Dokumentation Content Management Systeme

Inhalt

[Zielsetzung 1](#_Toc535768407)

[Besonderheiten des Projektes 1](#_Toc535768408)

[Funktionsweise für die Nutzer 1](#_Toc535768409)

[Projektstruktur + Funktionen 1](#_Toc535768410)

[Frontend 1](#_Toc535768411)

[Backend: PHP 1](#_Toc535768412)

[Allgemeine Struktur und Aufbau des Backends 1](#_Toc535768413)

[Funktionen eines Bereichs 2](#_Toc535768414)

[Besonderheiten des Backends 2](#_Toc535768415)

[Backend: Datenbank 3](#_Toc535768416)

[Ausblick 4](#_Toc535768417)

# Zielsetzung

# Besonderheiten des Projektes

Dieses Content Management System ist einem kommerziell verwendbaren Endprodukt so nahe wie möglich gekommen. Hierzu gehören nicht nur der Funktionsumfang, die Fehlertoleranz gegenüber Benutzereingaben sowie die Nutzerauthentifizierung sondern auch der Modulare Aufbau und die Sicherheit.

# Funktionsweise für die Nutzer

# Projektstruktur + Funktionen

Die Projektstruktur besteht aus einem „root“-Ordner, in dem die Html und Javaskript Dateien abgelegt sind.

## Frontend

## Backend: PHP

### **Allgemeine Struktur und Aufbau des Backends**

Um eine saubere Struktur festzulegen und eine einfache Erreichbarkeit der benötigten Daten zu ermöglichen, wurde das Backend in mehrere Bereiche aufgeteilt. Hierzu zählen die Bereiche **Nutzer, Projekte, Kategorien, Theme/ Layout, Sprache und Admin**. Diese Bereiche unterteilen sich zusätzlich in eine Router- und Funktionsdatei.

(Anmerkung: Für eine schnelle Operation mit der Datenbank wurde ein zusätzlicher PHP-Bibliothek geschrieben, welcher allgemeine und oft verwendete Datenbankzugriffsmethoden beinhaltet. Einige Beispiele sind: Datenbankverbindung herstellen, Prüfen ob ein Datenbankeintrag existiert, Liefere einen bestimmten Eintrag zurück, Dateiupload, Starten/Hinzufügen zu/Beenden einer Transaktion[[[1]](#footnote-2)], Führe SQL-Befehl aus, etc.)

#### **Routen eines Bereichs**

Unter dem Begriff „Route“ versteht man hauptsächlich einen festgelegten Weg. Hierbei war der Hintergedanke eine Möglichkeit zu schaffen viele Funktionalitäten in PHP zu implementieren, ohne eine Vielzahl von Dateien erstellen/anlegen zu müssen. Eine Routerdatei kann mit Hilfe einer AJAX-Anfrage angesteuert werden. Grundsätzlich stellt diese Datei zu Beginn immer eine Datenbankverbindung her, da jede der weiteren aufgerufenen Funktionen diese benötigt. Zusätzlich muss mindestens eine Funktionsdatei eingebunden werden. Weiter erwartet eine Routerdatei immer einen Post-Parameter unter dem Namen „route“, anhand dessen wird der „Weg“ festgelegt(Also die Route zum Aufruf bestimmter Funktionen in einer Funktionsdatei), um explizit an ein bestimmtes Ergebnis zu gelangen. Nachdem die angepeilte/n Funktion/en ausgeführt wurde/n und das entsprechende Ergebnis zurückgeliefert hat/haben, wird es in das richtige Format für das Frontend gebracht und an das Frontend zurückgegeben.

