

# Django İle Python Arka Yüz Yazılım Geliştirme

Teknik Bilimler  
Meslek Yüksekokulu

Öğr. Gör. Furkan DURMUŞ



# Bu Haftanın Ders Kazanımları



**Admin Panel**

**Yazar Detay Sayfası**

**ORM Sorguları**

**İletişim Sayfası**

**Yazarlar Sayfası**

# Furkandurmus.com

-> Dersler

-> Django

-> Ders Uygulamaları

-> Blog Proje Kodları

Üzerinden projenizi güncelleyebilirsiniz...





```
require 'capybara/re Rails'
```

```
1 Capybara.javascript_driver = :webkit
```

```
2 Category.delete_all; Category.create
```

```
3 Shoulda::Matchers.configure do |config|
```

```
4   config.integrate do |with|
```

```
5     with.test_framework :rspec
```

```
6     with.library :rails
```

```
7   end
```

```
8 end
```

```
9 # Add additional requires below this line. You may
```

```
10 # Requires supporting ruby files with support/
```

```
11 # spec/support/ and its subdirectories. These
```

```
12 # run as spec files by default. This will
```

```
13 # in _spec.rb will both be required and
```

```
14 # It is recommended that you configure
```

# Admin Panel



# Admin Paneli Nedir ?



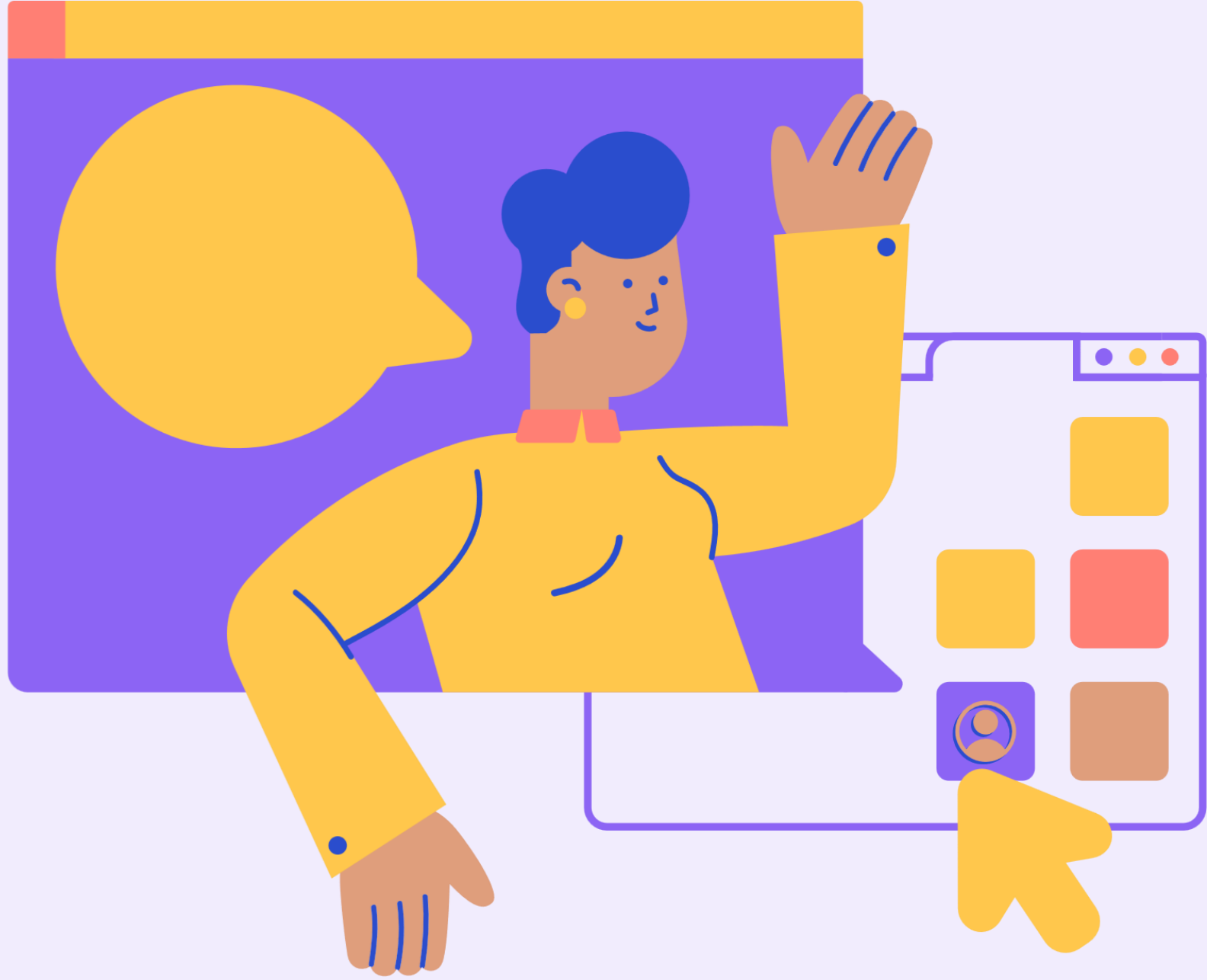
# Admin Paneli Nedir ?

Django içinde admin paneli, uygulamadaki verileri yönetmek için hazır gelen bir yönetim arayüzüdür.

Yani geliştirici veya site yöneticisi:

- veri ekleyebilir,
- silebilir,
- güncelleyebilir,
- kullanıcıları yönetebilir,
- modelleri kontrol edebilir.

