche genauso erweitert werden müssen.

2. Projektplanung

Das Projekt wird in dem Zeitraum vom 01.03.2022 bis zum 19.03.2022 durchgeführt.

2.1. Projektphasen

Projektphasen	Zeit in Stunden
Tägliche Absprachen mit Entwicklungsteam & Auftragsgeber	1
Wöchentliche Absprache zur Oberfläche	1
Ist-Analyse:	**6**
Analyse vorhandener Codebasen und Funktionalitäten	3
Gespräche mit dem Auftragsgeber	3
Sollkonzept:	**4**
Planung des Soll-Zustands Projekt allgemein	2
Planung des Soll-Zustands Fußzeilenfunktionalität	1
Abstimmung & Sichtung Workflows und Oberflächenentwürfe	1
Realisierung:	**30**
Frontend	**12**
Knöpfe eingebunden & Styling	3
Funktionalität Backend Anbindung, optionales Rendering	7
Eventhandling, Label mit i18n	2
Backend	**18**
Angebotsspeicherung	8
Kopiespeicherung	2
Dokumentengenerierung	7
Entwicklung Oberfläche	1
Dokumentation	**19**
Testen & Abnahme	**8**
Gesamt	**69**

2.2. Abweichung vom Projektantrag

Alle im Projektantrag beschriebenen Funktionalitäten wurden umgesetzt. Hinzu kam die Kopiespeicherung eines Angebotes. Außerdem war schon mehr Grundgerüst im Form von bestehenden Serviceprojekten und Schnittstellen gegeben, wodurch nur noch die reine Logik implementiert werden musste. Die UI-Elemente mussten neu erstellt werden oder ggf.. erweitert oder angepasst werden.

2.3. Ressourcenplanung

Für die Umsetzung des Projekts wurden Ressourcen verwendet für die K&M bereits Lizenzen hat oder die unentgeltlich zur Verfügung stehen. Des Weiteren wurde Hardware verwendet die bereits im Besitz des Unternehmens ist. Folgende Ressourcen wurden verwendet:

- **Technische Ressourcen**
  - Windows 10
  - Visual Studio 2022
  - Webstorm 2021
  - TeXstudio
  - GitLab
  - Fork - GitClient
  - Kubernetes/Docker
  - Microsoft Teams für Besprechungen
- **Personen**
  - Produktmanagement, Vertrieb, externer Versicherer für Anforderungsdefinition
  - Marketingabteilung für Designanforderungen
  - Auszubildender für die Umsetzung und Tests

2.4. Entwicklungsprozess

Das Projekt wurde mit dem Spiralmodell durchgeführt, unterstützt durch ein Kanban-Board. Vor dem Start der Bearbeitung wurden grobe Aufgaben und Anforderungen definiert. Es gab tägliche Absprachen zwischen dem Entwicklerteam sowie wöchentliche Absprachen mit dem Produktmanagement und der Marketingabteilung bezüglich der Oberfläche. Aus diesen Absprachen sind iterativ neue Anforderungen entstanden. Durch Continuous Integration/Continuous Delivery (CI/CD) wurden inkrementell umgesetzte Programmteile zum Testen zur Verfügung gestellt. besseres
Dieses Vorgehensmodell wurde gewählt damit man die Entwicklung schnell an neue Anforderung Wort
der Kunden, bzw. der Stakeholder anpassen kann. finden

3. Analysephase

3.1. Ist-Analyse

Die K&M bietet OTR für Makler an. Zur Zeit gibt es mehrere Generationen an Rechnern. Die älteste ist mit PHP umgesetzt und hat eine altmodische Benutzeroberfläche. Eine neuere Tarifrechnerbasis ist ähnlich wie diese Neuentwicklung aufgebaut auf einem .NET-Backend und einem Vue.js Frontend. Allerdings stellte sich bei dem Einführen neuer Tariflinien heraus, dass diese sich nicht einfach in diesen OTR einbinden ließ.
Demnach existieren funktionale OTR, die beim Vertrieb von Versicherungsprodukten helfen, allerdings sind diese entweder für den Endnutzer nicht anschaulich oder modern und/oder für den Entwickler schwierig zu warten oder zu erweitern.

