Softwaretechnik-Praktikum 2015

Entwurfsbeschreibung

Virtuelles Partizipatorisches Museum

1. Allgemein

Mit dem Blog “Fundstücke nachhaltiger Nachbarschaft“ existiert ein digitales Online-Museum, welches sich mit Gegenständen aller Art künstlerisch auseinandersetzt. Allerdings beschränkt die momentane Form das Angebot auf eine primär unidirektionale Präsentationsform.

Das Ziel ist es, unter Beibehaltung der initialen Idee eine Plattform zu schaffen, die die bestehende starre und einseitige Präsentationsform auflöst und durch verschiedenste Mittel zu Kommunikation und Partizipation einlädt. Weiterhin soll durch eine neue Filter- und Suchfunktion, welche besser auf die vorliegenden Datenstrukturen abgestimmt ist, die Verwendung der Seite angenehmer gestalten werden.

Dazu soll auf Basis von Wordpress in Kombination mit einer eigenen Plugin-Suite und ausgewählten Drittanbieter-Plugins eine möglichst robuste Lösung entwickelt werden, welche auch in Zukunft weiteren Spielraum für Erweiterungen lässt. Zusätzlich soll so der Bedienkomfort der Seite erhöht und der Wartungsaufwand minimiert werden.

2. Produktübersicht

Das Partizipatorische Virtuelle Museum ist als Gesamtprojekt eine Wordpress-Single-Site-Installation. Eine Reihe modifizierter Plugins stellen auf der Seite die gewünschten Funktionalitäten bereit. Zu Beginn sind außerdem bereits einige Einträge auf der Seite vorhanden, welche aus dem Vorgängerblog “Fundstücke nachhaltiger Nachbarschaft“ entstammen.

Als Besucher erreicht man die Seite im Regelfall auf der Startseite. Dort kann man neben News auch über verschiedene Auflistungen zu Objekten aus dem Museum gelangen. Als nicht angemeldeter Betrachter hat man Zugriff auf die Filterfunktion, mit deren Hilfe man die Objektmenge nach bestimmten Merkmalen durchsuchen und sortieren kann. Außerdem kann man auf den einzelnen Betrachtungsseiten die jeweiligen Objekte ansehen. Weitere zugängliche Funktionen sind die Registrierung, Anmeldung und Impressum- sowie Kontaktseite.

Über die Registrierung kann ein Besucher durch Angabe weniger Daten ein Benutzerprofil auf der Seite erstellen. Angemeldete Benutzer (nachfolgend als “User” bezeichnet) können weitere Funktionen der Plattform nutzen. Jegliche Partizipation an dem virtuellen Museum setzt ein Benutzerprofil voraus. User sind in der Lage neue Objekte einzustellen. Weiterhin können bestimmte Nutzer Objekte für die Zugehörigkeit zu einem Projekt markieren, sodass Objekte Gruppiert werden können.

Moderatoren und Administratoren erhalten die Möglichkeit Objekte zu Editieren oder Entfernen. Außerdem muss jede Einsendung eines Users erst freigeschalten werden, bevor sie im Frontend sichtbar wird.

3. Grundsätzliche Struktur- und Entwurfsprinzipien

Dieses Projekt basiert auf Wordpress. Zur Realisierung der benötigten Features soll, sofern möglich und angemessen, auf die native Wordpress-Funktionalität und auf Community-Plugins zurückgegriffen werden. Alle weiteren Features werden in Form von eigenen Erweiterungen umgesetzt. Diese Erweiterungen werden im PVMkit-Plugin zusammengefasst.

Aus Übersichtlichkeitsgründen und um die Updatefähigkeit von Wordpress zu wahren benutzt PVMkit ein eigenes Datenmodell, das die Wordpressdatenbank um mehrere Tabellen erweitert.

PVMkit hat eine eigene Verzeichnis- bzw. Dateistruktur. Im Hauptverzeichnis **pvmkit** befinden sich die Dateien **pvmkit.php**, **pvmkit_functions.php** und **pvmkit_install.php** sowie die Ordner **admin** und **includes**.
Der **admin**-Ordner enthält Funktionen und Erweiterungen, die gänzlich oder zum überwiegenden Teil ins Backend einzuordnen sind, während im **includes**-Ordner die Frontend Funktionalität ihren Platz findet.