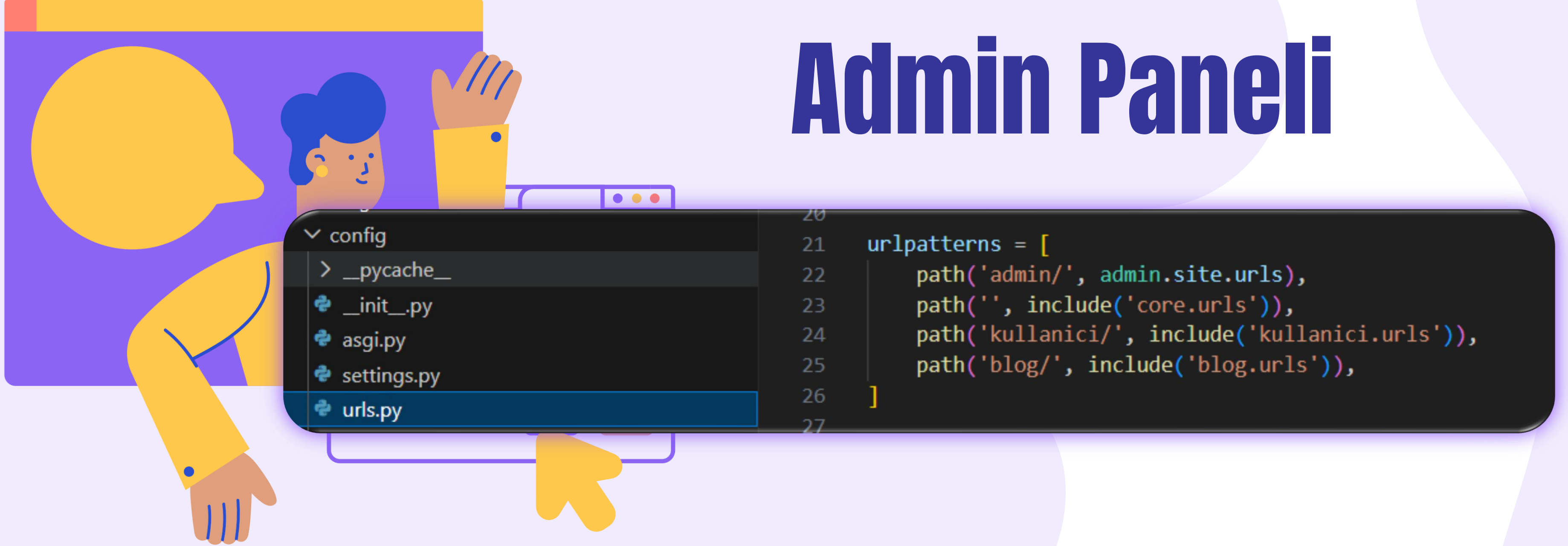




# Admin Paneli

Admin panelinin URL'ini projenizde bulabilir misiniz ?

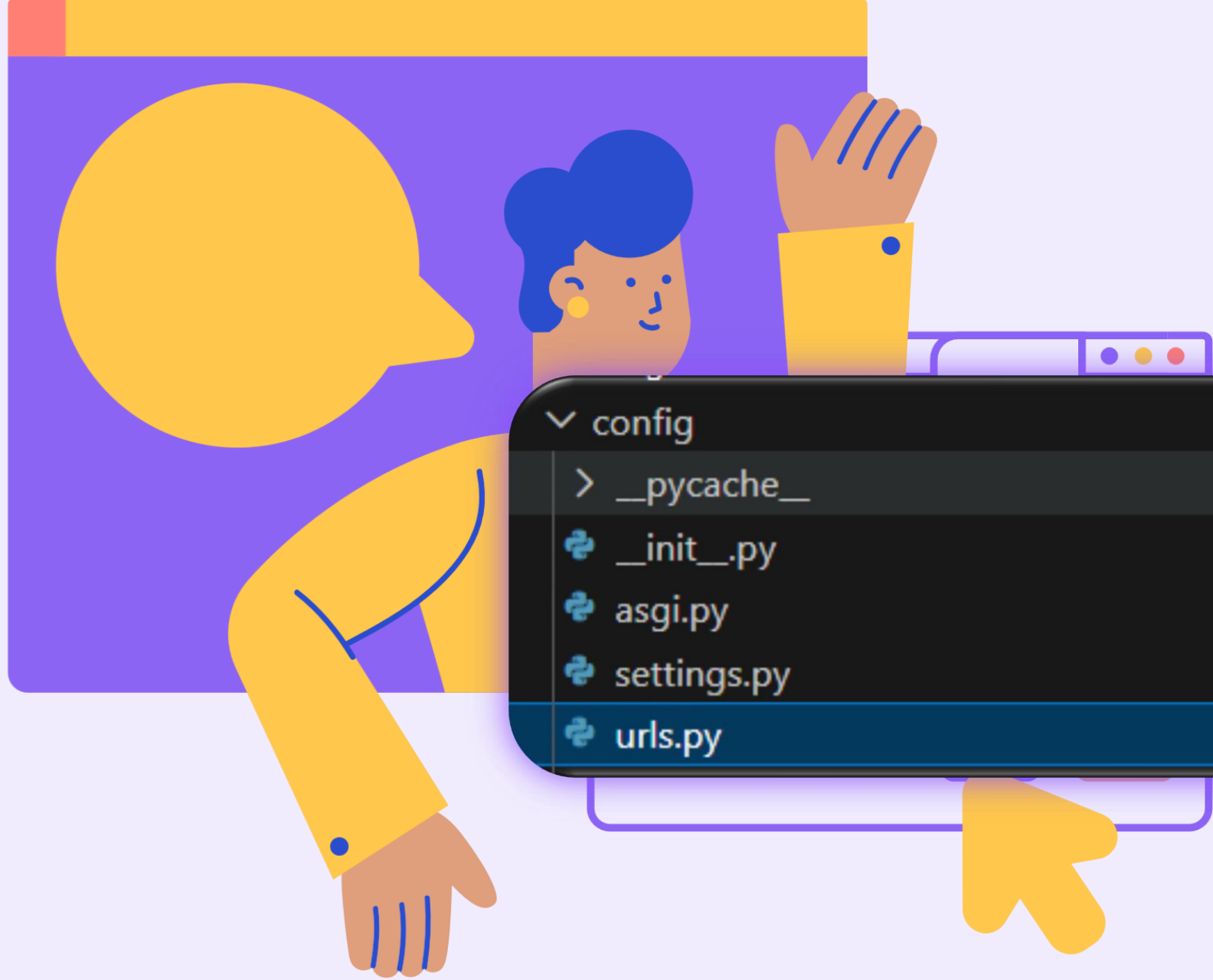
# Admin Paneli



```
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', include('core.urls')),
24     path('kullanici/', include('kullanici.urls')),
25     path('blog/', include('blog.urls')),
26 ]
27
```

Django'nun kurulumunda admin ile ilgili tüm yapı otomatik olarak kurulur.

