



IPA

Eingehende Message-Queue-Nachrichten im Web-UI

IPA von Joel Vontobel

Ergon Informatik AG
7. November 2024

Inhalt

I	Umfeld und Ablauf	3
1	Aufgabenstellung	4
1.1	Ausgangslage	4
1.2	Detaillierte Aufgabenstellung	4
1.3	Mittel und Methoden	5
1.4	Vorkenntnisse	5
1.5	Vorarbeiten	5
1.6	Neue Lerninhalte	6
1.7	Arbeiten in den letzten 6 Monaten	6
2	Projektaufbauorganisation	7
3	Benützte Firmenstandards	8
4	Arbeitsumgebung	9
4.1	Arbeitsplatz	9
4.1.1	Office Arbeitsplatz	9
4.2	Verwendete Tools	10
5	Versionierung und Sicherung der Arbeitsergebnisse	11
5.1	Verwendung von Git zu Versionierung	11
5.2	Quellcode	11
5.3	Probe-IPA Dokumentation	12
6	Projektmanagementmethode	14
6.1	IPERKA	14
6.2	Alternative Methode	15
7	Arbeitsprotokoll	16
II	Projekt	27
8	Kurzfassung	28
9	Informieren	29
9.1	Projektumfeld	29
9.1.1	Einsatzzweck	29
9.1.2	Funktion	29

9.2	Anforderungen	31
9.2.1	Minimalanforderungen	31
9.2.2	Erweiterung: Pagination	32
9.2.3	Erweiterung: Filter	33
10	Planen	35
10.1	Arbeitspakete	35
10.1.1	Informieren	36
10.1.2	Planen	36
10.1.3	Entscheiden	37
10.1.4	Realisieren	38
10.1.5	Kontrollieren	39
10.1.6	Auswerten	39
10.1.7	Rahmenaufgaben	40
10.2	Lösungskonzept für die Struktur vom Backend	42
10.2.1	Erstellung der Endpunkte	42
11	Entscheiden	43
12	Realisieren	44
13	Kontrollieren	45
14	Auswerten	46
	Glossar	47
	Abbildungsverzeichnis	47
	Quellenverzeichnis	48

Teil I

Umfeld und Ablauf

1 Aufgabenstellung

In diesem Kapitel ist die Aufgabenstellung der Probe-IPA aufgeführt. Die Inhalte wurden zu einem grossen Teil von der originalen Aufgabenstellung übernommen und Angepasst.

1.1 Ausgangslage

Die Firma Ergon Informatik AG entwickelt sein einigen Jahren eine Transaktions-Autorisierungs-Lösung für einige Banken in der Schweiz. Dieses Projekt heisst CardX und wird durch ein 12 zwölfköpfiges Team umgesetzt. In diesem Projekt ist der Lernende seit Januar 2024 tätig und kennt sich deshalb schon ein bisschen aus.

Das Projekt kommuniziert mit verschiedenen bankspezifischen IT-Systemen wie zum Beispiel dem Kernbankensystem oder Service-Büros über Message-Queues. Diese Message-Queues werden in der Datenbank-Tabelle MQ_TABLE (eingehende Nachrichten) und MQ_OUT (ausgehende Nachrichten) zwischengespeichert. Falls man diese Message-Queues anschauen oder bearbeiten möchte, muss man dies in der Datenbank machen. Weil das ziemlich umständlich ist, besteht die Aufgabe des Lernenden jetzt daraus diese Message-Queues in einem Web-GUI darzustellen und sinnvolle Interaktionen mit diesen Daten anzubieten.

1.2 Detaillierte Aufgabenstellung

Minimalanforderungen Das Ziel dieser Aufgabe ist es, den Inhalt der Tabelle MQ_TABLE in diesem Web-GUI sichtbar zu machen und dem Nutzer sinnvolle Interaktionen mit diesen Daten anzubieten.

Es soll im Web-GUI eine neue Seite erstellt werden mit dem Inhalt einer Tabelle, welche die Message-Queues abbilden soll. Die Tabelle soll eine Hand voll Spalten besitzen, sodass sie übersichtlicher ist. Die Seite soll stimmig in das UI eingebaut werden. Im Backend sollen die neuen Methoden mithilfe von Unit-Tests abgedeckt werden.

Zusätzlich kann der Lernende noch zwischen zwei Erweiterungen entscheiden, welche er implementieren möchte.

Erweiterung: Pagination Bei der ersten Erweiterungen ist das Ziel ein Paginator zur Tabelle hinzuzufügen. Eine Pagination ist wenn man eine Liste oder Tabelle auf ein paar Einträge limitiert und anschliessend weitere Einträge anzeigen kann mit einer Pfeiltaste. Dies hilft, das Laden der Seite zu verkürzen, da die Einträge, die nicht angezeigt werden, erst geladen werden, wenn sie auch wirklich gebraucht werden.

Erweiterung: Filter Die zweite Erweiterung ist ein Filtersystem. Die Seite soll nach dem Status, Inhalt der Nachricht und dem Datum gefiltert werden können. Die Filter-Werte sollen ausserdem in der URL Abgebildet werden, um das Teilen von gefilterten Ergebnissen zu vereinfachen oder den Filter als Lesezeichen ablegen zu können.

1.3 Mittel und Methoden

Technologien

- SQL
- Java
- TypeScript
- HTML
- Angular

Tools

- IntelliJ (IDE)
- Docker
- Bitbucket
- Confluence
- Jira
- Postman

1.4 Vorkenntnisse

Der Lernende hat bereits viele Arbeiten im Projekt CardX gemacht. Unter anderem im Backend und an CardX-spezifischen Tools. Die Codebasis hat der Lernende in den letzten 9 Monaten gut kennengelernt und findet sich gut zurecht. Der Lernende hat auch bereits die Tabelle TASK von der Datenbank in das Web-GUI gebracht, was eine ähnliche Aufgabe war wie die jetzige Aufgabenstellung.

