**Primii pași React + TypeScript pentru ecoMatch — Ce trebuie să facem**

**1. Structurarea proiectului frontend**

* Vom crea componente clare, de exemplu:
  + **Login** (formular autentificare)
  + **Register** (formular înregistrare, dacă e cazul)
  + **Dashboard** (pagina principală după login)
  + **RequestForm** (formularul pentru clienți să ceară servicii)
  + **ProviderList** (lista furnizorilor potriviți)

**2. Rularea aplicației + integrarea cu backend**

* La început vom face un formular simplu de login care să trimită datele la backend.
* Backendul trebuie să accepte autentificarea (token JWT deja făcut de tine).
* React va stoca tokenul și îl va folosi pentru cererile ulterioare.

**3. Ce vom face imediat acum?**

* Creăm o componentă Login.tsx cu un formular simplu: email + parola.
* Facem un **submit** care trimite datele la API-ul backend (/api/token/).
* Primim tokenul JWT înapoi și îl stocăm (temporar în localStorage).
* Pe baza autentificării, vom naviga către un dashboard (sau afișăm mesaj).

**4. Configurare de bază a router-ului React**

* Vom folosi react-router-dom pentru a naviga între pagini (login, dashboard etc.)

**24.06.2025**

**Ce ai făcut până acum**

**Backend (Django REST API)**

* **Ai creat un backend Django** care oferă API REST folosind Django REST Framework.
* Ai definit un **model de utilizator personalizat** (core.User).
* Ai configurat autentificarea bazată pe **JWT (JSON Web Tokens)**, folosind djangorestframework-simplejwt.
* Ai configurat CORS în settings.py cu django-cors-headers, pentru a permite frontend-ului React să comunice cu backend-ul pe domenii diferite (localhost:3000 pentru frontend, localhost:8000 pentru backend).
* Ai pornit serverul Django local pe portul 8000 și ai verificat că endpoint-ul de autentificare (/api/token/) funcționează corect.

**Frontend (React + TypeScript)**

* Ai creat un proiect React folosind TypeScript, în folderul frontend.
* Ai instalat librăria axios pentru a face cereri HTTP către backend.
* Ai creat un fișier auth.ts în src/api care conține funcția login ce trimite cererea de autentificare către backend și primește token-urile JWT.
* Ai creat un component Login.tsx în React, care:
  + Oferă un formular cu input pentru username și parolă.
  + Apelează funcția login din auth.ts.
  + Arată mesaje de eroare dacă login-ul eșuează.
  + Arată un alert cu token-ul când login-ul reușește.
* Ai configurat CORS corect pe backend pentru a nu primi erori de tipul CORS policy.

**Tehnologii folosite**

* **Backend:**
  + Python 3.x
  + Django 5.x
  + Django REST Framework
  + djangorestframework-simplejwt (JWT Auth)
  + django-cors-headers (pentru CORS)
  + SQLite (bază de date implicită)
* **Frontend:**
  + React 18.x
  + TypeScript
  + axios (pentru HTTP requests)
  + npm (node package manager)

**Ce mai avem de făcut**

**Frontend**

1. **Stocarea token-ului JWT:**
   * Salvarea token-ului în localStorage sau sessionStorage pentru a-l folosi ulterior.
   * Gestionarea stării autentificării (ex: context React sau librărie de management state).
2. **Gestionarea accesului la pagini protejate:**
   * Implementarea rutelor protejate (folosind react-router-dom).
   * Redirect către login dacă user-ul nu este autentificat.
3. **Trimiterea token-ului în header-ul Authorization pentru cereri API protejate:**
   * Configurarea unui interceptor axios care adaugă automat token-ul în cereri.
4. **Crearea de componente și pagini pentru funcționalitățile proiectului ecoMatch:**
   * Formular cerere servicii.
   * Listă prestatori / clienți.
   * Dashboard utilizator.
   * Alte funcționalități specifice.

**Backend**

1. **Extinderea API-ului:**
   * Crearea endpointurilor pentru cererile și profilurile utilizatorilor (ex. Requests, Providers).
   * Controlul accesului pe endpointuri (permisiuni, roluri).
2. **Validări, securitate, logging, rate limiting etc.**
3. **Deploy:**
   * Pregătirea pentru producție, eventual cu server WSGI (gunicorn, uWSGI), HTTPS etc.