IPA

Erstellen einer Adminmaske zur Erfassung von Standardwerten

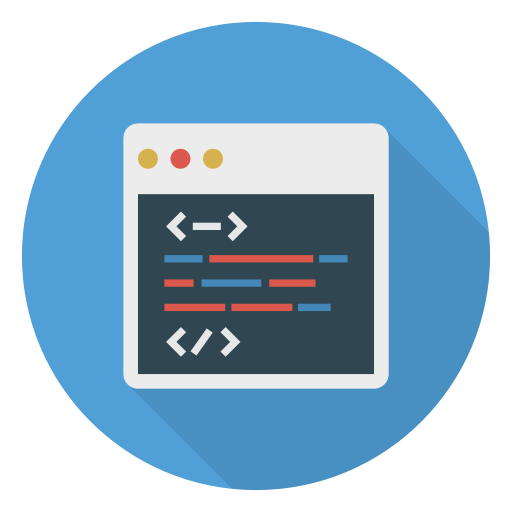


Abbildung , Titelbild

Inhaltsverzeichnis

[**Teil 1 – Umfeld und Ablauf** 4](#_Toc161237884)

[1 Abstract (Kurzfassung) 4](#_Toc161237885)

[1.1 Ausgangssituation 4](#_Toc161237886)

[1.2 Umsetzung 4](#_Toc161237887)

[1.3 Ergebnis 4](#_Toc161237888)

[2 Aufgabenstellung (gemäss PKOrg) 4](#_Toc161237889)

[2.1 Ausgangslage 4](#_Toc161237890)

[2.2 Detaillierte Aufgabenstellung 5](#_Toc161237891)

[2.2.1 Frontend 5](#_Toc161237892)

[2.2.2 Backend 5](#_Toc161237893)

[2.2.3 Features 6](#_Toc161237894)

[2.2.4 Allgemein 8](#_Toc161237895)

[2.2.5 Design 8](#_Toc161237896)

[2.2.6 Datenbasis 8](#_Toc161237897)

[2.2.7 Erweiterbarkeit 8](#_Toc161237898)

[2.2.8 Berechtigungskonzept 8](#_Toc161237899)

[2.2.9 Prüfbare / Messbare Ziele 8](#_Toc161237900)

[2.2.10 Tests 8](#_Toc161237901)

[2.2.11 Architektur 9](#_Toc161237902)

[2.2.12 Dokumentation 9](#_Toc161237903)

[2.2.13 Fehlerbehandlung 9](#_Toc161237904)

[2.2.14 Kompatibilität 9](#_Toc161237905)

[2.2.15 Lokalisierung 9](#_Toc161237906)

[2.2.16 Explizit ausgeschlossene Arbeiten 9](#_Toc161237907)

[3 Vorarbeiten 10](#_Toc161237908)

[4 Vorkenntnisse 10](#_Toc161237909)

[5 Projektaufbauorganisation 11](#_Toc161237910)

[6 Backups 12](#_Toc161237911)

[7 Projektmanagementmethode 12](#_Toc161237912)

[8 Versionsverwaltung 13](#_Toc161237913)

[9 Interne Coding-Guidelines 13](#_Toc161237914)

[9.1 Sprache 13](#_Toc161237915)

[9.2 Vergangenheitsbewältigung 13](#_Toc161237916)

[9.3 Bezeichnungen 13](#_Toc161237917)

[9.4 Aufbau von Klassen 13](#_Toc161237918)

[9.5 Ordner 13](#_Toc161237919)

[9.6 Namenskonventionen 14](#_Toc161237920)

[10 Zeitplan 14](#_Toc161237921)

[11 Arbeitsjournal 17](#_Toc161237922)

[12 Dailies 21](#_Toc161237923)

[13 Azure DevOps Board 23](#_Toc161237924)

[**Teil 2 – Projektdokumentation** 24](#_Toc161237925)

[14 Informieren 24](#_Toc161237926)

[14.1 Datenbank 24](#_Toc161237927)

[14.2 Projektanforderungen 25](#_Toc161237928)

[14.3 Projektumfeld: Systemgrenzen / Schnittstellen zur Aussenwelt 26](#_Toc161237929)

[15 Planen 26](#_Toc161237930)

[15.1 Realisierungskonzept 26](#_Toc161237931)

[15.1.1 Mockups 27](#_Toc161237932)

[15.2 Konzeptionelle Umsetzung 27](#_Toc161237933)

[15.3 Testkonzept 27](#_Toc161237934)

[16 Entscheiden 31](#_Toc161237935)

[16.1 Editor 31](#_Toc161237936)

[16.1.1 Entscheidungsmatrix 32](#_Toc161237937)

[16.2 RTF-Text 32](#_Toc161237938)

[17 Realisieren 34](#_Toc161237939)

[17.1 GIT-Branches 34](#_Toc161237940)

[18 Kontrollieren 35](#_Toc161237941)

