Storage Boxx PHP Wawi

Installation

Beschreibung:

Storageboxx installation in einen Docker container

Vorrausetzung:

```
apt install docker.io docker-compose curl net-tools git
```

Installation

Projektverzeichnis erstellen

```
mkdir -p /root/storageboxx/html
mkdir -p /root/storageboxx/db_data
```

Dockerfile erstellen, denn wir brauchen folgende Module drinnen

- Apache Mod Rewrite
- PHP MYSQL PDO Extension
- PHP OPENSSL Extension

```
nano /root/storageboxx/Dockerfile
```

Inhalt

```
# Ausgangsbild
FROM php:8.1-apache

# Aktivieren von Apache mod_rewrite
RUN a2enmod rewrite

# Installieren der PHP-Erweiterungen
RUN docker-php-ext-install pdo pdo_mysql
```

```
# Optional: Konfiguration anpassen, um .htaccess zu erlauben
RUN sed -i 's/AllowOverride None/AllowOverride All/' /etc/apache2/apache2.conf

# Setze Dateiberechtigungen (optional)
RUN chown -R www-data:www-data /var/www/html \
    && chmod -R 755 /var/www/html

# Exponiere den Standard-HTTP-Port
EXPOSE 80

# Apache neu starten (optional, falls benötigt)
CMD ["apache2-foreground"]
```

docker compose Datei erstellen

```
nano /root/storageboxx/docker-compose.yml
```

Inhalt
Die Passwörter abändern

```
version: '3.8'
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: storageboxx_web
    ports:
      - "80:80"
    volumes:
      - ./html:/var/www/html
    environment:
      - APACHE_DOCUMENT_ROOT=/var/www/html
    depends_on:
      - db

  db:
    image: mariadb:latest
    container_name: storageboxx_db
    restart: always
```

```
    environment:
      MYSQL_DATABASE: storageboxx
      MYSQL_USER: user
      MYSQL_PASSWORD: password
      MYSQL_ROOT_PASSWORD: rootpassword
    volumes:
      - ./db_data:/var/lib/mysql
```

Nun das HTML Verzeichnis mit leben füllen, dazu clonen wir das git repo von storageboxx

```
cd /root/storageboxx/html
git clone https://github.com/code-boxx/Storage-Boxx-PHP-Inventory-Management-System.git
```

Nun in das Verzichnis gehen und alle Dateien eine ebene runter verschieben und dann das leere Verzeichnis löschen

```
cd Storage-Boxx-PHP-Inventory-Management-System
mv * ../
rm -r Storage-Boxx-PHP-Inventory-Management-System
```

Nun Berechtigung fürs Verzeichnis setzten:

```
chown -R www-data:www-data /root/storageboxx/html/
```

Dockerimage bauen

```
docker-compose build
```

Nun den Container starten

```
docker-compose up -d
```

Nun kann im Webbrowser die url zum installieren aufgerufen werden

http://<ip>

Nun die Daten eingeben und weiter

COMPANY

Example Company — Company Name
example@example.com — Company Email
08003301000 — Company Tel
My Adress — Company Address

ADMIN USER

Admin — Name
example@example.com — Email
•••••••• — Password
* At least 8 characters alphanumeric.
•••••••• — Confirm Password

JSON WEB TOKEN

M4~.G__HmT2C8tfdMzRg2ol — Secret Key
* Click here to regenerate a random key.
192.168.178.187 — Issuer
* Your company name or domain name.

