IPA 2024 ABSCHLUSSARBEIT

***LEGACY DATEN IMPORT***

***von Raphael Blaauw***

Inhaltsverzeichnis

[Teil 1 - Administrativer Teil 5](#_Toc158971308)

[1 IPA-Projektorganisation 5](#_Toc158971309)

[1.1 Projektorganisation 5](#_Toc158971310)

[2 Organisation der IPA-Ergebnisse 6](#_Toc158971311)

[2.1 Datensicherung 6](#_Toc158971312)

[2.1.1 Code 6](#_Toc158971313)

[2.1.2 Dokumentation 9](#_Toc158971314)

[3 Standards 11](#_Toc158971315)

[3.1 Technologien und Tools 11](#_Toc158971316)

[4 Zeitplan 13](#_Toc158971317)

[5 Arbeitsjournal 14](#_Toc158971318)

[5.1 Tag 1: Montag, 05.02.2024 14](#_Toc158971319)

[5.2 Tag 2: Dienstag, 06.02.2024 15](#_Toc158971320)

[5.3 Tag 3: Mittwoch, 07.02.2024 16](#_Toc158971321)

[5.4 Tag 4: Donnerstag, 08.02.2024 17](#_Toc158971322)

[5.5 Tag 5: Freitag, 09.02.2024 19](#_Toc158971323)

[5.6 Tag 6: Montag, 12.02.2024 20](#_Toc158971324)

[5.7 Tag 7: Dienstag, 13.02.2024 22](#_Toc158971325)

[5.8 Tag 8: Mittwoch, 14.02.2024 23](#_Toc158971326)

[5.9 Tag 9: Donnerstag, 15.02.2024 24](#_Toc158971327)

[5.10 Tag 10: Freitag, 16.02.2024 25](#_Toc158971328)

[6 Projektvorgehen 27](#_Toc158971329)

[6.1 IPERKA Methode 27](#_Toc158971330)

[6.2 Kanban 28](#_Toc158971331)

[Teil 2 – Projekt Dokumentation 29](#_Toc158971332)

[Kurzfassung 29](#_Toc158971333)

[Ausgangssituation 29](#_Toc158971334)

[Umsetzung 29](#_Toc158971335)

[Ergebnis 29](#_Toc158971336)

[7 Informieren 30](#_Toc158971337)

[7.1 Ziele der Aufgabenstellung 30](#_Toc158971338)

[7.2 Vorgaben 30](#_Toc158971339)

[7.3 Ist-Zustand 30](#_Toc158971340)

[7.3.1 Frontend 30](#_Toc158971341)

[7.3.2 Backend 30](#_Toc158971342)

[7.3.3 Überblick 31](#_Toc158971343)

[7.3.4 Klassendiagramme 32](#_Toc158971344)

[7.3.5 Datenbankdiagramm 33](#_Toc158971345)

[7.3.6 Systemgrenzen 33](#_Toc158971346)

[7.4 Analysieren der Probe-Excels 33](#_Toc158971347)

[7.5 Aktivitätsdiagramm 34](#_Toc158971348)

[7.6 User Stories 35](#_Toc158971349)

[7.6.1 Rollen 35](#_Toc158971350)

[7.6.2 Funktionale Anforderungen 35](#_Toc158971351)

[7.6.3 Nichtfunktionale Anforderungen 36](#_Toc158971352)

[8 Planen 38](#_Toc158971353)

[8.1 Soll-Zustand 38](#_Toc158971354)

[8.1.1 Frontend 38](#_Toc158971355)

[8.1.2 Backend 38](#_Toc158971356)

[8.1.3 Klassendiagramm 39](#_Toc158971357)

[8.1.4 Datenbankanpassungen 39](#_Toc158971358)

[8.2 Mockups 39](#_Toc158971359)

[8.2.1 Variante 1 40](#_Toc158971360)

[8.2.2 Variante 2 41](#_Toc158971361)

[8.3 Testkonzept 42](#_Toc158971362)

[8.3.1 Testziele und -strategie 42](#_Toc158971363)

[8.3.2 Testumgebung 42](#_Toc158971364)

[8.3.3 Testfälle 42](#_Toc158971365)

[9 Entscheiden 46](#_Toc158971366)

[9.1 Lösungsvarianten im Vergleich 46](#_Toc158971367)

[10 Realisierung 47](#_Toc158971368)

[10.1 Anpassungen in der Datenbank und Model 47](#_Toc158971369)

[10.2 Erstellen eines Import Services 47](#_Toc158971370)

[10.3 Backend Validierung 48](#_Toc158971371)

[10.4 Import-Wizard im Frontend 48](#_Toc158971372)

[10.4.1 Statische Elemente 49](#_Toc158971373)

[10.4.2 Upload-Modul 49](#_Toc158971374)

[10.4.3 Validation 49](#_Toc158971375)

[10.4.4 Summary-Seite 49](#_Toc158971376)

[10.4.5 Übersicht 50](#_Toc158971377)

[10.5 Sequenzdiagramm 50](#_Toc158971378)

[10.6 Distributed Tracing 51](#_Toc158971379)

[10.7 Logging 51](#_Toc158971380)

[10.8 Storybook 51](#_Toc158971381)

[11 Kontrollieren 52](#_Toc158971382)

[11.1 Testing 52](#_Toc158971383)

[11.1.1 Testkonzept 52](#_Toc158971384)

[11.1.2 Testprotokoll 52](#_Toc158971385)

[11.1.3 Testbericht 53](#_Toc158971386)

[11.2 Unittests 53](#_Toc158971387)

[12 Auswerten 54](#_Toc158971388)

[12.1 Vergleich Ist / Soll 54](#_Toc158971389)

[12.2 Persönliches Fazit zur IPA 54](#_Toc158971390)

[12.2.1 Was war gut? 54](#_Toc158971391)

[12.2.2 Was lief weniger gut? 54](#_Toc158971392)

[12.3 Schlussreflexion 55](#_Toc158971393)

[13 Glossar 56](#_Toc158971394)

[14 Quellenverzeichnis 56](#_Toc158971395)

[14.1 Einsatz von generativen Tools 58](#_Toc158971396)

[15 Danksagungen 58](#_Toc158971397)

# Teil 1 - Administrativer Teil

# IPA-Projektorganisation

## Projektorganisation

**Lehrbetrieb und Durchführungsort:**

Generali Personenversicherungen AG

Soodmattenstrasse 10

8134 Adliswil   
058 472 44 44

**Kandidat:**

Raphael Blaauw

Marchwartstrasse 39, 8038 Zürich

079 622 61 76  
raphael.blaauw@generali.com

raphaelblaauw24@gmail.com

**Auftraggeber:**

Daniel Sager

Generali Personenversicherungen AG

Soodmattenstrasse 10, 8134 Adliswil

daniel.sager@generali.com

**Berufsbildner/ Lehrfirma:**

Roger Sorg

Generali Personenversicherungen AG

Soodmattenstrasse 10, 8134 Adliswil

079 779 53 69  
roger.sorg@generali.ch

**Verantwortliche Fachkraft:** Roman Babenko

Generali Personenversicherungen AG

Soodmattenstrasse 10, 8134 Adliswil

roman.babenko@generali.com

**Hauptexperte:**

Yves Kaufmann

y.kaufmann@yagan.ch

078 872 38 84

**Nebenexperte:**

Christian Soldan

christian.soldan@ubs.com

078 901 73 83

# Organisation der IPA-Ergebnisse

## Datensicherung

Die Datensicherung ist wichtiger Bestandteil des Gelingens der IPA. Für die Sicherstellung der Arbeit wurde auf diverse Versionierungs- und Speicherungssysteme gesetzt.

### Code

Der Codestack ist auf Github ersichtlich und für die IPA wurde ein neuer Branch erstellt.

**Branch**: [Link](https://github.com/IT-Apprentices/Offerten-ReferenzApplikation-Backend/tree/Probe-IPA-Vervollst%C3%A4ndigung-Other)

Folgend der Verlauf des Repositories:

|  |
| --- |
| commit 1025ccb91c0b15302f19a8d0676affbe136c17f5 (HEAD -> legacy-data-import-wizard, origin/legacy-data-import-wizard)  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Thu Feb 15 11:30:40 2024 +0100  cleanup(fe+be): cleanup and reformatting  commit 5757244343c84541f3f13b68279d5cbb80b1cba6  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Wed Feb 14 14:59:37 2024 +0100  refactor(fe): refactoring frontend details  commit 4ec0c9e98c55d2e30105ffed9a4fb6eb7ca5ea56  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Wed Feb 14 14:58:38 2024 +0100  test(fe): storybook for all ui-components  commit 58d94e885d124c81fd20cadfbff19331460d08ae  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Wed Feb 14 14:58:05 2024 +0100  test(be): unit-tests for ExcelProcessingService and ImportService  commit 5f4273ad9de47e67075f33c3808a2e4d74a8698b  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Wed Feb 14 14:57:29 2024 +0100  refactor(be): refactoring separation of excel and import logic  commit 11aaf4776acef14397a61b71ae44b4dfb2b17d9d  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Wed Feb 14 14:56:29 2024 +0100  fix(fe): bug fix FormDataFetch + Storybook upgrade and examples  commit 8acb9a4826877eafac88dc18a3da1d9187cdd1e5  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Tue Feb 13 14:23:08 2024 +0100  fix(fe): styled LoadingAnimation.tsx and Summary.tsx  commit 44efd19fa458fcaf6a8b94bc9f697a02cb6cbec1  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Tue Feb 13 14:07:51 2024 +0100  feat(be): structured logging  commit 84496d3cdcb09824c2c468b59130da7b0f038bc3  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Tue Feb 13 13:39:13 2024 +0100  feat(fe+be): introduced server action and tracing  commit 33413d042295d3cdfa9ca35d32f0e9b4934cc51f  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Mon Feb 12 16:43:56 2024 +0100  feat(fe): structured logging  commit 5c366e7c0bc2b6342eedcb8294a4be8577111be6  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Mon Feb 12 15:51:42 2024 +0100  fix(be): Spelling error fixed in Excels  commit 111881e2110648759bc2f72e4196332096fe76bf  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Mon Feb 12 15:51:02 2024 +0100  feat(fe): Import-Wizard implemented inclusive ErrorPage.tsx, LoadingAnimation.tsx and Summary.tsx  commit 413fd7fbbd7bb7d581ca6b709eda757a19563ce2  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Mon Feb 12 09:16:25 2024 +0100  wip - file-upload  commit 4e265df9cff9b04a4150736fdd394eb30f40dc63  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Fri Feb 9 18:01:45 2024 +0100  fix(be): Change from Multifile to Base64 String  commit 7b91926a0f480053afd193e88e4484d770e01162  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Fri Feb 9 18:01:16 2024 +0100  feat(fe): wip Import-Wizard  commit 506c3294909f36060f935628177c629e2654d39a  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Fri Feb 9 14:23:53 2024 +0100  feat(be): validation for ImportService + introducing ImportResult  commit e1c31d830ff8a85c7911a6383b07ea56caf394e2  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Thu Feb 8 15:59:05 2024 +0100  feat(be): introduced Service for importing customers via .xls  commit 00e0c35f63e7210e2e1c998c4e01b4b701a86997  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Thu Feb 8 15:47:56 2024 +0100  feat(be):added new column "partner\_number" to DB and Model |

### Dokumentation

Durch Microsoft Office wird der neuste Stand bei OneDrive automatisch gespeichert. Für zusätzliche Sicherheit wird ein Backup am Schluss jedes Tages auf einem separaten Github Repository erstellt:

**Repo**: [Link](https://github.com/blauWhale/IPA)

Folgend der Verlauf des Repository bzw. der Sicherung

|  |
| --- |
| commit c8ffd2309a639cc7f6be4f4abb745c2901fec7f7 (HEAD -> main, origin/main, origin/HEAD)  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Thu Feb 15 15:25:57 2024 +0100  Sicherung 15.02.2024  commit e08f487463c438279ef34008a0c78aa70ff8b44f  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Wed Feb 14 16:57:17 2024 +0100  Sicherung 14.02.2024  commit 2998ce6cf06900483db5d5b6a0c666353a1be637  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Tue Feb 13 16:30:08 2024 +0100  Sicherung 13.02.2024  commit 29f4cd86c702bad6c40309044befebdee2127815  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Tue Feb 13 08:38:21 2024 +0100  Sicherung 12.02.2024  commit 84edaa326c201d6a2ca2a94678a3ab2e8d876a1a  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Fri Feb 9 17:59:58 2024 +0100  Sicherung 09.02.2024  commit dc504915c84c4fed840c3c521d8f2f6b67bb2716  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Thu Feb 8 17:42:09 2024 +0100  Sicherung 08.02.2024  commit 08f126fc692ce6be56fdc1ea13450b3f16a5b53f  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Wed Feb 7 16:16:16 2024 +0100  Sicherung 07.02.2024  commit 46b60f5bc80c9a396370c49a9b9669bd4337a0c7  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Tue Feb 6 18:40:33 2024 +0100  Sicherung 06.02.2024  commit 970f0d1544d90735f1e8fd40b3a8c0ac3f8e399c  Author: Raphael Blaauw <raphaelblaauw24@gmail.com>  Date: Mon Feb 5 16:35:28 2024 +0100  Sicherung 05.02.2024  commit 5c6658a8eacac62317b5bef8f3fb840430ae4e32  Author: blauWhale <80290651+blauWhale@users.noreply.github.com>  Date: Mon Feb 5 08:22:54 2024 +0100  Initial commit |

# Standards

## Technologien und Tools

Im Team werden mit folgenden Tools und Technologien gearbeitet:

|  |  |
| --- | --- |
| **Frontend** | |
| **Name** | **Einsatz** |
| Next.js | Frontend-Technologie |
| shadcn | Komponenten Bibliothek |
| Typescript | Frontendsprache |
| Prettier | Code Formatierung |
| React Testing Library | Tests |
| Storybook | Frontend Workshop |

|  |  |
| --- | --- |
| **Backend** | |
| **Name** | **Einsatz** |
| Springboot | Backend-Technologie |
| Java | Backendsprache |
| shadcn | Komponenten Bibliothek |
| Typescript | Frontendsprache |
| Google Java Styleguide | Code Formatierung |
| JUnit | Tests |
| Spring Boot Testing | Tests |

|  |  |
| --- | --- |
| **Daten Layer** | |
| **Name** | **Einsatz** |
| PostgreSQL | Datenbank-Technologie |
| Flyway | Datenbankversionierungstool |

|  |  |
| --- | --- |
| **Identity Provider** | |
| **Name** | **Einsatz** |
| OpenID Connect | Identitätsprotokoll |
| Keycloak | Identitätsmanagement |

|  |  |
| --- | --- |
| **Lokale Entwicklung** | |
| **Name** | **Einsatz** |
| Jetbrains IntelliJ | Entwicklungsumgebung |
| Docker | Containerisierungstool |

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

|  |  |
| --- | --- |
| **Dokumentation und Admin** | |
| **Name** | **Einsatz** |
| Microsoft Office | Dokumentation |
| Google Chat | Kommunikation |
| ChatGPT | Textgenerierung, Informationen |
| Draw.io / Plantuml | UML-Diagramme |

# Zeitplan

# Arbeitsjournal

## Tag 1: Montag, 05.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Aufbau IPA-Bericht  Zeitplan erstellen  Analysieren von Beispieldaten  Dokumentieren des IST-Zustandes |
| **Erreichte Ziele** | Gute Grundlage im Bericht aufgebaut und IST-Zustand abgebildet |
| **Probleme** | Das Erstellen von UML-Diagrammen mit IntelliJ und PlantUML brauchte seine Zeit.  Der Zeitplan erforderte viel Vorausdenken, striktes Abschätzen und Einplanen von Aufgaben |
| **Hilfestellung** | - |
| **Ausserplanmässige Arbeiten** | - |
| **Reflexion** | *Was lief heute positiv?*  Voller Vorfreude auf diese IPA konnte ich meine Aufgabe fassen und damit beginnen. Ich bin mit viel Energie in diese zehn Tage gestartet. Der Aufbau des IPA-Berichts ging dank einer guten Vorarbeit/Vorlage sehr gut. Das Arbeiten mit IPERKA bringt viele Vorteile mit sich. Mit dieser Methode wird ein klarer Weg eingeschlagen, der keine Abkürzungen oder Überspringen von Schritten zulässt, sondern ein klares Konzept und eine saubere Planung fordert. Mir persönlich fehlt aber ein gesamter Überblick über den Fortschritt des Projektes. Daher habe ich für die IPA-Zeit ein Kanban-Board auf Notion.so mit den Aufgaben des Zeitplans eingerichtet. So weiss ich ganz genau, was noch offen ist, an was ich gerade bin und was ich bereits erledigt habe.  *Was sehe ich kritisch?*  Der Zeitplan war herausfordernd und brauchte viel genaues Arbeiten, um die Aufgaben über die zwei Wochen richtig zu verteilen. Ich kam aber gut voran und konnte schneller als gedacht den Plan fertigstellen. Ich möchte morgen den Zeitplan meinem Experten, Herrn Kaufmann, präsentieren, um ein Feedback einzuholen. Jeder Experte legt den Schwerpunkt bei der Planung anders.  *Meine Erkenntnisse von heute?*  Ich wollte heute so viel wie möglich angehen. Mir bleibt aber noch genügen Zeit und ich muss mich selbst nicht unnötig stressen. |
| **Nächste Schritte** | Einarbeiten in den Umgang mit Excels in Spring/Java  User-Stories und Akzeptanzkriterien  Aktivitätsdiagramm des Import-Prozesses |

## Tag 2: Dienstag, 06.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Arbeiten mit Excel-Sheets in Java  User-Stories und Akzeptanzkriterien schreiben  1. Expertenbesuch  Aktivitätsdiagramm |
| **Erreichte Ziele** | Die letzten Arbeiten in der Informationsphase sind abgeschlossen. Morgen kann ich zur Planung übergehen. |
| **Probleme** | Es war schwierig abzuschätzen, was eine User-Story ist und was eine einfache Anforderung bzw. Kriterium der IPA. Nicht alles gehört als User-Story erfasst. |
| **Hilfestellung** | Kurze Diskussion über den Zeitplan und Kriterien mit Roman. Er gab mir Feedback, wie ich meinen Dokumentationsaufwand am besten gegenüber dem Experten aufzeigen kann. |
| **Ausserplanmässige Arbeiten** | Kleinere Anpassungen am Zeitplan nach dem ersten Expertenbesuch. |
| **Reflexion** | *Was lief heute positiv?*  Ich konnte heute viele Arbeiten abschliessen: Ich musste an diversen Stellen in der Dokumentation etwas ergänzen, hinzuschreiben oder Grafiken erstellen. Trotzdem verlor ich den Überblick nicht. Die ersten Versuche ein Excel mit Java, bzw. der Apache POI Library, zu erstellen, gelangen sehr gut. Ich musste mich selber bremsen, um nicht gleich mit dem Entwickeln loszulegen. Der erste Expertenbesuch war sehr angenehm und ich bin froh, einen erfahrenen und fairen Experten bekommen zu haben.  *Was sehe ich kritisch?*  Die User-Stories auszuarbeiten kostete viel Zeit und ich war mir bei manchen Stellen unsicher, ob eine Anforderung in der Aufgabenstellung auch als User-Story anzusehen ist. Ein gutes Beispiel sind die Kriterien bezüglich dem Testing. Ich verfasste bewusst keine Stories oder Akzeptanzkriterien für jede Testart, die gefragt ist. Sonst würde es beim Testkonzept dazu kommen, dass getestet wird, ob so oder so getestet wird. Dies machte für mich für den Umfang der IPA wenig Sinn. Wiederum Themen, wie Tracing und strukturiertes Logging als Story zu erfassen, macht für mich durchaus Sinn. Aber nur im Umfang dieser Arbeit. So bin ich sicher, dass ich die Kriterien am Schluss erfüllt habe.  *Meine Erkenntnisse von heute?*  Durch das Feedback des Experten merkte ich, dass mein Zeitplan nicht klar meinen Dokumentationsaufwand widerspeigelt. Ich muss besser aufzeigen, dass ich genügend Zeit für meinen IPA-Bericht aufwende.  Nach den ersten Arbeiten fühle ich mich gut informiert und habe eine gute Grundlage, um jetzt die Planung anzugehen. Ich freue mich das ich einen guten Einstieg in die Arbeit gefunden habe und dass ich soweit konzentriert und zielgerichtet arbeiten kann. |
| **Nächste Schritte** | Festlegen des Soll-Zustands im Backend und Frontend  Mockup des Import Wizards in Figma |

## Tag 3: Mittwoch, 07.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Planen des Soll-Zustandes im Front- und Backend  Mockups des Import Prozesses in Figma angefangen zu gestalten. |
| **Erreichte Ziele** | Soll-Zustand des Projektes festgehalten. |
| **Probleme** | - |
| **Hilfestellung** | - |
| **Ausserplanmässige Arbeiten** | - |
| **Reflexion** | *Was lief heute positiv?*  Das Entwickeln der Idee zum Soll-Zustand konnte ich dank einer guten Vorbereitung in der Informationsphase gelassen angehen. Ich entschied mich mit Bulletpoints zu arbeiten und kam so gut voran. Die Mockups in Figma zu gestalten, machte mir Spass und ich bin bereits sehr zufrieden mit den ersten Ideen. Morgen möchte ich noch eine kleine Variation einbauen, um verschiedene Optionen anbieten zu können.  Während dem Gestalten geht die Zeit wie im Flug und ich muss aufpassen, dass ich das Dokumentieren nicht vernachlässige.  *Was sehe ich kritisch?*  Das Exportieren von diversen Diagrammen für die Dokumentation von IST- bzw. SOLL-Zustand gelang nicht auf Anhieb. Immer wieder entdeckte ich einen kleinen Fehler oder wollte noch eine kleine Änderung vornehmen.  Morgen Nachmittag werde ich mit dem Entwickeln beginnen. Ein Kollege teilte mir mit, dass der Pull-Request mit dem Modul zum Darstellen der Kunden im OVWEB immer noch nicht im Main-Branch ist. Ich bin auf Roman zu und teilte ihm mit, dass dies unbedingt Priorität finden muss. Ich mache mir keine Sorgen, bin aber ein wenig frustriert, dass der Main-Branch am dritten Tag meiner IPA noch nicht bereit für meine Entwicklungsarbeiten ist.  *Meine Erkenntnisse von heute?*  Mit den weiteren Erkenntnissen in der Planung kann ich mir nun immer mehr ein Bild machen, wie die Realisierung ausschauen wird. Die begonnen Mockups in Figma geben mir sogar bereits ein visuelles Bild, wie ich die Implementierung des Import-Wizards gestalten möchte. |
| **Nächste Schritte** | Variation des Mockups anfertigen  Testkonzept erstellen  Entscheid für eine Variation und Begründen im IPA-Bericht  Start der Entwicklung |

