FF Webpage

Anleitung

bis v1.1.x

Open Source Software
entwickelt durch JK Effects
von Julian Krauser

05. Februar 2025

Inhaltsverzeichnis

1 Einleitung

FF Webpage - Der flexible Webseitenbaukasten für Feuerwehren und Vereine

FF Webpage ist ein modularer Webseitenbaukasten, bestehend aus einem Nuxt-Frontend und Strapi als CMS. Struktur und Navigation der Seite können über das Strapi CMS individuell konfiguriert werden, was eine flexible Gestaltung ermöglicht.

Speziell entwickelte Module erleichtern die Darstellung von Listen, wobei vordefinierte Listen für Einsätze, Termine und Artikel zur Verfügung stehen. Ein besonderes Feature ist die direkte Übernahme von Terminen aus FF Admin in die Website, wodurch eine nahtlose Integration in das FF-Ökosystem gewährleistet ist.

2 Installation

FF Webpage kann über mehrere Wege betrieben werden. Zum einen werden Docker-Images versioniert zur Verfügung gestellt. Weiterhin kann auch das Release Projekt heruntergeladen und verwendet werden.

2.1 Docker

Disclaimer: Die Anleitung zum Betrieb von FF Webpage mit Docker setzt Kenntnisse mit Docker und Docker-Compose voraus.

Die Docker-Images können gemeinsam über eine Compose-File konfiguriert und gestartet werden. Auch können die Images einzeln gestartet werden.

Die Docker-Images sind versioniert. Der `<tag>` des Images kann entweder `latest` für die neueste Version oder `vX.Y.Z` für eine bestimmte Version sein. Die Versionen können auch in den Releases der Repositories der Anwendungen nachgeschlagen werden. Dort lassen sich auch auch Informationen zu neuen Funktionen, Änderungen oder Fehlerbehebungen der jeweiligen Funktion finden.

2.1.1 Docker-Compose

Webseite

```yaml
ff-landingpage:
  image: docker.registry.jk-effects.cloud/ehrenamt/ff-webpage/frontend:<version>
  container_name: ff_landingpage
  restart: unless-stopped
  environment:
    - NUXT_STRAPI_URL=<cms_url>
    - NUXT_PUBLIC_STRAPI_URL=<cms_url>
    - NUXT_PUBLIC_APP_TITLE=<website_titel>
  ports:
    - "3000:3000"
  volumes:
    - <volume|local path>/favicon.png:/app/.output/public/favicon.png
```

Die Verwendung der Werte des Typs Environment werden unter dem Punkt Konfiguration erklärt.

Das Volume dient zur erweiterten Personalisierung der Webseite mit eigenem Logo der Feuerwehr oder des Vereins. Hiervon betroffen ist nur Icon im Browser-Tab.

Die Konfiguration der Volumes ist optional, falls Sie die Standard-Logos verwenden wollen.
Die Dateien müssen exakt gleich geschrieben sein. Achten Sie deshalb auf Schreibfehler und Groß-/Kleinschreibung.

CMS - Inhaltsverwaltung

```yaml
ff-cms:
  image: docker.registry.jk-effects.cloud/ehrenamt/ff-webpage/cms:<version>
  container_name: ff_cms
  restart: unless-stopped
  environment:
    - DATABASE_CLIENT=mysql
    - DATABASE_HOST=ff-db
    - DATABASE_NAME=ffcms
    - DATABASE_USERNAME=cms
    - DATABASE_PASSWORD=<dbuserpasswd>
    - JWT_SECRET=<tobemodified>
    - ADMIN_JWT_SECRET=<tobemodified>
    - APP_KEYS=<tobemodified>,<tobemodified>
    - API_TOKEN_SALT=<tobemodified>
    - TRANSFER_TOKEN_SALT=<tobemodified>
  volumes:
    - <volume|local path>:/app/public/uploads
  ports:
    - "1337:1337"
```

Die Verwendung der Werte des Typs Environment werden unter dem Punkt Konfiguration erklärt.
Environment Werte können optional sein oder haben Standard-Werte.
Das Fehlen einer geforderten Variable oder die falsche Angabe eines Variablen-Werts verhindert das Starten des der Anwendung.