Durch die früheren Projekte, wie ein Fussballtippspiel und eine Anmelde-Plattform für Bewerbende, konnte er bereits viel Erfahrung mit Java und Angular sammeln.

1.5 Vorarbeiten

Durch das bereits existierende Projekt und die vielen Arbeiten, die der Lernende bereits gemacht hat, musste er keine Vorarbeiten leisten.

1.6 Neue Lerninhalte

- Pagination:

Mit Pagination hat der Lernende sich noch nie auseinandergesetzt. Er hat es schon oft auf anderen Seiten gesehen aber noch nie selbst implementiert.

- Filtersystem:

Im Projekt WM-Tippspiel gab es ein Filtersystem, aber dieses hat der Lernende nicht selbst implementiert und ist so ein neuer Lerninhalt.

1.7 Arbeiten in den letzten 6 Monaten

In den letzten 6 Monaten hat der Lernende sich, wie oben schon genannt, mit dem Projekt CardX auseinandergesetzt und ist ein aktives Team Mitglied. Er hat viele verschiedene Arbeiten umgesetzt. Einige davon hier:

- CheckDB Task:

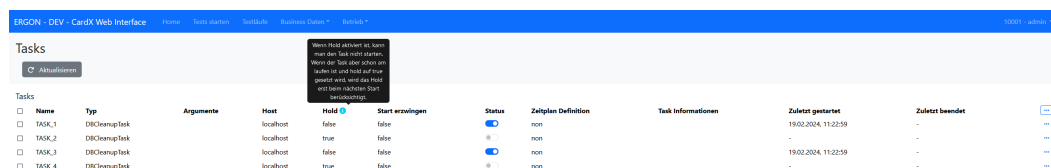
In dieser Aufgabe geht es darum, eine Aufgabe zu erstellen, welche periodisch oder manuell ausgeführt werden kann. Diese Aufgabe überprüft die Datenbankdefinition, ob immer noch alles fehlerfrei ist. Es werden Sequenzen, Indexe, Felder und mehr in eine Datei geschrieben, gespeichert und anschliessend mit der vorherigen Datei auf Veränderungen verglichen. Bei einer Veränderung schlägt die Aufgabe fehl und die Differenz wird in der Fehlermeldung angezeigt.

- ServiceLogMove und Zip:

Mit der Aufgabe ServiceLogMerge werden Protokolle des Systems in Dateien geschrieben und gespeichert. Ein Teil dieser Dateien wird jetzt mithilfe der Aufgabe in bestimmte Verzeichnisse verschoben und komprimiert. Die Dateien werden nach Bank sortiert und anschliessend in ihn vorgesehenes Verzeichnis verschoben. Auch diese Aufgabe wird periodisch jeden Tag ausgeführt, um die Dateiablage übersichtlich zu halten.

- Tasks im Frontend anzeigen und bearbeiten:

Diese Aufgabe zeigt alle Tasks im Frontend an und man kann sie dort auch bearbeiten. Das Ziel von dieser Aufgabe war gleich wie die Mindestanforderungen von der Aufgabenstellung.



Name	Type	Argumente	Host	Held	Start erzeugen	Status	Zeitplan Definition	Task Informationen	Zuletzt gestartet	Zuletzt beendet
TASK_1	DBCleanupTask		localhost	false	false	on	non		19.02.2024, 11:22:59	-
TASK_2	DBCleanupTask		localhost	true	false	on	non		-	-
TASK_3	DBCleanupTask		localhost	false	false	on	non		19.02.2024, 11:22:59	-
TASK_4	DBCleanupTask		localhost	true	false	on	non		-	-

Abbildung 1.1: Anzeigen und bearbeiten der Tasks

2 Projektaufbauorganisation

In der folgenden Tabelle sind die an der Probe-IPA beteiligten Personen und ihre jeweiligen Aufgaben aufgeführt.

Person	Rolle	Aufgabe/Verantwortung
Joel Vontobel	Kandidat (K)	Umsetzen der Facharbeit
Loris Diana und Dominic Monzón	Verantwortliche Fachkraft (VF)	Facharbeit begleiten, technische Fragen beantworten, Bewertung der Facharbeit
Bernd Lienberger	Hauptexperte (HEX)	IPA bezogene Fragen beantworten, Entscheiden bei auftretenden Problemen, Besuchstermine festlegen, Fachgespräch leiten, Bewertung der Facharbeit
	Nebenexperte (NEX)	Notizen erstellen zu Präsentation und zum Fachgespräch, Bewertung der Facharbeit

3 Benützte Firmenstandards

Es werden keine spezifischen Firmenstandards verwendet.

4 Arbeitsumgebung

In diesem Kapitel ist beschrieben, wie die Arbeitsumgebung des Lernenden während der Probe-IPA aussah.

4.1 Arbeitsplatz

4.1.1 Office Arbeitsplatz

Die Probe-IPA wird am gewohnten Arbeitsplatz im Fünferbüro des Lernenden durchgeführt. Als Arbeitsgerät wird ein Notebook verwendet, welches mithilfe einer Dockingstation das Gerät mit zwei Monitoren und dem Firmennetzwerk verbindet. Der Stuhl und Tisch sind höhenverstellbar, und der Lernende kann dadurch in verschiedenen Sitzpositionen oder stehend arbeiten.