## Tag 4: Donnerstag, 08.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Mockups in Figma abschliessen  Testkonzept anfertigen  Beginnen mit den Entwicklungsarbeiten |
| **Erreichte Ziele** | Planung und Entscheidungsphase ist so weit abgeschlossen  Die Entwicklungsarbeiten sind gestartet |
| **Probleme** | Heute Nachmittag ist mein Laptop abgestürzt. Die Fehlerquelle ist WSL 2. Ich ging mit meinem Problem kurz zu Roman. |
| **Hilfestellung** | Roman hat mir bei meinem plötzlichen Blackscreen geholfen, indem er der Ursache nachging.  Wir bemerkten das WSL 2 nicht auf der neusten Version ist. Ich werde am Ende des Tages ein Update durchführen. |
| **Ausserplanmässige Arbeiten** | Kurze Session zu den WSL 2 Problemen |
| **Reflexion** | *Was lief heute positiv?*  Der Einstieg in den Tag mit dem Erstellen einer Variation des Mockups war ideal. Ich konnte kreativ sein und doch war die Arbeit nicht zu anspruchsvoll. Ich entschied mich, nicht zu viel Zeit in eine komplett andere Variation zu stecken, sondern mir eher eine Variation zu bauen, bei der ich über Benutzerfreundlichkeit schreiben konnte. Bei der Entscheidung, welche Variante ich wähle, stieg ich gleich in die Dokumentation ein. Anstatt es für mich selbst zuerst zu überlegen, machte ich mir die Gedanken gleich schriftlich.  Das Testkonzept war sehr aufwendig, jedoch ein eigenes Kriterium, daher wichtig. Ich konnte ChatGPT verwenden, um eine gute Grundlage für meine Testfälle generieren zu lassen. Eigentlich war bereits klar, was zu testen ist: Sind die Akzeptanzkriterien erfüllt? Ich war froh, hatte ich bereits einen guten Ausgangspunkt.  Heute Morgen wurde mir bestätigt, dass der Main-Branch endlich bereit ist. Ich war beruhigt und konnte später am Tag den neusten Stand auschecken und meine Entwicklungsarbeiten beginnen. Mir machten die Arbeiten in Java und der Datenbank grossen Spass und die Frameworks für die Verarbeitung den Excels war auch gut aufgebaut und einfach zu implementieren.  So kam ich schnell zu einem Ergebnis. Mit dem Swagger UI konnte ich mein Code immer gleich ausprobieren und die Fehler ausmerzen. Am Ende des Tages las mein Code die Excels und speicherte diese in die Datenbank. Das gab mir ein gutes Gefühl. Der erste kleine Berg wurde erklimmt.  *Was sehe ich kritisch?*  Es war sehr frustrieren als mein Laptop abstürzte. Ich zögerte erst auf Roman zuzugehen, da ich konzentriert an meinen Programmieraufgaben war. Jedoch hatten wir bereits eine Woche zuvor eine Vermutung das es mit WSL 2 zu tun hat. Ich sammelte alle kritischen Ereignislogs von Windows und teilte sie mit Roman, sodass ich an meiner IPA weitermachen konnte, während er Lösungen suchte. Das funktionierte wunderbar und eine mögliche Lösung war schnell gefunden.  Am späteren Nachmittag merkte ich die Zeilen über Zeilen an der Dokumentation und dem Code schreiben in den Fingern.   Ich machte eine kleine Pause, nahm einen Kaffee und ging mit guter Musik nochmals in den Endspurt, um die Dokumentation auf den aktuellen Stand des Projektes zu bringen.  *Meine Erkenntnisse von heute?*  Mit den fertigen Mockups konnte ich mir ein gutes Bild machen, wie das User Interface ausschauen wird. Beim Testkonzept erstellen, merkte ich, dass das Testing aufwendig wird. Zum Glück habe ich mit rund vier Stunden genügend Zeit eingeplant. Allgemein liege ich gut in Zeitplan. Während dem Entwickeln habe ich auch einige neue Erkenntnisse über die Testdaten gewonnen und ihre eher kleineren Facetten. |
| **Nächste Schritte** | Backend Validierung  Frontend Erklär-Seite und Upload Modul |

## Tag 5: Freitag, 09.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Backend Validierung  Frontend Erklärseite und Uploadseite |
| **Erreichte Ziele** | Backend robuster gemacht mit mehr Kontrollen vor und während dem Import-Prozess  Begonnen mit der Einstiegsseite im Frontend |
| **Probleme** | Es gab heute beim Entwickeln einige Stolpersteine. Bei den meisten wusste ich mir selbst zu helfen, aber sobald ich merkte, dass ich zu viel Zeit verliere, bin ich auf Roman zugegangen. |
| **Hilfestellung** | Heute musste ich Roman einige Dinge fragen. Zuerst vielen mir immer mehr Rechtschreibfehler in den Testdaten auf. Wir beschlossen, dies beim zweiten Expertenbesuch anzusprechen und danach die korrigierten Daten hochzuladen.  Am Nachmittag wollte auch nach mehreren Anläufen mein Daten-Upload nicht funktionieren. Mit einem sehr hilfreichen Tipp bezüglich des Datenformats konnte ich weitermachen. |
| **Ausserplanmässige Arbeiten** | Überdenken des Filetypes beim Datenupload |
| **Reflexion** | *Was lief heute positiv?*  Nach der ersten Woche fühle ich mich tief in der Thematik drin. Mit einer klaren Planung, was umzusetzen ist und bereits einigen Stunden Programmierarbeit, habe ich ein gutes Gefühl für den Umfang der IPA. Ich kann konzentriert Schritt für Schritt den Import-Wizard umsetzen.  *Was sehe ich kritisch?*  Ich sträubte mich ein wenig auf Roman zuzugehen und wollte so gut es geht allein auf eine Lösung kommen. Jedoch musste ich nach einiger Zeit pragmatisch denken und bekam auch prompt einen guten Link oder Tipp für mein Problem.  Das Upload-Modul im Frontend hat das Thema von Base64 aufgemacht. Roman wies darauf hin, dass ich beim Upload mit Base64 Strings arbeiten sollte. Die Server Actions, die ich für den Datenupload angelegt habe, verlangt einen serialisierbaren Datentyp. Exceldaten bzw. Binärdaten kann Next.Js nicht einfach in eine simplere Datenform wie JSON oder einem String konvertieren. So muss ich mir eine andere Lösung überlegen. Ich nehme mir vor, nicht zu viel Zeit in dieses Thema zu investieren und pragmatisch zu bleiben.  *Meine Erkenntnisse von heute?*  Bei der Validation könnte ich fast endlose Szenarien ausdenken, die ich noch abdecken könnte. Ich nahm daher nochmals die Anforderungen zur Hand, um wirklich nur so weit zu validieren, damit der Umfang der IPA erfüllt ist.  Bei den kleineren Problemen beim Entwickeln vergass ich fast die Dokumentation nachzuführen.  Die Blockzeiten am Abend und am Morgen helfen mir dabei den IPA-Bericht vor lauter Next.Js und Spring Boot nicht ausser Acht zu lassen. |
| **Nächste Schritte** | Upload fertig machen  Frontend Validierung  Erstellen einer Summary Page  Ladeanimation |

## Tag 6: Montag, 12.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Frontend Validierung  Summary Page erstellen  Kleinere Anpassung im Frontend |
| **Erreichte Ziele** | Prozess komplett abgebildet im Frontend, es fehlen nur noch kleinere Sachen |
| **Probleme** | - |
| **Hilfestellung** | Ich suchte vergebens nach einem Beispiel in unserer Applikation für eine Weiterleitung, falls der User nicht eingeloggt ist. Roman bestätigte mir, dass wir tatsächlich nie so etwas implementiert bzw. konfiguriert haben. Er versprach auf mich zuzukommen, falls er Zeit findet, so etwas anzuschauen. |
| **Ausserplanmässige Arbeiten** | - |
| **Reflexion** | *Was lief heute positiv?*  Das zweite Expertengespräch brachte viel gutes Feedback. Herr Kaufmann wirkte zufrieden mit dem Umfang und Fortschritt meines Berichts. Das stimmte mich auch gut und ich konnte motiviert in die Woche starten. Ich konnte im Frontend fast alles umsetzten und kam sehr gut durch. Das lag sicher auch daran, dass ich zuvor klar geplant habe, was zu erledigen ist und wie es aussehen soll. Mit der gewonnen Zeit konnte ich mich dem IPA-Bericht widmen und die neuen Seiten und Funktionen dokumentieren. Ich war mit der Arbeit heute sehr glücklich und möchte diese Motivation für die restliche Zeit mitnehmen.  *Was sehe ich kritisch?*  Während den Entwicklungsarbeiten merke ich, dass die Implementation eines Fortschrittbalken, wie auf dem Mockup, eine Erweiterung des Backends oder das Umstellen bei der http-Übertragung bedeuten würde. Wir verwenden hierfür einen Client auf der Oberfläche, der die Typisierung gewährleistet. Ich konsultierte nochmals die Anforderungen und finde abgesehen vom Anspruch eines stetigen Informationsflusses an den Benutzer, kein direktes Kriterium, um einen Mehraufwand zu begründen. Ich habe mich daher entschieden, dass eine einfache Ladeanimation ausreicht, um anzuzeigen, dass der Importvorgang gestartet wurde und im Gange ist.  *Meine Erkenntnisse von heute?*  Der grösste Hügel in der Umsetzung ist hinter mir und die Detailarbeit kann beginnen.   Mit gutem Fokus und einem aufgeräumten Zeitplan konnte ich meine Arbeit gut einteilen und schnell an eine gute Version des Import-Wizards kommen. Ich bin mir nach diesem Tag sicher, dass ich mit meinem Zeitplan durchkomme.  Die grössten Stolpersteine habe ich hinter mir oder konnte sie mehr oder weniger vermeiden. Mit dem Feedback vom Experten bin ich positiv gestimmt, dass sogar eine sehr gute Note möglich ist, wenn ich dranbleibe. |
| **Nächste Schritte** | Ladeanimation fertigstellen |

## Tag 7: Dienstag, 13.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Ladeanimation ansprechend gestaltet und kleine Verbesserungen im Importprozess  Strukturiertes Logging und das Tracing integriert |
| **Erreichte Ziele** | Der Import Wizard ist fertig und fürs Testing bereit. |
| **Probleme** | Distributed Tracing |
| **Hilfestellung** | Die Beispiele in unserer Applikation für das Tracing waren nicht sehr aussagekräftig. Ich ging auf Roman zu und liess mir den Code nochmals erklären. |
| **Ausserplanmässige Arbeiten** | Umschreiben der Upload-Funktion in eine Server Action |
| **Reflexion** | *Was lief heute positiv?*  Der Import-Wizard ist so weit fertig. Ich bin stolz auf die Realisation und meine Arbeitsweise. Den Fokus möchte ich weiter mitnehmen und setzte gleich an, bei Arbeiten für die Individual-Kriterien. Für das strukturierte Logging kann ich mich an einem Beispiel orientieren und versuche aussagekräftige Logs auszugeben. Später in der Umsetzung des Tracings kann ich diese gleich verwenden. Das macht die Arbeit gleich viel interessanter. Ich freue mich, dass ich mich solchen Detailarbeiten widmen kann.  *Was sehe ich kritisch?*  Das Thema Tracing kenne ich erst seit ein paar Tagen. Kurz vor der IPA gab es dazu eine kleine Präsentation. Ich konnte mich im Selbststudium mit dem Konzept anvertrauen.  Jedoch war die Umsetzung in unsere Applikation bis heute fremd. Nachdem ich bereits gut vorangekommen bin und im Backend das Tracing einbauen konnte, steckte ich bei der Implementierung bei den letzten Details fest. Ich entschied mich mit Roman zusammen zu definieren, was genau verlangt war und wie ich das im Frontend richtig umsetzten kann. Nach einer kurzen Zeit konnte ich meine «Spans» in Grafana anschauen.  Beim Thema Storybook, um die UI-Komponenten separat aufzuzeigen, kam ich dank einer guten Dokumentation online schneller vorwärts.  *Meine Erkenntnisse von heute?*  Mit den letzten Realisationstasks schaue ich langsam Richtung Testen. Das Konzept vom Tracing gefiel mir sehr gut und es macht Freude, wenn solche Ideen in den eigenen Applikationen klarer und spürbarer sind. Ich freue mich morgen weiterzumachen und bin motiviert. |
| **Nächste Schritte** | Unit Tests schreiben  Refactoring wo nötig |

