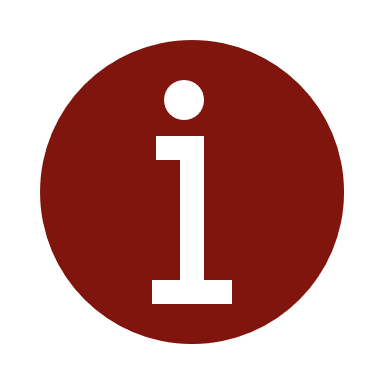
IPA 2024  
 Abschlussarbeit

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 4](#_Toc161321605)

[Teil 1 - Administration 5](#_Toc161321606)

[1. Projektorganisation 5](#_Toc161321607)

[2. Organisation der Arbeitsergebnisse 6](#_Toc161321608)

[3. Datensicherung 6](#_Toc161321609)

[3.1 Code 6](#_Toc161321610)

[3.2 Dokumentation 7](#_Toc161321611)

[4. Deklaration der benützten Firmenstandards 7](#_Toc161321612)

[5. Zeitplan 9](#_Toc161321613)

[6. Arbeitsjournal 10](#_Toc161321614)

[6.1 Dienstag, 05.03.2024 10](#_Toc161321615)

[6.2 Mittwoch, 06.03.2024 11](#_Toc161321616)

[6.3 Donnerstag, 07.03.2024 12](#_Toc161321617)

[6.4 Freitag, 08.03.2024 14](#_Toc161321618)

[6.5 Dienstag, 12.03.2024 15](#_Toc161321619)

[6.6 Mittwoch, 13.03.2024 17](#_Toc161321620)

[6.7 Donnerstag, 14.03.2024 18](#_Toc161321621)

[6.8 Freitag, 15.03.2024 18](#_Toc161321622)

[6.9 Dienstag, 19.03.2024 19](#_Toc161321623)

[6.10 Mittwoch, 20.03.2024 20](#_Toc161321624)

[Teil 2 – Projekt 21](#_Toc161321625)

[7. Kurzfassung 21](#_Toc161321626)

[7.1 Ausgangslage 21](#_Toc161321627)

[7.2 Umsetzung 21](#_Toc161321628)

[7.3 Ergebnis 21](#_Toc161321629)

[8. Projektbeschreibung 22](#_Toc161321630)

[8.1 IPERKA 22](#_Toc161321631)

[8.1.1 Informieren 22](#_Toc161321632)

[8.1.2 Planen 22](#_Toc161321633)

[8.1.3 Entscheiden 22](#_Toc161321634)

[8.1.4 Realisieren 23](#_Toc161321635)

[8.1.5 Kontrollieren 23](#_Toc161321636)

[8.1.6 Auswerten 23](#_Toc161321637)

[9. Einleitung 24](#_Toc161321638)

[10. Informieren 24](#_Toc161321639)

[10.1 Ziele der Aufgabenstellung 24](#_Toc161321640)

[10.2 Vorgaben 24](#_Toc161321641)

[10.3 Ist-Analyse 24](#_Toc161321642)

[10.3.1 Frontend 25](#_Toc161321643)

[10.3.2 Backend 26](#_Toc161321644)

[10.3.3 Überblick 27](#_Toc161321645)

[10.3.4 Klassendiagramme 28](#_Toc161321646)

[28](#_Toc161321647)

[10.3.5 Datenbankdiagramm 30](#_Toc161321648)

[10.4 User Stories 30](#_Toc161321649)

[10.4.1 Rollen 31](#_Toc161321650)

[10.4.2 Funktionale Anforderungen 31](#_Toc161321651)

[10.4.3 Nicht funktionale Anforderungen: 34](#_Toc161321652)

[11. Planen 37](#_Toc161321653)

[11.1 Realisierungskonzept 37](#_Toc161321654)

[11.1.1 Soll-Zustand 37](#_Toc161321655)

[11.1.2 Frontend 38](#_Toc161321656)

[11.1.3 Backend 39](#_Toc161321657)

[11.2 Testkonzept 39](#_Toc161321658)

