CMS Polit@ktiv

Anforderungsanalyse

Änderungshistorie

Version	Date	Author	Beschreibung
0.1.0	2023-08-31	CaM	Erstellung der initialen Version
0.1.1	2023-08-31	ScP	Erstellung der initialen Version
0.2.0	2023-09-02	MoM	Diverse Ergänzungen
0.2.1	2023-09-05	ScP	Diverse Ergänzungen
0.2.2	2023-09-06	ScP	"Technik - API" angelegt
			"Technik - Versionierung" angelegt
0.2.3	2023-09-07	ScP	"Technik - API" aktualisiert
			"Technik - Typen" angelegt
0.2.4	2023-09-11	ScP	"API und Typen" zu Baumstruktur
			"ErrorResponse" hinzugefügt
0.2.5	2023-09-11	ScP	Typ "Asset" hinzugefügt
			Typ, API "ComponentConfig" hinzugefügt
			Typ "Content" hinzugefügt
			Typ, API "ContentText" hinzugefügt
			Typ, API "Navigation" hinzugefügt
			Typ, API "NavigationItem" hinzugefügt
			Typ, API "Page" hinzugefügt
0.2.6	2023-09-14	ScP	Update, API [Typ]ListResponse
			Update, RechteRollen Beschreibung
			Update, Type Timestamp Beschreibung
0.2.7	2023-09-15	MoM	Anwenderverw., Rechte&Rollen, Themes
0.3.1	2023-10-12	MoM	Neu: Permanente Bürgerbeteiligung
0.3.2	2023-10-14	MoM	Struktur neu: Anwender, Rechte, Rollen
0.3.3	2023-10-15	MoM	Struktur neu: Elemente und Komponenten
			Oberfläche, Themes, Bedienbarkeit
0.4.1	2023-11-23	MoM	ChatGPT nutzen: Extraktion & Auskunft)
			Oberfläche, Themes, Bedienbarkeit

1. Einleitung

Für die Marke Polit@ktiv (PA) ist ein neues CMS System zu entwickeln. Es löst Liferay vollständig ab. Der derzeit genutzte Funktionsumfang wird übernommen. Neue Funktionen werden diskutiert, definiert und für die Umsetzung eingeplant.

Die bisherigen Anforderungen finden sich hier: ownCloud\Politaktiv\Anforderungen
Sie sind -inhaltlich - nach wie vor gültig, allerdings auf Liferay bezogen.
Im vorliegenden Dokument werden vor allem die Unterschiede und die Basis definiert, die nötig sind, um Liferay zu ersetzen. Dennoch sollen die Basisfunktionen hier nochmals vollständig beschrieben werden, damit sie an einem einzigen Ort gesammelt vorliegen.

Philosophie

Bürgerbeteiligung ist ein Kommunikationsprozess zwischen Verwaltung und Bürgern, bei dem beide Seiten sich über ein Thema (Sache) austauschen. PA möchte zu einer sachlichen und motivierenden Kommunikation beitragen. Die Kommunikation hat klare Rollen und bewusst einen zeitlichen Verlauf. Von der Verwaltung sollte sie zu verschiedenen Themen immer wieder neu angestoßen werden, um Beteiligung und Demokratie langfristig zu stärken.

Sach-Struktur

Auf einem Server (z.B. dem, der bei der Stiftung gehostet wird) laufen mehrere Instanzen.
Eine Instanz gehört zu einer Gemeinde oder wird von ihr gemietet (Instanz = Gemeinde, Kreis oder Land). In einer Instanz können mehrere Diskussionskreise (DK) zu verschiedenen Themen derselben Gemeinde gleichzeitig (oder auch sequenziell) laufen. Jede Instanz bekommt eine eigene URL, die z.B. bei IONOS angemietetet / verwaltet wird und auf die Instanz verweist.

- Jede Instanz hat übergeordnete Seiten, die den DK übergeordnete Funtkionen bietet. Ist in der Instant nur ein DK vorhanden, verschwimmt dies optisch.
- Jede Instanz hat ein eigenes Menü, das für alle enthaltenen DK gilt / zuständig ist. Darin können für jeden DK Untermenüs enthalten sein.

Ein DK besteht aus Seiten, auf die im Menü verlinkt wird.
Eine Seite enthält im Kopf immer den Namen des DK und das Menü der Instanz.
Eine Seite enthält mindestens eine, oft auch mehrere Komponenten. Komponenten sind:

- Accordion
- Artikel
- Extraktion
- Karussell
- Kärtchen
- Kommentar
- Pinnwand

Komponenten können geschachtelt sein.
Beispiel: Die Pinnwand enthält Kärtchen. Ein Kärtchen kann Text und Bild (und Extrakt als Kategorie) enthalten. Zu einem Kärtchen kann es Kommentare geben, die wiederum Text und Bild enthalten.
Eine Komponente kann ein oder mehrere Elemente enthalten. Elemente sind:

- Text
- Bild (oder ein Link darauf)
- Video (oder ein Link darauf)
- Extrakt

Kommunikationspartner (User)

Die Benutzer (Bürger, User, Admins, etc...) sind von der Sachstruktur unabhängig.

Gast

Der einfache Bürger muss sich nicht anmelden, hat dann aber nur eingeschränkte Rechte. Er heißt Gast und kann als solcher in allen DK mitmachen, von denen er Kenntnis hat.

User

Jede Instanz bietet die Möglichkeit, sich zu registrieren. Die Registrierung gilt dann für alle Instanzen, die auf demselben Server laufen. Durch die Registrierung entsteht aus Sicht PA ein User, der gekennzeichnet ist durch seine Mailadresse und sein Passwort. (Wenn zwei ältere Leute nur eine einzige Mailadresse haben, gelten Sie als ein einziger User.) Beim User ist in einer Liste abgespeichert, für welche Instanz (nicht DK!) er sich interessiert. In seinem Profil kann er dies ergänzen / ändern. Hat er in verschiedenen Instanzen unterschiedliche Rechte, wird dies in seinem Profil bei der Instanz vermerkt.

Admin

Ein User kann Admin für eine Instanz sein (nicht nur für einen darin enthaltenen DK). Der Admin hat kann User Rechte innerhalb der Instanz zuteilen.

Verwalter

Ein User kann spezielle Rechte für einen (Gemeinde-) Verwalter (Instanz) bekommen.

Superadmin

Ein Superadmin hat das zusätzliche Recht, neue Instanzen anzulegen und ist in allen Instanzen, die er angelegt hat immer auch Admin. Er wird dazu vom Server-Admin ernannt.