Logische Funktionen sind innerhalb dieser Ordner in separaten Dateien untergebracht. So enthält beispielsweise die Datei **pvmkit_viewer.php** (**includes**) den Code, der für das Anzeigen von Werken im Frontend verantwortlich ist. Die unter Punkt 4 genannten Pakete sind u.U. dateiübergreifend, d.h. ein Paket kann mehrere Dateien umfassen.

Die Datei **pvmkit.php** dient als Container, darin wird der Inhalt der **admin** und **includes** Ordner eingebunden.
pvmkit_functions.php enthält Funktionen, die keiner logischen Funktion konkret zugeordnet werden konnten bzw. die in mehreren Dateien inkludiert werden.
pvmkit_install.php ist für Anweisungen zur erstmaligen Installation des Plugins (Anlegen von Benutzerrollen, Erstellen von Datenbanktabellen) gedacht.

Das Darstellen von Werken und Projekten im Frontend wird über Shortcodes realisiert.
Dazu wird ein Wordpress-Post erstellt, der einen Shortcode mit der ID des jeweiligen Werks bzw. Projekts enthält.

Eingebundene Community-Plugins sollten nicht verändert werden. Falls eine Änderung im Code eines solchen Plugins unvermeidbar ist, wird diese als Patch eingespielt.

4. Struktur- und Entwurfsprinzipien einzelner Pakete

4.1 Erstellen und Verwalten von Werken und Projekten

Dateien: **pvmkit/admin/pvmkit_composition_manager_user.php**
pvmkit/admin/pvmkit_project_manager.php
pvmkit/admin/pvmkit_projects.php
pvmkit/admin/pvmkit_publish_compositions.php

Die Funktionalität des Einsendens neuer Werke durch Benutzer wird von einem Backend-Modul bereitgestellt. Der Einsendeprozess ist modular aufgebaut. Zunächst muss das Werk vom Benutzer durch das Hinzufügen der Einzelbestandteile (Titel, Text, Titelbild etc.) zusammengestellt werden. Dafür ist die Datei **pvmkit_composition_manager_user.php** verantwortlich. Darin werden etliche Formulare für diesen Zweck bereitgestellt.
Hochgeladene Bilder werden automatisch in drei Größen (small - 50x50, medium - 200x200, large - 600xVariabel) transformiert und unter **uploads/pvmkit** abgelegt. Der Dateiname wird automatisch erzeugt und setzt sich aus der element_id des Bildes und dessen Größenbezeichnung zusammen, z.B. **81_medium.jpg**.

Alternativ können Objekte (Bilder, Texte) auch im Frontend von anderen Werken durch einen Klick auf Bild verwenden bzw. Text verwenden übernommen werden. Bei Bildern wird lediglich die element_id mit dem neuen Werk verknüpft. Bei Texten hingegen wird der Text als neues Objekt kopiert, um ungewollte Modifikationen zu vermeiden.

Ist das Werk fertig, wird sein Status zu 3 geändert. Damit ist das Werk zur Moderation freigegeben. Diese Aufgabe übernimmt die Datei **pvmkit_publish_compositions.php**.
Die dortige Freischaltung setzt den Status auf 7. Es wird ein Wordpress-Post mit dem Shortcode *[pvmkit_viewer id=xx]* erstellt, woben xx für die ID des Werks steht. Das Werk ist damit veröffentlicht und wird im Frontend angezeigt.

pvmkit_projects.php stellt Funktionen zur globalen Verwaltung von Projekten und Projektleitern bereit, während **pvmkit_project_manager.php** Funktionen zur Verwaltung eigener Projekte enthält.