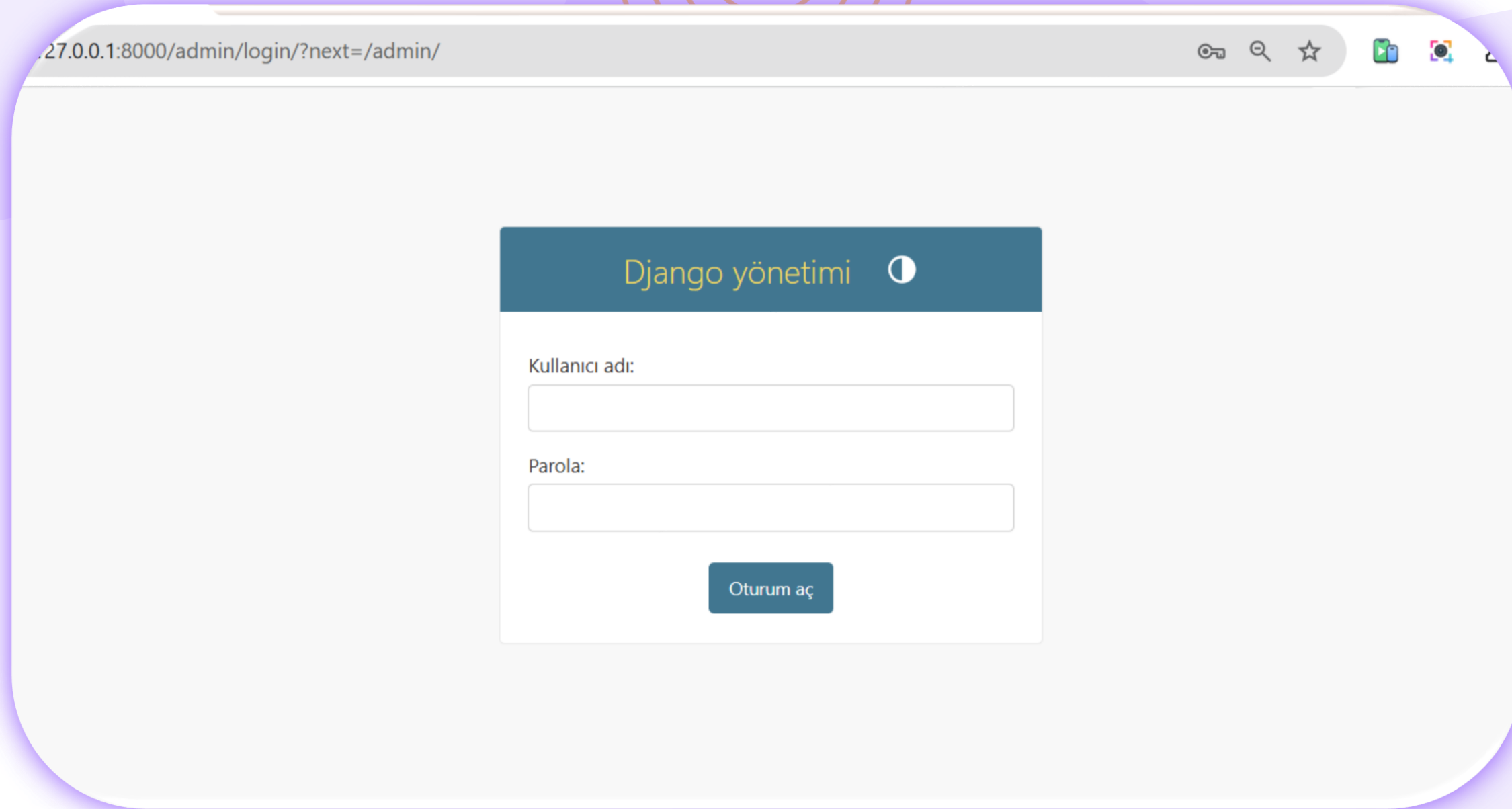
# Admin Paneli



```
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', include('core.urls')),
24     path('kullanici/', include('kullanici.urls')),
25     path('blog/', include('blog.urls')),
26 ]
27
```

Projeyi çalıştırıp /admin sayfasına gidiniz.

# Admin Paneli



127.0.0.1:8000/admin/login/?next=/admin/

Django yönetimi

Kullanıcı adı:

Parola:

Oturum aç

Bu alana giriş yapabilmek için terminal tarafında süper kullanıcı oluşturmalıyız.

# Admin Paneli

- 1- Projeyi durdurun.
- 2- Terminale **'python manage.py createsuperuser'** yazın.
- 3- 'Kullanıcı Adı : ' kısmına istediğinizi yazabilirsiniz. Tercihen **admin** yazılır.
- 4- 'E-posta Adresi ' kısmını **boş** bırakabilirsiniz.
- 5- Şifre girdiğinizde ekranda güvenlik sebebiyle gözükmez! Ama arka planda yazar.
- 6- 'Password' kısmını istediğiniz şifre yapabilirsiniz. Şimdilik **1234** yazabilirsiniz.
- 7- Şifreyi tekrar giriniz.

# Admin Paneli

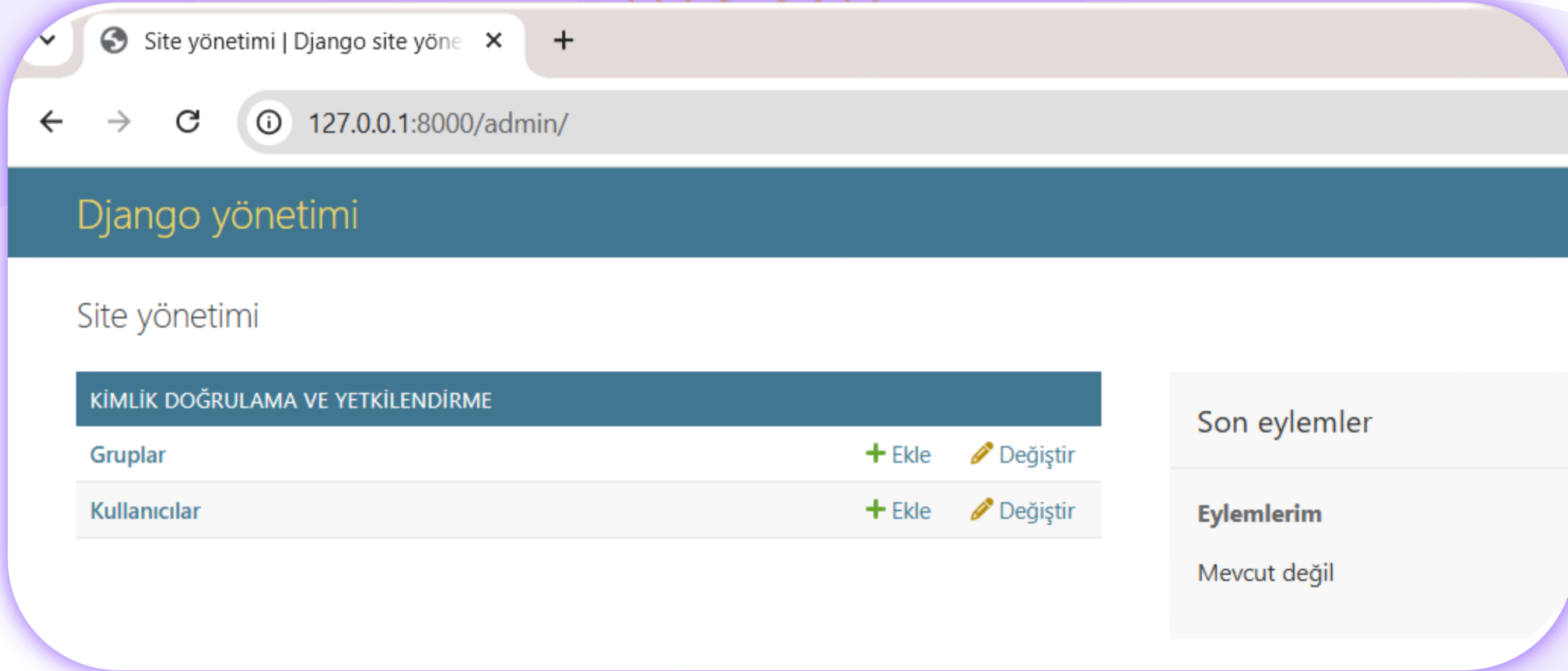
```
Bu parola çok kısa. En az 8 karakter içermek zorunda.  
Bu parola çok geneldir.  
Bu parola tamamıyla sayısaldır.  
Bypass password validation and create user anyway? [y/N]:
```