[11.2.1 Testziele 39](#_Toc161321659)

[11.2.2 Testumgebung 40](#_Toc161321660)

[11.2.1 Testfälle 40](#_Toc161321661)

[11.3 Aktivitätsdiagramme 44](#_Toc161321662)

[11.4 Sequenzdiagramm 46](#_Toc161321663)

[11.5 Klassendiagramm 47](#_Toc161321664)

[11.6 Datenmodell 47](#_Toc161321665)

[11.7 Design 48](#_Toc161321666)

[11.7.1 Variante 1 49](#_Toc161321667)

[11.7.2 Variante 2 50](#_Toc161321668)

[12. Entscheiden 50](#_Toc161321669)

[12.1 Entscheid 50](#_Toc161321670)

[12.1.1 Entscheidungsmatrix 51](#_Toc161321671)

[13. Realisieren 52](#_Toc161321672)

[13.1 Projektumgebung 52](#_Toc161321673)

[13.2 Datenbank und Model 53](#_Toc161321674)

[13.3 Controller und Service 54](#_Toc161321675)

[13.4 Workflow Zwischenspeicherung 54](#_Toc161321676)

[13.5 Workflow Fortsetzung 56](#_Toc161321677)

[13.6 Pendenzenliste 57](#_Toc161321678)

[13.6.1 Sortierung 57](#_Toc161321679)

[13.6.2 Filter 59](#_Toc161321680)

[13.6.3 Anzeige 60](#_Toc161321681)

[13.7 E-Mail Service 61](#_Toc161321682)

[13.8 Logging 61](#_Toc161321683)

[13.9 Distributed Tracing 62](#_Toc161321684)

[13.10 Storybook 63](#_Toc161321685)

[13.11 Backend Unit Tests 63](#_Toc161321686)

[13.12 Frontend Unit Tests 63](#_Toc161321687)

[14. Kontrollieren 64](#_Toc161321688)

[14.1 Testkonzept 64](#_Toc161321689)

[14.2 Testprotokoll Samuel Hajnik 64](#_Toc161321690)

[15. Auswerten 65](#_Toc161321691)

[15.1 Vergleich Ist / Soll 65](#_Toc161321692)

[15.2 Fazit zur IPA 65](#_Toc161321693)

[15.3 Reflexion 65](#_Toc161321694)

[Danksagung 66](#_Toc161321695)

[Glossar 67](#_Toc161321696)

[Tools 67](#_Toc161321697)

[Abbildungsverzeichnis 68](#_Toc161321698)

[Quellenverzeichnis 69](#_Toc161321699)

Vorwort

Workflow = Task = Pendenz

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]

Hier ist der Verlauf der Commits auf den Branch:

* 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]

Hier ist der Verlauf der Commits auf die Repository:

1. Deklaration der benützten Firmenstandards

Im Entwickler-Team werden folgende 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. Zeitplan
2. Arbeitsjournal
   1. ****Dienstag, 05.03.2024****

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

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| Roman hat mir nach dem Expertengespräch noch Tipps gegeben zu meinem Zeitplan. |
| Ausserplanmässige Arbeiten |
| - |
| 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 vorauskomme. Ich konnte den Zeitplan in einer relative schnellen Zeit erledigen, was ich mir nicht dachte, da es doch eine anstrengende Aufgabe ist. Zugleich hat mir das Dokumentieren auch einfach gefallen Heute, ich konnte schon vieles runterschreiben, vor allem in der Phase Informieren, was mich nach vorne drängen wird in den nächsten Tagen. Beim Expertengespräch konnte ich sehr gute Einblicke holen, 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 Formeln zu benutzen. Somit konnte ich meinen Zeitplan verbessern.**  ***Fazit:*** Heute war ein erfolgreicher Tag. Ich konnte super 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 |
| **Erreichte Ziele** | * 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 das neue Object richtig aussieht. |
| Ausserplanmässige Arbeiten |
| * Mehr Zeit in Planung reingesteckt |
| 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 den Workflow 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 designet hätte. Mit den Mockups hat es ich auch keine Mühe, es war einfach es 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 |
| **Erreichte Ziele** | * Realisierungskonzept * Entscheiden * Planen |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| - |
| Ausserplanmässige Arbeiten |
| * Testkonzept erstellen |
| 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 die Pseudolösung schnell umsetzen, da ich mir schon vorher Überlegungen gemacht habe, wie ich die Aufgabe angehen und umsetzen will. Danach habe ich mich an die Entscheidungsphase gemacht, 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, das 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 einigermassen wie geplant und morgen fange ich an mit dem Code, da erhoffe ich mir auch so einen rassanten Fortschritt zu machen. |