## Tag 8: Mittwoch, 14.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Debugging und Refactoring  Schreiben von Unit Tests  Komponenten in Storybook erfasst |
| **Erreichte Ziele** | Codestack aufgeräumt und automatisierte Tests implementiert |
| **Probleme** | - |
| **Hilfestellung** | Roman lieferte mir zwei elementare Bugfixes, die waren für das Umsetzten von durchgängiges Tracing und das Arbeiten mit Storybook. Der Commit mit SHA (11aaf47) beinhaltet alle von Roman zu Verfügung gestellten Fixes. |
| **Ausserplanmässige Arbeiten** | - |
| **Reflexion** | *Was lief heute positiv?*  Der Morgen brachte wichtige Bugfixes von Roman für unsere gesamte Applikation. Einerseits damit die Tracing Parent ID von Frontend weiter an das Backend richtig weitergegeben wird und andererseits Upgrades der Version von Storybook, dazu Beispiele, wie Server-Action zu mocken sind. Roman ladete die Änderungen auf einen anderen Branch hoch und ich ging umgehend ans Verwenden auf meinem IPA-Branch. Ich kam sehr gut vorwärts und konnte mich gut konzentrieren. Während weiteren Refactoring-Arbeiten hatte ich immer die Tests im Kopf. Was gehört zusammen? Was muss aufgeteilt werden. Ich entschied mich den Importprozess und die Logik zu Verarbeitung der Excels so gut wie möglich zu trennen. Der ImportService muss sich so nicht um Excel-Themen kümmern und falls andere Service Excels verarbeiten oder validieren möchten, gibt es neu einen ExcelsProcessingService. So konnte ich auch gute Unit-Tests schreiben. Für den weiteren Verlauf vom Tag nahm ich den Kriterienkatalog und mein Kanban zur Hand, um beim Refactoring ja keine wichtigen Punkte auszulassen. Mir half das sehr, um strukturiert vorzugehen.  *Was sehe ich kritisch?*  Die Hotfixes während meiner IPA waren notwendig, aber vermeidbar. Zum Glück war ich sowieso beim Anpassen von diversen Codestellen. Es fühlte sich trotzdem so an, als werde ich als Versuchskaninchen verwendet. Die Individual-Kriterien, die auf Storybook und Tracing aufbauen, wären ohne weitere Arbeiten von Roman nicht zu bewältigen gewesen. Erst nach meinen Versuchen wurde klar, was noch fehlt oder wo noch Bugs sind. Sicherlich nicht ideal für meine IPA-Zeit, trotzdem gönn’ ich meinen Mitlernenden eine jetzt gut aufgesetzte Applikation in Hinsicht auf die IPA-Themen.  *Meine Erkenntnisse von heute?*  Der Refactoring-Tasks hat nochmals die Codequalität gesteigert und ich bin froh, dass ich mir dafür Zeit genommen habe. Storybook erweist sich als sehr nützliches Tool. Codeänderungen bei einer Fehlermeldung können, ohne einen Fehler erzeugen zu müssen, angezeigt werden.  Morgen steht die Kontrolle an. Ich freue mich darauf meinen Import-Wizard auf Herz und Nieren zu testen und danach die IPA abschliessen zu dürfen. |
| **Nächste Schritte** | Testing  IPA-Bericht mit Kriterienkatalog abgleichen |

## Tag 9: Donnerstag, 15.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Testing  Kurzfassung  IPA-Bericht durchsehen und korrigieren |
| **Erreichte Ziele** | Kontrolle abgeschlossen  Überblick was noch ansteht |
| **Probleme** | - |
| **Hilfestellung** | - |
| **Ausserplanmässige Arbeiten** | - |
| **Reflexion** | *Was lief heute positiv?*  Dank meinem Testkonzept wusste ich genau, was mich beim Testen erwartet. Das Durchführen der Testfälle war sehr angenehm mit der Vorlage, die ich vorbereitet hatte. Ich konnte mich ganz auf das genaue Kontrollieren und Aufschreiben der Resultate konzentrieren. Die Tests resultierten alle erfolgreich, bis auf einige Randbemerkungen. Ich machte kleinere Anpassungen und konnte am Ende des Tages eine Version auf Github pushen, mit der ich sehr zufrieden war. Ich hatte noch Zeit meinen IPA-Bericht mit den Kriterien abzugleichen und mein Kanban-Board mit den letzten Aufgaben und Kontrollen zu ergänzen.  *Was sehe ich kritisch?*  In meinem Büro war Ferienzeit und darum niemand verfügbar, um meine Testfälle durchzuführen. Es war sicher nicht ideal, dass ich selbst die Kontrolle durchführte, aber für den Rahmen der IPA nicht weiter schlimm.  *Meine Erkenntnisse von heute?*  Ich bin fast fertig und es fehlt nur noch die Auswertung und Formsachen, um meine IPA guten Gewissens abzugeben. |
| **Nächste Schritte** | Reflexion und Auswertung  Anhang erstellen  Formatierung |

## Tag 10: Freitag, 16.02.2024

|  |  |
| --- | --- |
| **Tätigkeiten** | Reflexion  Überarbeitung des IPA-Berichts  Formatierungen  Export Code, Zeitplan und IPA-Bericht |
| **Erreichte Ziele** | Abschluss der IPA |
| **Probleme** | - |
| **Hilfestellung** | Gegenlesen durch meinen Vater |
| **Ausserplanmässige Arbeiten** | - |
| **Reflexion** | *Was lief heute positiv?*  Mein Vater nahm sich Zeit, um meinen IPA-Bericht gegenzulesen und zu korrigieren. Er sah die Fehler viel besser, da er sich nicht auf den Inhalt konzentrierten musste. Nach zwei Wochen viel mir das schwer. Trotzdem widmete ich mich vollkommen den Anforderungen an das Format. Ich arbeitete genau und exportierte die Dokumente immer wieder und verbesserte die Darstellung, damit sie für die Bewertung angenehm zum Lesen ist.  *Was sehe ich kritisch?*  Die Schlussreflexion kostete Zeit. Ich wollte verschiedene Aspekte beleuchten und trotzdem prägnant bleiben. Das Resultat war eine gute Mischung, aber mit mehr Zeit hätte ich sicher noch ein paar Zeilen streichen bzw. kürzen können.  *Meine Erkenntnisse von heute?*  Mit einer ausführlichen Reflexion der Arbeit kann ich zuversichtlich sagen, dass ich in dieser Lehre zu einer Fachperson in der Informatik angewachsen bin. Ich bin stolz auf meine Arbeit und diesen Bericht. |
| **Nächste Schritte** | Abgabe |

# Projektvorgehen

## IPERKA Methode

Die IPERKA-Projektmethode besteht aus sechs Schritten: Informieren, Planen, Entscheiden, Realisieren, Kontrollieren und Auswerten. Sie ist vielseitig einsetzbar, etwa in Lehrbetrieben, Schulen und im Alltag. Die Methode fördert ein strukturiertes Vorgehen und betont die Bedeutung von Planung und Informationsbeschaffung sowie die Auswertung nach Abschluss des Projekts.

Das Projekt ist nach der IPERKA-Methode aufgebaut, damit stets eine gegliederte Struktur im Projektablauf ersichtlich ist. Hierzu sind die 6 Phasen erforderlich.

**Informieren:**

Der Auftrag und dessen Aufgabenstellung werden so gut wie möglich geklärt, die Informationen werden gewertet und sortiert und wesentliche Punkte so früh wie möglich erkannt, sowie allfällige Fragen geklärt. Dies ist wichtig, um den Zeitplan fertigzustellen und zum Verständnis, was überhaupt erwartet wird.

**Planen:**

Um einen guten und strukturierten Projektablauf zu garantieren, muss das Projekt präzise geplant werden. Wenn möglich werden mehrere Lösungswege für komplexere Abläufe geplant. Dabei wird ein Testkonzept, sowie ein Realisierungskonzept erstellt.

**Entscheiden:**

Die unterschiedlichen Lösungswege werden miteinander verglichen und davon wird dann der bestmögliche ausgewählt. Dabei muss man sich überlegen, ob die Ideen sinnvoll sind und ob man es dann auch wirklich so umsetzen kann.

**Realisieren:**

Erst wenn der Ablauf klar definiert ist und alle Entscheidungen getroffen sind, wird mit der Realisierung begonnen. Die Arbeitsabläufe werden stetig protokolliert und die Ist-Werte werden im Zeitplan eingetragen.

**Kontrollieren:**

Die Resultate müssen so getestet werden, wie es im Testkonzept steht, aber erst wenn die Entwicklung vollständig beendet ist. Fehlgeschlagene Test werden hierbei direkt korrigiert und behoben oder als Fehler vermerkt und dokumentiert.

**Auswerten:**

Zum Schluss gibt es eine Reflexion über die Arbeit und die Erfahrungen, die man gesammelt hat. Dabei geht man durch alle Schritte, die man innerhalb des Projekts gemacht hat und es wird untersucht was gut war und wo Verbesserungen möglich sind. Das ist auch praktisch für die Umsetzung zukünftiger Projekte.

## Kanban

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte BeschreibungNeben der IPERKA-Methode setzte ich zwei Kanban Boards ein, um den Überblick zu behalten. Das erste für den Kriterienkatalog, damit ich erledigte Punkte abhaken und mich auf offene Punkte konzentrieren kann. Das zweite für die tägliche Aufgaben während der IPA. Ich verwende die Board-Funktion von Notion.so.

Kanban für die Schlusskontrolle

Kanban für tägliche Arbeit

# Teil 2 – Projekt Dokumentation

# Kurzfassung

## Ausgangssituation

Die Generali Schweiz AG, eine renommierte Versicherungsgesellschaft, bedient rund eine Million Kunden in der Schweiz mit einer breiten Palette von Versicherungs- und Vorsorgelösungen. Die Beratung erfolgt durch Versicherungsvertreter, die die hauseigene Software ANGEBOTsoftware nutzen. Diese Monolith-Anwendung, entwickelt in Delphi, dient dem Verwalten von Kundendaten und dem Berechnen von Prämien.

Im Rahmen des Projekts «OVWEB» wurde ein Lernenden-Team beauftragt, Teile der ANGEBOTsoftware in moderne, webbasierte Technologien zu überführen. Dies erfordert die Entwicklung eines Import-Wizards, der es Versicherungsberatern ermöglicht, Kundendaten in Form von Excel-Dateien (.xls) effizient und fehlerfrei in das neue System zu importieren. Dabei gilt es, Duplikate zu vermeiden und die Datenintegrität zu wahren.

## Umsetzung

Die Umsetzung des Projekts folgte der IPERKA-Methode, die eine strukturierte Arbeiten am Projekt ermöglichte. Parallel wurde ein Kanban-Board geführt, um die täglichen Arbeiten zu verfolgen. Nach einer umfangreichen Informations- und Planungsphase, in der der IST-Zustand festgehalten und der SOLL-Zustand aufgezeigt wurde, konnte mit einem klaren Bild in die Realisierung gestartet werden.

Es wurden Anpassungen in der Datenbank und im Model vorgenommen, um die Legacy-Daten aufnehmen zu können. Ein spezieller Import Service wurde mit der Apache-POI-Bibliothek entwickelt, um die Excel-Daten zu verarbeiten und in die "Customer"-Tabellen von OVWEB zu importieren. Das Frontend des Import-Wizards wurde anhand von vorher angefertigten Mockups in Figma implementiert. Durch das Zusammensetzen der UI-Oberfläche aus einzelnen Komponenten wurde eine modulare und erweiterbare Benutzeroberfläche geschaffen. Umfangreiche Validierungslogiken und ein strukturiertes Logging sorgten für einen robusten und nahvollziehbaren Importprozesses. Distributed Tracing mit OpenTelemetry wurde schichtübergreifend integriert.

## Ergebnis

Das Ergebnis des Projekts ist ein funktionsfähiger und benutzerfreundlicher Import-Wizard, der es Versicherungsberatern ermöglicht, ihre Kundendaten aus dem Legacy System ANGEBOTsoftware per Knopfdruck nahtlos in die neue OVWEB Applikation zu übertragen.

In diesem zweiten Teil des IPA-Berichts zeige ich die Umsetzung des Projektes mit dem IPERKA Modell auf. Die verschiedenen Phasen dienen als Orientierung für die Struktur des weiteren Berichtes.

# Informieren

## Ziele der Aufgabenstellung

Ziel ist es, Legacy Daten in Form von Excels (.xls Dateien) mittels einem Import Wizard in das OVWEB übernehmen zu können.

## Vorgaben

Es sollten doppelte Einträge vermieden werden durch Überprüfung der existierenden Kundedaten anhand folgender Kriterien:

* Nachname
* Vorname
* Geburtsdatum

Ist der Kunde bereits erfasst, sollte sein Datensatz lediglich aktualisiert werden.

## Ist-Zustand

Die Offerten-Verwaltung Web, kurz OVWEB, wird dazu dienen die Cloudfähigkeit der alten Applikation aufzuzeigen. Die Idee der Applikation ist, Offerten zu erstellen und Kunden zu verwalten. Der aktuelle Stand der Seite erlaubt das Login via Keycloak und das Erstellen eines Kunden via ein Formular. Um nicht jeden Kunden einzeln im neuen Tool erfassen zu müssen, gibt es ein grosses Interesse, bestehende Kunden via einen Export im alten Tool in das neue Tool importieren zu können. An dieser Anforderung der Berater setzte ich an, um den Transfer so einfach wie möglich zu gestalten. Ansätze zur Kunden-Verwaltung gibt es ebenfalls.

