**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 3

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

6. Kurzfassung 12

6.1 Ausgangssituation 12

6.2 Projektvorgehen 12

6.3 Ergebnis 12

7. Projektvorgehen 13

8. Informieren 14

8.1 Ziele der Aufgabenstellung 14

8.2 Vorgaben und Anforderungen 14

8.3 Ist-Zustand 14

8.3.1 Projekt-Überblick 14

8.3.2 Backend 14

8.3.3 Frontend 14

8.3.4 ANGEBOTsoftware Rechenkern 14

8.3.5 Systemgrenzen 14

8.4 User Stories und Akzeptanzkriterien 14

8.4.1 Funktionale Anforderungen 14

8.4.2 Nichtfunktionale Anforderungen 14

9. Planen 14

9.1 Soll-Zustand 14

9.2 Mockups 15

9.3 Testkonzept 15

10. Entscheiden 15

11. Realisieren 15

12. Kontrollieren 15

13. Auswerten 15

13.1 Vergleich Ist/Soll 15

13.2 Persönliches Fazit 15

13.2.1 Was lief gut? 15

13.2.2 Was lief schief? 15

13.2.3 Was habe ich aus der Erfahrung gelernt? 15

13.3 Schlussreflexion 15

14. Glossar 15

15. Quellenvezeichnis 15

15.1 Einsatz von generativen Tools 15

16. Danksagungen 16

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 2: Freitag, 05.04.2024

|  |  |
| --- | --- |
| **Tätigkeiten** |  |
| **Erreichte Ziele** |  |
| **Probleme** |  |
| **Hilfestellung** |  |
| **Ausserplanmässige**  **Arbeiten** |  |
| **Überstunden** |  |
| **Reflexion** |  |
| **Nächste Schritte** |  |

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, ist es möglich mit einem deutlichen Verständnis der Ziele in die Umsetzungsphase zu starten.

In der Realisierungsphase wurden 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 wird 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.

A computer screen shot of a diagram

Description automatically generated

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 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

**Response:**

A screenshot of a computer code

Description automatically generated

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: Weiterleitung auf die Offerten-Erstellungsseite (bereits umgesetzt)** |
| **Als** Versicherungsberater  **Möchte** **ich** beim Betätigen des Create Offer Knopfes auf die Offerten-Erstellungsseite weitergeleitet werden,  **um** eine Offerte erstellen zu können. |
| **Akzeptanzkriterien** |
| **AC 1.1:** Ein Berater findet auf die Offerten Übersichtsseite ein Knopf für die Erstellung einer neuen Offerte. |
| **AC 1.2:** Beim Anclicken des Knofpes wird auf die Offerten-Erstellungsseite weitergeleitet. |

|  |
| --- |
| **User Story 2: Offerte individualisieren (bereits umgesetzt)** |
| **Als** Versicherungsberater  **Möchte** **ich** die Daten für eine neue Offerte individualisieren können,  **um** meinem Kunden eine personalisierte Offerte zu berechnen. |
| **Akzeptanzkriterien** |
| **AC 2.1:** Der Berater hat die Möglichkeit, bei der Erstellung einer Offerte einen Kunden auszuwählen, den er schon erstellt hat. |
| **AC 2.2:** Der Berater kann die Versicherungssumme eingeben. |
| **AC 2.3:** Ein Berater kann die Dauer eingeben. |
| **AC 2.4:** Ein Berater kann die PZA eingeben. |
| **AC 2.4:** Ein Berater kann die Versicherungsart eingeben |
| **AC 2.4:** Ein Berater kann zwischen eine Offerte mit Prämienbefreiung und eine ohne. auswählen. |
| **AC 2.4:** Ein Berater kann zwischen den Säulen 3a oder 3b wählen. |

|  |
| --- |
| **User Story 3: Erstellung einer Offerte mit mehreren Versicherungssummen** |
| **Als** Versicherungsberater  **Möchte** **ich** eine Offerte mit maximal drei Versicherungssummen erstellen können,  **um** für meinem Kunden einen Vergleich erstellen zu können. |
| **Akzeptanzkriterien** |
| **AC 3.1:** Ein Berater findet auf der Offer Creation Seite drei Felder für die Versicherungssumme. |
| **AC 3.2:** Ein Berater kann mehrere Versicherungssummen eingeben. |

|  |
| --- |
| **User Story 4: Offertenvergleich** |
| **Als** Versicherungsberater  **Möchte** **ich** einen Vergleich zwischen den vorher erstellten Offerten sehen,  **um** diese mit dem Kunden zu vergleichen. |
| **Akzeptanzkriterien** |
| **AC 4.1:** 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 5: Offertenvergleich** |
| **Als** Versicherungsberater  **Möchte** **ich,** dass ich keine fehlerhaften Offerten erstellen kann,  **damit** ich dem Kunden vor Ort eine Offerte erstellen kann. |
| **Akzeptanzkriterien** |
| **AC 5.1:** Falls ein Berater eine Offerte mit fehlerhaften Daten erstellt, sollten diese nicht abgespeichert werden. |
| **AC 5.2:** Falls ein Berater eine Offerte mit fehlerhaften Daten erstellt, sollte der Berater nicht auf der Detail Seite weitergeleitet werden. |
| **AC 5.2:** 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 6: Benutzerfreundlichkeit der UI** |
| **Als** Versicherungsberater  **Möchte** **ich** das meine Offerten einfach miteinander zu vergleichen sind,  **damit** der Kunde diese auch verstehen kann. |
| **Akzeptanzkriterien** |
| **AC 6.1:** 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 6.2:** Fehlermeldungen bei der Erstellung einer Offerte sind klar dargestellt und einfach zu verstehen. |