### **Funktionen eines Bereichs**

In einer Funktionsdatei werden sämtliche Funktionen gesammelt, welche die benötigten Datenbankabfragen/-manipulationen eines Bereichs ausführen. Als eines der wichtigsten Elemente zählt hier die Datenbankverbindung, diese stellt sicher, dass die Abfragen/Manipulationen auf der richtigen Datenbank stattfinden. In den Funktionen selbst werden diese Zugriffe anhand von einem oder mehreren aufeinander abgestimmten SQL-Befehlen durchgeführt. Aufeinander abgestimmt bedeutet, dass die Befehle in der richtigen Reihenfolge ausgeführt werden. Ein Beispiel hierfür wäre: Eine Sprache mit Sprachelementen soll angelegt werden, dazu sollen die Sprache und Sprachelemente in verschiedenen Tabellen eingetragen werden. Werden nun zuerst die Sprachelemente angelegt, so können sie zunächst nicht „wissen“ zu welcher Sprache sie gehören. Deswegen sollte zuerst die Sprache hinzugefügt werden, dessen ID extrahiert und beim Eintragen der Sprachelemente mitgegeben werden, um ihre Zugehörigkeit festzulegen. Wurde die Funktion ausgeführt, so wird das Abfrage-Ergebnis oder der Manipulations-Erfolg an die aufrufende Routerdatei zurückgegeben.

### **Besonderheiten des Backends**

#### **Token-Mechanik**

Versucht sich ein Nutzer anzumelden wird ein Skript zur Erstellung eines Tokens[[[2]](#footnote-3)] aufgerufen. Hier wird zunächst überprüft, ob der Nutzer in der Datenbank eingetragen ist, also „existiert“. Sollte dies der Fall sein und der Nutzername mit dem Passwort übereinstimmen, so erfolgt die Generierung des Tokens. Dieses setzt sich in diesem Projekt aus drei Elementen zusammen. Das erste Element ist der Nutzertyp aufgeteilt in Admin, Redakteur und Nutzer. Im zweiten Element befindet sich ein Zeitstempel zur Erstellungszeit des Tokens. Um letztendlich eine schnelle Identifizierung des Nutzers zu ermöglichen, wird die Nutzer-ID im dritten Element gespeichert. Das Token wird anschließend in einer Session-Variable der aktuellen Session des Nutzers mit dem Backend abgelegt. Auf Session-Variablen kann im Backend-Skriptübergreifend zugegriffen werden. Anhand des Tokens kann ab diesem Zeitpunkt immer eindeutig festgestellt werden, welcher Nutzer gerade mit dem Backend kommuniziert. Im Frontend wird diese Form der Authentifizierung zusätzlich genutzt, um nach Abfrage der Gültigkeit des Tokens im Backend festzustellen, ob der derzeitige Nutzer die Rechte für den Zugang zu bestimmten Inhalten besitzt. Die hierfür benötigte Überprüfung findet in der sogenannten „Middleware[[[3]](#footnote-4)]“ statt.

(Anmerkung: Das Token wird wegen der schnellen Authentifizierung des Nutzers in vielen Bereichen der Datenbankabfrage/-manipulation genutzt. Ein Beispiel wäre hierfür: Will ein Nutzer die Projekte an denen er teilgenommen hat laden, so wird im Backend die Nutzer-ID aus dem Token gelesen und anhand dessen werden die entsprechenden Projekte zurückgeliefert)