3.2. 'Make, Buy, or Reuse' - Analyse

Dadurch, dass die K&M neue und innovative Versicherungskonzepte auf den Markt bringt, eignen sich sogenannte Baukastenrechner nicht für diese Tarife. Die Vorlagen dieser Rechner sind nicht so individuell gestaltbar wie die K&M es benötigt.
Da es schon ältere Generationen an Tarifrechner gibt, wäre auch eine Überlegung einfach eine ältere Tarifrechnerbasis weiter zu benutzen, anstatt einen neuen einzuführen. Die älteste Generation ist allerdings technologisch nicht tragbar und muss grundlegend erneuert werden. Ein neuer OTR sollte die neue Grundlage werden, allerdings war dieser Versuch nicht erfolgreich.
In der folgenden Nutzwertanalyse wurden die verschiedenen OTR miteinander verglichen. Hierbei hab ich mich für die Kriterien Aussehen, Aktualität der Technologie, bzw. die Zukunftssicherheit, Erweiterbarkeit und Anpassbarkeit entschieden. Erweiterbarkeit und Anpassbarkeit haben eine große Gewichtung, da dies für die K&M aufgrund individueller Tarife aber auch für die Entwicklung in der Mariosoft, sehr wichtig ist. Ebenso ist die Zukunftssicherheit und die Aktualität der eingesetzten Technologien wichtig. Weniger relevant ist das Aussehen der Benutzeroberfläche, trotzdem fließt diese in die Bewertung ein, da dieser Punkt für den Endkunden wichtig ist. Die Bewertung geht von 0-10 Punkten, wobei 10 die best möglichste Bewertung ist.

		1. Generation OTR ('Universal')		2. Generation OTR ('Spirit')		3. Generation OTR	
Kriterium	Gewichtung	Bewertung	Bewertung (gewichtet)	Bewertung	Bewertung (gewichtet)	Bewertung	Bewertung (gewichtet)
Aussehen	15%	3	4,5	7	10,5	9	13,5
Technologie	25%	2	5	7	17,5	9	22,5
Erweiterbarkeit	30%	3	9	6	18	8	24
Anpassbarkeit	30%	3	9	5	15	7	21
Summe	100%	11	**27,5**	25	**61**	33	**81**

Der Rechner 'Universal' ist in PHP programmiert, einer Programmiersprache mit welcher in der

Mariosoft nicht mehr weiter entwickelt werden soll. Das nötige Personal mit PHP-Fachwissen fehlt. Außerdem wird das verwendete Framework nicht mehr unterstützt und das Update auf eine neuere Version bringt zu viele Breakingchanges mit sich. Die Erweiterbarkeit und Anpassbarkeit ist bei diesem Rechner auch eingeschränkt, da das Frontend anhand von XML-Vorlagen vorgegeben wird. Dadurch ist es schwierig Besonderheiten bei einigen Tarifen umzusetzen und diese zu warten. Die Oberfläche ist sehr altmodisch.
Hingegen läuft der OTR 'Spirit' auf einer neuer Technologie, ähnlich dem neuen OTR. Das Backend ist eine .NET 5.0 API und das Frontend eine Vue.js 2 **SAP!** (**SAP!**). Dieser Rechner wurde zwar mit einem generischem Ansatz geplant, allerdings lief die Logik schon beim Einführen des zweiten Tarifs zu weit auseinander. So wurde unter anderem eine andere Datenhaltungsmethode benutzt und die Struktur im Frontend war zu spezifisch zum ursprünglichen Tarif. Dadurch ist die Implementierung neuer Tarife nicht einheitlich und kompliziert.

Der neue OTR soll ein universeller Ansatz werden um alle zukünftigen Produkte gleich abzubilden. Durch die aktuellsten Technologien wird eine gewisse Langlebigkeit garantiert und durch den modularen und generischen Ansatz ist es einfach neue Produkte abzubilden. Durch ein Corporate-Design und die Einbindung der Marketing-Abteilung in der Konzeption der Benutzeroberfläche ist diese anschaulicher und benutzerfreundlicher.

