FF Admin

Anleitung

bis v1.3.x

Open Source Software
entwickelt durch JK Effects
von Julian Krauser

05. Februar 2025

Inhaltsverzeichnis

1 Einleitung

FF Admin - Die zentrale Verwaltungssoftware für Feuerwehren und Vereine

FF Admin ist eine vielseitige Mitgliederverwaltungssoftware, die als Herzstück eines wachsenden Ökosystems dient. Neben der Mitgliederverwaltung ermöglicht das Programm die Organisation von Terminkalendern, die Erstellung von Newslettern und Protokollen sowie - in Zukunft - die Verwaltung von Gerätschaften und Prüfplänen.

Obwohl FF Admin in erster Linie für Feuerwehren konzipiert ist, kann es dank seines modularen Aufbaus auch für andere Organisationen angepasst werden. Die frei definierbaren Stammdaten ermöglichen einen flexiblen Einsatz, so dass die Software optimal an die individuellen Bedürfnisse angepasst werden kann.

2 Installation

FF Admin 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 Admin 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

App

```yaml
ff-admin-app:
  image: docker.registry.jk-effects.cloud/ehrenamt/ff-admin/app:<version>
  container_name: ff_admin
  restart: unless-stopped
  ports:
    - "80:80"
  environment:
    - SERVERADDRESS=<backend_url>
    - APPNAMEOVERWRITE=<appname>
    - IMPRINTLINK=<imprint link>
    - PRIVACYLINK=<privacy link>
    - CUSTOMLOGINMESSAGE=<betrieben von xy>
  volumes:
    - <volume|local path>/favicon.ico:/usr/share/nginx/html/favicon.ico
    - <volume|local path>/favicon.png:/usr/share/nginx/html/favicon.png
    - <volume|local path>/Logo.png:/usr/share/nginx/html/Logo.png
```

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

Alle Environment Werte sind Optional und haben Standard-Werte.
Ist ein Wert optional und hat keinen Fallback, so wird in der Anwendung nichts angezeigt.

Die Volumes dienen zur erweiterten Personalisierung der App mit eigenem Logo der Feuerwehr oder des Vereins. Hiervon betroffen ist das Icon im Browser-Tab, jede Anzeige des FF Admin Logos innerhalb der App und das Icon, wenn die WebApp auf einem Gerät installiert wird.

Die Konfiguration der Volumes ist optional, falls Sie die Standard-Logos verwenden wollen.

Ein Teil der Logos haben eine Anforderung an die Auflösung:

Icon	**Auflösung**	**Anzeigeort**
favicon.ico	48x48 px	Browser-Tab Icon
favicon.png	512x512 px	WebApp Icon zur Installation
Logo.png	beliebig	Innerhalb der Anwendung

Die Dateien müssen exakt gleich geschrieben sein. Achten Sie deshalb auf Schreibfehler und Groß-/Kleinschreibung.

Server

```yaml
ff-admin-server:
  image: docker.registry.jk-effects.cloud/ehrenamt/ff-admin/server:<version>
  container_name: ff_member_administration_server
  restart: unless-stopped
  ports:
    - "5000:5000"
  environment:
    - DB_TYPE=<database type>
    - DB_HOST=<database host>
    - DB_PORT=<database port>
    - DB_NAME=<database name>
    - DB_USERNAME=<database username>
    - DB_PASSWORD=<database password>
    - JWT_SECRET=<jwt secret>
    - JWT_EXPIRATION=<jwt expiration>
    - REFRESH_EXPIRATION=<refresh expiration>
    - PWA_REFRESH_EXPIRATION=<pwa refresh expiration>
    - MAIL_USERNAME=<mailadress|username>
    - MAIL_PASSWORD=<mail password>
    - MAIL_HOST=<mail server url>
    - MAIL_PORT=<port>
    - MAIL_SECURE=<boolean>
    - CLUB_NAME=<club name>
    - CLUB_WEBSITE=<club website>
    - BACKUP_INTERVAL=<backup interval>
    - BACKUP_COPIES=<backup parallel copies>
    - BACKUP_AUTO_RESTORE=<boolean>
  volumes:
    - <volume|local path>:/app/files
```

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 Backups und Ausdrucke der geschriebenen Protokolle und Newsletter 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.

2.1.2 Docker-AIO

In Arbeit
Ein Docker-Image, welches alle notwendigen Komponenten beinhaltet, ist in der Erstellung.

Das All-In-One Image abstrahiert das Routing zu App und Server und beinhaltet direkt die Datenbank. Auch werden gleiche Konfigurations-Daten zusammengefasst und an die Container übergeben.

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.

Für das Hosting von statischen Inhalten kann Apache oder Nginx verwendet werden. Eine Konfiguration für Nginx ist im Repo der App enthalten.
Die NodeJs Prozesse können auch durch Tools wie pm2 verwaltet werden.

Um die Konfiguration mittels ENV-Variablen an die Anwendungen weitergeben zu können, müs-

sen `.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-Datei im Frontend muss vor dem build-Prozess erstellt sein, da dort die Werte fest in den Code übernommen werden. Weiterhin muss die Datei im Frontend `.env.production` heißen. Die bestehende Datei kann modifiziert werden. Bei einer Änderung muss die App neu gebaut werden.
Die env-Datei im Backend muss vor der Ausführung von `npm run start` angelegt sein. Bei einer Änderung der Einträge muss der Server lediglich neu gestartet werden.

App

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

Der durch `npm run build` erstellte dist Ordner kann mit Apache oder Nginx zur Verfügung gestellt werden.