Innerhalb dem Ordner, der dem Volume zugeordnet ist, werden Uploads abgelegt.

Datenbank

Als Datenbank können MySQL, Postgres und SQLite verwendet werden. Postgres wird für den Produktiven Einsatz empfohlen.

Konfiguration von MySQL:

```yaml
ff-db:
  image: mariadb:<version (bsp 11.2)>
  container_name: ff_db
  restart: unless-stopped
  ports:
    - "3306:3306"
  environment:
    - MYSQL_DATABASE=<database name>
    - MYSQL_USER=<username>
    - MYSQL_PASSWORD=<user password>
    - MYSQL_ROOT_PASSWORD=<root password>
  volumes:
    - <volume|local path>:/var/lib/mysql
```

`MYSQL_USER` und `MYSQL_PASSWORD` sind optional. Werden diese nicht gesetzt, kann der Server entweder mit dem Nutzer `root` und dem gesetzten `MYSQL_ROOT_PASSWORD` Zugang zur Datenbank erhalten, oder es wird im nachhinein ein Nutzerzugang erstellt, der Zugriff auf die erstellte Datenbank hat. `MYSQL_DATABASE` erstellt direkt eine Datenbank, die durch einen angelegten `MYSQL_USER` verfügbar ist.

Konfiguration von Postgres:

```yaml
ff-db:
  image: postgres:<version (bsp 16)>
  container_name: ff_db
  restart: unless-stopped
  ports:
    - "5432:5432"
  environment:
    - POSTGRES_DB=<database name>
    - POSTGRES_USER=<username>
    - POSTGRES_PASSWORD=<user password>
  volumes:
    - <volume|local path>:/var/lib/postgresql/data
```

`POSTGRES_DB` erstellt direkt eine Datenbank, die durch einen angelegten `POSTGRES_USER` verfügbar ist.
Hinweis Wenn eine Docker-Compose Datei verwendet wird, kann zusätzliche ein Netzwerk angelegt werden. Dadurch ist das Veröffentlichen der Datenbank-Port-Exposes nicht mehr notwendig. Das Entfernern der port-Exposes verhindert den direkten Zugriff auf die Ports von außerhalb. Ergänzt muss hierfür das `network` und die Teilhabe des Backend-Containers am Netzwerk:

1. Ergänzung zu Server und Datenbank:

```
networks:
  - ff_internal
```

2. Ergänzung zur finalen Compose:

```
networks:
  ff_internal:
```

3. Optionale Ergänzung zum Server:

```
depends_on:
  - ff-db
```

Hierdurch kann der Server nicht starten, wenn die verwendete Datenbank nicht läuft.

2.2 Git

Eine Alternative zu Docker ist die direkte Ausführung der Anwendungen auf dem Server oder Desktop Gerät.
Hierzu müssen die App und der Server als Quellcode aud das System geladen und dort direkt verwendet werden.

Die Veröffentlichung der App und des Servers, damit diese aus dem Internet erreichbar sind muss gesondert eingerichtet werden.

Das System muss NodeJs und die bevorzugte Datenbank installiert haben.

Die NodeJs Prozesse können auch durch Tools wie pm2 verwaltet werden.

Um die Konfiguration mittels ENV-Variablen an die Anwendungen weitergeben zu können, müssen `.env` Dateien erstellt werden. Hierzu kann die `.env.example` Datei kopiert und die definierten Werte ausgefüllt werden. Nicht benötigte Einträge sollten entfernt werden.
Die env-Dateien müssen vor der Ausführung von `npm run start` angelegt sein. Bei einer Änderung der Einträge müssen die Anwendungen lediglich neu gestartet werden.

Webseite

```shell
git clone https://forgejo.jk-effects.cloud/Ehrenamt/ff-webpage.git
cd ff-webpage
npm install
npm run build
npm run start
```

`npm run start` nutzt das eingebaute Hosting von Nuxt, welches SSR ermöglicht, um die App zu betreiben.

