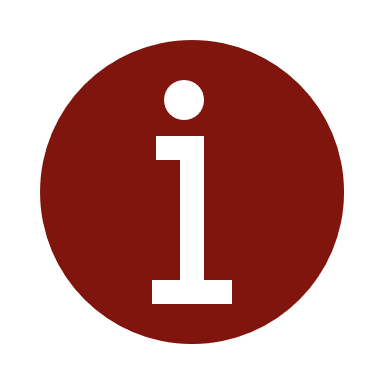
IPA 2024  
 Abschlussarbeit

OVWEB – Workflow Unterbruch

20. März 2024



|  |  |
| --- | --- |
| Kanditat / Autor:  Firma:  Berufsfachschule:  Abteilung:  Hauptexperte:  Nebenexperte:  Berufsbildner:  Verantwortliche Fachkraft:  Fachrichtung:  Projektvorgehensmodell: | Samuel Hajnik  Generali AG  TBZ  IT Apprentices  Marc Siegmund  Janis Marty  Roger Sorg  Roman Babenko  Applikationsentwicklung  IPERKA |

Inhalt

[Vorwort 5](#_Toc161820261)

[Teil 1 - Administration 6](#_Toc161820262)

[1. Projektorganisation 6](#_Toc161820263)

[2. Organisation der Arbeitsergebnisse 7](#_Toc161820264)

[2.1 Datensicherung 7](#_Toc161820265)

[2.1.1 Code 7](#_Toc161820266)

[2.1.2 Dokumentation 10](#_Toc161820267)

[3. Deklaration der benützten Firmenstandards 11](#_Toc161820268)

[4. Zeitplan 12](#_Toc161820269)

[5. Arbeitsjournal 13](#_Toc161820270)

[5.1 Dienstag, 05.03.2024 13](#_Toc161820271)

[5.2 Mittwoch, 06.03.2024 14](#_Toc161820272)

[5.3 Donnerstag, 07.03.2024 15](#_Toc161820273)

[5.4 Freitag, 08.03.2024 16](#_Toc161820274)

[5.5 Dienstag, 12.03.2024 17](#_Toc161820275)

[5.6 Mittwoch, 13.03.2024 18](#_Toc161820276)

[5.7 Donnerstag, 14.03.2024 19](#_Toc161820277)

[5.8 Freitag, 15.03.2024 20](#_Toc161820278)

[5.9 Dienstag, 19.03.2024 21](#_Toc161820279)

[5.10 Mittwoch, 20.03.2024 22](#_Toc161820280)

[Teil 2 – Projekt 23](#_Toc161820281)

[6. Kurzfassung 23](#_Toc161820282)

[7. Projektbeschreibung 24](#_Toc161820283)

[7.1 IPERKA 24](#_Toc161820284)

[7.1.1 Informieren 24](#_Toc161820285)

[7.1.2 Planen 24](#_Toc161820286)

[7.1.3 Entscheiden 24](#_Toc161820287)

[7.1.4 Realisieren 25](#_Toc161820288)

[7.1.5 Kontrollieren 25](#_Toc161820289)

[7.1.6 Auswerten 25](#_Toc161820290)

[8. Einleitung 26](#_Toc161820291)

[9. Informieren 26](#_Toc161820292)

[9.1 Ziele der Aufgabenstellung 26](#_Toc161820293)

[9.2 Vorgaben 26](#_Toc161820294)

[9.3 Ist-Analyse 26](#_Toc161820295)

[9.3.1 Frontend 27](#_Toc161820296)

[9.3.2 Backend 28](#_Toc161820297)

[9.3.3 Überblick 29](#_Toc161820298)

[9.3.4 Klassendiagramme 30](#_Toc161820299)

[9.3.5 Datenbankdiagramm 32](#_Toc161820301)

[9.4 User Stories 32](#_Toc161820302)

[9.4.1 Rollen 33](#_Toc161820303)

[9.4.2 Funktionale Anforderungen 33](#_Toc161820304)

[10. Planen 39](#_Toc161820305)

[10.1 Realisierungskonzept 39](#_Toc161820306)

[10.1.1 Soll-Zustand 39](#_Toc161820307)

[10.1.2 Frontend 40](#_Toc161820308)

[10.1.3 Backend 43](#_Toc161820309)

[10.2 Testkonzept 43](#_Toc161820310)

[10.2.1 Testziele 44](#_Toc161820311)

[10.2.2 Testumgebung 44](#_Toc161820312)

[10.2.1 Testfälle 44](#_Toc161820313)

[10.3 Aktivitätsdiagramme 49](#_Toc161820314)

[10.4 Sequenzdiagramme 51](#_Toc161820315)

[10.5 Klassendiagramm 52](#_Toc161820316)

[10.6 Datenmodell 53](#_Toc161820317)

[10.7 Design 53](#_Toc161820318)

[10.7.1 Variante 1 54](#_Toc161820319)

[10.7.2 Variante 2 55](#_Toc161820320)

[11. Entscheiden 55](#_Toc161820321)

[11.1 Entscheid 55](#_Toc161820322)

[11.1.1 Entscheidungsmatrix 56](#_Toc161820323)

[12. Realisieren 57](#_Toc161820324)

[12.1 Projektumgebung 57](#_Toc161820325)

[12.2 Datenbank und Model 58](#_Toc161820326)

[12.3 Controller und Service 59](#_Toc161820327)

[12.4 Workflow Zwischenspeicherung 59](#_Toc161820328)

[12.5 Workflow Fortsetzung 60](#_Toc161820329)

[12.6 Pendenzenliste 62](#_Toc161820330)

[12.6.1 Sortierung 62](#_Toc161820331)

[12.6.2 Filter 63](#_Toc161820332)

[12.6.3 Anzeige 65](#_Toc161820333)

[12.7 E-Mail Service 67](#_Toc161820334)

[12.8 Logging 68](#_Toc161820335)

[12.9 Distributed Tracing 69](#_Toc161820336)

[12.10 Storybook 71](#_Toc161820337)

[12.11 Benutzerfreundlichkeit 73](#_Toc161820338)

[12.12 Automatisiertes Testing 74](#_Toc161820339)

[12.12.1 Backend Unit Tests 74](#_Toc161820340)

[12.12.2 Frontend Unit Tests 74](#_Toc161820341)

[13. Kontrollieren 75](#_Toc161820342)

[13.1 Testkonzept 75](#_Toc161820343)

[13.2 Testprotokoll Samuel Hajnik 75](#_Toc161820344)

[13.2.1 Testbericht 77](#_Toc161820345)

[13.2.2 Testfall 1 77](#_Toc161820346)

[13.2.3 Testfall 2 77](#_Toc161820347)

[13.2.4 Testfall 3 77](#_Toc161820348)

[13.2.5 Testfall 4 78](#_Toc161820349)

[13.2.6 Testfall 5 78](#_Toc161820350)

[13.2.7 Testfall 6 78](#_Toc161820351)

[13.2.8 Testfall 7 78](#_Toc161820352)

[13.2.9 Testfall 8 79](#_Toc161820353)

[13.2.10 Testfall 9 79](#_Toc161820354)

[13.2.11 Testfall 10 79](#_Toc161820355)

[13.2.12 Testfall 11 80](#_Toc161820356)

[13.2.13 Testfall 12 80](#_Toc161820357)

[14. Auswerten 81](#_Toc161820358)

[14.1 Vergleich Ist / Soll 81](#_Toc161820359)

[14.2 Fazit zur IPA 81](#_Toc161820360)

[14.3 Reflexion 83](#_Toc161820361)

[Danksagung 84](#_Toc161820362)

[Glossar 85](#_Toc161820363)

[Tools 86](#_Toc161820364)

[Abbildungsverzeichnis 87](#_Toc161820365)

[Quellenverzeichnis 88](#_Toc161820366)

Vorwort

Dieses Dokument gehört zur IPA von Samuel Hajnik von Generali AG. In dieser Dokumentation ist der Ablauf sowie das Vorgehen hinter der Arbeit beschrieben.

Die Dokumentation ist in zwei Teile gegliedert: Administration und Projekt.

Im ersten Teil werden die Projektorganisation, Datensicherung, Firmenstandards, der Zeitplan sowie Arbeitsjournaleinträge aufgezeigt. Es ist ersichtlich, mit welchen Mitteln gearbeitet wurde, wie auch die tägliche Reflexion.

Der zweite Teil besteht aus der Projektdokumentation, in der die eigentliche Aufgabe beschrieben wird, was für Probleme aufgetreten sind, wie sie gelöst wurden und wie die Arbeit getestet wurde.

Die ganze IPA wurde komplett hinweg mit der Projektplanungsmethode IPERKA entwickelt. Im Dokument werden die Fachbegriffe Task und Pendenz mehrmals erwähnt, die beiden Begriffe deuten auf das gleiche hinaus.

Die Reflexion und das Fazit stehen am Schluss.

Teil 1 - Administration

1. Projektorganisation

**Lehrbetrieb und Durchführungsort: Nebenexperte:**Generali AG Marty Janis  
Soodmattenstrasse 4 janis.marty@ubs.com  
8134 Adliswil 079 227 53 58  
[058 472 75 00](https://www.google.com/search?client=firefox-b-d&q=generali+adliswil+&lqi=ChFnZW5lcmFsaSBhZGxpc3dpbCIDiAEBSKmIiN3lgICACFofEAAYABgBIhFnZW5lcmFsaSBhZGxpc3dpbCoECAIQAJIBEGluc3VyYW5jZV9hZ2VuY3mqAVcKDS9nLzExYjY2anJoc2QQASoMIghnZW5lcmFsaSgFMh8QASIb7rjLivujVg1VYAHWgXdZ_l1vC9h1_he9K8OnMhUQAiIRZ2VuZXJhbGkgYWRsaXN3aWw)  
life.ch@generali.com

**Kandidat:**Hajnik Samuel  
Im Heumoos 2  
8906 Bonstetten  
077 968 38 56  
samuel.hajnik@generali.com

**Berufsbildner / Lehrfirma:**Sorg Roger  
Generali AG  
Soodmattenstrasse 4  
8134 Adliswil  
058 472 40 42  
roger.sorg@generali.com

**Verantwortliche Fachkraft:**Babenko Roman  
Generali AG  
Soodmattenstrasse 4  
8134 Adliswil  
076 410 87 29  
roman.babenko@loewenzahn.dev

**Hauptexperte:**Siegmund Marc  
[marc.siegmund+pkorg@gmail.com](mailto:marc.siegmund+pkorg@gmail.com)  
078 656 35 16

1. Organisation der Arbeitsergebnisse

Dieses Kapitel beschreibt, wie während der IPA sicherstellt wird, dass alle relevanten Daten gesichert sind und nötigenfalls wiederhergestellt werden können.

* 1. Datensicherung

Die Datensicherung ist essenziell für den Fall, dass Probleme auftreten sollen, somit ist sie ein wichtiger Bestandteil der IPA. Für die tägliche Sicherung der Arbeit wurde auf verschiedene Speicherungssysteme gesetzt.

* + 1. Code

Für den Code wurde ein neuer Github-Branch in der Team-Repository erstellt.

**Branch**: [Link](https://github.com/IT-Apprentices/insurance-offer-management/tree/ipa-workflow-unterbruch)

Hier ist der Verlauf der Commits auf den Branch:

|  |
| --- |
| commit c4caf99f2c8578590148b2c18c874f293176f9a3  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Tue Mar 19 13:49:02 2024 +0100  feat(be): added missing test cases (null, empty collection)  commit bee83f1a15f425ce42a92d272bfe86ec68c6474d  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Tue Mar 19 13:08:52 2024 +0100  fix(fe+be): code cleanup + mapToGetDto function logic fix  commit d027775dea5ca079ff5df8ceb55965564b99d3d4  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Fri Mar 15 16:01:10 2024 +0100  feat(fe): table visualization on needsReminder + prefill resume fix  commit 872181c6491adc301f389446355e2bb3faac5730  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Fri Mar 15 10:38:14 2024 +0100  feat(be): added negative unit tests  commit 5c68fc8fc5bfa7559356708236de806e6b62dafe  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Thu Mar 14 11:45:54 2024 +0100  feat(be+fe): added storybook + unit tests  commit 0bb881e2e47d18e8efdbbcc52f562db7ac366062  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Wed Mar 13 17:07:19 2024 +0100  feat(be+fe): added logging + tracing  commit f3160e08bba5d9b2c7b8220ef368cd97a4a4e1b4  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Wed Mar 13 10:53:30 2024 +0100  feat(be): added email scheduler  commit afa077a787b86aea902a008bab9fe4f626de426e  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Tue Mar 12 15:47:27 2024 +0100  feat(fe+be): form prefill functional  commit 50efecaa02541aad54bfc7892ce1118b7a105866  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Tue Mar 12 10:21:01 2024 +0100  feat(fe): global sort + description filter  commit 96e8953112ecc667bc17fe88cec82e58b643b57c  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Fri Mar 8 14:59:13 2024 +0100  feat(be+fe): started resumeTask use case  commit bfa2cfbf8b1e1ecaa70e6bd967fbd3a225bf8346  Author: Samuel Hajnik <shajnik7@gmail.com>  Date: Fri Mar 8 11:32:37 2024 +0100  wip: save customer data on cancel |

* + 1. Dokumentation

Durch OneDrive wird der neuste Stand laufend gespeichert. Für zusätzliche Sicherheit wird eine separate Github Repository am Schluss von jedem Tag ein Backup erstellt.  
  
