IPA
Erstellen einer Adminmaske zur Erfassung von Standardwerten

Abbildung 1, Titelbild

Inhaltsverzeichnis

Teil 1 – Umfeld und Ablauf

1 Abstract (Kurzfassung)

1.1 Ausgangssituation

Finanzplanungen benötigen für die Berechnung verschiedene Parameter. Diese Parameter oder auch Standardwerte genannt, werden aktuell auf der Datenbank gepflegt. Es besteht jedoch kein UI, um diese einfach und simpel zu pflegen. Da jeweils eine manuelle Anpassung in der Datenbank sehr mühsam ist, wird als Lösung ein bestehendes Angular Projekt um ein neues Modul erweitert. Dieses Modul soll eine neue Adminmaske zur Erfassung der Standardwerte beinhalten. Zusätzlich sollen in der Adminmaske die «Enumerations» und Erläuterungen gepflegt werden können.

1.2 Umsetzung

Das Frontend der Adminmaske wird in einem bestehendem Angular Projekt als neues Modul erstellt. Die Kommunikation zwischen den Komponenten erfolgt mittels des Frameworks «NgRx». Die Endpunkte werden ebenfalls in einem bestehendem Backend hinzugefügt. Das bestehende Backend ist ein ASP .NET Core Web API (.NET 6.0) Projekt. Die neuen Endpunkte werden im bestehendem «ParamController» erstellt. Dieses Projekt wird innerhalb von 10 Tagen umgesetzt, mit der Projektmanagementmethode «IPERKA».

1.3 Ergebnis

Eine simple und einfache Webapplikation welche die Erfassung und Bearbeitung der Standardwerte, Enumerations und Erläuterungen vereinfacht.

Für die Enumerations ist eine Auflistung aller «EnumerationTypes» vorhanden. Nach Auswahl des Typen werden alle zugehörigen Enumerations angezeigt. Die jeweiligen Felder können bearbeitet werden und neue Einträge erstellt, sowie auch gelöscht werden.

Für die Standardwerte haben wir die gleiche Handhabung wie bei den Enumerations. Die Felder bestehen jedoch nur aus Startjahr und Wert.

Bei den Erläuterungstexten ist ein Editor ersichtlich welche die RTF-Texte darstellt. Die Erläuterungen sind nach Kategorie gruppiert. Die Texte können bearbeitet werden (Farben und Textdecorations setzen).

2 Aufgabenstellung (gemäss PKOrg)

2.1 Ausgangslage

Für die Berechnung von Finanzplanungen werden diverse gesetzlich definierte Parameter benötigt. Diese Parameter sind in einer unserer Anwendungen als Standardwerte hinterlegt. Die Erfassung dieser Parameter erfolgt heute meistens via Datachange direkt auf der Datenbank. In Zukunft soll eine Adminmaske mit sinnvollem Berechtigungskonzept angeboten werden, in der die Werte direkt vom Business gepflegt werden können.
Ziel der IPA ist es eine initiale Version (ohne Berechtigungskonzept) dieser Maske zu schaffen. Konkret bedeutet das, dass eine bestehende Angular Webapplikation um ein weiteres Modul erweitert werden soll, und im Backend entsprechende CRUD-Endpunkte für die Bearbeitung gebaut werden müssen.

2.2 Detaillierte Aufgabenstellung

2.2.1 Frontend

Es wird ein neues Modul in einer bestehenden Angular Anwendung erstellt, das Angular-CLI steht dem Kandidaten frei zur Verfügung.

Die Kommunikation zwischen verschiedenen Komponenten erfolgt mittels NGRX, es werden keine darauf aufbauenden Frameworks verwendet. Die Struktur ist dem bestehenden Projekt zu entnehmen

- Ordner “state”
  - <StateName>.actions.ts
    -> Actions als konstante erstellen
  - <StateName>.effects.ts
    -> beinhaltet die Effekte
  - <StateName>.reducer.ts
    -> beinhaltet den Reducer und Initialen State
  - <StateName>.selector.ts
    -> Alle Selectors
  - <StateName>.state.ts
    -> beinhaltet ein Interface, welches den State definiert

Das Modul beinhaltet 3 Menu punkte für die 3 zu erstellenden Masken

2.2.2 Backend

Verwendet wird das bestehende .Net 6.0 API-Projekt “VZ.Finanzdaten.Service”, die neu erstellten Endpunkte sind im **ParamController** unterzubringen.

Angeboten wird:

- EnumerationType
  - GET
- Enumeration
  - GET, PUT, POST, DELETE
- DefaultValue
  - GET, PUT, POST, DELETE
- ParamSimulationNote (Erläuterungen)
  - GET
- TextConstant
  - GET
- TextConstantTranslation
  - GET, PUT, POST, DELETE

Der Datenbank Zugriff erfolgt mittels Entity-Framework-Core.

Die DTOs sollen keine Kopie der DB-Entitäten sein, sie beinhalten nur die nötigen Werte. Das Mapping erfolgt auf dem Service Layer.

2.2.3 Features

2.2.3.1 Auflistungen

Die Auflistungen beinhalten sämtliche EnumerationTypes aus der Datenbank.

Die Typen sollen in einem SideNav mit ihrer Deutschen Beschreibung aus der Datenbank angezeigt werden. Bei Auswahl eines Typen werden Sämtliche Enumeration angezeigt, welche diesem Typen angehängt sind.

Angezeigt werden hier die Felder:

- Beschreibung
- Abkürzung
- ProgramCode
- Sortierung

Die Felder sind alle bearbeitbar, es besteht die Möglichkeit einzelne Einträge zu löschen und neue Einträge zu erstellen. Das Erstellen neuer Einträge erfolgt innerhalb eines EnumerationType, Der EnumerationType eines Eintrages kann nicht geändert werden.

Es können keine neuen Typen erstellt werden. Für die Typen muss aufgrund der geringen Datenmenge keine Suche angeboten werden.

Innerhalb der Typen wird eine Suche angeboten um die Einträge zu filtern. Gesucht werden kann nach ProgramCode und Beschreibung. Die Suche wird mit einem Debounce ausgelöst, es gibt keinen Button, um die Suche auszulösen.

Datenbasis:

Wie erwähnt ist die Grundlage die Tabelle “EnumerationType”. Die Editierbaren EnumerationTypes lassen sich identifizieren anhand ihrer Enumerations. Grundsätzlich ist die Logik: Lade alle EnumerationTypes wo keine Enumerations mit “IsSystem” = 1 vorhanden.

Zusätzlich soll der Type “DefaultValue” ausgeschlossen werden, da hierfür eine Separate Maske angeboten wird.

2.2.3.2 Standardwerte

Ähnlich wie bei den Auflistungen sollen hier sämtliche Standardwert-Typen in einem SideNav angezeigt werden.

Mittels Klick auf einen Typen werden sämtliche Werte des gesagten Typen angezeigt.

Angezeigte Felder:

- Startjahr
- Wert (Zahl mit 4 Dezimalstellen)

Die Felder sind bearbeitbar und es können bestehende Einträge gelöscht werden, wie auch neue Einträge erstellt werden.

Es können keine neuen Typen erstellt werden.

Datenbasis:

Die Standardwerte befinden sich auf der Tabelle “DefaultValue”, Pflichtfelder sind “Year” und “ValueDecimal”. Die Zuordnung erfolgt über die Spalte “DefaultValueTypeEnumerationID”.

Die Vorhandenen Standardwerte können entsprechend aus der Tabelle “Enumeration” ausgelesen werden. Es handelt sich hierbei um alle Enumerations mit EnumerationTypeID 15 (DefaultValue)

2.2.3.3 Erläuterungen

Bei den Erläuterungen handelt es sich um Textkonstanten in den 4 Sprachen des VZ.

Die Erläuterungen sind als RTF-Texte gespeichert und es muss die Möglichkeit bestehen einzelne Wörter einzufärben. Aktuell gibt es im VZ keine Komponente zur Anzeige / Bearbeitung von RTF-Texten. Falls auf eine externe Komponente gesetzt wird, sollen verschiedene Varianten verglichen und evaluiert werden.

Die Erläuterungen sind unterteilt nach Kategorie, Textbezeichnung und Übersetzungen. Der genaue Aufbau des GUIs ist durch den Kandidaten frei wählbar, Anforderungen an das GUI:

- Pro Erläuterungstext (Typ) werden die Texte in allen 4 Sprachen angezeigt (Unabhängig davon, ob sie auf der DB existieren oder nicht).
- Die Erläuterungen sind nach Kategorie gruppiert.
- Farben, Textdecorations (bold, underline, italic) können nach Belieben im Text gesetzt werden.

Vorsicht: In den Texten sind Variablen hinterlegt. Diese sind erkennbar anhand von pre- und postfix “%”, also “%<VariableName>%”. Die korrekte Handhabung von Variablen ist nicht Bestandteil der IPA, die Variablen müssen nur erkannt werden und dem User auf beliebige Art (z.B. Tooltip oder Text unterhalb der Eingabe) angezeigt werden.

Datenbasis:

Die Erläuterungen sind in der Tabelle “ParamSimulationNote” untergebracht. Aus der Tabelle ersichtlich ist eine “CategoryEnumeration” (Eingaben, Ausgaben, Steuern, etc.) und eine SubCategoryEnumeration (effektiver Textbaustein, bsp: Erwerbseinkommen, PK-Einkauf, Hypo-Zinsen, etc.)

Die beiden erwähnten Enumerations dienen der Identifizierung des Jeweiligen Textbausteins, also Kategorie und Unterkategorie, sie können auch direkt aus der Tabelle “Enumeration” geladen werden. Es handelt sich hier um alle Enumerations mit EnumerationTypeID:

- 100 (SimulationNoteCategory)
- 101 (SimulationNoteSubcategory)

Die zu bearbeitende Texte, finden sich auch in der Tabelle “ParamSimulationNote” und haben jeweils den Postfix “TextConstantID”:

- CategoryTextConstantID
  -> sollte in einem Übergeordneten Menu bearbeitbar sein.
- HeaderTextConstantID
  -> gehört zu einem Textbaustein und muss unter diesem bearbeitbar sein.
- BodyTextConstantID
  -> gehört zu einem Textbaustein und muss unter diesem bearbeitbar sein.

Anhand der TextConstantID können anschliessend die Texte aus der Tabelle “TextConstantTranslation” ausgelesen werden. Diese Tabelle beinhaltet einen Text “sysText” und eine Sprache “LanguageEnumerationID”. Es soll pro Sprache ein Text generiert werden. Sprachen sind alle Enumerations mit EnumerationTypeID: 7 (Language).

2.2.4 Allgemein

- Sämtliche Eingabefelder werden vor dem Speichern validiert
- Fehler werden abgefangen und behandelt. Anschliessend werden sie dem User leserlich in einem Toast oder Popup angezeigt.
- Die Speicherung der Daten erfolgt mittels eines Buttons “Speichern” eine Automatische Speicherung aufgrund von Fokus o.ä. ist nicht erwünscht.

2.2.5 Design

Da es sich um eine interne Applikation handelt gibt es keine Design Vorgaben. Es ist dem Kandidaten gestattet selbst zu entscheiden wie das UI am besten aufgebaut werden soll.

Für die Eingaben sollen Komponenten aus dem Angular Material Design verwendet werden, auf externe Komponenten ist generell zu verzichten. Falls dennoch externe Komponenten eingesetzt werden, muss dies begründet dokumentiert werden.

Wir verwenden für die Farben ein Style-Sheet, hard-codierte Farben in den CSS-Dateien sind nicht gestattet (mit Ausnahme “transparent”).

2.2.6 Datenbasis

Die bestehende Datenbank sowie Datenbasis müssen wiederverwendet werden. Erweiterungen um neue Felder oder Eigenschaften ist nicht gestattet.

Relevante Tabellen:

- Enumeration
- EnumerationType
- DefaultValue
- TextConstant
- TextConstantTranslation

2.2.7 Erweiterbarkeit

Es ist davon auszugehen, dass das erstellte Admin-Modul in Zukunft um weitere Funktionen erweitert wird. Entsprechend sollen Komponenten fürs Layouting angelegt werden, damit die Entwicklung neuer Funktionen vereinfacht werden.

Bei der Erstellung neuer Komponenten ist eine Sinnvolle Ordnerstruktur zu wählen.

2.2.8 Berechtigungskonzept

Die Berechtigungen werden über unsere Infrastruktur gesteuert. Es muss keine zusätzliche Rücksicht darauf genommen werden, alle neu erstellten Endpoints dürfen Public sein.

2.2.9 Prüfbare / Messbare Ziele

- Alle Unittests müssen fehlerfrei durchlaufen.
- Die Buildpipeline muss verhindern, dass eine Version mit fehlerhaften Unittests deployed werden kann.
- Fehleingaben werden dem User angezeigt
- Required Fields sind im UI entsprechend gekennzeichnet

2.2.10 Tests

Es müssen keine Tests fürs Frontend erstellt werden.

Im Backend ist Businesslogik wo sinnvoll mit Unittests zu testen. Wir testen keine externen Technologien (also keine Tests für CRUD-Operationen im Entity Framework).

2.2.11 Architektur

Die Maske wird in eine bestehende Anwendung eingebaut. Somit gilt es die bisherige Architektur auch weiterhin zu verwenden. Die Bestehende Anwendung ist in einer Client-Server-Architektur umgesetzt. Zwingend ist bei der Programmierung auch das Dependency Injection Pattern zu verwenden.

2.2.12 Dokumentation

- Aus den Anforderungen abgeleitete Use-Cases in Form von PBIs.
- Endpoints im Backend müssen im Swagger dokumentiert werden.
- Eigenschaften aller DTOs müssen im Code dokumentiert werden.
- Wo nötig sind Einschränkungen, Annahmen oder offene Punkte im Code zu dokumentieren.
- Eine Beschreibung der System-Architektur muss erstellt werden.
- Der Einsatz von externen Libraries muss dokumentiert sein (Ausnahme: Libraries von Microsoft im Zusammenhang mit ASP.Net oder Libraries, die bereits in der Applikation eingesetzt werden).

2.2.13 Fehlerbehandlung

Alle aufgetretenen Fehler im Backend müssen zwingend geloggt werden. Es dürfen keine Technischen Fehlermeldungen im Frontend angezeigt werden.

Bei unbehandelten Exceptions wird eine generische Meldung angezeigt.

2.2.14 Kompatibilität

- Die UI-Applikation muss in einer aktuellen Chrome-Version unter Windows bedienbar sein.
- Die UI-Applikation muss nur bei einem 16:9 Seitenverhältnis korrekt aussehen
- Die UI-Applikation muss auf 1080px und 2560px korrekt aussehen.

2.2.15 Lokalisierung

Der Aufbau der bestehenden Entitäten soll nicht erweitert werden. Texte welche nur auf Deutsch erfasst sind, werden also nicht übersetzt.

Statische Texte im GUI werden mittels der TranslatePipe übersetzt. Es ist für die Arbeit ausreichend, wenn nur die Deutschen Texte gepflegt werden.

2.2.16 Explizit ausgeschlossene Arbeiten

- Statische Texte im GUI müssen nicht in andere Sprachen übersetzt werden.
- Das Handling von Variablen in RTF-Texten muss nicht implementiert werden.
- Das GUI muss nicht auf einem Tablet oder Smartphone bedienbar sein.
- Das entwickelte System muss auf dem Rechner des Kandidaten laufen, ein Deployment ist nicht Teil der Arbeit. Für die Präsentation darf der Code in unserer “Dev” Umgebung ausgerollt werden.

3 Vorarbeiten

- Devops Board/Swimlane wurde bereits erstellt.
- Zugriff auf alte IPAs von vorherigen Absolventen.
- Probe-IPA mit einem internen Arbeitsprojekt.
- Bestehende Komponenten im bestehenden Projekt, können benutzt werden.