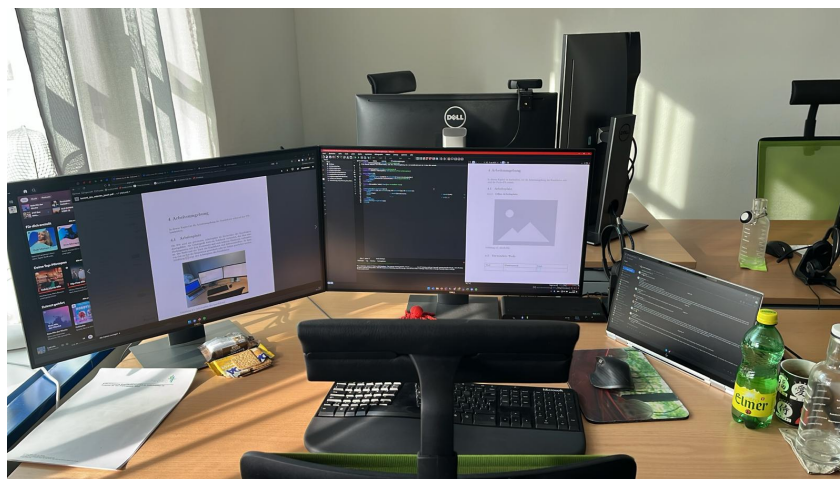


Abbildung 4.1: Arbeitsplatz des Lernenden

4.2 Verwendete Tools

Die folgende Tabelle zeigt auf, welche Tools für die Umsetzung der Probe-IPA eingesetzt wurden.

Tool	Einsatzzweck	Link
IntelliJ	Entwicklungsumgebung für die Programmierung	https://www.jetbrains.com/de-de/idea/
Docker	Ausführen der Programme	https://www.docker.com/
Git	Versionierung vom Quellcode	https://git-scm.com/
Postman	Ausführen von HTTP-Requests (Testen vom Bankend)	https://www.postman.com/
Bitbucket	Speicherung der Quellcodes	https://bitbucket.org/product/
Jenkins	Tool für Pipelines um Tests, Codequalität automatisch zu prüfen	https://www.jenkins.io/
Confluence	Probe-IPA Kriterien	https://www.atlassian.com/de/software/confluence
Jira	Aufgabenstellung	https://www.atlassian.com/de/software/jira
Draw.io	Erstellen von Diagrammen und Abbildungen	https://www.drawio.com/
TexStudio	Dokumentationstool	https://www.texstudio.org/
LaTeX	Ein Dokumentenvorbereitungssystem	https://www.latex-project.org/
Mattermost	Text basiertes Kommunikationsmittel	https://mattermost.com/
Microsoft Teams	Video basiertes Kommunikationsmittel	https://www.microsoft.com/de-ch/microsoft-teams/group-chat-software
Microsoft Excel	Erstellung und Bearbeitung des Zeitplans	https://www.microsoft.com/de-ch/microsoft-365/excel?market=ch

5 Versionierung und Sicherung der Arbeitsergebnisse

In diesem Kapitel wird beschrieben, wie der Lernende sicherstellt, dass die erarbeiteten Ergebnisse während der Probe-IPA sicher gespeichert und jederzeit wieder aufrufbar sind. Die Versionierung soll es ermöglichen, frühere erstellte Versionen der Daten jederzeit wiederherstellen zu können. Die Massnahmen werden hier vom Lernenden aufgeführt.

5.1 Verwendung von Git zu Versionierung

Für die Versionierung von der Probe-IPA wird Git verwendet. Git ist ein Versionierungstool und wird genutzt, um Quellcode zu versionieren und zu beschriften. In der Schule so wie auch in der Firma wurde Git bereits in diversen Projekten verwendet, um den Quellcode übersichtlich zu versionieren und in der Cloud zu sichern. Mit Git kann man sogenannte «Commits» machen, um einen kleinen Teil der Änderungen zu speichern und zu beschriften. Diese Änderungen kann man jederzeit wieder rückgängig machen oder aufrufen, um eine bestimmte Version genauer zu analysieren. Durch diese Commits ist der Quellcode für eine andere Person verständlicher zu lesen.

5.2 Quellcode

Der Quellcode der eingehenden Message-Queue-Nachrichten im Web-GUI wird mit Git verwaltet und ist in einem Repository auf Bitbucket gespeichert. In Abbildung 5.1 ist die Git Commit History des Quellcodes ersichtlich.



Abbildung 5.1: Git Commit History des Quellcodes

5.3 Probe-IPA Dokumentation

Die Probe-IPA-Dokumentation wird mithilfe von \LaTeX geschrieben, wodurch eine Versionierung mit Git auch möglich wird. Die \LaTeX - und alle anderen benötigten Dateien werden auf ein privates Repository in der Cloud geladen. In Abbildung 5.2 ist die Git Commit History der Probe-IPA-Dokumentation ersichtlich.



Abbildung 5.2: Git Commit History der Dokumentation

6 Projektmanagementmethode

In diesem Kapitel ist die Projektmanagement-Methode «IPERKA» beschrieben, die während der Probe-IPA verwendet wird. Es werden die Gründe für diese Projektmanagement-Methode aufgeführt und eine alternative Methode mit Gründen, warum sie nicht verwendet wurde.