- 8- Şifreden dolayı uyarı vermektedir. **'y'** yazıp enter'a basarak bypass edebilirsiniz.
- 9- 

```
Superuser created successfully.
```

 yazısını gördüyseniz başarılı bir şekilde süper user oluştu.
- 10- Projeyi tekrar başlatın.
- 11- Admin paneline oluşturduğunuz bilgilerle giriş yapabilirsiniz.

# Admin Paneli



Tablolarımız yok, bunun sebebi admin paneline bu modelleri register etmedik.

# Admin Paneli




```
✓ BLOG
  ✓ blog
    > __pycache__
    > migrations
    > templates
    🌀 __init__.py
    🌀 admin.py
    🌀 ...

blog > 🌀 admin.py
1   from django.contrib import admin
2
3   # Register your models here.
4
```

Blog app'i altındaki admin.py dosyasını açınız.



# Admin Paneli

```
blog >  admin.py
1   from django.contrib import admin
2   from .models import BlogPost
3
4   admin.site.register(BlogPost)
5
6
```

Blog/admin.py'yi üstteki gibi güncelleyiniz, ardından projeyi başlatarak admin paneline giriniz.

# Admin Paneli

APP adı  
Tablo adı

Site yönetimi | Django site yöne x +

127.0.0.1:8000/admin/

Django yönetimi

Site yönetimi

BLOG

Blog posts + Ekle Değiştir

KİMLİK DOĞRULAMA VE YETKİLENDİRME

Gruplar + Ekle Değiştir

Kullanıcılar + Ekle Değiştir

Son eylemler

Eylemlerim

Mevcut değil

Ekle & değiştir butonları ile projenize yeni post ekleyiniz ve mevcutların content kısmını daha uzun olacak şekilde güncelleyiniz. Toplamda 8 tane kayda değer blog ekleyiniz. <sup>16</sup>

# Admin Paneli

Default kısımların otomatik doldurulduğunu görebilirsiniz.

blog post ekle

**Title:** Siber Güvenlik ve Veri Koruma

**Summary:** Siber güvenlik ve veri koruma, dijital sistemler

**Content:**  
Kayıpların maddi ve manevi zararlar oluşturabilir. Bu nedenle şirketler güvenlik duvarları, antivirüs sistemleri ve düzenli yedekleme çözümleri kullanarak sistemlerini korumaya çalışmaktadır.  
Ayrıca HTTPS, şifreleme teknolojileri ve güvenli sunucu altyapıları internet üzerindeki veri iletişimini daha güvenli hale getirmektedir. Kullanıcıların bilinçli internet kullanımı da siber güvenlik açısından oldukça önemlidir. Bilinmeyen bağlantılara tıklamamak, şüpheli dosyaları indirmemek ve kişisel bilgileri güvenilmeyen platformlarda paylaşmamak temel güvenlik kuralları arasında yer alır.  
Gelecekte yapay zekâ ve bulut bilişim teknolojilerinin yaygınlaşmasıyla birlikte siber güvenlik alanının daha da önemli hale gelmesi beklenmektedir. Bu nedenle hem bireylerin hem de kurumların dijital güvenlik konusunda bilinçli hareket etmesi büyük bir gereklilik olarak görülmektedir.

**Author name:** Furkan DURMUŞ

**Views:** 0

Is published

**Static url:** img/default.png

# Admin Paneli

Admin panelinde eklenen blogların  
title kısımlarına göre listeleme  
yapacağına admin paneli nasıl karar  
verdi ?

- [Redis ile Veritabanı Yönetimi](#)
- [HTTP Protokolü ve Web Güvenliği](#)
- [HTML5 ve CSS3 ile Modern Web Tasarımı](#)
- [Node.js ile Sunucu Tarafı Geliştirme](#)
- [Veri Bilimi ve Python](#)
- [Angular ile Modern Web Geliştirme](#)
- [Django'ya Giriş](#)

8 blog posts

# Admin Paneli

```
from django.db import models

class BlogPost(models.Model):
    title = models.CharField(max_length=200)
    summary = models.CharField(max_length=300)
    content = models.TextField()
    author_name = models.CharField(max_length=100)
    views = models.IntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    is_published = models.BooleanField(default=True)

    static_url = models.CharField(max_length=200, default='img/default.png')

    def __str__(self):
        return self.title
```

- [Redis ile Veritabanı Yönetimi](#)
- [HTTP Protokolü ve Web Güvenliği](#)
- [HTML5 ve CSS3 ile Modern Web Tasarımı](#)
- [Node.js ile Sunucu Tarafı Geliştirme](#)
- [Veri Bilimi ve Python](#)
- [Angular ile Modern Web Geliştirme](#)
- [Django'ya Giriş](#)

8 blog posts

Def str self return self.title sayesinde bu model çekilmek istendiğinde içindeki objelerin ekranda temsilci olarak hangi alanı gösterileceğini belirlemiş oluyoruz.