4 Vorkenntnisse

In den vergangenen 6 Monaten hat der Kandidat folgende Technologien fast täglich eingesetzt:

- C#
- Visual Studio 2022
- .NET 6.0 / .NET Core
- Angular / Typescript

Arbeiten in den letzten 6 Monaten:

- Welche Art von Arbeiten hat die/der Lernende im letzten Halbjahr durchgeführt?
  - Mitarbeiten im Scrum-Team mit der Rolle "Entwickler"
  - Diverse Anpassung in bestehenden Applikationen (.Net Core, ASP.Net Web API, MSSQL)
- Welches waren die zwei grössten Aufträge?
  - Historisierung bzw. Snapshot-Funktionalität in eine von uns Berechnete Auswertung einbauen.
  - Redesign unseres Kundenreportings (inkl. Migration Crystal → Devexpress)
- Welche Tools wurden dafür eingesetzt?
  - Visual Studio 2022
  - Microsoft SQL Server Management Studio 18
  - Microsoft Azure Devops Server

5 Projektaufbauorganisation

Lehrbetrieb und Durchführungsort:
VZ VermögensZentrum AG
Gotthardstrasse 6, 8002 Zürich
044 207 27 27

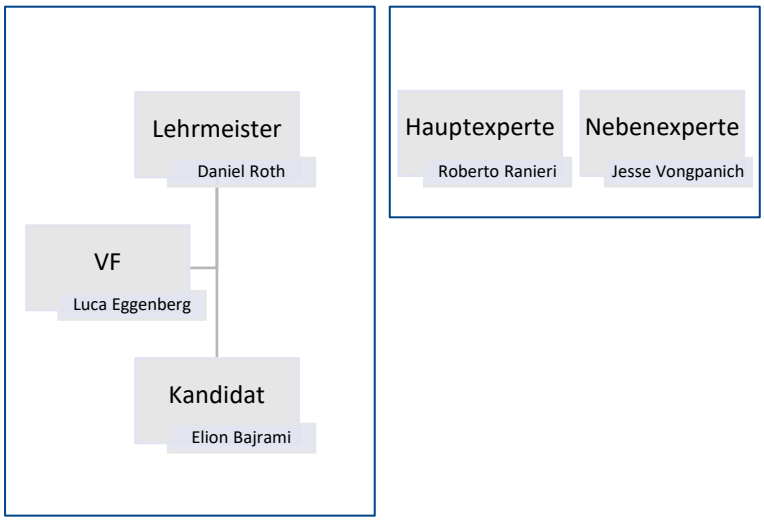
Kandidat:
Elion Bajrami
elion.bajrami@vzch.com

Lehrmeister:
Daniel Roth
daniel.roth@vzch.com

Verantwortliche Fachkraft:
Luca Eggenberg
luca.eggenberg@vzch.com

Hauptexperte:
Roberto Ranieri
roberto.ranieri@gmx.net

Nebenexperte:
Jesse Vongpanich
jesse.vongpanich@gmail.com



6 Backups

Alle Files die zum Bericht gehören werden in der persönlichen Dokumentenablage abgelegt. Es wird für jeden Tag ein neuer Ordner erstellt, somit kann jederzeit auf die alten Versionen zugegriffen werden.

Die persönliche Dokumentenablage befindet sich auf einem Server, von dem täglich über Nacht automatisch ein Backup erstellt wird. Alle Dokumente und Dateien werden dort abgelegt.

V1	07.03.24 17:41	Dateiordner
V2	08.03.24 17:29	Dateiordner
V3	11.03.24 19:12	Dateiordner
V4	13.03.24 19:27	Dateiordner
V5	14.03.24 18:52	Dateiordner
V6	15.03.24 18:53	Dateiordner
V7	18.03.24 19:46	Dateiordner
V8	20.03.24 20:35	Dateiordner
V9	21.03.24 16:47	Dateiordner

7 Projektmanagementmethode

Als Projektmanagementmethode wurde **IPERKA** gewählt. Durch diese Methode wird das Projekt in sechs Schritte aufgeteilt. Welche auch die Gliederung der Dokumentation im zweiten Teil darstellt.

Abbildung 2, IPERKA

8 Versionsverwaltung

Git wird im Zusammenhang mit Azure DevOps als Versionsverwaltung verwendet. Für jedes Feature/PBI wird ein neuer Branch erstellt. Somit kann man beliebig entwickeln, ohne den Haupt-Branch zu zerstören. Man kann also jederzeit wieder auf die bestehende Version wechseln sollte der Feature-Branch nicht funktionieren.

Der Main-Branch hat eine Policy, bei der es nicht möglich ist, darauf zu pushen. Neue Features können nur mit einem Pull Request eingebaut werden. Für diese IPA wird ein neuer Branch, vom Main-Branch erstellt. Dieser dient dann als Haupt-Branch für die ganze IPA und daraus werden auch die Feature-Branches erstellt. Sobald ein Feature/PBI abgeschlossen ist, wird ein Pull-Request erstellt, um den Feature-Branch in den Haupt-Branch zu mergen (in unserem Fall IPA-Branch).

Visualisiert sieht das so aus:

- Main-Branch
    - IPA-Branch
        - Feature-Branch

9 Interne Coding-Guidelines

9.1 Sprache

Programmcode sowie Kommentare sind in Englisch zu verfassen. Fach und/oder VZ-spezifische Begriffe sind in Deutsch zu ergänzen.

9.2 Vergangenheitsbewältigung

Muss im Rahmen von Bugfixes, Change Requests o.ä. Code angepasst/erweitert werden, so soll der betroffene Code gleich auf diese Vorgaben hin geprüft und ggf. angepasst werden.

9.3 Bezeichnungen

Die Bezeichnung von Variablen, Methoden, Klassen etc. sind so zu wählen, dass im Idealfall keine zusätzliche Kommentierung notwendig ist. Gestaltet sich dies schwierig, so ist dies Indiz für eine suboptimale Strukturierung u/o Architektur.

9.4 Aufbau von Klassen

Klassen sollen wie folgt strukturiert werden:

- Private Felder
- Properties
- Konstruktoren
- Public Methoden
- Private Methoden

9.5 Ordner

Die Solution soll eine sinnvolle Ordnerstruktur aufweisen (z.B. alle Repositories, Domänen-Objekte in einen separaten Ordner). Jeder Ordner erweitert den Namespace um den Namen des Ordners.

9.6 Namenskonventionen

Wir verwenden Pascal-Casing oder Camel-Casing. D.h. keine „_“-Präfixe und keine Datentypen als Präfix („intCounter“). Die verschiedenen Typen sind wie folgt zu benennen.

Type	Case	Notes
Class	Pascal Casing	No leading “C”. Do not add “Class” at the end.
Constants	Pascal Casing	
Delegate class	Pascal Casing	End with “EventHandler”.
Enum values	Pascal Casing	
Enum type	Pascal Casing	Do not add “Enum” at the end. Suffix with e.g. Type or State
Events	Pascal Casing	
Event Methodes	Pascal Casing	Start with “On”.
Exception class	Pascal Casing	End with “Exception”.
Fields (controls)	Camel Casing	Start with control type (e.g. buttonCancel).
Fields (internal)	Camel Casing	
Fields (private)	Camel Casing	
Fields (protected)	Camel Casing	
Fields (public)	Camel Casing	
Form class	Pascal Casing	End with “Form”.
Interface class	Pascal Casing	Starts with “I” following by a capitalized letter.
Methods	Pascal Casing	
Namespace	Pascal Casing	Use CompanyName.TechnologyName as root.
Parameters	Camel Casing	
Property	Pascal Casing	
Struct	Pascal Casing	
Variables	Camel Casing	

Table 1: Namenskonventionen by PKal

Abbildung 3, Namenskonventionen

10 Zeitplan

Starttermin: 07.03.2024

Enddatum: 22.03.2024

Mit Hilfe der Aufgabenstellung wurde folgender Zeitplan erstellt. Darin ist der Soll-Ist-Vergleich klar ersichtlich. Der Zeitplan ist im 2-Stunden-Raster (2h 15min) aufgebaut, Blau ist die Soll-Zeit und hell-Blau die Ist-Zeit. Grün sind die Meilensteine, der erste Meilenstein ist das Abschliessen der Phase Entscheidung und er zweite das Abschliessen der Realisierung.

Anfang:

Zeitplan - Adminmaske					Do., 7. März 2024		Fr., 8. März 2024		Mo., 11. März 2024		Mi., 13. März 2024		Do., 14. März 2024		Fr., 15. März 2024		Mo., 18. März 2024		Mi., 20. März 2024		Do., 21. März 2024		Fr., 22. März 2024	
					08:00	12:45	08:00	12:45	08:00	12:45	08:00	12:45	08:00	12:45	08:00	12:45	08:00	12:45	08:00	12:45	08:00	12:45	08:00	12:45
Aufgabe	**Anfang**	**Ende**		**Zeit in h**	12:15	17:00	12:15	17:00	12:15	17:00	12:15	17:00	12:15	17:00	12:15	17:00	12:15	17:00	12:15	17:00	12:15	17:00	12:15	17:00
Zeitplan erstellen	07.03.24	07.03.24	Soll	2	2.00																			
			Ist	2.5	2.50																			
Arbeitsjournal Vorlage erstellen	07.03.24	07.03.24	Soll	0.5	0.50																			
			Ist	0.5	0.50																			
INFORMIEREN ↓																								
Projektanforderungen studieren	07.03.24	07.03.24	Soll	0.5	0.50																			
			Ist		0.50																			
Datenbank studieren	08.03.24	08.03.24	Soll	0.5			0.50																	
			Ist																					
Projektumfeld	08.03.24	08.03.24	Soll	1			1.00																	
			Ist																					
PLANEN ↓																								
Realisierungskonzept erstellen	08.03.24	08.03.24	Soll	0.5			0.50																	
			Ist																					
Konzeptionelle Umsetzung	08.03.24	08.03.24	Soll	1.5			1.50																	
			Ist																					
Testkonzept erstellen	08.03.24	08.03.24	Soll	2			2.00																	
			Ist																					
PBIs erstellen	08.03.24	08.03.24	Soll	2.25				2.25																
			Ist																					
ENTSCHEIDEN ↓																								
Lösungsvariante festlegen	11.03.24	11.03.24	Soll	1					1.00															
			Ist																					
REALISIEREN ↓																								
Haupt-Feature-Branch erstellen	11.03.24	11.03.24	Soll	0.25					0.25															
			Ist																					
CRUD Elemente erstellen	11.03.24	13.03.24	Soll	5					1.00	2.75	1.25													
			Ist																					
Frontend Masken erstellen	13.03.24	15.03.24	Soll	15							2.00	3.00	4.25	0.75	3.25	1.75								
			Ist																					
Datenanbindung	15.03.24	18.03.24	Soll	5												0.50	2.75	1.75						
			Ist																					
KONTROLLIEREN ↓																								
Applikation testen	15.03.24	15.03.24	Soll	3															1.25	1.25				
			Ist																					
AUSWERTEN ↓																								
Fazit	21.03.24	21.03.24	Soll	3																	2.25	0.75		
			Ist																					
FORTLAUFENDE TÄTIGKEITEN ↓																								
Arbeitsjournal führen	07.03.24	22.03.24	Soll	5		0.50		0.50		0.50		0.50		0.50		0.50		0.50		0.50		0.50		0.50
			Ist																					
Dokumentation	07.03.24	22.03.24	Soll	36.25		3.75	0.25	1.75	2.00	1.00	1.00	1.25	2.00	1.25	1.00	1.50	1.50	2.00	2.00	2.00	2.00	2.00	4.25	3.75
			Ist																					
Expertenbesuch	07.03.24	22.03.24	Soll	2	1.00														1.00					
			Ist	1	1.00																			

86.25

Abbildung 4, Zeitplan Anfang

Ende: TODO

11 Arbeitsjournal

Tägliches Arbeitsjournal, Tag 1, 07.03.2024	
Geplante Ziele	Zeitplan erstellen Arbeitsjournalvorlage erstellen Dokumentation starten Expertenbesuch
Tätigkeiten und erreichte Ziele	Heute habe ich den Zeitplan erstellt und habe nun mal eine erste und solide Version. Heute hatte ich meine ersten Expertenbesuch mit Herr Ranieri. Er gab mir hilfreiche Tipps für meine IPA. Nachdem ich den Expertenbesuch hatte, ging es ans Erstellen der Dokumentation und einer guten Arbeitsjournalvorlage. Ich konnte die Arbeitsjournalvorlage mühelos erstellen. Die Dokumentation war heute reine Fleissarbeit. Ich konnte schon einen Grossteil des ersten Teils der Dokumentation abschliessen.
Aufgetretene Probleme	Heute gab es keine Probleme 😀
Beanspruchte Hilfestellungen	Als Grundgerüst für meinen Zeitplan habe ich dieses YouTube Video als Hilfestellung genommen: https://www.youtube.com/watch?v=e86l9FNQGsU VF: Mir war gemäss der Aufgabenstellung und dem Kriterienkatalog nicht ganz klar, ob ich ein Textkonzept erstellen muss. Dies habe ich dann mit meiner Verantwortlichen Fachkraft geklärt.
Vergleich mit dem Zeitplan/ Überzeiten	Zuerst war ich leicht im Rückstand, da die Erstellung des Zeitplans doch etwas länger dauerte. Jedoch konnte ich diese Zeit gut aufholen, während des Schreibens des ersten Teils der Dokumentation.
Ungeplante Arbeiten	Heute sind keine ungeplanten Arbeiten aufgetreten.
Reflexion	Ich war heute sehr aufgeregt, da dies mein erster IPA-Tag war. Die Erstellung des Zeitplans war am Anfang recht mühsam und zeitaufwendig. Im Nachhinein ist es jedoch gut gelungen und der Zeitplan ist auch sehr hilfreich. Die Erstellung der Dokumentation verlief gut und ich konnte schon viel dokumentieren.

Tägliches Arbeitsjournal, Tag 2, 08.03.2024	
Geplante Ziele	Datenbank studieren Projektumfeld aufzeichnen Realisierungskonzepterstellen Konzeptionelle Umsetzung aufzeichnen Testkonzept erstellen PBIs erstellen
Tätigkeiten und erreichte Ziele	Zuerst habe ich wie geplant die Daten und Queries die ich für dieses Projekt brauche studiert. Da sind ein paar Unklarheiten aufgetreten, welche ich aber am Daily besprechen konnte. Danach zeichnete ich ein Projektumfeld, welches gut geling. Dann erstelle ich ein kurzer Realisierungskonzept, um nochmals alles vor Augen zu haben. Dann zeichnete ich eine Konzeptionelle Umsetzung des Projektes. Dann kamen die Testkonzept, dies war mal wieder reine Fleissarbeit. Zum Schluss erstellte ich noch alle PBIs, welche für die Umsetzung benötigt werden (schon wieder viel Fleissarbeit 😅).
Aufgetretene Probleme	Unklarheiten bei den Daten
Beanspruchte Hilfestellungen	Am Daily Unklarheiten bezüglich Daten mit VF besprochen
Vergleich mit dem Zeitplan/ Überzeiten	Es läuft aktuell alles gemäss Zeitplan 🕒
Ungeplante Arbeiten	Heute gab es zum Glück keine Ungeplanten Arbeiten.
Reflexion	Heute ging es nur ums Informieren und Planen, da war sehr sehr viel schreiben angesagt. Ich konnte jedoch alles gut umsetzen und habe nun eine klare Vision wie das Projekt am Schluss aussehen soll. Nach so viel schreiben, freue ich mich jetzt dafür aufs coden 🧑‍💻.

Tägliches Arbeitsjournal, Tag 3, 11.03.2024	
Geplante Ziele	Lösungsvarianten festlegen Feature-Branch erstellen CRUD-Elemente erstellen Anfangen
Tätigkeiten und erreichte Ziele	Ich konnte heute leider nicht viel erreichen. Ich habe einen Editor evaluiert, welchen ich benutzen werde, um die Erläuterungen zu bearbeiten. Jedoch gibt es Probleme beim RTF-Text, den wir aus der DB holen. Nach stundenlanger Recherche habe ich ein gutes Nuget gefunden, wie wir RTF-Texte zu HTML konvertieren können im Backend. Schlussendlich habe ich auch eine alternative Lösung gefunden, um von HTML nach RTF zu konvertieren. Alternativ, weil diese Kostenpflichtig ist.
Aufgetretene Probleme	RTF-Text Konvertierung zu HTML und umgekehrt
Beanspruchte Hilfestellungen	VF
Vergleich mit dem Zeitplan/ Überzeiten	Ich bin nun leider recht zurückgefallen, wegen der Konvertierung von RTF-Text zu HTML
Ungeplante Arbeiten	Stundenlange Recherche von RTF-Text zu HTML-Konvertierungen
Reflexion	Heute war kein guter Tag 😟, obwohl ich eine gute Evaluierung für einen Editor vollbringen konnte. Beim Rest ging es leider nur im Schneckentempo weiter und ich konnte meine CRUD-Elemente noch nicht wie geplant Anfangen umzusetzen.

Tägliches Arbeitsjournal, Tag 4, 13.03.2024	
Geplante Ziele	CRUD-Elemente vollständig implementieren
Tätigkeiten und erreichte Ziele	Ich konnte heute erfolgreich alle benötigten CRUD-Elemente implementieren. Die CRUD-Elemente sind nicht 1 zu 1 wie in der Aufgabenstellung vermerkt, dies wurde aber mit der VF klar vorher besprochen.
Aufgetretene Probleme	Heute gab es zum Glück keine Probleme
Beanspruchte Hilfestellungen	Mit VF abgesprochen, wegen den Endpoints
Vergleich mit dem Zeitplan/ Überzeiten	Ich bin leider immer noch ein wenig im Verzug, wegen des Rückfalls am Montag
Ungeplante Arbeiten	Heute gab es zum Glück keine ungeplanten Arbeiten
Reflexion	Heute verlief der Tag schon viel besser als der Montag. Ich konnte alle benötigten CRUD-Elemente im Backend umsetzten. Morgen kann ich nun mit dem Frontend anfangen, dass wir eine Challenge 😅

Tägliches Arbeitsjournal, Tag 5, 14.03.2024	
Geplante Ziele	Anfangen mit der Erstellung der Frontendmasken, wenn es gut läuft auch schon heute abschliessen
Tätigkeiten und erreichte Ziele	Ich konnte heute ein neues Modul für die AdminPage im Frontend erstellen. Den Header und den SideNav konnte ich ebenfalls schon fertig implementieren. Beim SideNav sind es aber aktuell nur Testdaten.
Aufgetretene Probleme	Heute gab es keine grossen aufgetretenen Probleme
Beanspruchte Hilfestellungen	https://material.angular.io/
Vergleich mit dem Zeitplan/ Überzeiten	Ich konnte die Zeit leider noch nicht ganz aufholen
Ungeplante Arbeiten	Keine
Reflexion	Heute war ein produktiver Tag, in dem ich bedeutende Fortschritte im Bereich des Frontend-Designs erzielen konnte. Die Arbeit ging zügig voran und ich konnte erste, sichtbare Verbesserungen und Entwicklungen in der Benutzeroberfläche feststellen. Dieser Fortschritt ist ein ermutigender Schritt in Richtung der Fertigstellung meines Projektes.