* 1. Freitag, 08.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Zwischenspeicherung * Task in Pendenzenliste anzeigen * Formular Vorbefüllen |
| **Erreichte Ziele** | * Task in Pendenzenliste anzeigen * Formular Vorbefüllen Anfang |

|  |
| --- |
| **Probleme** |
| Bei der Zwischenspeicherung gab es ein Problem bei der OpenApi Generierung der Backenddaten. |
| Hilfestellung |
| Roman und ChatGPT |
| Ausserplanmässige Arbeiten |
| * Realisieren Dokumentation |
| 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, das es schnell gelöst wird. Ich konnte die verlorene Zeit nachholen und sie gut ausnutzen. |

* 1. Dienstag, 12.03.2024

|  |  |
| --- | --- |
| Tätigkeiten | * Formular Voerbefüllen * Task Liste Sortierung * Task Liste Filter * Realisieren Weiterdokumentieren |
| **Erreichte Ziele** | * Formular Voerbefüllen * Task Liste Sortierung * Task Liste Filter * Realisieren Weiterdokumentieren |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| Internet: Logging und Tracing |
| Ausserplanmässige Arbeiten |
| * Workflow Zwischenspeicherung |
| Reflexion |
| ***Was lief Heute positiv:***  **Heute konnte ich gut vorwärts kommen mit der Arbeit. Ich habe mehr Zeit invesiert, da ich letztes Mal ein Problem erhalten habe beim Zwischenspeichern. 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 Fehler gestossen, wie z.B. Wie soll die Customer Komponente die Task Id erkennen soll, bin aber ständig darangeblieben und konnte die Aufgabe endlich an die Ziellinie bringen.**  ***Fazit:*** Ich bin mir sicher, dass Heute nicht das letzte mal ist, das ich Überstunden machen werden muss. Heute war es weil ich im Zeitplan hintenrein war und aufholen wollte und das nächste mal wird es hoffentlich sein, weil ich die Soll-Zeit im Zeitplan überholen werden möchte. Vielleicht morgen, vielleicht übermorgen. 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 |
| **Erreichte Ziele** | * E-Mail Versenden * Logging * Unit Testing |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| - |
| Ausserplanmässige Arbeiten |
| * Anzeige in Task Liste |
| 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 die Applikation läuft oder nicht, gecheckt werden soll. Auf der offiziellen Spring Seite 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 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 |
| **Erreichte Ziele** | * Unit Tests |

|  |
| --- |
| **Probleme** |
| - |
| Hilfestellung |
| Roman und Internet bei Storybook |
| Ausserplanmässige Arbeiten |
| * Dokumentieren |
| 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 dan nans Dokumentieren gemacht, weil ich hier eine solide Struktur haben will für meine IPA. Am Ende vom Tag hatte ich das zweite Expertengespräch und ich konnte da gute Feedbacks einholen für mein Bericht, sowie ein Gefühl zu 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 |  |
| **Erreichte Ziele** |  |

|  |
| --- |
| **Probleme** |
|  |
| Hilfestellung |
|  |
| Ausserplanmässige Arbeiten |
|  |
| Reflexion |
| ***Was lief Heute positiv:***  ***Was lief nicht so gut:***  ***Fazit:*** |

* 1. Dienstag, 19.03.2024

|  |  |
| --- | --- |
| Tätigkeiten |  |
| **Erreichte Ziele** |  |

|  |
| --- |
| **Probleme** |
|  |
| Hilfestellung |
|  |
| Ausserplanmässige Arbeiten |
|  |
| Reflexion |
| ***Was lief Heute positiv:***  ***Was lief nicht so gut:***  ***Fazit:*** |