3.3. Projektkosten

Die Kosten des Gesamtprojekts, also der kompletten OTR Basis, belaufen sich auf ca. 80.000-100.000€. Dazu gehören allerdings mehrere Nebeneffekte die aus dem Gesamtprojekt resultieren. So wurde im Rahmen des OTRs die CI/CD Pipeline von Jenkins auf GitLab umgestellt. Außerdem wurde ein einheitliches Corporate-Design eingeführt, an welchem die Marketingabteilung gearbeitet hat. Der erste Tarif in dem OTR wird eine Betriebshaftpflicht geplant, welche ausgearbeitet werden musste und anhand welcher alle tarifspezifischen Komponenten des Rechners implementiert wurden. Dazu kommt, dass das Bestandsführungssystem für die Betriebshaftpflicht konfiguriert werden muss.

Die Kosten der Fußzeilenkomponente lässt sich anhand der aufgewendeten Stunden berechnen. Dabei wird eine Arbeitsstunde eines Auszubildenden mit 100€ berechnet. Weitere benötigte Mitarbeiter, z.B. für Absprachen, lassen sich mit durchschnittlich 130€ pro Stunde berechnen. Demnach ergibt sich daraus:
Für die Realisierung durch Auszubildenden:
69 Std * 100€/Std = 6900€
Tägliche Absprachen mit Entwicklerteam:
1 Std * 4 * 130€/Std = 520€

Wöchentliche Absprachen bezüglich der Oberfläche:
1 Std * 6 * 130€/Std = 780€
Allgemeine regelmäßige Anforderungsabsprachen mit Auftraggeber:
5 Std * 1 * 130€/Std = 650€

= 8850€ an Kosten für die Fußzeilenkomponente.

3.4. Amortisationsdauer

4. Entwurfsphase

Der OTR ist schon lange in Entwurf und wurde

4.1. Zielplattform

Das gesamte Projekt soll in einem Container innerhalb eines Kubernetes Cluster laufen. Demnach ist die Zielplattform eine Linuxdistribution. Durch Kubernetes besteht ein Loadbalancing, falls zu viele Anfragen gleichzeitig reinkommen. Außerdem wird durch die Gitlab Pipeline **cicd!** (**cicd!**) ermöglicht und in Kombination mit Helm wird das Veröffentlichen neuer Versionen stark vereinfacht.

4.2. Architekturdesign

Die Backend Architektur besteht aus Microservices nach dem Domain-Driven Design-Pattern. Außerdem wird für die Datenpersistenz das Repository-Pattern-Pattern benutzt. Im Frontend werden Komponenten benutzt, welche ihre Daten über eine zentrale Datenzustandsverwaltung beziehen. Die Kommunikation zwischen Front- und Backend geschieht über REST-Aufrufe. Daten aus dem Frontend werden per JSONPatch Protokoll und demnach über die HTTP-Patch-Methode übermittelt. Dadurch können Änderungen auch kleinteilig synchronisiert und persistiert werden.

4.3. Entwurf der Benutzeroberfläche

Der designseitige Entwurf der Benutzeroberfläche wurde durch die Marketing-Abteilung per Mockups übernommen. Diese mussten allerdings nicht eins zu eins umgesetzt werden, sondern konnten in Kombination mit den Corporate-Design-Vorlagen als Richtlinie genutzt werden. Außerdem gab es wöchentliche Absprache sowie Absprachen mit einem externen Dienstleister in denen speziell das Design abgestimmt wurde. Die Logik der Benutzeroberfläche und die technische Umsetzung wurde innerhalb des Entwicklerteams durch Prototyping erarbeitet.
Dadurch musste für die Fußzeilenkomponente speziell keine eigene Logik oder Design entworfen werden, es wurde anhand des Entwurfs für das Gesamtprojekt bearbeitet.