Tägliches Arbeitsjournal, Tag 6, 15.03.2024	
Geplante Ziele	Frontendmasken fertig implementieren und die Daten anbinden.
Tätigkeiten und erreichte Ziele	Ich konnte heute leider nicht alles abschliessen. Ich konnte die Daten-Typen/Kategorien für jeweils alle 3 Features laden und anzeigen. Inklusive richtiger anzeige mit TreeView
Aufgetretene Probleme	Heute gab es technische Probleme im ganzen VZ. Die ganze Server Domäne ist abgestürzt und keine einzige Datenbank funktionierte im VZ für eine Stunde. Start: 11:20 Uhr, Ende 12:13 Uhr
Beanspruchte Hilfestellungen	https://material.angular.io/
Vergleich mit dem Zeitplan/ Überzeiten	Wir sind leider immer noch ein wenig im Verzug
Ungeplante Arbeiten	TreeView hat zu viel Zeit beansprucht. Die Anzeige der Unterkategorien hat Mühe gemacht.
Reflexion	Heute war kein schlechter Tag. Ich konnte wieder Fortschritte im Frontend erzielen und die ersten Daten laden und entsprechend anzeigen.

Tägliches Arbeitsjournal, Tag 7, 18.03.2024	
Geplante Ziele	Datenanbindung so weit wie möglich fertigstellen
Tätigkeiten und erreichte Ziele	Ich konnte die Datenanbindung ans Frontend komplett fertigstellen. Daten werden alle richtig angezeigt und man kann diese auch bearbeiten. Es fehlen nur noch ein paar UI-Einstellungen.
Aufgetretene Probleme	UI hat Probleme gemacht. Bis alles so aussieht wie es aussehen soll hat ein wenig gedauert.
Beanspruchte Hilfestellungen	Mit VF besprochen wegen Userhandling
Vergleich mit dem Zeitplan/ Überzeiten	Ein wenig im Verzug
Ungeplante Arbeiten	Unnötige UI-Anpassungen
Reflexion	Ich konnte heute einen guten Fortschritt erzielen. Ich konnte die Datenanbindung fertigstellen und konnte im UI vieles umsetzen. UI ist jedoch immer recht mühsam und fordert viel zu viel Zeit.

Tägliches Arbeitsjournal, Tag 8, 20.03.2024	
Geplante Ziele	Daten korrekt und Benutzerfreundlich darstellen inklusive gutem Handling
Tätigkeiten und erreichte Ziele	Ich konnte heute alle Daten Benutzerfreundlich darstellen. Das ganze UI wurde aufgeräumt. Die Validierung und Feedback für den User wurden eingebaut.
Aufgetretene Probleme	Bei den Erläuterungstexten wird die Schriftart nicht übernommen. Daher wird die Schriftart im Backend bei der Konvertierung gesetzt. Loading-Spinner wird nicht angezeigt, Daten werden schnell geladen, aber Rendering dauert zu lange.
Beanspruchte Hilfestellungen	
Vergleich mit dem Zeitplan/ Überzeiten	Ich bin leider im Verzug im Vergleich mit dem Zeitplan. Jedoch habe ich genug Reserven eingeplant.
Ungeplante Arbeiten	Erläuterungstext Schriftarten setzen.
Reflexion	Ich konnte heute im Code alles abschliessen und kann mich nun auf das Abschliessen der Dokumentation fokussieren. Die Erläuterungstexte waren wieder recht mühsam und haben unnötig viel Zeit gekostet. Deswegen bin ich leider im Verzug zum Zeitplan.

Tägliches Arbeitsjournal, Tag 9, 21.03.2024	
Geplante Ziele	Applikation testen Dokumentation abschliessen
Tätigkeiten und erreichte Ziele	Ich konnte heute erfolgreich die Applikation testen und dies im Protokoll festhalten. Danach ging es ans Abschliessen der Dokumentation und alle wichtigen Punkte noch abschliessen.
Aufgetretene Probleme	Keine
Beanspruchte Hilfestellungen	--
Vergleich mit dem Zeitplan/ Überzeiten	Das Testen der Applikation hätte gestern schon stattfinden müssen, da aber der letzte Tag als Reserve geplant war, ist dies kein Problem.
Ungeplante Arbeiten	Heute verlief zum Glück alles gut und es gab keine ungeplanten Arbeiten
Reflexion	Heute konnte ich guten Fortschritt erzielen, da ich die Applikation gestern fertigstellen konnte, war heute das Testen dran. Da stach hervor, dass bei den Auflistungen gewisse Typen zu lange brauchen um zu Laden. Da könnte man in Zukunft eine «Pagination» oder etwas ähnliches bauen, um dies zu verbessern.

Tägliches Arbeitsjournal, Tag 10, 22.03.2024	
Geplante Ziele	
Tätigkeiten und erreichte Ziele	
Aufgetretene Probleme	
Beanspruchte Hilfestellungen	
Vergleich mit dem Zeitplan/ Überzeiten	
Ungeplante Arbeiten	
Reflexion	

12 Dailies

Es wird jeweils täglich ein Daily durchgeführt mit der VF. Das erste Daily konnte leider nicht umgesetzt werden, daher ist der Start erst ab dem zweiten Tag.

Tägliches Daily, Tag 2, 08.03.2024	
Was wurde heute besprochen	Ich habe heute mit der VF meine Arbeiten von gestern besprochen und die Arbeiten, welche ich heute geplant habe. Das Vorgehen wurde mit der VF angeschaut.
Aufgetauchte Fragen	Ich hatte noch Unklarheiten bezüglich der Daten, bei den Standardwerten und bei den Erläuterungstexten. Da dar mir nicht klar wie ich diese genau filtern und anzeigen soll. Dieses Problem konnte zusammen besprochen und gelöst werden. Siehe für Notizen vom Daily Anhang: **Abbildung 52, Daily 08.03.24**

Tägliches Daily, Tag 3, 11.03.2024	
Was wurde heute besprochen	Ich habe heute mit der VF meine Arbeiten für Heute besprochen und abgeklärt ob die Anforderungen richtig verstanden wurden.
Aufgetauchte Fragen	Für einen Editor zur Anpassung von Erläuterungstexten war ich mir ein wenig unsicher. Da habe ich die Umsetzung kurz mit der VF besprochen. Wir haben die Option diesen selbst umzusetzen, dafür müssten wir das zuerst evaluieren. Sollte dies den Aufwand überschreiten, müssten wir auf eine externe Komponente setzen.

Tägliches Daily, Tag 4, 13.03.2024	
Was wurde heute besprochen	Ich habe heute mit der VF meine Arbeiten für Heute besprochen und abgeklärt ob die Anforderungen richtig verstanden wurden.
Aufgetauchte Fragen	Wie sollen die Endpunkte genau aussehen? Müssen die Endpunkte genau gleich sein wie in der Aufgabenstellung oder habe ich da Spielraum? Mit der VF wurde abgemacht das die Endpunkte nicht 1 zu 1 wie vorgegeben umgesetzt werden müssen und ich da eigentlich Spielraum habe.

Tägliches Daily, Tag 5, 14.03.2024	
Was wurde heute besprochen	Ich habe heute mit der VF meine Arbeiten für Heute besprochen und abgeklärt ob die Anforderungen richtig verstanden wurden.
Aufgetauchte Fragen	Heute war es ein kurzes Daily. Es gab keine Fragen die gross mit der VF besprochen werden mussten.

Tägliches Daily, Tag 6, 15.03.2024	
Was wurde heute besprochen	Ich habe heute mit der VF meine Arbeiten für Heute besprochen und abgeklärt ob die Anforderungen richtig verstanden wurden.
Aufgetauchte Fragen	Heute war es ein kurzes Daily. Es gab keine Fragen die gross mit der VF besprochen werden mussten.

Tägliches Daily, Tag 7, 18.03.2024	
Was wurde heute besprochen	Ich habe heute mit der VF meine Arbeiten für Heute besprochen und abgeklärt ob die Anforderungen richtig verstanden wurden.
Aufgetauchte Fragen	Das Handling für den User für das Abspeichern von Daten war unklar, dies wurde mit der VF abgesprochen und geklärt.

Tägliches Daily, Tag 8, 20.03.2024	
Was wurde heute besprochen	Ich habe heute mit der VF meine Arbeiten für Heute besprochen und abgeklärt ob die Anforderungen richtig verstanden wurden.
Aufgetauchte Fragen	Schriftarten Problem wurde mit der VF besprochen konnten jedoch nicht direkt eine Lösung finden.

Tägliches Daily, Tag 9, 21.03.2024	
Was wurde heute besprochen	Ich habe heute mit der VF meine Arbeiten für Heute besprochen und abgeklärt ob die Anforderungen richtig verstanden wurden. Das Projekt wurde zusammen angeschaut und offene Punkte besprochen
Aufgetauchte Fragen	--

Tägliches Daily, Tag 10, 22.03.2024	
Was wurde heute besprochen	
Aufgetauchte Fragen	

13 Azure DevOps Board

Sobald der Coding-Teil beginnt, wird täglich ein Screenshot vom Azure DevOps Board gemacht. Die Priorität liegt jeweils immer beim obersten PBI.

08.03.24:

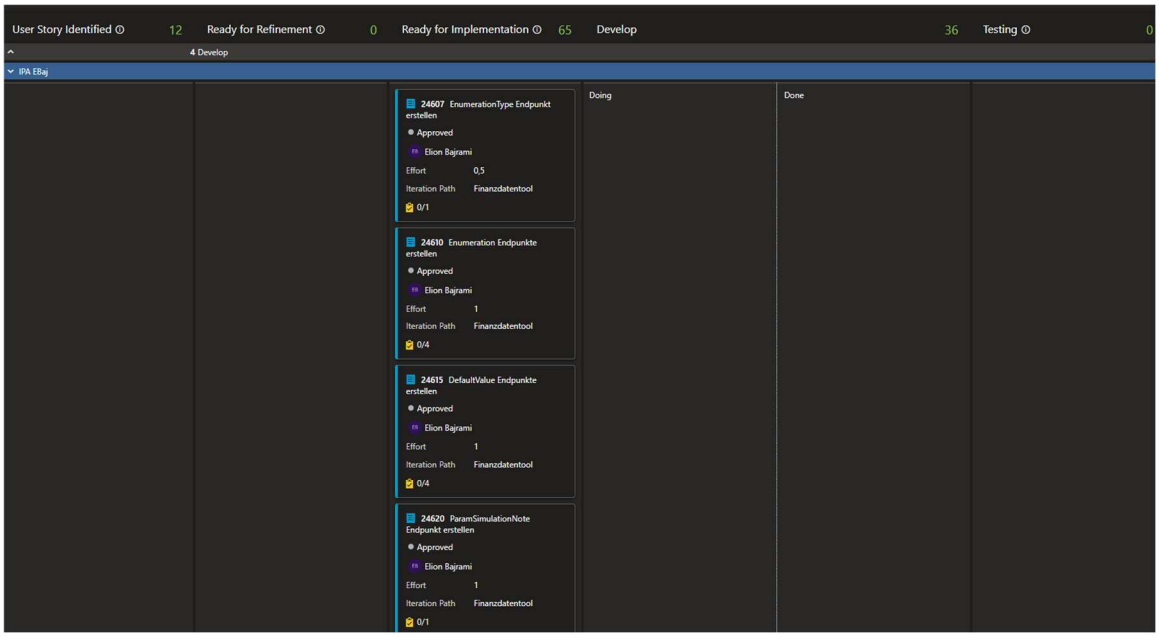


Abbildung 5, Board 01

13.03.24:

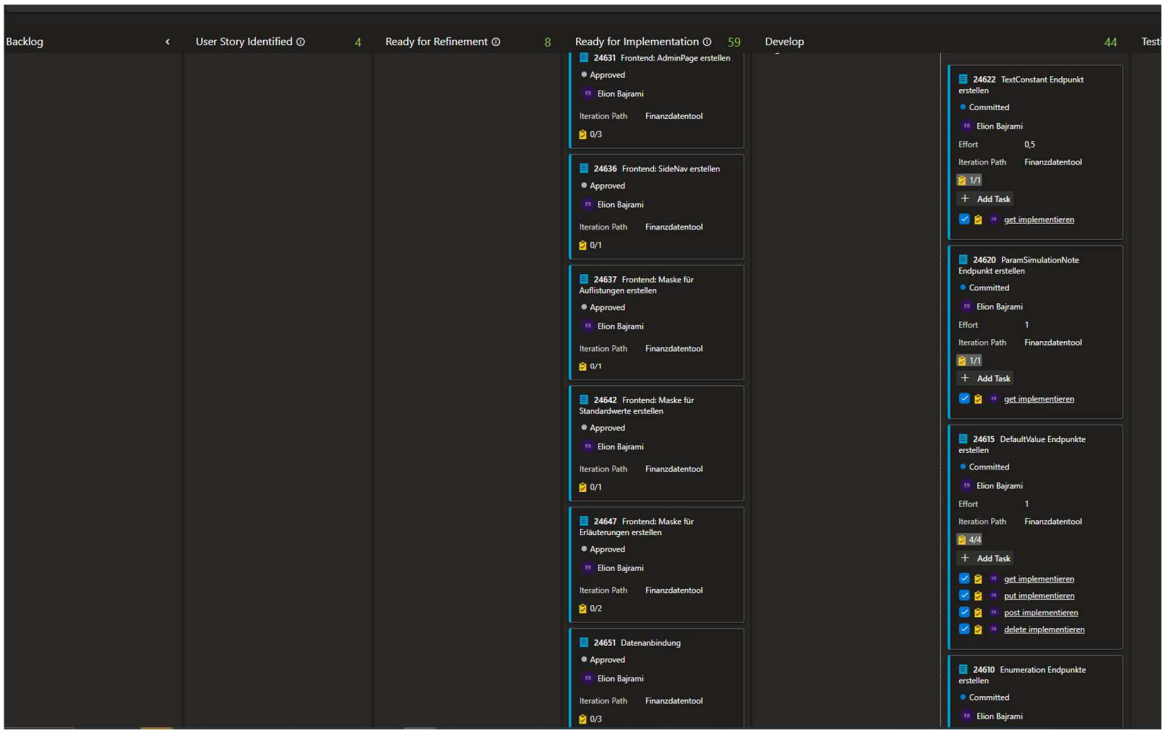


Abbildung 6, Board 02

14.03.24:

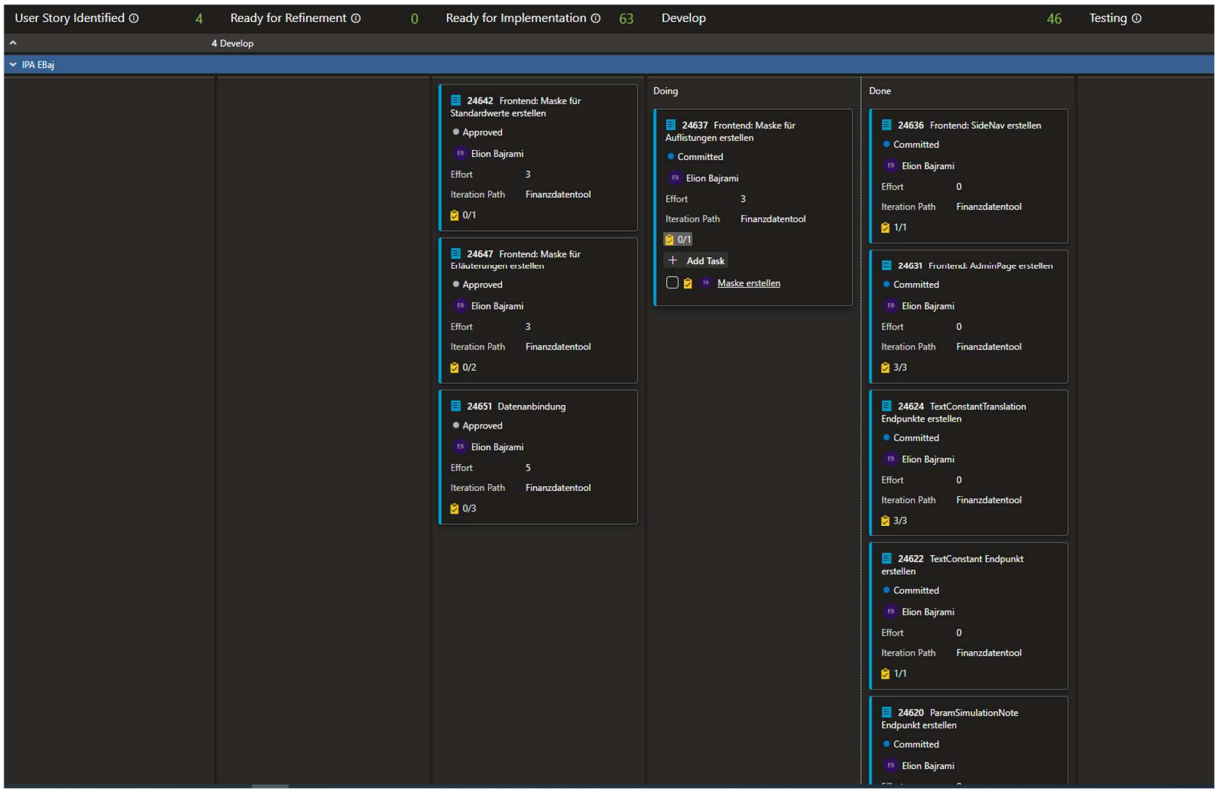


Abbildung 7, Board 03

15.03.24:

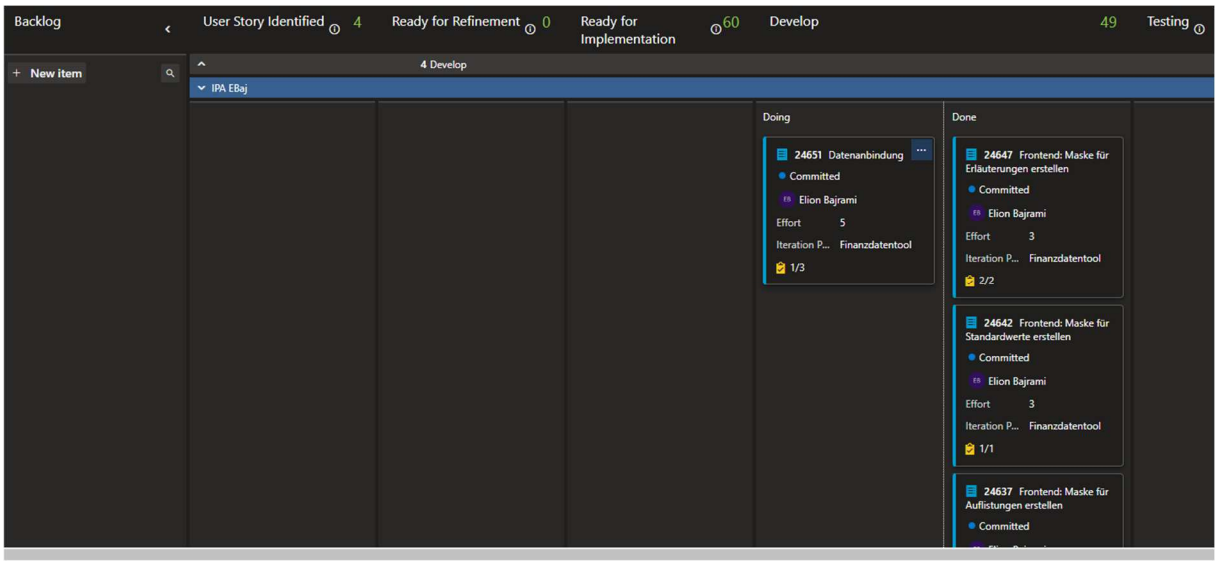


Abbildung 8, Board 04

18.03.24:

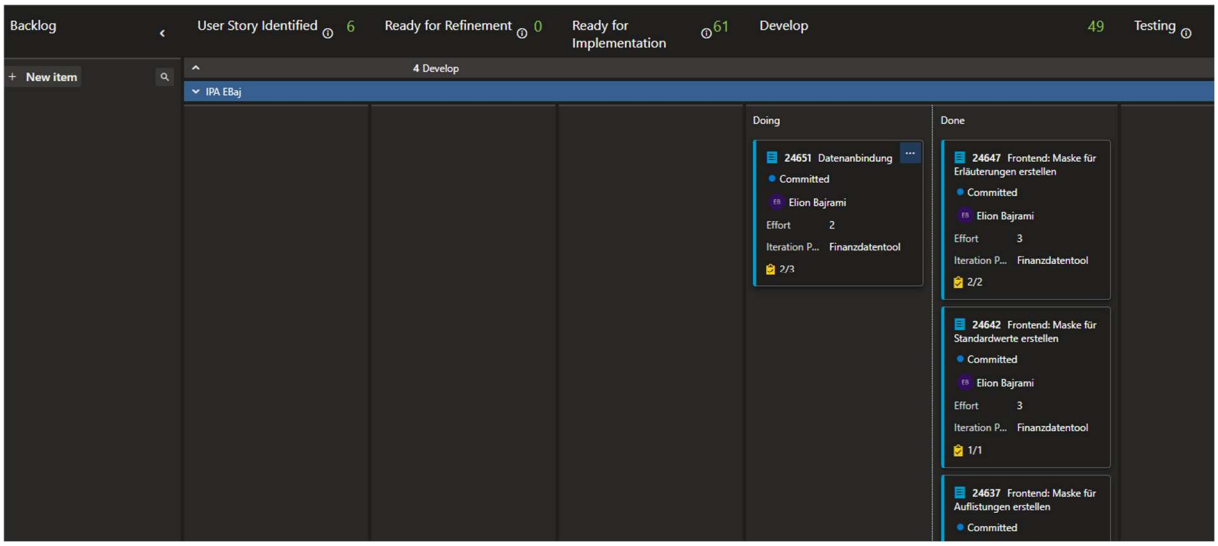


Abbildung 9, Board 05

20.03.24:

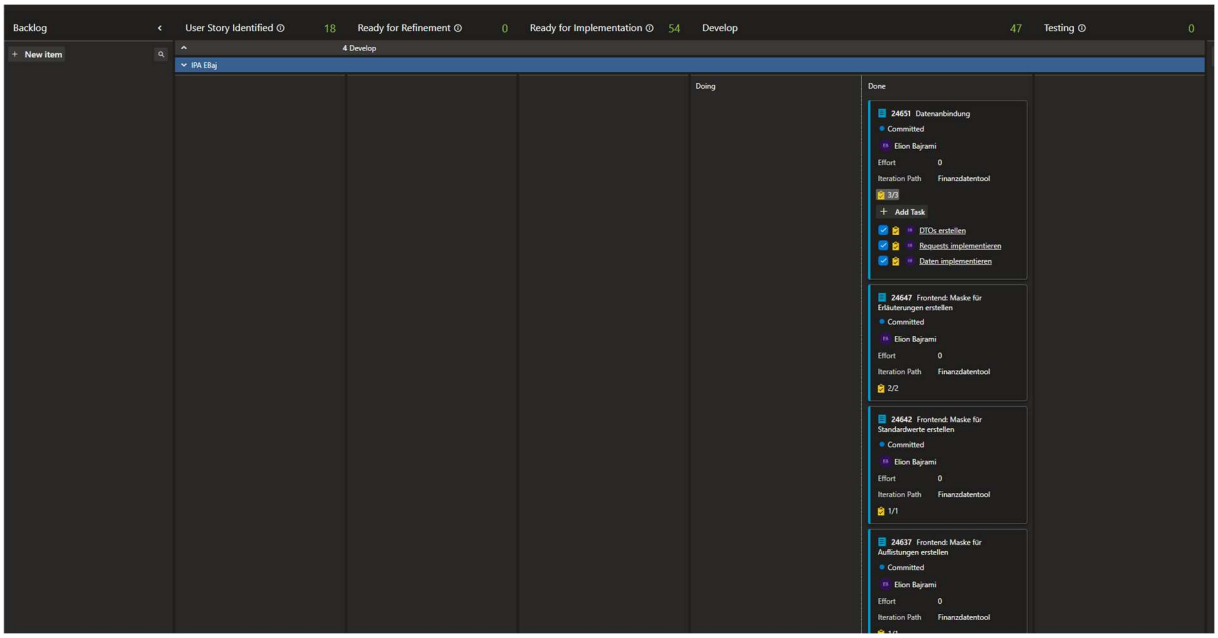


Abbildung 10, Board 06

Teil 2 – Projektdokumentation

14 Informieren

Die erste Phase der IPERKA-Methode ist Informieren. In diesem Schritt geht es darum, herauszufinden, worum es im Projekt genau geht. Was genau ist der Auftrag? Haben wir alle nötigen Informationen?

14.1 Datenbank

Beispiele wie die bestehenden Daten aussehen und geholt werden können.

1. **Auflistungen**
   Query (Enumerations gefiltert nach einem Typ):
   SELECT sysText AS Beschreibung, ShortText AS Abkürzung, ProgramCode, SortOrder AS Sortierung FROM Enumeration WHERE EnumerationTypeID = 7

   Daten:

Results | Messages

	Beschreibung	Abkürzung	ProgramCode	Sortierung
1	Deutsch	NULL	de_ch	1
2	Englisch	NULL	en_gb	1
3	Französisch	NULL	fr_ch	2
4	Italienisch	NULL	it_ch	1

Abbildung 11, Daten Auflistungen

2. **Standardwerte**
   Query (Standardwerte gefiltert nach einem Typ):
   SELECT StartYear AS Startjahr, ValueDecimal AS Wert FROM DefaultValue WHERE DefaultValueTypeEnumerationID = 60

   Daten:

	Startjahr	Wert
1	2000	0.015000

Abbildung 12, Daten Standardwerte

3. **Erläuterungstexte**
   Query (Erläuterungstext gefiltert nach Kategorie und Subgategorie):
   SELECT tctHeader.sysText AS HeaderText, tctBody.sysText AS BodyText FROM ParamSimulationNote psm
   JOIN TextConstant tcHeader ON psm.HeaderTextConstantID = tcHeader.TextConstantID
   JOIN TextConstantTranslation tctHeader ON tctHeader.TextConstantID = tcHeader.TextConstantID
   JOIN TextConstant tcBody ON psm.BodyTextConstantID = tcBody.TextConstantID
   JOIN TextConstantTranslation tctBody ON tctBody.TextConstantID = tcBody.TextConstantID
   WHERE CategoryEnumerationID = 3846 AND SubcategoryEnumerationID = 3865

Daten:

	HeaderText	BodyText
1	AHV-Beiträge für Nichterwerbstätige	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
2	AHV-Beiträge für Nichterwerbstätige	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
3	AHV-Beiträge für Nichterwerbstätige	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
4	AHV-Beiträge für Nichterwerbstätige	{\rtf1\ansi\deff0\nouicompat{\fonttbl{\f0\fnil Segoe UI;}}{\col...
5	AHV contributions for non-employed persons	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
6	AHV contributions for non-employed persons	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
7	AHV contributions for non-employed persons	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
8	AHV contributions for non-employed persons	{\rtf1\ansi\deff0\nouicompat{\fonttbl{\f0\fnil Segoe UI;}}{\col...
9	Contributi AVS per persone senza attività lucrativa	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
10	Contributi AVS per persone senza attività lucrativa	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
11	Contributi AVS per persone senza attività lucrativa	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
12	Contributi AVS per persone senza attività lucrativa	{\rtf1\ansi\deff0\nouicompat{\fonttbl{\f0\fnil Segoe UI;}}{\col...
13	Cotisations AVS pour personnes sans activité luc...	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
14	Cotisations AVS pour personnes sans activité luc...	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
15	Cotisations AVS pour personnes sans activité luc...	{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang2055{\font...
16	Cotisations AVS pour personnes sans activité luc...	{\rtf1\ansi\deff0\nouicompat{\fonttbl{\f0\fnil Segoe UI;}}{\col...

Abbildung 13, Daten Erläuterungstexte

14.2 Projektanforderungen

- Der Benutzer kann Enumerations und Standardwerte bearbeiten, erstellen und löschen.
- Der Benutzer sieht eine sortierte (nach Kategorien) Ansicht in einem SideNav.
- Der Benutzer kann Erläuterungstexte einfärben, dekorieren (kursiv, unterstreichen, etc.) und auch bearbeiten.
- Der Benutzer erhält immer passendes Feedback.

14.3 Projektumfeld: Systemgrenzen / Schnittstellen zur Aussenwelt

In diesem Projekt wird auf keine externen Schnittstellen zugegriffen. Es besteht also lediglich ein internes Frontend welches auch auf eine interne API zugreift. Die API verarbeitet dann die Daten in der Datenbank.

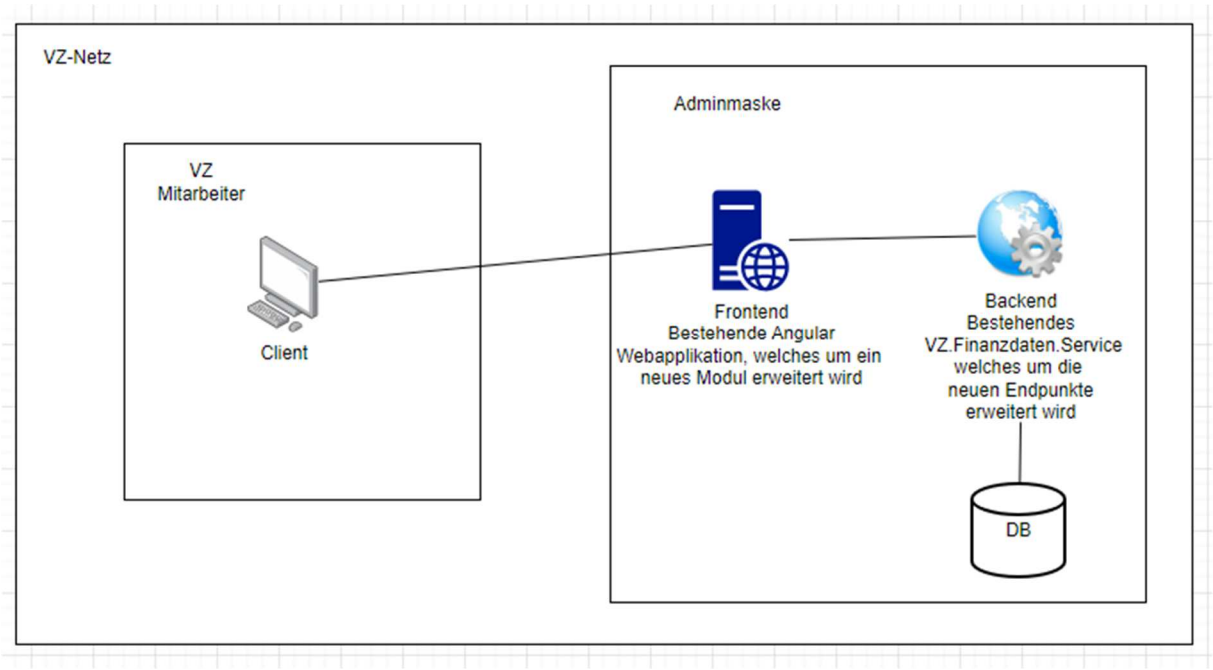


Abbildung 14, Projektumfeld

15 Planen

Das Planen ist die zweite Phase der IPERKA-Methode.

15.1 Realisierungskonzept

Das bestehende Frontend-Projekt wird um ein zusätzliches Modul namens "admin" erweitert. Die Admin-Oberfläche bietet Optionen zur Anzeige von Auflistungen, Standardwerten und Erläuterungstexten. Ein SideNav-Menü zeigt die verschiedenen Kategorien an. Wenn eine Kategorie ausgewählt wird, erscheinen die zugehörigen Daten in einer Bearbeitungsmaske rechts davon. Die genaue Darstellung der Elemente kann in den bereitgestellten Mockups eingesehen werden.

Das Backend-Projekt, das bereits existiert, wird ebenfalls erweitert, um die neuen Endpunkte für die Admin-Oberfläche zu unterstützen. Das Backend interagiert mit der Datenbank über das Entity-Framework, um Datenänderungen vorzunehmen.

15.1.1 Mockups

Mit Adobe XD wurden die Mockups erstellt, um die Umsetzung zu erleichtern. Diese Mockups sind keine definitive Vorgabe und dienen nur als Orientierung.