|  |
| --- |
| **User Story 7: 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 7.1:** Das Logging innerhalb des Systems muss strukturiert folgen. Die log Nachrichten haben ein einheitliches Format und relevanten Daten sind aufgezeichnet. |
| **AC 7.2:** Die Implementierung des Loggings ist durchgängig über alle Komponenten des Systems gewährleistet, damit die Logs auch nachvollziehbar sind. |
| **AC 7.3:** Alle Logs enthalten die notwendigen Informationen, um das Verhalten des Systems zu verstehen und bei Bedarf Fehlersuche vornehmen zu können. |
| **AC 7.4:** Sensible Daten, insbesondere Passwörter und personenbezogene Daten,  werden zu keinem Zeitpunkt in den Logdateien wiedergegeben. |

|  |
| --- |
| **User Story 8: Verbesserung der Nachverfolgbarkeit durch Trace-IDs** |
| **Als** Entwickler  **Möchte ich,** dass ich jede Systemanfrage mittels einer einzigartigen Trace-ID  zurückverfolgen kann,  **um** die Diagnose bei Problemen zu erleichtern. |
| **Akzeptanzkriterien** |
| **AC 8.1:** Für jede Nutzeranfrage im Frontend wird systematisch eine Trace-ID  generiert und für die Verarbeitung nahtlos an das Backend übermittelt. Diese  Identifikation ermöglicht es, den Weg der Anfrage durch das System  nachzuverfolgen. |
| **AC 8.2:** 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 8.3:** 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 9: Sicherstellung der Testabdeckung für OVWEB-Systemfunktionen** |
| **Als** Entwickler  **Möchte ich,** dass jede neu implementierte Funktion im OVWEB-System durch  umfangreiche automatische Tests abgesichert wird,  **damit** die Zuverlässigkeit und Funktionalität des Systems gewährleistet sind und  Fehler frühzeitig erkannt und behoben werden können, was zu einer höheren  Softwarequalität führt. |
| **Akzeptanzkriterien** |
| **AC 9.1:** Die Testfälle decken ein breites Spektrum ab, einschliesslich positiver und  negativer Szenarien, Tests mit Null- und Nicht-Null-Skalarwerten sowie leeren und  nicht-leeren Sammlungen (Collections), um die Vielfalt der möglichen Eingaben und  Zustände gründlich zu prüfen. |
| **AC 9.2:** Es werden keine IT-Tests  (Integrationstests) für Funktionen implementiert, die effektiv und effizient durch  Unit-Tests abgedeckt werden können. |
| **AC 9.3:** Für jede neu eingeführte Funktionalität werden sowohl im Frontend als auch  im Backend automatisierte Tests entwickelt und integriert, um eine durchgängige  Qualitätssicherung zu gewährleisten. |

|  |
| --- |
| **User Story 10: Modularität und Wiederverwendbarkeit der UI-Komponenten** |
| **Als** Entwickler  Möchte ich, dass die Benutzeroberflächenkomponenten des Systems modular  aufgebaut und dokumentiert sind,  **Um** ihre Wiederverwendung und Wartbarkeit zu fördern. |
| **Akzeptanzkriterien** |
| **AC 10.1:** Die Architektur der UI-Komponenten ermöglicht den Aufbau komplexer Elemente aus einfacheren Komponenten. |
| **AC 10.2:** Eine Dokumentation dieser Elemente ist durch Storybook zugänglich, um ihre Verwendungsmöglichkeiten klar darzustellen. |

## 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 gezeichnetes 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 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 so wie benutzerfreundlich

**Storybook Anpassungen**

**Tracing**

#### Backendänderungen

**Datenbankänderungen**

* Es braucht zwei neue Felder für die neuen Versicherungssummen, dies wird durch eine Flyway Migration ausgeführt

**Controller- und Serviceänderungen**

* Die bestehende Services und Controllers müssen angepasst werden, so dass er die Offerte zuerst validiert, bevor er diese abspeichert wird
* Die Mappers müssen angepasst werden, um drei Offerten auf einmal handeln zu können
* Es braucht eine Funktion, 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 generatedOffer Creation

Offer Details

A screenshot of a computer

Description automatically generated

### Testkonzept

## Entscheiden

Entscheidmatrix fuer Design

## Realisieren

## 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

### Einsatz von generativen Tools

## Danksagungen