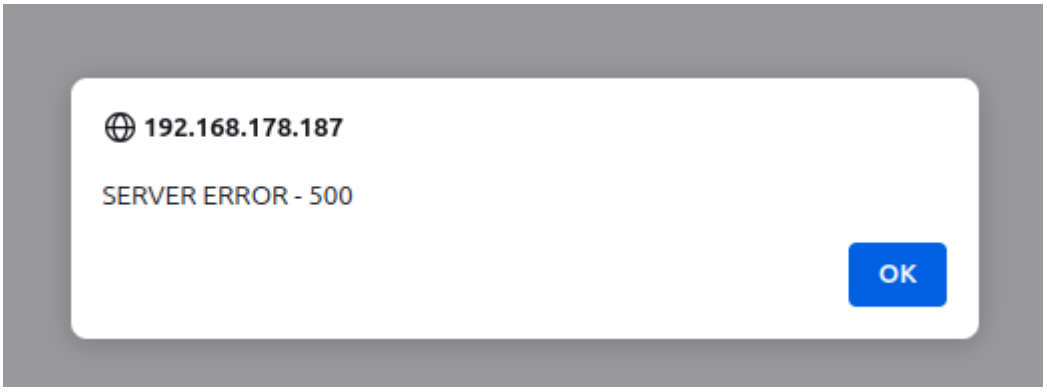
WEB PUSH VAPID KEYS

foFagXlpA3UhnsZTDeMd0Ut — Private Key
BKWBDwZ3HZSdjYe_lRmfXL — Public Key
* You can regenerate these with:

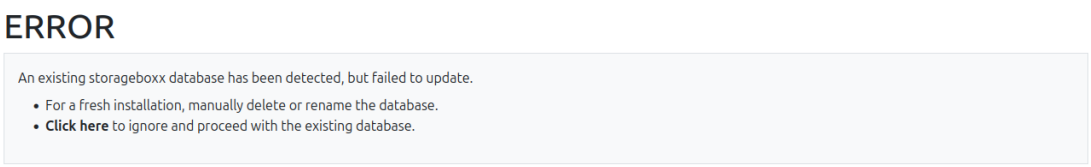
```
require "lib/webpush/autoload.php";
$keys = Minishlink\WebPush\VAPID::createVapidKeys();
```

Go!

Es gibt einen Error 500 Fehler.



Die Seite einfach neu laden.
Dann kommt folgende Meldung, einfach auf click here klciken



Nun haben wir die Anmelde Maske

Werkseinstellungen:

die html Dateien löschen und neuladen und die Datenbank löschen

```
rm -r /root/stoarageboxx/db_data/*
cd /root/storageboxx/html
rm -r *
git clone https://github.com/code-boxx/Storage-Boxx-PHP-Inventory-Management-System.git
cd Storage-Boxx-PHP-Inventory-Management-System
mv * ../
cd ..
rm -r Storage-Boxx-PHP-Inventory-Management-System
chown -R www-data:www-data /root/storageboxx/html/
```

HTTPS hinzufügen

Beschreibung:

Damit Push und Kamera funktioniert muss HTTPS benutzt werden.

Implementierung:

alles stoppen

```
docker-compose down
```

docker file anpassen

In der docker-compose Datei nginx Container hinzufügen :

```
nano /root/storageboxx/docker-compose.yml
```

Unsere neue Docker file

```
version: '3.8'
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: storageboxx_web
    #ports:
     # - "80:80"
    volumes:
      - ./html:/var/www/html
      - ./caddy_data:/data  # Caddy benötigt diesen Ordner für seine Konfiguration
      - ./caddy_config:/config  # Caddy benötigt diesen Ordner für seine Konfiguration
    environment:
      - APACHE_DOCUMENT_ROOT=/var/www/html
    depends_on:
      - db
```

```
    restart: unless-stopped



  db:
    image: mariadb:latest
    container_name: storageboxx_db
    restart: always
    environment:
      MYSQL_DATABASE: storageboxx
      MYSQL_USER: user
      MYSQL_PASSWORD: password
      MYSQL_ROOT_PASSWORD: rootpassword
    volumes:
      - ./db_data:/var/lib/mysql
  nginx:
    image: nginx:stable
    container_name: nginx-proxy
    volumes:
      - ./nginx-proxy.conf:/etc/nginx/nginx.conf:ro
      - ./certs/selfsigned.crt:/etc/ssl/certs/selfsigned.crt:ro
      - ./certs/private.key:/etc/ssl/private/private.key:ro
    ports:
      - "80:80"
      - "443:443"
    depends_on:
      - web
    restart: unless-stopped
```