Zu Bedenken: Im Betrieb der HIT GB2 kommt derzeit jeden Monat mindestens eine neue Instanz hinzu. Wenn der Erfolg der HIT sich so fortsetzt, ist damit zu rechnen, dass ab 2024 jede Woche eine neue Instanz angelegt werden muss. Dazu darf es nicht erforderlich sein, dass jedes Mal ein Server-Admin (Techniker) eingreifen muss!

Vertragstuktur

Die Stiftung betreibt das Hosting und hat einen Vertrag mit der HIT, die die Instantzen bedient.
Es muss möglich sein, weitere Lizenznehmer vertraglich zu einzubinden, die dann eigene Server betreiben für mehrere Instanzen und Bürgerbeteiligungsverfahren - und dafür eine Lizenzgebühr an die IST entrichten.

Nutzung OpenSource-Quellen

Soweit irgend sinnvoll und möglich sollen vorhandene Open-Source-Codequellen genutzt werden.
Absicht dahinter ist, dass in Zukunft nicht alles neu entwickelt werden muss - mit dem Ziel, frühestens in etwa 10 bis 15 Jahren wieder eine völlig neue Version erstellen zu müssen.
Und wenn es ein dafür brauchbares Open-Source-Basis-CMS gibt, sollte dies genutzt werden.

Zeitplan

Bei der Ablösung von Liferay soll nach einem Stufenplan vorgegangen werden.
Folgende Abschnitte sind aktuell geplant:

1. Erste brauchbare Version (V1.0) bis Ende 2023
2. ....

2. Basis Anforderungen an das CMS

Damit Liferay abgelöst werden kann, müssen grundsätzliche Anforderungen erfüllt sein. Diese bezeichnen wir als Basis Anforderungen. Sie werden hier gesammelt und spezifiziert.

Basis Anforderungen

1. Anwender Verwaltung
2. Konzept Elemente und Kompnenten
3. Leicht bedienbare Oberfläche und Responsive Design
4. Einbindung in Piwik und Auffindbarkeit durch Google-Suchmaschine (SEO)
5. Admin (nicht Server-Admin) kann jederzeit neue Instanz (Diskussionskreis, DK) anlegen.
6. versteckte Instanz und darin DK zum Üben (versteckt vor Öffentlichkeit).
7. "Permanente" Beteiligung mit darin enthaltenen DK
8. Es muss einfach möglich sein, weitere Funktionen dazuzubauen
9.

2.1. Anwender Verwaltung

Liferay verfügt über ein komplexes Rechte Rollen System (RRS). Dieser Grad ist für das Polit@aktiv CMS nicht notwendig. Unser RRS wird in der Basis Version eine den alltäglichen Anforderungen angepasste Komplexität haben. Erweiterung sind möglich und werden nach Bedarf ergänzt werden.

Die Anmeldung geschieht immer mit der Emailadresse und Passwort. Das Passwort sollte gewisse Mindeststandards erfüllen wie Sonderzeichen, Buchstaben und Zahlen enthalten.

Der Admin muss in der Lage sein, einen Account zu löschen. Nur stillegen reicht nicht, damit sich ein Gast, der sich vertippt hat, seinen Account neu anlegen kann.

Jeder User soll sich sein Passwort zurücksetzen lassen können (Passwort vergessen).

Das *Basis* RRS besteht aus folgenden **Rechten** und **Rollen**:

Prinzip:

- Es gibt Rollen (Admin, Redakteur, User, Gast (Wobei Gast üblicher Weise als unangemeldet gilt))
- Es gibt Rechte (darf neuen Kunden / Seite / etc anlegen)
- Die Rechte werden den Rollen zugewiesen. z.B. hat der Admin fast alle.
- Die Rechte bleiben auf der implementierungsebene, also dem User unsichtbar.
- Die Rollen werden über eine Oberfläche dem Benutzer zugewiesen.

> Welche Rechte wie verteilt werden, muss von den Admins und Redakteuren diskutiert werden.
> Das kann sich auch im Laufe der Zeit verändern. z.B. wenn neue Anforderungen entstehen.

Basis Rollen

Superadmin

Ein Superadmin hat das Recht, neue Instanzen anzulegen und ist in allen Instanzen, die er angelegt hat, immer auch Admin. Er wird dazu vom Server-Admin ernannt. Nur der Superadmin kann neue Rollen erfinden.

Admin

Beim Anlegen einer Instanz (mit vielen DK) wird ein Admin automatisch angelegt / vorgegeben. Aus dieser heraus kann man weitere User mit der Rolle Admin ausstatten.

Jeder Admin hat das Recht, Usern Rollen zuzuteilen und zu entziehen.
Zusätzlich kann der Admin einem User ein spezielles Recht individuell einräumen.

User

Gäste können sich einen Account anlegen. Danach haben sie dann automatisch die Rolle eines Users. Beim Anlegen des Accounts ist die Enmail-Adresse zu verifizieren, der Account also erst dann gültig, wenn eine Email verschickt und bestätigt wurde. Jeder User hast zusätzlich einen Namen, der frei gestaltet werden kann und der zum Anzeigen eines Beitrages verwendet wird. Wenn beim Registrieren vom Gast nichts spezielles angegeben wird, soll der Name sein bürgerlicher Name sein.

Gast

Gast ist jeder sonstige Besucher, der eine Website anschauen kann und die URl dazu kennt und der sich nicht als Admin, Redakteur oder User ausgewiesen hat. Um Gast zu sein, muss man sich nicht anmelden. Der Admin kann auch den Gästen Rechte zuweisen. .

Zusätzlich zu den Gästen, User, Admins und Superadmins gibt es weitere Rollen, die im Folgenden beschrieben sind. Sie können vom Superadmin definiert und vom Admin den Usern zugeteilt werden.

Weitere Rollen

Redakteur (Content Bearbeitung und Freigabe)

Hauptaufgaben sind:

1. Neue Texte / Seiten / Bilder innerhalb einer Instanz anlegen
2. Den Seiten Module zuweisen (Kartenportlet, Pinwand, ...) und konfigurieren.
3. User innerhalb einer Instanz in Gruppen einteilen.
   Es muss möglich sein, verschiedene Ausprägungen von Redakteuren zu erschaffen, die dann auch adners heißen dürfen.

Verwaltung (Statistiken und erstellen und exportieren)

Hauptaufgaben sind:

1. Statistiken erstellen
2. Statistiken exportieren

Rechte

tbc

Technik

Zur Authentifizierung soll keycloak verwendet werden.
Keycloak bringt out-of-the-box OAuth2 und ein additives RollenSystem mit. Im Prinzip lassen sich dort einem Benutzer / Rolle / Gruppe beliebige verschachtelungen von Rollen und Rechten realisieren.