**Repository:** [Link](https://github.com/Draxx7236/IPA-2024)

Hier ist der Verlauf der Commits auf die Repository:

|  |
| --- |
|  |

1. Deklaration der benützten Firmenstandards

Im Entwickler-Team werden die folgenden Technologien verwendet.

|  |  |
| --- | --- |
| Frontend | |
| **Name** | **Einsatz** |
| Next.js | Frontend-Technologie |
| shadcn | Komponenten Design Bibliothek |
| Typescript | Frontendsprache |
| Jest | Testing |
| Storybook | Frontend Dokumentation |
| **Backend** | |
| Springboot | Backend-Technologie |
| Java | Backendsprache |
| Lombok | Annotations für Automatisierung |
| JUnit | Testing |
| **Daten Layer** | |
| PostgreSQL | Datenbank |
| Flyway | Datenbankversionierungstool |
| **Identity Provider** | |
| OpenID Connect | Identitätsprotokoll |
| Keycloak | Identitätsmanagement |

1. Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

   Automatisch generierte BeschreibungZeitplan

1. Arbeitsjournal
   1. ****Dienstag, 05.03.2024****

|  |  |
| --- | --- |
| Tätigkeiten | * Zeitplan erstellen * Dokumentation starten * User Stories * Expertengespräch |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| Roman hat mir nach dem Expertengespräch noch Tipps gegeben zu meinem Zeitplan. |
| Reflexion |
| ***Was lief heute positiv:***  **Ich konnte heute positiv und motiviert in meine IPA starten. Ich habe mir vorgenommen, vieles möglichst schnell zu erledigen, damit ich der begrenzten Zeit vorankomme. Ich konnte den Zeitplan in einer relativ schnellen Zeit erledigen, was ich mir nicht dachte, da es doch eine anstrengende Aufgabe ist. Zugleich hat mir das Dokumentieren auch gut gelungen heute, ich konnte schon vieles dokumentieren, vor allem in der Phase Informieren, was mich nach vorne drängen wird in den nächsten Tagen. Beim Expertengespräch konnte ich sehr gute Einblicke bekommen, wie es laufen wird, was wichtig ist und auf was ich achten muss.**  ***Was lief nicht so gut:***  **Vor dem Expertengespräch habe ich mich kurzfristig entschieden, meinen Zeitplan noch zu umstrukturieren, da es für mich ungenau war. Nach dem Gespräch holte ich mir noch Hilfe bei Roman, damit ich eine Idee bekomme, wie ich es verbessern kann. Er hat mir vorgeschlagen, Excelformeln zu benutzen. Somit konnte ich meinen Zeitplan verbessern.**  ***Fazit:*** Heute war ein erfolgreicher Tag. Ich konnte gut starten und fortlaufend arbeiten. Das Expertengespräch hat mich nur sicherer gemacht und ich bin mir sicher, dass es in den nächsten Tagen gut fortlaufen wird. |

* 1. Mittwoch, 06.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Diagramme * Mockups * Planen anfangen |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| Roman gefragt, ob die Frist von N-Tagen auch in der Datenbank implementiert werden soll und ob mein Vorschlag für die neue Task-Entity richtig aussieht. |
| Reflexion |
| ***Was lief heute positiv:***  **Der Tag verlief schnell, da ich ständig am Arbeiten war und ich all meine heute vorgesehenen Aufgaben erledigen konnte. Am Anfang hatte ich Stress, da ich mir nicht sicher war, ob ich genug Zeit eingeplant habe für die Diagramme und Mockups. Mit den Aktivitätsdiagrammen konnte ich rechtzeitig fertig werden, jedoch beim Sequenzdiagramm habe ich ein bisschen mehr Zeit gebraucht, da ich ein Programm dafür benutzt habe: PlantUML, mit dem ich mich noch nicht auskannte, aber es hat meinen Arbeitsfluss verbessert. Es erlaubte mir, das Diagramm code-mässig zu schreiben, was mich Zeit gekostet hat, da ich die Syntax lernen musste, aber schlussendlich sind mir die Resultate gelungen und ich denke, es ist schneller gegangen, als wenn ich es von Hand entworfen hätte. Mit den Mockups hatte ich auch keine Mühe, es war einfach, sie zu gestalten mit Figma, zuerst habe ich die Hauptfarben aus dem Generali Styleguide als Variabeln deklariert und mit Frames gearbeitet, das hat meine Effizienz gesteigert.**  ***Was lief nicht so gut:***  **Die Datenmodellierung musste ich auf den Nachmittag verschieben, da die Vorarbeit noch nicht im Main Branch gemerged war, aber es hat mich nicht gehindert. Nach dem Modellieren habe ich Roman noch nach seiner Meinung gefragt.**  ***Fazit:*** Wiederum ein guter Tag. Trotz einiger Stolpersteine konnte ich den Tag zielgesetzt durcharbeiten. Die “Rekonstruktion” des Zeitplans gestern hat mir sicherlich geholfen, da ich jetzt eine bessere Übersicht habe. Ich habe meine Ziele erreicht und bin somit bereit für den nächsten Tag. |

* 1. Donnerstag, 07.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Realisierungskonzept * Entscheiden * Planen * Testkonzept |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| - |
| Reflexion |
| ***Was lief heute positiv:***  **Heute konnte ich vieles in meiner Dokumentation erledigen. Der Rest der Planung, also hauptsächlich das Realisierungskonzept und dessen Unterthemen, sind mir einfach gefallen. Ich konnte das Diagramm sowie den Sollzustand schnell umsetzen. Danach habe ich mit der Entscheidungsphase begonnen, auch hier erging es mir einfach. Ich habe mich entschieden, eine Entscheidungsmatrix zu verwenden, da es Text sparen und meine Überlegungen übersichtlicher darstellen würde. Wegen diesem Fortschritt habe ich mich auch entschieden, das Testkonzept heute zu erledigen, obwohl es ursprünglich erst am achten Tag vorgesehen war. Der Grund dafür war, dass ich beim Realisieren Test-Driven entwickeln kann, zumindest auf einer kleinen Skala, da die automatisierten Tests noch nicht implementiert sind. Es war nicht einfach, die Test-Cases zu definieren, aber da ich genug Zeit eingeplant habe, war es auch kein Problem.**  ***Was lief nicht so gut:***  **Ich bin auf keine Probleme gestossen heute, es lief alles wie geplant und besser.**  ***Fazit:*** Die ersten drei Teile der IPERKA-Methode konnte ich in der Dokumentation schon implementieren. Bis jetzt ging alles zum grossen Teil wie geplant und morgen fange ich an mit dem Code, da erhoffe ich mir auch so einen Fortschritt zu machen. |

* 1. Freitag, 08.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Zwischenspeicherung * Task in Pendenzenliste anzeigen * Formular Vorbefüllen |

|  |
| --- |
| **Probleme** |
| Bei der Zwischenspeicherung gab es ein Problem bei der OpenApi Generierung der API Definition. |
| Hilfestellung |
| Roman und ChatGPT |
| Reflexion |
| ***Was lief heute positiv:***  **Obwohl es heute ein grosses Problem gab, konnte ich meine verlorene Zeit nachholen, indem ich eine andere Realisierungsaufgabe schnell implementieren konnte sowie bei der Dokumentation weitermachte. Ich habe es geschafft, die Tasks anzuzeigen in der Tabelle unter meiner geplanten Zeit. Somit konnte ich mehr in die Dokumentation investieren und den Realisierungsteil anfangen zu schreiben, wie auch die Fortsetzung des Workflows anzufangen, wie geplant.**  ***Was lief nicht so gut:***  **Heute habe ich ein Problem entdeckt. Ich habe die erste User Story angefangen und als ich fast fertig wurde und es testen wollte, ist es nicht gegangen, der Payload enthielt nicht alle benötigten Daten. Ich konnte es selber nach einer halben Stunde nicht herausfinden, deshalb wollte ich mir bei ChatGPT behelfen, aber auch da gab es keine nützliche Antworten. Danach ging ich zu Roman und er hat es sofort gesehen, es war ein Problem bei OpenApi. Es generiert den benötigten API-Client, damit das Frontend mit dem Backend kommuniziert, aber hier gibt es ein Problem und zwar verschickt meine Methode ein benötigtes Feld für die Erstellung des Tasks nicht.**  ***Fazit:*** Es war ein neutraler Tag, natürlich gab es ein Problem, das mich behindert hat, aber ich bin zuversichtlich, dass es schnell gelöst wird. Ich konnte die verlorene Zeit nachholen. |

* 1. Dienstag, 12.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Formular Vorbefüllen * Task Liste Sortierung * Task Liste Filter * Realisieren Weiterdokumentieren |

|  |
| --- |
| **Probleme** |
| Formular VorbefüllungVorbefüllung |
| Hilfestellung |
| Internet: Logging und Tracing |
| Reflexion |
| ***Was lief heute positiv:***  **Heute konnte ich gut vorwärtskommen mit der Arbeit. Ich habe heute eine Stunde mehr gearbeitet, da ich letztes Mal auf ein Defekt beim API-Generator beim Zwischenspeichern gestossen bin. Das konnte gelöst werden und somit konnte ich sicherstellen, dass die Speicherung funktionierte. Danach habe ich mich an die Fortsetzung des Workflows gemacht und obwohl es Schwierigkeiten gab, konnte ich es fertigstellen. Die Sortierung und Filter habe ich schnell implementiert, deswegen habe ich heute auch Zeit in die Dokumentation investiert, damit ich später nicht in Stress gerate.**  ***Was lief nicht so gut:***  **Für die Implementierung der Fortsetzung habe ich den ganzen Tag gebraucht, ich bin auf mehrere Fehler gestossen, wie z. B. Wie soll die Customer Komponente die Task Id erkennen, machte aber ständig weiter und konnte die Aufgabe endlich an die Ziellinie bringen.**  ***Fazit:*** Ich bin mir sicher, dass heute nicht das letzte Mal ist, dass ich Überstunden machen werden müsse. Heute war es, weil ich im Zeitplan hintendrein war und aufholen wollte. Auf jeden Fall war es ein anständiger und erfolgreicher Tag, aber trotzdem bin ich glücklich, dass er vorbei ist. |

* 1. Mittwoch, 13.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * E-Mail Versenden * Logging * Unit Testing |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| Internet: Testing, Spring Scheduler, Logging und Tracing |
| Reflexion |
| ***Was lief heute positiv:***  **Die Arbeit verlief heute sehr gut. Die E-Mail Service konnte ich rapid umsetzen, da wir im Backend schon eine Vorlage hatten für das Versenden von E-Mails. Daher konnte ich diese Vorlage für meine Methode gebrauchen, um die Reminder E-Mails zu verschicken. Vor der IPA habe ich einen Tipp von Roman bekommen und zwar, dass ich einen Scheduler brauchen sollte für das Versenden, da die Frist unabhängig davon, ob der Benutzer online ist oder nicht, überprüft werden soll. Auf der offiziellen Spring-Website gab es ein Beispiel, also habe ich mich dem nach orientiert und es für meine Zwecke angepasst. Nach manuellem Testen von der Methode habe ich noch die Ansicht der Pendenzenliste angepasst, sodass ein Task, der die Frist überlaufen ist, visuell markiert wird. Am Nachmittag konnte ich dann schon mit den nicht-funktionalen Anforderungen anfangen, mit Logging und Tracing. Mit diesen Themen habe ich mich bisher nicht auseinandergesetzt, aber im Internet gab es gute Ansätze für die Implementierung, deshalb lief auch das reibungslos.**  ***Was lief nicht so gut:***  **Stolz kann ich sagen, dass heute nichts oder zumindest nur winzige Stolpersteine falsch gelaufen ist. Ich konnte meine Arbeit rasch fortsetzen und bin somit voraus im Zeitplan.**  ***Fazit:*** Am Nachmittag habe ich mich kurzfristig noch entschieden, eine Stunde länger zu bleiben, da ich schon fast mit dem Tracing fertig war, obwohl ich meine acht Stunden gearbeitet habe. Schlussendlich hat es sich meiner Meinung nach gelohnt, da ich fertig damit wurde früher als geplant, aber somit habe ich jetzt Zeit für die Dokumentation und Verfeinerungen. |

* 1. Donnerstag, 14.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Unit Tests |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| Roman und Internet bei Storybook |
| Reflexion |
| ***Was lief heute positiv:***  **Da ich gestern guten Fortschritt gemacht habe, habe ich mich entschieden, mein Tempo nicht zu verlieren. Logging und Tracing konnte ich gestern fertigbringen, deswegen konnte ich heute schon mit dem Unit Testing beginnen. Es gab keine Probleme, aber es war viel zum Testen, deswegen hat es mir auch viel Zeit in Anspruch genommen, was nicht schlimm war, da ich genügend Zeit eingeplant habe. Danach habe ich mit den Storybooks angefangen. Da hatte ich jedoch nicht so viel Erfahrung wie bei den automatisierten Tests und musste mir deswegen Hilfe holen, einerseits bei Roman und auch im Internet für spezifische Anwendungsfälle, wie z. B. ErrorStates. Am Nachmittag habe ich mich mit Dokumentieren beschäftigt, weil ich hier eine solide Struktur für meine IPA haben will. Am Ende vom Tag hatte ich das zweite Expertengespräch und ich konnte da gute Feedbacks einholen für mein Bericht sowie ein Gefühl bekommen für das Fachgespräch.**  ***Was lief nicht so gut:***  **Vor dem Expertengespräch wollte ich die Applikation und meine Anwendungsfälle testen, damit ich auch etwas zum Vorzeigen hatte, aber da habe ich einen Bug entdeckt und zwar kann man mit der Speicherung nicht fortfahren nach dem Vorbefüllen. Ich werde es mir morgen anschauen.**  ***Fazit:*** Ich konnte gut vorankommen heute mit meinen Aufgaben. Beim Expertengespräch habe ich Feedback und Verbesserungsvorschläge bekommen. Morgen werde ich versuchen, diese zu implementieren. |

* 1. Freitag, 15.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Korrekturschreiben * Use Cases * Bug Fixes |

|  |
| --- |
| **Probleme** |
| Korrekturschreiben |
| Hilfestellung |
| Roman hat mir bei Storybook geholfen |
| Reflexion |
| ***Was lief heute positiv:***  **Ich habe mir vorgenommen heute meine Dokumentation anhand vom Feedback von den Experten, das ich gestern erhalten habe, zu verbessern. Dafür habe ich Use-Case Diagramme erstellt, die die Benutzerinteraktionen im Kontext der Aufgabe darstellen. Während des Tages habe ich auch noch Korrekturen vorgenommen am Code. Ich hatte Bugs beim Vorbefüllen, Anzeigen der Fristüberschreitung im Task Table, Storybook und Frontend Tests. Das Problem beim Vorbefüllen war, dass im Formular die Felder auf dirty gesetzt waren, was keinen Sinn ergibt, da die Daten vom System vorbefüllt werden und nicht von Hand. Das Anzeigen habe ich aus Versehen korrigiert. Ich habe gemerkt, dass Delete auch nicht funktionierte und da war die Id, auf die der Button referenzierte, falsch, das hat auch den Defekt in der Anzeige gelöst. Beim Storybook musste ich mir Hilfe holen bei Roman und der hat mir den Rat gegeben, die CreateCustomerForm analog einer anderen bei uns in der Applikation bereits bestehenden Form umzubauen. Ich habe es gemacht und konnte somit auch das Storybook der anderen Form inspiriert verbessern. Die Tests konnte ich mittels Debugging korrigieren, meistens konnte Jest den Text auf einem Button nicht erkennen oder es wurde ein Objekt falsch gemockt.**  ***Was lief nicht so gut:***  **Anfangs hatte ich Mühe, meinen Bericht umzuformulieren, deswegen habe ich es für heute aufgegeben und ich werde die nächste Woche daran arbeiten. Der Tag hat sich lange angefühlt, obwohl ich heute perfekt im Zeitrahmen war.**  ***Fazit:*** Müde, aber gut gelaunt kann ich sagen, dass die Woche erfolgreich war. Ich konnte meine Ziele erreichen und bin sogar voraus im Zeitplan. Die nächste Woche werde ich jedoch einige Textpassagen überarbeiten müssen, wegen der Umgangssprache, aber ich bin zuversichtlich, dass ich es noch im Zeitrahmen erreichen werde. |

