001-Nutzerverwaltung

Frontend Aufgabe

Anforderungsniveau: Leicht

1 Ziel der Aufgabe

Ziel dieser Aufgabe ist es, Ihre Kenntnisse in moderner Frontend-Entwicklung mit Next.js und TypeScript zu überprüfen. Sie erstellen eine kleine Anwendung, die Daten von einer externen API abruft und darstellt. Dabei sollen folgende Techniken demonstriert werden:

- Nutzung von `getStaticProps` oder `getServerSideProps`
- Dynamische Routen (`pages/users/[id].tsx`)
- Typisierung aller Komponenten mit TypeScript
- Einfache Fehler- und Ladezustände

2 Aufgabenbeschreibung

1. **Projektinitialisierung**
   - Erstellen Sie ein neues Next.js–Projekt mit TypeScript:

     ```
     npx create-next-app@latest my-next-ts-app --typescript
     ```

   - Legen Sie eine Ordnerstruktur an:

     ```
     my-next-ts-app/
      components/
      pages/
         index.tsx
         users/
              [id].tsx
      types/
      public/
     ```

2. **Datentypen definieren**
   - Erstellen Sie in `types/User.ts` ein Interface `User`:

     ```
     export interface User {
       id: number;
       name: string;
       email: string;
       address: {
         street: string;
         city: string;
         zipcode: string;
       };
     ```

```
  company: {
    name: string;
  };
}
```

3. **Startseite: Benutzerliste**
   - In `pages/index.tsx` nutzen Sie `getStaticProps`, um alle Benutzer von `https://jsonplaceh` zu laden:

```
export const getStaticProps = async () => {
  const res = await fetch('https://jsonplaceholder.typicode.com/users');
  const users: User[] = await res.json();
  return { props: { users } };
};
```

   - Stellen Sie die Liste dar, jeweils mit Name, E-Mail und einem Link zu `/users/[id]`.

4. **Detailseite: Ein Benutzer**
   - Erstellen Sie `pages/users/[id].tsx` als dynamische Route:

```
export const getStaticPaths = async () => {
  const res = await fetch('https://jsonplaceholder.typicode.com/users');
  const users: User[] = await res.json();
  const paths = users.map(u => ({ params: { id: u.id.toString() } }));
  return { paths, fallback: false };
};

export const getStaticProps = async ({ params }) => {
  const res = await fetch(
    `https://jsonplaceholder.typicode.com/users/${params.id}`
  );
  const user: User = await res.json();
  return { props: { user } };
};
```

   - Zeigen Sie Name, E-Mail, Adresse (Straße, Stadt, PLZ) und Firmenname an.
   - Falls der Benutzer nicht existiert, Next.js erzeugt automatisch eine 404-Seite (bei `fallback: false`).

5. **Komponenten und Styling**
   - Legen Sie zwei einfache Komponenten an:
     - `components/UserCard.tsx` für Anzeige in der Liste
     - `components/UserDetail.tsx` für Detailseite
   - Beispielhaft `UserCard.tsx`:

```
import Link from 'next/link';
import { User } from '../types/User';

interface Props { user: User; }
```

```
export default function UserCard({ user }: Props) {
  return (
    <div>
      <h3>{user.name}</h3>
      <p>{user.email}</p>
      <Link href={`/users/${user.id}`}>Details</Link>
    </div>
  );
}
```

- Verwenden Sie eine minimale CSS-Lösung wie Tailwind CSS oder CSS-Module für grundlegendes Layout.

6. **Meta-Daten und Navigation**
   - Fügen Sie in jeder Seite ein `<Head>` hinzu:

     ```
     import Head from 'next/head';

     <Head>
       <title>Benutzerliste</title>
       <meta name="description" content="Eine Liste von Benutzern anzeigen" />
     </Head>
     ```

   - Implementieren Sie eine einfache Navigation, z. B. ein Header mit Link zu „Home“:

     ```
     <nav>
       <Link href="/">Home</Link>
     </nav>
     ```

3 Anforderungen

- **TypeScript-Typisierung**: Alle Komponenten und Funktionen müssen vollständig typisiert sein. Kein Einsatz von `any`.
- **Fehler- und Ladezustände**: Zeigen Sie während des Ladens einen Spinner oder „Loading...“ an. Bei Fetch-Fehlern eine einfache Fehlermeldung.
- **Saubere Ordnerstruktur**: Trennen Sie `components/`, `pages/`, `types/` und `styles/` klar.
- **Dokumentation**: Fügen Sie eine kurze `README.md` bei mit:
  - Installationsschritte: `npm install`
  - Starten im Development-Modus: `npm run dev`
  - Bauen der Produktion: `npm run build` und `npm start`

Viel Erfolg bei der Umsetzung!
Stand: 31. Mai 2025