4.2 Darstellung von Werken und Projekten im Frontend sowie die Verwaltung gemeldeter Verstöße

Dateien: **pvmkit/includes/pvmkit_viewer.php**
pvmkit/includes/pvmkit_view_project.php
pvmkit/includes/pvmkit_list_widget.php
pvmkit/includes/pvmkit_filter.php
pvmkit/admin/pvmkit_report_manager.php

Werke werden mittels des Shortcodes *[pvmkit_viewer id=xx]* veröffentlicht, für das Abrufen der Daten aus der Datenbank ist **pvmkit_viewer.php** verantwortlich. Für Projekte wird der Shortcode *[pvmkit_view_project pid=xx]* verwendet, die zuständige Datei ist **pvmkit_view_project.php**.

pvmkit_list_widget.php erstellt eine Liste der neusten veröffentlichten Werke, indem die Datensätze in der **pvm_compositions** Tabelle nach dem Erstellungsdatum sortiert und die ersten 10 Einträge ausgelesen werden.

Das Formular zum Melden eines Verstoßes wird in **pvmkit_viewer.php** erzeugt und ist zunächst mittels des CSS-Tags display: none verborgen, bis der Link *“Einen Verstoß melden”* angeklickt wird.
Abgesendete Daten werden in der Tabelle **pvm_reported_compositions** gespeichert.

Gemeldete Versöße werden im Backend unter dem Punkt “Gemeldete Werke” (**pvmkit_report_manager.php**) gelistet und verwaltet.

4.3 Benutzerregistrierung und Authentifizierung, Benutzerprofil anzeigen und bearbeiten

Dateien und Ordner: **wp-members**
really-simple-captcha
pvmkit/includes/pvmkit_list.php
pvmkit/includes/pvm_user_profile.php

Für die Benutzerregistrierung und Authentifizierunurg sowie zur Verwaltung des Benutzerprofils kommt das Plugin wp-members zum Einsatz. Es bietet die Basisfunktionalität, die man von anderen Community-basierten Portalen gewohnt ist (Registrierungs- und Loginformular, Aktivierung des Accounts durch eine Bestätigungsemail, “Password vergessen”-Funktion etc.) und kann ohne direkte Veränderungen am Code aus dem Backend konfiguriert werden.

Zur Verhinderung automatischer Registrationen durch Bots wird wp-members durch das Plugin really-simple-captcha erweitert, da dieses speziell auf wp-members zugeschnitten ist.

Wp-members nutzt das Datenmodell von Wordpress und speichert registrationsbezogene Daten der Benutzer in der Tabelle **users** sowie weitere Metadaten in der Tabelle **usermeta.**

Das Benutzerprofil wird durch **pvmkit_list.php** in Verbindung mit **pvm_user_profile.php** um eine Liste der von dem jeweiligen Benutzer veröffentlichten Werke erweitert. **pvmkit_list.php** stellt dabei die Liste mittels eines Shortcodes (*[pvmkit_list]*) zur Verfügung und **pvm_user_profile.php** inkludiert diesen Shortcode.
Durch die Trennung dieser beiden Aufgaben ist es möglich, die Liste auch an anderen Stellen des PVM durch den Shortcode anzuzeigen.

4.4 Statistiken

Dateien: **pvmkit/admin/pvmkit_statistics.php**

pvmkit_statistics.php erzeugt unterschiedliche Statistiken zum PVM (Gesamtzahl der Werke, beliebtestes Werk, aktivster Benutzer etc.) im Backend unter dem Menüeintrag “Statistiken”.

4.5 Social-Media Buttons (Shariff)

Auf verschiedenen Seiten sollen (nicht kontextsensitive) Social Media Buttons für Aktionen wie Teilen und “Liken” angeboten werden. Damit sollen alle großen Netzwerke abgedeckt werden. Die Beachtung von Datenschutz spielt eine große Rolle.

Unter den wenigen Datenschutzkonformen Lösungen wird Shariff noch regelmäßig mit Updates versorgt. Außerdem handelt es sich um ein Wordpress-Plugin, welches somit einfach mit Updates versogt werden kann. Es kann sowohl als Widget in die Sidebar als auch als Shortcode eingebunden werden, womit die nötige Flexibilität gegeben ist. Shariff alle größeren Netzwerke.