* 1. Dienstag, 19.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Testfälle Testen * Auswertung |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| - |
| Reflexion |
| ***Was lief heute positiv:***  **Heute habe ich meinem Zeitplan gefolgt. Ich habe die Testfälle vom Testkonzept getestet und alles dokumentiert. Da gab es keine Probleme, da ich schon letzte Woche alle Bugs im Code korrigiert und getestet habe. Dann machte ich mich an die letzte Phase von IPERKA: Auswerten. Hier war ich überrascht, ich habe mir nicht gedacht, dass das Reflektieren so viel Zeit in Anspruch nehmen kann. Als Hilfe und Erfrischung für mein Gedächtnis habe ich meine Journaleinträge nochmals durchgelesen, um die Schlussreflexion detailliert dokumentieren zu können. Ich habe versucht, den “Sweet Spot” zu finden zwischen einer genügenden Reflexion und einem Roman. Schlussendlich kann ich jedoch sagen, dass ich damit zufrieden bin, es könnte immer detaillierter oder besser sein, aber ich bin mit der Selbstauswertung glücklich.**  ***Was lief nicht so gut:***  **Letzte Woche habe ich mir erhofft, heute möglichst viel vom Bericht sprachlich überarbeiten zu können, dazu bin ich jedoch nicht gekommen. Wenn ich aber auf den Zeitplan vorausschaue, ist das nicht schlimm, da ich für die Korrektur immer noch drei Stunden eingeplant habe, was mir genau noch reichen sollte.**  ***Fazit:*** Ich habe viel geschrieben heute. Wahrscheinlich ist das der niedrigste Punkt der IPA, aus der Sicht der Motivation zumindest. Gleichzeitig ist es aber auch nötig, nicht nur um eine gute Note zu erreichen, sondern auch für die allgemeine Beharrlichkeit im Leben. Morgen ist der letzte Tag, ich werde nur noch die Kurzfassung, das Vorwort und die Auswertung schreiben müssen und ansonsten nur korrigieren. |

* 1. Mittwoch, 20.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Dokumentation * Korrektur |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| - |
| Reflexion |
| ***Was lief heute positiv:***  **Heute war der letzte Tag meiner IPA. Da ich in den letzten Tagen vorwärts gearbeitet habe, gab es heute nicht vieles zu tun. Am Morgen habe ich die Kurzfassung und das Vorwort geschrieben. Am Ende des Tages habe ich noch die Git-Commit-Historie ausgelesen für das Kapitel «Organisation der Arbeitsergebnisse». Ansonsten war der Tag von Korrekturschreiben geprägt. Es war nicht einfach, sich zu entscheiden, was ich im Bericht behalten soll, aus der Sicht der Prägnanz oder welche Ausdrücke ich ersetzen soll. Doch zuletzt denke ich, dass es mir gut gelungen ist. Es gibt sicher Stellen, wo man ein Paragraf besser ausformulieren könnte, doch ich habe mein Bestes getan, um dies zu vermeiden. Dann habe ich noch die Ci-Checks laufen lassen, um zu sehen, dass alles den Richtlinien nach implementiert wurde. Allerletzt überprüfte ich noch meinen Inhalt mit dem Kriterienkatalog, damit ich sicherstellen konnte, dass ich alles vollständig hatte.**  ***Was lief nicht so gut:***  **Von der Arbeit her war heute kein anstrengender Tag, aber ich fühlte mich trotzdem ein bisschen gestresst wegen der Korrekturen. Bei den Ci-Checks musste ich noch eine Anpassung vornehmen bei den Frontend-Unit-Tests, da diese die Bedingungen nicht bestanden haben.**  ***Fazit:*** Glücklich kann ich sagen, dass eine Hälfte der IPA hinter mir ist, jetzt steht nur noch die Präsentation mit dem Fachgespräch vor mir. Nicht nur heute, sondern die ganze Woche ist mir gelungen. Es keine gröberen Probleme aufgetreten, aber ich hatte dennoch vieles zu tun. |

Teil 2 – Projekt

1. Kurzfassung

**Ausgangslage**

Die Generali Schweiz AG, ein führender Versicherungsanbieter in der Schweiz, setzte traditionell auf eine in Delphi entwickelte Monolith-Anwendung für das Management von Kundendaten und die Prämienkalkulation. Um diesen veralteten Ansatz durch moderne, webbasierte Technologien zu ersetzen, wurde das Projekt «OVWEB» ins Leben gerufen. Die OVWEB-Plattform umfasst bereits Erstellungsfunktionen für die Kundenkreierung. Ein kritischer, noch fehlender Bestandteil war die Möglichkeit für Berater, den Prozess zu unterbrechen.

**Umsetzung**

Die Umsetzung des Projekts folgte der IPERKA-Methode, die eine strukturierte Arbeiten am Projekt ermöglichte. 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 Taskdaten aufnehmen zu können. Der Taskservice wurde erweitert, um die Fortführung zu ermöglichen. Der E-Mail-Service wurde angepasst für das Versenden von E-Mails. Das Frontend wurde anhand von vorher angefertigten Mockups in Figma implementiert. Durch das Zusammensetzen der UI-Oberfläche aus einzelnen Komponenten wurde eine modulare und erweiterbare Benutzeroberfläche mit Sortierung und Filter geschaffen. Strukturiertes Logging sorgten für einen robusten und nachvollziehbaren Abbrechprozess. Distributed Tracing mit OpenTelemetry wurde schichtübergreifend integriert.

**Ergebnis**

Das Ergebnis des Projektes ist ein funktionsfähiger und benutzerfreundlicher Workflow Unterbruch, der es den Beratern ermöglicht, den Kundenerstellungsprozess abzubrechen und später fortzufahren. Der globale Filter auf der Taskliste konnte jedoch nicht rechtzeitig implementiert werden.

1. Projektbeschreibung
   1. IPERKA

Diese Arbeit wird mit der IPERKA-Methode implementiert. Sie wird in sechs Schritte gegliedert: Informieren, Planen, Entscheiden, Kontrollieren und Auswerten. Die Bedeutung von Planung wird in dieser Methodik besonders hervorgehoben, um eine genaue Vorgehensweise zu fördern. Es ist die bevorzugte Projektmanagementmethode an der Berufsschule.

Das Projekt ist in die sechs Phasen unterteilt, um eine geordnete Struktur beizubehalten.

* + 1. Informieren

Die Ziele und Herausforderungen des Prozesses werden identifiziert. Es wird untersucht, welche Prozesse gelöst werden müssen, welche Ressourcen dafür gebraucht werden und Fragen werden geklärt. Dies ist wichtig für das Verständnis der Aufgabe sowie für die Erstellung des Zeitplans.

* + 1. Planen

Es wird ein detaillierter Plan erstellt, um einen strukturierten Projektablauf zu garantieren. Dies umfasst Festlegung von Aufgaben, Verantwortlichkeiten sowie auch ein Realisierungskonzept und ein Testkonzept.

* + 1. Entscheiden

Der bestmögliche Lösungsweg wird ausgewählt, in dem man die verschiedenen Ansätze zusammen vergleicht. Es ist wichtig, sich zu überlegen, ob man den Plan umsetzen kann und ob die Ideen überhaupt sinnvoll sind.

* + 1. Realisieren

Sobald alles 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.

* + 1. Kontrollieren

Erst wenn die Realisierung vollständig abgeschlossen wurde, werden die Resultate so getestet, wie es im Testprotokoll steht. Fehlgeschlagene Tests werden entweder direkt korrigiert und behoben oder als Fehler vermerkt und dokumentiert.

* + 1. Auswerten

Als Abschluss gibt es eine Reflexion über die Arbeit und die Erfahrungen, die man gesammelt hat. Man geht durch all seine Schritte, welche man innerhalb des Projektes gemacht hat, und es wird untersucht, was gut gelaufen ist und wo Verbesserungen möglich sind. Das ist auch hilfreich für die Umsetzung zukünftiger Projekte.

1. Einleitung

In diesem Teil wird die Umsetzung des Projektes mit der IPERKA-Methode aufgezeigt. Die Kapitel sind in die Phasen unterteilt, um sich einfacher durch die Struktur des Dokumentes orientieren zu können.

1. Informieren
   1. Ziele der Aufgabenstellung

Das Ziel ist, dem Benutzer zu ermöglichen, die Kundenerstellung zu stoppen, um später fortzufahren. Wenn die Aufgabe die Frist von N-Tagen übersteigt, wird an den Benutzer eine E-Mail geschickt mit dem Link zu dieser Aufgabe, um zu fortfahren.

* 1. Vorgaben

Die Kundenerstellung soll bei einem beliebigen Punkt unterbrochen werden können. Die ausgefüllten Daten sollten in der Datenbank gespeichert werden ohne Validierung. Die Erstellung kann später weitergeführt werden und die Liste sortiert sowie filtriert werden. Sobald die Frist von N-Tagen bei einer Aufgabe abläuft, wird der Benutzer per E-Mail informiert.

* 1. Ist-Analyse

Die Offerten Verwaltung Web, kurz OVWEB, wird dazu dienen, die alte Applikation in eine Cloud-basierte Webanwendung zu verwandeln. Das Programm ist gedacht für Kundenverwaltung und Erstellung von Offerten. Im jetzigen Stand der Applikation ist es möglich, sich einzuloggen über den Keycloak-Server und Kunden erstellen via ein Formular. In der bestehenden Legacyapplikation mussten die Anwender die Erstellung von Kunden manchmal unterbrechen. Zum Beispiel mussten zusätzliche Abklärungen beim Kunden gemacht werden. Dieser war aber in diesem Moment nicht verfügbar. Bei einem teilerfassten Kunden gingen so die bisherigen Eingaben verloren, da die Validierung im halbfertigen Zustand der Erfassung eine Speicherung nicht zulässt. Diese Anforderung wird angesetzt, um den Unterbruch so einfach wie möglich zu gestalten.

* + 1. Frontend

Das Frontend wurde mit Next.js 14 umgesetzt und ist nach dem Generali-Styleguide gestaltet. Die Session wird mit NextAuth geregelt. Für Components wird die Shadcn Library benutzt. Für die Backendkommunikation wird ein TypeScript-Client verwendet.

Ein Bild, das Text, Screenshot, Schrift, Logo enthält.

Automatisch generierte Beschreibung

Abbildung 1: Home-Page

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 2: Kundenformular Ist Zustand

* + 1. Backend

Das Backend wurde mit dem Java-Framework Springboot realisiert. Es besteht ein Model, das User, Customer, Task und Adresse beinhaltet. Als Query-Sprache wird PostgreSQL verwendet. Die Datenbank Anbindung läuft via einen lokalen Docker-Container und wird durch Flyway versioniert.

* + 1. Überblick

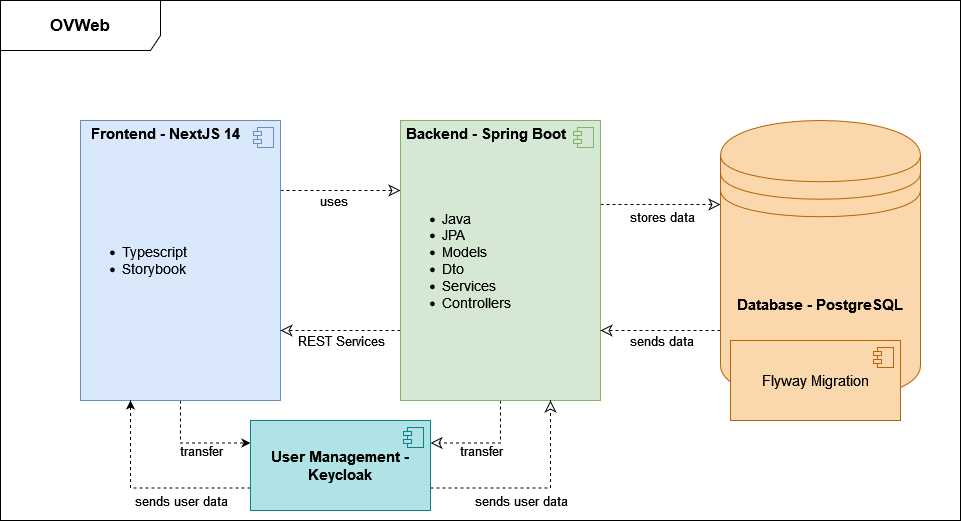
In der folgenden Abbildung wird die Architektur der OVWEB-Applikation dargestellt.

Abbildung 3: OVWeb-Struktur

**Docker-Container**: Die Verwendung von Docker-Containern für Keycloak, die PostgreSQL-Datenbank und wesentliche Dienste wie Logging und Tracing unterstützt die Skalierbarkeit und vereinfacht Wartungsprozesse durch konsistente Umgebungen über Entwicklungs-, Test- und Produktionsstufen hinweg.

**Skalierbarkeit und Flexibilität**: Die containerbasierte Architektur ermöglicht eine agile Anpassung an sich ändernde Anforderungen und erleichtert die Skalierung der einzelnen Komponenten.

* + 1. Klassendiagramme

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

Automatisch generierte BeschreibungFür die Customer, User und Task Modelle gibt es Service und Controller für CRUD-Operationen (Create, Read, Update und Delete).

Abbildung 4:Klassendiagramm Models

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

Automatisch generierte Beschreibung

Die Task Klasse besteht aus weiteren Subklassen.

Abbildung 5: Klassendiagramm Task mit Subklassen

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

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

Automatisch generierte Beschreibung

Abbildung 6: Klassendiagramm Controller

Abbildung 7: Service Klassen

* + 1. Datenbankdiagramm

Die Datenbank wird durch Flyway verwaltet und enthält die gleichen Entitäten wie die Models im Backend.

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 8: Datenbankdiagramm

* 1. User Stories

Um ein tieferes Verständnis der Anforderungen zu gewinnen, werden User Stories erfasst. Die Anforderungen der Benutzer und der Implementierung der neuen Workflow-Funktionalität liegen dabei im Vordergrund. Es werden jedoch auch nicht-funktionale Anforderungen als User Stories erfasst, um einen besseren Überblick zu behalten über die IPA relevanten Kriterien. Somit kann auch beim Testen gleich überprüft werden, ob die Entwicklerthemen vollständig umgesetzt wurden sowie am Ende den Ist- und Sollzustand vergleichen.

* + 1. Rollen

Es gibt Benutzer (Berater, Versicherungsvertreter), Entwickler und Administratoren.

* + 1. Funktionale Anforderungen

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Workflow Zwischenspeicherung | 4 | 1 |
| Beschreibung |  |  |

Als ein **Benutzer** kann ich während der Erstellung eines Kunden auf einen **Button klicken**, damit ich **den Fortschritt speichern kann**.

|  |
| --- |
| Acceptance Criteria |

**AC 1.1:** Auf der Customer Creation Page soll ein klar ersichtlicher «Später Fortfahren» Button ersichtlich sein.

**AC 1.2:** Die Daten werden in der Datenbank abgespeichert.

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Anzeige in Task Liste | 2 | 2 |
| Beschreibung |  |  |

Als ein **Benutzer** kann ich **nicht fertige Customer Creation Tasks sehen**, damit ich **einen Überblick habe über meine Tasks**.

|  |
| --- |
| Acceptance Criteria |

**AC 2.1:** Es sollte relevante Details wie Datum/Uhrzeit, Erfassungsobjekt und/oder den Namen des Kunden anzeigen.

**AC 2.2:** Jede unterbrochene Aufgabe in der Übersichtsliste sollte angeklickt werden können, um die Fortsetzung zu ermöglichen.