2.2. Konzept Elemente und Komponenten

Hier wird die Sachstruktur beschrieben, bestehend aus den Elementen und den Komponenten. Siehe dazu auch die übergeordneten Bemerkungen hzur Sachstruktur in der Einleitung.
Einige für Bürgerbeteiligung typische Komponenten sind in der Owncloud ausführlich beschrieben. Da sie eine spezielle und wichtige Funktion zu erfüllen haben, dürfen sie nicht naiv umgesetzt werden. Das sind die Pinnwand, die Extrakte und die Kommentare. DAher bei der Umsaetzung unbedingt die Ausfürhliche Beschreibung in der Ownclodu beachten!

Server

Ein Server wird von der Stiftung betrieben. Darauf laufen alle Instanzen und Diskussionskreise für die Bürgerbeteiligungsverfahren, die von der Stiftung oder der HIT betrieben werden. Für diesen Server übernimmt die Stiftung die technische Verantwortung und berechnet es an die HIT in Kombination mit der Lizenz, die die Stiftung an die HIT vergeben hat.

Instanz

Jede Instanz ist einer Gemeinde (oder Kreis oder Land) zugeordnet. In einer Instanz können mehrere Diskussionskreise laufen. Jede Instanz bekommt eine Domain (URL) zugeteilt, über die die Instanz im Internet erreicht werden kann. Eine Instanz wird vom Super-Admin eingerichtet / angelegt.
Einer Instanz (nicht aber einem DK) kann ein Theme (nicht Thema) zugeordnet werden, also eine Struktur für den Seitenaufbau. Im Prinzip kann also jede Instant ein eigenes, spezielles Theme haben. Jede Instanz hat ein Menü, das als Obermenü für alle darin enthaltenen Diskussionskreise (DK) fungiert.

Diskussionskreis

Jeder Diskussionskreis hat sein eigenes Beteiligungs-Thema (nicht Theme). Es gibt dazu einen Start-Zeitpunkt, einen klaren Ablauf des Beteiligungsprozesses und einen Ende-Zeitpunkt, nach dem keine Beteiligungsereignisse mehr stattfinden. Danach kann er nur noch betrachtet werden und nur noch der Admin kann weitere, eventuell auch sehr viel spätere Ergebnisse einbringen. Beispiel: Der Gemeinderat beschließt nach 2 Jahren, nur einen Teil der besprochenen und ursprünglich auch mal beschlossenen Teile umzusetzen.

Seiten

Jeder Diskussionskreis besteht aus Seiten, die man im Internet ansehen und im Menü aufrufen kann. Auf den Seiten werden Komponenten angezeigt, die je nach Theme (nicht Thema)

entsprechend angeordnet sind.

Komponenten

Es gibt u.a. folgende Komponenten, die ihrersseits aus den verschiedenen Elementen aufgebaut sein können:

- Accordion
  - 1 bis X Flächen, die man aufklappen kann, in denen man Inhalte darstellen Kann.
- Article
  - Artikel aus der Datenhaltung verknüpft
  - Auch teilweise hier bearbeitbar
  - alle Artikel müssen Kriterien haben können, nach denen sortiert werden kann.
  - Beiträge der Bürger sind Artikel, wenn sie nicht Kommentare sind.
- **Kartenfunktion**
  - Einfache Lage-Darstellung
  - Erweiterte Funktionen siehe auch Owncloud
- Karussell (Banner <=>)
  - Banner Karussell mit mehreren slides, die mit Inhalten gefüllt werden können (Asset, Link, Text, etc)
- **Kommentarfunktion**
  - Die Kommentar-Komponente ist ein Inhalt (wie ein Artikel).
  - Es soll möglich sein, Kommentare angemeldet oder anonym zu posten
  - Eine einfache Moderation ist hier erforderlich
  - Kommentare müssen (einstellbar) möglich sein für: Assets, Articles, Pinnwände, Karten. nicht aber für: Karussell, Accordion, Formulare.
- **Pinnwand**
  - Eine Pinnwand ist ein Inhalt
  - Die Pinnwand dient dazu, Themen in Kurzform sichtbar niederzulegen.
  - Es muss mehrere Pinnwände pro DK (Instanz) geben können, aber immer nur eine pro Seite.
  - an eine Pinnwand können beliebige Texte und Bilder gepinnt werden. Sie sind durch entsprechende Begriffe sortierbar zu machen.
  - ein Kommentar an eine Pinnwand ist ein "Kärtchen"
  - für die Pinnwand muss es **Massenimport** geben: Bei einer physischen Versammlung werden die Kärtchen an die physische Pinnwand geheftet / umsortiert. Das wird abfotografiert und die Kärtchen werden eingesammelt. Sie werden dananch (hoffentlich automatisch) erfasst und als Massenimport an die DK-Pinnwand "geheftet". Typisch sind das mehr als 100 Kärtchen pro Versammlung.
- Dynamisches Formular (z.B. Befragung)
  - (Diskussionsbedarf)
  - Idee: Es soll eine Möglichkeit geben, bestimmte Daten (Frage / Antwort) zu konfigurieren, ohne dass dazu noch Implementierung notwendig ist.
- Vorformulierte Formulare
  - Eingabe eines Kommentars
  - Kontakt Daten formular
  - Newsletter Anmeldung
- Statistik

    - Soll automatisch über bestimmte Elemente (Extrakte, etc...) gemacht werden können (später)

Elemente

Elemente müssen Kriterien haben, nach denen sortiert werden kann.

Texte

- verschiedene Schriftarten
- Überschriften in verschiedenen Schriftgrößen
- eingerückte Abschnitte
- Fett, kursiv, unterschrichten, etc...

Bilder

- Größe muss definierbar sein

Videos und Sound

- Link auf Videos, die auf einer Seite angesehen werden können
- Link auf Podcasts, die auf einer Seite angehört werden können
- dürfen in Youtube abgelegt sein. Hosting-Platform ist noch offen.

Extrakt

- Ein Extrakt ist eine (extrem) kurz formulierte Anforderung (reiner Text, max 10 Worte). Siehe auch Owncloud
- Über Extrakte kann Statistik gemacht werden, hat also "Häufigkeit" als Kategorie
- Extrakte können nur von HIT-Mitarbeitern angelegt werden (vorläufig manuell). Sie werden aus Kommentaren oder Beiträgen der Bürger erstellt und diesen zugeordnet. Aus einem Kommentar oder Beitrag können mehrere Extrakte erstellt werden, auch wenn das eher selten vorkommt.
- Pro DK existiert jedes Extrakt nur ein einziges Mal.
- Extrakte sind Kategorien der Beiträge der Bürger und können auf einer Pinnwand dargestellt werden.