[19 Auswerten 35](#_Toc161237942)

[20 Glossar 35](#_Toc161237943)

[21 Abbildungsverzeichnis 36](#_Toc161237944)

[22 Quellenverzeichnis 36](#_Toc161237945)

[23 Anhang 37](#_Toc161237946)

[23.1 Daily Notizen 37](#_Toc161237947)

# **Teil 1 – Umfeld und Ablauf**

# Abstract (Kurzfassung)

## 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.

## 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».

## 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).

# Aufgabenstellung (gemäss PKOrg)

## 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.

## Detaillierte Aufgabenstellung

### 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

### 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.

### Features

#### 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.

#### 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)

#### 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).

### 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.

### 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”).

### 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

### 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.

### Berechtigungskonzept

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

### 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

### 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).

### 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.

### 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).

### 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.

### 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.

### 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.

### 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.

# 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.

# 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

# Projektaufbauorganisation

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

**Kandidat:**   
Elion Bajrami  
[elion.bajrami@vzch.com](mailto:elion.bajrami@vzch.com)

**Lehrmeister:**   
Daniel Roth  
[daniel.roth@vzch.com](mailto:daniel.roth@vzch.com)

**Verantwortliche Fachkraft:**   
Luca Eggenberg  
[luca.eggenberg@vzch.com](mailto:luca.eggenberg@vzch.com)

**Hauptexperte:**  
Roberto Ranieri  
[roberto.ranieri@gmx.net](mailto:roberto.ranieri@gmx.net)

**Nebenexperte:**  
Jesse Vongpanich  
[jesse.vongpanich@gmail.com](mailto:jesse.vongpanich@gmail.com)

# 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.

# Projektmanagementmethode

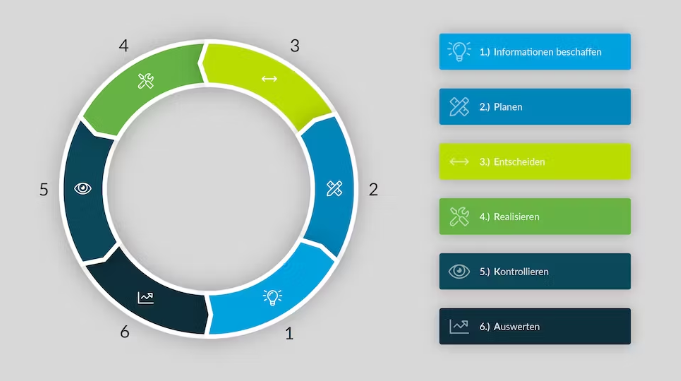
Als Projektmanagementmethode wurde **IPERKA** gewählt. Durch diese Methode wird das Projekt in sechs Schritte aufgeteilt. Welche auch die Gliederung der Dokumentation im zweiten Teil darstellt.

Abbildung , IPERKA

# 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

# Interne Coding-Guidelines

## Sprache

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

## 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.

## 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.

## Aufbau von Klassen

Klassen sollen wie folgt strukturiert werden:

* Private Felder
* Properties
* Konstruktoren
* Public Methoden
* Private Methoden

