**IPA 2024  
ABSCHLUSSARBEIT**

**OVWEB**

**Validation in insurance offer creation**

**27.03.2024**

**Individuelle Praxisarbeit Kanton Zürich**

**Andrei Mititelu**

**Generali AG**

**TBZ**

**IT Apprentices**

**Aschvin Umachandran**

**Sukhnaaz Singh**

**Roger Sorg**

**Roman Babenko**

**Applikationsentwicklung**

**IPERKA**

**Kandidat / Author:**

**Firma:**

**Berufsfachschule:**

**Abteilung:**

**Hauptexperte:**

**Nebenexperte:**

**Berfusbildner:**

**Verantwortliche Fachkraft:**

**Fachrichtung:**

**Projektvorgehensmodell:**

A red circle with a white letter i in it

Description automatically generated

Teil 1 – Administrativer Teil

1. Projektaufbauorganisation 4

2. Organisation der Arbeitsergebnisse 5

2.1 Codesicherung 5

2.2 Dokumentationssicherung 5

3. Standards 6

4. Zeitplan 9

5. Arbeitsjournal 10

5.1 Tag 1: Mittwoch, 03.04.2024 10

5.2 Tag 2: Donnerstag, 04.04.2024 11

5.3 Tag 2: Freitag, 05.04.2024 11

6. Kurzfassung 13

6.1 Ausgangssituation 13

6.2 Projektvorgehen 13

6.3 Ergebnis 13

7. Projektvorgehen 14

Teil 2 – Individueller Teil Error! Bookmark not defined.

8. Informieren 15

8.1 Ziele der Aufgabenstellung 15

8.2 Vorgaben und Anforderungen 15

8.3 Ist-Zustand 15

8.3.1 Projekt-Überblick 16

8.3.2 Frontend 18

8.3.3 Backend 20

8.3.4 ERM-Datenbankdiagramm 26

8.3.5 ANGEBOTsoftware Rechenkern 27

8.3.6 Systemgrenzen 27

8.4 User Stories und Akzeptanzkriterien 27

8.4.1 Funktionale Anforderungen 27

8.4.2 Nichtfunktionale Anforderungen 28

9. Planen 30

9.1 Soll-Zustand 31

9.1.1 Frontendänderungen 32

9.1.2 Backendänderungen 33

9.2 Mockups 35

9.3 Testkonzept 37

9.3.1 Testziele und Strategie 37

9.3.2 Testumgebung 37

9.3.3 Testcases 38

10. Entscheiden 42

11. Realisieren 44

12. Kontrollieren 45

12.1 Testprotokoll 45

12.2 Testbericht 45

13. Auswerten 46

13.1 Vergleich Ist/Soll 46

13.2 Persönliches Fazit 46

13.2.1 Was lief gut? 46

13.2.2 Was lief schief? 46

13.2.3 Was habe ich aus der Erfahrung gelernt? 46

13.3 Schlussreflexion 46

14. Glossar 46

15. Quellenvezeichnis 46

15.1 Einsatz von generativen Tools 46

16. Danksagungen 46

1. Teil 1 – Administrativer Teil

## Projektaufbauorganisation

**Lehrbetrieb und Durchführungsort**Generali AG   
Soodmattenstrasse 4   
8134, Adliswil   
[058 472 75 00](https://www.google.com/search?client=firefox-b-d&q=generali+adliswil+&lqi=ChFnZW5lcmFsaSBhZGxpc3dpbCIDiAEBSKmIiN3lgICACFofEAAYABgBIhFnZW5lcmFsaSBhZGxpc3dpbCoECAIQAJIBEGluc3VyYW5jZV9hZ2VuY3mqAVcKDS9nLzExYjY2anJoc2QQASoMIghnZW5lcmFsaSgFMh8QASIb7rjLivujVg1VYAHWgXdZ_l1vC9h1_he9K8OnMhUQAiIRZ2VuZXJhbGkgYWRsaXN3aWw)  
[life.ch@generali.com](mailto:life.ch@generali.com)  
  
  
**Hauptexperte**Aschvin Umachandran   
aschvin00@gmail.com   
078 826 61 57  
  
  
**Verantwortliche Fachkraft:**Roman Babenko  
[Generali](mailto:marc.siegmund+pkorg@gmail.com) AG  
Soodmattenstrasse 4  
8134, Adliswil  
076 410 87 29  
roman.babenko@generali.com  
  
  
**Auftraggeber:**Daniel Sager  
Generali AG  
Soodmattenstrasse 4  
8134, Adliswil  
daniel.sager@generali.com

**Kandidat**Andrei Mititelu   
Breitwiesstrasse 15   
8135, Langnau am Albis   
079 343 49 40  
[andrei@gimli.ro](mailto:andrei@gimli.ro)  
  
  
**Nebenexpertin**Sukhnaaz Singh  
sukhnaazsingh@hotmail.com  
076 411 42 35  
  
  
**Berufsbildner / Lehrfirma:**Sorg Roger  
[Generali](mailto:marc.siegmund+pkorg@gmail.com) AG  
Soodmattenstrasse 4  
8134, Adliswil  
058 472 40 42  
roger.sorg@generali.com

## Organisation der Arbeitsergebnisse

Um den Verlust der Dateien im Falle eines Ausfalles zu vermeiden, werden während der IPA verschiedene Versionierungs- und Speicherungssysteme genutzt.   
Mithilfe dieser Systeme kann ebenfalls auf ältere Versionen der Dokumente und Dateien zugegriffen werden.

### Codesicherung

Für den Code wird einen eigenständigen Branch in der Monorepo des Projekts erstellt. Änderungen am Code oder an der Projektdokumentation werden auf diesem Branch gepusht und somit gespeichert.

**Link zum Branch:** https://github.com/IT-Apprentices/insurance-offer-management/tree/OVB-170-Offerten-Vergleich-IPA

Folgend steht die Historie des Branches :

### Dokumentationssicherung

Für die Sicherung aller Dateien für die Dokumentation wird ein separates GitHub Repository erstellt. Auf diesem wird täglich ein Mal gepusht. Ebenfalls werden die Word-Dokumente automatisch auf OneDrive gespeichert.

