Registrierungs-Endpunkt erstellen – Detaillierte Schritt-für-Schritt-Anleitung

Ihr Name

4. Februar 2025

Inhaltsverzeichnis

1 Einleitung

In diesem Abschnitt erstellen wir einen Registrierungs-Endpunkt (`/api/register`) in unserem Express-Backend. Dieser Endpunkt nimmt Nutzereingaben wie Name, E-Mail und Passwort entgegen, hasht das Passwort mithilfe von `bcryptjs`, speichert die Daten in unserer PostgreSQL-Datenbank (über das `pool`-Objekt aus unserer `db.js`) und gibt eine Erfolgsmeldung zurück – optional inklusive eines JWT-Tokens zur Authentifizierung. Wir gehen jeden Schritt detailliert durch, sodass auch Einsteiger den Prozess nachvollziehen können.

2 Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass:

- Ihr Backend-Projekt bereits eingerichtet ist (Node.js, Express, CORS, etc.).
- Die Datei `db.js` existiert und eine Verbindung zu Ihrer PostgreSQL-Datenbank herstellt.
- Die benötigten Pakete `bcryptjs` und `jsonwebtoken` installiert sind:

  ```
  npm install bcryptjs jsonwebtoken
  ```

- In Ihrer Datenbank eine Tabelle `users` existiert, z. B. mit folgendem SQL-Befehl:

  ```
  CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
  );
  ```

3 Schritt 1: Endpunkt definieren

Öffnen Sie Ihre `server.js`-Datei im Backend-Ordner. Fügen Sie unterhalb der bereits vorhandenen Routen (z. B. der GET-Anfrage für '/') einen neuen POST-Endpunkt hinzu, der Anfragen an `/api/register` entgegennimmt.

4 Schritt 2: Nutzerdaten aus dem Request entnehmen

Verwenden Sie `express.json()` (bereits in Ihrem Setup), um JSON-Daten aus dem Request-Body zu lesen. Im Registrierungs-Endpunkt extrahieren Sie die Felder `name`, `email` und `password`.

5 Schritt 3: Passwort mit bcryptjs hashen

Nutzen Sie `bcryptjs` um das übergebene Passwort zu hashen:

- Erzeugen Sie ein Salt mit `bcrypt.genSalt(10)`.
- Haschen Sie das Passwort mit `bcrypt.hash(password, salt)`.

6 Schritt 4: Nutzerdaten in der Datenbank speichern

Verwenden Sie das in `db.js` exportierte `pool`-Objekt, um eine SQL-Abfrage auszuführen, die die Nutzerdaten in die Tabelle `users` einfügt. Nutzen Sie dabei parameterisierte Queries, um SQL-Injektionen zu vermeiden.

7 Schritt 5: JWT-Token generieren (optional)

Nach erfolgreicher Registrierung können Sie ein JWT-Token generieren, das z. B. die Benutzer-ID enthält. Nutzen Sie dazu `jsonwebtoken`:

- Verwenden Sie `jwt.sign(payload, secret, options)`, wobei `payload` mindestens die Benutzer-ID enthalten sollte.
- Legen Sie ein Ablaufdatum fest, z. B. `expiresIn: '1h'`.

8 Schritt 6: Erfolgsmeldung und Fehlerbehandlung

Geben Sie bei Erfolg eine JSON-Antwort zurück, die eine Erfolgsmeldung und das Token (falls generiert) enthält. Nutzen Sie `try-catch`-Blöcke, um Fehler abzufangen, und geben Sie bei Fehlern einen Statuscode (z. B. 500) sowie eine entsprechende Nachricht zurück.

9 Beispielcode für den Registrierungs-Endpunkt

Fügen Sie den folgenden Code in Ihre `server.js` ein (ohne die bestehenden Teile zu entfernen):

```
app.post('/api/register', async (req, res) => {
    const { name, email, password } = req.body;
    try {
        // Passwort hashen
        const salt = await bcrypt.genSalt(10);
        const hashedPassword = await bcrypt.hash(password, salt);

        // Benutzer in der Datenbank speichern
        const newUser = await pool.query(
            'INSERT INTO users (name, email, password) VALUES ($1, $2, $3) RET
            [name, email, hashedPassword]
        );
```

```
        // JWT-Token generieren (optional)
        const token = jwt.sign(
            { userId: newUser.rows[0].id },
            'IhrGeheimerSchluessel', // Ersetzen Sie diesen Schl ssel durch e
            { expiresIn: '1h' }
        );

        res.json({ message: 'Benutzer erfolgreich registriert', token });
    } catch (error) {
        console.error(error.message);
        res.status(500).send('Serverfehler');
    }
});
```

10 Zusammenfassung

- Öffnen Sie Ihre `server.js` im Backend-Ordner.
- Definieren Sie einen POST-Endpunkt `/api/register`, der JSON-Daten entgegennimmt.
- Extrahieren Sie in diesem Endpunkt die Felder `name`, `email` und `password` aus dem Request.
- Verwenden Sie `bcryptjs`, um das Passwort zu hashen.
- Speichern Sie die Daten mittels einer parameterisierten SQL-Abfrage in der Tabelle `users`.
- Generieren Sie optional ein JWT-Token, um die Authentifizierung zu unterstützen.[a4paper,12pt]article [utf8]inputenc [T1]fontenc [ngerman]babel enumitem listings geometry left=2.5cm, right=2.5cm, top=2.5cm, bottom=2.5cm titlesec hyperref

  Registrierungs-Endpunkt erstellen – Detaillierte Schritt-für-Schritt-Anleitung
  Ihr Name 4. Februar 2025

Inhaltsverzeichnis

11 Einleitung

In diesem Abschnitt erstellen wir einen Registrierungs-Endpunkt (`/api/register`) in unserem Express-Backend. Dieser Endpunkt nimmt Nutzereingaben wie Name, E-Mail und Passwort entgegen, hasht das Passwort mithilfe von `bcryptjs`, speichert die Daten in unserer PostgreSQL-Datenbank (über das `pool`-Objekt aus unserer `db.js`) und gibt eine Erfolgsmeldung zurück – optional inklusive eines JWT-Tokens zur Authentifizierung. Wir gehen jeden Schritt detailliert durch, sodass auch Einsteiger den Prozess nachvollziehen können.