Funktionen

- leichte Sprache
- Texte vorlesen lassen
- Videos sollen automatisch erklärt werden können (später)

- über Extrakte soll automatisch Staistik gemacht werden können

2.3. User Interface

- Polit@aktiv wird für verschiedene Kunden zur Verfügung gestellt. Es ist unbedingt erforderlich, verschiedenen Kunden verschiedene Designs und Layouts anbieten zu können. Dies soll mit verschiedenen Themes erreicht werden.
- Für Mitarbeiter und für Benutzer sollte das System, insbesondere auch die Oberfläche / GUI, möglichst leicht / intuitiv bedienbar sein. Dazu zählen folgende Funktionen:

Aus Sicht Redakteur

- Inhalte in die Site einfügen
  - neue Seiten anlegen
  - Webcontents erstellen
  - Bilder hochladen und einfügen
- Spalten und Content-Container

Themes

Ein Theme (nicht Thema) ist die Zusammenstellung von Formen, Schriften und Farben. z.B.

- Hat ein Textblock abgerundete Ecken oder kantige?
- Gibt es Schlagschatten an einem Bild oder harte Kanten?
- Ist die Schrift Helvetica oder Arial? Dunkelblaue Schriftfarbe auf grünem Hintergrund oder Braun auf Grau?
- Ist die überschrift einer seite zwei oder drei mal so groß wie der restliche Text.
- etc.
  Das Theme bestimmt also, wie die Standardkomponenten aussehen.
- Es sollten verschiedene Themes zur Verfügung stehen. Anfangs nur einige wenige. Im LAufe der Nutzungszeit sollten weitere dazukommen können.
- Ein Theme wird ausgewählt bei der Erzeugung einer Instanz (für die jeweilige Gemeinde). Es gilt dann für alle darin enthaltenen Diskussionskreise (Themen).
- Neue Seiten werden immer im vorgegebenen Theme angelegt.

Layout

Ein Layout bestimmt die grundlegende Struktur der Seite. Es hängt von der Bildschirmgröße ab. z.B.

- Navigation immer oben oder Navigation immer links an der Seite?
- Content auf der vollen Breite darstellen oder in der Breite beschränken und den Inhalt zentriert darstellen? Bitte eher nicht.
- Wird der Inhalt in einer oder in zwei Spalten angezeigt? Bei großen Bildschirmen kann es durchaus bis zu vier Spalten geben.

Es soll mehrere Layouts zur Auswahl für einen Diskussionskreis geben. Diese hängen aber stark von der Bidlschirmgröße ab: Responsive Design!
- Responsive Design ist KO-Kriterium für alle Themes. Beispiele:
- Der Artikel aus der Mitte eines Containers soll auf die volle Breite des Containers gestreckt werden, wenn seine Breite unter Größe "md (medium)" fällt.
- Eine Spalte sollte mindestens 40 und maximal 100 Zeichen (der "normalen" Schrift) breit sein.
- Auf einen großen Bildschirm passen dann bis zu vier Spalten. (Vorbild: Tageszeitung). der Redakteur gestaltet Artikel in Spalten, von denen - je nach Bildschirmgröße - mehrere nebeneinander stehen dürfen / sollen. Ist der Bildschirm kleiner, werden automatisch weniger Spalten angezeigt. Ist er sehr klein, werden die Spalten schmäler angezeigt.

Design mit Theme einfach anpassbar

Es wird ein Basis-Theme (nicht Thema) zur Verfügung gestellt.
Vorschlag: Das Basis-Theme ist ein "Tailwind" basierendes UI namens "Skeleton UI". Damit gibt es ein Grundset an CSS. Ein neues Thema muss dann nicht von null anfangen und kann den Komfort von "Tailwind" in der Umsetzung seiner individuellen Style-Anpassungen zurückgreifen. (Tailwind, Skeleton UI)
Folgende Funktionen werden in der Basis Version umgesetzt werden:
Für die Seite+SeitenLayout soll es möglich sein, CSS einzugeben, deren Auswirkung dann auch angezeigt wird.
Ein Theme soll mit einem Layout verknüpft werden können. So kann man ein Theme für verschiedene Seiten verwenden.

Themeentwicklung

- Um ein Theme zu erzeugen, sollte es ein spezielles Menu geben, das dem Admin zur Verfügung steht. Kann eventuell auch später zur Verfügung gestellt werden.

> **Es ist genau zu klären, wie das geschehen sollen. Hier fehlen weitere Detail-Informationen.**

- Eine Idee (weniger Aufwand): Ein Theme wird in der Form einer CSS-Eingabe mit Namen als Datensatz gespeichert. Diese CSS werden dann mit einer Seite verknüpft und werden dann mit der Seite entsprechend ausgeliefert. Die Bearbeitung des Themes erfolgt über ein Text-Eingabefeld, in das die CSS geschrieben werden können.
- Eine weitere Idee (mehr Aufwand): es gibt einen Editor, in denen man Formen, Farben und Typen angibt und aus dem dann die CSS generiert werden. (Version v0.9.x? oder später?)

Schriften

Es muss eine ganze Reihe von Schriften geben. Sie müssen fett oder kursiv oder unterstrichen eingesetzt werden können. Ist hochgestellt erforderlich? Auf jeden Fall sind alle Schriften in verschiedenen Größen erforderlich. Ein Schrift sind Sonderzeichen (z.B. griechische Zeichen).
Die Schriftarten legt Anni fest. Es muss möglich sein, später weitere Schriften zu implementieren.

2.4. Funktionale Templates

Die Funktionalen templates sollen als visualisierter Editor auf eine Seite gezogen/geklickt werden können. Die Datenhaltung soll von den Templates getrennt verwaltbar sein, um so zu ermöglichen, dass z.B. Assets und Artikel an verschiedenen Stellen angezeigt werden können.

Für die Templates soll es Einstellungsmöglichkeiten geben, wie z.B. Einem Artikel-Layout einen Artikel (Datensatz) zuweisen oder Styles für den Artikel anpassen, wenn es erforderlich ist.

Assets / Verwaltung (Videos, Bildern, Dateien)

Es soll eine Verwaltung von Assets geben. Dort können Dateien hoch geladen, gelöscht und mit verschiedenen AssetListen verknüpft werden.