**Link zum Repository:** https://github.com/Rad14nt/IPA\_2024\_Docs

Folgend steht die Historie des Repositorys :

## Standards

Im Ramen des Projektes OVWEB und dieser IPA werden folgende Technologien eingesetzt:

|  |  |
| --- | --- |
| **Frontend** | |
| Name | Einsatz |
| React 18 | Frontend UI Framework |
| NextJS 14 | React-based Framework |
| Typescript | Frontendsprache |
| Jest | Testing Framework |
| Storybook | Development und Testing Framework |
| Shadcn/ui | Komponenten Library |
| Prettier und ESLint | Code Formattierungs- und Analysetool |
| NextAuth | Authentifizierung |

|  |  |
| --- | --- |
| **Backend** | |
| Name | Einsatz |
| Java | Backendsprache |
| Springboot | Java-Framework |
| Lombok | Annotationen |
| JUnit | Testing |
| Hibernate | ORM-Framework |

|  |  |
| --- | --- |
| **Daten Layer** | |
| Name | Einsatz |
| PostgreSQL | Datenbank |
| Flyway | Datenbank-Versionierungstool |

|  |  |
| --- | --- |
| **Daten Layer** | |
| Name | Einsatz |
| PostgreSQL | Datenbank |
| Flyway | Datenbank-Versionierungstool |

|  |  |
| --- | --- |
| **Identity Provider** | |
| Name | Einsatz |
| OpenID Connect | Identitätsprotokoll |
| Keycloak | Identitäts- und Zugriffsmanagement |

|  |  |
| --- | --- |
| **Entwicklungsumgebung** | |
| Name | Einsatz |
| JetBrains IntelliJ IDEA | Entwicklungsumgebung |
| Docker | Containerisierungsplatform |
| Windows 11 | Operating System |

|  |  |
| --- | --- |
| **UI/UX Design** | |
| Name | Einsatz |
| Figma | Design Tool |
| Generali-Styleguide | Styleguide |

|  |  |
| --- | --- |
| **Dokumentation und Administration** | |
| Name | Einsatz |
| Microsoft Office (Word und Excel) | Dokumentation |
| Draw.io und IntelliJ IDEA | Diagramme bzw. Skizzen |
| OneDrive | Automatische Speicherung |
| Github | Versionierung der Dokumentation und Backups |
| ChatGPT | Textgenerierung und Informationsbeschaffung |
| Duden.de | Rechtschreibprüfung |

|  |  |
| --- | --- |
| **Projektmanagement und Development Tools** | |
| Name | Einsatz |
| Jira | Projektmanagement-Tool |
| Github | Versionskontroll- und Kollaborationsplattform |

## Zeitplan

**A screenshot of a computer

Description automatically generated**

## Arbeitsjournal

### Tag 1: Mittwoch, 03.04.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | * Aufgabenstellung und Anforderung studieren * Zeitplan erstellen * Ist-Zustand Analyse * User Stories und Akzeptanzkriterien definieren * Dokumentation – Teil 1, Administrativer Teil * Dokumentation – Informieren * - Dokumentation – Arbeitsjournal |
| **Probleme** | SendGrid stoppte das Backend vom starten, da mir die Authentifizierungsdaten im .env File fehlten. |
| **Hilfestellung** | Ich habe mein Arbeitskollege für die Authentifizierungsdaten für SendGrid gefragt, da er diese hatte und die Daten nicht online in der Dokumentation waren. |
| **Reflexion** | Ich konnte heute gut mit der IPA starten und kam auch sehr zügig voran, vor allem war die Vorlage besonders hilfreich. Den Zeitplan musste ich jedoch ganz neu gestalten, dies hat ein wenig mehr Zeit in Anspruch genommen als erwartet, aber ich konnte diese Zeit schnell mit dem Ausfüllen des administrativen Teils meines Dokumentes aufholen, bei der ich ein wenig schneller war als erwartet.  Weil ich mir für den Zeitplan jedoch Zeit genommen habe, erlaube dieser mir sehr strukturiert in meiner IPA einzusteigen. Etwas Stress gab es trotzdem, da ich noch am Anfang der IPA bin, ist alles noch ein wenig unsicher.  Ebenfalls hatte ich heute Probleme beim Aufsetzen meines Projektes, ich brauchte diesen aufzustarten, um Screenshots für den IST-Zustand zu machen, hatte aber Probleme mit einer neuen Implementation von SendGrid, ein E-Mail Service, welcher von jemand anders implementiert wurde. Es fehlten die Login Daten im Environment Variables File, und diese waren nicht in der Dokumentation zu finden, daher musste ich meinen Kollegen Samuel fragen, mir diese von seinem Projekt zu kopieren.  Allgemein bin ich aber mit dem heutigen Fortschritt zufrieden und freue mich, dass meine IPA so weit nach Plan läuft. |
| **Nächste Schritte** | * Informierphase abschliessen und fertig dokumentieren. * Planungsphase beginnen: * SOLL-Zustand definieren * Mockups erstellen * Planen dokumentieren |

### Tag 2: Donnerstag, 04.04.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | * User Stories und Akzeptanzkriterien definieren * Dokumentation - Informieren * Soll-Zustand definieren inkl. Diagramme * Mockups erstellen (Figma) * Dokumentation - Planen * Dokumentation - Arbeitsjournal |
| **Reflexion** | Der heutige Tag lief genau nach Plan. Ich fing an mit das Definieren der User Stories und Akzeptanzkriterien. Dannach dauerte das Dokumentieren der Information Phase etwa eine halbe Stunde länger als erwartet. Weil ich mir aber schon Gedanken zur IPA gemacht hatte, konnte ich den Soll-Zustand schneller schreiben als gedacht.  Mittels Beispiele von andere Websiten konnte ich auch meine Mockups zeitgerecht fertig zeichnen und diese dannach auch fast komplett dokumentieren. Für die Dokumentation der Planung habe ich aber noch 2.5 Stunden, also sollte der Rest morgen auch gut machbar sein.  Hilfestellung brauchte ich heute keine und ich hatte auch keine Probleme.  Heute war ein erfolgreicher Tag an dem ich ohne viel Stress gut vorangekommen bin. |
| **Nächste Schritte** | * Model der Datenbank erstellen * Testkonzept erstellen * Planen dokumentieren * 1. Expertengespräch * Lösungsvariante festlegen * Entscheiden dokumentieren |

