Backend-Unterstützung für QR-Code-Scan: Endpunkt definieren und absichern

Ihr Name

5. Februar 2025

Inhaltsverzeichnis

1. Einleitung

In diesem Abschnitt erstellen wir einen neuen API-Endpunkt im Express-Backend, der basierend auf einer übergebenen Geräte-ID (`deviceId`) die zugehörige Trainingshistorie zurückliefert. Der Endpunkt wird unter `/api/device/:id` erreichbar sein. Zusätzlich implementieren wir Maßnahmen zur Datenvalidierung und Fehlerbehandlung, um sicherzustellen, dass der Endpunkt robust gegenüber ungültigen Eingaben ist. Wir gehen dabei auch darauf ein, wie die Integration in die bestehende `server.js` erfolgt und was beim Testen zu beachten ist.

2. Schritt-für-Schritt-Anleitung

2.1 Schritt 1: Endpunkt in der `server.js` definieren

1. **Datei öffnen:** Öffnen Sie Ihre `server.js`-Datei im Backend-Ordner.
2. **Bestehende Routen lokalisieren:** Suchen Sie den Abschnitt, in dem bereits andere Routen definiert sind (z. B. die GET-Anfrage für `'/'`).
3. **Neuen Endpunkt hinzufügen:** Fügen Sie unterhalb der bestehenden Routen den folgenden Code ein:

```
app.get('/api/device/:id', async (req, res) => {
    const deviceId = req.params.id;
    // Datenvalidierung: Überprüfen, ob die Geräte-ID vorhanden und numerisch ist
    if (!deviceId || isNaN(deviceId)) {
        return res.status(400).json({ error: 'Ungültige Geräte-ID' });
    }

    try {
        // Beispiel: Abfrage der Trainingshistorie für das gegebene Gerät aus der
        // Wir verwenden eine parameterisierte Query, um SQL-Injektionen zu verme
        const result = await pool.query(
            'SELECT * FROM training_history WHERE device_id = $1',
            [deviceId]
        );

        // Prüfung: Wenn keine Trainingsdaten gefunden wurden, senden wir einen 4
        if (result.rows.length === 0) {
            return res.status(404).json({ error: 'Keine Trainingshistorie für Ger
        }

        // Erfolgreiche Rückgabe der Trainingshistorie
        res.json({ message: 'Trainingshistorie für Gerät ' + deviceId, data: resu
    } catch (error) {
        console.error('Fehler beim Abrufen der Trainingshistorie:', error.message
        res.status(500).json({ error: 'Serverfehler beim Abrufen der Trainingshis
    }
});
```

2.2 Schritt 2: Datenvalidierung und Fehlerbehandlung implementieren

1. **Validierung der Geräte-ID:**
   - Verwenden Sie `if (!deviceId || isNaN(deviceId))` um sicherzustellen, dass die Geräte-ID existiert und numerisch ist.
   - Bei ungültiger ID senden Sie eine Antwort mit HTTP-Statuscode 400 (Bad Request) und einer entsprechenden Fehlermeldung.
2. **Fehlerbehandlung der Datenbankabfrage:**
   - Umgeben Sie die SQL-Abfrage mit einem `try-catch`-Block.
   - Im `catch`-Block geben Sie einen HTTP-Statuscode 500 (Internal Server Error) zurück, zusammen mit einer aussagekräftigen Fehlermeldung.
3. **Prüfung, ob Trainingsdaten gefunden wurden:**
   - Überprüfen Sie, ob `result.rows.length` gleich 0 ist.
   - Falls ja, senden Sie einen HTTP-Statuscode 404 (Not Found) mit einer entsprechenden Nachricht zurück.

2.3 Schritt 3: Integration in die bestehende `server.js`

1. **Überprüfen des Pool-Objekts:** Stellen Sie sicher, dass am Anfang Ihrer `server.js` das Pool-Objekt aus der `db.js` importiert ist:

   ```
   const pool = require('./db');
   ```

2. **Endpunkt einfügen:** Fügen Sie den oben erstellten Endpunkt an einer sinnvollen Stelle ein – idealerweise unterhalb der anderen API-Routen, aber vor dem `app.listen`-Aufruf.
3. **Datei speichern und Server neu starten:** Speichern Sie die `server.js`-Datei und starten Sie Ihren Backend-Server neu (z. B. mit `node server.js`).

3. Zusammenfassung

In diesem Schritt haben Sie:

- Einen neuen GET-Endpunkt unter `/api/device/:id` erstellt, der die Geräte-ID aus der URL extrahiert.
- Eine Validierung implementiert, um ungültige Geräte-IDs abzufangen (HTTP 400 bei Fehlern).

- Eine SQL-Abfrage mittels parameterisierter Query ausgeführt, um die Trainingshistorie des entsprechenden Geräts abzurufen.
- Eine Fehlerbehandlung eingebaut, die aussagekräftige Fehlermeldungen (HTTP 404, 500) zurückgibt, falls keine Daten gefunden werden oder ein Serverfehler auftritt.
- Den neuen Endpunkt in die bestehende `server.js` integriert und den Server neu gestartet.

Diese detaillierte Anleitung stellt sicher, dass Ihr Backend-Endpunkt robust und sicher arbeitet und dass er die zugehörige Trainingshistorie basierend auf der übergebenen Geräte-ID korrekt zurückliefert.