✅ **Was ich bereits gemacht habe:**

🔧 **1. Django-Projekt Setup:**

* Django + Django REST Framework installiert ✔
* Projekt und App core erstellt ✔
* AUTH\_USER\_MODEL angepasst und konfiguriert ✔
* settings.py korrekt eingerichtet ✔
* Lokaler Server gestartet ✔

🗃️ **2. Datenmodelle (models.py):**

* Benutzerdefiniertes User-Modell ✔
* ServiceType-Modell ✔
* ProviderProfile-Modell ✔
* Request-Modell ✔

🛠️ **3. Migrationen:**

* Migrationen erstellt und angewendet ✔

🔐 **4. Authentifizierung:**

* Superuser erstellt und in der Shell getestet ✔

🧪 **5. Lokale Tests:**

* Server ausgeführt, Admin-Bereich aufgerufen ✔

🌐 **6. API:**

* RequestViewSet erstellt ✔
* Router in core/urls.py konfiguriert ✔
* In backend/urls.py eingebunden ✔

🔜 **Was noch fehlt:**

🔹 **7. API komplettieren:**

* ViewSets für User, ServiceType, ProviderProfile
* Serializers für alle Modelle
* Authentifizierung mit Token oder JWT
* Individuelle Validierungen (Region, Status etc.)

🔹 **8. Erweiterte Funktionen:**

* Datei-Uploads (optional)
* Business-Logik (z. B. automatisches Matching)
* Filter (z. B. nach Region oder Status)
* Berechtigungen: Wer darf was sehen oder senden

🔹 **9. Abschluss:**

* API-Dokumentation mit Swagger/OpenAPI
* (Optional) Automatisierte Tests

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Software, Zahl enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Software, Zahl enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

23.06.25

**✅ Ce am realizat deja:**

1. **Setat proiect Django + Django REST Framework**
2. **Creat aplicația core**
3. **Definit modelele: User, ServiceType, ProviderProfile, Request**
4. **Setat autentificare JWT**
5. **Creat serializers.py și views.py pentru toate modelele**
6. **Configurat urls.py pentru API și JWT**
7. **Testat local cu succes (admin + endpointuri)**

**🔜 Următorii pași (backend):**

**📌 8. Validare și logică specială în ViewSet-uri:**

* În RequestViewSet → perform\_create să seteze automat userul autentificat ca client
* Verificare ca doar useri cu is\_client=True să creeze cereri
* Eventual: ProviderProfile poate fi creat doar de is\_provider

**📌 9. Permisiuni și acces:**

* Fiecare client vede doar cererile sale (RequestViewSet → get\_queryset)
* Doar providerii văd cererile disponibile
* Adminii văd tot (eventual)

**📌 10. Testare endpointuri cu JWT:**

* Obține token
* Accesează endpointuri cu Authorization: Bearer <token>
* Testează GET/POST/PUT/DELETE pentru fiecare ViewSet

**📌 11. Upload fișiere (opțional):**

* Poate la ProviderProfile (ex: CV, exemple proiecte)

**📌 12. Swagger / Documentație:**

* Integrare cu drf-yasg sau drf-spectacular pentru OpenAPI

**💡 După backend:**

* ➡️ Trecem la **frontend** (Angular, dacă dorești)
* Login / Signup
* Formulare pentru clienți
* Pagini profil furnizori
* Match automat între cereri și furnizori

“”Ce urmează, pas cu pas:

**7. Completarea API-ului**

* Adăugarea ViewSet-urilor lipsă (User, ServiceType, ProviderProfile) dacă nu sunt gata complet
* Validări suplimentare la nivel de serializator (ex: validare câmpuri, relații)
* Permisiuni personalizate: cine poate vedea/edita ce (ex: doar userul să-și vadă cererile, sau doar providerii pot vedea anumite date)

**8. Funcții avansate**

* Upload de fișiere (dacă e cazul)
* Logica de business, ex: matching automat între cereri și furnizori
* Filtre pe rute (search, filter după regiune, status, etc)
* Testare automată cu Django TestCase / pytest

**9. Documentație API**

* Swagger/OpenAPI pentru documentare ușoară și testare interactivă