Webanwendung "MyWiki"

Sie sollen eine Anwendung erstellen, die ein einfaches Wiki-System mit Themen und Kategorien implementiert.

Fachliche Anforderungen

Allgemeine Anforderungen

Das Wiki besteht aus einzelnen *Themen*, die im Inhaltsbereich angezeigt werden können. Die Themen enthalten Verweise auf andere Themen (als Link). Themen werden zunächst im Ansichtsmodus dargestellt.

Wenn in diesem Modus ein Link bedient wird, wird das im Link referenzierte Thema im Inhaltsbereich im Anzeigemodus dargestellt (und ersetzt damit das bisher angezeigte Thema). Falls es das referenzierte Thema noch nicht in der Datenbasis gibt, wird es automatisch neu erstellt:

- die Bezeichnung wird aus der im Link angegebenen Adresse abgeleitet
- Titel und Text werden vorbesetzt mit dem Text "`(noch anzugeben)`"
- die anderen Daten werden sinnvoll eingetragen.

(Man erstellt also neue Themen einfach dadurch, dass man in einem bestehenden Thema einen neuen Link einträgt und diesen dann im Ansichtsmodus bedient.)

Durch Wechsel in den Bearbeitungsmodus können Bezeichnung, Titel und Text eines Themas geändert werden. Zugleich können zu jedem Thema *Kategorien* angegeben werden. Im Ansichtsmodus sind werden die zugeordneten Kategorien als Links dargestellt und können bedient werden: damit wird eine Kategorie ausgewählt und mit den ihr zugeordneten Themen angezeigt (siehe auch "Spezielle Anfragen, Alle Kategorien" weiter unten).

Themen können nur im Bearbeitungsmodus gelöscht werden. Die Entfernung aller Referenzen auf ein Thema löscht das Thema **nicht**.

Themen können mit Hilfe der Zuordnung der Kategorie `favorites` in eine Favoritenliste aufgenommen werden (siehe auch weitere Beschreibungen). Wird diese Kategorie bei einem Thema entfernt, ist das Thema auch nicht mehr in der Favoritenliste vorhanden.

Spezielle Anfragen

Über eine Menüzeile sind folgende spezielle Anfragen ausführbar:

- *Home*: das spezielle Thema `home`
  - wird als Startseite angezeigt
  - kann wie alle anderen Themen bearbeitet werden
  - kann aber **nicht** gelöscht werden

  Die weiteren speziellen Anfragen führen zu Zusammenstellungen, die nicht direkt bearbeitet werden können; die aufgeführten Links sind aber bedienbar.

- *Favoriten*: Zusammenstellung, die alle Themen referenziert (Links), die der Kategorie `favorites` zugeordnet wurden
- *Alle Themen*: Zusammenstellung, die alle vorhandenen Themen referenziert (Links)
- *Alle Kategorien*: Zusammenstellung, die alle vorhanden Kategorien referenziert (Links)
  - die Bedienung eines Links wählt eine einzelne Kategorie aus und löst die Erstellung einer neuen Zusammenstellung aus:
    - weist als Titel die ausgewählte Kategorie aus
    - referenziert (Links) alle Themen, denen die ausgewählte Kategorie zugeordnet ist

- *Fehlende Themen*: Zusammenstellung, die alle zwar referenzierten, aber noch nicht in der Datenbasis erzeugten Themen referenziert (Links)
  - durch die Bedienung eines Links wird dann das jeweilige Thema neu erstellt
- *Verwaiste Themen*: Zusammenstellung, die alle vorhandenen, aber nicht referenzierten Themen referenziert (Links).

Diese speziellen Anfragen werden serverseitig ausgewertet und können wie Themen an den Client ausgeliefert werden; die unterschiedlichen Regelungen zur möglichen Bearbeitung müssen beachtet werden.

Datenmodell

Das Datenmodell umfasst die einzelnen Themen:

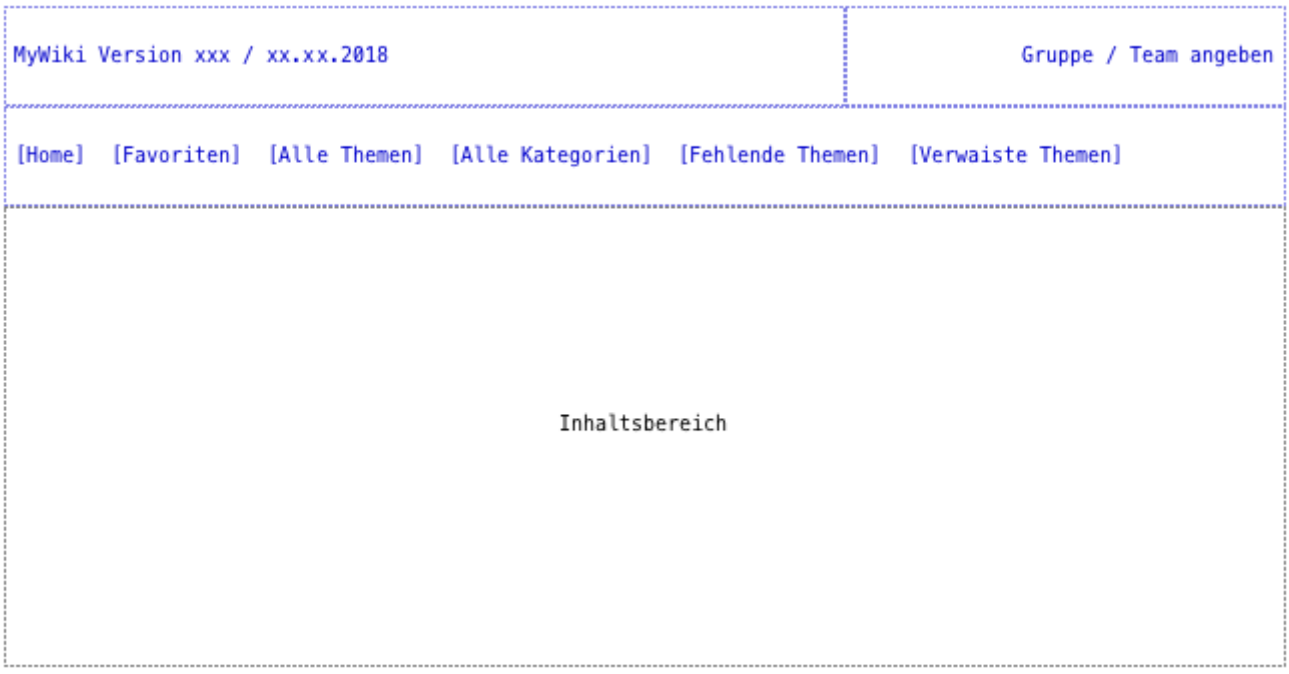
- zu jedem Thema werden gespeichert:
  - Bezeichnung
  - Titel
  - Text zum Thema
  - Datum und Uhrzeit der Erstellung
  - Datum und Uhrzeit der letzten Änderung
  - die zugeordneten Kategorien.

Die Bezeichnung wird auch zur Referenzierung verwendet, also in Links verwendet. Sie muss daher den formalen Anforderungen für URI folgen. Es ist empfehlenswert, in den Bezeichungen keine Sonderzeichen / Umlaute oder Leerzeichen zu verwenden.

Die Kategorien werden lediglich bei den Themen gespeichert.

Aufbau der Benutzungsschnittstelle

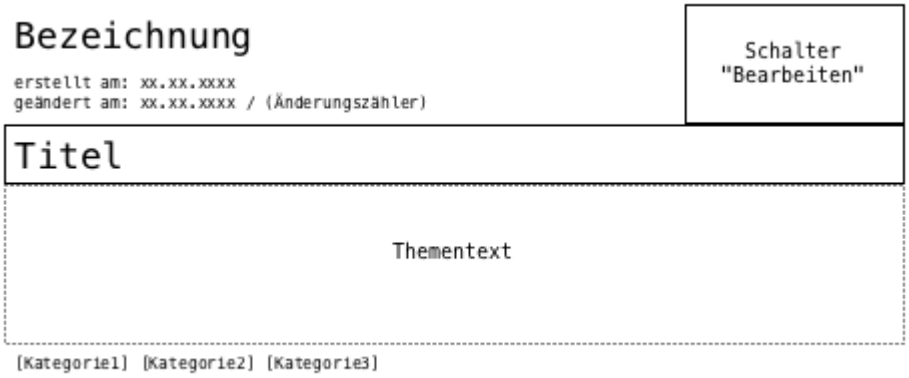
Die Benutzungsschnittstelle muss entsprechend der Darstellung der Wireframes aufgebaut werden:



Wireframe User Interface

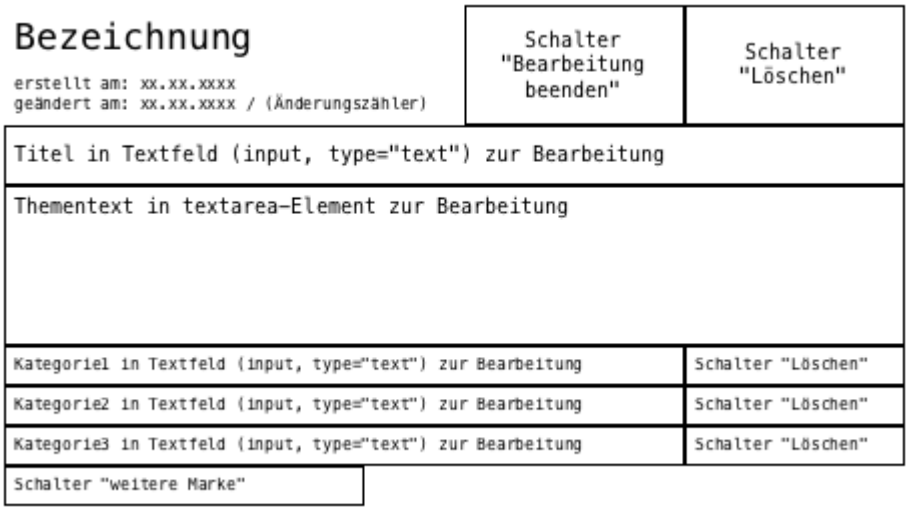
Beim Start der Anwendung wird die Anfrage *Home* ausgeführt.

Der Ansichts-Modus eines Themas ist im nachfolgenden Wireframe dargestellt.



Wireframe User Interface Thema (Ansicht)

Durch die Bedienung des Schalters "Bearbeiten" wird in den Bearbeitungsmodus gewechselt. In diesem Modus wird im Inhaltsbereich nur das aktuell bearbeitete Thema dargestellt.



Wireframe User Interface Thema (Bearbeitung)

In den Wireframes sind beispielhaft 3 Kategorien angedeutet. Es muss selbstverständlich möglich sein, keine, eine oder viele Kategorien zuzuordnen.

Das Löschen eines Themas muss ausdrücklich bestätigt werden.

Zur Bearbeitung der Thementexte reicht ein Textarea-Element aus.

Anforderungen an die Umsetzung

Allgemeine Anforderungen

Die Webanwendung "MyWiki" wird als Client-Server-Anwendung realisiert:

- Architektur:
  - Single-Page-Applikation
  - REST-Interface (siehe weiter unten)
- Webclient:
  - Verwendung HTML5 (XML-konforme Notation)
  - Überprüfung des Markup mit Hilfe der w3c-Validator-Dienste (siehe Anhang)
  - Präsentation mit CSS, ausgelagert in eine externe CSS-Datei
    - die Gestaltung muss auf Basis des vorgegebenen Wireframe von Form und Farbe her so erfolgen, dass die Anwendung ästhetisch ansprechend und gut bedienbar ist (siehe auch weiter unten)
  - Verwendung JavaScript
  - Verwendung Template-Engine `te.js/tm.js`
  - Verwendung Eventservice `es.js`
  - Verwendung XMLHTTRequest, aufbauend auf `xhr.js`
- Webserver:
  - Verwendung Python (Version 3)
  - Verwendung Framework "cherrypy" (siehe auch Hinweise im Anhang)
    - Verwendung "MethodDispatcher".

Die Verwendung von weiteren JavaScript-Bibliotheken / -Frameworks sowie CSS-Frameworks ist unzulässig!

Verzeichnisstruktur

Verwenden Sie folgende Verzeichnisstruktur und erstellen Sie die angegebenen Dateien:

```
web
   /p4
       /mw                  <--- server.py
           /app             <--- __init__.py, application.py, database.py, view.py, ggf. weitere
           /content         <--- Datei index.html, css-Datei(en), js-Datei(en)
           /data            <--- Daten in JSON-formatierten Dateien
           /doc             <--- Dateien der Dokumentation
           /templates       <--- Vorlagen für te/tm
           /test            <--- curl-Test-Skripte und Testergebnisse
```

REST-Schnittstellen

Orientieren Sie sich bei der Konzeption und Implementierung der REST-Schnittstellen an folgenden Vorschlägen, die Sie ggf. erweitern / anpassen müssen!