6.1 IPERKA

Als Projektmanagement-Methode während der Probe-IPA wird IPERKA verwendet. IPERKA ist eine Vorgehensmethode, die sich gut für Projekte mit überschaubarem Umfang und klar definierte Ziele eignet. Die Methode wurde bereits im ersten Lehrjahr in der Schule behandelt und ist durch das bereits bekannt. Der Name «IPERKA» setzt sich aus den Anfangsbuchstaben der sechs verschiedenen Schritten zusammen, nach denen vorgegangen wird:

I nformieren: Den Auftrag verstehen, eine Vorstellung der Lösung erhalten, fehlende Informationen einholen, ordnen und bewerten

P lanen: Nötige Arbeitsschritte definieren, einen Zeitplan erstellen, Methoden und Arbeitsmittel definieren

E ntcheiden: Verschiedene Lösungsvarianten vergleichen, ausschlaggebende Kriterien definieren, eine Lösungsvariante auswählen

R ealisieren: Arbeit umsetzen, Arbeitsschritte und Ergebnisse dokumentieren, auftretende Probleme behandeln

K ontrollieren: Arbeit testen, Resultate mit den Anforderungen vergleichen, Soll-Ist-Vergleich des Zeitplans, Dokumentation nochmals durchlesen

A uswerten: Rückblick auf das Vorgehen, Arbeitsschritte beurteilen, Selbsteinschätzung vornehmen, mögliche Optimierungen für weitere Projekte definieren

Die Aufteilung der Arbeit in die genannten Schritte unterstützt dabei, die Aufgaben sinnvoll zu strukturieren und systematisch vorzugehen. Ausserdem hilft die bewusste Steuerung des Arbeitsprozesses, die persönlichen Kompetenzen weiterzuentwickeln (vgl. ICT Berufsbildung Bern 2024(Bern 2024)). Die klare Trennung der Schritte stellt sicher, dass der Umfang und die erwarteten Ergebnisse der Aufgabe sich im Verlauf der Arbeit nicht mehr stark ändern, da IPERKA grundsätzlich kein «Rückwärtsgehen» in den Phasen, wie dies aus iterativen Modellen bekannt ist, vorsieht. Ausserdem stellt sie sicher, dass die Realisierung nicht zu schnell in Angriff genommen wird.

6.2 Alternative Methode

Neben IPERKA wurde noch eine andere Vorgehensmethode angeschaut. Diese ist folgend, jeweils mit einer Begründung, wieso IPERKA der Methode vorgezogen wurde, kurz beschrieben.

Scrum ist eine bekannte agile Vorgehensmethode, die heutzutage in der Softwareentwicklung weit verbreitet ist. Die Methode legt den Fokus mehr auf Punkte wie laufende Software, gute Zusammenarbeit, oder flexibles Reagieren auf Veränderungen, anstatt einem strikten Plan zu folgen. Scrum sieht einen iterativen Prozess vor, der laufend optimiert werden soll, und definiert verschiedene Rollen, die jeweils ihre Aufgabe in diesem Scrum-Prozess haben. Ein «agiles» Vorgehen ist in der Softwareentwicklung grundsätzlich sinnvoll, da oft nicht von Beginn her klar ist, wie das Resultat schlussendlich aussehen soll. Da die Probe-IPA schlussendlich aber eine Prüfung ist, sind die Vorgaben und erwarteten Resultate relativ klar. Auch der Umfang der Probe-IPA und das Enddatum sind von Beginn an bekannt und verändern sich nicht während der Arbeit. Ausserdem wird die Aufgabe von nur einer Person bearbeitet, wofür die Abläufe von Scrum nicht optimal geeignet sind. Aus diesen Gründen wird IPERKA gegenüber Scrum vom Lernenden bevorzugt.

7 Arbeitsprotokoll

Datum	06.11.2024
Bearbeitete Arbeitspakete	7.1, 7.2, 7.3, 7.4
Arbeitszeit	8h
Überzeit	0h
Vergleich mit dem Zeitplan	Zeitplan noch nicht fertig
Erfolge und Probleme	Durch die Vorlage des Dokuments und \LaTeX konnte ich direkt mit dem ersten Teil der Dokumentation beginnen, was mir viel Zeit erspart hat. Ich habe heute mein Ziel, den ersten Teil grösstenteils abzuschliessen, erreicht und hatte keine grösseren Probleme die aufgetreten sind.
Tagesreflexion	Ich bin heute gut in die Probe-IPA gestartet. Anfangs war ich ein wenig übervordert und wusste nicht wo ich anfangen sollte. Nach der ersten Stunde hat sich das aber wieder gelegt und ich konnte konzentriert an meinem Ziel arbeiten.
In Anspruch genommene Hilfe	Keine

Datum	07.11.2024
Bearbeitete Arbeitspakete	1.1, 1.2, 2.1, 2.2
Arbeitszeit	8h
Überzeit	0h
Vergleich mit dem Zeitplan	Ich konnte alle geplanten Arbeiten von heute Erledigen und hatte auch eine Stunde übrig. Ich habe also bereits mit der Aufgabe 2.3 angefangen.
Erfolge und Probleme	Heute konnte ich mit dem 2. Teil der Dokumentation anfangen. Die Phase Informieren konnte ich Zeitgerecht abschliessen und habe bereits in der Phase Planen die Arbeitspakete und der Zeitplan fertig gestellt. Momentan habe ich mit der Aufgabe 10.1.2 begonnen die für Morgen eingeplant ist.
Tagesreflexion	Ich konnte heute motiviert in den Tag starten, da ich gestern meine Tagesziele erreicht habe. Ich konnte mich nicht wirklich für 10.1.1 und 10.1.1 motivieren, wusste aber, dass ich mich danach mit den Arbeitspaketen und dem Zeitplan beschäftigen konnte. Diese zwei Teile haben mir Spass gemacht, da ich mich danach an dem Zeitplan orientieren kann.
In Anspruch genommene Hilfe	Ich habe mich bei Loris Diana 2 erkundigt, ob ich die Codequalität mit dem Tool, welches mein Projekt nutzt, überprüfen darf, oder ich selbst diese Überprüfung machen muss.

Datum	...
Bearbeitete Arbeitspakete	...
Arbeitszeit	...
Überzeit	...
Vergleich mit dem Zeitplan	...
Erfolge und Probleme	...
Tagesreflexion	...
In Anspruch genommene Hilfe	...

Datum	...
Bearbeitete Arbeitspakete	...
Arbeitszeit	...
Überzeit	...
Vergleich mit dem Zeitplan	...
Erfolge und Probleme	...
Tagesreflexion	...
In Anspruch genommene Hilfe	...