#### **Logout**

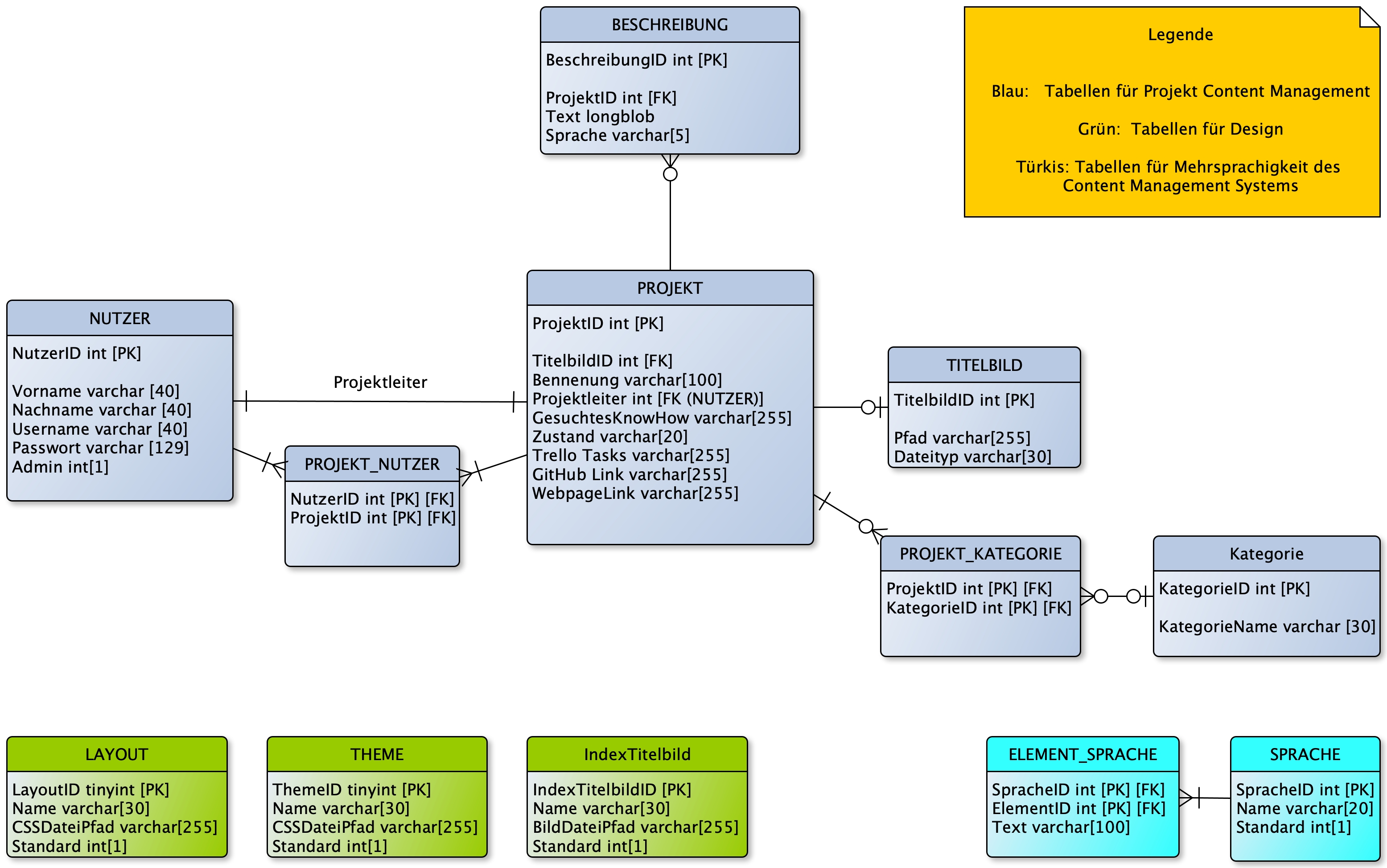
Zur Realisierung des Logouts wurde das Token genutzt. Dabei wird das Token gelöscht und die Session beendet. Anschließend ist der Nutzer nicht mehr identifizierbar. Das heißt, in der Middleware wird er als normaler Besucher registriert.

#### **Middleware**

Um relevante Unterschiede der Bedienungsmöglichkeiten für verschiedene Nutzern erwirken zu können, wird eine strikte Einteilung durch das System benötigt. In Folge dessen fiel die Wahl darauf verschiedene Nutzertypen einzuführen. Damit überflüssige und häufige Überprüfungen des Nutzertyps ausgeschlossen werden, lag es nahe, eine Middleware zu implementieren die speziell darauf ausgelegt ist das Token zu überprüfen. In dieser Middleware kann ein Modus-Parameter übergeben werden, welcher angibt, auf welche Kriterien überprüft werden soll. Hauptsächlich kann hier unterschieden werden, ob es sich um einen Besucher, registrierten Nutzer, Redakteur oder Admin handelt. Zusätzlich wird das Token anhand des Zeitstempels überprüft ob es noch gültig ist. War die Überprüfung durch die Middleware erfolgreich, so wird der Erfolgsstatus und, wenn verfügbar, der Nutzertyp zurückgegeben. Ansonsten gilt die Prüfung als fehlgeschlagen.