Methode	Pfad	Reaktion
GET	/	Anzeige Startseite , wie GET /topics/home/

REST-Schnittstellen (1)

Methode	**Pfad**	**Reaktion**
GET	/topics/	alle Themen
GET	/topics/home/	Thema `home`, wird für die Startseite verwendet
GET	/topics/favorites/	Thema `favorites`, zur Anzeige der Favoriten
GET	/topics/missing/	Thema `missing`, zur Anzeige der fehlenden Themen
GET	/topics/orphans/	Thema `orphans`, zur Anzeige der verwaisten Themen
GET	/topics/:id	das durch die id bezeichnete Thema (siehe Hinweise unten)
POST	/topics/:id	Thema mit der angegebenen Bezeichnung (id) und Vorgabewerten neu erstellen
PUT	/topics/:id + Daten	bestehendes Thema aktualisieren
DELETE	/topics/:id	bestehenden Thema löschen
GET	/tags/	alle Kategorien
GET	/tags/:tag	die angegebene Kategorie mit allen Themen, denen sie zugeordnet ist

REST-Schnittstellen (2)

Hinweise:

- die Bezeichnungen `home`, `favorites`, `missing` und `orphans` sind vorgegebenen und mit einer bestimmten Auswertung verbunden; es können keine Themen mit diesen Bezeichnungen erstellt oder gelöscht werden
- wird bei `GET /topics/:id` festgestellt, dass es das angegebene Thema noch nicht gibt, wird dieses erzeugt
  - dazu wird eine interne Anfrageumleitung auf `POST /topics/:id` durchgeführt, wodurch die Erzeugung erfolgt
  - als Ergebnis wird das neu erzeugte Thema zurückgeliefert
  - damit wird sichergestellt, dass die Semantik der REST-Schnittstelle nicht verändert wird

Mit Doppelpunkt gekennzeichnete Elemente sind variable Werte. Die Daten bei PUT- und POST-Anfragen werden im Message-Body übertragen.

Die Ergebnisse der Anfragen werden stets im JSON-Format zurückgeliefert. Bei Lösch-Anfragen wird als Inhalt des Ergebnisses die Id der gelöschten Daten zurückgegeben.

Der HTTP-Statuscode wird verwendet, um auf korrekt durchgeführte oder nicht ausführbare Anfragen hinzuweisen (z.B. 200 = Ok, 404 = angefragte Daten sind nicht vorhanden).

Der Template-Manager `tm.js` verwendet außerdem eine GET-Anfrage mit dem Pfad `/templates/`. Diese sollte ebenfalls per Method-Dispatching implementiert werden.

Präsentation mit CSS

Definieren Sie das Layout mit Hilfe von Flexboxes für die einzelnen Bereiche der Benutzungsschnittstelle sowie die beiden Formen der Themen.

Verwenden Sie für das Gesamtlayout etwa folgende Bereiche

- Header mit zwei Teilbereichen (HeaderLeft / HeaderRight)
- Menubar
- Inhaltsbereich.

Diese Bestandteile können Sie in der Datei `index.html` mit Markup beschreiben. Geben Sie für den Textbereich der Themen (in beiden Darstellungsmodi) eine maximale Höhe an und sehen Sie vor, dass der Inhalt des Thementextes ggf. mit Hilfe eines Scrollbars verschoben werden kann. Verwenden Sie auch hier Flexboxes für das Layout.

Clientseitige Templates

Menubar und Inhaltsbereich werden dynamisch mit verschiedenen Inhalten versehen. Verwenden Sie geeignete Templates, die durch `te.js`/`tm.js` clientseitig bereitgestellt und ausgewertet werden.

Anforderungen an die Dokumentation

Erstellen Sie eine Dokumentation, die Ihre Lösung beschreibt. Legen Sie dazu in einem Unterverzeichnis `doc` die Datei `mw.md` an. Sehen Sie folgende Gliederung vor:

- einleitend: Ihre Gruppenzugehörigkeit, Aufbau Ihres Team, Gültigkeitsdatum der Dokumentation
- allgemeine Beschreibung Ihrer Lösung
  - Aufgabe der Anwendung (*von Ihnen erstellter Text!*)
  - Übersicht der fachlichen Funktionen (*von Ihnen erstellter Text!*)
- Beschreibung der Komponenten des Servers
  - für jede Komponente:
    - Zweck
    - Aufbau (Bestandteile der Komponente)
    - Zusammenwirken mit anderen Komponenten
    - API (Programmierschnittstellen), die die Leistungen der Komponente anbieten
- Beschreibung der Komponenten des Client
  - für jede Komponente:
    - Zweck
    - Aufbau (Bestandteile der Komponente)
    - Zusammenwirken mit anderen Komponenten
    - API (Programmierschnittstellen), die die Leistungen der Komponente anbieten
- Datenablage
- Durchführung und Ergebnis der Überprüfung des generierten Markups durch den W3C-Validatordienst.