### Frontend

Das Frontend wurde mit Next.js 14 umgesetzt und nach dem Generali-Styleguide

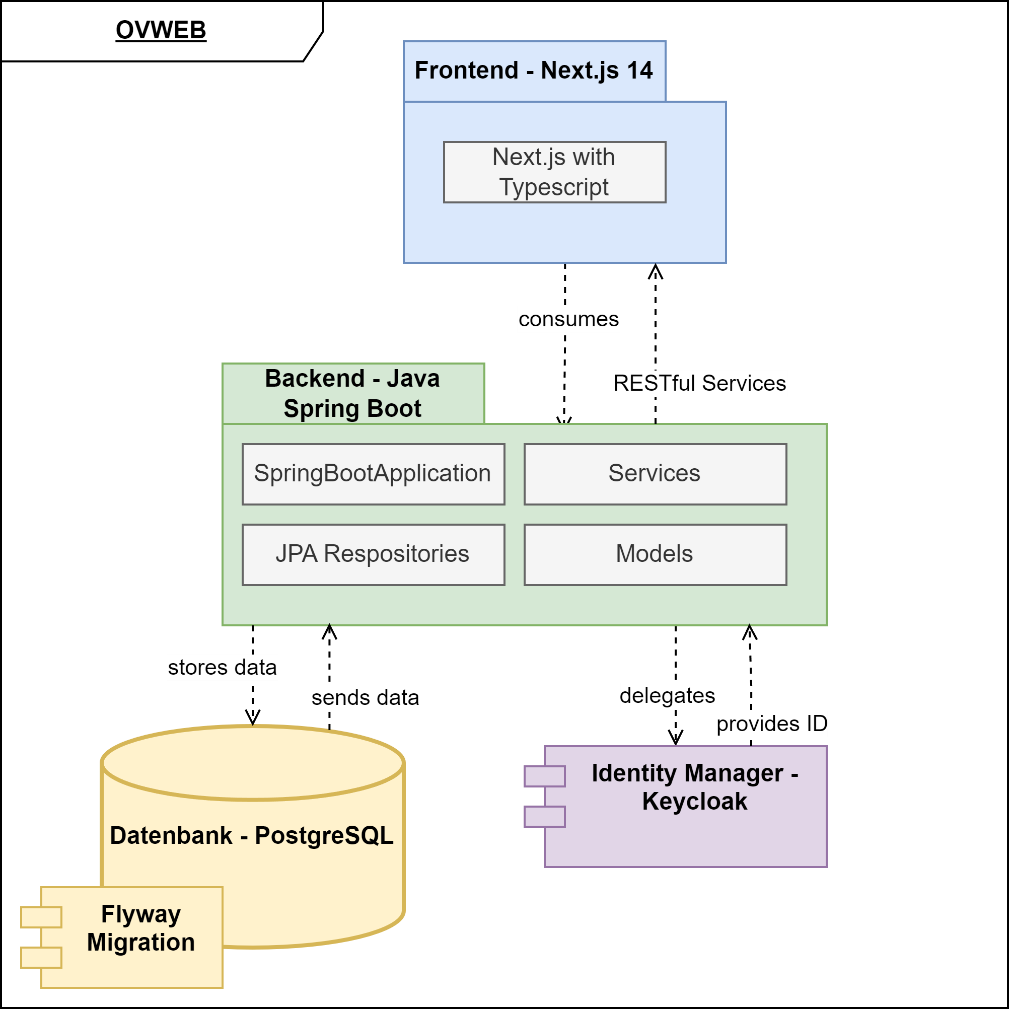
gestaltet. Das Sessionhandling wird mit NextAuth geregelt. Als Component Library

wird ShadCn eingesetzt. Es wird ein TypeScript-Client für das Backend verwendet.

### Backend

Das Backend wurde mit Springboot aufgebaut. Ein simples Model wurde integriert, das User, Customer und ihre Adressen beinhaltet. Flyway versioniert und initialisiert die Datenbank-Anbindung, die lokal via einem Docker-Container läuft. Als Query Sprache wird PostgreSQL verwendet.

### Überblick

Die folgende Abbildung gibt einen konzeptionellen Überblick über die Kernarchitektur der OVWEB-Anwendung. Sie konzentriert sich auf die wesentlichen Bestandteile.

### Ein Bild, das Text, Screenshot, 3D-Modellierung, Grafiksoftware enthält. Automatisch generierte BeschreibungKlassendiagramme

Model Klassendiagramm im IST-Zustand

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte BeschreibungFür das Customer und User Model gibt es Service und Controller für CRUD-Operationen

Service Klassendiagramm im IST-Zustand

### Datenbankdiagramm

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte BeschreibungDie Datenbank wird durch Flyway verwaltet und enthält die gleichen Entitäten, wie das Model im Backend

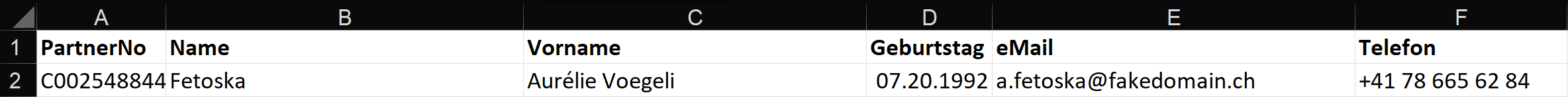
### Systemgrenzen

Der Auftrag kann in der vorhandenen Architektur der «OVWEB» eingebunden werden und braucht lediglich den Export der Kundendaten als Excel-Datei vorliegend. Für den Umfang der IPA werden diese Daten bereitgestellt und daher erfordert es keine Interaktion mit der Legacy-Applikation.

## Analysieren der Probe-Excels

Beim Sichten der zur Verfügung gestellten drei Exceldaten «**AngebotSoftware\_Export\_TC001.xls**» (gute Daten), «**AngebotSoftware\_Export\_TC002.xls**» (gute Daten + doppelte Einträge), «**AngebotSoftware\_Export\_TC003.xls**» (Kopie von TC001 mit inkonsistenten Daten) sind mir folgende Punkte aufgefallen:

* Es wird das alte Dateiformat «.xls» verwendet
* Die Spalte «PartnerNo» ist nicht in unserem Model vorhanden und muss ergänzt werden
* Die Formatierung des Geburtsdatums muss eventuell konvertiert werden
* Bei den nicht konsistenten Daten gibt es im «Addresses»-Sheet falsche «PartnerNo»

Es gab zwei kleinere Schreibfehler in den Excels, die im Verlauf der IPA in Absprache mit dem Experten. Herr Kaufmann, korrigiert wurden.

Beispielzeile des «Addresses»-Sheet

Beispielzeile des «Customers»-Sheet

Diese Erkenntnisse helfen mir eine saubere Planung des Import-Wizards für diese und ähnliche Datensätze zu gestalten.

## Aktivitätsdiagramm

Um den Prozess zu veranschaulichen, bietet sich das Aktivitätsdiagramm an. Es gibt einen Überblick des Verlaufs und kann als Grundlage dienen um mit allfälligen Stakeholdern sicher zu sein, dass man vom Gleichen redet.Ein Bild, das Text, Screenshot, Software, Website enthält.

Automatisch generierte Beschreibung

## User Stories

Um ein Verständnis der Anforderungen zu gewinnen, werden User Stories erfasst. Der Kern liegt in den Anforderungen der Berater und der Umsetzung der neuen Import-Funktion, jedoch habe ich mir die Freiheit genommen, IPA relevante Kriterien ebenfalls in User Stories zu verpacken. So gewinne ich einen guten Überblick, was zu erledigen ist und kann beim Testen gleich überprüfen, ob auch die Entwicklerthemen richtig umgesetzt wurden.

### Rollen

Es gibt Versicherungsvertreter, Administratoren und Entwickler.

### Funktionale Anforderungen

|  |
| --- |
| **Story 1** |
| Als Versicherungsberater möchte ich meine Kundendaten aus dem alten "Angebot Software" System in das neue "OVWEB" System importieren können. |
| **Akzeptanzkriterien** |
| **AC 1.1**: Der Berater kann einen Workflow im OVWEB starten, um die Daten zu importieren.  **AC 1.2**: Das System validiert das Excel-Format und importiert die Daten korrekt in die Customer-Tabellen von OVWEB.  **AC 1.3**: Das System verhindert den Import von Duplikaten, indem es vorhandene Kunden anhand von Nachnamen, Vorname und Geburtsdatum identifiziert und aktualisiert. |

|  |
| --- |
| **Story 2** |
| Als Versicherungsberater möchte ich eine Übersicht über den Import erhalten, um nachzuvollziehen, was der Import-Wizard gemacht hat. |
| **Akzeptanzkriterien** |
| **AC 2.1**: Nach dem Import wird eine Summary-Seite angezeigt, die dem Berater anzeigt, wie viele Kundendaten erfolgreich importiert wurden und wie viel aktualisiert wurden. |

|  |
| --- |
| **Story 3** |
| Als Versicherungsberater möchte ich in der OVWEB-Navigation auf die Import-Funktion zugreifen können. |
| **Akzeptanzkriterien** |
| **AC 3.1**: Die Import-Funktion ist in der Hauptnavigation von OVWEB integriert und leicht zugänglich. |

### Nichtfunktionale Anforderungen

|  |
| --- |
| **Story 4** |
| Als Versicherungsberater möchte ich, dass der Import-Wizard benutzerfreundlich ist, damit ich meine Daten selbstständig importieren kann. |
| **Akzeptanzkriterien** |
| **AC 4.1**: Der Import-Wizard verwendet eine klare und verständliche Sprache, die für nicht-technische Benutzer geeignet ist.  **AC 4.2**: Im Falle eines Fehlers während des Imports erhält der Berater klare Hinweise. |

|  |
| --- |
| **Story 5** |
| Als Entwickler möchte ich, dass alle neuen Funktionalitäten des OVWEB-Systems durch automatisierte Tests abgedeckt sind. |
| **Akzeptanzkriterien** |
| **AC 5.1**: Für die neue Funktion gibt es automatisierte Unit-Tests, die sowohl positive als auch negative Szenarien abdecken.  **AC 5.2**: Automatisierte Tests für null und non-null Datensätze, sowie für leere und nicht-leere Kundenlisten sind implementiert und verifizieren die Handhabung dieser Fälle. |

|  |
| --- |
| **Story 6** |
| Als Administrator möchte ich, dass das System strukturiertes Logging implementiert, um die Nachverfolgung von Aktivitäten und die Fehlerbehebung zu erleichtern. |
| **Akzeptanzkriterien** |
| **AC 6.1**: Strukturiertes Logging ist durchgehend implementiert.  **AC 6.2**: Die Logs enthalten alle notwendigen Informationen, um die Aktionen nachvollziehen zu können.  **AC 6.3**: Persönlich identifizierbare Informationen (PII) werden nicht in den Logdateien ausgegeben. |

|  |
| --- |
| **Story 7** |
| Als Entwickler möchte ich, dass jede Anfrage mit einer Trace-ID nachverfolgt werden kann, um die Fehlersuche zu vereinfachen. |
| **Akzeptanzkriterien** |
| **AC 7.1**: Jede Anfrage im Frontend generiert eine eindeutige Trace-ID, die an das Backend weitergeleitet wird.  **AC 7.2**:Die Trace-ID wird konsistent über alle beteiligten Dienste hinweg weitergeleitet und in deren Logs festgehalten.  **AC 7.3**: Es ist möglich, ein Diagramm der Service-Aufrufe basierend auf den Trace-IDs zu erstellen. |

|  |
| --- |
| **Story 8** |
| Als Entwickler möchte ich, dass die UI-Komponenten des Systems wiederverwendbar sind. |
| **Akzeptanzkriterien** |
| **AC 8.1**: UI-Komponenten sind so strukturiert, dass komplexere Komponenten aus einfacheren zusammengesetzt sind.  **AC 8.2**: Komponenten sind im Storybook dokumentiert. |

# Planen

Das Planen ist die zweite Phase der IPERKA-Methode.