## 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.

## 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.

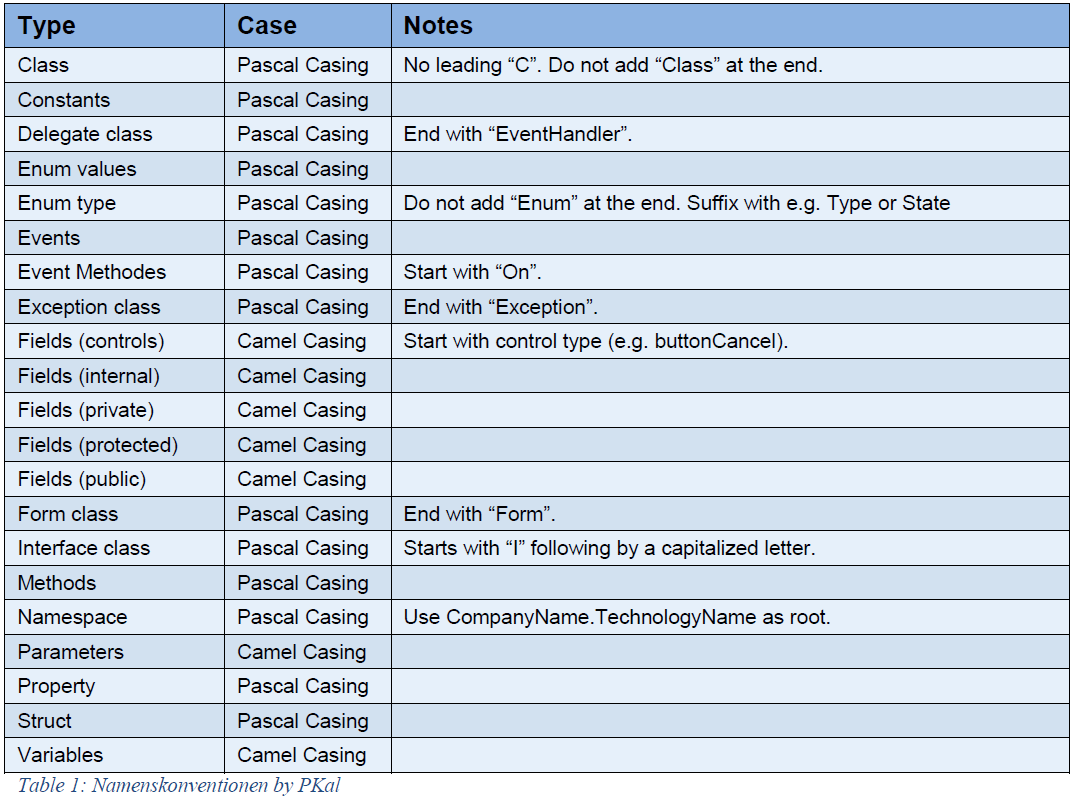


Abbildung , Namenskonventionen

# 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.

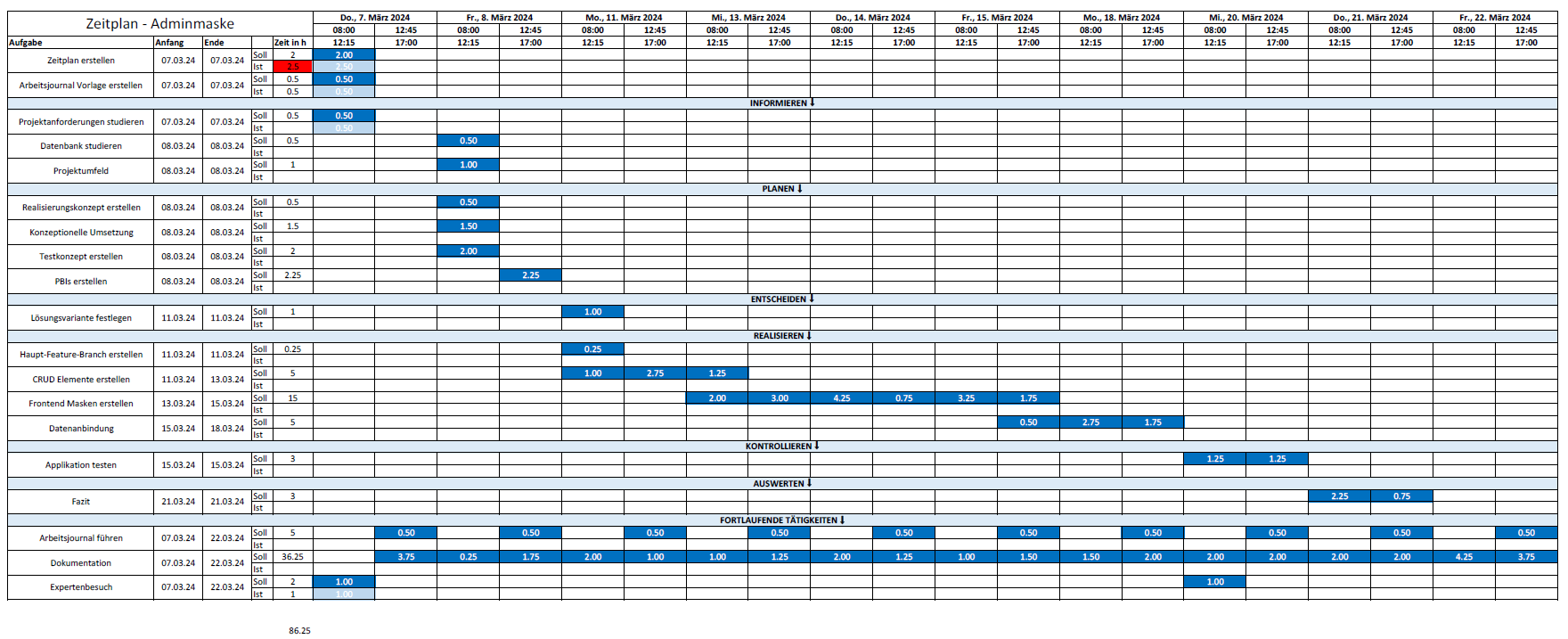
 Anfang:

Abbildung , Zeitplan Anfang

Ende:

# 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 |  |
| Tätigkeiten und erreichte Ziele |  |
| Aufgetretene Probleme |  |
| Beanspruchte Hilfestellungen |  |
| Vergleich mit dem Zeitplan/ Überzeiten |  |
| Ungeplante Arbeiten |  |
| Reflexion |  |

|  |  |
| --- | --- |
| Tägliches Arbeitsjournal, Tag 8, 20.03.2024 | |
| Geplante Ziele |  |
| Tätigkeiten und erreichte Ziele |  |
| Aufgetretene Probleme |  |
| Beanspruchte Hilfestellungen |  |
| Vergleich mit dem Zeitplan/ Überzeiten |  |
| Ungeplante Arbeiten |  |
| Reflexion |  |

|  |  |
| --- | --- |
| Tägliches Arbeitsjournal, Tag 9, 21.03.2024 | |
| Geplante Ziele |  |
| Tätigkeiten und erreichte Ziele |  |
| Aufgetretene Probleme |  |
| Beanspruchte Hilfestellungen |  |
| Vergleich mit dem Zeitplan/ Überzeiten |  |
| Ungeplante Arbeiten |  |
| Reflexion |  |

|  |  |
| --- | --- |
| 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 |  |

# 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 10, 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. |

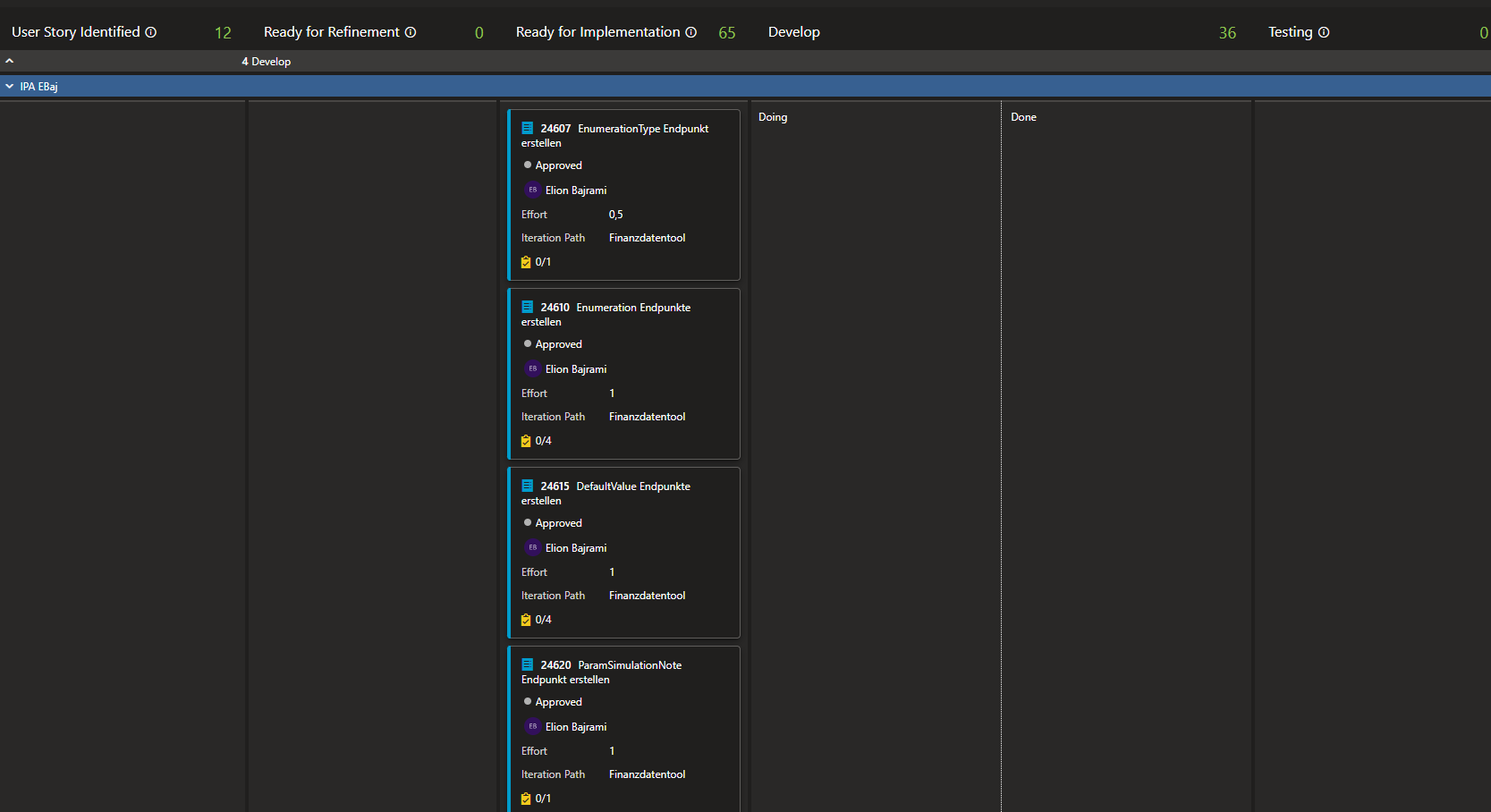
|  |  |
| --- | --- |
| Tägliches Daily, Tag 4, 13.03.2024 | |
| Was wurde heute besprochen | Ich habe heute mit der VF meine Arbeiten für Heute besprochen und abgeklärt ob die Anforderungen richtig verstanden wurden. |
| Aufgetauchte Fragen | Wie sollen die Endpunkte genau aussehen? Müssen die Endpunkte 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. |