**AC 2.3:** Nachdem die konfigurierbare Frist von N-Tagen abgelaufen ist, wird die Aufgabe visuell vorgehoben.

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Formular Vorbefüllung | 4 | 3 |
| Beschreibung |  |  |

Als ein **Benutzer** kann ich auf den **«Fortsetzen» Button klicken**, damit ich **mit** **der Aufgabe weiterfahren kann**.

|  |
| --- |
| Acceptance Criteria |

**AC 3.1:** Auf der Customer Creation Page sollen die abgerufenen Daten ersichtlich und korrekt formatiert sein.

**AC 3.2:** Die Backend-Logik sollte unterbrochene Aufgaben aus der Datenbank abrufen

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Task Liste Sortierung | 2 | 4 |
| Beschreibung |  |  |

Als ein **Benutzer** kann ich auf den **auf das «Beschreibung» Feld klicken**, damit ich **die Tasks alphabetisch sortieren kann**.

|  |
| --- |
| Acceptance Criteria |

**AC 4.1:** Die Tasks werden von A-Z absteigend oder aufsteigend, wenn man doppelt klickt, angezeigt

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Task Liste Filter | 2 | 5 |
| Beschreibung |  |  |

Als ein **Benutzer** kann ich **einen Begriff in das «Suchen» Feld eingeben**, damit ich **die Tasks filtrieren kann**.

|  |
| --- |
| Acceptance Criteria |

**AC 5.1:** Das Suchfeld erscheint auf der Task-Liste

**AC 5.2:** Es werden nur Tasks angezeigt, die die Sucheingabe beinhalten.

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| E-Mail versenden | 5 | 6 |
| Beschreibung |  |  |

Als ein **Benutzer** kann ich **per E-Mail über überfällige Aufgaben benachrichtigt werden**.

|  |
| --- |
| Acceptance Criteria |

**AC 6.1:** Nachdem die konfigurierbare Frist von N-Tagen abgelaufen ist, wird eine E-Mail an den Benutzer geschickt mit dem folgenden Inhalt: Ein Link zur Weiterführung des abgebrochenen Workflows.

**AC 6.2:** Der Link leitet auf die richtige Aufgabe weiter.

**Nicht funktionale Anforderungen:**

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Usability | 1 | 7 |
| Beschreibung |  |  |

Als ein **Benutzer** möchte ich, dass **der Unterbruch benutzerfreundlich ist**, damit ich **ohne Hilfe die Aufgabe beenden kann**.

|  |
| --- |
| Acceptance Criteria |

**AC 7.1:** Die Anzeige ist in einer klar verständlichen Sprache, die Knöpfe sind klar beschriftet.

**AC 7.2:** Soll ein Fehler vorkommen, dann wird der Benutzer durch Hinweise darauf aufmerksam gemacht.

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Logging | 2 | 8 |
| Beschreibung |  |  |

Als ein **Administrator** möchte ich, dass **die neuen Funktionalitäten strukturiertes Logging implementieren**, damit **die Fehler einfacher zu beheben sind.**

|  |
| --- |
| Acceptance Criteria |

**AC 8.1:** Strukturiertes Logging wurde aktiv umgesetzt.

**AC 8.2:** Alle notwendigen Informationen wurden in den Logs enthalten, um die Aktionen nachvollziehen zu können.

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Tracing | 4 | 9 |
| Beschreibung |  |  |

Als ein **Entwickler** möchte ich, dass **jede Funktion mit einer Trace-Id versehen werden kann**, damit ich **die Fehlerbehebung vereinfacht werden kann**.

|  |
| --- |
| Acceptance Criteria |

**AC 9.1:** Bei jeder Frontend-Anfrage wird eine Trace-Id generiert, die dann an den Backend geschickt wird.

**AC 9.2:** Die Trace-Id wird konsquent durch alle Dienste hindurch weitergeleitet und in den Logs festgehalten.

**AC 9.3:** Es kann ein Diagramm der Service-Aufrufe erstellt werden, indem man die Trace-IDs als Grundlage verwendet.

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Unit Testing | 4 | 10 |
| Beschreibung |  |  |

Als ein **Entwickler** möchte ich, dass **die neuen Funktionalitäten durch automatisiertes Testing abgedeckt sind**, damit **ich die Wirksamkeit überprüfen kann**.

|  |
| --- |
| Acceptance Criteria |

**AC 10.1:** Für die neuen Funktionen wurden folgende Unit-Tests implementiert: Positiv- Negativszenarien, Null/Non-Null Tests für Datensätze, Empty/Not-Empty für Collections.

**AC 10.2:** Die Unit-Tests wurden im Backend, sowie im Frontend implementiert.

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Wiederverwendbarkeit | 2 | 11 |
| Beschreibung |  |  |

Als ein **Entwickler** möchte ich, dass **UI-Komponenten wiederverwendbar sind**, damit **ich sie in der Zukunft gebrauchen kann**.

|  |
| --- |
| Acceptance Criteria |

**AC 11.1:** Die Komponenten sind so aufgebaut, dass sie aus kleineren und einfacheren Komponenten bestehen.

1. Planen

Planen ist die zweite Phase des IPERKA-Vorgehenmodels. Es wird geplant und mit verschiedenen Varianten aufgezeigt, wie man von Istzustand zum Sollzustand kommt.

* 1. Realisierungskonzept

Ein Realisierungskonzept dient dazu, die Aufgabe bis in Detail zu planen. Der Zweck ist von der bisherigen Übersicht eine Strategie zu entwickeln, um einen Leitfaden zu haben für eine koordinierte Umsetzung. Das Konzept wurde in drei Unterthemen aufgeteilt, damit man eine erleichterte Übersicht der Aktivitäten zwischen den Schnittstellen hat.

* + 1. Soll-Zustand

Einfach gesagt besteht der ganze Prozess aus drei Hauptaufgaben: Kundenerstellung abbrechen, Kundenerstellung fortfahren und E-Mail versenden bei Fristablauf. Dazu gibt es noch drei Zusatzfunktionalitäten: Sortieren, Filtrieren und Hervorheben bei Fristablauf. Bei Errors wird der Benutzer darauf hingewiesen, durch Meldungen in der Applikation.   
Die Implementierung wird benutzerfreundlich entwickelt, mittels Server Functions im Frontend und durch klare Hinweise auf der Website. Zusätzlich wird der Code durch Logging mittels Slf4j versehen, auch durch Tracing mittels OpenTelemetry und Grafana zur Darstellung. Im Frontend werden Component Tests geschrieben mit Jest, einer Unit-Testing Library und im Backend mit JUnit.

Im folgenden Diagramm wurde dargestellt, welche Methoden wo implementiert werden müssen in der aktuellen Applikationsstruktur.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 9: Realisierungskonzept Diagramm

* + 1. Frontend

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

Automatisch generierte Beschreibung**Neuer Button "Später Fortfahren" im Kunden-Erstellungs-Formular:** Ein neuer Button mit der Bezeichnung «Später Fortfahren» wird im Kunden-Erstellungs-Formular hinzugefügt. Wenn der Benutzer auf diesen Button klickt, werden die bisher eingegebenen Daten ohne Validierung in der Datenbank gespeichert. Diese Daten werden in einem Zwischenspeicher abgelegt, um später vom Benutzer wieder aufgerufen werden zu können. Um dies zu implementieren, muss im Kunden-Erstellungs-Komponenten eine neue Methode geschrieben werden, die auf einen ebenfalls neuen Endpoint im Backend zugreift. Dieser Endpoint persistiert dann die ausgefüllten Daten in der Datenbank, die angepasst werden muss.

Abbildung 10: Use Case Workflow Zwischenspeicherung

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

Automatisch generierte Beschreibung**Pendente Tasks auf der Task-Seite:** Auf der Task-Seite werden alle pendenten Tasks aufgelistet. Jeder Task wird mit einem «Resume» Button versehen, der es dem Benutzer ermöglicht, mit dem entsprechenden Task fortzufahren.

Abbildung 11: Use Case Anzeige in Task Liste

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

Automatisch generierte Beschreibung

Nach dem Klicken auf den Button wird der Benutzer weitergeführt auf die Kunden-Erstellen Seite und die bestehenden Daten werden vorbefüllt in das Formular. Dafür muss eine Funktion im Frontend programmiert werden, die auf ein Backend-Endpoint zugreift, Daten aus der Datenbank herausliest und sie in die korrekten Felder im Formular hineinschreibt.

Abbildung 12: Use Case Formular Vorbefüllen

**Sortier- und Filterfunktion:** Eine Sortierfunktion wird implementiert, die es dem Benutzer ermöglicht, die Liste der pendenten Tasks nach verschiedenen Kriterien wie z. B. «Description» auf- oder absteigend zu sortieren. Zusätzlich wird eine Filterfunktion integriert, die es dem Benutzer ermöglicht, nach spezifischen Einträgen zu suchen. Wenn z.B. "cust" eingegeben wird, werden nur Einträge angezeigt, die den Suchbegriff enthalten.

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

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

Abbildung 13: Use Case Task Table Filter

Abbildung 14: Use Case Task Table Sortierung

**Variable für die Frist von pendenten Tasks:** Es wird eine Variable definiert, die festlegt, wie lange eine Aufgabe pendent sein kann. Diese Variable ist im Code veränderbar und anpassbar. Sobald die definierte Frist von N-Tagen abläuft, wird der Benutzer visuell auf die Frist aufmerksam gemacht. Die Variable wird dem DtoMapper hinzugefügt und wird über die Application-Settings veränderbar sein.

* + 1. Backend

**Versenden von Erinnerung E-Mails:** Nach Ablauf der definierten Frist wird automatisch eine Erinnerungs-E-Mail an den Benutzer gesendet. Diese E-Mail enthält einen Link, der den Benutzer direkt zur entsprechenden Aufgabe weiterleitet, um mit dieser fortzufahren. Hierfür wird der bestehende E-Mail-Service von OVWeb benutzt, die mit der neuen N-Frist Variable konfiguriert werden muss.

Ein Bild, das Text, Diagramm, Reihe, Kreis enthält.

Automatisch generierte Beschreibung

Abbildung 15: Use Case E-Mail-Versenden

* 1. Testkonzept

Mit dem folgenden Testkonzept wird überprüft, ob die Implementierung korrekt umgesetzt wurde. In der Phase «Kontrollieren» werden die Testfälle getestet. Das Konzept ist den User Stories und deren Acceptance Criteria aufgebaut. Auf Mobile-Tests wurde verzichtet, da es sich um eine Desktop-Applikation handelt.

* + 1. Testziele
* Das Ziel ist es, die Funktionalität und die Benutzerfreundlichkeit des Workflow Unterbruch zu testen.
* Es werden Unit-Tests bevorzugt für das Überprüfen und die restlichen Kriterien werden mit manuellen Akzeptanzkriterien abgedeckt.
  + 1. Testumgebung

Die Tests werden lokal auf dem PC innerhalb der Generali AG durchgeführt. Die Springboot Applikation, das Next.js Frontend sowie eine Instanz der aktuellen Version der Datenbank und Keycloak werden per Docker gestartet und in einem Chrome-Browser getestet. Es wird dieser Browser benutzt, da es der geschäftliche Standardbrowser ist. Dies widerspiegelt möglichst genau eine Produktionsumgebung.

* + 1. Testfälle

Die User-Story 10 besteht aus der Implementierung der Unit-Tests, daher ist es nicht nötig einen Testfall hier zu kreieren.

|  |
| --- |
| **ID: WU-1** |
| **Anforderung:** AC 1.1, AC 7.1  **Voraussetzung:** Benutzer ist im OVWeb angemeldet und befindet sich auf der Customer Creation Seite **Beschreibung:** Anzeige des «Später Fortfahren» Button **Ablauf:** - |
| **Erwartetes Resultat:** |
| Der Button ist sichtbar und kann einfach erreicht werden. |

|  |
| --- |
| **ID: WU-2** |
| **Anforderung:** AC 1.2  **Voraussetzung:** Benutzer ist im OVWeb angemeldet und befindet sich auf der Customer Creation Seite **Beschreibung:** Überprüfung, ob der «Später fortfahren» Button speichert **Ablauf:** Der Benutzer füllt den ersten Schritt des Formulars aus und klickt dann den Button. |
| **Erwartetes Resultat:** |
| Das System speichert die Anfrage in der Datenbank. |

|  |
| --- |
| **ID: WU-3** |
| **Anforderungen:** AC 1.2, AC 7.2  **Voraussetzung:** Benutzer ist im OVWeb angemeldet und befindet sich auf der Customer Creation Seite **Beschreibung:** Überprüfung, ob der «Später fortfahren» Button eine Fehlermeldung ausgibt. **Ablauf:** Der Benutzer füllt nichts aus und klickt dann den Button. |
| **Erwartetes Resultat:** |
| Es kommt eine Fehlermeldung. |

|  |
| --- |
| **ID: WU-4** |
| **Anforderungen:** AC 2.1  **Voraussetzung:** Benutzer ist im OVWeb angemeldet, hat eine Aufgabe in der Datenbank und befindet sich auf der Task Seite **Beschreibung:** Überprüfung der Task Details **Ablauf:** - |
| **Erwartetes Resultat:** |
| Datum/Uhrzeit, Objekt und/oder der Name des Kunden sind alle sichtbar in der Aufgabenliste. |

|  |
| --- |
| **ID: WU-5** |
| **Anforderungen:** AC 2.2  **Voraussetzung:** Benutzer ist im OVWeb angemeldet, hat eine Aufgabe in der Datenbank und befindet sich auf der Task Seite **Beschreibung:** Überprüfung, ob der «Fortfahren» Button funktioniert. **Ablauf:** Der Benutzer klickt auf den «Fortfahren» Button |
| **Erwartetes Resultat:** |
| Der Benutzer wird weitergeleitet auf den letzten Stand der Aufgabe. |

|  |
| --- |
| **ID: WU-6** |
| **Anforderungen:** AC 3.1 - 3.2  **Voraussetzung:** Benutzer ist im OVWeb angemeldet, hat eine Aufgabe in der Datenbank, befindet sich auf der Task Seite und sieht den «Fortfahren» Button  **Beschreibung:** Überprüfung, ob das Formular mit den bestehenden Daten vorbefüllt wird. **Ablauf:** Der Benutzer klickt auf den «Fortfahren» Button |
| **Erwartetes Resultat:** |
| Nach dem Weiterleiten wird das Formular mit den bestehenden Daten vorbefüllt. |