Nun das cert Verzeichnis erstellen

```
mkdir -p /root/storageboxx/certs
```

Nun die ngninx conf erstellen

```
nano /root/storageboxx/nginx-proxy.conf
```

Inhalt

```
events {
  worker_connections 1024;
}
```

```
http {
  server {
    listen 80;
    return 301 https://$host$request_uri;
  }

  server {
    listen 443 ssl;
    ssl_certificate /etc/ssl/certs/selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/private.key;

    location / {
      proxy_pass http://web:80;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
      client_max_body_size 100M;
    }
  }
}
```

Nun das Zertifikat erstellen

```
openssl req -newkey rsa:4096 -x509 -sha256 -days 365000 -nodes -out /root/storageboxx/certs/selfsigned.crt -keyout /root/storageboxx/certs/private.key
```

Die Fragen benatworten.

Nun in der config von Storageboxx https ändern

```
nano /root/storageboxx/html/lib/CORE-Config.php
```

alt

```
<?php
// (A) HOST
define("SITE_NAME", "Storage Boxx");
define("HOST_BASE", "http://192.168.178.187/"); // CHANGED BY INSTALLER  #um diese zeile geht es !!!!!!!
define("HOST_NAME", parse_url(HOST_BASE, PHP_URL_HOST));
```

```
define("HOST_BASE_PATH", parse_url(HOST_BASE, PHP_URL_PATH));
define("HOST_ASSETS", HOST_BASE . "assets/");

// (B) API ENDPOINT
define("HOST_API", "api/");
```

Neu

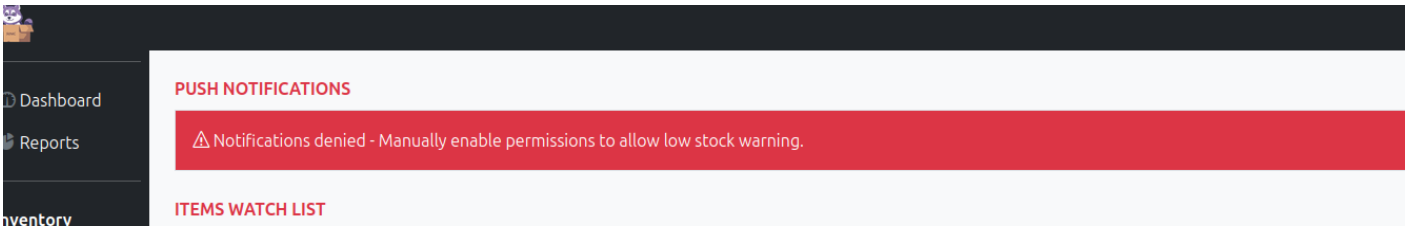
```
<?php
// (A) HOST
define("SITE_NAME", "Storage Boxx");
define("HOST_BASE", "https://192.168.178.187/"); // CHANGED BY INSTALLER
define("HOST_NAME", parse_url(HOST_BASE, PHP_URL_HOST));
define("HOST_BASE_PATH", parse_url(HOST_BASE, PHP_URL_PATH));
define("HOST_ASSETS", HOST_BASE . "assets/");

// (B) API ENDPOINT
define("HOST_API", "api/");
....
```

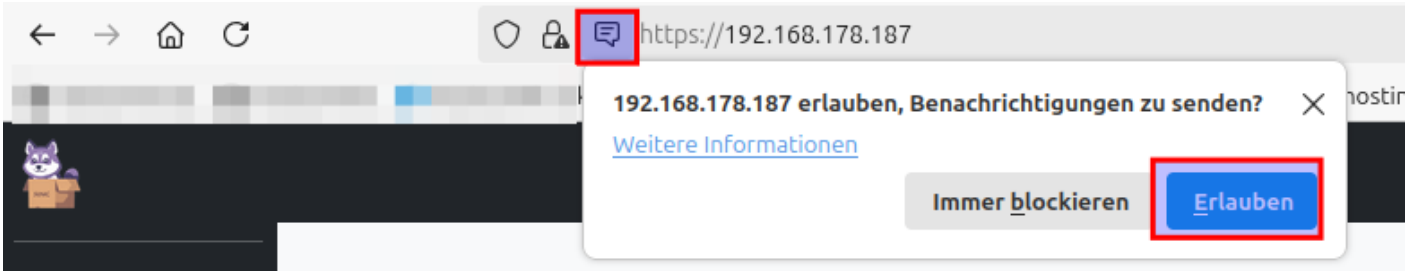
Den container starten

```
docker-compose up -d
```