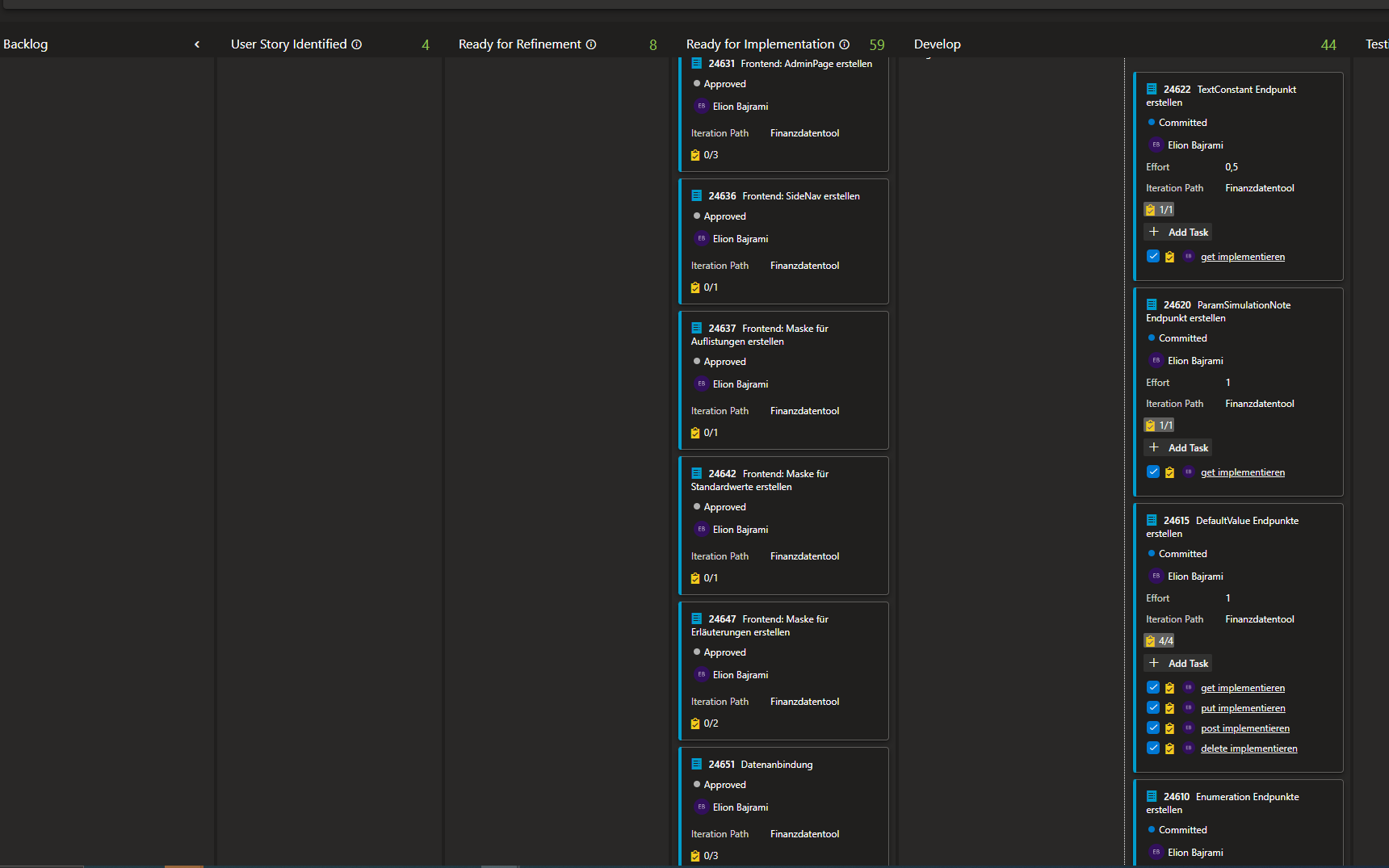
|  |  |
| --- | --- |
| Tägliches Daily, Tag 5, 14.03.2024 | |
| Was wurde heute besprochen | Ich habe heute mit der VF meine Arbeiten für Heute besprochen und abgeklärt ob die Anforderungen richtig verstanden wurden. |
| Aufgetauchte Fragen | Heute war es ein kurzes Daily. Es gab keine Fragen die gross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