|  |
| --- |
| **ID: WU-7** |
| **Anforderungen:** AC 4.1  **Voraussetzung:** Benutzer ist im OVWeb angemeldet, hat eine Aufgabe in der Datenbank, befindet sich auf der Task Seite **Beschreibung:** Überprüfung ob die Liste sortiert werden kann. **Ablauf:** Der Benutzer klickt auf das «Name» Feld |
| **Erwartetes Resultat:** |
| Die Felder werden alphabetisch aufsteigend sortiert. |

|  |
| --- |
| **ID: WU-8** |
| **Anforderungen:** AC 5.1 – 5.2  **Voraussetzung:** Benutzer ist im OVWeb angemeldet, hat eine Aufgabe in der Datenbank, befindet sich auf der Task Seite. **Beschreibung:** Überprüfung**,** ob die Liste filtriert werden kann. **Ablauf:** Der Benutzer gibt «Customer» in das Suchfeld ein. |
| **Erwartetes Resultat:** |
| Es werden nur Aufgaben angezeigt, die «Customer beinhalten». |

|  |
| --- |
| **ID: WU-9** |
| **Anforderungen:** AC 6.1 – 6.2, AC 2.3  **Voraussetzung:** Benutzer ist im OVWeb angemeldet, hat eine Aufgabe in der Datenbank, die sich nach der N-Frist befindet. Und im Backend ist die E-Mail des Testers hardcoded.  **Beschreibung:** Überprüfungdes E-Mail-Versands, sowie der visuellen Hervorhebung. **Ablauf:** Der Benutzer wird automatisch von System benachrichtigt. |
| **Erwartetes Resultat:** |
| Das Feld wird visuell hervorgehoben  Der Benutzer erhält eine E-Mail mit einem Link auf die Taskliste. |

|  |
| --- |
| **ID: WU-10** |
| **Anforderungen:** AC 8.1 - 8.2  **Voraussetzung:** Workflow wurde unterbrochen **Beschreibung:** Überprüfung des Loggings **Ablauf:** - |
| **Erwartetes Resultat:** |
| Die Logs wurden durchgeführt, sind strukturiert und enthalten alle notwendigen Informationen. |

|  |
| --- |
| **ID: WU-11** |
| **Anforderungen:** AC 9.1 – 9.3  **Voraussetzung:** Es wurden mehrere Anfragen, vom Frontend aus, gesendet. **Beschreibung:** Überprüfung der Anfragen mit Trace-Id **Ablauf: -** |
| **Erwartetes Resultat:** |
| * Jede Anfrage hat eine einzigartige Id, die in dazugehörigen Services angehängt ist. * Es kann ein Diagramm erstellt werden mit der Trace-Id und den dazugehörigen Services. |

|  |
| --- |
| **ID: WU-12** |
| **Anforderung:** AC 11.1 - 11.2  **Voraussetzung:** UI-Komponenten sind implementiert **Beschreibung:** Wiederverwendbarkeit der UI-Komponenten **Ablauf:** - |
| **Erwartetes Resultat:** |
| Die UI-Komponenten setzten sich aus mehreren kleineren Komponenten zusammen und sind im Storybook dokumentiert. |

* 1. Aktivitätsdiagramme

Um den Prozess zu veranschaulichen, wurden zwei Aktivitätsdiagramme erstellt. Eins für den Unterbruch und ein anderes für das Fortsetzen der Erstellung eines Kunden. Es dient als Grundlage, um sicherzustellen, dass alle Projektbeteiligten vom Gleichen reden.

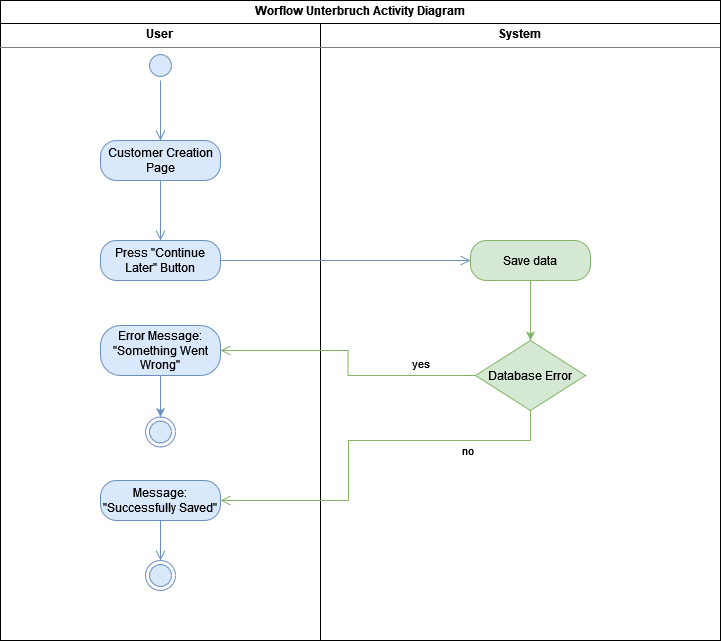


Abbildung 16: Aktivitätsdiagramm Workflow Unterbruch

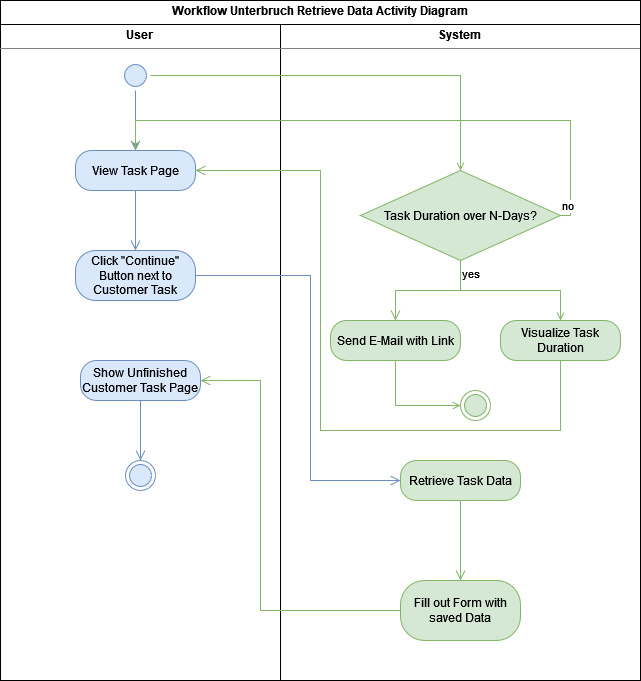
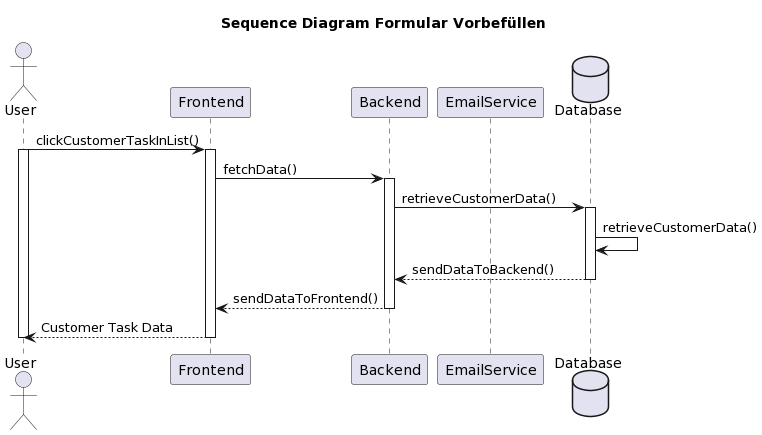


Abbildung 17: Aktivitätsdiagramm Pendenzenliste

Die Aktivitätsdiagramme sind den Use Cases nach aufgebaut, es wird aufgezeigt, wie der Vorgang abläuft, wenn der User Funktionen auslöst und was somit im System passieren soll.

* 1. Sequenzdiagramme

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

Automatisch generierte BeschreibungDas Sequenzdiagramm verhilft eine Übersicht zu geben, wie die Schnittstellen miteinander kommunizieren, vom User bis zur Datenbank. Dazu ergibt sich auch ein tieferes Verständnis, was für Methoden gebraucht werden.

Abbildung 18: Sequenzdiagramm Formular Vorbefüllen

Abbildung 19: Sequenzdiagramm Workflow Zwischenspeicherung

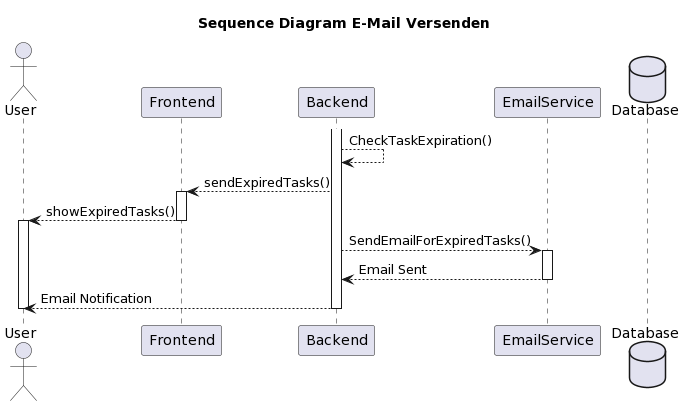


Abbildung 20: Sequenzdiagramm: E-Mail-Versenden

* 1. Ein Bild, das Schwarz, Dunkelheit enthält.

     Automatisch generierte BeschreibungKlassendiagramm

Für die Implementierung des Workflow Unterbruchs muss eine neue Subklasse «ResumeCustomerWorkflow» an Task hinzugefügt werden, um die Zwischenspeicherung sowie die Fortsetzung zu ermöglichen. Die Subklasse soll die üblichen Taskdaten: «description», «creationDate», «dueDate», «owner» und «state» beinhalten. Dabei müssen auch alle Felder aus der Customer Entity implementiert werden, damit diese persistiert werden können.

Abbildung 21: ResumeCustomerWorkflow Klassendiagramm Soll-Zustand

* 1. Datenmodell

Die Datenbank muss wie das Java-Model in «Task» erweitert werden. Die neue «ResumeCustomerWorkflow» Sub-Entity wird via eine Flyway-Migration dem Schema hinzugefügt.

* 1. Design

Der Sinn der Mockups ist es, ein besseres Verständnis wie auch einen Überblick zu verschaffen, wie alles am Ende aussehen soll und was alles zu implementieren ist. Dabei wird die User Experience besonders in Betracht bezogen. Die Mockups stellen die zwei zu bearbeitenden Seiten dar: Kundenerstellung und Pendenzenliste. Für die Umsetzung des User Interfaces für die Pendenzenliste, im Fall der Fristablauf nach N-Tagen, wurden zwei Varianten in Figma umgesetzt. Zusätzlich wurde eine Seite erstellt für den Fall, dass ein Error auftreten sollte.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Abbildung 22: Kundenerstellung

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Abbildung 23: Error Message

* + 1. Variante 1

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 24: Fristablauf Variante 1

* + 1. Variante 2

Ein Bild, das Text, Screenshot, Schrift, Design enthält.

Automatisch generierte Beschreibung

Abbildung 25: Fristablauf Variante 2

1. Entscheiden

Direkt nach dem Planen muss über die verschiedenen Lösungsansätze entschieden werden. Es wurden zwei UI-Varianten als Mockups in Figma dargestellt. In diesem Kapitel werden die beiden Möglichkeiten zusammen verglichen und es wird eine Entscheidung getroffen, welche umgesetzt wird. Somit behaltet man ein klares Bild für die Realisierung.

* 1. Entscheid

Die beiden Varianten sind unter Kapitel 10.7.1 respektiv 10.7.2 zu finden.

Die Versionen sind im Grunde gleich, der einzige Unterschied, der besteht ist das in der zweiten Version noch ein Pop-up hinzugefügt wird. Zugleich ist das benutzerfreundlicher, aber auch komplexer zu implementieren. Da stellt sich die Frage, ob die erste Variante nicht schon benutzerfreundlich genug ist. Ein Argument dafür ist, dass das Feld mit der abgelaufenen Frist im starken Kontrast zu den anderen steht und es die Funktionalität klar darstellt.

* + 1. Entscheidungsmatrix

Um die Varianten noch besser vergleichen zu können, wurde eine Entscheidungsmatrix erstellt. Damit kann man die Varianten anhand verschiedener Kriterien benoten (1 – 6) und die bessere auswählen.

Es wurden die folgenden 4 Kriterien ausgewählt:

* Benutzerfreundlichkeit: Wie intuitiv ist das Design für den Benutzer?
* Ästhetik: Welches Design ist visuell ansprechender?
* Konsistenz: Entspricht das Design der Generali Styleguide?
* Komplexität: Wie schwierig ist es, das UI zu programmieren?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Variante 1 | | Variante 2 | |
| **Kriterium** | **Gewichtung** | **Note** |  | **Note** |  |
| Benutzerfreundlichkeit | 0.3 | 4.5 |  | 5 |  |
| Ästhetik | 0.2 | 4.5 |  | 5 |  |
| Konsistenz | 0.2 | 5 |  | 4.5 |  |
| Komplexität | 0.3 | 6 |  | 4 |  |
| **Total Note:** |  |  | **5.1** |  | **4.6** |

Somit gewinnt die erste Variante, da sie dem Benutzer die abgelaufene Frist genug klar zeigt, obwohl es nicht die schönere UI ist. Gleichzeitig wird es auch weniger Zeit in Anspruch nehmen, was mehr Zeit für andere Funktionalitäten bieten wird.

1. Realisieren
   1. Ein Bild, das Text, Screenshot, Schrift, Dokument enthält.

      Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Zahl enthält.

      Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift enthält.

      Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Zahl enthält.

      Automatisch generierte BeschreibungProjektumgebung

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 26: Projektumgebung

Hier ist die Projektumgebung aufgelistet, für die Aufgabe mussten Server Actions im «action» Ordner, create-customer/[resumeTaskId], Unit Tests und Storybook erstellt werden, ansonsten wurden die folgenden Dateien bearbeitet: **«TaskService»**, **«Task»**, **«TaskGetDto»**, **«TaskPostDto»**, **«TaskController»**, **«create-customer-form»**, **«task-table»** und **«tasks/columns»**.

* 1. Datenbank und Model

Um die Daten richtig persistieren zu können, sind Anpassungen im Model und in der Datenbank nötig. Mithilfe von Flyway wurden Datenbankänderungen vorgenommen, es wurde eine neue Sub-Entity in der Task Tabelle erstellt. Die neue Tabelle heisst **«task$resume\_customer\_workflow»** und beinhaltet die gleichen Spalten, wie die Customer Tabelle, da ein Teil des Customers-Objekts dort gespeichert werden soll. Hierfür wurde die Datei **«V1\_0\_16\_\_add\_resume\_customer\_workflow\_to\_task.sql»** erstellt. Weiter musste die Task-Tabelle erweitert werden mit einem neuen Enum für den Status: «Resumed», somit kann unterschieden werden, welcher Task abgebrochen wurde und an welchem noch gearbeitet wird.

Flyway erleichtert das Versionsmanagement der Datenbankstruktur, indem es SQL-Skripte, wie diese auf automatisierte Weise während des Hochfahrens der Applikation ausführt.