PW: IPA_Adminmaske2024

Link: Link zu den Mockups

15.2 Konzeptionelle Umsetzung

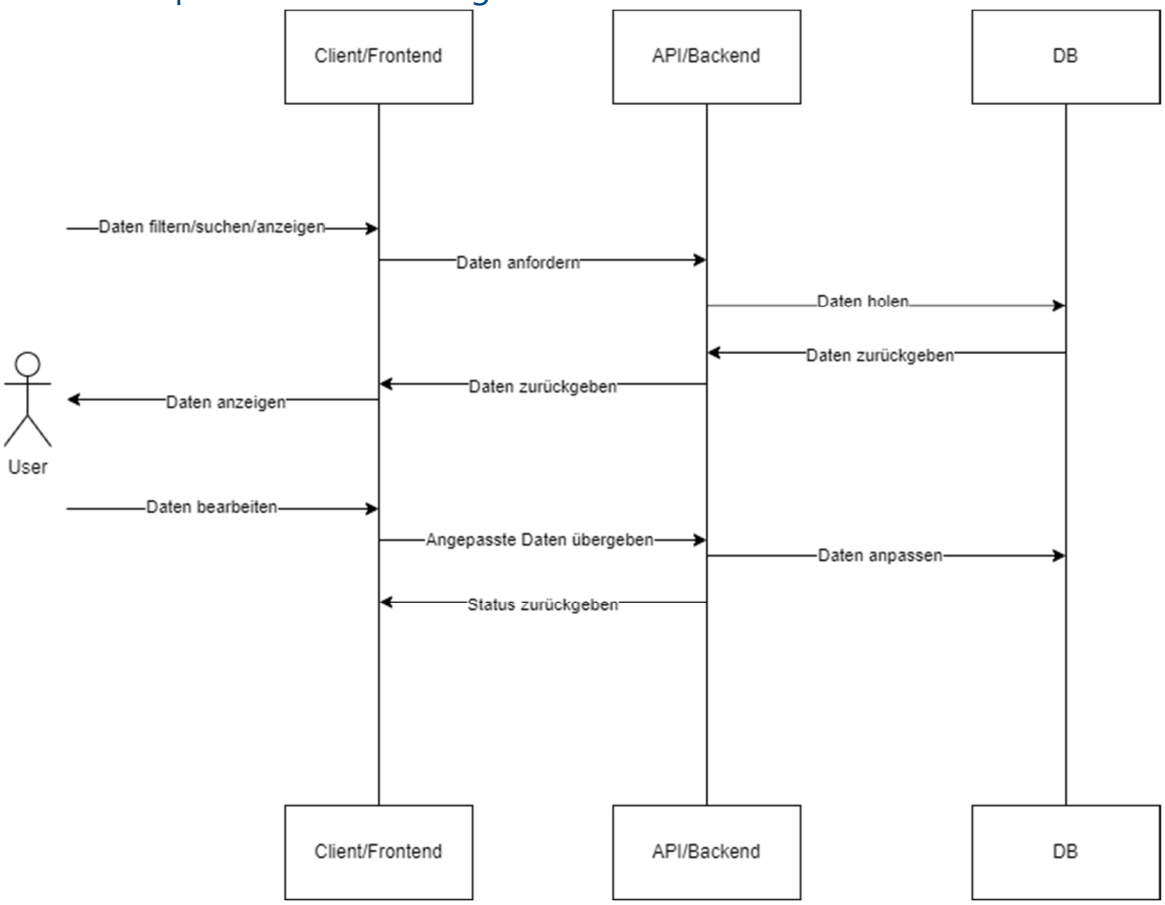


Abbildung 15, Konzeptionelle Umsetzung

15.3 Testkonzept

- Betriebssystem: Windows 10
- Browser: Edge
- Server Umgebung: lokal

Abschnitt	**Inhalt**
ID	T-01
Anforderungen	User kann Webseite öffnen
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite
Erwartetes Resultat	User sollte nun einen Header sehen mit den 3 Texten: Auflistungen, Standardwerte und Erläuterungstexte

Abschnitt	**Inhalt**
ID	T-02
Anforderungen	User kann Auflistungen sehen und filtern
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Auflistungen» im Header 3. User klickt auf einen beliebigen Typen im SideNav 4. User sucht nach beliebigem Wert
Erwartetes Resultat	User sollte die Auflistungen öffnen können und die Daten sollten nach seinem eingegebenen Wert gefiltert werden.

Abschnitt	**Inhalt**
ID	T-03
Anforderungen	User kann Auflistungen bearbeiten
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Auflistungen» im Header 3. User klickt auf einen beliebigen Typen im SideNav 4. User bearbeitet für eine Auflistung/Enumeration die Felder Beschreibung, Abkürzung, ProgramCode und Sortierung 5. User klickt den Button «Speichern»
Erwartetes Resultat	Die Daten, die durch den User eingegeben wurden, sind gespeichert und auch nach einem refresh der Webseite vorhanden.

Abschnitt	**Inhalt**
ID	T-04
Anforderungen	User kann Auflistungen erstellen
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Auflistungen» im Header 3. User klickt auf einen beliebigen Typen im SideNav 4. User klickt auf das Icon«+» 5. Eine neue und leere Reihe wurde erstellt 6. Der User kann in allen Feldern seine Daten eingeben 7. Der User klickt auf den Button «Erstellen»
Erwartetes Resultat	Die neue Auflistung/Enumeration ist nun in der Liste ersichtlich, dies auch nach einem refresh der Webseite. (Beachte Sortierung !!!)

Abschnitt	**Inhalt**
ID	T-05
Anforderungen	User kann Auflistungen löschen
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Auflistungen» im Header 3. User klickt auf einen beliebigen Typen im SideNav 4. User klickt auf den Button/Icon mit einem Mülleimer
Erwartetes Resultat	Der Eintrag wurde gelöscht und ist nicht mehr ersichtlich auch nach einem refresh der Webseite. Es kann aber auch sein das man diesen Eintrag nicht löschen darf, dann wird eine Fehlermeldung angezeigt.

Abschnitt	**Inhalt**
ID	T-06
Anforderungen	User kann Standardwerte sehen
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Standardwerte» im Header 3. User klickt auf einen beliebigen Typen im SideNav
Erwartetes Resultat	Daten werden geladen und der User kann diese rechts vom SideNav nun sehen

Abschnitt	**Inhalt**
ID	T-07
Anforderungen	User kann Standardwerte bearbeiten
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Standardwerte» im Header 3. User klickt auf einen beliebigen Typen im SideNav 4. Der User bearbeitet einen Standardwert, d.h. er bearbeitet die Felder Startjahr und Wert 5. User klickt auf den Button «Speichern»
Erwartetes Resultat	Die Daten, die durch den User eingegeben wurden, sind gespeichert und auch nach einem refresh der Webseite vorhanden.

Abschnitt	**Inhalt**
ID	T-08
Anforderungen	User kann Standardwerte erstellen
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Standardwerte» im Header 3. User klickt auf einen beliebigen Typen im SideNav 4. Der User klickt auf den das Icon «+» 5. Eine neue und leere Reihe wurde erstellt 6. Der User kann in allen Feldern seine Daten eingeben 7. Der User klickt auf den Button «Erstellen»
Erwartetes Resultat	Der neue Standardwert ist nun in der Liste ersichtlich, dies auch nach einem refresh der Webseite.

Abschnitt	**Inhalt**
ID	T-09
Anforderungen	User kann Standardwerte löschen
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Standardwerte» im Header 3. User klickt auf einen beliebigen Typen im SideNav 4. User klickt auf den Button/Icon mit einem Mülleimer
Erwartetes Resultat	Der Eintrag wurde gelöscht und ist nicht mehr ersichtlich auch nach einem refresh der Webseite

Abschnitt	**Inhalt**
ID	T-10
Anforderungen	User kann den Kategorie Titel der Erläuterungstexte sehen und bearbeiten
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Erläuterungstexte» im Header 3. User klickt auf einen beliebigen **Ober-**Typen im SideNav 4. User bearbeitet nun den Kategorie Titel 5. Der User klickt auf den Button «Speichern»
Erwartetes Resultat	Der Eintrag wurde gespeichert und ist auch nach einem refresh noch ersichtlich (Titel ohne Markierungen und Dekorierungen)

Abschnitt	**Inhalt**
ID	T-11
Anforderungen	User kann den Header Titel und Body Text der Erläuterungstexte sehen und bearbeiten (färben, dekorieren und Text anpassen)
Vorbedingungen	User kann Webseite öffnen und Daten können geladen werden
Ablauf	1. User navigiert auf Admin Seite 2. User klickt auf «Erläuterungstexte» im Header 3. User klickt auf einen beliebigen **Ober**-Typen im SideNav 4. User klickt auf einen beliebigen **Unter**-Typen im SideNav 5. User sieht Header Titel und Body Text in allen 4 Sprachen 6. User bearbeiten die Texte beliebig 7. Der User klickt auf den Button «Speichern»
Erwartetes Resultat	Die Anpassungen der Texte wurden gespeichert und sie sind auch nach einem refresh der Webseite noch vorhanden. (Titel ohne Markierungen und Dekorierungen, Body sollte es aber beibehalten)

16 Entscheiden

Das Entscheiden ist die dritte Phase der IPERKA-Methode.

16.1 Editor

Die Erstellung eines eigenen Rich Text Editors in Angular ohne externe Bibliotheken oder Komponenten besteht aus mehreren Schritten:

1. Zuerst muss man eine neue Angular-Komponente erstellen, die als Texteditor dienen wird.
2. Dann muss man in der HTML-Datei der Komponente z.B. eine Text Area erstellen, welches als Texteingabefeld dient. Da gäbe es aber auch verschiedene Möglichkeiten.
3. Nun beginnt man mit der Implementierung der Formatierungsfunktionen. Darunter müssten verschiedene Funktionen zur Verfügung stehen, um Befehle wie 'bold', 'italic', 'underline' usw. ausführen zu können.
4. Diese Funktionen bindet man dann an Klickereignisse in Ihrer HTML-Datei. Dies kann man erreichen, indem man Schaltflächen für jede Funktion erstellt und das Klickereignis an die entsprechende Funktion bindet.

Die Dauer der Erstellung eines solchen Editors hängt stark von der Komplexität der gewünschten Funktionen und der Erfahrung des Entwicklers ab. Ein einfacher Editor mit grundlegenden Funktionen könnte in ein paar Tagen erstellt werden, während ein komplexerer Editor mehrere Wochen oder sogar Monate in Anspruch nehmen könnte. In unserem Fall benötigen wir folgenden Funktionen: färben und dekorieren (**bold**, underline und *kursiv*). Die Umsetzung könnte ein paar Tage dauern. Da wir dieses Projekt aber als IPA durchführen, ist dies zu lange und ich muss auf eine externe Komponente setzen.

16.1.1 Entscheidungsmatrix

Nach einer Recherche bin ich auf 2 Libraries gestossen die für eine Verwendung in Frage kommen würden. Um nun eine Entscheidung zu treffen, wurde eine Entscheidungsmatrix erstellt:

Tabelle 1, Entscheidungsmatrix

Entscheidungsmatrix ngx-quill VS TinyMCE		**ngx-quill**		**TinyMCE**	
Kriterium	**Gewichtung**	**Bewertung ***	**Total**	**Bewertung ***	**Total**
Integration mit Angular	2	10	**20**	8	**16**
Dokumentation	2	8	**16**	10	**20**
Anpassungsfähigkeit	1	10	**10**	8	**8**
Funktionen	1	6	**6**	8	**8**
Leistung	1	10	**10**	8	**8**
Lizenz (OpenSource? Kosten?)	4	10	**40**	6	**24**
Total			**102**		**84**
Entscheid			**1**		**2**

* Bewertung auf einer Skala von 1 bis 10:
1 = wirkt sehr negativ auf das Kriterium
10 = wirkt sehr positiv auf das Kriterium

16.2 RTF-Text

Nachdem wir uns für den Einsatz des ngx-quill Editors entschieden haben, steht nun die wichtige Entscheidung an, wie wir den RTF-Text in diesem Editor übergeben und darstellen. ngx-quill bietet uns hierfür verschiedene Möglichkeiten an: HTML, JSON, Object und Text.

Abbildung 16, Editor Formate

Wir können jedoch die Optionen 'Object' und 'Text' direkt ausschließen. Der Grund dafür ist, dass wir bei der Verwendung von 'Text' sämtliche Formatierungen des RTF-Textes verlieren würden, was für unsere Zwecke nicht akzeptabel ist. Bei der 'Object'-Option stösst man auf das Problem, dass uns kein passender Converter zur Verfügung steht, der eine effiziente und zuverlässige Konvertierung gewährleisten könnte.

Dies lässt uns mit den Optionen 'HTML' und 'JSON' übrig. Bei der Wahl zwischen diesen beiden müssen wir berücksichtigen, dass wir in der Lage sein müssen, den RTF-Text sowohl in das gewählte Format zu konvertieren als auch aus diesem Format zurück in RTF zu konvertieren. Das Programmieren einer solchen Konvertierungsfunktion von Grund auf wäre leider sehr zeitaufwendig und könnte potenziell zu Fehlern führen.

In unserem Backend-System verwenden wir ein Nuget-Paket namens "RtfPipe" zur Konvertierung von Rich Text Format (RTF) in Hypertext Markup Language (HTML). Dieses Tool ermöglicht es uns, die Konvertierung von RTF zu HTML effizient und ohne großen Aufwand durchzuführen.

Für die umgekehrte Konvertierung, also von HTML zu RTF, setzen wir auf das Nuget-Paket "SautinSoft". Es ist dringend zu beachten, dass "SautinSoft" kostenpflichtig ist. Allerdings bietet es eine Testversion an, die wir für unseren aktuellen Bedarf nutzen können.

Sollte die weitere Nutzung von "SautinSoft" in Betracht gezogen werden, besteht die Möglichkeit, eine Lizenz zu erwerben. Dies würde uns den dauerhaften Zugang zu den Funktionen des Tools sichern und uns ermöglichen, unsere Konvertierungsprozesse auch in Zukunft effizient zu gestalten.

Eine andere Option wäre die RTF-Texte komplett in allen Projekten durch HTML-Texte zu ersetzen, da RTF-Texte auch schon älter sind und nicht immer überall unterstützt werden. Dann bräuchte man in Zukunft keine Konvertierung mehr.

17 Realisieren

In der vierten Phase der IPERKA-Methode haben wir Realisieren.

17.1 Branches

Im Azure DevOps wurde wie geplant ein IPA-Haupt-Branch erstellt. Von diesem Branch werden die einzelnen Feature-Branches erstellt. Sobald alle Features durch einen Pull-Request in den IPA-Haupt-Branch gemerged wurden und die IPA fertig ist, wird dieser Branch dann in den DEV(Haupt)-Branch gemerged.

17.1.1 Backend

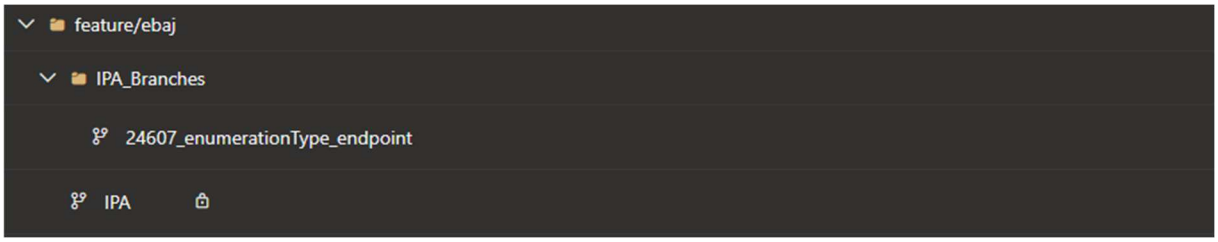


Abbildung 17, Branches Backend

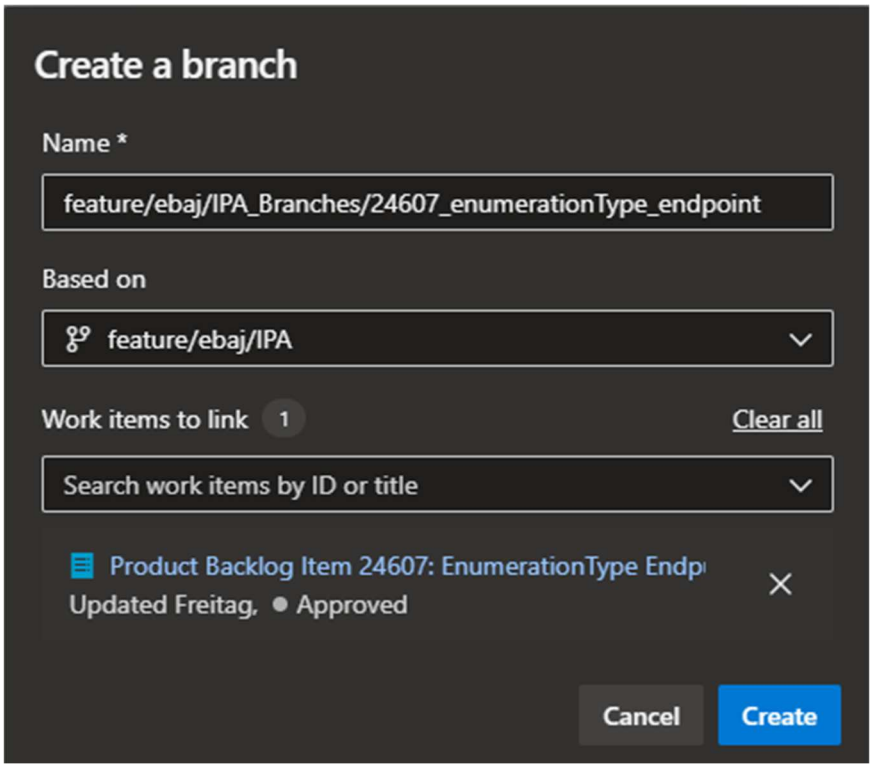


Abbildung 18, Branch BasedOn Backend

17.1.2 Frontend

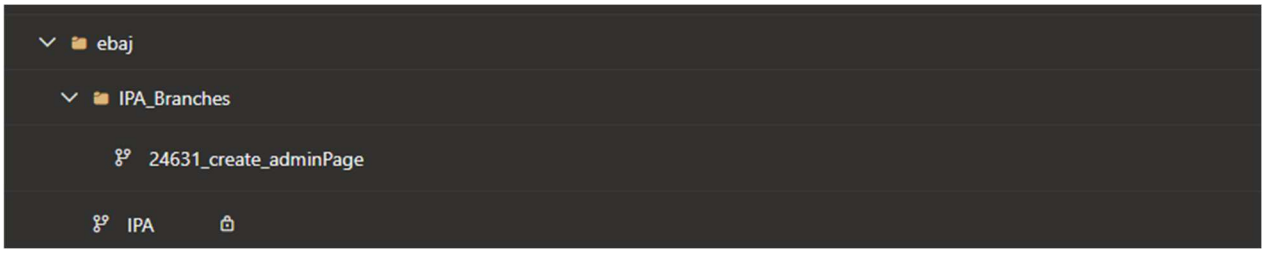


Abbildung 19, Branches Frontend

17.2 Backend

17.2.1 Endpunkte

Die neuen Endpunkte, welche für die Adminmakse benötigt werden, wurden in einem bestehenden Controller erstellt («ParamController»). Es wurden nur die nötigsten Endpunkte erstellt, damit die entsprechenden Werte geladen und angepasst werden können. Wo nötig wurden neue DTOs erstellt (im untenstehenden Bild markiert). Beim Enumeration DTO wurden die nötigsten Daten aus EnumerationEntry übernommen. Um Duplikate zu vermeiden erbt EnumerationEntry neu von Enumeration. Somit kann duplizierter Code vermieden werden.