# Admin Paneli

Blog Post tablosuna tıkladığınızda açılan sayfada Blogların yanındaki kutucukları seçtikten sonra 'Eylem' dropdown'undan silme işlemi yapılabilir.

Değiştirmek için blog post seçin

Eylem:   1 / 8 seçildi

- BLO
- [Siber Güvenlik ve Veri Koruma](#)
- [Redis ile Veritabanı Yönetimi](#)
- [HTTP Protokolü ve Web Güvenliği](#)
- [HTML5 ve CSS3 ile Modern Web Tasarımı](#)
- [Node.js ile Sunucu Tarafı Geliştirme](#)
- [Veri Bilimi ve Python](#)
- [Angular ile Modern Web Geliştirme](#)
- [Django'ya Giriş](#)

8 blog posts

# Admin Paneli

Admin anasayfasında sağ frame'de  
Son Eylemleri görebilirsiniz.

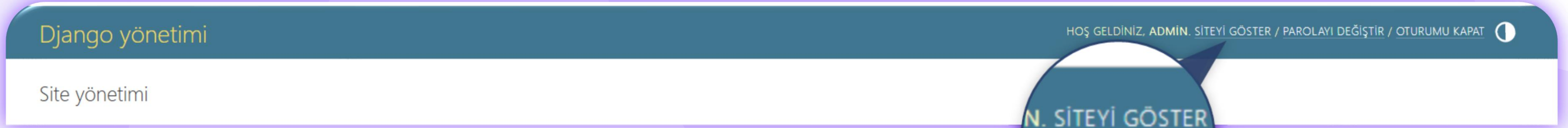
The screenshot displays an Admin Panel interface with the following sections:

- Site yönetimi** (Site Management)
- BLOG** section with a "Blog posts" row containing "+ Ekle" (Add) and "Değiştir" (Change) buttons.
- KİMLİK DOĞRULAMA VE YETKİLENDİRME** (Identity Verification and Authorization) section with "Gruplar" (Groups) and "Kullanıcılar" (Users) rows, each containing "+ Ekle" and "Değiştir" buttons.
- Son eylemler** (Recent Actions) section with the sub-heading "Eylemlerim" (My Actions) and a list of recent blog posts:

- + [Siber Güvenlik ve Veri Koruma](#) Blog post
- [Django'ya Giriş](#) Blog post
- [Angular ile Modern Web Geliştirme](#) Blog post
- [Veri Bilimi ve Python](#) Blog post
- [Node.js ile Sunucu Tarafı Geliştirme](#) Blog post
- [HTML5 ve CSS3 ile Modern Web Tasarımı](#) Blog post
- [HTTP Protokolü ve Web Güvenliği](#) Blog post
- [Redis ile Veritabanı Yönetimi](#) Blog post



# Admin Paneli



Admin Nawbar'ın en sağında 'Siteyi göster' e tıklayarak admin panelinden doğrudan web sitenizin ana sayfasına geçiş yapabilirsiniz.

# Admin Paneli

## Sonuç

The screenshot displays a blog website with a navigation bar at the top containing links for 'Ana Sayfa', 'İletişim', 'Hakkımızda', 'Yazarlar', 'Blog', and 'Kullanıcı'. A search bar is located on the right side of the navigation bar. The main content area is titled 'Son Yazılar' and features a grid of article cards. Each card includes a header image, a title, the author's name, the number of views, a short description, and a 'Yazının Devamı...' button. The articles are as follows:

- Django'ya Giriş** by Ahmet Faruk DURSUN (120 views). Description: Django framework'üne hızlı bir başlangıç. Button: Yazının Devamı...
- Angular ile Modern Web Geliştirme** by Tuncay ALTUN (95 views). Description: Angular kütüphanesi ile dinamik kullanıcı arayüzleri oluşturma. Button: Yazının Devamı...
- Veri Bilimi ve Python** by Murat KAYA (150 views). Description: Python'un veri bilimi alanındaki gücü. Button: Yazının Devamı...
- Node.js ile Sunucu Tarafı Geliştirme** by Ahmet Faruk DURSUN (100 views). Description: Node.js kullanarak scalable ve yüksek performanslı sunucu tarafı uygulamalar geliştirme. Button: Yazının Devamı...
- HTML5 ve CSS3 ile Modern Web Tasarımı** by Elif GÖKÇE (130 views). Description: Modern web tasarımı için HTML5 ve CSS3 teknikleri. Button: Yazının Devamı...
- HTTP Protokolü ve Web Güvenliği** by Ahmet Faruk DURSUN (85 views). Description: Web güvenliği ve HTTP protokolü hakkında bilgiler. Button: Yazının Devamı...
- Redis ile Veritabanı Yönetimi** by Mehmet ÖZTÜRK (110 views). Description: Redis kullanarak hızlı ve etkili veritabanı yönetimi. Button: Yazının Devamı...
- Siber Güvenlik ve Veri Koruma** by Furkan DURMUŞ (0 views). Description: Siber güvenlik ve veri koruma, dijital sistemleri ve kişisel bilgileri internet üzerindeki saldırılara karşı korumayı amaçlayan güvenlik yöntemlerinin bütünüdür. Button: Yazının Devamı...

At the bottom of the page, there are social media icons for Facebook, Twitter, Google+, Instagram, LinkedIn, and YouTube. A copyright notice at the very bottom reads '© 2026 Copyright'.



# ORM Sorgularları

```
require 'capybara/re Rails'
```

```
1 Capybara.javascript_driver = :webkit
```

```
2 Category.delete_all; Category.create
```

```
3 Shoulda::Matchers.configure do |config|
```

```
4   config.integrate do |with|
```

```
5     with.test_framework :rspec
```

```
6     with.library :rails
```

```
7   end
```

```
8 end
```

```
9 # Add additional requires below this line
```

```
10 # Requires supporting ruby files with support/ and its subdirectories. This will be
```

```
11 # spec/support/ and its subdirectories. This will be run as spec files by default. This
```

```
12 # in _spec.rb will both be required and run. It is recommended that you can configure
```

# ORM Sorguları

## ALL()

Tablodaki tüm kayıtları getirir.

```
SELECT * FROM blog_blogpost;
```

```
def anasayfa(request):  
    bloglar = blog.models.BlogPost.objects.all()  
  
    context = {  
        "bloglar": bloglar  
    }  
  
    return render(request, "anasayfa.html", context)
```

# ORM Sorguları

## GET()

Tek bir kayıt getirir.

```
BlogPost.objects.get(id=3)
```

## ÖNEMLİ!

Bulamazsa veya birden fazla bulursa **hata verir**.

```
def blog_detay(request, id):  
  
    try:  
        yazi = blog.models.BlogPost.objects.get(id=id)  
  
    except:  
        return render(request, "404.html", status=404)
```