Hier werde ich aufzeigen, was die nötigen Schritte sind, um aus dem Ist-Zustand zum Soll-Zustand zu kommen.

## Soll-Zustand

Vereinfach besteht der Prozess aus drei Schritten: Einführung, Verarbeitung der Daten und Abschlussbericht. Bei einem Fehler wird der Berater aufgefordert die Generali zu kontaktieren.

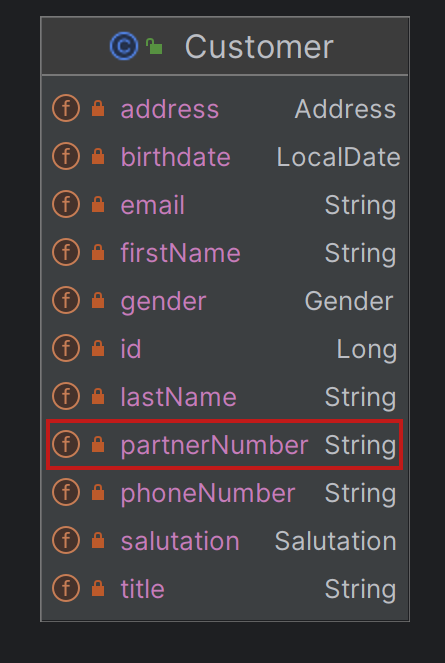
### Frontend

* Integrieren eines Links zum Import Wizard in die Hauptnavigation
* Erstellung einer Einführungsseite für den Import-Wizard, die den Prozess und die nächsten Schritte erklärt
* Entwicklung einer Upload-Komponente, für das Hochladen von Excels
* direktes Feedback über den Status der hochgeladenen Datei geben via Ladeanimation. Bei Fehlern -> Hinweis zur Kontaktaufnahme Generali AG
* Design und Implementierung einer Summary-Seite, die nach dem Import die Anzahl der neuen und aktualisierten Kundendaten anzeigt
* Anzeigen von Fehlermeldungen

### Backend

* Services zur Verarbeitung des Imports und Speicherung der Daten in den Customer-Tabellen programmieren
* Validierung im Backend
* Logik für Erkennung bereits vorhandenen Kunden
* Aufbau strukturiertes Logging
* Implementierung von Distributed Tracing

### Klassendiagramm

Für die Implementierung des Import-Wizards und die damit verbundene Speicherung von Kundendaten aus der alten «ANGEBOT Software», ist eine Anpassung des Models «Customer» notwendig. Die Spalte «PartnerNo» in den zu Verfügung gestellten Excels ist nicht nur wichtige Information über den Kunden, sondern dient als Primärschlüssel, um die beiden Excel-Sheets zu verbinden. Das Model im Backend soll um das Attribut «partnerNumber» erweitert werden, um die Identifikationsnummer des alten Tools speichern zu können.

Erweiterung des «Customer» Models

### Datenbankanpassungen

Die Datenbank muss wie das Java Model in der «Customer» Tabelle angepasst werden. Die Spalte «partner\_number» wird via einer Flyway Migration dem Schema hinzugefügt.

## Mockups

Für die Planung des User Interfaces habe ich in Figma zwei Varianten umgesetzt. Ziel dieser Mockups ist es einen visuellen Überblick zu verschaffen, was alles umzusetzen ist und die User Experience so gut wie möglich aufzuzeigen. Die Mockups bestehen aus den drei Seiten bzw. Schritte des Importwizards: Einführung, Verarbeitung und Abschluss. Zusätzlich kommt eine Fehlerseite, die bei falschen Formaten oder inkonsistenten Daten angezeigt wird.

### Variante 1

Zusammenfassung Variant 1

Fehlerseite Variant 1

Einführungsseite Variant 1

Verarbeitung Variant 1

### Variante 2

Verarbeitung Variant 2

Einführungsseite Variant 2

Zusammenfassung Variant 2

Fehlerseite Variant 2

Die beiden Varianten unterscheiden sich durch die Platzierung des Erklärtextes und des Import-Wizards. Bei der ersten Variante ist das Import-Model zu oberst. Es legt Gewicht auf die Import-Funktion und bietet dem Berater diese sofort an. Bei dem wiederholten Besuch der Seite, ist der Prozess bekannt und die Erklärung wird ignoriert. Diese Variante bietet das Wichtigste zuerst: Die Exportdatei auszuwählen und in das OVWEB einzulesen. Bei der zweiten Varianten wird der Berater bei jedem Besuch kurz durch den Prozess geführt. Am Ende der Einführung wartet der Import-Wizard am unteren Bereich der Seite. Bei Unsicherheiten muss der Berater nur kurz hinaufsehen bzw. scrollen, um die Anleitung zu konsultieren.

## Testkonzept

Es wird mit dem folgenden Testkonzept überprüft, ob die Arbeit korrekt umgesetzt wurde. In der fünften IPERKA Phase «Kontrollieren» werden die Testfälle dann getestet. Das Konzept baut stark auf den definierten Akzeptanzkriterien in den User Stories auf.

### Testziele und -strategie

* Ziel ist es, die Funktionalität, Validierung und Benutzerfreundlichkeit des Import-Wizards zu testen
* Ich verwende, wo möglich, Unit-Tests und decke die restlichen Kriterien mit manuelle Akzeptanztests ab

### Testumgebung

Die Tests werden innerhalb der Generali AG mit dem Google-Chrome-Browser auf der lokalen Umgebung getestet. Die Springboot-Applikation, sowie das Next.Js Frontend wird gestartet und via Docker eine Instanz der neusten Version der Datenbanken und Keycloak hochgefahren. So baue ich eine Umgebung auf, die so gut wie möglich eine Produktionsumgebung widerspiegelt.

### Testfälle

Die Story 5 und die dazugehörigen Akzeptanzkriterien sollten durch Unit-Tests automatisch getestet werden. Daher ist es an dieser Stelle nicht notwendig, einen Testfall anzulegen.

Alle weiteren Anforderungen werden mit manuellen Akzeptanztests überprüft.

Die folgenden Testfälle ergeben sich direkt aus den User Stories bzw. den Akzeptanzkriterien kurz AC.

|  |
| --- |
| **Testfall 1.1** |
| **Anforderung**: Story 1, AC 1.1  **Beschreibung**: Überprüfung, ob Berater-Import-Workflow starten kann  **Voraussetzung**: Benutzer ist im OVWEB-System als Versicherungsberater angemeldet  **Eingabe:** Benutzer navigiert zum Import-Bereich |
| **Erwartetes Resultat** |
| Der Import-Workflow wird durch das Auswählen erfolgreich gestartet. |

|  |
| --- |
| **Testfall 1.2** |
| **Anforderung**: Story 1, AC 1.2  **Beschreibung**: Validierung der Excel-Datei und Import der Kundendaten.  **Voraussetzung**: Eine Excel-Datei im korrekten Format ist bereit zum Upload.  **Eingabe:** Hochladen der Excel-Datei durch den Benutzer. |
| **Erwartetes Resultat** |
| 1. Der Wizard erkennt das Format der Excel-Datei und bestätigt die Validierung. 2. Die Daten werden korrekt in die Customer-Tabellen von OVWEB importiert. |

|  |
| --- |
| **Testfall 1.3** |
| **Anforderung**: Story 1, AC 1.3  **Beschreibung**: Verhinderung des Imports von Duplikaten.  **Voraussetzung**: Eine Excel-Datei mit Daten, die Duplikate enthält.  **Eingabe:** Hochladen der Excel-Datei durch den Benutzer. |
| **Erwartetes Resultat** |
| Der Wizard erkennt die Duplikate und aktualisiert die Einträge. |

|  |
| --- |
| **Testfall 2.1** |
| **Anforderung**: Story 2, AC 2.1  **Beschreibung**: Anzeige der Summary-Seite nach dem Import.  **Voraussetzung**: Erfolgreicher Abschluss des Importvorgangs.  **Eingabe:** - |
| **Erwartetes Resultat** |
| Die Summary-Seite zeigt die Anzahl der erfolgreich importierten und aktualisierten Kundendatensätze an. |

|  |
| --- |
| **Testfall 3.1** |
| **Anforderung**: Story 3, AC 3.1  **Beschreibung**: Zugänglichkeit der Import-Funktion in der Hauptnavigation.  **Voraussetzung**: Benutzer ist im OVWEB-System angemeldet.  **Eingabe:** - |
| **Erwartetes Resultat** |
| Die Import-Funktion ist sichtbar und kann von der Hauptnavigation aus erreicht werden. |

|  |
| --- |
| **Testfall 3.1** |
| **Anforderung**: Story 3, AC 3.1  **Beschreibung**: Zugänglichkeit der Import-Funktion in der Hauptnavigation.  **Voraussetzung**: Benutzer ist im OVWEB-System angemeldet.  **Eingabe:** - |
| **Erwartetes Resultat** |
| Die Import-Funktion ist sichtbar und kann von der Hauptnavigation aus erreicht werden. |

|  |
| --- |
| **Testfall 4.1-4.2** |
| **Anforderung**: Story 4, AC 4.1-4.2  **Beschreibung**: Benutzerfreundlichkeit und Fehlerbehandlung im Import-Wizard.  **Voraussetzung**: Mehreres Ausführen des Imports  **Eingabe:** Korrekte und inkorrekte Excel-Dateien werden vom Benutzer hochgeladen. |
| **Erwartetes Resultat** |
| Der Wizard verwendet eine klare Sprache und gibt bei Fehlern klare Hinweise. |

|  |
| --- |
| **Testfall 6.1-6.2** |
| **Anforderung**: Story 6, AC 6.1-6.2  **Beschreibung**: Überprüfung des strukturierten Loggings.  **Voraussetzung**: Importvorgang wurde durchgeführt.  **Eingabe:** - |
| **Erwartetes Resultat** |
| Logs sind strukturiert und enthalten alle notwendigen Informationen ohne PII. |

|  |
| --- |
| **Testfall 7.1-7.3** |
| **Anforderung**: Story 7, AC 7.1-7.3  **Beschreibung**: Nachverfolgbarkeit der Anfragen mittels Trace-ID.  **Voraussetzung**: Mehrere Anfragen wurden vom Frontend initiiert.  **Eingabe:** - |
| **Erwartetes Resultat** |
| 1. Jede Anfrage hat eine eindeutige Trace-ID, die in allen beteiligten Services propagiert ist. 2. Ein Diagramm der Service-Aufrufe kann erstellt werden. |

|  |
| --- |
| **Testfall 8.1-8.3** |
| **Anforderung**: Story 8, AC 8.1-8.2  **Beschreibung**: Wiederverwendbarkeit der UI-Komponenten.  **Voraussetzung**: UI-Komponenten sind vorhanden.  **Eingabe:** - |
| **Erwartetes Resultat** |
| Komplexere Komponenten setzen sich aus einfacheren zusammen und sind im Storybook dokumentiert. |

# Entscheiden

Im Anschluss der Planung gilt es sich für eine Lösungsvariante zu entscheiden. Es wurden zwei Variationen als Mockups im Figma umgesetzt. In diesem Kapitel möchte ich die beiden Optionen vergleichen und eine definitive Entscheidung treffen, welche umgesetzt wird. So kann ich mit einem klaren Bild in die Realisierung übergehen.

## Ein Bild, das Text, Screenshot, Schrift, Design enthält. Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Design enthält. Automatisch generierte BeschreibungLösungsvarianten im Vergleich

Einführungsseite Variant 1

Einführungsseite Variant 2

Hier sind die beiden Einführungsseiten nochmals zur Veranschaulichung.

Die Versionen sind sich ähnlich. Der Unterscheid liegt in der Gewichtung des Import-Wizards auf der Seite. Soll er die grösste Gewichtung haben, noch mehr als der Info-Text? Bei der ersten Variante ist der Import-Wizard zuoberst. Ich sehe das Argument, das ein versierter Benutzer sofort von der Funktion Gebrauch machen will und nach mehreren Imports die Erklärung überspringt. Jedoch ist einer der zentralsten Anforderungen die Benutzerfreundlichkeit und das selbständige Verwenden ohne technischen Hintergrund. So bietet sich die zweite Version an. Zuerst die Erklärung und dann die Anwendung. Auch während dem Verwenden des Wizard kann der Berater so die Anleitung nochmals kurz konsultieren mit einem kurzen Blick nach oben.