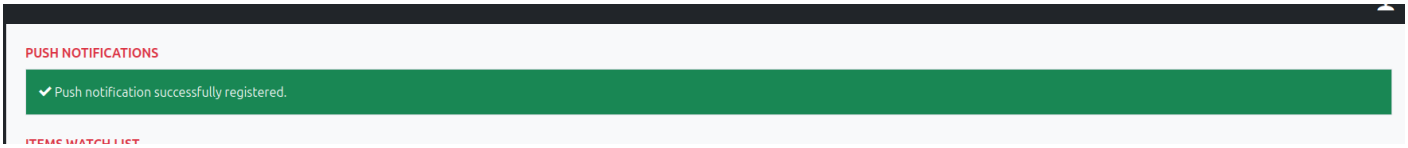
Nun die Seite laden Zertifikat akzeptieren und einloggen.
Wenn dieser Fehler kommt.



Benachrichtigungen zulassen im Browser



Push funktioniert



Fertig.

Beschreibung in der Suche hinzufügen

Beschreibung:

Wenn man einen QR-Code oder NFC scannt möchte man nicht nur wissen, wie viel man hat sondern auch wo.
Hier wird folgende HTMl Datei angepasst.
Da wir unser HTMl ja in Docker weiterleiten ist das sehr leicht zu bearbeiten.

Datei anpassen:

Vorher Sicherung anlegen:

```
cp /root/storageboxx/html/pages/PAGE-check-main.php /root/storageboxx/html/pages/PAGE-check-main.php.bak
```

Editieren

```
nano /root/storageboxx/html/pages/PAGE-check-main.php
```

Inhalt

Vorher

```
<?php
// (A) GET ITEM
$item = $_CORE->autoCall("Items", "get"); ?>
<!-- (B) NAVIGATION -->
<nav class="d-flex align-items-center">
  <div class="flex-grow-1">
    <div class="display-6">
      [<?=$item["item_sku"]?>] <?=$item["item_name"] ?>
    </div>
    <div class="fw-bold">STOCK : <?=$item["item_qty"]?> <?=$item["item_unit"]?></div>
  </div>
```

```
  <button type="button" class="btn btn-danger p-3 mx-1 ico-sm icon-undo2" onclick="cb.page(1)"></button>
</nav>

<!-- (C) ITEM MOVEMENT HISTORY -->
<div id="check-list" class="zebra my-4"></div>
```

Wir ändern diesen Abschnitt und fügen
Diesen Tag hinzu ans ende

```
desc: <?=$item["item_desc"]?>
```

Dann sieht die Zeile nun so aus

```
  </div>
    <div class="fw-bold">STOCK : <?=$item["item_qty"]?> <?=$item["item_unit"]?> desc:
<?=$item["item_desc"]?></div>
  </div>
```

Nun noch mal die Ganze geänderte Datei

```
<?php
// (A) GET ITEM
$item = $_CORE->autoCall("Items", "get"); ?>
<!-- (B) NAVIGATION -->
<nav class="d-flex align-items-center">
  <div class="flex-grow-1">
    <div class="display-6">
      [<?=$item["item_sku"]?>] <?=$item["item_name"] ?>
    </div>
    <div class="fw-bold">STOCK : <?=$item["item_qty"]?> <?=$item["item_unit"]?> desc:
<?=$item["item_desc"]?></div>
  </div>
  <button type="button" class="btn btn-danger p-3 mx-1 ico-sm icon-undo2" onclick="cb.page(1)"></button>
</nav>

<!-- (C) ITEM MOVEMENT HISTORY -->
<div id="check-list" class="zebra my-4"></div>
```