In der Verwaltung kann der Anwender Assets hochladen, löschen und sie zu Asset-Listen hinzufügen. Dies geschieht in einer eigenen Verwaltungsoberfläche für Assets. Beim Konfigurieren der Seite kann man sich dann ein Asset oder eine Asset-Liste bei den relevanten Templates auswählen.

> Version v0.9.x

Mediengalerie

Die Mediengalerie ist eine Anzeige von Videos (Vorschau), AudioFiles und/oder Bildern, Files. Eine Liste an Kacheln haben wird als Darstellung angedacht. Ene Detailansicht kann bei einem Bild eine Größere Darstellung und bei einem Video/AudioFile ein Player sein, der das Video abspielt. Für die Verwaltung: siehe #Assets!

Die Mediengalerie kann von einem Anwender auf eine Seite als Komponente eingebunden werden. Dann kann dieser eine Asset-Liste zugewiesen werden. Damit werden alle Assets der Liste dann entsprechend angezeigt.

> Version v0.9.x

Videoanzeige

Videos werden über einen Player angezeigt. Dazu soll es ein Video-Template geben, dass dann mit einem Asset (Video) verknüpft wird.

Der VideoPlayer kann von einem Anwender als Komponente auf die Seite konfiguriert werden. Der kommt auch zum Einsatz bei der Detailansicht in der Mediengalerie.

> Version v0.9.x

Pinnwand

Die Pinnwand ist angedacht als eine Liste von einfachen Artikeln. Diese Artikel beinhalten eine Überschrift, einen Text, den Zeitpunkt der Erstellung, eine Referenz auf den Ersteller und eine Liste von Schlagworten. Die Artikel der Pinnwand können nach jeweils einem Schlagwort gefiltert werden.

Die Pinnwand kann vom Anwender als Komponente auf eine Seite konfiguriert werden.

> Version v0.9.x

Formulare

Formulare für bekannte Prozesse werden je nach Bedarf implementiert.

- Kontaktformular
- Newsletteranmeldung
- etc

> Zu klären: "Eine vereinfachte form als Frage/Antwort kann zusammen geklickt werden." ?

> Version v0.9.x

Extrakte

sind extrem kurz formulierte Auszüge (Kernaussagen) aus den Beiträgen (Artikel, Kommentare) der Bürger. Jedes Extrakt gibt es nur einmal. Den Beiträgen der Bürger werden die Extrakte zugewiesen. Über die Extrakte können Statistiken erstellt werden - auch grafisch.
In einer späteren Version ist geplant, dazu ChatGPT einzubauen, das aus den eingebrachten Argumenten die Extrakte erstellt.

Karussell

Das Karussell ist eine Liste von Flächen, die mit Inhalten (Bild, Video, Artikel) gefüllt werden können. Es wird immer nur eine Angezeigt. Mit einer Navigation kann man von einer zur anderen Fläche umschalten.

Das Karussell kann vom Anwender als Komponente auf eine Seite konfiguriert werden. Dem Karussell können einzelne Flächen und deren Inhalte hinzugefügt werden.

> Version v0.9.x

Artikel

Ein Artikel ist eine Zusammenstellung von Assets und Texten. Zum einen wird der Inhalt in einer Tabelle gepflegt. Für die Darstellung wird dann ein Artikel-Layout und ein Artikel gewählt.

verwaltung des Artikels

Artikel können in einer separaten Oberfläche (Tabelle, Suche Formular) angelegt und bearbeitet werden. Dadurch kann ein Artikel an mehreren Stellen wieder verwendet werden.

Darstellung des Artikels

Ein Artikel kann auf einer Seite figuriert werden. Für einen Artikel wird es verschiedene Layouts geben. z.B. "Bild oben, Text unten" oder "Überschrift links, Text rechts"

> Version v0.9.x

Fragebögen Einbinden, iFrame

Für Fragebögen ist angedacht, diese aus externen Quellen einzubinden. Dafür wird ein entsprechender Platzhalter verwendet, der dann auf den Fragebogen referenziert.

> Details sind hier zu klären..

> Version v0.9.x

Verschlagwortung, Tagging

Für verschiedene Entities soll eine verschlagwortung stattfinden. Hierfür wird eine möglichkeit geschaffen, Schlagworte in den Bearbeitungseditoren einzubinden. #HashTags

> Zu klären ist hier, was soll genau verschlagwortet werden? Was soll dann wo über diese Schlagworte gefiltert werden?

> Version v0.9.x

Accordion

Es wird ausklappbare Boxen geben. Diese können auch in Abhängigkeit stehen. z.B. Es kann nur eine offene Box geben, also schließen sich die anderen Boxen, wenn man eine öffnet.

Das Accordion kann vom Anwender als Komponente auf eine Seite konfiguriert werden. Dem Accordion können einzelne Flächen und deren Inhalte hinzugefügt werden.

> Version v0.9.x

TAB's

Es wird Container geben, die man über TAB's steuern kann.

Die TAB-Darstellung kann vom Anwender als Komponente auf eine Seite konfiguriert werden. Der TAB-Darstellung können einzelne Flächen und deren Inhalte hinzugefügt werden.

> Version v0.9.x

Modal (Popup)

Man wird Inhalte in ein Modal konfigurieren können. Also in eine Fläche, die man wieder beliebig mit Inhalt füllen kann und als Popup über der Seite "schwebt".

> Konfiguration und Handhabung muss geklärt werden. Erwartung unbekannt.

> Version v0.9.x

Chat (angedacht)

> Version v2.x.x

Newsletter Portlet (angedacht)

> Version v2.x.x

Map

Openstreetmaps soll eingebunden werden können. In einer Einfachen form um eine Position anzuzeigen. Das KartenPortlet mit seiner komplexeren Anforderung ist eine andere, eigene Komponente.

Die Karte kann vom Anwender als Komponente auf eine Seite konfiguriert werden. In der Konfiguration kann Start-Position und Zoom-Faktor angegeben werden.

> Version v0.9.x

Cookiebanner

Um den Vorschriften nachzukommen wird es ein Cookiebanner geben, dass nach dem Stand der Technik die Vorschriften erfüllt.

> Zu klären, wie und wo speichern wir die Einstellung des Besuchers? Tun wir das überhaupt?

> Version v0.9.x

Inhaltliche Erläuterungen

Geplant für eine spätere Version ist, ChatGPT einzubinden, das Erläuterungen geben kann zu:

1. allen bereits im DK von PA diskutierten Argumenten.
2. zu den Verhältnissen in der Gemeinde allgemein.
3. zu Vorschriften, die im Umfeld zur laufenden Bürgerbeteiligung gelten.