12 Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass:

- Ihr Backend-Projekt bereits eingerichtet ist (Node.js, Express, CORS, etc.).
- Die Datei `db.js` existiert und eine Verbindung zu Ihrer PostgreSQL-Datenbank herstellt.
- Die benötigten Pakete `bcryptjs` und `jsonwebtoken` installiert sind:

  ```
  npm install bcryptjs jsonwebtoken
  ```

- In Ihrer Datenbank eine Tabelle `users` existiert, z. B. mit folgendem SQL-Befehl:

  ```
  CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
  );
  ```

13 Schritt 1: Endpunkt definieren

Öffnen Sie Ihre `server.js`-Datei im Backend-Ordner. Fügen Sie unterhalb der bereits vorhandenen Routen (z. B. der GET-Anfrage für '/') einen neuen POST-Endpunkt hinzu, der Anfragen an `/api/register` entgegennimmt.

14 Schritt 2: Nutzerdaten aus dem Request entnehmen

Verwenden Sie `express.json()` (bereits in Ihrem Setup), um JSON-Daten aus dem Request-Body zu lesen. Im Registrierungs-Endpunkt extrahieren Sie die Felder `name`, `email` und `password`.

15 Schritt 3: Passwort mit bcryptjs hashen

Nutzen Sie `bcryptjs` um das übergebene Passwort zu hashen:

- Erzeugen Sie ein Salt mit `bcrypt.genSalt(10)`.
- Haschen Sie das Passwort mit `bcrypt.hash(password, salt)`.

16 Schritt 4: Nutzerdaten in der Datenbank speichern

Verwenden Sie das in `db.js` exportierte `pool`-Objekt, um eine SQL-Abfrage auszuführen, die die Nutzerdaten in die Tabelle `users` einfügt. Nutzen Sie dabei parameterisierte Queries, um SQL-Injektionen zu vermeiden.

17 Schritt 5: JWT-Token generieren (optional)

Nach erfolgreicher Registrierung können Sie ein JWT-Token generieren, das z. B. die Benutzer-ID enthält. Nutzen Sie dazu `jsonwebtoken`:

- Verwenden Sie `jwt.sign(payload, secret, options)`, wobei `payload` mindestens die Benutzer-ID enthalten sollte.
- Legen Sie ein Ablaufdatum fest, z. B. `expiresIn: '1h'`.

18 Schritt 6: Erfolgsmeldung und Fehlerbehandlung

Geben Sie bei Erfolg eine JSON-Antwort zurück, die eine Erfolgsmeldung und das Token (falls generiert) enthält. Nutzen Sie `try-catch`-Blöcke, um Fehler abzufangen, und geben Sie bei Fehlern einen Statuscode (z. B. 500) sowie eine entsprechende Nachricht zurück.

19 Beispielcode für den Registrierungs-Endpunkt

Fügen Sie den folgenden Code in Ihre `server.js` ein (ohne die bestehenden Teile zu entfernen):

```
app.post('/api/register', async (req, res) => {
    const { name, email, password } = req.body;
    try {
        // Passwort hashen
        const salt = await bcrypt.genSalt(10);
        const hashedPassword = await bcrypt.hash(password, salt);

        // Benutzer in der Datenbank speichern
```

```
        const newUser = await pool.query(
            'INSERT INTO users (name, email, password) VALUES ($1, $2, $3
            [name, email, hashedPassword]
        );

        // JWT-Token generieren (optional)
        const token = jwt.sign(
            { userId: newUser.rows[0].id },
            'IhrGeheimerSchluessel', // Ersetzen Sie diesen Schl ssel du
            { expiresIn: '1h' }
        );

        res.json({ message: 'Benutzer erfolgreich registriert', token });
    } catch (error) {
        console.error(error.message);
        res.status(500).send('Serverfehler');
    }
});
```

20 Zusammenfassung

- Öffnen Sie Ihre `server.js` im Backend-Ordner.
- Definieren Sie einen POST-Endpunkt `/api/register`, der JSON-Daten entgegennimmt.
- Extrahieren Sie in diesem Endpunkt die Felder `name`, `email` und `password` aus dem Request.
- Verwenden Sie `bcryptjs`, um das Passwort zu hashen.
- Speichern Sie die Daten mittels einer parameterisierten SQL-Abfrage in der Tabelle `users`.
- Generieren Sie optional ein JWT-Token, um die Authentifizierung zu unterstützen.
- Geben Sie eine Erfolgsmeldung (und das Token) zurück und fangen Sie Fehler ab.

Mit diesen Schritten und dem Beispielcode haben Sie eine klare Anleitung, wie Sie den Registrierungs-Endpunkt in Ihrem Express-Backend umsetzen können. Dies ist ein wesentlicher Bestandteil der Nutzerverwaltung in Ihrem Gym Progress Tracking System.