Danach mussten auch die beiden Dtos angepasst werden, **«TaskGetDto»** und **«TaskPostDto»**. Die GetDto Klasse wird für GET-Requests gebraucht, die an Task requested werden, andererseits ist die PostDto für die POST-Requests nötig. Der Hauptunterschied liegt darin, dass die **«TaskGetDto»** noch eine Id benötigt. In die beiden Klassen musste **«ResumeCustomerWorkflowGetDto»** respektiv **«…PostDto»** hinzugefügt werden, mit den gleichen Daten wie im Model. Da die Antwort als JSON zurückkommt und es schon zwei andere Task-Typen gibt, muss man zwischen ihnen unterscheiden können. Dafür wurde mit Jackson gearbeitet, einer Library für Java, die für JSON-Handling zuständig ist.

*TaskGetDto.java*

|  |
| --- |
| @DiscriminatorMapping(  value = "ResumeCustomerWorkflowGetDto",  schema = TaskGetDto.ResumeCustomerWorkflowGetDto.class) |

Mit dieser Annotation kann man die Type Eingenschaft des JSON-Responses korrekt mappen und somit sicherstellen, dass die Datensätze in der richtigen Dto serialisiert werden.

* 1. Controller und Service

Damit das Backend mit dem Frontend kommunizieren kann, müssen die Controller und Services angepasst werden im Backend. Ein Controller ist dafür zuständig, die einkommenden Requests zu verarbeiten, er liest die Daten im Request, kontrolliert den Ablauf der Applikation und schickt eine Response zurück. Dafür konsumiert er Methoden, die von dem Service bereitgestellt werden, sie beinhalten die Logik des Programms. Für die Realisierung dieser Aufgabe musste der **«TaskService»** angepasst werden, sowie der **«TaskController»**. Für die Speicherung gab es schon einen Endpoint, den **createTask**, aber für weitere Funktionalitäten, wie die Fortsetzung wurde eine neue Funktion geschrieben: **«resumeTask»**. Diese Methode benutzt die Jpa-Repository **«TaskRepository»**, um in der Datenbank den Task herauszulesen, der die angefragte Id hat. Falls es einen Fehler gibt, wird dieser behandelt und falls es funktioniert, wird der Task herausgelesen.

* 1. Workflow Zwischenspeicherung

Für die Speicherung der Task-Daten gibt es eine Funktion im Backend, die **«createTask»**. Damit konnte der Workflow abgebrochen werden. Neben den Erweiterungen am Model und den Dtos musste nur das Frontend angepasst werden für die Zwischenspeicherung. Im **«CustomerCreateForm.tsx»** wurde ein Knopf hinzugefügt namens **«Später Fortfahren»**, beim Klicken wurden alle bisher ausgefüllten Daten abgespeichert in der Datenbank. Um die Persistierung zu ermöglichen, musste die Backend-Methode im Frontend implementiert werden mittels einer Server-Action: **«create-task.ts»**. Die Methode spricht das Backend an mit den Daten und sendet es weiter.

Da das TaskPostDto auch noch den Typ verlangt, musste dieser mitgegeben werden, hardcoded als **«type: ‘ResumeCustomerWorkflowPostDto’»**.

Um die Bedienung zu vereinfachen, bekommt der Benutzer bei jeder Anfrage einen Toast, also eine Meldung zurück, ob die Anfrage entweder erfolgreich war oder fehlgeschlagen hat.

create-customer-form.tsx

|  |
| --- |
| const cancelCustomerWorkflow = async (data: Input) => {  const taskPostDto: CreateTaskRequest = {  taskPostDto: {  type: 'ResumeCustomerWorkflowPostDto',  description: 'Customer Create Workflow',  dueDate: new Date(),  title: data.title,  salutation: data.salutation,  gender: data.gender,  …  }  }  try {  await createTask(taskPostDto)  toast({  title: 'Task wurde erfolgreich gespeichert!',  variant: 'success'  })  router.push('/tasks')  } … |

Der Code-Ausschnitt zeigt, dass die Daten aus dem Formular herausgelesen und mittels der **«createTask»** Methode via ein POST-Request an das Backend geschickt werden.

* 1. Workflow Fortsetzung

Für die Fortsetzung musste eine Get- sowie eine Update-Methode gebraucht werden, da man die Daten vorbefüllen will im Formular (Get) und die Speicherung der Änderungen persistieren möchte (Update). Die Update-Methode wird auf dem Task aufgerufen, sobald man auf den «Resume» Knopf klickt in der Task-Liste, gleichzeitig wird man weitergeleitet auf die «CustomerCreateForm» mit der Id des Tasks, die durch PageProps mitgegeben wird und die Daten werden durch die Get-Funktion vorbefüllt.

CreateCustomer/[resumeTaskId]/page.tsx

|  |
| --- |
| interface PageProps {  params: {resumeTaskId: number} }  function Page(props: PageProps) {  return <CreateCustomerForm resumeTaskId={props.params.resumeTaskId}></CreateCustomerForm> } |

Durch dieses Prop Drilling wird der «ResumeTaskId» Datensatz an der «CreateCustomerForm» Komponente weitergeschickt, somit wird die Id in der Komponente erkannt und die Daten können vorbefüllt werden.

Die Daten werden mit dem form.reset Hook in ihre respektive Felder hineingefügt, falls es einen Eintrag dafür gibt in der Datenbank. Da die Namensgebung des Form Inputs nicht dem des TaskDto entspricht, muss dieser mit der «resumeTaskDtoToInput» Methode umgewandelt werden, damit sich die jeweiligen Daten auch finden.

Der useAsyncEffect-Hook wird verwendet, um die Taskdaten abzurufen, wenn die Komponente mounted, also wenn der Status verändert oder nachverfolgt wird oder wenn sich die resumeTaskId-Prop ändert und TaskId sowie form.reset vorhanden sind. Die Funktion getTaskById wird mit der resumeTaskId aufgerufen, um die Taskdaten abzurufen.

create-customer-form.tsx

|  |
| --- |
| if (resumeTaskId) {  const customerData = await getTaskById(resumeTaskId)  if (customerData.type === 'ResumeCustomerWorkflowGetDto') {  const formData = resumeTaskDtoToInput(customerData)  form.reset(formData)  } } |

Es wird nur vorbefüllt falls der Benutzer auf den Resume Knopf klickt in der Pendenzenliste, da bei einem gewöhnlichen Aufruf der Seite eine teilbefüllte Form nicht erscheinen soll.

* 1. Pendenzenliste

An der Pendenzenliste mussten mehrere neue Funktionalitäten umgesetzt werden. Die neuen Implementierungen wurden in die folgenden drei Unterthemen geordnet: Sortierung, Filter und Anzeige.

* + 1. Sortierung

Die Sortierung wird mittels der **«Tanstack/react-table»** Library ermittelt.

Zuerst wird der Sortierstatus in der Variable **«SortingState»** gespeichert. Dabei handelt es sich um ein Array von Objekten, wobei jedes Objekt eine Spalte und die Richtung der Sortierung (aufsteigend oder absteigend) darstellt. Es wird als leeres Array initialisiert, d. h. es wird zunächst keine Sortierung vorgenommen.

task-table.tsx

|  |
| --- |
| const [sorting, setSorting] = React.useState<SortingState>([]) |

Danach wird die **«onSortingChange»**-Anweisung des **«useReactTable»**-Hooks auf die **«setSorting»**-Funktion gesetzt. Das bedeutet, dass jedes Mal, wenn eine Sortieraktion in der Tabelle durchgeführt wird (Klicken auf einen Header zum Sortieren), wird die Funktion **«setSorting»** mit dem neuen Sortierstatus aufgerufen.

In den Spaltendefinitionen (columns.tsx) wird für jede Spalte eine Knopf-Komponente als Überschrift verwendet. Diese Schaltfläche hat einen **«onClick»**-Handler, der **«column.toggleSorting»** aufruft. Die Funktion **«toggleSorting»** wird von der Bibliothek **«Tanstack/react-table»** bereitgestellt und schaltet den Sortierstatus der Spalte um, wenn sie aufgerufen wird. Wenn die Spalte nicht sortiert war, wird sie in aufsteigender Reihenfolge sortiert. War sie aufsteigend sortiert, wird sie absteigend sortiert und umgekehrt.

tasks/columns.tsx

|  |
| --- |
| <Button  variant="ghost"  onClick={() =>  column.toggleSorting(column.getIsSorted() === 'asc')  } >  Id  <ArrowUpDown className="ml-2 h-4 w-4" /> </Button> |

Die Funktion **«getSortedRowModel»** wird dem **«useReactTable»**-Hook zur Verfügung gestellt. Diese Funktion ist für die Sortierung der Zeilen auf der Grundlage des aktuellen Sortierstatus verantwortlich. Die sortierten Zeilen werden dann zum Rendern der Tabelle verwendet.

Kurz gesagt, wenn eine Spaltenüberschrift angeklickt wird, wird der Sortierstatus aktualisiert, was eine Neudarstellung der Tabelle auslöst. Während dieser Neudarstellung wird die Funktion **«getSortedRowModel»** mit dem aktualisierten Sortierstatus aufgerufen und gibt die nach dem neuen Status sortierten Zeilen zurück. Diese sortierten Zeilen werden dann zum Rendern der Tabelle verwendet.

* + 1. Filter

Der Filter ist ähnlich aufgebaut wie die Sortierung. Am Anfang wird ein Array von Objekten namens **«columnFilters»** erstellt, wo jedes Objekt einer Spalte und Filterwert entspricht. Es wird leer initialisiert, sodass keine Filtrierung vorerst stattfindet.

task-table.tsx

|  |
| --- |
| const [columnFilters, setColumnFilters] = React.useState<ColumnFiltersState>([]) |

Die Prop **«onColumnFiltersChange»** des **«useReactTable»** Hooks wird auf die Funktion **«setColumnFilters»** gesetzt. Das bedeutet, dass die Funktion **«setColumnFilters»** mit dem neuen Filterstatus aufgerufen wird, sobald eine Filteraktion in der Tabelle durchgeführt wird (Sucheingabe).

In der TaskTable wird eine Input-Komponente für die Suche in der Spalte **«description»** verwendet. Diese Eingabe hat einen **«onChange»** Handler, der **«setFilterValue»** in der **«description»** Spalte mit dem neuen Eingabewert aufruft. Diese Funktion **«setFilterValue»** aktualisiert den Filterwert der Spalte, wenn sie aufgerufen wird.

Zuletzt wird die Tabelle mit der **«getFilteredRowModel»** Funktion gerendert. Sie ist dafür zuständig, die Zeilen zu filtern basiert auf den jeweiligen Filtering Status.

task-table.tsx

|  |
| --- |
| <Input  placeholder="Search"  value={table.getColumn('description')?.getFilterValue() as string}  onChange={event => table.getColumn('description')?.setFilterValue(event.target.value)}  className="w-full md:max-w-sm" /> |

Im aktuellen Zustand ist es nicht möglich, alle Felder in der Tabelle zu filtrieren, dafür müsste eine Global-Filter-Methode implementiert werden, die auf alle Elemente zugreifen könnte.

* + 1. Anzeige

Falls ein Task über die Frist hinausgegangen ist (creationDate + reminderDays), muss dieser visuell markiert sein in der Tabelle.

task-table.tsx

|  |
| --- |
| <TableRow  key={row.id}  data-state={  row.getIsSelected() && 'selected'  }  className={  row.original.state === 'RESUMED'  ? 'bg-gray-200'  : row.original.needsReminder  ? 'bg-red-200'  : ''} > |

Der Code-Abschnitt zeigt die Stelle, wo geschaut wird, ob der Task visuell markiert werden soll oder nicht. Das «row.original» beinhaltet alle Daten von TaskGetDto und «needsReminder» ist ein boolean. Falls «needsReminder» true ist, wird der Hintergrund der Zeile auf Rot gesetzt, um den Task von den anderen zu unterscheiden. Gleichzeitig wird geschaut, ob der Task «Resumed» ist. Ist er Resumed, wird der Hintergrund auf Grau gesetzt und man kann den Task nicht mehr bearbeiten.

Die Variable «needsReminder» wird im Backend in TaskGet erstellt und im «DtoMapper.java» überprüft, ob es auf true gesetzt werden soll.

DtoMapper.java

|  |
| --- |
| LocalDate reminderDate = entity.getCreationDate().plusDays(reminderDays); taskGetDto.setNeedsReminder(reminderDate.isBefore(LocalDate.now())); |

Der reminderDate ist in der **«application.yaml»** definiert mit Standardwert = 7. Initialisiert wird er im **«DtoMapper»**. Die Methode **«mapToGetDto»**, welche die obenstehende Rechnung braucht, wird im «TaskController.java» eingesetzt. Die Methode wird gebraucht, um Task-Objekte in **«TaskGetDto.TaskGet»** umzuwandeln, damit diese im erforderten Format des APIs sind.

Die **«getAllTasks»** Funktion des **«TaskController.java»** bezieht alle Tasks aus der Datenbank und wird im Frontend verwendet, um in der Tabelle alle Tasks anzuzeigen. Somit wird auch der **«needsReminder»** Boolean mitgeschickt und es kann darauf zugegriffen werden.

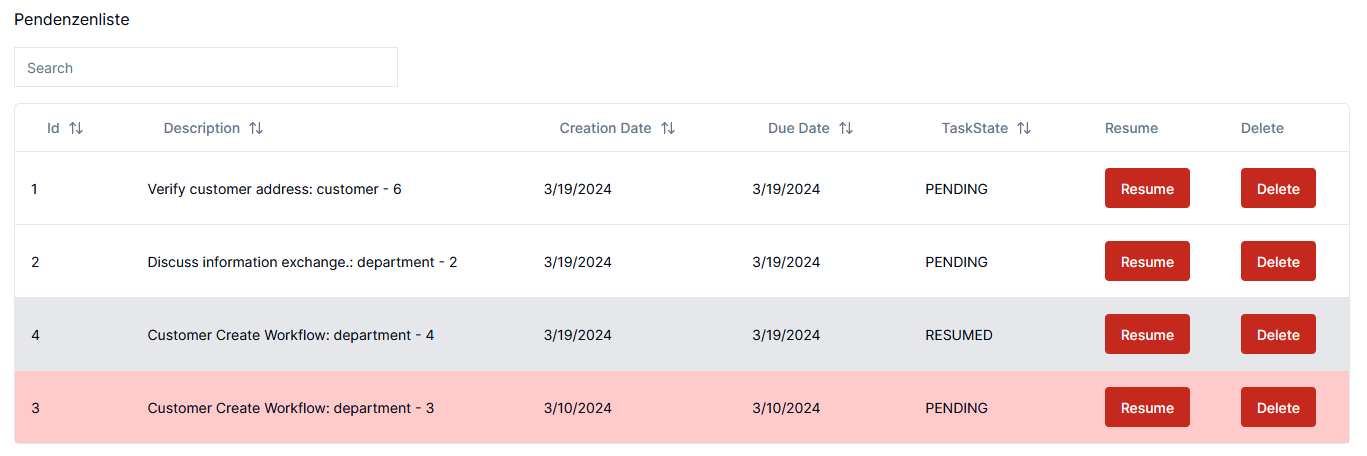


Abbildung 27: Task-Table

Das Design entspricht dem der Generali-Styleguide. Rot ist die Standardfarbe für anwendbare Funktionalitäten wie Buttons oder Tabs.