# ORM Sorguları

## FILTER()

Şarta uyan tüm kayıtları getirir.

```
BlogPost.objects.filter(is_published=True)
```

### ÖNEMLİ!

Filter tek başına sql'deki **LIKE gibi çalışmaz.**

Filter **büyük küçük harf duyarlıdır.**

**Bulamazsa hata vermez.**

```
def anasayfa(request):  
  
    bloglar = blog.models.BlogPost.objects.filter(author_name="Ahmet Faruk DURSUN")  
  
    context = {  
        "bloglar": bloglar  
    }  
  
    return render(request, "anasayfa.html", context)
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

# ORM Sorguları

## FILTER(\_\_contains)

Şarta uyan tüm kayıtları %like% ile getirir.

`BlogPost.objects.filter(title__contains='veri')`

### ÖNEMLİ!

Filter `__contains` ile sql'deki **%LIKE%** gibi çalışır.

`__contains` büyük küçük harf duyarlıdır.

Fakat Db nizde bu duyarlılık yoksa büyük küçük bakmaksızın getirir.

```
def anasayfa(request):  
  
    bloglar = blog.models.BlogPost.objects.filter(title__contains="veri")  
  
    context = {  
        "bloglar": bloglar  
    }  
}
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

### Son Yazılar



python™

Veri Bilimi ve Python  
Murat KAYA - 150 görüntülenme  
Python'un veri bilimi alanındaki gücü.



redis

Redis ile Veritabanı Yönetimi  
Mehmet ÖZTÜRK - 110 görüntülenme  
Redis kullanarak hızlı ve etkili veritabanı yönetimi.



NEW  
BLOG POST

Siber Güvenlik ve Veri Koruma  
Furkan DURMUŞ - 0 görüntülenme

# ORM Sorguları

## FILTER(\_\_icontains)

Şarta uyan tüm kayıtları %like% ile getirir.

```
BlogPost.objects.filter(title__icontains='veri')
```

### ÖNEMLİ!

Filter \_\_contains ile sql'deki **%LIKE%** gibi çalışır.

\_\_contains **büyük küçük harf duyarlıdır.**

```
def anasayfa(request):  
  
    bloglar = blog.models.BlogPost.objects.filter(title__icontains="veri")  
  
    context = {  
        "bloglar": bloglar  
    }  
}
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

#### Son Yazılar



##### Veri Bilimi ve Python

Murat KAYA - 150 görüntülenme

Python'un veri bilimi alanındaki gücü.



##### Redis ile Veritabanı Yönetimi

Mehmet ÖZTÜRK - 110 görüntülenme

Redis kullanarak hızlı ve etkili veritabanı yönetimi.

NEW  
BLOG POST

##### Siber Güvenlik ve Veri Koruma

Furkan DURMUŞ - 0 görüntülenme

# ORM Sorguları

## FILTER(\_\_gt)

Sayısal olarak ..'dan büyük olanları getirir.

```
BlogPost.objects.filter(views__gt=100)
```

```
def anasayfa(request):  
  
    bloglar = blog.models.BlogPost.objects.filter(views__gt=50)  
  
    context = {  
        "bloglar": bloglar  
    }  
  
    return render(request, "anasayfa.html", context)
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

## ÖNEMLİ!

Gt açılımı **greater than**.

Eşit olanları getirmez, **eşit veya büyük** olanlar için **\_\_gte** kullanılmalı.

**Gte** açılımı **greater than or equal**'dir.

# ORM Sorguları

## FILTER(\_\_lt)

Sayısal olarak ..'dan küçük olanları getirir.

```
BlogPost.objects.filter(views__lt=100)
```

### ÖNEMLİ!

lt açılımı **less than**.

Eşit olanları getirmez, **eşit veya küçük** olanlar için **\_\_lte** kullanılmalı.

**lte** açılımı **less than or equal**'dir.

```
def anasayfa(request):  
    bloglar = blog.models.BlogPost.objects.filter(views__lt=50)  
  
    context = {  
        "bloglar": bloglar  
    }
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

# ORM Sorguları

## ORDER BY()

Db verilerinde sıralama yapar.

`BlogPost.objects.order_by('created_at')`

### ÖNEMLİ!

Eskiden yeniye, küçükten büyüğe sıralar.

```
def anasayfa(request):  
  
    bloglar = blog.models.BlogPost.objects.order_by("views")  
  
    context = {  
        "bloglar": bloglar  
    }  
  
    return render(request, "anasayfa.html", context)
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

# ORM Sorguları

## ORDER BY(-)

Db verilerinde sıralama yapar.

`BlogPost.objects.order_by('-created_at')`

### ÖNEMLİ!

Yeniden eskiye, büyükten küçüğe sıralar.

```
def anasayfa(request):  
    bloglar = blog.models.BlogPost.objects.order_by("-views")  
  
    context = {  
        "bloglar": bloglar  
    }  
  
    return render(request, "anasayfa.html", context)
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

# ORM Sorguları

## FILTER + ORDER BY

Filter ve order by birlikte kullanılabilir.

```
def anasayfa(request):  
    bloglar = blog.models.BlogPost.objects.filter(is_published=True).order_by("-created_at")  
  
    context = {  
        "bloglar": bloglar  
    }  
  
    return render(request, "anasayfa.html", context)
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

### ÖNEMLİ!

Önce Filter ardından order kullanılmalıdır.

# ORM Sorguları

## FILTER + ORDER BY

Çoklu filter ve order by birlikte kullanılabilir.

```
def anasayfa(request):  
  
    bloglar = blog.models.BlogPost.objects.filter(  
        is_published=True,  
        title__icontains="django",  
        views__gte=50  
    ).order_by('-views')  
  
    context = {  
        "bloglar": bloglar  
    }  
  
    return render(request, "anasayfa.html", context)
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

### ÖNEMLİ!

Önce Filter ardından order kullanılmalıdır.

# ORM Sorguları

## SLICE[ ]

DB den gelen veriyi [ ] listelerde yaptığımız gibi : veya :: işlemleri ile ihtiyacımıza göre bölebiliriz.

### ÖNEMLİ!

ORM den gelen veriyi düzenlememize yarar.  
Her zaman en sona yazılır.

