# Formulare erstellen in React – Schritt-für-Schritt-Anleitung

Ihr Name

5. Februar 2025

## Inhaltsverzeichnis

# 1 Einleitung

In diesem Abschnitt erstellen wir zwei separate Komponenten in unserem React-Projekt: eine Komponente für die Registrierung (Registrierungsformular) und eine Komponente für den Login (Login-Formular). Ziel ist es, dass Benutzer ihre Daten (z. B. Name, E-Mail und Passwort) eingeben können und diese Daten dann per POST-Anfrage an unsere Backend-Endpunkte (`/api/register` bzw. `/api/login`) gesendet werden. Diese Anleitung führt Sie Schritt für Schritt durch den Prozess.

# 2 Schritt 1: Erstellen des Registrierungsformulars

## 2.1 1.1 Komponente anlegen

1. Navigieren Sie in Ihrem React-Projekt in den Ordner `src`.
2. Erstellen Sie eine neue Datei namens `RegistrationForm.js`.

## 2.2 1.2 Grundgerüst der Komponente schreiben

Öffnen Sie `RegistrationForm.js` und fügen Sie folgenden Basis-Code ein:

```
import React, { useState } from 'react';

function RegistrationForm() {
  // State-Variablen für die Formularfelder
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [confirmPassword, setConfirmPassword] = useState('');
  const [error, setError] = useState('');
  const [success, setSuccess] = useState('');

  // Funktion, die beim Absenden des Formulars ausgeführt wird
  const handleSubmit = async (event) => {
    event.preventDefault();

    // Einfache Validierung: Überprüfen, ob alle Felder ausgefüllt sind
    if (!name || !email || !password || !confirmPassword) {
      setError('Bitte füllen Sie alle Felder aus.');
      return;
    }

    // Validierung: Passwort und Passwort-Bestätigung müssen übereinstimmen
    if (password !== confirmPassword) {
      setError('Die Passwörter stimmen nicht überein.');
      return;
    }
```

```
  // Daten als Objekt zusammenfassen
  const userData = { name, email, password };

  try {
    // API-Anfrage an den Registrierungs-Endpunkt
    const response = await fetch('http://localhost:5000/api/register', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(userData),
    });
    const result = await response.json();

    if (response.ok) {
      setSuccess(result.message);
      setError('');
    } else {
      setError(result.error || 'Registrierung fehlgeschlagen.');
    }
  } catch (error) {
    console.error('Registrierungsfehler:', error);
    setError('Ein Fehler ist aufgetreten.');
  }
};

return (
  <div>
    <h2>Registrierung</h2>
    {error && <p style={{color: 'red'}}>{error}</p>}
    {success && <p style={{color: 'green'}}>{success}</p>}
    <form onSubmit={handleSubmit}>
      <div>
        <label>Name:</label>
        <input
          type="text"
          value={name}
          onChange={(e) => setName(e.target.value)}
        />
      </div>
      <div>
        <label>E-Mail:</label>
        <input
          type="email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
        />
      </div>
      <div>
        <label>Passwort:</label>
```

```
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
          />
        </div>
        <div>
          <label>Passwort bestätigen:</label>
          <input
            type="password"
            value={confirmPassword}
            onChange={(e) => setConfirmPassword(e.target.value)}
          />
        </div>
        <button type="submit">Registrieren</button>
      </form>
    </div>
  );
}

export default RegistrationForm;
```

### 2.3 1.3 Erklärung

- **State-Verwaltung:** Wir verwenden `useState`, um den Zustand der Eingabefelder (Name, E-Mail, Passwort, Bestätigung) sowie Fehler- und Erfolgsmeldungen zu speichern.
- **Formularabsendung:** Die Funktion `handleSubmit` verhindert das Standardverhalten des Formulars und führt Validierungen durch, bevor sie eine POST-Anfrage an unser Backend sendet.
- **API-Anfrage:** Mithilfe von `fetch` senden wir die Nutzerdaten im JSON-Format an `http://localhost:5000/api/register`.