2.5. Funktionale Templates, Plug-Ins

Unter funktionalen Templates oder Plug-Ins verstehen wir weitere Oberflächen Funktionalitäten, die auch Auswirkungen auf die Daten in der Datenbank haben können.
Folgende Plug-Ins sollten implementiert werden:

- Ausklappbare Boxen
- Reiter
- PopUps
- Chatfunktion
- Newsletter

Ausklappbare Boxen

Was ist das?

Reiter

Beschreibung

PopUps

Beschreibung

Chatfunktion

Beschreibung

Newsletter

Beschreibung

3. Konzept, welche Themen das CMS abdecken soll

Fokusthemen

Datenverwaltung

Dies ist eine klassische Datenverwaltung mit.

- Benutzer
  - Über den Benutzer werden Zugriffe und Informationsbeschränkungen reguliert.
- Assets (Video, Bild, Datei)
  - Ein Asset ist ein Inhalt
- Artikel
  - Artikel ist ein Inhalt
  - Ein Artikel enthält Assets und Texte
- Seiten
  - Eine Seite enthält untergeordnete Seiten
  - es gibt eine Verknüpfung über eine konfigurierte Navigation
- Befragungen
  - Felder, Fragen und Antworten
  - Auswertungsmöglichkeiten
- Extract (? Klärung)
- etc.

Visueller Seiteneditor

Der Visuelle Editor ermöglicht es, die Seite, die bearbeitet wird, auch gleich zu sehen. Wenn also eine neue Komponente auf die Seite gebracht ist, dann wird sie sofort dort angezeigt. Wenn es keine Inhalte gibt, wird ein Platzhalter angezeigt.

- Bearbeitung mit Drag n'Drop
- Eigenschaften der konfigurierbaren Komponenten ermöglichen die Anpassung des Designs.
- Einige Globale Einstellungen sollen hier auch möglich sein, wie z.B. einen kompletten Satz CSS
- HTML-Element Hierarchie wird auf Grund der technischen Notwendigkeit vorgegeben

Befragungen / Dynamische Formulare

- Befragungen werden sich aus dynamischen Formularen generieren lassen.
- Befragungen sollen auch aus externer Quelle eingebunden werden können (???).

Authentifizierung

Für die Authentifizierung wird Kaycloack verwendet. Weiterführend ist für Geste eine Möglichkeit zu schaffen an verschiedenen Interaktionen auch Anonym teilzunehmen.

Plugin Unterstützung

- es wird eine Möglichkeit geben Plugins einzubinden.
- Angedacht ist "per IFrame" oder per implementierung.
- für die Implementierung wird eine technische Dokumentation erstellt.

Technologien

- im backend wird Rust ([Framework XYZ]) zum Einsatz kommen.
- Im Frontend kommt SvelteKit zum Einsatz
- Design Konzept basiert auf der "Skeleton UI"
    - die Flexibilität von tailwind wird von der "Skeleton UI" eingebunden
    - die Kontrolle des Designs ist damit dem Ersteller einer Seite ermöglicht

4. DEMO

Beispiel einer technischen Doku.

Endpunkt A

```
{
  "name": "String",
  "vorname": "String"
}
```

CMSP-1

4.1. API - Objekte und Typen

Hier ist eine Liste der Objekte, die in der API - Schnittstelle aufgelistet werden.

- Asset
- ComponentConfig
- Content
- ContentText
- Domain
- ErrorResponse
- Navigation
- NavigationItem
- Page
- Site
- Timestamp

Asset

Das Asset ist die Repräsentation einer Datei. Wobei dazu intern noc mehr gehören kann. z.B. Thumbnails für Videos und Bilder.

> Details zusätzlicher Metadaten müssen noch bestimmt werden.

```
{
  "id": "uuid",
  "type": "string",
  "label": "Bezeichnung"
}
```

type

Der Type ist element aus ["video","image","file"].

ComponentConfig

Die Komponente ist der Baustein, aus dem alles mit Inhalten zusammengesetzt wird. Mit der ComponentConfig werden Komponenten dynamisch zusammengestellt und mit Inhalt gefüllt.

> Die Inhalte werden das Objekt noch einmal Verändern. Zuerst bekommen alle ein "Lorem Ipsum..." bis die Konfiguration definiert ist.

```
{
  "id": "uuid",
  "type": "component-type",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

type

Der "type" bestimmt welcher Type Komponente hier konfiguriert wird.
z.B.

- Accordion
- Carussel
- Artikel
- Gallery
- etc

cssClasses | []

Die "cssClasses" sind manuelle Eingaben des Nutzers.

cssStyles | []

Die "cssStyles" sind manuelle Eingaben des Nutzers.

contents | []

Die "contents" sind die Inhalte der Komponente.

Content

Ein Content ist ein Inhalt. Diese Inhalte können in Komponenten eingefügt werden.

Ein Inhalt kann folgendes sein:

- ContentText Freie Texte.
- Asset Bilder, Videos, Dateien zum Download
- Component Andere Komponenten

ContentText

Inhalts-Texte sind Text-Abschnitte, die man als Inhalt hinzufügen kann.

> Als späteres Feature kann man darüber eine Übersetzung von Inhalten realisieren.

```
{
  "id": "uuid",
  "content": "Lorem Ipsum Dolor"
}
```

Domain

Eine Domain ist die Adresse, unter der die Seite erreichbar sein soll.
Eine Seite kann mehrere domains haben, braucht aber mindestens eine.

> Syntax: http://[domain]/
> Beispiel: http://www.name-der-domain.de/

```
{
  "id": "uuid",
  "siteId": "uuid",
  "name": "www.name-der-domain.de"
}
```

ErrorResponse

Das Standard-Fehler-Objekt dient zur Vereinheitlichung der Fehlerbehandlung.

```
{
  "code": "error-code",
  "messageCode": "message-code",
  "violations": [{
    "sourceId": "field-name-or-identifier-of-source-inside-the-request",
    "messageCode": "message-code"
  }]
}
```

violations

Das Attribut "violations" kann leer sein "[]".

violations.sourceId

Die "sourceId" ist als Identifizierungsmerkmal für eine Fehlerquelle gedacht, wo es mehrere in einem Request geben kann. Üblicher Weise ist es der "fieldName" in Formularen. In Formularen können denen mehrere Felder gleichzeitig invalide sein. Das soll dem Anwender angezeigt werden können.

messageCode und violations.messageCode

Der messageCode soll als Code übertragen werden um damit übersetzungen zu ermöglichen.

Navigation

Die Navigation dient als Ankerpunkt um die Struktur der Navigationsitems (idr. Links) eingehängt werden und die dann in eine dafür geeignete Komponente eingehängt werden kann (Header, Sidebar, etd).