4.5 Theme

Dateien und Ordner: **twentyfifteen-pvmkit/**
twentyfifteen-pvmkit/author.php
twentyfifteen-pvmkit/content.php
twentyfifteen-pvmkit/functions.php
twentyfifteen-pvmkit/index.php
twentyfifteen-pvmkit/single-project.php
twentyfifteen-pvmkit/single-search.php
twentyfifteen-pvmkit/style.css

Als Theme wurde Twenty Fifteen ausgewählt, da es sich durch Einfachheit im Design und eine unproblematische Modifizierbarkeitauszeichnet. Für Projektspezifische Änderungen des Themes wurde unter **twentyfifteen-pvmkit/** ein Child-Theme angelegt.

index.php und **content.php** folgen dem Wordpress Prinzip und stellen eine Liste sowie die Einzelbetrachtungsseite der Posts (Werke) dar.

In single-search.php und single-project.php sind, wie die Namen bereits verraten, Funktionen für den Suchvorgang sowie die Darstellung der Projektseiten untergebracht.

4.6 Erweiterungsmöglichkeiten

Grundlegende Veränderungen am Layout sollten innerhalb des Child-Themes geschehen. Designanpassungen gehören in **twentyfifteen-pvmkit/style.css.**

Zusätzliche Objekte können dem Werk durch das Erzeugen eines neuen Case in **pvmkit/admin/pvmkit_composition_manager_user.php** hinzugefügt werden. Das Datenmodell muss gegebenenfalls dem entsprechend angepasst werden.

5. Datenmodell

Das Datenmodell orientiert sich im Wesentlichen an einer Standard-Wordpress-Installation. Veränderungen werden daran nicht vorgenommen, um eine reibungslose Funktion und Updatefähigkeit des Grundsystems sicherzustellen.

Ergänzt wird der Datenbestand durch (zunächst) vier Tabellen, welche durch das Präfix “pvm_” gekennzeichnet sind:

pvm_compositions: Ein Datensatz dieser Tabelle repräsentiert ein Werk, welches von einem User eingesendet wurde.

pvm_objects: Ein Datensatz dieser Tabelle stellt ein Element eines Werkes da, also ein Text oder Bild. Dieses Element kann in mehreren Werken Verwendung finden.

pvm_projects: Ein Datensatz dieser Tabelle repräsentiert ein Projekt, welches von einem User initiiert wurde. Werke können maximal einem Projekt zugeordnet werden.

pvm_terms: Ein Datensatz dieser Tabelle stellt die Zuordnung des Tags (“term”) zu einem Werk (“composition_id” aus pvm_compositions) dar. Dabei handelt es sich technisch gesehen um eine m:n-Beziehung, welche allerdings bewusst nicht aufgelöst wurde. Im momentanen Projekt wäre der zu erwartende Mehraufwand in der Verwaltung ungünstiger als die zu erwartenden Kollisionen/Redundanz.

pvm_reported_compositions: Ein Datensatz dieser Tabelle stellt einen gemeldeten Verstoß dar. Die Zuordnung erfolgt über einen Fremdschlüssel auf die composition_id des gemeldeten Werks. Ein Betreff bzw. eine Kurzbeschreibung des Verstoßes (subject) ist obligatorisch, die detaillierte Beschreibung (description) ist fakultativ.

Die wesentliche Datenbasis stellen die Tabellen pvm_compositions (Werke) und pvm_objects (Elemente) dar. Einem Werk können über die Attribute text, image und picture aus pvm_compositions bis zu drei Elemente (aus pvm_objects) zugeordnet werden. Ist eines dieser Attribute nicht belegt, wird es auf NULL gesetzt. Als Untergrenze für ein Werk sind minimal zwei Elemente vorgesehen, was allerdings auf Seiten des Datenbanksystems nicht als Integritätsbedingung festgelegt ist. Damit ist es beispielsweise später möglich Teile eines Werkes zu Löschen und zur Überarbeitung an den Nutzer zu Melden.