Datum	...
Bearbeitete Arbeitspakete	...
Arbeitszeit	...
Überzeit	...
Vergleich mit dem Zeitplan	...
Erfolge und Probleme	...
Tagesreflexion	...
In Anspruch genommene Hilfe	...

Datum	...
Bearbeitete Arbeitspakete	...
Arbeitszeit	...
Überzeit	...
Vergleich mit dem Zeitplan	...
Erfolge und Probleme	...
Tagesreflexion	...
In Anspruch genommene Hilfe	...

Datum	...
Bearbeitete Arbeitspakete	...
Arbeitszeit	...
Überzeit	...
Vergleich mit dem Zeitplan	...
Erfolge und Probleme	...
Tagesreflexion	...
In Anspruch genommene Hilfe	...

Datum	...
Bearbeitete Arbeitspakete	...
Arbeitszeit	...
Überzeit	...
Vergleich mit dem Zeitplan	...
Erfolge und Probleme	...
Tagesreflexion	...
In Anspruch genommene Hilfe	...

Datum	...
Bearbeitete Arbeitspakete	...
Arbeitszeit	...
Überzeit	...
Vergleich mit dem Zeitplan	...
Erfolge und Probleme	...
Tagesreflexion	...
In Anspruch genommene Hilfe	...

Datum	...
Bearbeitete Arbeitspakete	...
Arbeitszeit	...
Überzeit	...
Vergleich mit dem Zeitplan	...
Erfolge und Probleme	...
Tagesreflexion	...
In Anspruch genommene Hilfe	...

Datum	...
Bearbeitete Arbeitspakete	...
Arbeitszeit	...
Überzeit	...
Vergleich mit dem Zeitplan	...
Erfolge und Probleme	...
Tagesreflexion	...
In Anspruch genommene Hilfe	...

Teil II

Projekt

8 Kurzfassung

Ausgangssituation CardX ist eine Transaktions-Autorisierungs-Lösung, die bei einigen Banken im Einsatz ist. CardX kommuniziert mit diversen anderen bankspezifischen IT-Systemen, zum Beispiel dem Kernbankensystem oder Service-Büros über Message-Queues. Diese Queues werden asynchron durch CardX befüllt und dabei in den Datenbank-Tabellen MQ_TABLE (eingehende Nachrichten) und MQ_OUT (ausgehende Nachrichten) zwischengespeichert. Diese Tabellen können momentan nur direkt auf der Datenbank eingesehen werden.

Umsetzung Im Frontend wird auf die Seite Business Daten navigiert. Nachdem die Seite erreicht wurde, wird im Hintergrund ein GET-Request an das Backend gesendet. Im Backend sorgt dieser Request dafür, dass die Message-Queues aus der Datenbank hervorgeholt werden mithilfe von Hibernate. Die Daten werden bereits durch die SQL-Query sortiert und gefiltert. Anschliessend werden die Daten ins Frontend geschickt und dort mit einer Tabelle angezeigt. Um zu garantieren, dass der Code fehlerfrei läuft, werden noch Tests im Backend geschrieben.

Ergebnis Die neue Seite Business Daten ist korrekt umgesetzt und beinhaltet keine Fehler. Bei jedem Push wird der Code getestet und auf Styling geschaut.

9 Informieren

Dieses Kapitel zeigt die in der IPERKA-Phase «Informieren» durchgeführten Arbeiten auf. In dieser Phase wird der Einsatzzweck und die Funktionsweise genauer analysiert. Basierend darauf werden Anforderungen an das Projekt definiert.

9.1 Projektumfeld

In diesem Abschnitt wird das Projektumfeld genauer analysiert und unter anderem grafisch dargestellt. Es gilt den groben Ablauf der Aufgabenstellung zu kennen um die Implementation zu vereinfachen.

9.1.1 Einsatzzweck

Momentan sind die Message-Queues nur in der Datenbank ersichtlich. Für den alltäglichen Gebrauch ist das Zugreifen auf die Datenbank umständlich und können Probleme auftreten. Da wenig bis gar keine Sicherheitsmassnahmen bei der Bearbeitung von Elementen in der Datenbank existieren, kann das Unterbrüche und andere Probleme verursachen. Diese Probe-IPA soll das Lesen und Bearbeiten dieser Message-Queues vereinfachen.

9.1.2 Funktion

Die neue Seite Business Daten soll eine Tabelle beinhalten mit den folgenden Spalten.

- MODIFIED_AT: Die letzte Änderung an der Message-QUEUE
- MQ_QUEUE_ID: Message-Queue-ID
- JOB_KEY: Der dazugehörige JOB
- MESSAGE_SHORT / _LONG: Die Nachricht
- MQ_IN_STATUS: Der Verarbeitungs-Status
- MQ_IN_STATUS_STRING: Das Verarbeitungs-Ergebnis
- MQ_IN_STATUS_STRING: Die Anzahl an Verarbeitungsversuche

Diese Daten werden dann durch ein Get-Request vom Backend geholt. Das Ziel ist, dass die Daten im Frontend nur noch angezeigt werden müssen für die Mindestanforderungen. Im Backend wird durch eine SQL-Query dafür gesorgt, dass nur die oben genannten Spalten aus der Datenbank hervorgeholt werden. In der Abbildung 9.1 ist der Ablauf eines solchen GET-Requests ersichtlich.