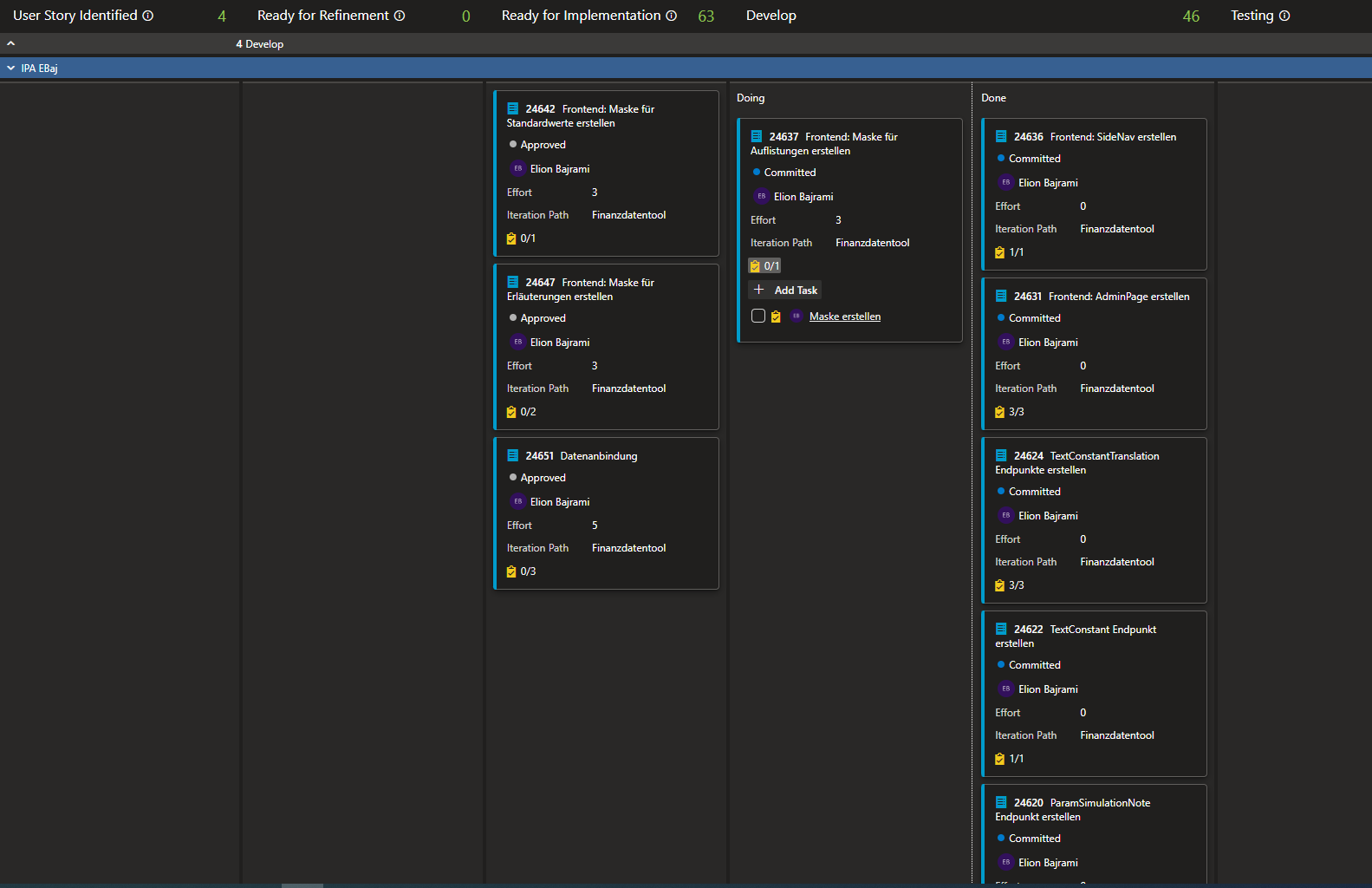
# Azure DevOps Board

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

08.03.24:  


13.03.24:

14.03.24:



# **Teil 2 – Projektdokumentation**

# 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?

## Datenbank

Beispiele wie die bestehenden Daten aussehen und geholt werden können.

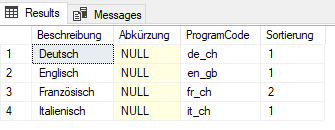
1. **Auflistungen**  
   Query (Enumerations gefiltert nach einem Typ):  
   SELECT sysText AS Beschreibung, ShortText AS Abkürzung, ProgramCode, SortOrder AS Sortierung FROM Enumeration WHERE EnumerationTypeID = 7  
     
   Daten:  
   

Abbildung , Daten Auflistungen

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

Abbildung , Daten Standardwerte

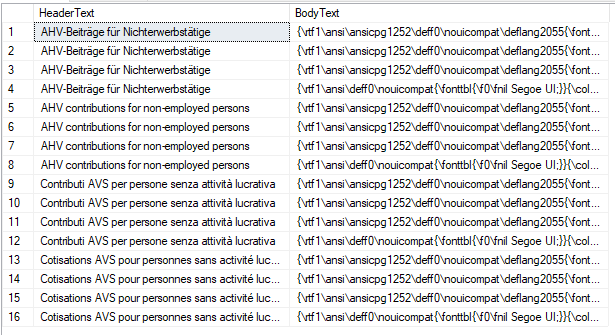
1. **Erläuterungstexte**Query (Erläuterungstext gefiltert nach Kategorie und Subgategorie):  
   SELECT tctHeader.sysText AS HeaderText, tctBody.sysText AS BodyText FROM ParamSimulationNote psm  
   JOIN TextConstant tcHeader ON psm.HeaderTextConstantID = tcHeader.TextConstantID  
   JOIN TextConstantTranslation 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:  
   ****

