Betriebsdokumentation

Mobiles Logbuch für Modellflugplatz (E09)

Inhaltsverzeichnis

1. Systemvoraussetzungen

Hardwareanforderungen:

- CPU: Mindestens Dual-Core-Prozessor (Empfohlen: Quad-Core)
- RAM: Mindestens 2 GB (Empfohlen: 4 GB)
- Festplatte: Mindestens 50 GB freier Speicherplatz, jedoch deutlich mehr empfohlen
- Netzwerk: Stabile Netzwerkverbindung (Empfohlen: Ethernet-Verbindung)

Softwareanforderungen:

- Betriebssystem: Linux (z.B. Ubuntu noble 24.04 LTS oder neuer)
- Datenbank: PostgreSQL 16 oder neuer empfohlen
- Reverse-Proxy: Nginx (stable) empfohlen
- Browser: Aktuelle Version von Firefox, Chrome oder Edge
    - auf dem Smartphone den offiziellen Browser des Herstellers, um die PWA installieren zu können

2. Systemeinrichtung

2.1. Installationsguide für Backend und PostgreSQL

Diese Anleitung beschreibt die manuelle Installation des Backends auf einem (möglichst "frischen") Ubuntu Server.

2.1.1. Voraussetzungen

Bevor du beginnst, stelle sicher, dass folgende Voraussetzungen erfüllt sind:

1. Ein Ubuntu-Server (oder LXC-Container oder eine VM) mit Root-Zugriff.
2. Eine funktionierende Internetverbindung.
3. Möglichkeit der Konfiguration der Firewall.

2.1.2. Angestrebte Verzeichnisstruktur

Mit den folgenden Schritten soll diese Verzeichnisstruktur erreicht werden:

```
.
├──── /home/logbuch
│    ├──── modellflug-logbuch
│    │    ├──── bin
│    │    │    └──── modellflug-logbuch
│    │    ├──── lib
│    │    └──── config
│    │         └──── config.toml
└──── ...
```

und das Systemd Unit File bei `/etc/systemd/system/logbuch.service`.

2.1.3. Schritt 0: Separaten Nutzer anlegen

Es wird empfohlen, einen separaten Nutzer für den Backend-Server zu erstellen:

```
sudo useradd -m logbuch
```

2.1.4. Schritt 1: Update und Upgrade des Systems

Führe zuerst ein Update und Upgrade des Systems durch:

```
sudo apt update
sudo apt upgrade -y
```

2.1.5. Schritt 2: Installieren von Java

Der Backend-Server benötigt Java, um ausgeführt zu werden. Installiere OpenJDK:

```
sudo apt install openjdk-17-jdk -y
```

Überprüfe die Installation:

```
java --version
```

2.1.6. Schritt 3: Installieren und Konfigurieren von PostgreSQL

Installiere PostgreSQL als Datenbankserver:

```
sudo apt install postgresql -y
```

Starte und aktiviere den PostgreSQL-Dienst:

```
sudo systemctl enable --now postgresql
```

2.1.7. Schritt 4: Erstellen einer Datenbank und eines Benutzers in PostgreSQL

Wechsle zum PostgreSQL-Benutzer und öffne die PostgreSQL-Shell:

```
sudo -i -u postgres
psql
```

Erstelle eine neue Datenbank und einen neuen Benutzer:

```
CREATE DATABASE logbook_db;
CREATE USER myuser WITH ENCRYPTED PASSWORD 'SET_PASSWORD_HERE';
GRANT ALL PRIVILEGES ON DATABASE logbook_db TO logbook_user;
```

Beende die PostgreSQL-Shell:

```
q
```

Wechsle zurück zum vorherigen Benutzer:

```
exit
```

2.1.8. Schritt 5: Installieren des Backend-Servers

Lade das gepackte Backend-Projekt aus den GitHub Releases herunter. Das Projekt liegt als `.tar.gz` oder `.zip` Datei vor. Lade die Datei auf deinen Server hoch und entpacke sie:

```
# für eine .tar.gz Datei
tar -xvzf releasename.tar.gz

# für eine .zip Datei
unzip releasename.zip
```

Benenne das entpackte Verzeichnis um und navigiere in das Verzeichnis:

```
mv releasename modellflug-logbuch
cd modellflug-logbuch
```

Stelle sicher, dass die Datei `bin/modellflug-logbuch` ausführbar ist:

```
chmod +x bin/modellflug-logbuch
```

2.1.9. Schritt 6: Konfigurieren des Backend-Servers

Führe den Server einmal manuell aus um die Konfigurationsdatei zu generieren:

```
./bin/modellflug-logbuch
```

Bearbeite die TOML-Konfigurationsdatei `config.toml`, um die Datenbankverbindungsdaten und andere Einstellungen festzulegen. Beispiel:

```
# The time zone to use for the application.
# If not set, the system default is used.
# See https://kotlinlang.org/api/kotlinx-datetime/kotlinx-datetime/kotlinx.datetime/-
time-zone/-companion/of.html for more information.
timeZone = "Europe/Berlin"

[database]
  serverName = "localhost"
  # The PostgreSQL default port is 5432.
  port = 5432
  username = "logbook_user"
  password = "SET_PASSWORD_HERE"
  # The name of the database to use. (You chose this during database setup.)
  databaseName = "logbook_db"

[server]
```

```
# The port the server should listen on.
port = 8080
# In development mode the server runs without many security features.
# This is useful for testing and debugging in local environments.
developmentMode = false
```

Stelle sicher, dass der `developmentMode` auf `false` gesetzt ist.