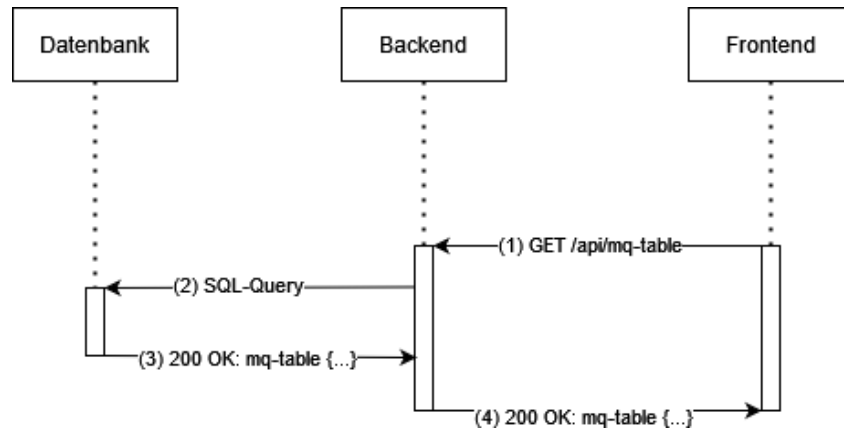


Abbildung 9.1: Ablauf eines GET-Requests für das Erstellen der Tabelle

Folgende Schritte werden im Ablaufdiagramm durchgeführt.

1. Das Frontend sendet einen GET-Request für alle Message-Queues um sie anschliessend in der Tabelle darzustellen.
2. Das Backend empfängt diesen Request und führt eine SQL-Query aus um die entsprechenden Message-Queues zu erhalten. Die SQL-Query filtert bereits alle Spalten heraus die nicht genutzt werden.
3. Die Datenbank schickt eine gefilterte Liste mit Message-Queues zurück an das Backend.
4. Das Backend mappt die entsprechenden Daten in Objekte die das Frontend kennt und schickt es an sie.

Danach kann das Frontend die Daten anzeigen.

9.2 Anforderungen

Auf Basis der unter aufgeführten detaillierten Aufgabestellung und den individuellen Beurteilungskriterien werden die folgenden Anforderungen definiert. Die Anforderungen sind den folgenden drei Teilbereichen entsprechend gruppiert und bezeichnet.

MA... Minimalanforderungen

EP... Erweiterung Pagination

EF... Erweiterung Filter

9.2.1 Minimalanforderungen

Folgende funktionale und nicht funktionale Anforderungen sind hier für die Minimalanforderungen aufgelistet.

Funktionale Anforderungen

Anforderung	Beschreibung
MA1	Die neue Seite ist im Web-GUI auffindbar über den Pfad /admin/mq-table-search und übers Menü unter "Business Daten" → "Queue-Messages".
MA2	Die Tabelle beinhaltet maximal 25 Einträge.
MA3	Alle Einträge sind im Fehlerzustand, d.h. MQ_IN_STATUS == 3.
MA4	Die tabellarische Darstellung von MQ_TABLE beinhaltet folgende Spalten: MODIFIED_AT, MQ_QUEUE_ID, JOB_KEY, MESSAGE_SHORT / _LONG, MQ_IN_STATUS, MQ_IN_STATUS_STRING und MQ_IN_STATUS_STRING.
MA5	Anhand von MESSAGE_IS_SHORT wird entschieden ob MESSAGE_SHORT oder MESSAGE_LONG abgefüllt werden soll.
MA6	Im Web-GUI sollen in der tabellarischen Ansicht nur die ersten 50 Zeichen von MESSAGE_SHORT / _LONG angezeigt werden.
MA7	Der MQ_IN_STATUS wird in textuelle Stati, gemäss ch.ergon.cardx.shared.database.enumeration.MqInStatus, übersetzt.
MA8	Das Kontextmenü beinhaltet eine Aktion für die Anzeige des kompletten Inhalts der Spalte "Nachricht".
MA9	Das Kontextmenü beinhaltet eine Aktion, um einen erneuten Verarbeitungsversuch eines Eintrages anzustossen. D.h. durch das Setzen des MQ_IN_STATUS auf 0.
MA10	Die neuen public-Methoden im Admin-Backend sind mit Unit-Tests bzw. SpringBoot-Tests abgedeckt. Eintrag 4
MA11	Es gibt einen Mock-Datensatz für den neuen MqTable-Rest-Service.

Nicht funktionale Anforderungen

Anforderung	Beschreibung
MA12	Die Seite ist stimmig ins UI eingebaut, bereits existierende UI-Komponenten werden wiederverwendet, oder falls nötig erweitert.
MA13	Die Seite wird in kurzer Zeit geladen, notwendiges Filtering wird auf der Datenbank oder dem Server gemacht.

9.2.2 Erweiterung: Pagination

Folgenden funktionale und nicht funktionale Anforderungen sind hier für die Erweiterung Pagination aufgelistet.

Funktionale Anforderungen

Anforderung	Beschreibung
EP1	Es können mittels Pagination auch mehr als 25 Einträge im Web-GUI angeschaut werden.
EP2	Eine Page beinhaltet maximal 25 Einträge.
EP3	Der Pagination-Mechanismus ist durch Unit-Tests bzw. SpringBoot-Tests abgedeckt.

Nicht funktionale Anforderungen

Anforderung	Beschreibung
EP4	Einzelne Pages werden erst wenn diese gebraucht werden vom Server geladen.
EP5	Die Navigation zwischen den Seiten ist intuitiv und nutzerfreundlich.

9.2.3 Erweiterung: Filter

Folgenden funktionale und nicht funktionale Anforderungen sind hier für die Erweiterung Filter aufgelistet.