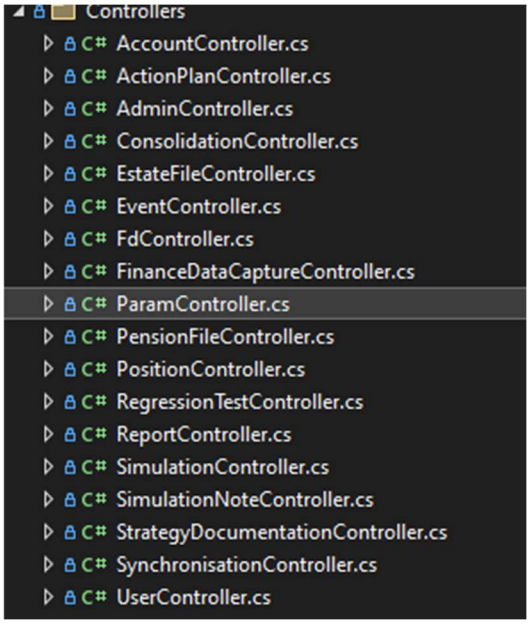


Abbildung 21, ParamController

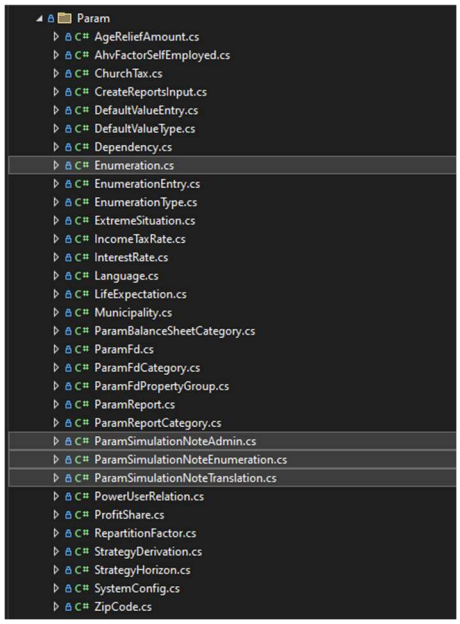


Abbildung 20, DTOs Backend

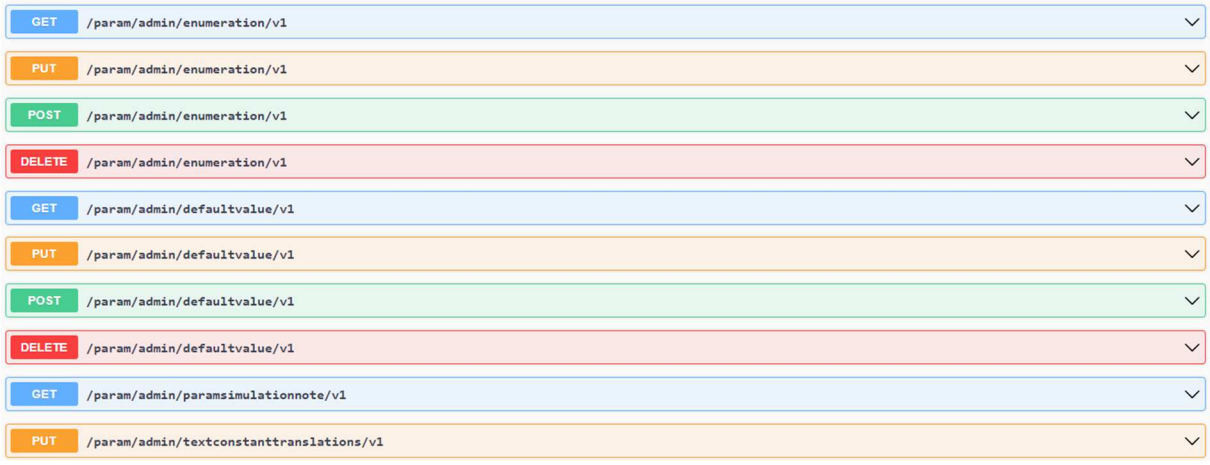


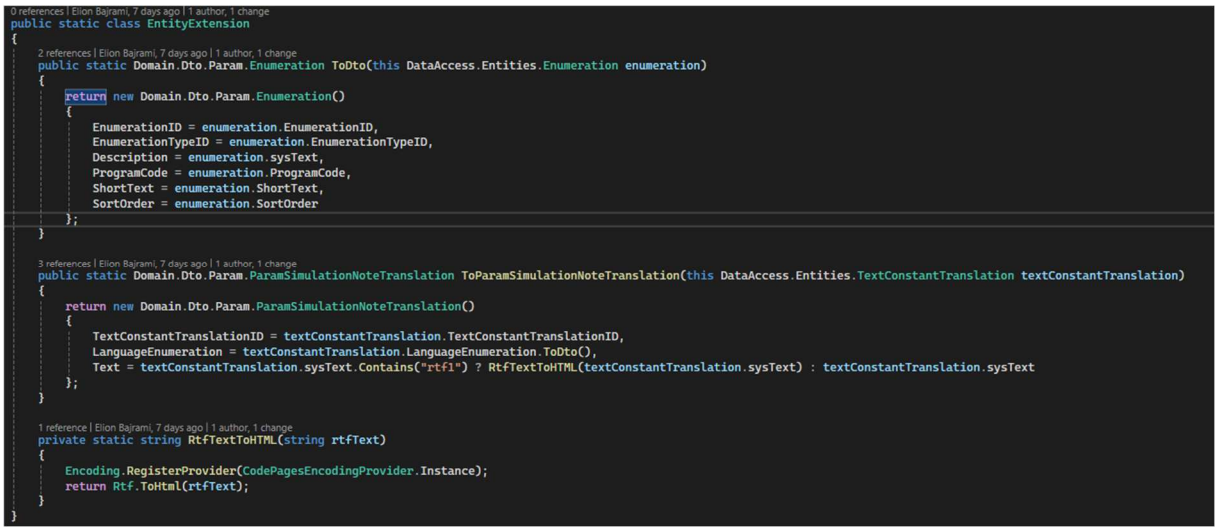
Abbildung 22, Endpoints

17.2.2 Service

Im bestehendem «ParamService» wurde jeweils das Mapping gemacht zwischen den Entitäten und den DTOs. Der bestehende ParamService hat ein Interface namens «IParamInterface» darin wurden jeweils die Funktionen deklariert, um sie dann im Controller per DependencyInjection auch aufrufen zu können.

Für die komplizierteren Mappings habe ich eine «EntityExtension» erstellt. Diese Extension führt dann das Mapping aus für die entsprechenden Entitäten.

Die Texte, welche im RTF-Format sind, werden ebenfalls hier mit der Hilfe eines Packages zu HTML-Format konvertiert.

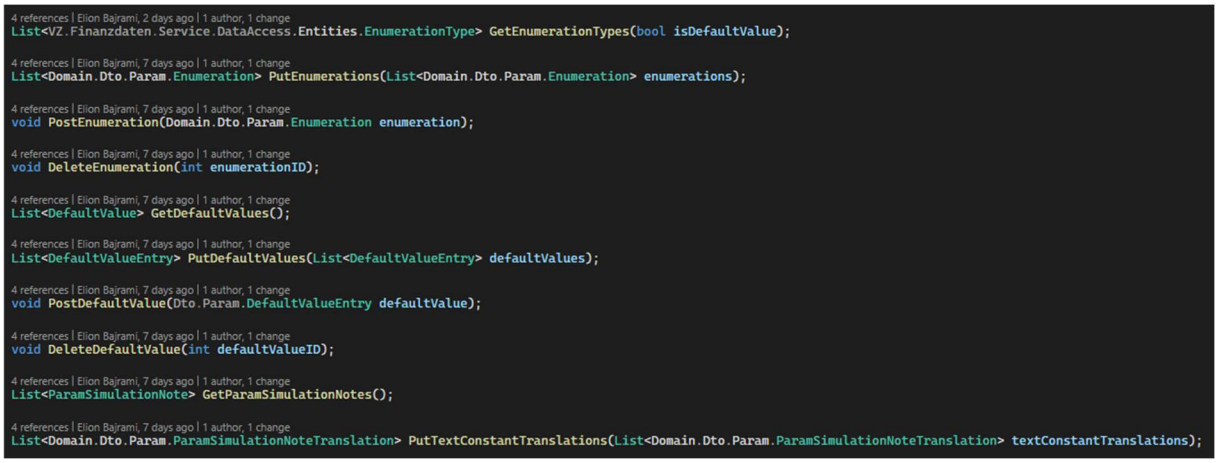


```
0 references | Elion Bajrami, 7 days ago | 1 author, 1 change
public static class EntityExtension
{
    2 references | Elion Bajrami, 7 days ago | 1 author, 1 change
    public static Domain.Dto.Param.Enumeration ToDto(this DataAccess.Entities.Enumeration enumeration)
    {
        return new Domain.Dto.Param.Enumeration()
        {
            EnumerationID = enumeration.EnumerationID,
            EnumerationTypeID = enumeration.EnumerationTypeID,
            Description = enumeration.sysText,
            ProgramCode = enumeration.ProgramCode,
            ShortText = enumeration.ShortText,
            SortOrder = enumeration.SortOrder
        };
    }

    3 references | Elion Bajrami, 7 days ago | 1 author, 1 change
    public static Domain.Dto.Param.ParamSimulationNoteTranslation ToParamSimulationNoteTranslation(this DataAccess.Entities.TextConstantTranslation textConstantTranslation)
    {
        return new Domain.Dto.Param.ParamSimulationNoteTranslation()
        {
            TextConstantTranslationID = textConstantTranslation.TextConstantTranslationID,
            LanguageEnumeration = textConstantTranslation.LanguageEnumeration.ToDto(),
            Text = textConstantTranslation.sysText.Contains("rtf1") ? RtfTextToHTML(textConstantTranslation.sysText) : textConstantTranslation.sysText
        };
    }

    1 reference | Elion Bajrami, 7 days ago | 1 author, 1 change
    private static string RtfTextToHTML(string rtfText)
    {
        Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);
        return Rtf.ToHtml(rtfText);
    }
}
```

Abbildung 23, EntityExtension

17.2.3 Repository

Im Repository wird auf die Datenbank zugegriffen. Hier werden die benötigten Daten in der Datenbank geladen, angepasst, erstellt oder auch gelöscht. Dafür wurden die benötigten Funktionen im bestehendem Repository erstellt und auch im Interface «IParamRepository» erfasst. Damit der Service darauf zugreifen kann.



```
4 references | Elion Bajrami, 2 days ago | 1 author, 1 change
List<VZ.Finanzdaten.Service.DataAccess.Entities.EnumerationType> GetEnumerationTypes(bool isDefaultValue);

4 references | Elion Bajrami, 7 days ago | 1 author, 1 change
List<Domain.Dto.Param.Enumeration> PutEnumerations(List<Domain.Dto.Param.Enumeration> enumerations);

4 references | Elion Bajrami, 7 days ago | 1 author, 1 change
void PostEnumeration(Domain.Dto.Param.Enumeration enumeration);

4 references | Elion Bajrami, 7 days ago | 1 author, 1 change
void DeleteEnumeration(int enumerationID);

4 references | Elion Bajrami, 7 days ago | 1 author, 1 change
List<DefaultValue> GetDefaultValues();

4 references | Elion Bajrami, 7 days ago | 1 author, 1 change
List<DefaultValueEntry> PutDefaultValues(List<DefaultValueEntry> defaultValues);

4 references | Elion Bajrami, 7 days ago | 1 author, 1 change
void PostDefaultValue(Dto.Param.DefaultValueEntry defaultValue);

4 references | Elion Bajrami, 7 days ago | 1 author, 1 change
void DeleteDefaultValue(int defaultValueID);

4 references | Elion Bajrami, 7 days ago | 1 author, 1 change
List<ParamSimulationNote> GetParamSimulationNotes();

4 references | Elion Bajrami, 7 days ago | 1 author, 1 change
List<Domain.Dto.Param.ParamSimulationNoteTranslation> PutTextConstantTranslations(List<Domain.Dto.Param.ParamSimulationNoteTranslation> textConstantTranslations);
```

Abbildung 24, Repositorys

17.2.4 Packages

Um die Erläuterungstexte editieren zu können wird im Frontend ein Editor(ngx-quill) benötigt. Da es aber für unsere Technologien keine Editoren gibt, welche RTF-Texte unterstützen, müssen wir diesen zuerst zu HTML konvertieren. Wenn wir den Text abspeichern, muss er wieder zu RTF zurück konvertiert werden. Da ein eigener Konverter zu Zeitaufwendig ist für eine IPA, wurde auf externe Packages gesetzt, welche diese Funktionen ausführen können. Das Package «RtfPipe» benötigt zusätzlich das Package System.Text.Encoding.CodePages.

Intern werden alle Applikationen mit einem externen Tool auf Vulnerabilitäten in den Packages überprüft. Da im Package «sautinsoft.htmltortf» eine Version von «System.Private.Uri» installiert war, welche vulnerable ist. Somit kann der Build in der Pipeline nicht erfolgreich durchlaufen, deswegen muss eine sichere Version von «System.Private.Uri» installiert werden, welche die andere Version dann überschreibt/ersetzt. Das gleiche Problem war mit dem Package «System.Text.RegularExpressions», welches auch von «sautinsoft.htmltortf verwendet wurde.