### Tag 3: Freitag, 05.04.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | * Model der Datenbank erstellen * Testkonzept erstellen * Planen dokumentieren * 1. Expertengespräch * Lösungsvariante festlegen * Entscheiden dokumentieren * Dokumentation - Arbeitsjournal |
| **Ausserplanmässige**  **Arbeiten** | Ich hatte beim anschauen der Dokumentation gemerkt, dass das Response der ANGEBOTsoftware im Kapitel 8.3.3 nicht richtig war, ich musste dieses korrigieren. |
| **Reflexion** | Heute war ein entspannter Tag. Ich konnte schnell den Model der Datenbank erstellen und sprang dann direkt im Testkonzept, der dank meiner Akzeptanzkriterien schnell geschrieben wurde. Bei der Dokumentation, Teil Planen, brauchte ich ein wenig länger Zeit als erwartet, dies war weil ich gemerkt hatte, dass ich in der Dokumentation die falsche Schriftart für ein paar Kapiteln auswählte, ich musste dies korrigieren. Dies konnte ich mir aber leisten, da ich an der meine Lösungsvariante für die Detail Seite eine halbe Stunde schneller fertig hatte als gedacht. Heute war auch das erste Expertengespräch. Ich war recht nervös, aber weil meine Experten sehr freundlich waren konnte ich mich schnell beruhigen. Zuletzt musste ich dann noch die Lösungsvariante in meiner Dokumentation unter dem Kapitel Entscheiden dokumentieren, und schrieb dann zuletzt noch diese Reflexion. |
| **Nächste Schritte** | Realisieren - Backend:   * Datenbank Anpassungen * Anpassungen Model und DTOs * Anpassungen an Controller und Services * Realisieren dokumentieren |

### Tag 4: Montag, 08.04.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | * Datenbank Anpassungen * Anpassungen Model und DTOs * Anpassungen an Controller und Services * Realisieren dokumentieren * Frontend : Anpassungen an der Create-Offer Seite * Dokumentation - Arbeitsjournal |
| **Ausserplanmässige**  **Arbeiten** | * Frontend : Anpassungen an der Create-Offer Seite |
| **Reflexion** | Ich startete heute mit der Realisierugnsphase meiner IPA. Am morgen ging es direkt los mit den Anpassungen an der Datenbank. Damit meine Datenbank mit meine Applikation harmonisiert musste ich dannach Models, DTOs, Mappers und Tests an den neuen Änderungen anpassen. Somit mussten dann auch die Controllers und Services leicht angepasst werden.  In meiner IPA stehen zwei User Aktivitäten im Vordergrund. Die Validierung einer Offerte auf der Erstellungsseite und das Offertenvergleich auf der Detail Seite. Ich implementierte heute die Logik für beide in meinen Backend. Das Error Handling funktioniert jedoch nicht perfekt.  Grund dafür ist eine kleine Abweichung vom Zeitplan.  Nach dem ich die Möglichkeit hinzufügte, eine Offerte mit mehreren Vericherungssummen zu erstellen und diese zu validieren im Backend implementierte, habe ich entschieden, dass es Sinn machen würde, wenn ich die Create-Offer Seite auch schon anpasse. Dies damit ich meine Applikation auch gerade testen konnte, ohne Postman zu benutzen.  Eine kurze Implementation war auch schnell erledigt. Ich fand dann heraus, dass die Implementation tatsächlich auch funktionierte. Somit muss ich das Error Handling im Backend Teil des Projektes aber am Dienstag oder am Mittwoch realisieren.  Zum Schluss implementierte ich auch noch, dass das Backend gleich drei Offerten berechnen kann, mittels Network Tool auf dem Browser konnte ich bestätigen, dass dies auch funtkionierte. Dies war schnell erledigt. Am Schluss vom Tag dokumentierte ich das Erreichte. |
| **Nächste Schritte** | * Anpassungen an Controller und Services * Anpassungen an der Create-Offer Seite * Realisieren dokumentieren |

Teil 2 – Indivudueller Praktischer Teil

## Kurzfassung

### Ausgangssituation

In der Applikation OVWEB kann man schon eine Todesfallversicherung-Offerte erstellen. Jedoch ist ein Vergleich mittels verschiedene Versicherungssummen noch nicht möglich. Ziel dieser IPA ist, dem Berater und dem Kunden einen benutzerfreundlichen Vergleich zu präsentieren und die Erstellung dieser zu ermöglichen.

### Projektvorgehen

Die Realisierung des Projekts orientiert sich an der IPERKA-Methode. Mehr zu dieser Methodik finden Sie unter Kapitel 7.

Nach eine detaillierte Informations- und Planungsphase, während der der IST-Zustand dokumentiert und der SOLL-Zustand definiert werden, wird es möglich mit einem deutlichen Verständnis der Ziele in die Umsetzungsphase zu starten.

In der Realisierungsphase werden Änderungen am Controller und am Service vorgenommen. Um diese Änderungen zu benutzen werden auch das Models, DTOs und die Datenbank angepasst.

Im Frontend Teil des Projektes werden die Detail- und die Erstellungsseiten angepasst. Auf der Erstellungsseite brauch es neue Felder sowie Validation. Die Detailseite muss so angepasst werden, dass sie übersichtlich höchstens drei Offerten gleichzeitig anzeigen kann.

### Ergebnis

## Projektvorgehen

Die IPERKA-Methode ist ein systematischer Ansatz zur Problemlösung und Projektplanung.   
Sie gliedert den Prozess in sechs Phasen:   
Informieren, Planen, Entscheiden, Realisieren, Kontrollieren und Auswerten.   
Diese Methode hilft dabei, komplexe Aufgaben strukturiert anzugehen, Ziele klar zu definieren und die Umsetzung effektiv zu gestalten.

**Informieren**:

In der ersten Phase geht es darum, alle relevanten Informationen zu sammeln und zu bewerten. Dies beinhaltet das Verständnis der Aufgabe, das Identifizieren von notwendigen Ressourcen und das Analysieren des jetzigen Zustandes sowie der Systemumgebung.

