Dokumentation

Vanessa Nock, Lucas Hardt, Sofia Rodriguez, Lucas Lindstedt

Installation

Die Installation erfolgt mit Docker. Hierfür wird eine funktionierende Installation einer Docker Instanz benötigt.

Zunächst muss das Image gebaut werden. Dies ist Hauptverzeichnis mithilfe des `docker build -t rituale .` Befehls möglich.

Der Container selbst lässt sich mithilfe des Befehls `docker run -p 8080:8080 rituale` starten.

Die Webseite ist nun unter http://localhost:8080 verfügbar.

Speicherung personalisierter Daten

Beim Aufruf der Seite wird dem User ein Cookie mit einer einzigartigen User-ID, die mithilfe des uuid-Pakets erstellt wird, zugeordnet. Diese wird im Backend im UserData-Objekt als Key des jeweiligen User-Objektes hinterlegt. Das User-Objekt enthält dann entsprechend folgende Daten:

- favouriteSites => Das String-Array enthält alle geliketen Seiten.
- visitCounter => Dieses Objekt enthält für jede bereits besuchte Seite einen entsprechenden Counter, der mit 0 initialisiert und bei jedem Besuch der jeweiligen Seite um 1 erhöht wird.
- name => Dieser String ist ein durch das Modul "random-username-generator" zufällig generierter Nutzername.

Anzeige personalisierter Daten

Beim Besuch der Startseite wird unter der Navigationsleiste eine Begrüßung inklusive Nutzernamen angezeigt. Dahinter steht, welche Seite vom User am meisten besucht wurde. Sollte noch keine Seite besucht worden sein, so wird das dort angezeigt. Hat der User schon eine der drei Unterseiten geliked, so wird der entsprechende Like-Button in der "Rituale"-Tabelle nun grün und enthält "LIKED!" als Schriftzug.

Kommentare

Die Kommentarsektion befindet sich direkt über der Fußleiste. Einen Kommentar kann man in das dort befindliche Feld eintragen und dann mithilfe des "Kommentieren"-Buttons senden. Danach lädt die Seite automatisch neu und der neue Kommentar wird darunter angezeigt. Zu jedem Kommentar wird die Kommentarzeit, der Nutzername des jeweiligen Nutzers und der Kommentar selbst nebeneinander angezeigt.

Favoriten / Likes

Im Abschnitt "Rituale" befinden sich an der rechten Seite der Tabelle die Like-Buttons. Klickt man einen der Buttons an, so lädt die Seite neu und das entsprechende Ritual wird als "LIKED!" angezeigt, da es nun ein Favorit ist. Gelikete Seiten werden auch beim erneuten Besuch weiterhin als solche gekennzeichnet.

Schnittstellen

statische Dokumente

GET

Alle statischen Dokumente wie die HTML-Dokumente der Rituale oder die CSS-Datei sind über "/" abrufbar. Beispielsweise kann man die Tauf-Seite über "/Rituale/Taufen.html" aufrufen.

/api/users/@me

GET

Wird eine GET-Anfrage an diese URL gestellt, werden anhand des NutzerID-Cookies folgende Daten im JSON Format zurück gegeben:

- Die User ID als String (uid)
- Der Name als String (name)
- Die am meisten besuchte Ritual-Seite als String, falls vorhanden, sonst null (mostVisited)
- Die favorisierten Seiten als String Array (favouriteSites)

PATCH

Wird eine PATCH-Anfrage an diese URL gestellt, können mithilfe des Nutzer Cookies und eines JSON bodies die Nuzterdaten verändert werden. Folgende Daten können verändert werden:

- Die favorisierten Seiten als Array von Strings (favouriteSites)

Falls der Angegebene Wert dem angefragten Typ entspricht, wird das aktualisierte User Objekt wie in der GET Anfrage zurückgegeben. Doppelte Einträge werden automatisch gelöscht.

Falls nicht, wird ein Fehler mit dem Statuscode 400 zurückgegeben.

/api/users/:id

GET

Wird eine GET-Anfrage an diese URL gestellt und enthält diese eine ungültige oder keine User-ID anstelle von ":id", so wird eine Antwort mit dem HTTP-Statuscode 404 gesendet, da kein entsprechender User existiert. Existiert ein User mit der entsprechenden ID, so wird ein Objekt mit folgenden Elementen zurückgeliefert:

- id: Das ist die User-ID als String, die auch schon in der Anfrage enthalten ist.
- name: Der Username als String, der zur angefragten User-ID gehört.

/api/comments

GET

Sind keine Kommentare vorhanden, wird eine Antwort mit dem HTTP-Statuscode 204 und ein leeres Objekt gesendet. Sind Kommentare vorhanden, so enthält die Antwort mit dem Statuscode 200 ein Array, das die vorhandenen Kommentare als Objekte enthält. Diese Objekte enthalten die Information über die Kommentarzeit als UNIX-Timestamp, die User-ID des Verfassers und den Kommentartext selbst.

POST

Hier wird ein Objekt mit dem Attribut "comment" erwartet, das einen nicht leeren String darstellt. Ist das nicht der Fall, wird diese Bedingung in der Antwort mitgeteilt und der HTTP-Statuscode 400 gesetzt. Werden alle Bedingungen erfüllt, so wird der Kommentartext in einem Objekt im "comments"-Array gespeichert. Die abrufbaren Informationen (User-ID und Zeit) werden ebenfalls gespeichert.