Die Dokumentation wird als utf-8 kodierter Text mit der einfachen Auszeichnungssprache `Markdown` erstellt. Mit Hilfe des Werkzeugs `pandoc` (siehe https://pandoc.org) kann eine Umsetzung in eine HTML-Datei erfolgen:

```
pandoc -f markdown -t html5 -s <IhreDatei> -o <IhreHTML5Datei>
```

Die in pandoc verfügbaren Erweiterungen der Auszeichnungssprache Markdown sollen genutzt werden.

Bewertung / Testat

Zur Bewertung Ihrer Lösung im Hinblick auf die mögliche Erteilung des Testats müssen Sie vorlegen und erläutern:

- den von Ihnen erstellten Quellcode Ihrer Webanwendung
- die von Ihnen erstellte Dokumentation dazu

Sie müssen die Lauffähigkeit Ihrer Lösung, die Durchführung von Tests mit cURL und die Durchführung der Validierung (W3C) nachweisen.

Um die korrekte Zuordnung zu Monaten und die automatische Anpassung der Liste der Monate im Sidebar zu demonstrieren, müssen Sie geeignete Testdaten in der Datenbasis erstellen.

Anhang

Nutzung des W3C-Validatordienstes für Markup

- **w3c-Validator-Dienst (Markup):** http://validator.w3.org/
  - Überprüfung der Korrektheit des Markup
  - Zeigen Sie den Quelltext der zu prüfenden Webseite an (z.B. Kontextmenü Webseite, dort etwa "Seitenquelltext" auswählen)
  - Den angezeigten Quelltext markieren und in die Zwischenablage kopieren
  - Inhalt der Zwischenablage in der Registerkarte "Direct Input" einfügen und Überprüfung starten

JavaScript-Bibliotheken `te.js/tm.js`, `es.js` und `xhr.js`

Die Quellen und die Dokumentation zu diesen JavaScript-Bibliotheken können Sie den zusätzlich veröffentlichten Unterlagen zur Veranstaltung WEB entnehmen. Erweitern Sie ggf. das Script in `xhr.js`.

Verwenden Sie die für Ihren Webbrowser geeignete Variante. Damit Sie clientseitig die Dateinamen `te.js`, `tm.js`, `es.js` und `xhr.js` unabhängig von der verwendeten Version benutzen können, können Sie mit dem "staticfile-Tool" von "cherrypy" die tatsächlichen Dateien referenzieren (siehe nächsten Abschnitt).

Hinweise zur Verwendung des Frameworks "cherrypy"

Method-Dispatching

Zur Implementierung der REST-Schnittstellen wird im Framework "cherrypy" das sog. *Method-Dispatching* verwendet: die HTTP-Methode entscheidet, welche Methode aufgerufen wird. In der Dokumentation zu "cherrypy" wird dies im Tutorial 7 demonstriert (siehe http://docs.cherrypy.org/en/latest/tutorials.html#tutorial-7-give-us-a-rest). Es ist **nicht** erforderlich, die im Abschnitt "Advanced" der Dokumentation zu "cherrypy" beschriebene Vorgehensweise "RESTFul-style dispatching" zur Auswertung komplexerer Pfade in den URI anzuwenden.

Referenzierung statischer Ressourcen

In der "cherrypy"-Dokumentation wird unter http://docs.cherrypy.org/en/latest/basics.html#serving-a-single-file beschrieben, wie mit Hilfe der Konfiguration und des "staticfile-Tool" eine Anfrage zur Auslieferung einer statischen Ressource auf eine bestimmte Datei abgebildet werden kann.

Wenn Sie beispielsweise die Datei `es_c6_to.js` (Eventservice, notiert mit ES6-Klassenschreibweise, Verwendung TimeOut-Mechanismus) bei einer Anfrage der Art `http://localhost:8080/es.js` ausliefern wollen, können Sie in der Konfigurationsdatei angeben:

```
[/es.js]
tools.staticfile.on = True
tools.staticfile.filename = "content/es_c6_to.js"
```

Damit ist es auf einfachem Wege möglich, die Dateien serverseitig auszutauschen ohne die clientseitige Referenzierung zu ändern.

Test REST-Interface mit cURL

Testen Sie das REST-Interface des Webservers mit dem Werkzeug **cURL**. Varianten dieses Werkzeugs für gängige Betriebssysteme und Hinweise zur Anwendung finden Sie unter

- https://curl.haxx.se, insbesondere https://curl.haxx.se/docs/manual.html
- http://alvinalexander.com/web/using-curl-scripts-to-test-restful-web-services
- oder werten Sie eine Suche bei Google z.B. mit den Stichworten "cURL REST" aus.

Erstellen Sie für die einzelnen Ressourcen und Methoden Testskripte an und speichern Sie diese zusammen mit den Ergebnisse im Verzeichnis `test`.