```
{
  "id": "uuid",
  "siteId": "uuid",
  "label": "Bezeichnung"
}
```

NavigationItem

Das "NavigationItem" stellt einen Link innerhalb der Navigation dar.

```
{
  "id": "uuid",
  "navigationId": "uuid",
  "targetPageId": "uuid",
  "label": "Bezeichnung"
}
```

Page

Die "Page" ist eine einzelne unterseite. Es kann mehrere Pages unter einer Site geben. Um irgendetwas Anzuzeigen, benötigt man mindestens eine Page auf einer Site.

```
{
  "id": "uuid",
  "siteId": "uuid",
  "name": "string",
  "titel": "Page title",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

name

Der "name" ist das, was man dann in der URL sieht.
Da es Teil der URL ist muss der Name unique pro seite sein.

> Beispiel: https://www.domain.de/[name]

title

Der "title" ist der Teil des `<title>` TAG's.

> Aufbau: "[site.title] - [page.title]"

contents | []

Die "contents" sind die Inhalte der Komponente.

Site

Die Site ist ein Web-Auftritt. Die Site enthält 1-n Pages.

```
{
  "id": "uuid",
  "label": "Seitenname im CMS sichtbar",
  "title": "Titel, der auf der Seite als Standard angezeigt wird => HTML<title>"
}
```

title

Der "title" ist der Teil des `<title>` TAG's.

> Aufbau: "[site.title] - [page.title]"

Time

Zeitstempel werden nach ISO 8601 übertragen.
Beispiele:

- YYYY-mm-dd HH:mm:ss.ms[+/-]00X
- Termin: 2023-09-24 05:45
- Datum: 2023-09-24
- Uhrzeit: 05:45

Angenommen wird als default-Timezone "europe/berlin".

Im CMS werden folgende Zeit-Typen verwendet:

- timestamp: "2023-09-24 05:45:00"
- date: "2023-09-24"
- time: "05:45:00"

4.2. API - Schnittstelle

API - Endpunkte

- ComponentConfig
- ContentText
- Domain
- Navigation
- NavigationItem
- Page
- Site
- Version

Grundsätzliches

`GET` Requests haben reguläre URL-Parameter als parameter.
Bespiel: `/api/endpoint?foo=bar&id=2`

`POST/PUT/DELETE` requests transportieren die Parameter im body als `JSON` objekt. Ausnahme ist die ID, welche das zu modifizierende Entity referenziert.
Beispiel `DELETE /api/endpoint/[id]`

Responses sind im Regelfall `JSON` objekte.

[Entity]ListResponse

ListResponses sind immer gleich aufgebaut.

```
{
  "items": [
    {
      "id": "uuid"
    }
  ],
  "total": 0
}
```

- `items` beinhaltet die Liste der abgefragten Elemente. Das kann bei Serve-Seitiger Pagination auch eine Teilmenge der abgefragten Elemente sein.
- `total` die Gesamtanzahl der gefundenen Elemente, die abgefragt wurde.

component

> WICHTIG: Feld "contents" für diese Iteration ignorieren. Die Definition der API dafür kommt mit der nächsten iteration.

> GET /api/component-config

Request

```
{}
```

Response

```
{
  "items": [
    {
      "id": "uuid",
      "type": "component-type",
      "cssClasses": [
        "class-1",
        "class-2",
        "etc"
      ],
      "cssStyles": [
        "background-color: #CCCCCC;",
        "font-size: 1em;",
        "etc"
      ],
      "contents": []
    },
    {
      "id": "uuid",
      "type": "component-type",
      "cssClasses": [
        "class-1",
        "class-2",
        "etc"
      ],
      "cssStyles": [
        "background-color: #CCCCCC;",
        "font-size: 1em;",
        "etc"
      ],
      "contents": []
    }
  ],
  "total": 2
}
```

> GET /api/component-config/[id]

Request

```
{}
```

Response

```
{
  "id": "uuid",
  "type": "component-type",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

> POST /api/component-config

Request

```
{
  "type": "component-type",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

Response

```
{
  "id": "uuid",
  "type": "component-type",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

> PUT /api/component-config/[id]

Request

```
{
  "type": "component-type",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

Response

```
{
  "id": "uuid",
  "type": "component-type",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

DELETE /api/component-config/[id]

Request

```
{
}
```

Response

```
{
}
```

Tickets: PA-6

ContentText

GET /api/content-text

Request

```
{}
```

Response

```
{
  "items": [
    {
      "id": "uuid",
      "content": "Lorem Ipsum Dolor"
    },
    {
      "id": "uuid",
      "content": "Lorem Ipsum Dolor"
    }
  ],
  "total": 2
}
```

GET /api/content-text/[id]

Request

```
{}
```

Response

```
{
  "id": "uuid",
  "content": "Lorem Ipsum Dolor"
}
```

POST /api/content-text

Request

```
{
  "content": "Lorem Ipsum Dolor"
}
```

Response

```
{
  "id": "uuid",
  "content": "Lorem Ipsum Dolor"
}
```

> PUT /api/content-text/[id]

Request

```
{
  "content": "Lorem Ipsum Dolor"
}
```

Response

```
{
  "id": "uuid",
  "content": "Lorem Ipsum Dolor"
}
```

> DELETE /api/content-text/[id]

Request

```
{
}
```

Response

```
{
}
```

> Tickets: PA-11

Domain

GET /api/domain

Request

```
{}
```

Response

```
{
  "items": [
    {
      "id": "uuid",
      "name": "www.hintertupfingen.de"
    },
    {
      "id": "uuid",
      "name": "www.oberamagau.de"
    }
  ],
  "total": 2
}
```

GET /api/domain/[id]

Request

```
{}
```

Response

```
{
  "id": "uuid",
  "name": "www.first-domain.de"
}
```

POST /api/domain

Request

```
{
  "siteId": "uuid",
  "name": "www.new-domain.de"
}
```

Response

```
{
  "id": "uuid",
  "name": "www.new-domain.de"
}
```

PUT /api/domain/[id]

Request

```
{
  "siteId": "uuid",
  "name": "www.new-value-for-domain.de"
}
```

Response

```
{
  "id": "uuid",
  "siteId": "uuid",
  "name": "www.new-value-for-domain.de"
}
```

DELETE /api/domain/[id]

Request