4.4. Datenmodell

4.5. Geschäftslogik

4.6. Maßnahmen zur Qualitätssicherung

Durch die Arbeit mit Gitflow wurde jede Unteraufgabe in einem eigenem Gitbranch bearbeitet. Diese Branches werden erst nach Codereviews vom Entwicklerteam in den develop-Branch gemerged. Automatisch laufen durch die Gitlab Pipeline Unittests im Front- und Backend. Im Backend habe ich die bestehenden Unittests erweitert um neu implementeirte Logik zu testen. Zu dem wurde im Frontend ein Linter konfiguriert.
Wenn eine Änderung auf den Developbranch gepusht wurde wird diese von einem anderen Entwickler getestet. Außerdem gibt es weitere Mitarbeiter außerhalb des Entwicklerteams die regelmäßig auf dem Testsystem Blackbox-Akzeptanztests durchführen.

git ausdrück benut- zen???

5. Implementierungsphase

5.1. Implementierung der Datenstruktur

5.2. Implementierung der Benutzeroberfläche

Die Benutzeroberfläche wird mit Vue.js, Vuetify und TypeScript realisiert. Außerdem gibt es eine interne Komponentenbibliothek als npm-Paket, welche Vuetify-Elemente kapselt und im Corporate-Design mit geringfügiger aber erweiterbarer Funktionalität anbietet. Vue.js' großer Vorteil ist es, dass die Oberfläche reaktiv gemacht wird. Das bedeutet, dass wenn sich Daten die mit einem Oberflächenelement verbunden sind ändern, wird dieses Element automatisch aktualisiert. Dies geschieht über das Observer-Designpattern. Vuetify, bzw. die Komponentenbibliothek hilft bei einem einheitlichem Design und erleichtert die Einbindung. Die durch TypeScript bereit gestellte Typsicherheit macht das Arbeiten im Team, das Benutzen von Bibliotheken, vorausgesetzt diese haben Typescriptdatentypen deklariert sowie die Wartung und zukünfitge Erweiterungen einfacher. Für die Zustandsverwaltung der Daten im Frontend wird der Pinia-Store benutzt, eine Erweiterung des Vue.js-Stores. Dieser ist typsicher, was das Arbeiten mit Typescript erleichtert. Allgemein können mit so einem Store die Daten Komponenten übergreifend einheitlich gespeichert werden. In dem OTR werden dort wichtige Daten welche das Backend bei der Initialisierung liefert gespeichert. Dazu gehören unter anderem Konfigurationen wie der OTR aufgebaut werden soll welche Eingabemöglichkeiten gegeben sind und eventuelle Sitzungsdaten, wenn eine bestehende Sitzung geladen wurde. Außerdem werden berechnete Beiträge sowie eingegebene Daten gespeichert. Alle eingegebenen Daten werden automatisch beim Speichern im Store gleichzeitig ans Backend geschickt. Es gibt mehrere Teilstores, einen allgemeingültigen Store, in welchen Daten gespeichert werden welche tarifunabhängig sind und ein Store für den jeweilig aktiven Tarif. Auf die Daten aus einer anderen Komponente zugreifen kann man mit sogenannten Gettermethoden, welche im Store definiert werden.
Eine weitere Funktion des Stores ist ein Benachrichtigungssystem. Hierbei wird bei jeder Änderung von bestimmten Daten ein Ereignis losgetreten, so dass dann eine Benachrichtigung, z.B. ein Fehler oder eine Erfolgsnachricht bei einer Aktion, ausgegeben wird.
Als Fußzeilencontainer war die <*footer*> Komponente von Vuetify gegeben. Die einzelnen Schaltflächen habe ich mit <*km-button*> umgesetzt. Dieser ist in der besagten K&M Komponentenbibliothek gegeben. Eine Besonderheit ist hierbei der Button für den Dokumentenversand. Da der Dokumentenversand relativ viel Eigenlogik benötigt und schon in einer vorherigen OTR Generation vorhanden war, gibt es diesen als eigenes npm-Paket. Im Endeffekt wird in dieser Komponente allerdings auch nur der <*km-button*> verwendet.