**Planen**:

Nachdem alle Informationen gesammelt wurden, wird in dieser Phase ein detaillierter Plan erstellt. Dabei werden Ziele definiert (Soll-Zustand), erforderliche Schritte zur Zielerreichung ausgearbeitet, Diagramme erstellt, Mockups angefertigt und ein Testkonzept festgelegt.

**Entscheiden**:

Basierend auf den gesammelten Informationen und dem erarbeiteten Plan wird eine Entscheidung über die optimale Implementierung getroffen. Die Auswahl erfolgt anhand eines Kriterienkatalogs.

**Realisieren**:

Die Realisierungsphase umfasst die aktive Umsetzung des Plans. In dieser Phase werden die geplanten Schritte ausgeführt und kontinuierlich dokumentiert, um den Fortschritt nachvollziehbar zu machen.

**Kontrollieren**:

Nach der Umsetzung ist es wichtig, den Fortschritt zu überprüfen. Dies geschieht anhand der zuvor definierten Tests, um sicherzustellen, dass alles wie geplant funktioniert und alle Ziele erreicht wurden.

**Auswerten**:

Nach Abschluss der Arbeit wird der gesamte Prozess ausgewertet. Diese Reflexion beinhaltet die Bewertung des erreichten Ergebnisses im Vergleich zu den gesetzten Zielen. Ebenfall werden gesammelten Erfahrungen analyisiert, um sowohl die Stärken aus auch die Verbesserungspotenzialle des Projektes zu bewerten.

## Informieren

### Ziele der Aufgabenstellung

Das Ziel dieser Aufgabe ist es, dem Berater einen Prämienvergleich mit maximal drei Todesfallversicherungsofferten darzustellen. Auch sollte dem Berater eine Fehlermeldung gezeigt werden, falls die eingegebene Daten für die Erstellung der Offerte die Business-Rules verletzen.

### Vorgaben und Anforderungen

Der Berater sollte in der Lage sein, bis zu drei verschiedene Versicherungssummen einzugeben. Auf der Detailseite sollte es dem Berater möglich sein, alle drei Offerten gleichzeitig einzusehen, um diese präsentieren und vergleichen zu können. Die für die Offerte eingegebenen Daten müssen mittels des Rechenkerns der bestehenden Angebotssoftware validiert werden. Eventuell auftretende Fehler sollten im Frontend angezeigt werden, damit der Berater diese einsehen kann.

### Ist-Zustand

Im Moment kann man in der OVWEB Applikation eine Todesfallversicherung mittels Formular erstellen. Diese wird dann im Backend der OVWEB Applikation gespeichert. Im Frotnend wird nach der Erstellung auf der Detailseite mit der Offerten-ID weitergeleitet, wo noch mal ein Call auf die Datenbank gemacht wird, um die Daten mittels Rechenkern der alten ANGEBOTSoftware zu berechnen. Diese Antwort wird dann zum Frontend auf der Detail-Seite weitergeleitet, wo diese angezeigt wird, egal ob ein Fehler oder eine Offerte zurückgegeben wird.

Neben dem gibt es auch eine Übersichtsseite für Offerten, wo alle bisher erstellte Offerten stehen. Wenn man auf diese klickt, wird man wieder auf die Detailseite der spezifischen Offerte weitergeleitet, mittels des gleichen Prozesses wie oben beschrieben.

#### Projekt-Überblick

Das folgende Diagramm beschreibt übersichtlich die OVWEB Applikation so wie die Abhängigkeiten.

A diagram of a software

Description automatically generated

Da die Erstellung einer Offerte schon im jetzigen Zustand kompliziert ist, wurde ein Sequenzdiagramm gezeichnet, um diesen Prozess vereinfacht darzustellen.

A diagram of a company

Description automatically generated with medium confidence

#### Frontend

Hier werden die IPA relevante Seiten dargestellt.

**A white background with red and blue text

Description automatically generatedOffer Dashboard**

**A white paper with red and blue lines

Description automatically generatedOffer Creation**

**A screenshot of a computer

Description automatically generatedOffer Details – Erfolgreiche Berechnung der Offerte**

**Offer Details – Fehlgeschlagene Berechnung der OfferteA screenshot of a computer

Description automatically generated**

#### Backend

Diese Klassendiagramme dienen dem Überblick und dem besseren Verständnis der Systemarchitektur und der Beziehungen bezüglich Offerten zwischen den verschiedenen Klassen.

*Für Offerten gibt es Controller und Services für die Verwaltung der Offerten, sowie auch einen Controller und einen Service für die Berechnung der Offerten.*

Im Rahmen dieser IPA wird nur die Erstellung so wie die Berechnung einer Offerte in Detail betrachtet.

**Controller und Service, Verwaltung der Offerten**

Die Offerten werden nicht berechnet und inklussive der Kundendaten in der Datenbank gespeichert. Grund dafür ist, dass sich die Kundendaten nach der Erstellung der Offerte nicht geändert werden dürfen. Die Berechnung könnte sich jedoch ändern, zum Beispiel wenn in 2025 die Prämien steigen, deswegen wird die Offerte bei jedem Aufruf der Detail Seite neu berechnet, anstatt das Sie gespeichert wird.

A screenshot of a computer screen

Description automatically generated

**Model und DTOs, Verwaltung der Offerten**

Diesen DTO benutzen wir, um die im Frontend eingegebenen Daten mittels Controller für die Offerte zu erfassen. Das Frontend gibt nur den Kunden ID weiter, nicht den ganzen Kunden.

A screenshot of a computer program

Description automatically generated

Die Kunden ID, so wie das Keycloak ID des eingeloggten Beraters werden in separaten Variabeln gesetzt und dann dem Service mit dem obigen Offertendaten aus dem DTO weitergeleitet, dort werden die Kundendaten mittels obig genannten ID aus der Datenbank herausgesucht. Ebenfalls wird die ID des Beraters aus der Sponsor Tabelle mittels Keycloak ID herausgesucht.

A screenshot of a computer

Description automatically generatedZusammen mit den Offertendaten aus dem DTO und dem Sponsor ID des Beraters, werden die Kundendaten in diesem Model gemappt.