## 3 Schritt 2: Erstellen des Login-Formulars

### 3.1 2.1 Komponente anlegen

1. Erstellen Sie in `src` eine neue Datei namens `LoginForm.js`.

### 3.2 2.2 Grundgerüst der Komponente schreiben

Öffnen Sie `LoginForm.js` und fügen Sie folgenden Code ein:

```
import React, { useState } from 'react';

function LoginForm() {
  const [email, setEmail] = useState('');
```

```
const [password, setPassword] = useState('');
const [error, setError] = useState('');
const [success, setSuccess] = useState('');

const handleLogin = async (event) => {
  event.preventDefault();
  if (!email || !password) {
    setError('Bitte E-Mail und Passwort eingeben.');
    return;
  }
  const loginData = { email, password };

  try {
    const response = await fetch('http://localhost:5000/api/login', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(loginData),
    });
    const result = await response.json();
    if (response.ok) {
      setSuccess(result.message);
      setError('');
      // Token kann hier in localStorage gespeichert werden, z.B.:
      localStorage.setItem('token', result.token);
    } else {
      setError(result.error || 'Login fehlgeschlagen.');
    }
  } catch (error) {
    console.error('Loginfehler:', error);
    setError('Ein Fehler ist aufgetreten.');
  }
};

return (
  <div>
    <h2>Login</h2>
    {error && <p style={{color: 'red'}}>{error}</p>}
    {success && <p style={{color: 'green'}}>{success}</p>}
    <form onSubmit={handleLogin}>
      <div>
        <label>E-Mail:</label>
        <input
          type="email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
        />
      </div>
      <div>
```

```
          <label>Passwort:</label>
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
          />
        </div>
        <button type="submit">Login</button>
      </form>
    </div>
  );
}

export default LoginForm;
```

### 3.3 2.3 Erklärung

- **State-Verwaltung:** Die Komponente verwaltet den Zustand für E-Mail, Passwort, Fehler- und Erfolgsmeldungen.
- **API-Anfrage:** Beim Absenden des Formulars wird eine POST-Anfrage an den Login-Endpunkt `/api/login` gesendet. Erfolgt der Login, wird das zurückgegebene Token (falls vorhanden) in `localStorage` gespeichert.
- **Validierung:** Es wird überprüft, ob beide Felder ausgefüllt sind, bevor die Anfrage gesendet wird.

## 4 Schritt 3: Integration der Formulare ins Frontend

### 4.1 3.1 Komponenten importieren und verwenden

1. Öffnen Sie Ihre Hauptdatei (z.B. `src/App.js`).
2. Importieren Sie die beiden neuen Komponenten:

   ```
   import RegistrationForm from './RegistrationForm';
   import LoginForm from './LoginForm';
   ```

3. Fügen Sie die Komponenten in das Rendern der App ein, z. B.:

   ```
   function App() {
     return (
       <div className="App">
         <h1>Gym Progress Tracking System</h1>
         <RegistrationForm />
         <LoginForm />
       </div>
     );
   ```

```
}
```

### 4.2 3.2 Testen der Formulare

- Starten Sie Ihre App mit `npm start` im Frontend-Ordner.
- Überprüfen Sie, ob beide Formulare (Registrierung und Login) korrekt angezeigt werden.
- Testen Sie, ob die Eingaben korrekt verarbeitet werden (z.B. Fehleranzeige bei leeren Feldern).

## 5 Zusammenfassung

- **Registrierungsformular:** Erstellen Sie in `RegistrationForm.js` eine Komponente, die Felder für Name, E-Mail, Passwort und Passwort-Bestätigung enthält. Implementieren Sie eine Funktion, die beim Absenden des Formulars die Daten per POST an den Endpunkt `/api/register` sendet und einfache Validierungen vornimmt.
- **Login-Formular:** Erstellen Sie in `LoginForm.js` eine Komponente mit Feldern für E-Mail und Passwort. Implementieren Sie eine Funktion, die die Daten an den Endpunkt `/api/login` sendet und bei Erfolg das JWT-Token (z.B. in localStorage) speichert.
- **Integration:** Importieren und verwenden Sie beide Komponenten in Ihrer Haupt-App-Komponente (z. B. in `App.js`), um sie im Browser anzuzeigen.

Diese detaillierte Anleitung führt Sie Schritt für Schritt durch das Erstellen und Integrieren der Registrierungs- und Login-Formulare in Ihr React-Projekt, sodass Sie später die API-Anfragen an Ihr Backend implementieren und testen können.