Die Angebot- bzw. Kopiespeichern-Schaltfläche soll nur vorhanden sein, wenn der OTR über das

Maklerportal aufgerufen wurde. Für so etwas bietet Vue.js ein konditionales Rendering als HTML-Attribut an. Im Pinia-Store wird überprüft, ob das Angebot speicherbar sein soll und dieser Wert wird an das Attribut übergeben. Beim Drücken auf den Knopf wird ein Javascript-fetch Aufruf ans Backend an den Management-Controller an die SaveOrder bzw SaveOrderCopy Action gemacht. Dieser ist durch automatisch generierte Klassen anhand der Schnittstellendefinition des Backends gekapselt. Diese kümmern sich um Fehlerbehandlung anhand des HTTP-Statuscodes, erwarten die Argumente als richtigen Datentyp und geben die Response als eigenen Datentyp zurück. Ein Angebot wird anhand einer Globally Unique Identifier (GUID) gespeichert, welche die aktuelle Sitzung des OTR repräsentiert. Außerdem wird die Identifikationsnummer des angemeldeten Maklers, bzw. Vertriebspartners benötigt. Dementsprechend werden diese Daten mit ans Backend übergeben. Dieses kann sich anhand der Session die Angebotsdaten aus dem Cache laden und diese dann an das TRG zum Speichern weiterschicken. Auf der Seite des Frontends passiert beim Kopiespeichern das Gleiche wie bei der normalen Speicherung, außer dass ein anderer Endpunkt aufgerufen wird.

Zu Spezifisch mit den Namen?, vllt eher in Geschäftslogik teil

Die Angebot- und Antragdruck-Schaltflächen sind zwar immer vorhanden, sind allerdings nicht immer aktiviert, bzw. benutzbar. Diese Logik setzt sich aus zwei Bedingungen zusammen. Zum einem muss überhaupt für den aktuellen Tarif der Antrags- bzw. Angebotsdruck aktiviert sein. Zum anderen ist das Drucken erst möglich, wenn ein Beitrag berechnet werden konnte. Diese Bedingungen werden im Pinia-Store überprüft und bestimmen in den Schaltflächenkomponenten ob sie aktiviert sind oder nicht. Beim Drücken der Buttons wird ein Backend Aufruf an die Drucklogik gestartet. Dieser erwartet die Sessionguid, damit sich das Backend die benötigten Daten aus dem Speicher laden kann. Zurückgeliefert wird ein PDF-Dokument als Filestream. Das wird dann im Frontend aufbereitet, so dass automatisch ein PDF-Dokument heruntergeladen wird. Da die Logik hierbei im Frontend zwischen dem Antrag und Angebot sich nur im aufgerufenem Endpunkt unterscheidet, ist die Logik zusammengefasst und der Aufruf des Backends kann als Action per Parameter reingegeben werden.
In die Fußzeile kommt außerdem noch das Impressum und das Copyright. Das Impressum ist hierbei eine Verlinkung auf das Impressum der Homepage von K&M.

5.3. Implementierung der Geschäftslogik

Die Geschäftslogik, bzw. das Backend ist eine .NET 6.0 API. Als Programmiersprache wird dementsprechend C# 10 benutzt. Das Backend persistiert Daten in einem Storage oder Cache. Da die Kommunikation zwischen dem Backend und den anliegenden Systemen größtenteils über XML läuft, werden die gegebenen XML-Schemata eingelesen und beim Bauen des Backends als C# Klassen generiert. Dadurch können die Daten einheitlich und einfach geparst und ausgelesen wer-

writer lock,cach repository pattern

den. Allerdings kann es trotzdem vorkommen, dass Daten in anderen Datentypen verarbeitet werden. Um diese dann in das besagte XML-Format zu stecken wird der Automapper genutzt, welchen man nur konfigurieren muss um die Daten in der jeweils andere Klasse speichern zu können.