Diese Daten werden dann mittels Repository in der Datenbank abgespeichert, und das automatisch generierte Offerten ID wird zurückgegeben und am Frontend weitergeleitet. Dort wird mittels dieses ID‘s die entsprechende Detail Seite aufgerufen.

Die Detail Seite ruft mit der Offerten ID das Endpoint auf, welches die Offerte mittels ANGEBOTSoftware berechnet. Diese Berechnung kann bei Verletzung der Business-Rules im Moment auch einen Fehler zurückgeben, wie im Kapitel 8.3.2 abgebildet.

**Controller und Service, Berechnung der Offerte**

A screenshot of a computer

Description automatically generated

Im Controller wird der Request mit der Path Variable ID der Offerte empfangen. Mittels dieser ID wird dann die Calculate Offer Methode im Service aufgerufen.

Dort werden in der im Repository abgespeicherten Daten wieder ausgesucht, diese werden mittels der mapToFosoftRequest Methode so angepasst, dass sie die richtige Struktur für die ANGEBOTSoftware API haben.

Folgend die Struktur des Requests und des Responses.

**Request:**

A screenshot of a computer code

Description automatically generated

A white background with black text

Description automatically generated**Response:**

Diese Response könnte jedoch auch ein Fehler zurückweisen, dies würde wie folgend aussehen:

A computer screen shot of a message

Description automatically generated

Diese Responses werden dann durch den OVWEB Backend wieder am Frontend als Antwort zurückgewiesen, wo sie für den Berater dargestellt werden.

#### ERM-Datenbankdiagramm

A screenshot of a computer

Description automatically generated

#### ANGEBOTsoftware Rechenkern

Das ANGEBOTsoftware ist eine bestehende Applikation und kann somit nicht geändert bzw. angepasst werden. Wir greifen auf dem **Endpoint D2** zu, welches für die Berechnung und Validierung der Todesfallversicherungen zuständig ist. Die Request und Response Daten sind in Kapitel 8.3.3 dokumentiert.

#### Systemgrenzen

Im Rahmen dieser Arbeit liegt der Fokus auf die Erstellung, Berechnung, Validierung und Anzeige mehrer Todesfallversicherungsofferten. Daher gehört das ANGEBOTSoftware, spezifisch das Rechenkern davon, als Abhängigkeit zu dem Projekt.

Es ist wichtig zu erwähnen, dass SendGrid, welches für E-Mails implementiert ist, und Keycloak, nicht Teil der Systemgrenzen dieser spezifischen Aufgabe sind.

### User Stories und Akzeptanzkriterien

Im folgenden Kapitel werden die Anforderung aus der Aufgabenstellung aufgeschlüsselt und genauer analysiert. Dies geschieht mit dem Definieren der User Stories, welche durch Akzeptanzkriterien überprüft werden können.

#### Funktionale Anforderungen

|  |
| --- |
| **User Story 1: Erstellung einer Offerte mit mehreren Versicherungssummen** |
| **Als** Versicherungsberater  **Möchte** **ich** eine Offerte mit maximal drei Versicherungssummen hinzufügen können,  **um** für meinem Kunden einen Vergleich erstellen zu können. |
| **Akzeptanzkriterien** |
| **AC 1.1:** Ein Berater findet auf der Offer Erstellungsseite drei Felder für die Versicherungssumme. |
| **AC 1.2:** Ein Berater kann bis zu drei Versicherungssummen eingeben. |

|  |
| --- |
| **User Story 2: Offertenvergleich** |
| **Als** Versicherungsberater  **Möchte** **ich** einen Vergleich zwischen den vorher erstellten Offerten sehen,  **um** diese mit dem Kunden zu vergleichen. |
| **Akzeptanzkriterien** |
| **AC 2.1:** Falls der Berater eine Offerte mit mehrern Versicherungssummen erstellt hat, sollte er eine Übersicht über die drei Berechnungen haben, wo er diese mit dem Kunden vergleichen kann. |

|  |
| --- |
| **User Story 3: Offerten-Validierung** |
| **Als** Versicherungsberater  **Möchte** **ich,** dass ich keine fehlerhaften Offerten erstellen kann,  **damit** ich dem Kunden vor Ort eine Offerte erstellen kann. |
| **Akzeptanzkriterien** |
| **AC 3.1:** Falls ein Berater eine Offerte mit fehlerhaften Daten erstellt, sollten diese nicht abgespeichert werden. |
| **AC 3.2:** Falls ein Berater eine Offerte mit fehlerhaften Daten erstellt, sollte der Berater nicht auf der Detail Seite weitergeleitet werden. |
| **AC 3.3:** Falls ein Berater eine Offerte mit fehlerhaften Daten erstellt, sollte eine Benutzerinterkation stattfinden, wo beschrieben wird, was mit der Offerte falsch ist. |

#### Nichtfunktionale Anforderungen

|  |
| --- |
| **User Story 4: Benutzerfreundlichkeit der UI** |
| **Als** Versicherungsberater  **Möchte** **ich** das meine Offerten einfach miteinander zu vergleichen sind,  **damit** der Kunde diese auch verstehen kann. |
| **Akzeptanzkriterien** |
| **AC 4.1:** Die verschiedene Offerten sind einfach zu vergleichen. Das heisst, dass die wichtigen Daten so wie die Prämie auf der gleichen Höhe stehen und klar dargestellt sind. |
| **AC 4.2:** Fehlermeldungen bei der Erstellung einer Offerte sind klar dargestellt und einfach zu verstehen. |

|  |
| --- |
| **User Story 5: Observability-Logging** |
| **Als** Systemadministrator  **Möchte** **ich** detaillierte Logs über fehlgeschlagene Offertenerstellungen so wie Berechnungen erhalten,  **um** mögliche Probleme bei der Erstellung so wie bei der Berechnung analysieren und beheben zu können. |
| **Akzeptanzkriterien** |
| **AC 5.1:** Das Logging innerhalb des Systems muss strukturiert folgen. Die log Nachrichten haben ein einheitliches Format und relevanten Daten sind aufgezeichnet. |
| **AC 5.2:** Die Implementierung des Loggings ist durchgängig über alle Komponenten des Systems gewährleistet, damit die Logs auch nachvollziehbar sind. |
| **AC 5.3:** Alle Logs enthalten die notwendigen Informationen, um das Verhalten des Systems zu verstehen und bei Bedarf Fehlersuche vornehmen zu können. |
| **AC 5.4:** Sensible Daten, insbesondere Passwörter und personenbezogene Daten,  werden zu keinem Zeitpunkt in den Logdateien wiedergegeben. |