Funktionale Anforderungen

Anforderung	Beschreibung
EF1	Es gibt folgende Filter-Kriterien: Filtern nach MQ_IN_STATUS, Filtern mittels Begriffen im Nachrichteninhalt und Filtern nach «Datum von» und «Datum bis»
EF2	Bei Verwendung mehrerer Filter werden diese mittels logischem UND verknüpft.
EF3	Die Filter-Werte können einfach aus dem Web-UI gesetzt werden.
EF4	Die Filterung passiert im Hintergrund, d.h. auf der Datenbank und/oder dem Server.
EF5	Die Einträge sind weiterhin sortiert nach MODIFIED_DATE.
EF6	Alle Filterkriterien und -werte sind in der URL abgebildet.
EF7	Alle Filter sowie ein Beispiel mit einer Kombination von Filtern sind als Unit-Tests bzw. SpringBoot-Tests abgedeckt.

Nicht funktionale Anforderungen

Anforderung	Beschreibung
EF8	Alle Filterkriterien und -werte sind in der URL abgebildet, um das Speichern und Teilen der Filter zu vereinfachen.

10 Planen

Dieses Kapitel zeigt die in der IPERKA-Phase «Planen» durchgeführten Arbeiten auf. In dieser Phase werden basierend auf den Anforderungen Arbeitspakete definiert und in einem Zeitplan auf die zehn Probe-IPA Tage aufgeteilt. Zudem werden Lösungskonzepte für die Umsetzung der Seite Business Daten erarbeitet und ein Testkonzept erstellt.

10.1 Arbeitspakete

Die gesamte Arbeit der Probe-IPA wird in Arbeitspakete aufgeteilt. Die Arbeitspakete bestehen aus einer zugehörigen Nummer, einem Namen, dem geschätzten Aufwand und ein erwartetes Ergebnis. In den geschätzten Aufwand sind Zeitreserven mit einberechnet, sodass unvorhergesehenes kompensiert werden kann. Da der Zeitplan in 2-Stunden-Blöcke aufgeteilt ist, wird der geringste Aufwand 2 Stunden sein und im zweier Takt nach oben gehen.

Die Arbeitspakete sind nach der Projektmanagementmethode IPERKA gegliedert. Probe-IPA-spezifische Arbeiten, wie die Expertenbesuche oder das Erstellen des Anhangs, die ausserhalb der eigentlichen Projekts stehen, werden unter «Rahmenaufgaben» aufgeführt.

10.1.1 Informieren

Folgende Arbeitspakete gehören zu der IPERKA-Phase «Informieren».

Nummer	1.1
Name	Projektumfeld analysieren und beschreiben
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Die Aufgabestellung ist beschrieben und für den Lernenden der Auftrag klar. Der Lernende kennt das Projektumfeld und hat es dokumentiert.

Nummer	1.2
Name	Anforderungen definieren
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Die Anforderungen werden klar definiert und unterteilt in funktionale und nicht funktionale Anforderungen.

10.1.2 Planen

Folgende Arbeitspakete gehören zu der IPERKA-Phase «Planen».

Nummer	2.1
Name	Arbeitspakete definieren
Geschätzter Aufwand	4h
Erwartetes Ergebnis	Die gesamten Arbeitspakete sind definiert und nummeriert nach den Phasen von IPERKA.

Nummer	2.2
Name	Zeitplan erstellen
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Der Zeitplan wird mithilfe der Arbeitspakete erstellt und die bereits erfüllten Aufgaben werden entsprechend markiert.

Nummer	2.3
---------------	------------

Name	Lösungskonzept für die Struktur vom Backend
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Ein Lösungskonzept für die Struktur im Backend wird erarbeitet und dokumentiert.

Nummer	2.4
Name	Lösungskonzept für die Struktur vom Frontend
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Ein Lösungskonzept für die Struktur im Frontend wird erarbeitet und dokumentiert.

Nummer	2.5
Name	Lösungskonzept für die Struktur von einer Erweiterung
Geschätzter Aufwand	4h
Erwartetes Ergebnis	Der Lernende entscheidet sich zwischen einer der beiden Erweiterungen und erarbeitet für dieses ein Lösungskonzept und dokumentiert diese anschliessend.

Nummer	2.6
Name	Testkonzept erstellen
Geschätzter Aufwand	4h
Erwartetes Ergebnis	Ein Testkonzept wird erarbeitet. Die zu schreibenden Tests und Testergebnisse sind definiert.

10.1.3 Entscheiden

Folgende Arbeitspakete gehören zu der IPERKA-Phase «Entscheiden».

Nummer	3.1
Name	Lösungsvarianten evaluieren
Geschätzter Aufwand	4h
Erwartetes Ergebnis	Mögliche Lösungsvarianten wurden evaluiert und die umzusetzende Lösungsvariante ist definiert.

10.1.4 Realisieren

Folgende Arbeitspakete gehören zu der IPERKA-Phase «Realisieren».

Nummer	4.1
Name	Endpoints für Mindestanforderungen erstellen
Geschätzter Aufwand	4h
Erwartetes Ergebnis	Die Endpoints für die Mindestanforderungen werden erstellt, sodass das Frontend alle Daten, die es braucht, und nur die, die es braucht, bekommt.

Nummer	4.2
Name	Seite und Tabelle im Frontend erstellen
Geschätzter Aufwand	4h
Erwartetes Ergebnis	Die Seite ist mit der entsprechenden URL und im Menü unter Business Daten → Queue-Messages (eingehend) erreichbar. Die Tabelle ist stimmig ins UI eingebaut und entspricht der tabellarischen Darstellung.

Nummer	4.3
Name	Endpoints für Erweiterung erstellen
Geschätzter Aufwand	4h
Erwartetes Ergebnis	Die Endpoints für die Erweiterung werden erstellt, sodass das Frontend alle Daten, die es braucht, und nur die, die es braucht, bekommt.

Nummer	4.4
Name	Erweiterung in die Tabelle integrieren
Geschätzter Aufwand	4h
Erwartetes Ergebnis	Die Erweiterung wird im Frontend in die Tabelle integriert. Das Design der Erweiterung muss stimmig zu der Tabelle und der Seite eingebaut werden.