Die Wahl fällt somit auf die Variante 2, da sie den Benutzer sofort bei der Hand nimmt und ihn durch die Applikation führt.

# Realisierung

## Anpassungen in der Datenbank und Model

Um die Daten richtig persistieren zu können, sind Anpassungen in der Datenbank und im Model nötig. Mit Hilfe von Flyway wurden Datenbankanpassungen an der Tabelle **«customer»** vorgenommen. Es wurde eine neue Spalte hinzugefügt: **«partner\_number»** vom Typ varchar(255).

Hierfür wurde die Datei **«V1\_0\_10\_\_add\_partner\_number\_to\_customertabel.sql»** erstellt.Flyway erleichtert das Versionsmanagement der Datenbankstruktur, indem es SQL-Skripte wie diese auf automatisierte Weise während des Hochfahrens der Applikation ausführt.

Wie in der Planung erwähnt, wurde das Attribut «partnerNumber» dem «Customer» Model hinzugefügt.

## Erstellen eines Import Services

Ich entscheide mich, einen neuen Import Service anzulegen. Der Code ist so getrennt von den CRUD-Operationen des Customer Services. Für das Verarbeiten der Excel- Daten verwende ich die Apache-POI Libraries. Apache bietet diverse Klassen und Methoden, um Excels und ähnliche Spreadsheets zu verarbeiten. Auch wenn dies nicht in den Anforderungen war, schrieb ich zuerst eine Hilfsmethode, die zwischen .xls und .xlsx, dem neueren auf XML basierendem Excel-Format, unterscheidet und sie richtig verarbeitet. Während der Implementierung des Services zum Lesen und Speichern der Kundendaten in den Excel-Dateien, arbeitete ich eng mit den mir zur Verfügung gestellten Testdaten. Bei den numerischen Formaten, wie zum Beispiel der Postleitzahl, entschied ich mich, die Datentypen vorher zu überprüfen, anstatt einfach anzunehmen das die Excels mir immer konsistente und richtige Datentypen liefern. So kann der Service auch bei String-Werten in diesen Feldern, die Daten verarbeiten.

Die Hauptmethode der Klasse **«ImportService.java»** ist die **importData**-Methode. Sie gibt eine Liste aller erstellten Kunden zurück, bei Fehlern eine leere Liste zurück. Sie ist mit einer Annotation als transaktional deklariert worden, damit sie auch bei Fehlern auf der aller letzten Zeile der importierten Excel alle Speicherungen von Kundendaten wieder rückgängig macht.

Die Methode **«saveOrUpdateCustomer»** ist nicht nur für das richtige Speichern zuständig, sondern auch für dir Kontrolle, ob ein Kunde bereits in der Datenbank existiert. Dank JPA-Repository kann ich im Interface **«CustomerRepository.java»** bequem eine neue Methode **«findByLastNameAndFirstNameAndBirthdate»** schreiben und sie verwenden, um zu überprüfen, ob ein Datensatz neu angelegt wird oder aktualisiert werden muss. Während des Speichervorgangs wird das Geschlecht und die Anrede hardcodiert, da diese Informationen nicht aus den Excels bezogen werden kann.

Alle Methoden, die für das Einlesen und Verarbeiten der Excel-Tabellen verantwortlich sind, wurden in den Service **«ExcelProcessingService.java»** ausgelagert. Dadurch wird der Importprozess übersichtlicher und die Logik für die Verarbeitung der Excel-Dateien steht zur Wiederverwendung zur Verfügung.

Im **«CustomerController.java»** wurde ein neuer Endpunkt angelegt für das Importieren von Kunden. Ein DTO mit dem Namen **«ImportResult.java»** wurde angelegt, um dem Frontend alle nötigen Informationen für den Import-Wizard mitzugeben.

public class ImportResult {  
  
 private int addedCount;  
 private int updatedCount;  
 private List<String> errors;  
 private List<Customer> importedCustomers;  
}

ImportResult.java im DTO-Ordner

## Backend Validierung

Die Validierung im Backend wird durch einige private Methoden im **«ImportService.java»** sowie **«ExcelProcessingService.java»** gelöst. Ich habe mich hauptsächlich auf drei Kontrollen konzentriert:

* Aufbau der Excels (Naming der Sheets und Spalte)
* Datenkonsistenz
* Allgemeines «Exceptionhandling»

Die Methoden **«validateWorkbookStructure»** und **«validateSheetStructure»** sorgen dafür, dass der richtige Name für die beiden Sheets «Customers» und «Addresses» vergeben wurden. Bei falschen Strukturen in der Excel können wir so gleich einen Fehler zurückgeben.

Die Datenkonsistenz wird mit der **«validateAddresses»-**Methode sichergestellt. Bevor wir in die Datenbank schreiben, wird eine Kontrolle gemacht, ob es Adressen ohne zugehörigen Kunden gibt. Alternativ wäre es möglich, diese Probe nach dem Speichern durchzuführen und bei Bedarf die Transaktion wieder rückgängig zu machen, aber ich entschied mich, keine unnötigen Schreibarbeiten auf der Datenbank auszulösen, falls es inkonsistente Daten in den Importdaten gab.

Weiter wurde an diversen Stellen allgemeine Catch-Blöcke für Exceptions eingebaut, zum Beispiel wenn Apache POI beim Auslesen der Daten einen Fehler wirft.

## Import-Wizard im Frontend

Das Gesicht des Import-Wizards ist das User-Interface. Es sollte benutzerfreundlich sein und gleichzeitig den sauberen Import der Daten sicherstellen.

### Statische Elemente

Der statische Einführungstext wurde in der **«page.tsx»** hineingepflegt und nach dem Mockup gestaltet. Es wurde mit dem «dangerouslySetInnerHTML» gearbeitet, die React Version von innerHTML, um gewisse Details wie die Grösse und Dicke der Schrift so genau wie möglich umzusetzen.

### Upload-Modul

Das Upload-Modul wurde als separate Komponente umgesetzt. Unter dem **«component»-Ordner** gibt es neu den wiederverwendbaren **«FileUploader.tsx»**. Sie verwaltet verschiedene Zustände des Prozesses. Von Auswahl der Daten, Validierung des Formats bis hin zu Fehlerzuständen und Anzeigen des Ergebnisses des Importvorgangs.

Die **«uploadFile»**-Server-Action überträgt die Daten an das Backend. Die Serverkomponente wird gleich nach dem Auswählen von einer gültigen Exceldatei ausgelöst. Kommt es zu einem Fehler oder sind die Daten inkonsistent wird der Benutzer aufgefordert mit Generali Kontakt aufzunehmen, um das Problem zu beheben.

Es gab Probleme bei der Übertragung der Formdaten an den Backend-Service. Mit der **«fetchMultipartFix»-**Methode, die ich von Roman erhalten habe, konnte ein Workaround gefunden werden. Sie passt die Request-Header an und wandelt den FormData-Körper in einen lesbaren Stream um, sodass er von der Serverseite korrekt verarbeitet werden kann.

Um den Benutzer darüber zu informieren, dass der Importvorgang gestartet ist, wird eine simple Ladeanimation angezeigt. Die Animation ist in die Komponente **«LoadingAnimation.tsx»** ausgelagert. Laut Mockup war eine Art Fortschrittsbalken geplant, jedoch wurde davon in dieser ersten Version abgesehen.

### Validation

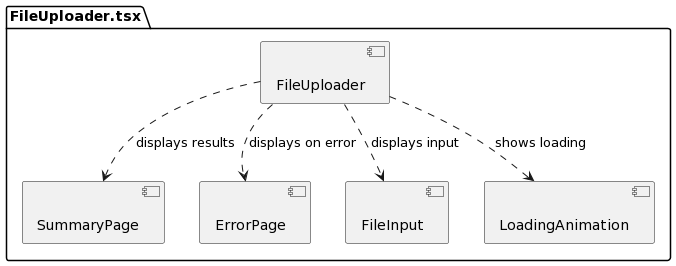
Die Validation im Frontend beschränkt sich auf die Kontrolle des Datentyps. Dies passiert im Verlauf der **«handleFileChange»**-Methode. Sobald diese verifiziert ist, wird die Datei für weitere Kontrollen dem Backend übergeben.

### Summary-Seite

Nach einem erfolgreichen Upload wird die **«Summary.tsx»**-Page angezeigt, der Informationen über die Anzahl der hinzugefügten und aktualisierten Kunden enthält. Zwei Buttons am Ende der Seite bieten an, den Vorgang wieder von vorne zu starten oder die importierten Kunden in der Kundenübersicht anzuschauen.

### Übersicht

Zusammenfassend zeigt der FileUploader je nach Zustand eine unterschiedliche Seite an.



Komponentendiagramm des FileUploader.tsx

## Sequenzdiagramm

Das Sequenzdiagramm dient der Übersicht für die diversen Methoden und Funktionen des Import-Wizards im Front- und Backend

Ein Bild, das Text, Diagramm, parallel, Screenshot enthält.

Automatisch generierte Beschreibung

## Distributed Tracing

Das Tracing wird im OVWEB von OpenTelemetry verwaltet. Um einen neuen Span im Backend auszulösen, kann die Annotation **«@NewSpan()»** verwendet werden. Um im Frontend das Tracing zu starten und Logs einem Span zuzuordnen, wird der http-Request mit einem **«context»** von der OpenTelemtry-API umhüllt. Im «finally»-Block der Funktion wird der Span geschlossen.

Mit der **«fetchMultipartFix»**-Methode wird sichergestellt, dass der gleiche Parent-Trace im Backend verwendet wird und so ein durchgehendes Bild der Anfrage erstellt werden kann. Hier ein Beispiel aus Grafana:

Diagramm von ImportService-Aufruf

Ein Bild, das Screenshot, Multimedia-Software, Software, Text enthält.

Automatisch generierte Beschreibung

## Logging

Für das strukturierte Logging wird im Frontend das Custom-Packet **«log-util»** und im Backend **«SLF4J»** verwendet. Mit einer Key-Value-Struktur können wir nach den Logs innerhalb der Traces bzw. Spans suchen und sie so für Debuggingarbeiten verwenden. Hier noch ein Beispiel für die Struktur im Backend:

*log*.atInfo()  
 .setMessage("Füge neuen Kunden hinzu")  
 .addKeyValue("partnerNumber", customer.getPartnerNumber())  
 .log();

## Storybook

Jede UI-Komponente wurde in Storybook erfasst. Es ermöglicht Entwicklern, Komponenten unabhängig von der restlichen Applikation zu visualisieren, zu testen und zu dokumentieren. Die Server-Action **«uploadFile»** wird gemocked. So lassen sich die umgesetzten Elemente isoliert betrachten, ohne einen Zustand manuell auslösen zu müssen. Die Weiterentwicklung wird dadurch wesentlich angenehmer.

# Kontrollieren

## Testing

### Testkonzept

Auf das Testkonzept wird im [Kapitel 8.3 «Testkonzept»](#_Testkonzept) vertieft eingegangen.

### Testprotokoll

Angaben PC & Software:

|  |  |
| --- | --- |
| **Name** | Lenovo ThinkBook 15 G4 ABA |
| **Prozessor** | AMD Ryzen 7 5825U, 2000 MHz, 8 Kerne, 16 logische Prozessor |
| **Ram** | 16 GB |
| **Testumfeld / Browser** | Google Chrome Version Version 121.0.6167.161 (Offizieller Build) (64-Bit) |
| **Betriebssystem** | Microsoft Windows 11 Pro |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Testfall** | **Resultat** | **Datum** | **Tester** | **Bemerkung** |
| **1.1** | Passed | 15.02.2024 | R.Blaauw |  |
| **1.2** | Passed | 15.02.2024 | R.Blaauw | Bestätigung durch Starten des Verarbeitungsprozesses |
| **1.3** | Passed | 15.02.2024 | R.Blaauw |  |
| **2.1** | Passed | 15.02.2024 | R.Blaauw |  |
| **3.1** | Passed | 15.02.2024 | R.Blaauw |  |
| **4.1-4.2** | Passed | 15.02.2024 | R.Blaauw | Klarere Fehlermeldungen in Zukunft |
| **6.1-6.2** | Passed | 15.02.2024 | R.Blaauw | Wichtigste Info ausgegeben, erweiterbar |
| **7.1-7.3** | Passed | 15.02.2024 | R.Blaauw |  |
| **8.1-8.3** | Passed | 15.02.2024 | R.Blaauw |  |

### Testbericht

Das Testprotokoll zeigt, dass alle definierten Testfälle erfolgreich bestanden wurden.