Die Tabelle pvm_compositions erlaubt außerdem für weitere Attribute den NULL-Wert:

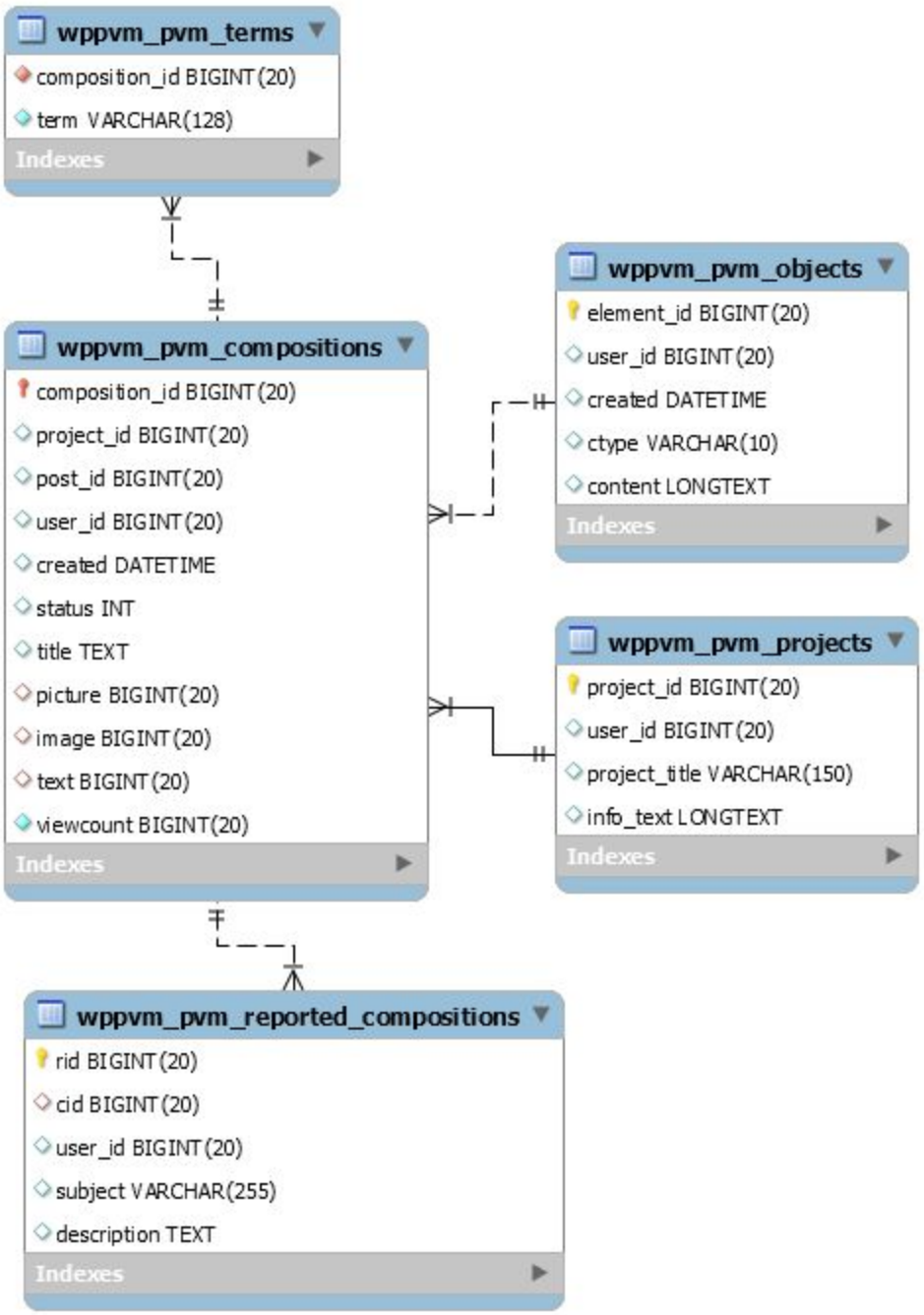
project_id: NULL bedeuted in diesem Fall, dass das Werk nicht explizit einem Projekt zugeordnet ist. Es impliziert im Gegenzug, dass Aufgaben wie die Tagauswahl oder Zulassung zu einem Projekt nicht an einen Projektleiter aus der Menge der User delegiert werden kann.

post_id: Dieses Attribut bezieht sich auf das ID-Attribut der posts-Tabelle von Wordpress. Wenn es NULL ist, ist der Post noch nicht veröffentlich oder gelöscht worden. Wenn es einen numerischen Wert besitzt, ist es der Post, unter dem das Werk in Wordpress veröffentlicht wurde. Eine eindeutige Information über den Status des Werkes ist über das Feld status zu beziehen (7 und größer: veröffentlicht).