## Backend: Datenbank

Die Datenbank ist in drei Bereiche gegliedert. Diese entsprechen dem jeweiligen Verwendungszweck. Dazu gehört die eigentliche Kontent Verwaltung, das Erscheinungsbild des Content Management Systems und die Mehrsprachigkeit des Systems. Diese Aufteilung in Bereiche bietet eine einfache Verständlichkeit. Zusätzlich wurde eine **einheitliche Strukturierung und Benennung** verwendet, z.B. die Spalte „Standard“ beinhaltet, ob das zugehörige Tupel aktuell verwendet werden soll oder nicht. Auch dies verbessert die Verständlichkeit und ermöglicht ein **modulartieges Erweitern** der Datenbank um neue Aspekte.



Der Kontent-Bereich besteht aus den zwei Haupt-Tabellen Nutzer und Projekt. Ein Nutzer kann dabei als Projektleiter oder als Projektnutzer an einem Projekt teilnehmen. Zwecks des **Sicherheitsaspekts** ist das Passwort des Nutzers bereits im Frontend mit **SHA512** gehasht worden! Ein Projekt kann dabei maximal einen Projektleiter aber mehrere Projektteilnehmer umfassen. Das Projekt kann durch die *1 zu 0 bis n* Beziehung mit Beschreibung für mehrere Sprachen eine Projektbeschreibung haben (im Frontend noch nicht umgesetzt). Zudem kann ein Projekt mehreren Kategorien zugeordnet werden mittels der Beziehungsentität PROJEKT\_KATEGORIE.

Das Erscheinungsbild beinhaltet drei Tabellen, eine für das Titelbild der Website, eine für das Layout und eine für das Theme (Farbauswahl), wobei alle die selbe Struktur aufweisen. Diese ermöglicht eine Wiederverwendung von PHP-Funktionen und unterstützt den **modulartigen Aufbau** des Gesamtsystems.

Die Mehrsprachigkeit wird mittels zweier Tabellen ermöglicht. In der ersten wird für jede dem System hinzugefügten Sprache **(Maximal unterstützte Anzahl an Sprachen = Zahlenbereich von INT/Anzahl Elemente der Website die Mehrsprachig sind)** eine ID zugewiesen. Mittels dieser ID kann jeder Eintrag in der Tabelle ELEMENT\_SPRACHE einer eindeutigen Sprache zugeordnet werden. In ELEMENT\_SPRACHE selbst ist für jede *SpracheID*, für jedes Element im Html-Dokument der anzuzeigende Text hinterlegt.

# Ausblick

Dieses Content Management System ist einem kommerziell verwendbaren Endprodukt so nahe wie möglich gekommen. Mit Hilfe der implementierten Middleware ist ein Schutz gegen Angriffe wie Code Injection und Cross-Side-Skripting bereits vorhanden. Zur kommerziellen Verwendung sind noch zwei Schritte nötig. Zum einen kann noch eine professionelle Abwehr gegen SQL-Injections hinzugefügt werden, zum anderen wäre ein modernes Design angedacht. Dieses könnte mit den Theme- und Layout-Upload Funktionen ohne eine nötige Änderung am Quellcode erfolgen, z.B. erstellt von einem externen Webapplication-Designer!

1. Transaktion: Bei einer Transaktion handelt es sich um eine Reihe von Datenbankzugriffen(-manipulationen), die zunächst provisorisch ausgeführt werden, jedoch bei Abbruch der Transaktion rückgängig gemacht werden. [↑](#footnote-ref-2)
2. Token: *Zeitlich begrenztes Element zur Erkennung eines Nutzers, welches zusätzlich Informationen zu dem entsprechenden Nutzer beinhalten kann*. [↑](#footnote-ref-3)
3. Middleware: *Skript der vor dem Aufruf von Funktionen durchlaufen werden muss, um zusätzliche Funktionen auszuführen. Steht meist zwischen Front- und Backend. Zum Beispiel um zusätzliche Nutzerspezifische Inhalte hinzuzufügen, oder um festzustellen, ob eine gewünschte Funktion ausgeführt werden darf*. [↑](#footnote-ref-4)