Strapi

```shell
git clone https://forgejo.jk-effects.cloud/Ehrenamt/ff-webpage-cms.git
cd ff-webpage-cms
npm install
npm run build
npm run start
```

2.3 Konfiguration

Folgende Werte können zu einem Container konfiguriert werden:

Variable	Zweck	Fallback	optional
⬇️ App-Variablen			
NUXT_STRAPI_URL	URL, unter der das CMS erreichbar ist.		💥
NUXT_PUBLIC_STRAPI_URL	URL, unter der das CMS erreichbar ist.		💥
NUXT_PUBLIC_APP_TITLE	Text im Browser-Tab	FF Webpage	✅
⬇️ Server-Variablen			
DB_TYPE	Folgende Datenbanktypen sind verfügbar: mysql, postgres		💥
DB_HOST	URL zur Datenbank		💥
DB_PORT	Port der Datenbank		💥
DB_NAME	Name der Datenbank in welcher die Tabellen erstellt werden.		💥
DB_USERNAME	Nutzername für Zugang zu Datenbank		💥
DB_PASSWORD	Passwort zum Zugang zur Datenbank		💥
JWT_SECRET			💥
ADMIN_JWT_SECRET			✅
APP_KEYS			✅
API_TOKEN_SALT			✅
TRANSFER_TOKEN_SALT			✅
⬇️ Database-Variablen			
MYSQL_DATABASE	Name der Datenbank, die bei Erstellung direkt angelegt wird.		💥

Variable	Zweck	Fallback	optional
MYSQL_USER	Benutzername des Users, der bei Erstellung direkt angelegt wird.		✅
MYSQL_PASSWORD	Passwort zum User, das bei Erstellung gesetzt wird.		✅
MYSQL_ROOT_PASSWORD	Passwort für den User root, das bei Erstellung gesetzt wird.		💥
POSTGRES_DB	Name der Datenbank, die bei Erstellung direkt angelegt wird.		💥
POSTGRES_USER	Benutzername des Users, der bei Erstellung direkt angelegt wird.		💥
POSTGRES_PASSWORD	Passwort zum User, das bei Erstellung gesetzt wird.		💥

💥: Ein Fehlen dieser Variable verhindert das Starten der Anwendung! 🚨: Bei Verwendung von SQLite sind diese Variablen nicht notwendig!

Hinweis: Eine fehlerhafte Konfiguration der optionalen oder geforderten Variable verhindert das Starten der Anwendung.
Hinweis: Eine Änderung der Datenbank übernimmt nicht automatisch die Daten in die neue Datenbank.

2.4 Update der Version

Um eine Version auf eine Neuere zu aktualisieren, muss meist nur der Docker-Tag oder das Repo ersetzt werden.
Wer Docker mit `latest` nutzt, kann das neue Image direkt mit `docker pull` neu beziehen und dann den Container neustarten.

Informationen zu neuen Versionen können innerhalb der App im Account des Eigentümers oder in den Release-Pages gefunden werden.
Die Releases beinhalten Informationen zu einem Update und was zu beachten ist. So enthalten die Release-Informationen beispielsweise Vorbereitungen vor einem Update.

Bei Verwendung mittels Git, müssen die Repos neu bezogen werden. Anschließend müssen die Dependencies neu installiert und die Anwendungen neu gebaut werden.

3 Strapi

Disclaimer: Fehler, die auf der Webseite angezeigt werden, können auf eine fehlerhafte Konfiguration in Strapi hinweisen.

3.1 Einrichtung

Um die Webseite nutzen zu können, müssen Global die Navigation sowie die Startseite konfiguriert sein.

