

# Betriebsdokumentation

# Inhaltsverzeichnis

1. Systemeinrichtung .....	1
1.1. Anforderungen .....	1
1.2. Grundkonfiguration .....	1
1.3. Django-Projekt einrichten .....	1
1.4. Apache2 konfigurieren .....	2
1.5. Berechtigungen einrichten .....	3
2. Datensicherung unter Linux .....	5
2.1. Backup einrichten .....	5
2.2. Wiederherstellung der Datenbank aus einem Backup .....	6
3. Update durchführen .....	7
4. Fehlerberichte .....	8

# 1. Systemeinrichtung

Es folgt eine kurze Anleitung zum Deployment der Mitgliederdatenbank mit Apache2 und mod\_wsgi. Für ausführliche Hinweise (auch zu anderen Konfigurationen) siehe:

<https://docs.djangoproject.com/en/3.0/howto/deployment/> oder

[https://www.digitalocean.com/community/tutorials/how-to-serve-django-applications-with-apache-and-mod\\_wsgi-on-ubuntu-16-04](https://www.digitalocean.com/community/tutorials/how-to-serve-django-applications-with-apache-and-mod_wsgi-on-ubuntu-16-04)

## 1.1. Anforderungen

- Webserver
- Linuxinstallation
- Python 3
- Git

## 1.2. Grundkonfiguration

Schritt	Befehl	Kommentar
1	<code>sudo apt update &amp;&amp; sudo apt upgrade</code>	Distribution aktualisieren
2	<code>sudo apt-get install python3-pip apache2 libapache2-mod-wsgi-py3</code>	Abhängigkeiten für das Deployment installieren
3	<code>sudo pip3 install virtualenv</code>	virtualenv installieren, um eine isolierte Python-Umgebung zu erstellen
4	<code>cd foo/</code>	In den Ordner wechseln, in dem die Software installiert werden soll
5	<code>git clone https://github.com/Sumarbrander/Stura-Mitgliederdatenbank.git</code>  <code>cd Stura-Mitgliederdatenbank/</code>	Clonen des Git-Repository und Wechseln in den Ordner
6	<code>virtualenv venv</code>  <code>source venv/bin/activate</code>	Virtuelle Umgebung erstellen und aktivieren
7	<code>pip install -r requirements.txt</code>	Anforderungen in der virtuellen Umgebung installieren

## 1.3. Django-Projekt einrichten

Zuerst müssen einige Einstellungen angepasst werden. Dazu muss die Datei "Stura-Mitgliederdatenbank/bin/settings.py" mit einem beliebigen Editor geöffnet werden.

z.B.: `nano bin/settings.py`

Schritt	Einstellung	Kommentar
1	<code>DEBUG = False</code>	diese Variable muss in einer produktiven Umgebung <b>unbedingt</b> auf <code>False</code> gestzt werden
2	<code>SECRET_KEY = 'xyz'</code>	Secret Key setzen, dieser muss sich <b>unbedingt</b> vom Key im Git-Repository unterscheiden, da dieser öffentlich sichtbar ist! (z.B. mit Hilfe von <a href="https://djecrety.ir/">https://djecrety.ir/</a> )
3	<code>ALLOWED_HOSTS = ["IP_oder_Domain"]</code>	"IP_oder_Domain" mit der öffentlich zugänglichen IP-Adresse oder Domain des Webserver ersetzten
4	<code>STATIC_ROOT = os.path.join(BASE_DIR, 'mystatic/')</code>	Verzeichnis für die "static files" festlegen
5	optional: <code>ADMINS, EMAIL_HOST, EMAIL_HOST_USER, EMAIL_HOST_PASSWORD, SERVER_EMAIL</code>	Es können optional weitere Einstellungen getroffen werden, die das Senden von Fehlerberichten ermöglichen, siehe <a href="#">Fehlerberichte</a>

Jetzt kann die Datei gespeichert und geschlossen werden.

Als nächstes muss ein Ordner "static" erstellt werden: `mkdir static`

Zuletzt müssen noch die Befehle zum Setup ausgeführt werden:

```
python ./manage.py makemigrations
python ./manage.py migrate
python ./manage.py collectstatic
python ./manage.py createsuperuser
```

Optional können jetzt noch einige Organisationseinheiten/Unterbereiche/Funktionen hinzugefügt werden:

```
cd importscripts
python main.py
```

Die Ausführung dieses Skripts kann einen Moment dauern.

Im Anschluss kann die virtuelle Umgebung deaktiviert werden:

```
deactivate
```

## 1.4. Apache2 konfigurieren

Um Apache2 als Webserver zu verwenden, muss WSGI konfiguriert werden. Dazu muss die "Virtual Host"-Datei bearbeitet werden:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

Hier muss die folgende Konfiguration eingefügt werden: (hier ist der Nutzernamen "pi", dieser muss natürlich angepasst werden)

*/etc/apache/sites-available/000-default.conf*

```
<VirtualHost *:80>

. . .

Alias /static /home/pi/StuRa-Mitgliederdatenbank/mystatic
<Directory /home/pi/StuRa-Mitgliederdatenbank/mystatic>
    Require all granted
</Directory>

<Directory /home/pi/StuRa-Mitgliederdatenbank/bin>
    <Files wsgi.py>
        Require all granted
    </Files>
</Directory>

</VirtualHost>
```

Dabei wird erst der Pfad zum "static"-Verzeichnis konfiguriert und dann der Pfad zur Datei "wsgi.py". Als nächstes müssen noch die folgenden Zeilen in die Datei hinzugefügt werden:

*/etc/apache/sites-available/000-default.conf*

```
<VirtualHost *:80>

. . .

WSGIDaemonProcess StuRa-Mitgliederdatenbank python-home=/home/pi/StuRa-
Mitgliederdatenbank/venv python-path=/home/pi/StuRa-Mitgliederdatenbank
WSGIProcessGroup StuRa-Mitgliederdatenbank
WSGIScriptAlias / /home/pi/StuRa-Mitgliederdatenbank/bin/wsgi.py

</VirtualHost>
```

## 1.5. Berechtigungen einrichten

Der erste Schritt ist, die Berechtigungen der Datenbankdatei so zu ändern, dass die Gruppe lesen und schreiben kann. Anschließend müssen dem Apache2-Nutzer einige Berechtigungen gewährt werden.

```
chmod 664 ~/Stura-Mitgliederdatenbank/db.sqlite3
sudo chown www-data:www-data ~/Stura-Mitgliederdatenbank/db.sqlite3
sudo chown www-data:www-data ~/Stura-Mitgliederdatenbank
```

Falls es Probleme mit der Firewall geben sollte, kann man Apache die Möglichkeit geben, auf die Firewall zuzugreifen:

```
sudo ufw allow 'Apache Full'
```

Zu guter Letzt sollte überprüft werden, ob die Apache-Dateien korrekt konfiguriert sind:

```
sudo apache2ctl configtest
```

Wenn der Output **Syntax OK** ist, ist die Einrichtung abgeschlossen und das Apache2-Gerät kann neugestartet werden:

```
sudo systemctl restart apache2
```

## 2. Datensicherung unter Linux

Im Folgenden ist die Einrichtung eines Cronjobs beschrieben, der jede Woche ein Backup der Datenbank durchführt.

### Voraussetzungen

- der Admin muss über root Rechte verfügen
- ein Terminal muss geöffnet worden sein

### 2.1. Backup einrichten

Schritt	Befehl	Kommentar
1	<code>sudo -i</code>	Login aufrufen
2	<code>[sudo] password: * * *</code>	Passwort eingeben

#### (1) Backup-Skript erstellen und abspeichern

Schritt	Befehl	Kommentar
3	<code>cd /bar</code>	Zu einem beliebigen Verzeichnis wechseln
4	<code>nano db-backup-skript</code>	Backup Skript mit einem beliebigen Editor erstellen und abspeichern

#### *db-backup-skript*

```
#!/bin/bash
DIR=/pfad/zur/datenbank
BACKUPDIR=/gewünschter/speicherort/für/das/backup
WEEK=`date +"%W"`
OLDWEEK=`date -d "-3 week" +"%W"`

#Generiert das Backup
sqlite3 ${DIR}/db.sqlite3 .dump > ${BACKUPDIR}/db-backup-kw${WEEK}.txt

#Löscht Backups, die älter als 3 Wochen alt sind
rm ${BACKUPDIR}/db-backup-kw${OLDWEEK}.txt
```

#### (2) CronJob erstellen und speichern

Schritt	Befehl	Kommentar
5	<code>cd /etc</code>	etc-Verzeichnis aufrufen
6	<code>/nano crontab</code>	crontab mit beliebigen Editor öffnen, CronJob am Ende der Datei einfügen und speichern

```
#Backup-Skript "db-backup-skript" wird jeden Sonntag 00:15 aufgerufen
15 0 * * sun user test -x /bin/db-backup-skript && /bin/db-backup-skript-
>/dev/null 2>&1
```

## 2.2. Wiederherstellung der Datenbank aus einem Backup

Schritt	Befehl	Kommentar
1	<code>cd /backup</code>	Verzeichnis aufrufen, in der das Backup gespeichert wurde
2	<code>sqlite3 foo.sqlite3 &lt; db-backup-kwXX.txt</code>	XX durch die jeweilige Kalenderwoche des Backups ersetzen, aus der die neue Datenbank "foo" generiert werden soll
3	<code>mv /backup/foo.sqlite3 /baz/</code>	Die Datenbank "foo" kann nun in einen beliebigen Ordner verschoben werden



### 3. Update durchführen

Es folgt eine kurze Beschreibung, welche Schritte notwendig sind, um die Anwendung in einem bestehenden Deployment zu aktualisieren.

Schritt	Befehl	Kommentar
1	<code>sudo chown pi:pi ~/Stura-Mitgliederdatenbank/db.sqlite3</code>  <code>sudo chown pi:pi ~/Stura-Mitgliederdatenbank</code>	Berechtigungen werden an den User "pi" zurückgegeben (Nutzername muss angepasst werden)
2	<code>git stash</code>	Die Änderungen zur Konfiguration des Deployments müssen vorübergehend weggespeichert werden
3	<code>git pull</code>	Aktualisierte Version vom Git-Repository laden
4	<code>git stash pop</code>	Konfiguration für das Deployment wieder anwenden
5	<code>sudo chown www-data:www-data ~/Stura-Mitgliederdatenbank/db.sqlite3</code>  <code>sudo chown www-data:www-data ~/Stura-Mitgliederdatenbank</code>	Berechtigungen wieder an den Apache-Nutzer "www-data" übergeben

## 4. Fehlerberichte

Django bietet die Möglichkeit, bei aufgetretenen Fehlern in der Anwendung oder bei "kaputten Links" einen Fehlerbericht per E-Mail an bestimmte Personen (Admins) zu senden. Dieser beinhaltet:

- eine Fehlerbeschreibung,
- ein komplettes Python-Traceback,
- Details über die HTTP-Request, die den Fehler ausgelöst hat.

Um diese Funktionalität zu aktivieren, müssen einige Einstellungen in der Datei `settings.py` getroffen werden.

Siehe dazu: <https://docs.djangoproject.com/en/3.2/howto/error-reporting/#email-reports>