|  |
| --- |
| **User Story 6: Verbesserung der Nachverfolgbarkeit durch Trace-IDs** |
| **Als** Entwickler  **Möchte ich,** dass ich jede Systemanfrage mittels einer einzigartigen Trace-ID in Grafana  zurückverfolgt werden kann,  **um** die Fehlerbehebung zu ermöglichen. |
| **Akzeptanzkriterien** |
| **AC 6.1:** Für jede Anfrage im Frontend wird eine Trace-ID generiert und wird am Backend weitergegeben. |
| **AC 6.2:** Die generierte Trace-ID wird konsequent in den Logs aller Dienste, die an  der Bearbeitung der Anfrage beteiligt sind, aufgezeichnet. Dies erleichtert die  Nachverfolgung und Diagnose durch das Logging. |
| **AC 6.3:** Auf Basis der durchgängigen Trace-IDs ist es möglich, ein detailliertes  Diagramm der Service-Aufrufe zu erstellen, welches die Interaktionen und den  Datenfluss zwischen den verschiedenen Diensten visualisiert. |

|  |
| --- |
| **User Story 7: Testabdeckung für neue OVWEB-Systemfunktionen** |
| **Als** Entwickler  **Möchte ich,** dass jede neue Funktion sowohl im Frontend als auch im Backend durch automatisierte Tests abgedeckt ist,  **damit** die Zuverlässigkeit und Funktionalität des Systems gewährleistet sind und  Fehler frühzeitig erkannt und behoben werden können. |
| **Akzeptanzkriterien** |
| **AC 7.1:** Die Testfälle decken positiver und  negativer Szenarien so wie Tests mit Null- und Nicht-Null-Skalarwerten. |
| **AC 7.2:** Es werden keine Integrationstests für neue Funktionen implementiert, die effektiv und effizient durch Unit-Tests getestet werden können. |
| **AC 7.3:** Es müssen alle neu implementierte Funtkionalitäten getestet werden, sowohl im Frontend als auch im Backend. |

|  |
| --- |
| **User Story 8: Modularität und Wiederverwendbarkeit der UI-Komponenten** |
| **Als** Entwickler  Möchte ich, dass die neuen Komponenten des Systems modular  aufgebaut und dokumentiert sind,  **damit** sie wiederverwendbar sind. |
| **Akzeptanzkriterien** |
| **AC 8.1:** Komplexe Komponente sind aus kleinere, wiederverwendbare Komponente zusammengebaut. |
| **AC 8.2:** Diese Elemente müssen in der Storybook dokumentiert werden. |

## Planen

### Soll-Zustand

Der Soll-Zustand sollte es dem Berater erlauben, maximal drei Versicherungssummen einzugeben, um einen Vergleich der Prämien dem Kunden zu präsentieren.

Ebenfalls sollte eine Validierung bei der Erstellung dieser neue Offerte passieren.

Um dies zu erreichen werden grösstenteils Komponente verändert und verwendet, die schon existieren, es werden aber auch neue erstellt.

Damit das neue Prozess einfacher zu verstehen ist, wurde ein das im Kapitel 8.3.1 gezeichnete Diagramm mit den neuen geplannten Änderungen angepasst.

A close-up of a document

Description automatically generated

#### 

#### Frontendänderungen

Um die Änderungen visuell zu sehen, finden Sie in Kapitel 9.2 die dazugehörige Mockups.

**Offer Creation Page:**

* Anpassungen an der Form um zwei neue Versicherungssumme Felder hinzuzufügen
* Das Empfangen eines Fehlers muss möglich sein
* Das Anzeigen des Fehlers muss implementiert werden

**Offer Detail Page:**

* Ermöglichen maximal drei Offerten auf einmal zu empfangen (in einem Response)
* Diese Drei Offerten gleichzeitig anzeigen
* Neue UI Elemente müssen einfach zu lesen und zu vergleichen sein so wie auch benutzerfreundlich

**Storybook Anpassungen**

**Tracing**

#### Backendänderungen

**Datenbankänderungen**

* Es braucht zwei neue Felder für die neuen Versicherungssummen in der Offerten Tabelle, dies wird durch eine Flyway Migration ausgeführ

**A white background with black text

Description automatically generated**

**Controller- und Serviceänderungen**

* Die bestehende Services und Controllers müssen angepasst werden, so dass er die Offerte zuerst validiert, bevor er diese abgespeichert wird
* Es braucht eine Funktion im Service, um die berechnete Offerte zusammen zu einem Response zu führen

**Model und DTO Änderungen**

* Die Models und DTOs müssen so angepasst warden, dass sie drei Versicherungssummen handeln können
* Es braucht ein neues DTO, um die berechnete, zusammengeführte Offerten zu verschicken

**Tracing und Logging**

**Unittests**

### Mockups

A screenshot of a computer

Description automatically generated**Offer Creation**

A screenshot of a computer

Description automatically generated**Offer Creation – Fehlermeldung**

A screenshot of a computer

Description automatically generated**Offer Details – Variante 1**

A screenshot of a computer

Description automatically generated**Offer Details – Variante 2**

### Testkonzept

Das Testkonzept beschreibt eine umfassende Teststrategie für dieses Projekt, welche sowohl manuelle als auch automatisierte Testverfahren beinhaltet. Die Implementierung dieser Tests findet in der Phase "Kontrollieren" nach der IPERKA-Methode statt. Dadurch wird am Ende des Entwicklungszyklus eine ganzheitliche Evaluation ermöglicht. Diese Vorgehensweise stellt sicher, dass die Projektziele erreicht und die Anforderungen gemäss der Aufgabenstellung korrekt umgesetzt wurden.

#### Testziele und Strategie

Das Ziel des Testprozesses ist, die Funktionsfähigkeit, Sicherheit, Benutzerfreundlichkeit und Zuverlässigkeit der neuen Implementierungen zu gewährleisten. Dabei wird geprüft, ob die Software die in den User Stories und Akzeptanzkriterien festgelegten Anforderungen erfüllt. Zusätzlich wird untersucht, ob sie die funktionale so wie auch die nicht-funktionalen Anforderungen und spezifischen Kriterien erfüllt sind.