* 1. Mittwoch, 20.03.2024

|  |  |
| --- | --- |
| Tätigkeiten |  |
| **Erreichte Ziele** |  |

|  |
| --- |
| **Probleme** |
|  |
| Hilfestellung |
|  |
| Ausserplanmässige Arbeiten |
|  |
| Reflexion |
| ***Was lief Heute positiv:***  ***Was lief nicht so gut:***  ***Fazit:*** |

Teil 2 – Projekt

1. Kurzfassung
   1. Ausgangslage
   2. Umsetzung
   3. Ergebnis
2. 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 den Verstand 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 die Herangehensweise auch implementieren 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 implementiert wurde, werden die Resultate so getestet, wie es im Tesprotokoll 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 praktisch 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 Legacy Applikation 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 Backend Kommunikation 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

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte BeschreibungIn der folgenden Abbildung wird die Architektur der OVWEB-Applikation dargestellt.

Abbildung 3: OVWeb-Struktur

Jede Schnittstelle befindet sich in einem Docker Container.

* + 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. Ich habe jedoch auch nicht-funktionale Anforderungen als User Stories zu hinunterschreiben, um einen besseren Überblick zu behalten über die IPA relevanten Kriterien. Somit kann ich auch beim Testen gleich überprüfen, ob die Entwicklerthemen vollständig umgesetzt wurden, sowie am Ende den Ist- und Soll-Zustand vergleichen.

* + 1. Rollen

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

* + 1. Funktionale Anforderungen

|  |  |  |
| --- | --- | --- |
| Titel der User Story | Schätzung in Stunden | User Story Nummer |
| Frontend Button | 4 | 1 |
| Beschreibung |  |  |

Als ein **Benutzer** kann ich 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 auf den **ein 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:** Nach dem 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.

* + 1. 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 Ist-Zustand zum Soll-Zustand 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 jetzigen Applikationsstruktur.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 9: Realisierungskonzept Diagramm

* + 1. Frontend

**Neuer Button "Später Fortfahren" im Kunden-Erstellung Formular:** Ein neuer Button mit der Bezeichnung «Später Fortfahren» wird im Kunden-Erstellung 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-Erstellung Komponent 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.

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

**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.

**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 Task-DTO hinzugefügt und wird über die Application Settings veränderbar sein.

* + 1. Backend

**Versenden von Erinnerungs-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.

* 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.

* + 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. Dies widerspiegelt möglichst genau einer 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.  **Beschreibung:** Überprüfungdes E-Mail-Versendens, 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 für den abgerochenen Workflow  Der Link leitet den Benutzer auf den richtigen Workflow weiter. |

|  |
| --- |
| **ID: WU-7** |
| **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-8** |
| **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 unique Id, die in dazugehörigen Services angehängt ist. * Es kann ein Diagramm erstellt werden mit der Trace-Id und den dazugehörigen Services. |

|  |
| --- |
| **ID: WU-9** |
| **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. Einer für den Unterbruch und ein anderer für die Task-Liste. Es dient als Grundlage, um sicherzustellen, dass alle Projektbeteiligten vom Gleichen reden.

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

Automatisch generierte Beschreibung

Abbildung 10: Aktivitätsdiagramm Workflow Unterbruch

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

Automatisch generierte Beschreibung

Abbildung 11: Aktivitätsdiagramm Pendenzenliste

* 1. Sequenzdiagramm

Das 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.

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

Automatisch generierte Beschreibung

Abbildung 12: Sequenzdiagramm

* 1. Klassendiagramm

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.

Ein Bild, das Schwarz, Dunkelheit enthält.

Automatisch generierte Beschreibung

Abbildung 13: 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 14: Kundenerstellung

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

Automatisch generierte Beschreibung

Abbildung 15: Error Message

* + 1. Variante 1

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 16: Fristablauf Variante 1

* + 1. Variante 2

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

Automatisch generierte Beschreibung

Abbildung 17: 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 11.7.1 respektiv 11.7.2 zu finden.