```
def anasayfa(request):  
    bloglar = blog.models.BlogPost.objects.all()[:3]  
  
    context = {  
        "bloglar": bloglar  
    }  
  
    return render(request, "anasayfa.html", context)
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

# ORM Sorguları

Bu temel ORM sorgularının dışında ;

exclude() – first() – last() – exist() – values() – reverse() – distinct() - startswith – endswith – date filtreleri – range – isnull – aggregate() vs. gibi bir çok sorgu bulunmaktadır.

Bu sorguları da bireysel olarak inceleyebilirsiniz.



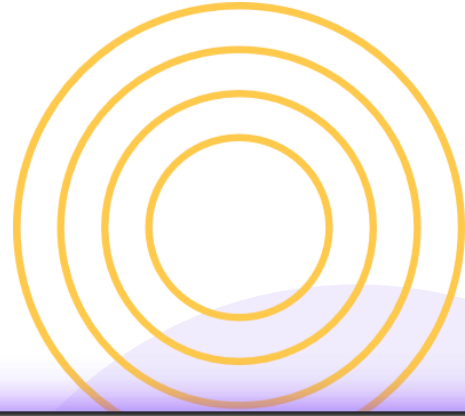
# ORM Sorguları



Öğrendiklerimiz doğrultusunda anasayfada en yeniden en eskiye son 6 blok yazısını yazdıracak şekilde anasayfa def'i güncelleyiniz.



# ORM Sorguları



```
def anasayfa(request):  
  
    bloglar = blog.models.BlogPost.objects.order_by('-created_at')[:6]  
  
    context = {  
        "bloglar": bloglar  
    }  
  
    return render(request, "anasayfa.html", context)
```





```
require 'capybara/re Rails'
```

```
1 Capybara.javascript_driver = :webkit
```

```
2 Category.delete_all; Category.create
```

```
13 Shoulda::Matchers.configure do |config|
```

```
14 config.integrate do |with|
```

```
15 with.test_framework :rspec
```

```
16 with.library :rails
```

```
17 end
```

```
18 end
```

```
19 # Add additional requires below this
```

```
20 # Requires supporting ruby files with support/
```

```
21 # spec/support/ and its subdirectories. These
```

```
22 # run as spec files by default. This will
```

```
23 # in _spec.rb will both be required and
```

```
24 # It is recommended that you configure
```

# Yazarlar Sayfası



# Yazarlar Sayfası



Yazarlar sayfasını db den beslenecek şekilde öğrendiklerimiz doğrultusunda düzenleyelim.





# Yazarlar Sayfası



Yazarlar sayfasını db den beslenecek şekilde öğrendiklerimiz doğrultusunda düzenleyelim.

1- Admin panelinden Authors modeline erişmek için doğru admin.py içerisine register yapılmalı.





# Yazarlar Sayfası



Yazarlar sayfasını db den beslenecek şekilde öğrendiklerimiz doğrultusunda düzenleyelim.

- 1- Admin panelinden Authors modeline erişmek için doğru admin.py içerisine register yapılmalı.
- 2- Yazarlar def'inde elle girilen veri silinmeli, db deki ilgili tablodaki tüm veri orm ile çekilmeli.



# Yazarlar Sayfası



Yazarlar sayfasını db den beslenecek şekilde öğrendiklerimiz doğrultusunda düzenleyelim.

- 1- Admin panelinden Authors modeline erişmek için doğru admin.py içerisine register yapılmalı.
- 2- Yazarlar def'inde elle girilen veri silinmeli, db deki ilgili tablodaki tüm veri orm ile çekilmeli.
- 3- Admin panelinden yazar eklenmeli. (profile url kısmına şimdilik <http://127.0.0.1:8000/yazarlar/yazınız>.) Toplamda 6 yazar ekleyiniz.





# Yazarlar Sayfası

1- Admin panelinden Authors modeline erişmek için core/admin.py içerisine register yapılmalı.

```
core > admin.py
1 from django.contrib import admin
2 from .models import Author
3
4 admin.site.register(Author)
5
6
```

# Yazarlar Sayfası

2- Yazarlar def'inde elle girilen veri silinmeli, db deki ilgili tablodaki tüm veri orm ile çekilmeli.

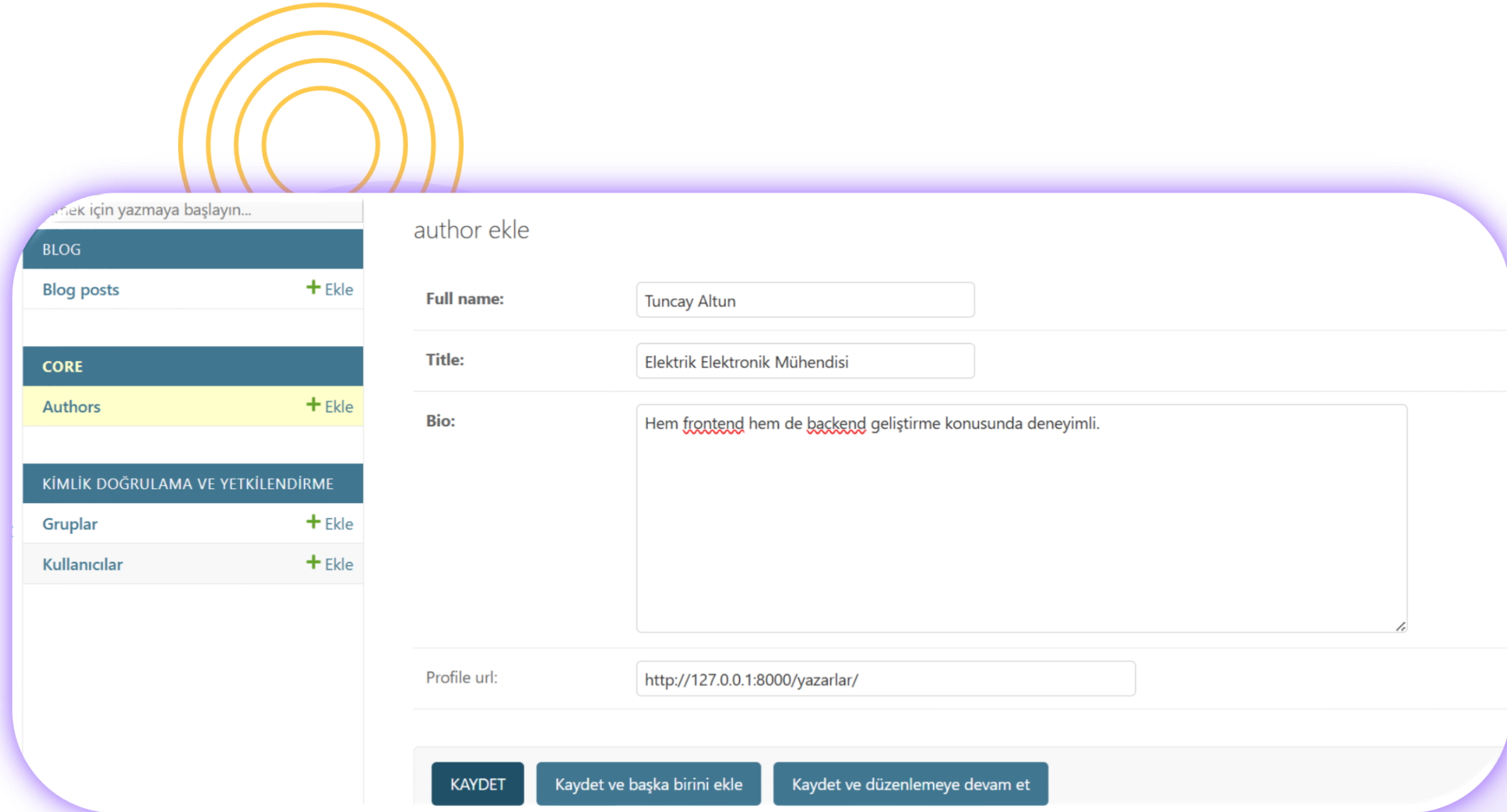
```
def yazarlar(request):  
    yazarlar = core.models.Author.objects.all()  
  
    context = {  
        "yazarlar": yazarlar,  
        "title": "Yazarlar",  
        "toplam_yazar": len(yazarlar),  
    }  
  
    return render(request, "yazarlar.html", context)
```