Die in einigen Tabellen angegebenen user_id-Attribute beziehen sich auf die Wordpress-User-ID (Feld ID in der users-Tabelle von Wordpress).

Die Datentypen orientieren sich im Regelfall an den Werten, die auch Wordpress in seinem Datenmodell verwendet. Für IDs wird BIGINT(20) verwendet und Zeitpunkte als DATETIME.

Die Rückzuordnung eines Posts zum Werk ist auf semantischer Ebene gegeben durch den Parameter im Shortcode des post_content-Attributs. Durch die Einbindung als Shortcode wird eine Datenunabhängigkeit erreicht, sodass durch Austausch des Viewers beispielsweise das Aussehen komplett verändert werden kann.



Die fünf PVMkit-Tabellen mit Wordpress-Präfix “wppvm_”

6. Testkonzept

Die Entwicklung und Modifikation der einzelnen Softwarepakete wird von Einzelpersonen oder Zweierteams umgesetzt. Aufgrund der verwendeten Webtechniken gestaltet sich das Testen weitestgehend schwierig, da Debugschnittstellen weitestgehend fehlen oder nur begrenzt flexibel sind. Weiterhin ist es schwieriger automatische Tests einzurichten als die Tests manuell durchzuführen. Daher wird auf automatisierte Testverfahren verzichtet.

Aus den Voraussetzungen ergeben sich somit drei Testphasen. Die ersten beiden werden jeweils von dem beteiligten Entwickler durchgeführt. Die letzte hingegen ist als Aufgabe des gesamten Teams anzusehen.

Während der Entwicklung finden kontinuierlich Tests statt. Dabei obliegt es dem Entwickler eigene Testinstallationen zu verwenden oder auf die Projektmittel zurückzugreifen. Die Wordpress-Instanzen sind dabei i.d.R. mit einer aktuellen Version des Codes ausgestattet. Änderungen werden ohne Umweg über ein Repository in die Installation eingespielt. Die Test beschränken sich dann darauf zu prüfen, ob der Code die gewünschten Funktionen bereitstellt.

Die Arbeiten an einem Paket sind abgeschlossen, wenn die gewünschten Funktionen davon bereitgestellt werden und das Produkt funktionsfähig ist. Die geänderten Dateien werden dann ins Git-Repository übernommen und in einer weiteren Testinstallation ausgerollt. Diese ist wiederum gemäß dem Rollout-Dokument eingerichtet und simuliert eine Produktivumgebung. Hier finden weitere Test bzgl. der Verträglichkeit mit anderen Komponenten statt. Dazu werden verschiedene User-Stories abgearbeitet und Fehlverhalten provoziert (bspw. durch falsche URL-Parameter, etc.).

Nach Implementierung aller im Arbeitsplan aufgelisteten Muss-Funktionalitäten findet ein Gesamttest statt, bei dem alle Features noch einmal im Gesamtkontext des Projekts auf eine korrekte Funktion überprüft werden. Dazu werden mehreren Personen mit unterschiedlichen Konfigurationen (Browser, Endgerät, Betriebssystem) alle User-Stories mehrfach anwenden und außergewöhnliche Anwendungsfälle provozieren (fehlende/fehlerhafte An- und Eingaben). So soll das Zusammenspiel der verschiedenen Softwarepakete und deren Robustheit geprüft werden. Außerdem soll durch das Mehraugenprinzip sichergestellt werden, dass in den vorangegangenen Kurztests keine Fehler übersehen wurden.