Die Versionen sind im Grunde gleich, der einzige Unterschied, der besteht, ist das in der zweiten Version noch ein Popup 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 18: Projektumgebung

Hier ist die Projektumgebung aufgelistet, für die Aufgabe mussten nur Server Actions im «action» Ordner erstellt werden, ansonsten werden nur existierende Dateien, die mit den Use Cases zusammenhängen, erweitert.

* 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 Columns, wie die Customer Tabelle, da ein Teil des Customers-Objekts dort gespeichert werden soll. Hierfür wurde die 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 State: «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 Response als JSON zurückkommt und es schon zwei andere Task-Types 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 Property des JSON-Responses korrekt mappen und somit sicherstellen, dass die Datensätze in das richtige Entity reinkommen und ebenfalls richtig gefetched 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 Abspeicherung 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 Error gibt, wird dieser gehandlet 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 Dto’s musste nur das Frontend angepasst werden für die Zwischenspeicherung. Im **«CustomerCreateForm.tsx»** wurde ein Button 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 Type verlangt, musste dieser mitgegeben werden, hardcoded als **«type: ‘ResumeCustomerWorkflowPostDto’»**, da nur ein Workflow benutzt wird.

Um die Bedienung zu vereinfachen, bekommt der Benutzer bei jeder Anfrage einen Toast, also eine Message zurück, ob die Anfrage entweder durchgekommen ist 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')  } catch (error) {  toast({  title: 'Der Task konnte aufgrund eines Serverfehlers nicht erstellt werden.',  variant: 'error'  })  } } |

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

* 1. Workflow Fortsetzung

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

Die Daten werden mit dem form.reset Hook in ihre respektiven Felder hineingefügt, falls es einen Eintrag dafür gibt in der Datenbank. Da die Namensgebung des Form Inputs 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 State verändert oder getracked wird oder wenn sich die resumeTaskId-Prop ändert. 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 prefilled falls der Benutzer auf den Resume Button klickt in der Pendenzenliste, da nicht bei einem gewöhnlichen Aufruf der Seite ein halbfertiger Task erscheinen soll.

* 1. Pendenzenliste

An der Pendenzenliste mussten mehrere neue Funktionalitäten vorgenommen werden. Die neuen Implementierungen wurden in die folgenden drei Unternehmen 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 (z. B. Klicken auf einen Header zum Sortieren), die Funktion **«setSorting»** mit dem neuen Sortierstatus aufgerufen wird.

In den Spaltendefinitionen (columns.tsx) wird für jede Spalte eine Button-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 Column und Filter Value 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. Es besteht kein Global Filter.

Zuletzt wird die Tabelle mit der **«getFilteredRowModel»** Funktion rendered. Sie ist dafür zuständig die Rows zu filtern basiert auf den jeweiligen Filtering State.

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" /> |

* + 1. Anzeige

Falls ein Task überfällig ist oder über die Frist hinausgeganen 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.needsReminder  ? 'border-red-500'  : ''  } > |

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 Border des Rows, auf rot gesetzt, um den Task von den anderen zu unterscheiden.

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.isAfter(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» zu umwandeln, damit diese im erforderten Format des API’s sind.

Die «getAllTasks» Funktion des «TaskController.java» fetched 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.

* 1. E-Mail Service

Scheduler @EnableSchedule

* 1. Logging

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

TaskService.java

|  |
| --- |
| 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:

Update-task.ts

|  |
| --- |
| 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:**

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 19: Grafana Dashboard

* 1. Storybook
  2. Backend Unit Tests
  3. Frontend Unit Tests

1. Kontrollieren
   1. Testkonzept
   2. Testprotokoll Samuel Hajnik
2. Auswerten
   1. Vergleich Ist / Soll

Checkliste User Stories

* 1. Fazit zur IPA
  2. Reflexion

Danksagung

Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
|  |  |

Tools

Abbildungsverzeichnis

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 |
| Shadcn: Table Sorting & Filtering <https://ui.shadcn.com/docs/components/data-table#sorting> [Abrufdatum: 12.03.2024] |
| 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 |