2.1.10. Schritt 7: Server als Dienst einrichten

Es ist ratsam, den Backend-Server als Systemdienst zu konfigurieren, damit er automatisch startet:

Erstelle eine neue Dienstdatei:

```
sudo nano /etc/systemd/system/logbuch.service
```

Füge folgendes in **angepasster** Form in die Datei ein:

```
[Unit]
Description=Modellflug Logbuch
After=network.target

[Service]
User=your-username
ExecStart=/your/path/bin/modellflug-logbuch
WorkingDirectory=/your/path/project
Restart=unless-stopped

[Install]
WantedBy=multi-user.target
```

Aktualisiere die Systemdienste und starte den Dienst:

```
sudo systemctl daemon-reload
sudo systemctl enable --now logbuch
```

2.1.11. Schritt 8: Überprüfung

Überprüfe, ob der Server läuft:

```
sudo systemctl status logbuch
```

Besuche schließlich deine Server-IP oder Domain auf Port 8080 (oder dem in deinem Projekt konfigurierten Port), um sicherzustellen, dass der Ktor-Server und das eingebettete Frontend korrekt laufen.

2.1.12. Schritt 9: Reverse Proxy (optional)

Wenn du den Ktor-Server über einen Reverse Proxy wie Nginx oder Apache bereitstellen möchtest, konfiguriere den Proxy entsprechend.

Falls der jeweilige Modellflugverein bereits einen Webserver bzw. Reverse Proxy betreibt, sollte dieser genutzt werden.

Nginx Beispiel

Erstelle eine neue Konfigurationsdatei und füge z.B. folgendes in die Datei ein:

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:8080;
        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;
    }
}
```

Apache Beispiel

Erstelle eine neue Konfigurationsdatei und füge z.B. folgendes in die Datei ein:

```
<VirtualHost *:80>
    ServerName your-domain.com

    ProxyPreserveHost On
    ProxyPass / http://localhost:8080/
    ProxyPassReverse / http://localhost:8080/
</VirtualHost>
```

2.2. Konfigurationsoptionen

2.2.1. Allgemeine Einstellungen

Top Level Scope

`timeZone`

- Beschreibung: Gibt die Zeitzone an, die von der Anwendung verwendet werden soll. Wenn diese Option nicht gesetzt ist, wird die Standardzeitzone des Systems verwendet.
- Mögliche Werte: Ein gültiger Zeitzonen-String (z.B. "Europe/Berlin"). Siehe Kotlinx DateTime

TimeZone für mehr Informationen.

- Standardwert: `null`
- Beispielwert: `"Europe/Berlin"`

2.2.2. Datenbankeinstellungen

`[database]` Scope

`serverName`

- Beschreibung: Der Name oder die IP-Adresse des PostgreSQL-Servers.
- Beispiel: `"localhost"`

`port`

- Beschreibung: Der Port, auf dem PostgreSQL lauscht.
- Standardwert: `5432`

`username`

- Beschreibung: Der Benutzername, der für die Verbindung zur Datenbank verwendet wird.
- Beispiel: `"logbook_user"`

`password`

- Beschreibung: Das Passwort für den Datenbankbenutzer.
- Standardwert: `"SET_PASSWORD_HERE"`
- muss geändert werden

`databaseName`

- Beschreibung: Der Name der Datenbank, die verwendet werden soll. Dieser Name wurde während der Datenbankeinrichtung festgelegt.
- Beispiel: `"logbook_db"`

2.2.3. Servereinstellungen

`[server]` Scope

`port`

- Beschreibung: Der Port, auf dem der Anwendungsserver lauschen soll.
- Beispiel: `8080`

`developmentMode`

- Beschreibung: Wenn auf `true` gesetzt, läuft der Server im Entwicklungsmodus, wobei viele Sicherheitsfunktionen deaktiviert sind. Dies ist nützlich für Tests und Debugging in lokalen Umgebungen.

- Mögliche Werte: `true` (Entwicklungsmodus) oder `false` (Produktionsmodus)
- Standardwert: `false`

3. Systembetreuung

3.1. FAQ für Benutzersupport

> Wie kann ich mich einloggen?

Stellen Sie sicher, dass Sie die richtigen Anmeldedaten verwenden. Wenn Sie Ihr Passwort vergessen haben, kontaktieren Sie den Administrator.

> Was tun bei Verbindungsproblemen?

Überprüfen Sie Ihre Netzwerkverbindung und stellen Sie sicher, dass der Server läuft. Überprüfen Sie zusätzlich in der Sektion "Konfiguration" die Einstellung für die Backend-URL.

> Ich bekomme ständig den Fehler 401 Unauthorized. Was kann ich tun?

Deine Session ist wahrscheinlich corrupted. Klicke auf den Logout Button um das Problem zu beheben.

3.2. Fehlerdiagnose

Überprüfe insbesondere den Status des Systemd Dienstes:

```
sudo systemctl status logbook
```

Überprüfe die Logdatei des Backends:

```
sudo journalctl -u logbook
```

Dort werden insbesondere Fehler, der aktuelle Status der Anwendung und Probleme beim Auslesen der Konfigurationsdatei `config.toml` angezeigt.

3.3. Datensicherung und -wiederherstellung

Die Inhalte der Datebank können z.B. in mittels `pg_dump` in `.sql` Dateien gesichert werden.

Datensicherung:

```
pg_dump -U logbook_user logbook_db > /backup/logbook_db_backup.sql
```

Datenwiederherstellung:

```
psql -U logbook_user -d logbook_db -f /backup/logbook_db_backup.sql
```

4. Weiterführende Links

- Dokumentation von PostgreSQL: PostgreSQL Dokumentation
- Dokumentation von Nginx: Nginx Dokumentation
- `systemd` Unit Files Guide: Writing unit files
- Mfc-Rossendorf: Mfc-Rossendorf