**🔐 auth Modülü – Şimdiye Kadar Yaptıklarımız**

**1️⃣ RegisterSerializer**

📁 auth/serializers/register\_serializer.py

**Amacı:**  
Kullanıcıdan gelen verileri (JSON) doğrulayıp, User modeline uygun hale getirir.

**Ne işe yarar?**

* username, email, password alanlarını kontrol eder
* password’ü gizli tutar (write\_only)
* Hatalı girişleri (eksik alanlar, format hatası) ayıklar

**Neden önemli?**

* Django REST Framework’te **veri doğrulama (validation)** işini serializer yapar.
* Hem güvenli hem kontrollü veri girişini sağlar.

**2️⃣ UserService**

📁 auth/services/user\_service.py

**Amacı:**  
Gerçek kullanıcı oluşturma işini tek bir yerde toplar (model erişimi).

**Ne işe yarar?**

* User.objects.create\_user() çağrısı sadece burada yapılır
* View bu işten soyutlanmış olur (SOLID’in S ve D’si)

**Neden önemli?**

* Yarın kullanıcı oluştururken farklı bir işlem yapılması gerekirse sadece burayı değiştirirsin (email gönderimi, audit log vs.)

**3️⃣ RegisterView**

📁 auth/views/register\_view.py

**Amacı:**  
API isteğini alır → veriyi kontrol eder → kullanıcıyı oluşturur → token döner

**Ne işe yarar?**

* POST /api/auth/register/ endpoint'ini yönetir
* İstemciden gelen username, email, password verilerini işler
* Başarılıysa: Token üretip geri döner
* Hatalıysa: Hata mesajlarını JSON olarak döner

**Neden önemli?**

* API’nin dış dünyaya açılan kapısı burası
* View, serializer + service katmanına bağlı ama onlara iş yaptırıyor (kendi doğrudan işlem yapmaz)

**4️⃣ URL Tanımları**

📁 auth/urls.py ve msc\_core/urls.py

**Amacı:**  
Dış dünyadan gelen URL’leri doğru view’lere yönlendirir.

**Ne işe yarar?**

* /api/auth/register/ → RegisterView'a gider
* /api/auth/login/ → DRF’in hazır token endpoint’ine gider

**Neden önemli?**

* Burası API Gateway’in ilk giriş noktası gibi
* Her servisin erişilebilirliğini burada kontrol ediyoruz

**🔑 Token Authentication**

python

KopyalaDüzenle

REST\_FRAMEWORK = {

'DEFAULT\_AUTHENTICATION\_CLASSES': [

'rest\_framework.authentication.TokenAuthentication',

],

}

**Ne işe yarar?**

* Kullanıcı login olduğunda bir token alır
* Bu token’ı sonraki API isteklerinde Authorization: Token abc123 olarak gönderirse sistem onu tanır

**🔄 Süreç Akışı (Kullanıcı Kaydı)**

swift

KopyalaDüzenle

İstemci → POST /api/auth/register/

{

"username": "mehmet",

"email": "mehmet@example.com",

"password": "gizli123"

}

→ RegisterView → RegisterSerializer (doğrula) → UserService (kullanıcı oluştur) → Token üret → Token’ı JSON ile geri döner

**📌 Neden SOLID?**

| **Prensip** | **Uygulama** |
| --- | --- |
| S – Single Responsibility | View sadece yönlendiriyor, UserService iş yapıyor |
| D – Dependency Inversion | View doğrudan modelle konuşmuyor, araya servis koyduk |
| O – Open/Closed | Yeni kimlik doğrulama türü eklenebilir, mevcut kodu bozmadan |