Abbildung , Daten Erläuterungstexte

## 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.

## Projektumfeld: Systemgrenzen / Schnittstellen zur Aussenwelt

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

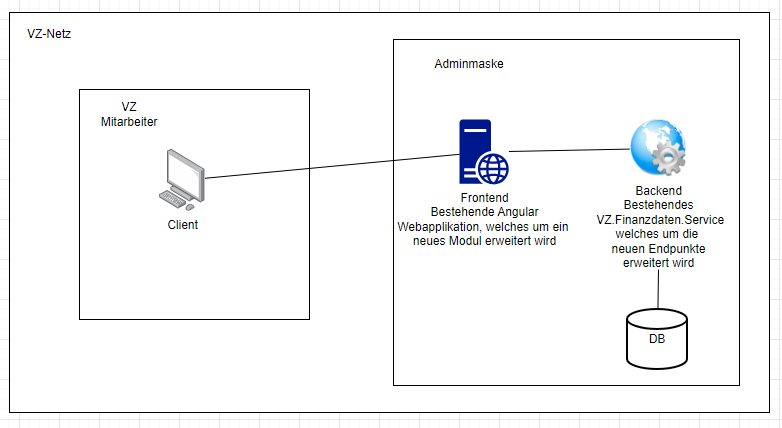


Abbildung , Projektumfeld

# Planen

Das Planen ist die zweite Phase der IPERKA-Methode.

## 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.

### Mockups

## Konzeptionelle Umsetzung

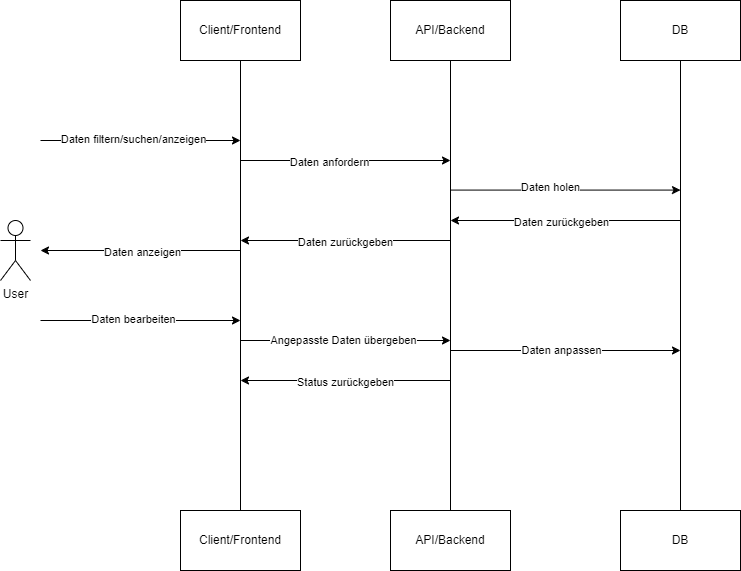


Abbildung , Konzeptionelle Umsetzung

## Testkonzept

* Betriebssystem: Windows 10
* Browser: Chrome
* Server Umgebung: TST

|  |  |
| --- | --- |
| **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-05 |
| 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. Ein Dialog öffnet sich 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. |

|  |  |
| --- | --- |
| **Abschnitt** | **Inhalt** |
| ID | T-06 |
| 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 |

|  |  |
| --- | --- |
| **Abschnitt** | **Inhalt** |
| ID | T-07 |
| 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 4. User klickt auf eine Unterkategorie des Typen |
| Erwartetes Resultat | Daten werden geladen und der User kann diese rechts vom SideNav nun sehen |

|  |  |
| --- | --- |
| **Abschnitt** | **Inhalt** |
| ID | T-08 |
| 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. User klickt auf eine Unterkategorie des Typen 5. Der User bearbeitet einen Standardwert, d.h. er bearbeitet die Felder Startjahr und Wert 6. 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-09 |
| 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. User klickt auf eine Unterkategorie des Typen 5. Der User klickt auf den das Icon «+» 6. Ein Dialog öffnet sich 7. Der User kann in allen Feldern seine Daten eingeben 8. 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-10 |
| 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 eine Unterkategorie des Typen 5. 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-11 |
| Anforderungen | User kann den «CategoryText» 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 «CategoryText» 5. Der User klickt auf den Button «Speichern» |
| Erwartetes Resultat | Der Eintrag wurde gespeichert und ist auch nach einem refresh noch ersichtlich |

|  |  |
| --- | --- |
| **Abschnitt** | **Inhalt** |
| ID | T-12 |
| Anforderungen | User kann den «HeaderText» und «BodyText» 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 «HeaderText» und «BodyText» 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. |

# Entscheiden

Das Entscheiden ist die dritte Phase der IPERKA-Methode.