* 1. E-Mail Service

Um die E-Mails periodisch versenden zu können, damit der Benutzer benachrichtigt werden kann, wurde ein Scheduler eingebaut. Im Istzustand erlaubte die Applikation jedoch noch nicht eine solche Funktion einzubauen, deswegen musste in der Applikationsklasse **«OVWEBBackendApplication»** die folgende Annotation hinzugefügt werden: **«@EnableScheduling»**. Somit konnte der Support für periodische Methoden im System sichergestellt werden.

In der **«EmailService.java»** Klasse wurde eine neue Methode erstellt: **«sendReminderEmails»**, die mit **«@Scheduled»** annotiert ist. Die Methode wird jede Stunde aufgerufen. Als erstes liest die Methode alle Tasks aus der Datenbank heraus und kreiert eine neue Variable für den heutigen Tag: **«currentDate»**. Als nächstes wird für jede Pendenz mithilfe der **«reminderDays»** Variable, die in den Application Settings definiert ist, und dem Task **«creationDate»**, ob die Pendenz überfällig. Ist dies der Fall, wird eine E-Mail mit der **«sendEmail»** Methode an den Eigentümer des Tasks versendet.

Momentan verläuft die For-Loop immer alle Tasks durch und es wird somit jede Stunde eine Erinnerung verschickt für einen befristeten Task. Eine Verbesserung wäre einen Flag zu setzen, falls der Benutzer für einen Task schon erinnert wurde, z. B. **«Reminded»**, um dann nicht mehr darüber benachrichtigt zu werden.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 28: E-Mail Reminder

Im Postfach sieht die E-Mail dann so aus. Man erhält die wichtigsten Informationen sowie den Link für die Task-Liste. Der Link könnte spezifischer sein, z. B. **«localhost:3000/create-customer/{taskId}»**, da aber die Methode für alle Task Typen verfügbar sein soll und nicht nur für ResumeCustomerWorkflow, wurde es auf der obersten Ebene gelassen, und zwar **«localhost:3000/tasks»**, wo man alle Pendenzen sieht.

* 1. Logging

Dieses Kapitel widmet sich der Beobachtbarkeit und Nachverfolgbarkeit innerhalb des **«TaskService»** durch strukturiertes Logging. Mit der Annotation **«@Slf4j»**, welche durch die Lombok Bibliothek zur Verfügung gestellt wird, wird die Logger-Instanz automatisch in die Klasse eingefügt und erlaubt das Erfassen von systematischen, kontextreichen Log-Nachrichten. Hier ein Beispiel des Logging im Backend sowie Frontend:

TaskService.java

Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung

Abbildung 29: Log Response Backend

|  |
| --- |
| log.atError().addKeyValue("taskId", id).log("Task not found with id: {}", id); return new IllegalArgumentException("Task not found"); |

Hier ein Beispiel für die Struktur im Frontend:

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte BeschreibungUpdate-task.ts

Abbildung 30: Log Response Frontend

|  |
| --- |
| const logger = getLogger('ovweb-frontend') const logContext: Record<string, unknown> = {taskId: id, task} logger.info(logContext, 'Updating task') |

* 1. Distributed Tracing

Das Tracing wird im OVWEB von OpenTelemetry verwaltet.

**Frontend:**

Als Beispiel für das Tracing im Frontend wurde die **«updateTask»** Methode verwendet. Die Funktion **«withSpan»** wird verwendet, um einen neuen Span für die **«updateTask»** Operation zu erstellen. Diese Funktion nimmt den Namen des Spans, einige Anfangsattribute und eine Callback-Funktion entgegen. Die Callback-Funktion enthält den Code für den **«updateTask»** Vorgang.

Wenn die Operation **«updateTask»** fehlschlägt, werden der Span zusätzliche Attribute hinzugefügt, darunter der Antwortstatus, der Text des Antwortstatus und der Antwortkörper. Diese Attribute können für die Fehlersuche im Vorgang nützlich sein.

update-task.ts

|  |
| --- |
| return await withSpan('updateTask', {taskId: id}, async span => {  ... }); … span.setAttribute('responseStatus', responseStatus) span.setAttribute('responseStatusText', responseStatusText) span.setAttribute('responseBody', responseBody) |

**Backend:**

Um einen neuen Span im Backend auszulösen, kann die Annotation **«@NewSpan()»** verwendet werden.

TaskService.java

|  |
| --- |
| @NewSpan("createTask") public Task createTask(Task task, String ownerId) {} |

**Grafana:**

Die Traces können dann über das Grafana Dashboard angeschaut werden.

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

Automatisch generierte Beschreibung

Abbildung 31: Grafana Dashboard

* 1. Storybook

Storybooks werden verwendet, um verschiedene Zustände Ihrer Komponenten zu demonstrieren. Storybook ist ein Tool für die isolierte Entwicklung von UI-Komponenten für React, Next, Angular und andere. Es macht den Aufbau von UIs organisiert und effizient.

Für die Aufgabe wurden die Komponente **«create-customer-form.tsx»** und **«task-table.tsx»** bearbeitet und erweitert, deswegen wurde auch je ein Storybook für sie implementiert.

task-table.stories.tsx

In den Stories werden drei States beschrieben und wie sie sich verhalten. Da die Komponente ein Task Objekt benötigt, wird ein solches Objekt zuerst gemockt.

|  |
| --- |
| const mockData: TaskGetDto[] = [  {  id: 1,  description: 'Test Task',  dueDate: new Date(),  …}] export const WithTasks: Story = {  args: {  columns: columns,  data: mockData,  refreshData: () => Promise.resolve()}} |

**withTasks**: Dieser Status zeigt, wie sich die Komponente verhält, wenn sie eine Liste von Tasks (mockData) erhält. Die Funktion refreshData ist eine Mock-Funktion, die sofort aufgelöst wird. Es wird ein Table angezeigt mit dem gemockten Task.

**withoutTasks**: Dieser Status zeigt, wie sich die TaskTable-Komponente verhält, wenn ihr keine Aufgaben gegeben werden. Die Funktion refreshData ist eine Mock-Funktion, die sofort aufgelöst wird. Es wird eine leere Tabelle angezeigt.

**ErrorOnGet**: Dieser Status zeigt an, wie sich die TaskTable-Komponente verhält, wenn die Funktion refreshData mit einem Fehler abgewiesen wird. Dies passiert, wenn z. B. ein Fehler auftritt beim Abrufen von Tasks.

customer-create-form.stories.tsx

In dieser Story werden 4 States aufgezeigt: Empty, FailedValidation, Filled und die neue Funktion Prefilled.

**Empty**: Dieser Zustand zeigt, wie sich die Komponente CreateCustomerForm verhält, wenn sie keine Ausgangsdaten erhält.

**Prefilled**: Dieser Status zeigt an, wie sich die Komponente CreateCustomerForm verhält, wenn sie eine resumeTaskId von 1 erhält.

**FailedValidation**: Dieser Status zeigt an, wie sich die Komponente CreateCustomerForm verhält, wenn sie eine Reihe von Standardwerten für die Formularfelder erhält, die die Validierung nicht bestehen.

Hier sieht man den Empty State des CreateCustomerForm

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

Automatisch generierte Beschreibung

Abbildung 32: Storybook CreateCustomerForm Empty State

* 1. Benutzerfreundlichkeit

Der Workflow Unterbruch in OVWEB zeichnet sich durch eine klare Nutzerführung aus. Es ist simpel gestaltet. Der Button «Später Fortfahren» ist auf der Benutzermaske klar auffindbar. Die Texte/Labels sind in einfacher Sprache, die speziell für Versicherungsberater verständlich sind. Anstatt technische Details oder komplexe IT-Terminologie zu verwenden, sind die Formularfelder und Anleitungen in allgemein verständlicher Sprache gehalten. Auch die Meldungen zur erfolgreichen und fehlerhaften Abspeicherung/Fortfahrung im «AlertDialog» gibt dem Nutzer klare Rückmeldung bezüglich des Status seiner Anfrage.

* 1. Automatisiertes Testing
     1. Backend Unit Tests

Im Backend wurden die TaskServiceTest und EmailServiceTest getestet, um sowohl positive als auch negative Szenarien abzudecken. Dabei wurden verschiedene Arten von Inputs überprüft, wie null und non-null Skalarwerte, um die Robustheit und Zuverlässigkeit der Service-Logik zu gewährleisten.

Beispielsweise wird im «testCreateTaskWithInvalidOwner» sichergestellt, dass die richtige Exception geworfen wird, falls man versucht, einen Task zu kreieren mit einer ungültigen OwnerId.

Die Tests folgen dem Prinzip der geringsten Macht ("principle of least power"), indem sie sicherstellen, dass Funktionalitäten effizient auf der niedrigsten möglichen Ebene getestet werden, ohne unnötige IT-Tests zu implementieren.

* + 1. Frontend Unit Tests

Im Frontend wurde durch Jest-Tests die Funktionalität des Kunden-Erstellungs-Abbruchs und Fortsetzung geprüft sowie die Pendenzenliste. Dabei wurde insbesondere die Benutzerinteraktion mit dem Kunden-Erstellen-Formular getestet, wie im Test «always renders the "Später Fortfahren" Button» dargestellt, wo die Anwesenheit des Knopfs verifiziert wird. Weitere Tests, wie das erfolgreiche Absenden des Formulars und die Fehlerbehandlung bei der Formularübermittlung wurden implementiert, um eine vollständige Testabdeckung der neuen Funktionalitäten zu erreichen. Diese Tests simulieren Benutzereingaben und überprüfen, ob das System wie erwartet auf korrekte und inkorrekte Daten reagiert.

1. Kontrollieren
   1. Testkonzept

Auf das Testkonzept wird vertieft in Kapitel 10.2 eingegangen. Die deklarierten Testfälle werden in diesem Kapitel getestet.

* 1. Testprotokoll Samuel Hajnik

|  |  |
| --- | --- |
| 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 ID | Datum | Resultat | Kommentar |
| WU-1 | 19.03.2024 | Passed | Der Knopf wird angezeigt. |
| WU-2 | 19.03.2024 | Passed | Die Daten werden persistiert. |
| WU-3 | 19.03.2024 | Passed | Es kommt nur eine Fehlermeldung, falls in der Datenbank ein Fehler auftritt. |
| WU-4 | 19.03.2024 | Passed | Die wichtigsten Daten sind in der Liste zu sehen. |
| WU-5 | 19.03.2024 | Passed | Die Weiterleitung funktioniert. |
| WU-6 | 19.03.2024 | Passed | Die Daten werden vorbefüllt. |
| WU-7 | 19.03.2024 | Passed | Die Spalten werden alphabetisch auf- oder absteigend sortiert. |
| WU-8 | 19.03.2024 | Passed | Die Daten werden der Description nach filtriert. |
| WU-9 | 19.03.2024 | Passed | Pendenz visuell hervorgehoben, E-Mail angekommen. |
| WU-10 | 19.03.2024 | Passed | Logs funktionieren. |
| WU-11 | 19.03.2024 | Passed | Traces erfolgreich implementiert |
| WU-12 | 19.03.2024 | Passed | Storybooks vollständig |

* + 1. Testbericht

Dem Testprotokoll entsprechend sind alle manuellen Tests erfolgreich bestanden. Die Funktionen konnten wie geplant getestet werden und die Applikation reagierte bei den einzelnen Testfällen wie erwartet. Der Einsatz der neu implementierten Funktionen ist daher unbedenklich möglich und sicher. Nun folgt ein Fazit zu jedem der Testfälle.

* + 1. Testfall 1

**Fazit:** Die Anforderungen wurden erfolgreich erfüllt. Der Button ist auf jedem Schritt des Formulars klar ersichtlich und einfach zugänglich.

* + 1. Testfall 2

**Fazit:** Die eingegebenen Daten werden in der Datenbank gespeichert, daher wurden die Anforderungen abgedeckt.

* + 1. Testfall 3

**Fazit:** Da die Felder auch leer abgeschickt werden können, kommt kein Fehler bei einer leeren Abgabe. Jedoch wenn man in der Datenbank einen Fehler hat, z. B. mehrere Berater mit der gleichen Id, dann können die Daten nicht abgeschickt werden und man wird darauf hingewiesen.

Ein Bild, das Text, Schrift, Screenshot, weiß enthält.

Automatisch generierte Beschreibung

Abbildung 33: Fehlermeldung beim Speichern

* + 1. Testfall 4

**Fazit:** Der Task wird mit den Hauptdaten in der Liste angezeigt: Id, Description, Creation Date, Due Date und State. Zusätzlich sieht man zwei Knöpfe, einen zum Fortfahren mit dem Task und einen zum Löschen des Tasks.



Abbildung 34: Task in Taskliste

* + 1. Testfall 5

**Fazit:** Die Anforderungen wurden implementiert, der Benutzer wird weitergeleitet auf das Kunden Erstellen Formular mit der richtigen Task Id.

* + 1. Testfall 6

**Fazit:** Die Kundendaten aus dem Task werden im Formular vorbefüllt, sofern sie in der Datenbank bestehen.

* + 1. Testfall 7

**Fazit:** Beim einmal draufklicken werden die Felder aufsteigend sortiert, beim zweiten Mal absteigend und beim dritten Mal werden sie zum Default zurückgesetzt. Das funktioniert für alle Felder, ausser die Actions (Resume und Delete).

* + 1. Testfall 8

**Fazit:** Die Felder werden dem Suchbegriff nach filtriert. Doch der Filter bezieht sich nur auf die «Description», also auf den Namen des Tasks. Wenn man z. B. nach Tasks suchen will, die auf «Resumed» gesetzt sind, ist dies nicht möglich.

Ein Bild, das Text, Schrift, Reihe, Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 35: Taskliste Filter

* + 1. Testfall 9

**Fazit:** Die Tasks, die über die Frist hinaus sind, werden rot markiert in der Task-Tabelle, zusätzlich wird eine E-Mail verschickt, falls dies der Fall ist.

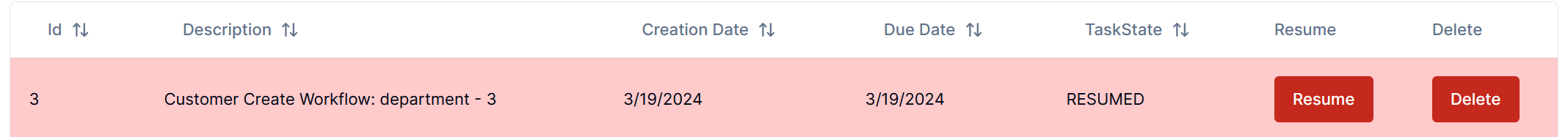


Abbildung 36: Task überfristet

* + 1. Testfall 10

**Fazit:** Das Logging funktioniert im Testfall 10 korrekt und liefert die notwendigen Informationen zur Überwachung und Fehleranalyse.