```
{
}
```

Response

```
{
}
```

Tickets: PA-6

Navigation

GET /api/navigation

Request

```
{}
```

Response

```
{
  "items": [
    {
      "id": "uuid",
      "label": "Bezeichnung"
    },
    {
      "id": "uuid",
      "label": "Bezeichnung"
    }
  ],
  "total": 2
}
```

GET /api/navigation/[id]

Request

```
{}
```

Response

```
{
  "id": "uuid",
  "label": "Bezeichnung"
}
```

POST /api/navigation

Request

```
{
  "label": "Bezeichnung"
}
```

Response

```
{
  "id": "uuid",
  "label": "Bezeichnung"
}
```

PUT /api/navigation/[id]

Request

```
{
  "label": "Bezeichnung"
}
```

Response

```
{
  "id": "uuid",
  "label": "Bezeichnung"
}
```

DELETE /api/navigation/[id]

Request

```
{
}
```

Response

```
{
}
```

Tickets: PA-12

NavigationItem

GET /api/navigation-item

Request

```
{}
```

Response

```
{
  "items": [
    {
      "id": "uuid",
      "navigationId": "uuid",
      "targetPageId": "uuid",
      "label": "Bezeichnung"
    },
    {
      "id": "uuid",
      "navigationId": "uuid",
      "targetPageId": "uuid",
      "label": "Bezeichnung"
    }
  ],
  "total": 2
}
```

GET /api/navigation-item/[id]

Request

```
{}
```

Response

```
{
  "id": "uuid",
  "navigationId": "uuid",
  "targetPageId": "uuid",
  "label": "Bezeichnung"
}
```

POST /api/navigation-item

Request

```
{
  "navigationId": "uuid",
  "targetPageId": "uuid",
  "label": "Bezeichnung"
}
```

Response

```
{
  "id": "uuid",
  "navigationId": "uuid",
  "targetPageId": "uuid",
  "label": "Bezeichnung"
}
```

PUT /api/navigation-item/[id]

Request

```
{
  "navigationId": "uuid",
  "targetPageId": "uuid",
  "label": "Bezeichnung"
}
```

Response

```
{
  "id": "uuid",
  "navigationId": "uuid",
  "targetPageId": "uuid",
  "label": "Bezeichnung"
}
```

DELETE /api/navigation-item/[id]

Request

```
{
}
```

Response

```
{
}
```

Tickets: PA-13

Page

> WICHTIG: Feld "contents" für diese Iteration ignorieren. Die Definition der API dafür kommt mit der nächsten iteration.

> GET /api/page

Request

```
{}
```

Response


```
{
  "items": [
    {
      "id": "uuid",
      "siteId": "uuid",
      "name": "string",
      "titel": "Page title",
      "cssClasses": [
        "class-1",
        "class-2",
        "etc"
      ],
      "cssStyles": [
        "background-color: #CCCCCC;",
        "font-size: 1em;",
        "etc"
      ],
      "contents": []
    },
    {
      "id": "uuid",
      "siteId": "uuid",
      "name": "string",
      "titel": "Page title",
      "cssClasses": [
        "class-1",
        "class-2",
        "etc"
      ],
      "cssStyles": [
        "background-color: #CCCCCC;",
        "font-size: 1em;",
        "etc"
      ],
      "contents": []
    }
  ],
  "total": 2
}
```

GET /api/page/[id]

Request

```
{}
```

Response

```
{
  "id": "uuid",
  "siteId": "uuid",
  "name": "string",
  "titel": "Page title",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

POST /api/page

Request

```
{
  "siteId": "uuid",
  "name": "string",
  "titel": "Page title",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

Response

```
{
  "id": "uuid",
  "siteId": "uuid",
  "name": "string",
  "titel": "Page title",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

PUT /api/page/[id]

Request

```
{
  "siteId": "uuid",
  "name": "string",
  "titel": "Page title",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

Response

```
{
  "id": "uuid",
  "siteId": "uuid",
  "name": "string",
  "titel": "Page title",
  "cssClasses": ["class-1", "class-2", "etc"],
  "cssStyles": ["background-color: #CCCCCC;", "font-size: 1em;", "etc"],
  "contents": []
}
```

DELETE /api/page/[id]

Request

```
{
}
```

Response

```
{
}
```

Tickets: PA-10

Site

GET /api/site

Request

```
{}
```

Response

```
{
  "items": [
    {
      "id": "uuid",
      "label": "Seite eins, Kunde XYZ",
      "title": "XYZ Seite - [subtitle]"
    },
    {
      "id": "uuid",
      "label": "Seite eins, Kunde XYZ",
      "title": "XYZ Seite - [subtitle]"
    }
  ],
  "total": 2
}
```

GET /api/site/[id]

Request

```
{}
```

Response

```
{
  "id": "uuid",
  "label": "Seite eins, Kunde XYZ",
  "title": "XYZ Seite - [subtitle]"
}
```

POST /api/site

Request

```
{
  "label": "Seite eins, Kunde XYZ",
  "title": "XYZ Seite - [subtitle]"
}
```

Response

```
{
  "id": "uuid",
  "label": "Seite eins, Kunde XYZ",
  "title": "XYZ Seite - [subtitle]"
}
```

PUT /api/site/[id]

Request

```
{
  "label": "Seite eins, Kunde XYZ",
  "title": "XYZ Seite - [subtitle]"
}
```

Response

```
{
  "id": "uuid",
  "label": "Seite eins, Kunde XYZ",
  "title": "XYZ Seite - [subtitle]"
}
```

DELETE /api/site/[id]

Request

```
{
}
```

Response

```
{
}
```

Tickets: PA-5

Version

GET /api/version

Request

```
{}
```

Response

```
{
    "version": "0.23.12"
}
```

Tickets: PA-6

4.3. Versionierung

Wir verwenden für die Nummerierung von Versionen ein etabliertes System. Wiki - Versionsnummer

Wobei wir die Build-Nummer aktuell nicht verwenden.

- Haupt-Versionsnummer wird dann inkrementiert, wenn ein Braking-Change passiert. Funktionen können hier entfernt werden, ersetzt oder Konzepte geändert.
- Neben-Versionsnummer wird dann inkrementiert, wenn Features fertiggestellt und hinzugefügt werden. Alte funktionen können erweitert werden. Wichtig ist hier: Alte Funktionalitäten werden nicht gebrochen. No Braking-Change.
- Revisionsnummer wird dann inkrementiert, wenn Bugfixes oder leichte Verbesserungen der User-Erfahrung vorgenommen werden. Hier werden keine neuen Funktionen hinzugefügt.

5. Download

Hier kann die aktuelle Spezifikation als PDF heruntergeladen werden:

Spezifikation als PDF