## 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. ein contenteditable div erstellen. Dieses div wird als Texteingabefeld dienen. 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 italic). 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.

### 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:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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

## 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.  


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 wichtig 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.

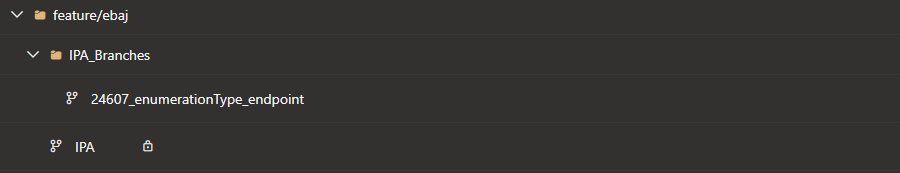
# Realisieren

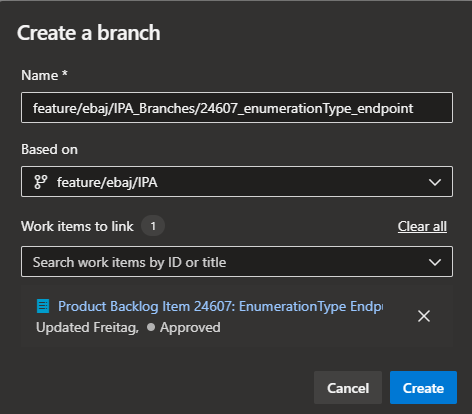
In der vierten Phase der IPERKA-Methode haben wir Realisieren.

## GIT-Branches

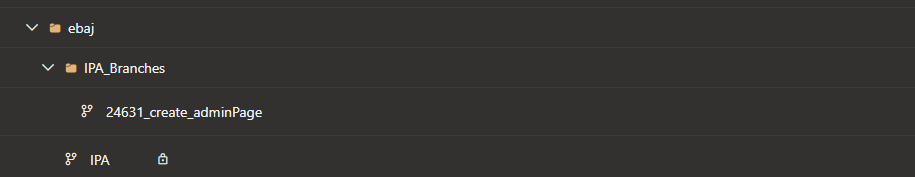
Im Git wurde wie geplant ein IPA-Haupt-Branch erstellt. Von diesem Branch werden die einzelnen Feature-Branches erstellt. Sobald alle Features durch einen Pull-Request in den IPA-Haupt-Branch gemerged wurden und die IPA fertig ist, wird dieser Branch dann in den DEV(Haupt)-Branch gemerged.

### Backend





### Frontend



## Endpunkte

# Kontrollieren

# Auswerten

# Glossar

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Abbildungsverzeichnis

[Abbildung 1, Titelbild 1](file:///\\vz.ch\privat$\vz-privat$\EBaj\Eigene%20Dokumente\16_IPA\V2\IPA_Elion_Bajrami.docx#_Toc160794571)

[Abbildung 2, IPERKA 12](file:///\\vz.ch\privat$\vz-privat$\EBaj\Eigene%20Dokumente\16_IPA\V2\IPA_Elion_Bajrami.docx#_Toc160794572)

[Abbildung 3, Namenskonventionen 14](file:///\\vz.ch\privat$\vz-privat$\EBaj\Eigene%20Dokumente\16_IPA\V2\IPA_Elion_Bajrami.docx#_Toc160794573)

[Abbildung 4, Zeitplan Anfang 15](file:///\\vz.ch\privat$\vz-privat$\EBaj\Eigene%20Dokumente\16_IPA\V2\IPA_Elion_Bajrami.docx#_Toc160794574)

[Abbildung 5, Daten Auflistungen 24](#_Toc160794575)

[Abbildung 6, Daten Standardwerte 25](#_Toc160794576)

[Abbildung 7, Daten Erläuterungstexte 25](#_Toc160794577)

[Abbildung 8, Projektumfeld 26](#_Toc160794578)

[Abbildung 9, Konzeptionelle Umsetzung 27](#_Toc160794579)

[Abbildung 10, Daily 08.03.24 37](file:///\\vz.ch\privat$\vz-privat$\EBaj\Eigene%20Dokumente\16_IPA\V2\IPA_Elion_Bajrami.docx#_Toc160794580)

# Quellenverzeichnis

Bild-IPERKA: <https://www.bexio.com/de-CH/blog/view/iperka-methode>

YouTube-Tutorial Excel: <https://www.youtube.com/watch?v=e86l9FNQGsU>

<https://komfortzonen.de/entscheidungs-matrix-template-vorlage/>

# Anhang

## Daily Notizen

Abbildung , Daily 08.03.24