besser formulieren

Das Backend ist im Domain-Driven Design aufgebaut. Dementsprechend gibt es mehrere Services, welche jeweils in verschiedene Projekte aufgeteilt sind. Unter anderem gibt es im Backend Schnittstellen für Anträge, Beitragsberechnung, Tarifkonfigurationen und allgemeine Logik unter Management. Jede dieser Schnittstellen haben nach dem Domain-Driven Design ein eigenes Projekt für die eigentliche Schnittstelle, für die Infrastruktur und für die Logik. Außerdem wird die Logik für die verschiedenen Versicherungstarife aufgeteilt, da sie sich dort unterscheiden kann. Die unterschiedliche Logik der Tarife wird nach Strategy Pattern dynamisch je nach aktivem Tarif benutzt und per Dependency Injection reingegeben. Für die Fußzeilenkomponente ist nur der Managementservice interessant, da dort allgemeingültige und tarifunspezifische Logik zu finden ist.

nicht wirklich strategy? prüfen?

Zum Speichern und Kopieren von Angeboten wird die Session als **guid!** (**guid!**) benötigt. Anhand dieser Sessionguid, wird sich die Konfiguration und die gespeicherten Angebotsdaten aus dem Repositorycache geladen. Sowohl beim Speichern als auch beim Kopieren wird größtenteils die gleiche Logik benutzt. So wird bei beiden Varianten die Sessionspeicherungsschnittstelle des TRGs aufgerufen, welches einen Eintrag in einer Datenbank hinterlegt oder falls schon einer mit dieser Sessionguid vorhanden ist, bearbeitet. Anschließend wird das Angebot im Repositorystorage gespeichert.
Die Besonderheit bei der Kopie ist, dass vor dem Aufruf ans TRG eine neue **guid!** erzeugt wird. Dadurch wird ein neuer Eintrag in die Datenbank geschrieben. Außerdem muss das Angebot mit der neuen Sessionguid auch im Cache gespeichert werden. Die neue Sessionguid wird nun ans Frontend zurückgegeben, damit dort mit der Kopie weitergearbeitet wird.
Bei den Aufrufen für das Drucken von einem Angebot oder Antrag wird die Sessionguid und der Produkttyp, also welcher Tarif, benötigt. Hierbei wird der Pdftoolsservice aufgerufen, eine interne Schnittstelle die PDF-Dateien anhand eines Angebotes, bzw. Antrags als Stream liefert. Vorher muss allerdings, genau wie im Frontend, überprüft werden ob ein PDF überhaupt druckbar ist. Dafür werden die eingegebenen Daten validiert und testweise eine Beitragsberechnung durchgeführt. Erst wenn dies erfolgreich ist, wird ein PDF zurückgeliefert. Da diese Logik, bis auf den Aufruf vom Pdfservice identisch ist, wird der Pdfserviceaufruf per anonyme Funktion als Argument reingegeben.
Da man sich genau diese PDF-Dateien auch per E-Mail zu senden kann, benutzt die Dokumente-Versenden-Logik den gleichen Programmcode wie das Drucken. Allerdings können auch andere Dokumente versendet werden. Diese müssen beim Aufruf der Action mit angegeben werden. Die Dokumente werden dann runter geladen und anschließend wird ein Aufruf an den Mailservice

getätigt, welcher diese Dokumente per angegebene E-Mail versendet.

6. Abnahmephase

7. Einführungsphase

8. Dokumentation

9. Fazit

9.1. Soll-Ist-Vergleich

9.2. Gewonnene Erkenntnisse

9.3. Ausblick

Als neuste Generation der OTRs sollen nun alle neuen Tarife anhand dieser Basis implementiert werden. Außerdem ist geplant mind. einen alten Tarif in die neue Basis zu übertragen.
Für die Fußzeilenkomponente sind noch kleinere Erweiterungen ausstehend, die nicht im Rahmen des Abschlussprojektes umgesetzt werden konnten oder nicht geplant waren. Dazu gehört eine Ladeanimation und Eingabesperre im Frontend, solange auf das Drucken eines Dokuments gewartet wird. Außerdem sind kleinere Refactorings im Backend sowie ein Refactoring der Datenzustandsverwaltung im Frontend geplant.

Literaturverzeichnis

Abbildungsverzeichnis

A. Anhänge