Die Applikation reagierte wie erwartet und es traten keine Fehler auf. Der Einsatz der

Anwendung ist somit ohne Bedenken möglich.

Bemerkungen zu Tests:

1. Kein **Testfall 5.1-5.2**, diese werden durch Unittests abgedeckt
2. **Testfall 1.2 (Validierung und Import der Kundendaten):**

* *Bemerkung*: Die Validierung wird implizit mit dem Start des Importvorgangs und dem Wechsel zur Ladeanimation bestätigt.
* *Massnahmen/Empfehlungen*: Bei Bedarf kann eine explizite Meldung ausgegeben werden, dass die Exceldatei dem richtigen Format entspricht.

1. **Testfall 4.1-4.2 (Benutzerfreundlichkeit und Fehlerbehandlung):**

* *Bemerkung*: Die von der IPA-Aufgabe übernommene Fehlermeldung, dass Generali kontaktiert werden sollte, könnte präziser sein.
* *Massnahmen/Empfehlungen*: In Betracht ziehen, diese Fehlermeldung durch eine genauere zu ersetzen.

1. **Testfall 6.1-6.2(strukturierten Loggings):**

* *Bemerkung*: Die wichtigsten Hinweise werden ausgegeben.
* *Massnahmen/Empfehlungen*: Bei Bedarf können noch mehr Informationen über den importierten Kunden oder Fehlermeldungen geloggt werden.

## Unittests

Für die Akzeptanzkriterien der Story 5 wurden automatisierte Unit-Test für die zwei zentralen Service Klasse **«ImportService.java»** und **«ExcelProcessingService.java»** erstellt. Einer der wichtigsten Tests ist die Überprüfung der Methode «**importData»**. Dabei wird sichergestellt, dass Kundendaten korrekt gesammelt, verarbeitet und in die Datenbank gespeichert werden, ohne dass Fehler auftreten.

# Auswerten

## Vergleich Ist / Soll

Das OVWEB ist um einen Import-Wizard für Legacy Daten erweitert worden. Die Implementierung bietet einen einfachen Weg, Kundendaten vom alten in das neue Tool einzulesen. Weiter wurde der Codestack im Frontend sowie im Backend mit Klassen ergänzt, die wiederverwendbar sind, falls eine ähnliche Aufgabe in Zukunft anstehen.

## Persönliches Fazit zur IPA

Im Allgemeinen blicke ich auf eine intensive, aber erfolgreiche Zeit zurück. Ich konnte gut in die IPA einsteigen, da ich dank einer bereits absolvierten Probe-IPA den Ablauf im Kopf hatte. Die bevorstehenden Tasks konnte ich mir schnell vorstellen und den ungefähren Aufwand abschätzen. In der Analyse und Informationsphase musste ich viel Flexibilität zeigen und immer wieder von einer Evaluierung von z.B. den Testdaten zu der Dokumentation hin- und herwechseln. Das war nicht ganz einfach. Aber bei der Planung konnte ich mich voll mit der Dokumentation auseinandersetzten und mein Konzept auf Papier bringen. Das geling mir sehr gut. Mit diesem Plan und dem parallel erstellten Zeitplan fiel es mir leicht die Arbeiten anzugehen. Ich konnte mich am Morgen von jedem IPA-Tag an dem Zeitplan orientieren, was ansteht. Es gab einige Stolpersteine, die ich zunächst allein anging, aber wo nötig auch Hilfe von Roman geholt habe. Ich bin sehr zufrieden, wie fokussiert ich während der Realisierungsteil der IPA sein konnte. Das half natürlich sehr beim Vorankommen. Beim Testing konnte ich dank einem Testkonzept, das auf den User Stories aufbaut, schnell kontrollieren, was noch Verbesserungspotenzial hat. Nach der Kontrolle hatte ich einen guten Überblick, was alles erreicht wurde. Ich war sehr zufrieden, dass ich die Kernaufgabe vollständig implementieren konnte.

### Was war gut?

Meine Aufgaben konnte ich immer gut nach Zeitplan erledigen und es fiel mir leicht, motiviert bis zum Feierabend daran zu arbeiten. Die Dokumentation und das Arbeitsjournal konnte ich immer aktuell behalten. Wenn nicht bis Tagesende, als erstes am nächsten Morgen. Ich glaube, mein Fokus auf die IPA und das Priorisieren mit meinem Kanban-Board half mir gut voranzukommen und nur wenig Ablenkungen zuzulassen.

### Was lief weniger gut?

Ich versuchte so ungestört wie möglich zu arbeiten. Unser OVWEB Projekt liess dies nicht immer zu. Roman musste einige Male Bugfixes pushen oder ein Modul auf den neusten Stand bringen. Ich war sehr froh, um die Orchestrierung und Hilfe meines Fachvorgesetzten, aber das häufige Auffinden von Unvollständigkeiten im OVWEB kostete mich sicher Zeit.

Weiter gab es auch während meinen Entwicklungsarbeiten Dinge, die mir nicht gleich gelangen. Ich brauchte ein paar Anläufe und musste viel ausprobieren, zum Beispiel bei der Übertragung der Files ins Backend. In Zukunft werde ich diese besser planen.

## Schlussreflexion

Ich blicke auf zehn spannende IPA-Tage zurück. Es hat mir sicherlich Spass gemacht so exakt und genau dokumentiert eine Aufgabe in der Applikationsentwicklung anzugehen. Im Alltag wird vielleicht schneller umgesetzt, aber die genaue Planung hat mir sehr geholfen und mir eine Orientierung am Morgen geboten. Dass ich nur wenig Hilfestellung in Anspruch nehmen musste, ist sicherlich auch ChatGPT zu verdanken. Einfache Fragen oder Diskussionen konnte ich mit diesem Tool klären. Ich finde, ich habe meine Sache gut gemacht und bin fast ein wenig überrascht, wie konzentriert ich diese Arbeit angehen konnte. Es bestand nie die Situation, dass ich etwas vernachlässigt habe oder im Zeitplan zurückgefallen bin.

Die für mich grösste Schwierigkeit, das Dokumentieren, konnte ich gut einteilen und fand immer wieder am Anfang und an Schluss des Tages Zeit dafür. Mit dem Endergebnis bin ich sehr zufrieden. Auch die Art und Weise, wie ich das Programmieren angegangen bin, finde ich bemerkenswert. Mir reichte es nicht einfach eine funktionierende Lösung zu schreiben, mir war es wichtig, dass diese wiederverwendbar ist und für meine Arbeitskollegen so einfach wie möglich zu verstehen wäre. Allgemein habe ich durch diese Arbeit ein gutes Gefühl, dass ich auf einem Niveau einer Fachperson arbeiten und mich austauschen kann. Im Austausch mit Roman oder auch in meinen eigenen Nachforschungen hatte ich kaum Probleme mit der Fachsprache. Ich mache mir keine Sorgen, dass mir das nach der Lehre als Informatiker Sorgen bereitet. Die IPA war sicher fordernd, aber ich finde, ich konnte sie gut meistern. Schlussendlich bin ich trotzdem froh, ist diese intensive Zeit hinter mir.

# Glossar

|  |  |
| --- | --- |
| **Begriff** | **Erklärung** |
| **ANGEBOTsoftware** | Eine ältere FAT-Client-Software zur Verwaltung von Versicherungsdaten und Prämienberechnungen. |
| **OVWEB** | Eine moderne Webanwendung für die Verwaltung von Versicherungsangeboten. |

# Quellenverzeichnis

|  |
| --- |
| Wikipedia: Aktivitätsdiagramm  <https://de.wikipedia.org/wiki/Aktivit%C3%A4tsdiagramm>  [Abrufdatum: 06.02.2024]  Bemerkung: Richtige Notation und Orientierung für Aktivitätsdiagramm in der Planung |
| Thomas Gabathuler: Was ist IPERKA  <https://www.tgabathuler.ch/Iperka>  [Abrufdatum: 08.02.2024]  Bemerkung: Im Kapitel «IPERKA-Methode» verwendet für Wording und Umfang |
| swagger: File Upload  <https://swagger.io/docs/specification/2-0/file-upload/>  [Abrufdatum: 23.10.2023]  Bemerkung: Verwendet in der Umsetzung des Imports |
| geeksforgeeks: Apache POI | Getting Started  <https://www.geeksforgeeks.org/apache-poi-getting-started/>  [Abrufdatum: 08.02.2024]  Bemerkung: Verwendet für Wording in der Dokumentation und Umsetzung des Import Services |
| baeldung: Spring ResponseStatusException  <https://www.baeldung.com/spring-response-status-exception>  [Abrufdatum: 09.02.2024]  Bemerkung: Verwendet, um die richtigen HTTP-Responses abzusetzen bei Errors und allgemein für Error Handling im Backend |
| Rykuno via Github: App Router: File Upload File objects are not supported  <https://github.com/vercel/next.js/discussions/50358>  [Abrufdatum: 09.02.2024]  Bemerkung: Alternative zum direkten Senden von Files über Client – Formdata via einer Server Action. |
| Nextjs : Server Actions and Mutations  <https://nextjs.org/docs/app/building-your-application/data-fetching/server-actions-and-mutations>  [Abrufdatum: 09.02.2024]  Bemerkung: Wichtige Ressource zum Verständnis, warum Dateien nicht so einfach über Server Actions versendet werden können: Sie können nicht serialisiert werden. |
| Doğacan Bilgili: Using dangerouslySetInnerHTML in a React application  <https://blog.logrocket.com/using-dangerouslysetinnerhtml-react-application/>  [Abrufdatum: 12.02.2024]  Bemerkung: Verwendet in Gestaltung der Erklärseite |
| Fontawesome : Animate Icons  <https://fontawesome.com/v6/docs/web/style/animate>  [Abrufdatum: 12.02.2024]  Bemerkung: Eingesetzt bei der Ladeanimation im Frontend |
| OpenTelemetry: Traces  <https://opentelemetry.io/docs/concepts/signals/traces/>  [Abrufdatum: 13.02.2024]  Bemerkung: Für die Umsetzung von Distributed Tracing und konzeptionelles Verständnis von Tracing |
| OpenTelemetry: Traces  <https://storybook.js.org/docs/writing-stories>  [Abrufdatum: 14.02.2024]  Bemerkung: Verwendet für das Erstellen von Storybooks für UI-Komponenten |

## Einsatz von generativen Tools

Ich habe generative Tools verwendet, wobei ich den Output selbst interpretiert bzw. projektbezogen ausformuliert habe.

**User Stories**

* Generieren von Grundlage für User Stories aus Anforderungen.

**Testkonzept**

Generieren von Testfällen basierend auf den Akzeptanzkriterien.

**Generieren von Swagger-Dokumentation**

* Automatische Erstellung von API-Dokumentationen im Swagger-Format aus dem Quellcode.

**Styling in Tailwind**

* Korrekturen bei Syntax-Fehlern im Styling mit Tailwind CSS.

**Log Nachrichten**

* Generierung von Log-Nachrichten basierend auf meinen Beschreibungen und Quellcode.

**Schlusskontrolle**

* Überprüfung der Vollständigkeit der IPA und ihren Kriterienkatalog

# Danksagungen

Zum Schluss möchte ich mich bei allen bedanken, die meine IPA ermöglicht und unterstützt haben: bei meinem Vater für das Korrekturlesen, bei Daniel Sager und Roger Sorg für ihren Einsatz für uns Lernende, bei Roman Babenko, meinem Fachvorgesetzten, ohne den mein Abschluss nicht möglich gewesen wäre, und bei Yves Kaufmann, meinem Experten, der sich die Zeit genommen hat, diese Arbeit zu lesen und zu bewerten.