Server

```shell
git clone https://forgejo.jk-effects.cloud/Ehrenamt/ff-admin-server.git
cd ff-admin-server
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			
SERVERADDRESS	URL, über welche das Backend erreicht werden kann. Die URL muss mit http:// oder https:// starten und darf keinen Pfad beinhalten. Wenn das Backend auf der gleichen URL wie die App läuft, kann diese Variable weggelassen werden.		✅
APPNAMEOVERWRITE	Anzeige eines anderen Namens als FF Admin.	FF Admin	✅

Variable	Zweck	Fallback	optional
IMPRINTLINK	Link zum Impressum des Betreibers.		✅
PRIVACYLINK	Link zur Datenschutzerklärung des Betreibers.		✅
CUSTOMLOGINMESSAGE	Nachricht auf der Login-Seite. (Bsp.: betrieben von xy)		✅
⬇️ Server-Variablen			
DB_TYPE	Folgende Datenbanktypen sind verfügbar: mysql, sqlite, postgres	mysql	✅
DB_HOST	URL zur Datenbank oder Dateipfad zur SQLite-Datenbank		💥
DB_PORT	Port der Datenbank	3306	🚨
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	Zufällige Zeichenkette zur Validierung der Session-Tokens.		💥
JWT_EXPIRATION	Gültigkeitsdauer eines Session-Tokens. Format: [0-9]*(y|d|h|m|s)	15m	✅
REFRESH_EXPIRATION	Gültigkeitsdauer eines Logins nach letzter Nutzung der App im Browser Format: [0-9]*(y|d|h|m|s)	1d	✅
PWA_REFRESH_EXPIRATION	Gültigkeitsdauer eines Logins nach letzter Nutzung der installierten App Format: [0-9]*(y|d|h|m|s)	5d	✅
MAIL_USERNAME	Nutzername oder Mailadresse		💥
MAIL_PASSWORD	Passwort zum Nutzernamen oder der Mailadresse		💥
MAIL_HOST	URL des Mailservers		💥
MAIL_PORT	Port des Mailservers für Versand (SMTP). Ports sind 25, 465, 587	587	✅
MAIL_SECURE	Soll eine Secure Verbindung aufgebaut werden. Muss true sein bei Port 465.	false	✅

Variable	Zweck	Fallback	optional
CLUB_NAME	Wird für TOTP Titel und Kalender-ICS verwendet.	FF Admin	✅
CLUB_WEBSITE	Wird für Kalender-ICS verwendet		✅
BACKUP_INTERVAL	Wie viele Tage Abstand sollen zwischen Backups liegen? (min. 1)	1	✅
BACKUP_COPIES	Wie viele parallele Kopien von Backups sollen parallel Verfügbar sein? (min. 1)	7	✅
BACKUP_AUTO_RESTORE	Soll das neueste Backup bei Server-Start automatisch geladen werden, wenn die Datenbank als leer erkannt wird?	true	✅
⬇️ Database-Variablen			
MYSQL_DATABASE	Name der Datenbank, die bei Erstellung direkt angelegt wird.		💥
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 die Daten nur automatisch in die neue Datenbank, wenn `BACKUP_AUTO_RESTORE` aktiviert ist und ein Backup angelegt ist. Es werden dann die Daten des gefundenen Backups in die neue Datenbank eingefügt.

2.4 Einrichtung

Um die Anwendung nutzen zu können, kann ein erster Administrator-Account wie folgt erstellt werden:

1. **Admin Benutzer erstellen**: Erstellen Sie einen Admin Benutzer unter dem Pfad /setup, um auf die Mitgliederverwaltung Zugriff zu erhalten. Nach der Erstellung des ersten Benutzers wird der Pfad automatisch geblockt.
2. **Rollen und Berechtigungen**: Unter `Benutzer > Rollen` können die Rollen und Berechtigungen für die Benutzer erstellt und angepasst werden.
3. **Nutzer einladen**: Unter `Benutzer > Benutzer` können weitere Nutzer eingeladen werden. Diese erhalten dann eine E-Mail mit einem Link, um ein TOTP zu erhalten.

2.5 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 Konzepte

3.1 Stammdaten

3.2 Berechtigungen

3.3 Engines

3.3.1 Template-Engine

3.3.2 Query-Engine

3.3.3 Scheduling-Engine (bald)

4 Module

4.1 Mitgliederverwaltung & Stammdaten

4.2 Kalender

4.3 Protokolle

4.4 Newsletter

4.5 Backups

4.6 Query Builder & Query Store

4.7 Templates & Template Builder

4.8 Benutzer & Rollenverwaltung

4.9 Webapi

5 Ökosystem FF Admin

6 Roadmap

Folgende Funktionalitäten sind in Planung (Auszug):

- **Calendar Link Dictionary:** Speicherung von Kalender-Link-Configs mit Aliase wie eine Name.
- **Reihenfolge von Protokoll-Inhalten:** Änderung der Reihenfolge von Abstimmungen, Beschlüssen und TOPs.
- **Mitglieder Ausdruck:** Druck der Daten eines Mitglieds anhand eines Templates.
- **Listen Ausdruck:** Druck von Listen mit Daten eines Queries anhand eines Templates.
- **Verbesserung der Template-Erstellung:** Verbesserung oder Änderung des Prozesses und Interfaces zu Erstellung eigener Templates.
- **Query Builder Erweiterung:** Erweiterung der Abfrage-Funktionalitäten des Query Builders im Bereich der Sortierung und Daten-Verbindung.
- **Kalendereinträge und Webpage:** Versand der Eingetragenen Termine des Kalenders an die Webseite mit der Möglichkeit von Änderungen eines Termins.
- **Geräteverwaltung & Prüfpläne:** Erfassung von Gerätschaften mittels Barcode und Erstellung von Prüfplänen.
- **Erinnerungen:** Versand von Erinnerungen zu anstehenden Prüfungen oder Wartungen.