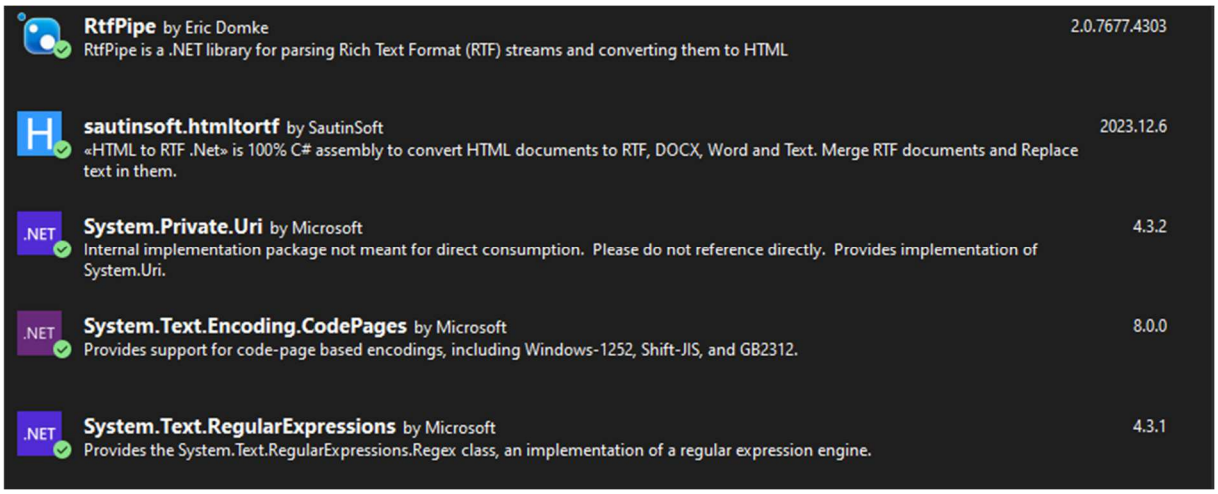


Abbildung 25, Packages

17.2.4.1 RTF/HTML Konverter

Um den RTF-Text zu HTML zu konvertieren, wird das Nuget «RtfPipe» verwendet. Dies geschieht jeweils in der EntityExtension siehe Abbildung: **Abbildung 23, EntityExtension**

Um den HTML wieder zu RTF-Text konvertieren zu können wird das Nuget «sautinsoft.htmltortf» verwendet. Dies passiert jeweils im Service, sobald der Text gespeichert wird.

```
//CONVERT HTML TO RTF
if (textConstantTranslation.Text.Contains(HtmlTextIdentifier))
{
    var stream = new MemoryStream();
    var writer = new StreamWriter(stream);
    writer.Write(textConstantTranslation.Text);
    writer.Flush();
    stream.Position = 0;
    HtmlConvertOptions opt = new HtmlConvertOptions();
    opt.OutputFormat = HtmlToRtf.OutputFormat.Rtf;
    opt.TextSetup = new HtmlToRtf.TextSetup() { DefaultFontFamily = FontFamily, DefaultFontSize = FontSize, SingleFontFamily = FontFamily };
    MemoryStream result = new MemoryStream();

    SautinSoft.HtmlToRtf.SetLicense(SautinSoftLicence);
    SautinSoft.HtmlToRtf h = new HtmlToRtf();

    h.Convert(stream, result, opt);

    result.Position = 0;
    using (StreamReader reader = new StreamReader(result, Encoding.UTF8))
    {
        textConstantTranslation.Text = reader.ReadToEnd();
    }
}
```

Abbildung 26, Converter

17.2.5 Logging & Fehlerbehandlung

Im Repository wird jeweils der Fehler geloggt sollte der Versuch die Daten anzupassen fehlschlagen. Das Logging war im bestehenden Projekt bereits implementiert, somit konnte ich diesen einfach wieder verwenden. Zusätzlich wird der bestehende ErrorHandler «FinanzdatenServiceException» verwendet. Der Fehler wird dann im Frontend entsprechend behandelt.

```
catch (Exception e)
{
    logger.LogError(e, $"Deleting defaultValue with ID: {defaultValueID} failed");
    throw new FinanzdatenServiceException($"DefaultValue mit der ID: {defaultValueID} konnte nicht gelöscht werden", e);
}
```

Abbildung 27, Logging und Fehlerbehandlungs

17.3 Frontend

17.3.1 Modul

Im bestehendem Frontend Projekt wurde ein neues Modul erstellt, welches für die Adminmaske genutzt wird. Dies wurde mit dem Angular-CLI gemacht, mit folgendem Befehl: «ng generate module admin». Zuerst werden alle Komponenten deklariert, welche zu diesem Modul gehören. Dann werden alle benötigen Module, die verwendet werden importiert. Zuletzt wird der Service als «Injectable» deklariert und kann somit in den Komponenten verwendet werden.

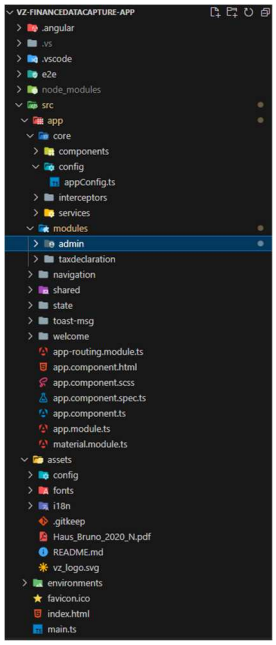


Abbildung 29, Admin Modul Verzeichnis

```
@NgModule({
  declarations: [
    AdminDashboardComponent,
    AdminHeaderComponent,
    AdminSidenavComponent,
    AdminFormComponent
  ],
  imports: [
    CommonModule,
    MatTooltipModule,
    MatTreeModule,
    RouterModule.forChild(admindeclarationRoutes),
    SharedModule,
    StoreModule.forFeature('adminState', adminStateReducer),
    EffectsModule.forFeature([AdminStateEffects]),
    MaterialModule,
    MatProgressSpinnerModule,
    QuillModule.forRoot()
  ],
  exports: [],
  providers: [
    SidenavService,
  ]
})
export class AdminModule { }
```

Abbildung 28, Admin Modul

17.3.2 Service

Der vorhandene Service namens «financedata.service», welcher im Shared Ordner liegt, wird um die neuen Requests erweitert. Diese Request werden an die neuen Endpunkt ins Backend gesendet.

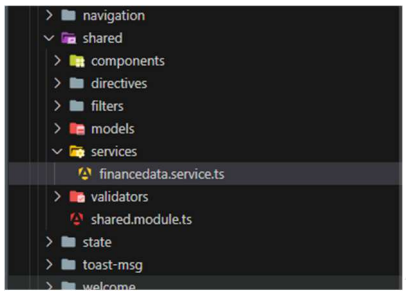


Abbildung 30, financeDataService

```
//PARAM / ADMIN
getEnumerationTypes(isDefaultValue: boolean) {
  return this.http.get<EnumerationType[]>(this.endpointUrl, 'param/admin/enumeration/v1', {
    params: new HttpParams().set('isDefaultValue', isDefaultValue)
  })
    .pipe(catchError(this.http.handleError));
}
```

Abbildung 31, Request Beispiel

Die entsprechenden Models werden im Shared Ordner unter model/admin erstellt. Das Model Enumeration war bereits erstellt, weswegen ich auf den Namen EnumerationCompact ausgewichen bin, denn nur die benötigten Daten sollen transportiert werden.

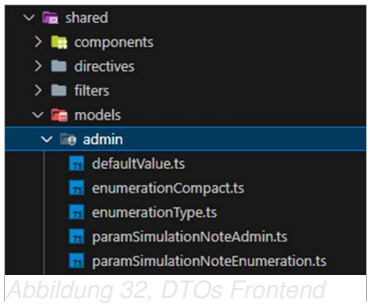


Abbildung 32, DTOs Frontend

17.3.3 State

Im NgRx-Pattern spielen vier «Hauptkomponenten» eine entscheidende Rolle: Actions, Effects, Reducers und Selectors.

Eine "Action" repräsentiert eine Art von Änderung oder eine Operation, die auf den Daten durchgeführt werden soll. Sie ist der Auslöser für jede Änderung im Zustand. Durch das Auslösen einer Action können wir eine bestimmte Operation anstossen.

Der "Effect" ist dafür verantwortlich, den entsprechenden Service aufzurufen, um die Anforderung zu senden. Effects sind im Grunde genommen Listener für verschiedene Actions. Wenn eine bestimmte Action ausgelöst wird, führt der Effect eine oder mehrere Aufgaben aus, wie z.B. das Aufrufen eines Service, um Daten abzurufen.

Der "Reducer" ist die Komponente, die den aktuellen Zustand und die ausgelöste Action entgegennimmt und einen neuen Zustand generiert. Nachdem der Service die Daten abgerufen hat, werden diese im Reducer empfangen und im State gespeichert.

Schließlich ermöglicht der "Selector" den Zugriff auf den State. Mit Selectors können wir Daten aus dem State abrufen und anzeigen. Sie sind Funktionen, die einen Teil des States als Argumente erhalten und daraus die benötigten Daten extrahieren.

Zusammen ermöglichen diese vier «Komponenten» eine effiziente Verwaltung des Zustands in Angular-Anwendungen.

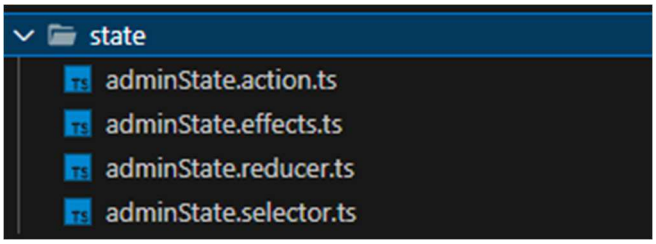


Abbildung 33, AdminState

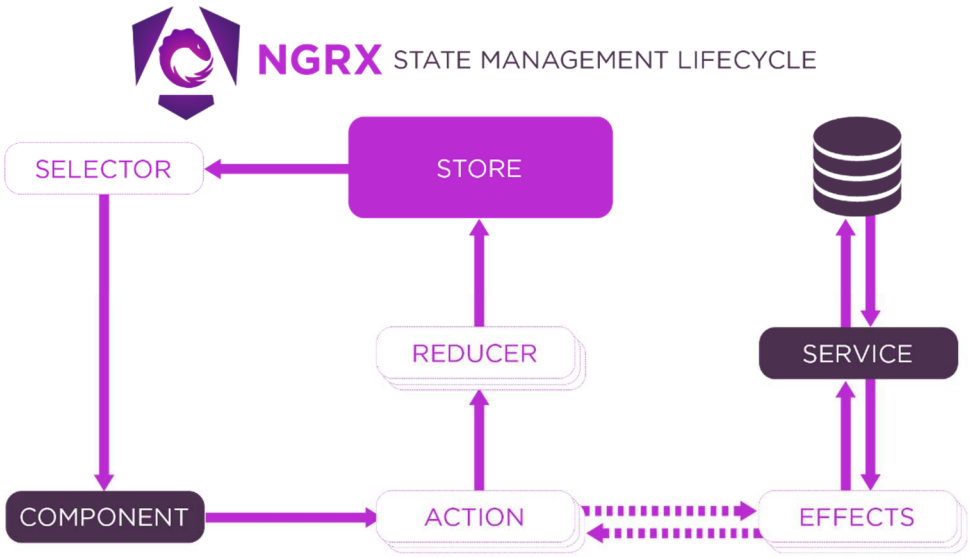


Abbildung 34, NgRx

17.3.4 Components

Folgende Komponente wurden mit dem Befehlt «ng generate component [name]» im Angular-CLI erstellt:

Im Admin Modul:

- Um das «admin-sidenav» von anderen Komponenten zu bedienen wurde zusätzlich ein «sidenav.service» erstellt.

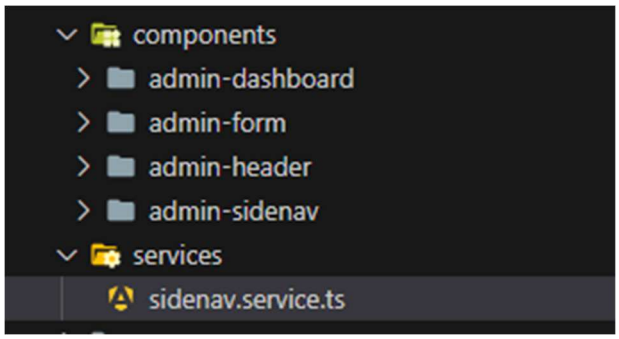


Abbildung 35, Admin Komponenten

Im Shared Modul (nur treeview):

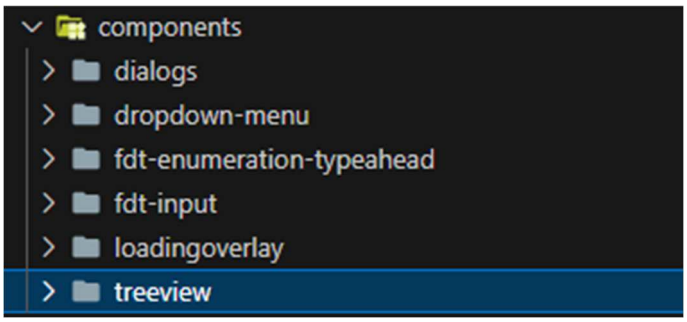


Abbildung 36, Shared Komponenten

Die Komponente «admin-dashboard» beinhaltet zwei Komponenten: «admin-header» und «admin-sidenav».

Die «admin-header»-Komponente ist, wie der Name schon sagt, die Kopfzeile des Admin-Dashboards. Sie enthält die 3 verschiedenen Features, die implementiert wurden: Auflistungen, Standardwerte und Erläuterungstexte.

Die «admin-sidenav»-Komponente ist die Seitenleiste des Admin-Dashboards. Sie bietet eine sekundäre Navigation innerhalb der Daten.

Innerhalb der «admin-sidenav»-Komponente werden zwei weitere Komponenten verwendet: «treeview» und «admin-form».

Die «treeview»-Komponente ist ein interaktives Element, das eine hierarchische Struktur von Informationen darstellt. Sie ermöglicht es den Benutzern, durch eine Baumstruktur von Daten zu navigieren und spezifische Elemente auszuwählen. Diese wurde im Shared Modul erstellt, damit sie in Zukunft wiederverwendet werden kann.

Die «admin-form»-Komponente wird verwendet, um Formulare innerhalb des Admin-Dashboards zu erstellen und zu verwalten. Sie ermöglicht es den Benutzern, Daten einzugeben, bearbeiten und löschen, die dann ans Backend gesendet werden.

Zusammen bilden diese Komponenten die Adminmaske.

17.3.5 ngx-quill

In der Entscheidungsphase habe ich mich für das Package ngx-quill entschieden um den Editor darzustellen. Der Editor wird je nach Texttyp im HTML-Format angepasst und zurückgegeben oder als Plain-Text. Durch die «modules» wird festgelegt, wie die Toolbar im Editor aussehen soll, also was alles erlaubt ist.

```
<quill-editor formControlName="htmlText"
              [format]="row.value.format"
              [modules]="modules"
              [styles]="{width: '500px'}"></quill-editor>
```

Abbildung 37, Quill-Editor

```
modules = {
  toolbar: [
    ['bold', 'italic', 'underline', 'strike'],        // toggled buttons
    [{ 'list': 'ordered' }, { 'list': 'bullet' }],
    [{ 'direction': 'rtl' }],                         // text direction
    [{ 'color': [] }, { 'background': [] }],          // dropdown with defaults from theme
    [{ 'align': [] }],
    ['clean'],                                         // remove formatting button
    ['link']
  ]
};
```

Abbildung 38, Editor Modules

Bis zum Erreichen des AHV-Referenzalters sind Sie bei der AHV beitragspflichtig. Die Beiträge für Nichterwerbstätige berechnen sich jährlich neu aufgrund Ihres Einkommens und Ihres Vermögens.

Frau KUNDENNAME kann Ihre Beitragspflicht mit einem Erwerbseinkommen von CHF XX'000 abdecken, so dass keine Beiträge für Nichterwerbstätige fällig werden.

Abbildung 39, Editor Beispiel

17.3.6 Routing

Man gelangt auf die Adminmaske per «/admin» zum Beispiel http://localhost:4200/admin. In Zukunft muss hier ein Berechtigungskonzept eingeführt werden, denn nicht jeder sollte diese Anpassungen vornehmen dürfen. Das Routing wurde um das neue Admin Modul ergänzt und im Admin Modul wurde dann festgelegt welche Komponente angezeigt werden soll:

```
const routes: Routes = [
  {
    path: '',
    component: WelcomeComponent,
  },
  {
    path: 'toast', component: ToastMsgComponent
  },
  {
    path: 'customer',
    loadChildren: () =>
      import('./modules/taxdeclaration/taxdeclaration.module').then((m) => m.TaxdeclarationModule),
  },
  {
    path: 'admin', loadChildren: () => import('./modules/admin/admin.module').then((m) => m.AdminModule)
  },
  {
    path: '**', component: PageNotFoundComponent
  }
];
You, last week | 2 authors (Thomas Gemmrich and others)
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
export class AppRoutingModule { }
```

Abbildung 41, Routing

```
const admindeclarationRoutes: Routes = [
  {
    path: '',
    component: AdminDashboardComponent,
  },
];
```

Abbildung 42, Routing Pfad

```
imports: [
  CommonModule,
  MatTooltipModule,
  MatTreeModule,
  RouterModule.forChild(admindeclarationRoutes),
  SharedModule,
  StoreModule.forFeature('adminState', adminStateReducer),
  EffectsModule.forFeature([AdminStateEffects]),
  MaterialModule,
  MatProgressSpinnerModule,
  QuillModule.forRoot()
],
```

Abbildung 40, Routing Implementierung

17.3.7 Validierung & Rückmeldung

Damit der User nichts speichern kann was ungültig ist, werden die Eingaben validiert. Sollte der User zum Beispiel aber einen Eintrag vergessen haben, so wird ihm diese Fehlermeldung auch angezeigt und es ist erst möglich zu speichern, wenn alle Eingaben valid sind.

```
<mat-form-field>
    <input matInput
        placeholder="{{'Admin.Description' | translate}}"
        formControlName="description"
        required
        [maxlength]="100">
<mat-error *ngIf="row.get('description').errors?.required">{{'Admin.ErrorTextDescription' | translate}}</mat-error>
</mat-form-field>
```

Abbildung 43, Validierung Code Beispiel

Damit der User auch weiss, ob die Daten Erfolgreich gesichert wurden, wird jeweils ein Toast angezeigt. Der Toast war im bestehenden Projekt bereits implementiert, ich musste diese nur noch richtig in meinem Modul implementieren und aufrufen.