Es wird ebenfalls getestet, damit Fehler und Sicherheitslücken vor der Inbetriebnahme erkannt werden können.

Um zu testen werden wo möglich automatisierte Tests benutzt. Im Frontend mittels JEST und im Backend mit Unit-Tests.   
Fallls dies nicht möglich ist, werden basierend auf die User Stories und Akzeptanzkriterien manuelle Tests eingesetzt.

#### Testumgebung

**Gerät:** Lenovo ThinkBook 15 G4 ABA

**Betriebssystem:** Windows 11 Pro

**Systemtyp:** 64-Bit

**Prozessor:** AMD Ryzen 7 5825U with Radeon Graphics (2.00 GHz)

**RAM:** 16 GB 3200 MHz

**Browser:** Chrome Version 122.0.6261.128 (Offizieller Build) (64-Bit)

**URL:** Frontend: <http://localhost:3000>

Backend: <http://localhost:8080>

#### Testcases

|  |  |
| --- | --- |
| **Testcase 1** | |
| **Anforderung:** | Story 1, AC 1.1 – 1.2 |
| **Beschreibung** | Berater kann eine Offerte mit bis zu 3 Versicherungssummen erstellen. |
| **Voraussetzung** | Der Berater ist eingeloggt und hat auf der Offerten Erstellungsseite navigiert. |
| **Schritte** | Berater füllt das Formular aus, und gibt eine bis drei Versicherungssummen an. |
| **Erwartetes Resultat** | Die Applikation validiert die Eingaben (Frontend-Seitig) |

|  |  |
| --- | --- |
| **Testcase 2** | |
| **Anforderung:** | Story 2, AC 2.1 |
| **Beschreibung** | Offerten vergleichen |
| **Voraussetzung** | Der Berater ist eingeloggt und hat entweder eine Offerte mit zwei oder drei Versicherungssummen erstellt. |
| **Schritte** | Nach der erfolgreichen Erstellung einer neuen Offerte wird der Berater auf die Detail Seite weitergeleitet. Ebenfalls kann von der Offerten-Dashboardseite aus auf eine Offerte klicken, um auf die Detail Seite zu kommen. |
| **Erwartetes Resultat** | Die Offerte wird geholt. Falls der Benutzer zwei oder drei Versicherungssummen angegeben hat, wird ein Bundle von berechneten Offerten zum Vergleich dargestellt. Falls nur eine Versicherungssumme eingegeben wurde, wird nur eine berechnete Offerte angezeigt. |

|  |  |
| --- | --- |
| **Testcase 3** | |
| **Anforderung:** | Story 3, AC 2.1 – 2.3 |
| **Beschreibung** | Offerten Erstellung Validierung |
| **Voraussetzung** | Der Berater ist eingeloggt und hat auf der Offerten Erstellungsseite validiert und die Daten eingefüllt. |
| **Schritte** | Der Berater füllt die Felder mit Daten ein und versucht die Offerte mittels dem Speichern Knopf zu erstellen. |
| **Erwartetes Resultat** | Falls die Offerte die Business-Rules verletzt:   * Die Offerte wird nicht in der Datenbank abgespeichert * Der Berater wird nicht auf die Detail Seite weitergeleitet wie zuvor * Dem Berater wird das von der ANGEBOTSoftware angegebene Fehler angezeigt |

|  |  |
| --- | --- |
| **Testcase 4** | |
| **Anforderung:** | Story 4, AC 4.1 – 4.2 |
| **Beschreibung** | Benutzerfreundlichkeit der Detail Seite |
| **Voraussetzung** | Der Berater ist eingeloggt und hat eine Offerte erfolgreich erstellt. |
| **Schritte** | Der Berater erstellt erfolgreich eine neue Offerte oder navigiert auf eine mittels der Tabelle auf der Übersichtsseite. |
| **Erwartetes Resultat** | Die Offerte und deren Daten werden klar dargestellt. Die Prämie steht im Vordergrund. Falls mehrere Versicherungssummen für die Offerte eingegeben wurden, werden die berechneten Offerten nebeneinander und einfach zum vergleichen aufgezeichnet. |

|  |  |
| --- | --- |
| **Testcase 5** | |
| **Anforderung:** | Story 5, AC 5.1 – 5.4 |
| **Beschreibung** | Implementierung Observability-Logging |
| **Voraussetzung** | Es werden Aktionen im OVWEB System ausgeführt. |
| **Schritte** | Überprüfen der Existenz von Log Dateien. |
| **Erwartetes Resultat** | Die Logs enthalten strukturierte und relevante Informationen.  Sensible Daten werden nicht geloggt. |

|  |  |
| --- | --- |
| **Testcase 6** | |
| **Anforderung:** | Story 6, AC 6.1 – 6.3 |
| **Beschreibung** | Sicherstellung der Nachverfolgbarkeit durch Trace-ID’s in Grafana. |
| **Voraussetzung** | Trace ID’s wurden sowohl im Backend als auch im Frontend richtig und überall wo nötig eingesetzt.  Eine Anfrage wird ausgeführt. |
| **Schritte** | Überprüfen der Traces in Grafana Web Interface. |
| **Erwartetes Resultat** | Jede Anfrage sollte mit einer einzigartigen Trace-ID erstellt werden, die durch alle Services propagiert werden soll In Grafana sollte ein Baum den Prozess der ausgeführten Methoden veranschaulichen. |

|  |  |
| --- | --- |
| **Testcase 7** | |
| **Anforderung:** | Story 7, AC 7.1 – 7.3 |
| **Beschreibung** | Überprüfung der Testabdeckung und der Effektivität der Tests für die neu implementierte Funtkionen, sowohl im Frontend als auch im Backend. |
| **Voraussetzung** | Neue Funktionen wurden hinzugefügt und Tests wurden geschrieben. |
| **Schritte** | Durchführung der neu erstellten Tests. |
| **Erwartetes Resultat** | Die Tesfälle decken alle neue Funktionen ab. UnitTests im Backend und JEST Tests im Frontend. Die Resultate der Tests werden dokumentiert. |