Nummer	4.5
Name	ReleaseNotes schreiben

Geschätzter Aufwand	1h
Erwartetes Ergebnis	Für die neue Tabelle werden ReleaseNotes geschrieben, um die Tabelle und die Erweiterung zu beschreiben und erklären.

10.1.5 Kontrollieren

Folgende Arbeitspakete gehören zu der IPERKA-Phase «Kontrollieren».

Nummer	5.1
Name	Tests
Geschätzter Aufwand	6h
Erwartetes Ergebnis	Das geplante Testkonzept wird Umgesetzt und bei Bedarf ergänzt.

Nummer	5.2
Name	Codequalität prüfen
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Die Codequalität wird mithilfe von der Jenkins Pipeline überprüft und bei Bedarf überarbeitet.

Nummer	5.3
Name	Dokumentation finalisieren
Geschätzter Aufwand	8h
Erwartetes Ergebnis	Die Dokumentation ist nachvollziehbar und verständlich. Das Dokument wird nach Schreibfehlern durchsucht und verbessert, sodass zu diesem Zeitpunkt fast bis gar keine mehr übrig sind. Die Struktur ist einheitlich und Unschönheiten wurden behoben.

10.1.6 Auswerten

Folgende Arbeitspakete gehören zu der IPERKA-Phase «Auswerten».

Nummer	6.1
---------------	------------

Name	Kurzfassung schreiben
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Das Projekt wird in einer Kurzfassung zusammengefasst.

Nummer	6.2
Name	Reflexion schreiben
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Das Projekt wird vom Lernenden reflektiert und dokumentiert.

10.1.7 Rahmenaufgaben

Folgende Arbeitspakete gehören zu den Rahmenaufgaben.

Nummer	7.1
Name	Projektstruktur aufsetzen
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Aufbau der Gerüstes des L ^A T _E X-Berichtes, welches die Titelseite, ein Glossar, Das Quellenverzeichnis und ein Abbildungsverzeichnis beinhaltet. Ein Git Repository wird für die Dokumentation aufgesetzt.

Nummer	7.2
Name	Aufgabenstellung und Rahmenbedingungen beschreiben
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Die Aufgabenstellung und Rahmenbedingungen hat der Lernende verstanden und beschreibt sie nochmals um dies zu bestätigen.

Nummer	7.3
Name	Projektmanagementmethode definieren
Geschätzter Aufwand	2h

Erwartetes Ergebnis	Eine Projektmanagementmethode wird definiert und durch einer anderen wird erleutert warum sie gewählt wird.
----------------------------	---

Nummer	7.4
Name	Expertenbesuche
Geschätzter Aufwand	2h
Erwartetes Ergebnis	Die Expertenbesuche werden erfolgreich geplant und durchgeführt. Wichtige Informationen bezüglich der Besuche werden an passenden Plätzen erwähnt und dokumentiert.

Nummer	7.5
Name	Anhang erstellen
Geschätzter Aufwand	4h
Erwartetes Ergebnis	Ein Anhang mit allen gemachten Änderungen am Quellcode ist dokumentiert und klar von dem unveränderten Code unterscheidbar.

10.2 Lösungskonzept für die Struktur vom Backend

In diesem Abschnitt wird das Lösungskonzept für einen Teil der unter 9.2.1 definierten Anforderungen beschrieben.

10.2.1 Erstellung der Endpunkte

Im Backend existieren noch keine Endpunkte für die Message-Queues, um sie im Frontend darzustellen. Deshalb müssen diese neu implementiert werden. Die Endpoints werden in einer Service-Klasse erstellt, welche von einem Interface implementiert wird. Die Standardfunktionen, wie GET oder UPDATE, sind bereits im Interface definiert und müssen so nur noch in der Service-Klasse implementiert werden. Der Grund für das Interface ist, dass das Interface für die Service-Klasse und die Mock-Service-Klasse genutzt werden kann und beide Klassen die gleichen Funktionen haben aber mit einer unterschiedlichen Implementierung. Der Zugriff auf die Datenbank erfolgt in der Klasse ...

11 Entscheiden

12 Realisieren

13 Kontrollieren

14 Auswerten

Glossar

Backend Der Server-Teil eine Applikation, welcher meistens als Verbindung zwischen der Datenbank und dem Web-GUI genutzt wird. 4

CardX Eine Transaktions-Authorisierungs-Lösung von der Firma Ergon Informatik AG, die bei einigen Banken in der Schweiz im Einsatz ist. 4, 28, 47

Ergon Informatik AG Die Firma, in der ich meine Lehre absolviere. 4, 47

UI Ein UI (User Interface) ist eine Benutzeroberfläche von einer Software oder einem Gerät um mit der Software zu Interagieren. 4

Web-GUI Eine Webseite, die genutzt wird um das Projekt CardX zu überwachen und anzupassen. 4, 47

Abbildungsverzeichnis

1	Logo der Ergon Informatik AG (Ergon Informatik AG 2024)	1
1.1	Anzeigen und bearbeiten der Tasks	6
4.1	Arbeitsplatz des Lernenden	9
5.1	Git Commit History des Quellcodes	12
5.2	Git Commit History der Dokumentation	13
9.1	Ablauf eines GET-Requests für das Erstellen der Tabelle	30

Quellenverzeichnis

Bern, ICT Berufsbildung (2024). *Die 6-Schrittmethode IPERKA*. URL: https://www.ict-berufsbildung-bern.ch/resources/Iperka_OdA_200617.pdf (besucht am 06.11.2024).

Ergon Informatik AG (2024). *Logo der Ergon Informatik AG*. URL: <https://www.ergon.ch/dam/jcr:c42b10d3-a5c7-4ada-b8e5-ccc94f802da3/ergon-logo.png> (besucht am 06.11.2024).