**Massnahmen/Empfehlungen:** Wenn neue Funktionen implementiert werden, sollte sichergestellt werden, dass alle relevanten Ereignisse erfasst werden und keine sensiblen Daten protokolliert werden.

* + 1. Testfall 11

**Fazit:** Die erfolgreiche Implementierung von Trace-IDs im Testfall 9 ermöglicht eine eindeutige Nachverfolgung von Anfragen durch das System.

* + 1. Testfall 12

**Fazit:** Die vollständige und klare Dokumentation im Storybook im Testfall 10 unterstützt die Wiederverwendbarkeit und Anpassbarkeit der UI-Komponenten.

1. Auswerten
   1. Vergleich Ist / Soll

Im IST-Zustand verfügte die OVWeb-Platform bereits über Funktionen wie Kundenerstellung und Taskliste. Allerdings war es den Beratern nicht möglich, den Erstellungsprozess zu unterbrechen. Der SOLL-Zustand erforderte die Implementierung eines Workflow-Unterbruchs, um diese Funktionslücke zu schliessen. Dieses Ziel wurde gemäss den Anforderungen flächendeckend umgesetzt und realisiert. Es konnten alle erforderten Funktionalitäten ausser den globalen Filter implementiert werden. Der neue Prozess wurde nahtlos in die bestehende Infrastruktur integriert und erfüllt die geforderten Funktionalitäten.

* 1. Fazit zur IPA

Insgesamt bin ich mit den letzten zehn Tagen zufrieden. Der Einstieg fiel mir leicht, da durch die Probe-IPA bekannt war, was auf mich zukommen würde, obwohl der Validierungsprozess sehr knapp war. Insgesamt bin ich mit den letzten zehn Tagen zufrieden. Der Ablauf der IPA war angenehm, ab und zu intensiv und das Dokument sowie die Implementation konnten in angemessener Zeit erarbeitet werden. Die Aufgabenstellung war klar formuliert und der Auftrag hatte wenige Abhängigkeiten, so konnte schnell abgeschätzt werden, wie viel Zeit für die einzelnen Aufgaben benötigt wird und es wurde im Zeitplan festgehalten.

Durch die genaue Herangehensweise beim Informieren und Planen, insbesondere beim detaillierten Definieren der User Stories und dem Erstellen der Diagramme, konnte eine solide Basis geschaffen werden, auf die während der Realisierung zurückgegriffen werden konnte. Beim Planen der Sollzeit wurde jedoch der Anspruch an Bugfixing nicht berücksichtigt. Dies hat sich als ein Problem herausgestellt, während des Realisierens, da ich zum Schluss vieles korrigieren musste im Code, was mich mehr Zeit als geplant gekostet hat. Dies ist wahrscheinlich darauf zurückzuführen, dass ich nicht genug weit vorausschauend geplant habe. Deswegen sollte ich in der Zukunft pro Aufgabe eine Stunde mehr einplanen oder am Ende noch eine Korrekturphase für den Code berücksichtigen.

Ich habe mich darauf verlassen, dass meine Planung ausreichend war und war sehr motiviert, mit der Umsetzung zu beginnen. Dies betont die Bedeutung einer gründlichen und ausführlichen Planungsphase. Während der zehn Tage habe ich konzentriert und zielgerichtet gearbeitet. Ich habe Ablenkungen, so gut es geht vermieden. Besonders in der ersten Woche bin ich gut vorangekommen und habe von morgens bis abends durchgearbeitet. In der zweiten Woche waren jedoch wieder mehr Mitlernende im Büro und ich fühlte mich mit dem bisher erreichten Stand ziemlich sicher, weshalb ich auch kleinere Pausen während der Arbeit zugelassen habe. Diese Unterbrechungen haben meinen Arbeitsfluss nicht wesentlich beeinträchtigt. Es ist jedoch wichtig, sich bis zum Ende diszipliniert und voll konzentriert auf die Abschlussphase der Arbeit zu fokussieren. Als Verbesserung sollte man den Filter zum Globalfilter ausbauen, aber für diese Umsetzung fehlte im Rahmen der IPA die Zeit.

Grössere Hilfestellungen waren während meiner Arbeit nicht nötig, abgesehen vom Storybook, welches mir Probleme zubereitete. Da musste ich Roman nach Hilfe fragen, einmal bei der Implementation und dann noch mal bei der Korrektur.

Das Schreiben des Berichts verlief reibungslos. Ich hatte genug Zeit eingeplant, habe aber auch mehr Zeit investiert, falls ich mal schneller fertig geworden war mit einer Realisierungsaufgabe. Das einzige Problem, mit dem ich zu kämpfen hatte, war die Korrektur. Ich lies den Bericht von meinem Vater gegenlesen und er hat mir das gleiche mitgeteilt, was ich auch beim zweiten Expertengespräch hörte, dass ab und zu die Fachsprache nicht eingehalten wird und dass man einige Erklärungen nicht versteht. Dennoch setzte ich mich durch und versuchte möglichst alle dieser Stilfehler zu beheben.

Hier und da gibt es noch Stellen, an welchen man den Code oder die Dokumentation verbessern könnte. Es ist allerdings klar, dass es immer besser ginge. Jedoch muss man immer die Zeit im Blick behalten und kann nicht Ewigkeiten an einem kleinen Detail verbringen.

Schlussendlich konnten fast alle Ziele der IPA erfüllt werden. Ich bin mit dem finalen Produkt zufrieden.

* 1. Reflexion

Ich blicke zurück auf eine sehr lehrreiche und intensive Zeit. Die IPA ist eine spannende, ich würde sogar sagen, eine tolle Erfahrung auf dem Weg zu meinem Lehrabschluss. Ich konnte meine erarbeiteten Fähigkeiten auf gesamter Breite testen und mich bestens für die Arbeitswelt vorbereiten.

In die IPA gestartet bin ich mit einer kleinen Ungewissheit, denn ich war mir nicht sicher, wie ich die Implementierung der E-Mail-Service angehen sollte. Deswegen habe ich auf die Informierungs- und Planungsphase konzentriert, um ein starkes und flächendeckendes Gerüst aufzubauen.

Während der ersten Tage hat dann diese Ungewissheit stetig abgenommen, da ich mit der Planung nicht nur meine Selbstsicherheit, sondern auch meine Motivation steigern konnte. Ich habe mich gut eingearbeitet und kam ohne grössere Probleme vorwärts. Der obligatorische Teil sowie auch die Analyse verliefen sehr gut und ich bin sehr zufrieden mit dem erarbeiteten Ergebnis. Der Teil der Realisierung fiel mir etwas schwerer. Dies aus dem Grund, weil ich auf einige nicht vorhergesehene Probleme gestossen bin, wie die Vorbefüllung des Formulars oder die Storybooks. Jedoch durch Hilfe und Nachlesen im Internet konnte ich eine passende Lösung implementieren. Somit habe ich auch bei diesem Teil ein gutes Gefühl und bin mit dem erarbeiteten Ergebnis zufrieden. Diejenigen Teile, die mit dem Projektabschluss in Verbindung stehen, beispielsweise das Testing und die Dokumentation, waren wieder einfacher für mich.

Gesamthaft gesehen ist die IPA eine sehr stärkende Erfahrung. Ich war während der gesamten IPA sehr positiv eingestellt, manchmal etwas gestresst oder verunsichert, jedoch immer zuversichtlich. Ich habe gelernt, dass ich schnell für Hilfe fragen muss, da ich ansonsten nicht nachkommen würde. Die Zeit der IPA werde ich so schnell nicht vergessen. Ich möchte nicht bald wieder eine solche Arbeit schreiben müssen, da die Belastung doch sehr gross ist. Ich bin nun sehr glücklich, mit dieser Arbeit fertig zu sein. Jetzt muss ich die Präsentation vorbereiten und das Erlernte nicht vergessen. Aus meiner Sicht ist diese ein gelungener Meilenstein in meiner Karriere.

Danksagung

Zum Ende dieser Arbeit möchte ich mich bei allen beteiligten Personen bedanken, die diese IPA ermöglicht haben. Insbesondere geht mein Dank an Roman Babenko, meinen Fachvorgesetzten und verantwortlicher Fachkraft dieser IPA. Ohne ihn hätte ich die Möglichkeit, eine anspruchsvolle und interessante IPA-Arbeit zu verfassen, nicht gehabt und ohne ihn wäre mein Abschluss nicht möglich gewesen. Weiter bedanke ich mich bei meinem Vater, Herrn Hajnik, für das Gegenlesen meiner Arbeit und bei Daniel Sager und Roger Sorg für ihren Einsatz für uns Lernende. Ich möchte mich auch bei meinem Hauptexperten Herrn Siegmund für die Zeit bedanken, die er aufgewendet hat, um meine Arbeit zu lesen, zu bewerten und mich im Rahmen der Expertengespräche zu unterstützen.

Herzlichen Dank!

Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
| ANGEBOTsoftware | Bisherige FAT-Client Software für die Verwaltung von Kunden und Angeboten der Generali. |
| OVWeb | Moderne cloudbasierte Webanwendung für die Verwaltung von Versicherungs-angeboten der Generali. (OffertenVerwaltungWeb) |
| shadcn | Shadcn-UI ist eine UI-Komponentenbibliothek, die auf Tailwind CSS basiert und eine Reihe von vordefinierten, anpassbaren und responsiven Komponenten bietet, um die Entwicklung von modernen Web-Oberflächen zu beschleunigen und zu vereinfachen. |

Tools

Das generative Tool ChatGPT wurde für verschiedene Sachen verwendet, wichtig anzumerken ist jedoch, dass der Input (Prompt) sauber und mit genügend eigenem Kontext gefüllt wurde, zudem wurden die Outputs überprüft, überarbeitet und projektbezogen ausformuliert. Es wurde Nichts blind übernommen oder kopiert.

Verwendung:

* Bugfixing
* Mockdaten Erstellung (Storybook und Unit Tests)
* Überarbeitung/Umformulierung von Textpassagen

Für die Use Cases und Sequenzdiagramme wurde PlantUML benutzt, da es den Arbeitsfluss vereinfachte.

Abbildungsverzeichnis

[Abbildung 1: Home-Page 27](#_Toc161737336)

[Abbildung 2: Kundenformular Ist Zustand 28](#_Toc161737337)

[Abbildung 3: OVWeb-Struktur 29](#_Toc161737338)

[Abbildung 4:Klassendiagramm Models 30](#_Toc161737339)

[Abbildung 5: Klassendiagramm Task mit Subklassen 30](#_Toc161737340)

[Abbildung 6: Klassendiagramm Controller 31](#_Toc161737341)

[Abbildung 7: Service Klassen 31](#_Toc161737342)

[Abbildung 8: Datenbankdiagramm 32](#_Toc161737343)

[Abbildung 9: Realisierungskonzept Diagramm 40](#_Toc161737344)

[Abbildung 10: Use Case Workflow Zwischenspeicherung 40](#_Toc161737345)

[Abbildung 11: Use Case Anzeige in Task Liste 41](#_Toc161737346)

[Abbildung 12: Use Case Formular Vorbefüllen 41](#_Toc161737347)

[Abbildung 13: Use Case Task Table Filter 42](#_Toc161737348)

[Abbildung 14: Use Case Task Table Sortierung 42](#_Toc161737349)

[Abbildung 15: Use Case E-Mail-Versenden 43](#_Toc161737350)

[Abbildung 16: Aktivitätsdiagramm Workflow Unterbruch 49](#_Toc161737351)

[Abbildung 17: Aktivitätsdiagramm Pendenzenliste 50](#_Toc161737352)

[Abbildung 18: Sequenzdiagramm Formular Vorbefüllen 51](#_Toc161737353)

[Abbildung 19: Sequenzdiagramm Workflow Zwischenspeicherung 51](#_Toc161737354)

[Abbildung 20: Sequenzdiagramm: E-Mail-Versenden 52](#_Toc161737355)

[Abbildung 21: ResumeCustomerWorkflow Klassendiagramm Soll-Zustand 52](#_Toc161737356)

[Abbildung 22: Kundenerstellung 53](#_Toc161737357)

[Abbildung 23: Error Message 54](#_Toc161737358)

[Abbildung 24: Fristablauf Variante 1 54](#_Toc161737359)

[Abbildung 25: Fristablauf Variante 2 55](#_Toc161737360)

[Abbildung 26: Projektumgebung 57](#_Toc161737361)

[Abbildung 27: Task-Table 66](#_Toc161737362)

[Abbildung 28: E-Mail Reminder 67](#_Toc161737363)

[Abbildung 29: Log Response Backend 68](#_Toc161737364)

[Abbildung 30: Log Response Frontend 69](#_Toc161737365)

[Abbildung 31: Grafana Dashboard 71](#_Toc161737366)

[Abbildung 32: Storybook CreateCustomerForm Empty State 73](#_Toc161737367)

[Abbildung 33: Fehlermeldung beim Speichern 77](#_Toc161737368)

[Abbildung 34: Task in Taskliste 78](#_Toc161737369)

[Abbildung 35: Taskliste Filter 79](#_Toc161737370)

[Abbildung 36: Task überfristet 79](#_Toc161737371)

Quellenverzeichnis

|  |
| --- |
| Lucidchart: Sequenzdiagramm <https://www.lucidchart.com/pages/de/uml-sequenzdiagramme> [Abrufdatum: 06.03.2024] Bemerkung: Richtige Notation für das Sequenzdiagramm in der Planung |
| Bexio: IPERKA <https://www.bexio.com/de-CH/blog/view/iperka-methode> [Abrufdatum: 06.03.2024] Bemerkung: Zur Verifizierung bei Kapitel 8: Projektbeschreibung |
| TanStack: GlobalFiltering <https://tanstack.com/table/v8/docs/guide/global-filtering> [Abrufdatum: 12.03.2024] Bemerkung: Für das filtern aller Columns im Table |
| StackOverflow: Next.js PageProps <https://stackoverflow.com/questions/66972049/how-to-set-pageprops-property-in-next-js-page> [Abrufdatum: 12.03.2024] |
| Spring: Spring Boot Scheduler <https://spring.io/guides/gs/scheduling-tasks> [Abrufdatum: 13.03.2024] Bemerkung: Für das Versenden von E-Mails |
| OpenTelemetry: Traces <https://opentelemetry.io/docs/concepts/signals/traces/> [Abrufdatum: 13.03.2024] Bemerkung: Für die Umsetzung von Distributed Tracing und konzeptionelles Verständnis von Tracing. |
| Jest: Testing <https://jestjs.io/docs/getting-started> [Abrufdatum: 14.03.2023] Bemerkung: Für das Testing von Frontend Components |
| Storybook: Writing Stories <https://storybook.js.org/docs/writing-stories> [Abrufdatum: 14.03.2023] Bemerkung: Zut Vervollständigung der Frontend Stories für Components |