|  |  |
| --- | --- |
| **Testcase 8** | |
| **Anforderung:** | Story 8, AC 8.1 – 8.2 |
| **Beschreibung** | Überprüfung der Modularität und Dokumentation der UI  Komponenten im Storybook. |
| **Voraussetzung** | Es wurden neue UI Komponenten erstellt. |
| **Schritte** | * Inspektion der Struktur der UI-Komponenten im Storybook. * Überprüfung der Zusammensetzung komplexerer Komponenten aus einfacheren Bausteinen. * Überprüfung der Dokumentation der UI-Komponenten im Storybook auf Vollständigkeit und Verständlichkeit. |
| **Erwartetes Resultat** | * Komplexere UI-Komponenten sind modular aus einfacheren Komponenten aufgebaut. * Alle UI-Komponenten sind verständlich im Storybook dokumentiert, was ihre Wiederverwendbarkeit und Anpassbarkeit unterstützt. |

### Kriterien für die Entscheidungsphase

In diesem Kapitel werden die Kriterien für die Entscheidungsphase definiert. Es wird entschieden werden, welches der zwei Varianten der Offer Details Seite im Kapitel 9.2 implementiert werden sollte. Jede Variante sollte eine Note zwischen 1 und 5 für jedes Kriterium bekommen.

|  |  |  |
| --- | --- | --- |
| **Kriterium** | **Gewichtung** | **Begründung** |
| **GUI Design** | 0.4 | Die Benutzbarkeit steht im Fordergrund, da dies einer Kriterium der IPA ist. Es ist wichtig, dass die Prämien zum Vergleich im Vordergrund stehen. |
| **Komplexität** | 0.3 | Wenn die Implementation einfacher ist, bleibt mehr Zeit für andere Aufgaben. |
| **Ästhetik** | 0.3 | Die Ästhetik spielt nicht die grösste Role, daher wird sie am wenigsten gewichtet. Sie ist aber trotzdem wichtig für eine gute Benutzererfahrung und Klarheit. |

## Entscheiden

Es wird zwischen den zwei Varianten für das Design der Detail Seite im Kapitel 9.2 entschieden.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Variante 1** | **Variante 2** |
| **Kriterium** | **Gewichtung** | **Note** | **Note** |
| Benutzerfreundlichkeit | 0.4 | 5.5 | 4.5 |
| Komplexität | 0.3 | 4 | 4 |
| Ästhetik | 0.3 | 5.5 | 4.5 |
| **Note:** | | **5.1** | **4.4** |

**Begründung der Noten**

Benutzerfreundlichkeit:

In der Variante 1 steht die Prämie mehr im Vordergrund, da alle andere Daten viel kleiner sind. In der 2. Variante steht sie zwar auch im Vordergrund, da die andere Daten grösser sind und die Tabelle auch ein wenig ablenkt, könnte es sein, dass man nicht zuerst auf die Prämie schaut.

Komplexität:

Beide Varianten sind relativ komplex umzusetzen. Die erste weil es schwierig sein kann, die Elemente richtig zu positionieren.

Weil wir Tabellen vom Componenten Library Shadcn UI benutzen, kann es aus Erfahrung recht schwierig sein Variante 2 umzusetzen.

Ästhetik:

Die Ästhetik der ersten Variante ist viel schöner und moderner. Somit wird sie besser bewertet als Variante 2.

## Realisieren

### Datenbank

Es wurden wenige Änderungen an der Datenbank vorgenommen.   
Wie im vorherigen Kapitel geplannt, mussten zwei neue Felder in der life\_insurance\_request Tabelle hinzugefügt werden. Ebenfalls wurde das bisher existierende Feld für Klarheit von VS auf VS1 umbenannt. Dies wurde durch löschen und neuerstellung des Columns erreicht.

Für das Update der Datenbank wurde eine Flyway Migration benutzt. Das File musste richtig benannt werden:

<Prefix><Version>\_\_<Description>.sql

In dem Fall wurde das File auf V1\_0\_17\_\_extend\_life\_insurance\_request\_entity.sql genannt.

Durch folgendes Code wurde die Migration dann fehlerfrei ausgeführt.

ALTER TABLE if exists "life\_insurance\_request"  
DROP COLUMN vs;  
  
ALTER TABLE if exists "life\_insurance\_request"  
 ADD COLUMN vs1 INT,  
 ADD COLUMN vs2 INT,  
 ADD COLUMN vs3 INT;

### Backend

#### Änderungen an Models und DTOs

Wegen den Änderungen an der Datenbank und um die neue Funktionalitäten umzusetzen, mussten existente Models und DTOs angepasst werden. Die bestehende vs Variable musste mit eine vs1 Variable getauscht werden, und es mussten zwei neue Variabeln, vs1 und vs2 hinzugefügt werden. Dies musste in LifeInsruanceRequestPostDto und in LifeInsuranceRequestWithCustomerInfoGetDto gemacht werden.

Im Model LifeInsuranceRequestWithCustomerInfo musste diese Änderung ebenfalls vorgenommen werden.

Damit die Applikation noch läuft mussten auch sämtliche Tests angepasst werden. Das DataPopulationTest, welches für die Population der Datenbank zuständig ist, musste angepasst werden, es musste das Feld vs mit vs1 ersetzt werden. Das DtoMapperTest benötigte die gleichen Änderungen.

### Controller und Services

### Frontend

### Tracing

### Observability Logging

## Kontrollieren

### Testprotokoll

### Testbericht

Beschreiben was bewusst nicht getestet wurde.

## Auswerten

### Vergleich Ist/Soll

### Persönliches Fazit

#### Was lief gut?

#### Was lief schief?

#### Was habe ich aus der Erfahrung gelernt?

### Schlussreflexion

## Glossar

## Quellenvezeichnis

Flyway db migration naming convention

[Database Migrations with Flyway | Baeldung](https://www.baeldung.com/database-migrations-with-flyway)

Rsc show message always

<https://stackoverflow.com/questions/62089792/how-to-display-error-message-into-a-json-using-responsestatusexception-on-spring>

https://tailwindcss.com/docs/

### Einsatz von generativen Tools

## Danksagungen