Wichtig ist, dass ein paar Einstellungen vor der ersten Verwendung vorgenommen werden. Dazu zählen:

1. **Admin Benutzer erstellen**: Erstellen Sie einen Admin Benutzer, um auf das CMS zugreifen zu können.
2. **Einstellungen anpassen**: In den Einstellungen müssen unter dem Punkt `Users & Permissions Plugin` die Rollen und Berechtigungen angepasst werden. Unter `Roles > Public` muss die Berechtigung `find`, `findOne` und falls vorhanden auch `distinctYears` und `findByYear` für alle Modelle aktiviert werden.
3. **Medienverwaltung**: Es empfiehlt sich unter `Global Settings > Media Library` alle Optionen auf true zu stellen. Dadurch wird zum Beispiel auch die automatische Rotation von Bildern aktiviert.
4. **Rollen und Berechtigungen**: Unter `Administration Panel > Roles` können die Rollen und Berechtigungen für die Benutzer angepasst werden. Es empfiehlt sich, die Standardrollen zu übernehmen und nur die Berechtigungen anzupassen. Diese können aber auch beliebig erweitert werden. Dieser Punkt betrifft nur die Erstellung von Inhalten in der Adminstrationsoberfläche.
5. **Nutzer einladen**: Unter `Administration Panel > Users` können weitere Nutzer eingeladen werden. Diese erhalten dann eine E-Mail mit einem Link, um sich ein Passwort zu setzen.
6. **Erstellung von Inhalten**: Nachdem die Einstellungen vorgenommen wurden, können Inhalte erstellt werden. Dazu können die Inhalte über die Oberfläche zu den bestehenden Modellen erstellt werden.
7. **Einrichtung Sammlungsreferenz**: Die Sammlungsreferenz ist ein eigenes Modell, das die Verknüpfung von Artikeln, Einsätzen, Fahrzeugen und Terminen ermöglicht. Die Werte für Collection können sind die Referenzen auf die genannten Listen. Diese Werte müssen Plural sein und können nur die Werte `articles`, `operations`, `vehicles` und `events` annehmen.
   Es empfiehlt sich diese Werte direkt zu setzen, da diese dann als Referenz in anderen Modellen genutzt werden können.

3.1.1 Verwendung der Modelle

Beispiele zu den beschriebenen Inhalten finden sie in der Demo.

3.1.2 Allgemeine Bezeichnungen

- **Slug**: Einzigartige Bezeichnung als Identifikation in der URL.
- **Identifier**: Wert, zu Identifikation in Referenz-Feldern.

3.1.3 Modelle

1. **Global**: Hier wird festgelegt, wie die Navigation und der Footer aufgebaut sind. Die Navigation nutzt Informationen zu verwiesenen Seiten und Pfaden zu diesen Seiten. Die Werte der Pfade sollten mit den Werten aus der Sammlungsreferenz übereinstimmen. Dadurch wird gewährleistet, dass Detailansichten der Artikel, Einsätze, Fahrzeuge und Termine korrekt angezeigt werden können.
2. **Startseite**: Hier können die Inhalte der Startseite angepasst werden.
   - **Backdrop**: Bildschirmfüllendes Bild, das im Hintergrund angezeigt wird.
   - Das in Global angegebene Logo wird unten im Eck über den Backdrop gelegt.
3. **Artikel**: Verwaltung und Erstellung von Artikeln.
4. **Einsätze**: Verwaltung und Erstellung von Einsätzen.
5. **Fahrzeuge**: Verwaltung und Erstellung von Fahrzeugen.
6. **Sammlungs Referenz**: Verwaltung und Erstellung von Referenzen auf Artikel, Einsätze, Termine und Fahrzeuge.
   - **Image Item**: Soll das Element in der Liste ein Bild anzeigen.
   - **Date List**: Sollen die Listenelemente gruppiert nach Jahren angezeigt und navigiert werden.
   - **Numbered Item**: Soll bei den ListenElementen ohne Bild statt dem Tag als große Zahl eine Nummerierung erfolgen.
   - **Inverse Count**: Soll die Zählweise umgekehrt sein. (Sinnvoll bei Einsätzen)
7. **Seiten**: Verwaltung und Erstellung von Seiten mit Dynamischen Inhalten.
   - **Hero**: Bild mit Text als Einleitung.
8. **Termine**: Verwaltung und Erstellung von Terminen.
9. **User**: wird nicht benötigt

3.2 Verwendung