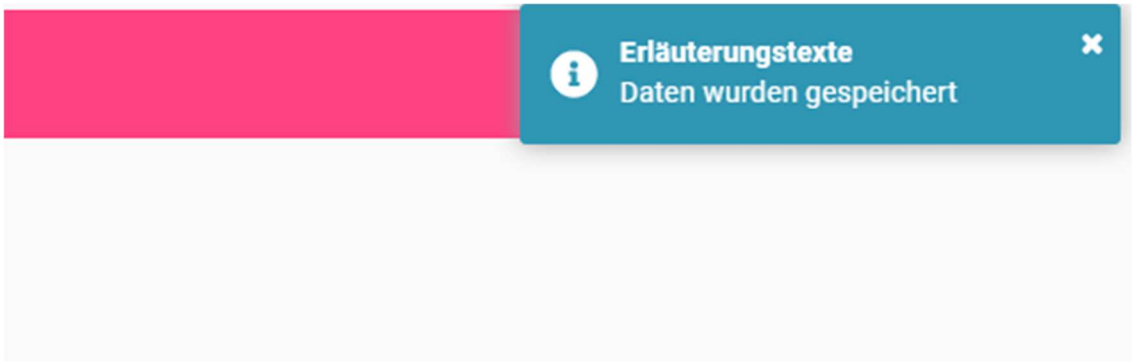


Abbildung 44, Toast Beispiel

Sollte jedoch etwas nicht funktionieren muss der User auch informiert werden. Dann wird jeweils eine Fehlermeldung als Popup angezeigt. Diese war ebenfalls im Projekt schon bestehen, musste jedoch noch jeweils richtig aufgerufen werden.

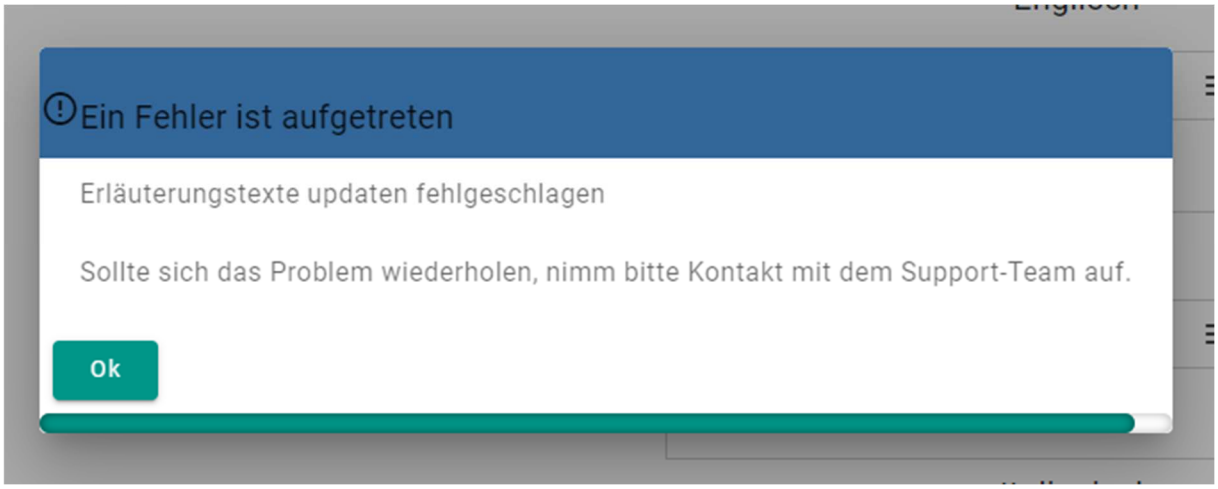


Abbildung 45, Error Meldung Beispiel

17.3.8 Lokalisierung

Damit die Texte in Zukunft auch in anderen Sprachen unterstütz werden können, wurde der vorhandene TranslationService verwendet. Die konstanten Texte wurden jeweils im JSON-File deklariert und dann im Code entsprechend aufgerufen.

```json
"Admin":{
    "EnumerationTitle": "Auflistungen",
    "DefaultValueTitle": "Standardwerte",
    "ParamSimulationNoteTitle": "Erläuterungstexte",
    "NewEntry": "Neuen Eintrag erstellen",
    "NoEntries": "Keine Enträge vorhanden",
    "Save": "Speichern",
    "EnumerationSearchPlaceholder": "Beschreibung oder ProgramCode",
    "Description": "Beschreibung",
    "ShortText": "Abkürzung",
    "ProgramCode": "ProgramCode",
    "SortOrder": "Sortierung",
    "StarYear": "Startjahr",
    "Value": "Wert",
    "ParamSimulationNoteCategoryTitle": "Kategorie Titel",
    "ParamSimulationNoteHeaderTitle": "Header Titel",
```

Abbildung 46, Lokalisierung JSON

```
<input matInput
        placeholder="{{'Admin.EnumerationSearchPlaceholder' | translate}}"
```

Abbildung 47, Lokalisierung Beispiel

17.3.9 Suchfunktion

Damit die Werte innerhalb von den Auflistungen gefiltert werden können, wurde im Shared Modul eine «FilterPipe» erstellt. Diese kann in Zukunft auch von weiteren Komponenten verwendet werden.

```
You, yesterday | 1 author (You)
@Pipe({
    name: 'filter'
})
export class FilterPipe implements PipeTransform {

    transform(items: any[], fields: string[], searchTerm: string): any[] {
        if (!items) {
            return [];
        }
        if (!fields || !searchTerm) {
            return items;
        }

        return items.filter(singleItem => fields.some(field => singleItem.value[field].toLowerCase().includes(searchTerm.toLowerCase())));
    }
}       You, yesterday • implemented searchBar for enumerations
```

Abbildung 48, Suchfunktion

17.3.10 Resultat Design

Auflistungen:

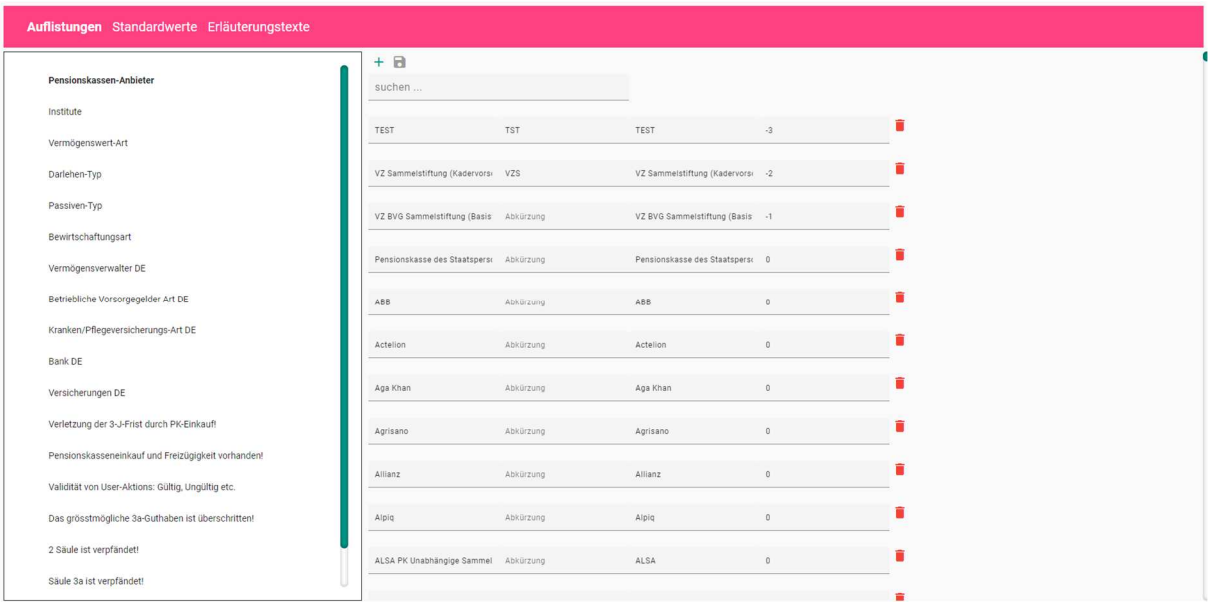


Abbildung 49, Auflistungen Beispiel

Standardwerte:

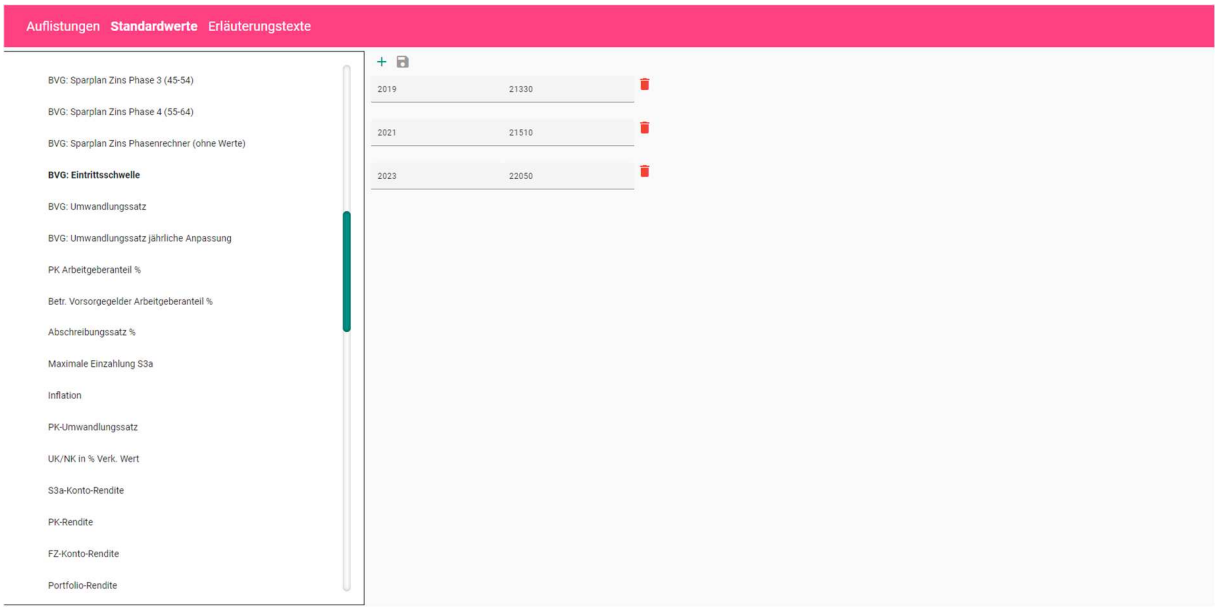


Abbildung 50, Standardwerte Beispiel

Erläuterungstexte:

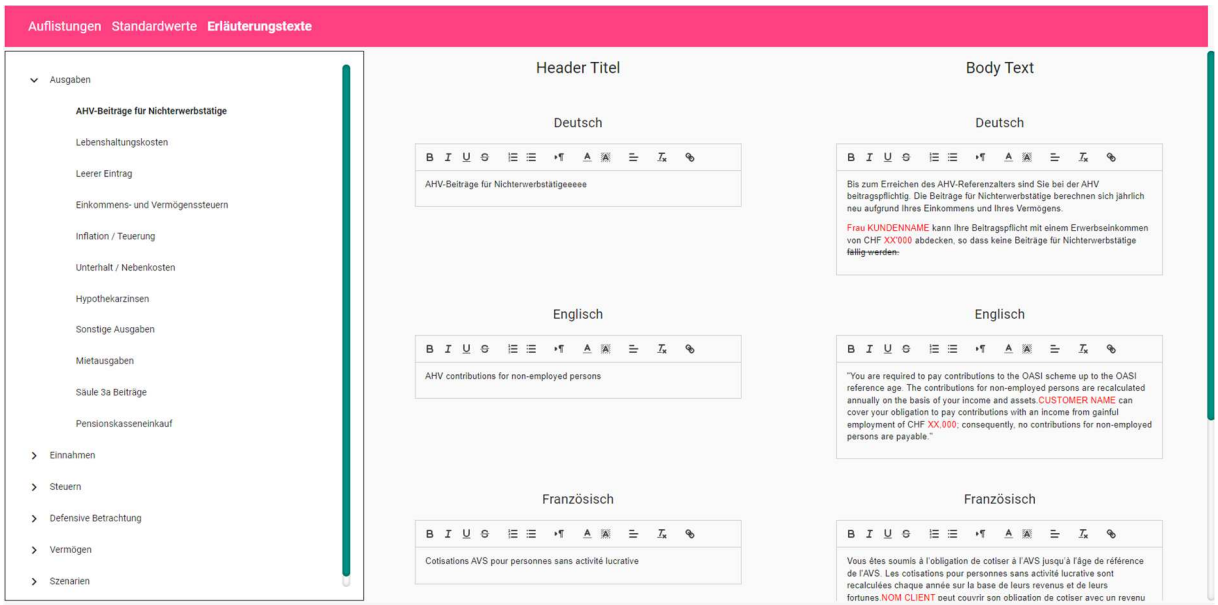


Abbildung 51, Erläuterungstexte Beispiel

18 Kontrollieren

18.1 Testprotokoll

ID	Tester	Erfolgreich	Bemerkungen
T-01	Elion Bajrami	Ja	Es konnte ohne Probleme auf die Admin-Seite navigiert werden und der Header mit den 3 Features war ersichtlich
T-02	Elion Bajrami	Ja	Bei gewissen Typen dauert es eine Weile, bis das Forms mit den Daten angezeigt wird.
T-03	Elion Bajrami	Ja	Daten konnten ohne weitere Probleme gespeichert werden und sind auch nach dem neu Laden der Seite vorhanden.
T-04	Elion Bajrami	Ja	Die neu erstellten Daten konnten erfolgreich hinzugefügt werden und sind auch nach dem neu Laden der Seite vorhanden.
T-05	Elion Bajrami	Ja	Die Daten konnten erfolgreich gelöscht werden, jedoch dauerte dies ein wenig zu lange und könnte in Zukunft optimiert werden.
T-06	Elion Bajrami	Ja	Die Standardwerte konnten ohne Probleme geladen werden.
T-07	Elion Bajrami	Ja	Der Standardwert konnte ohne Probleme gespeichert werden und war auch nach dem neu Laden der Seite vorhanden.
T-08	Elion Bajrami	Ja	Der Eintrag konnte ohne Probleme hinzugefügt werden und war auch nach dem neu Laden der Seite vorhanden.

T-09	Elion Bajrami	Ja	Der Eintrag konnte ohne Probleme gelöscht werden und war auch nach dem neu Laden der Seite vorhanden.
T-10	Elion Bajrami	Ja	Der Kategorie Titel konnte ohne Probleme gespeichert werden und war auch nach dem neu Laden der Seite vorhanden. (Markierungen und Dekorationen waren nicht mehr vorhanden, dies sollte aber auch so sein, denn der Titel sollte nicht markiert oder dekoriert werden)
T-11	Elion Bajrami	Ja	Der Kategorie Titel und auch Body Text konnte ohne Probleme gespeichert werden und war auch nach dem neu Laden der Seite vorhanden. (Markierungen und Dekorationen waren beim Header Titel nicht mehr vorhanden, dies sollte aber auch so sein, denn der Titel sollte nicht markiert oder dekoriert werden)

19 Auswerten

19.1 Fazit

//TODOs

20 Glossar

Begriff	Erklärung
UI	**U**ser **I**nterface
Enumeration/s	Auflistung/en
CRUD	**C**reate **R**ead **U**pdate **D**elete
RTF	**R**ich **T**ext **F**ormat
CLI	**C**ommand **L**ine **I**nterface
DTO	**D**ata **T**ransfer **O**bject
VF	**V**erantwortliche **F**achkraft
PBI	**P**roduct **B**acklog **I**tem
Interface	Schnittstelle
Vulnerable	verletzlich/angreifbar

21 Abbildungsverzeichnis

22 Tabellenverzeichnis

23 Quellenverzeichnis

- https://www.bexio.com/de-CH/blog/view/iperka-methode
- https://www.youtube.com/watch?v=e86l9FNQGsU
- https://komfortzonen.de/entscheidungs-matrix-template-vorlage/
- https://stackblitz.com/edit/angular-form-array-basic-solution?file=app%2Fapp.component.html,app%2Fapp.component.ts,app%2Fapp.module.ts
- https://stackoverflow.com/questions/35461203/angular-2-how-to-apply-orderby
- TODO ergänzen

24 Anhang

24.1 Daily Notizen

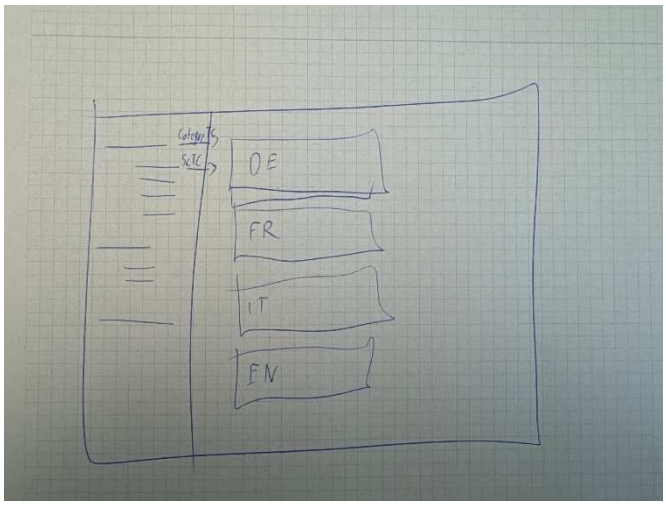


Abbildung 52, Daily 08.03.24

24.2 Code

//TODO