# Yazarlar Sayfası

3- Admin panelinden yazar eklenmeli.

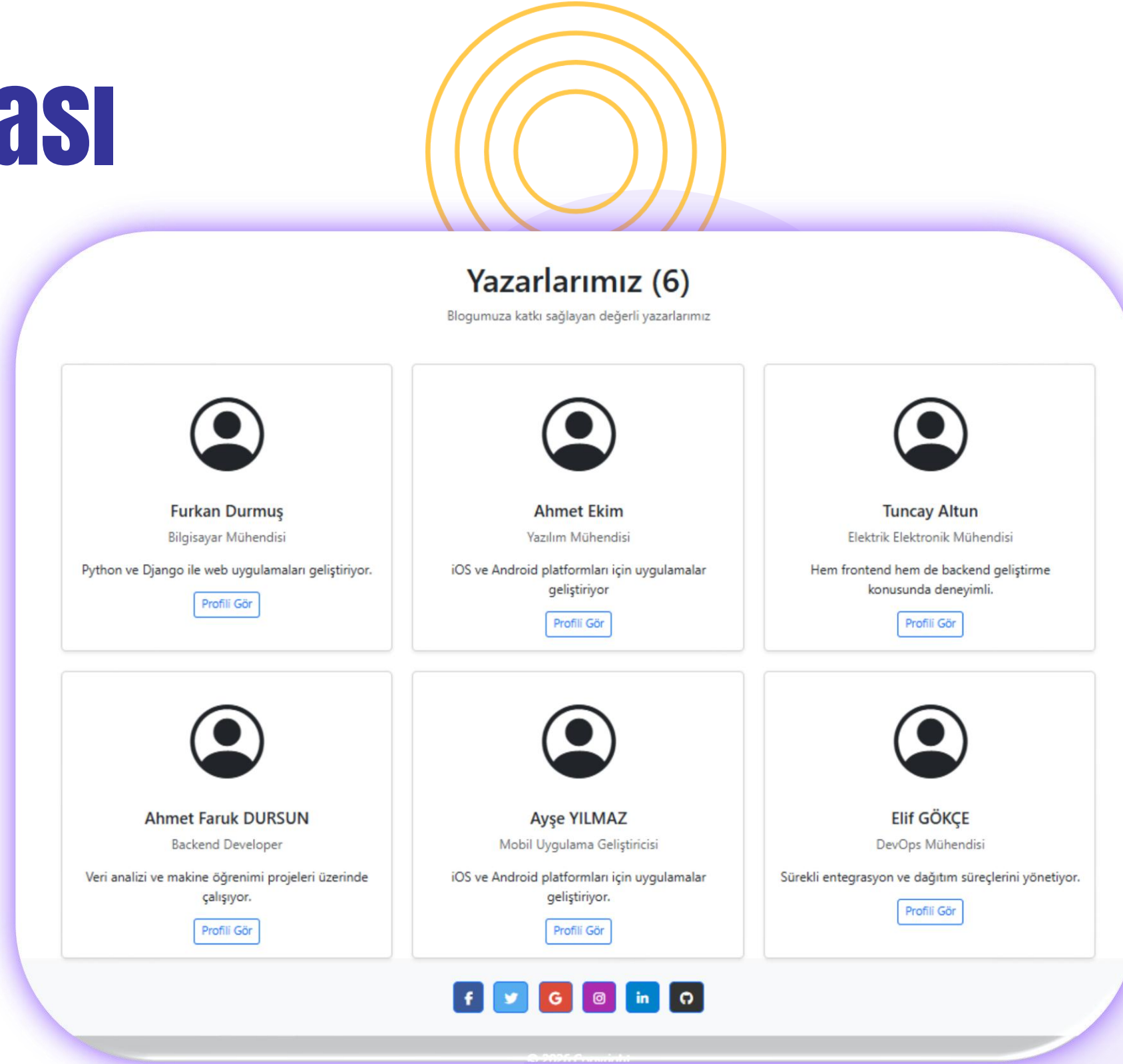
(profile url kısmına şimdilik <http://127.0.0.1:8000/yazarlar/> yazınız.)

Toplamda 6 yazar ekleyiniz.









The screenshot displays the admin panel interface. On the left is a sidebar menu with the following items: 'BLOG' (with 'Blog posts' and '+ Ekle'), 'CORE' (with 'Authors' and '+ Ekle'), 'KİMLİK DOĞRULAMA VE YETKİLENDİRME' (with 'Gruplar' and '+ Ekle', and 'Kullanıcılar' and '+ Ekle'). The main content area is titled 'author ekle' and contains the following form fields: 'Full name:' (Tuncay Altun), 'Title:' (Elektrik Elektronik Mühendisi), 'Bio:' (Hem frontend hem de backend geliştirme konusunda deneyimli.), and 'Profile url:' (http://127.0.0.1:8000/yazarlar/). At the bottom, there are three buttons: 'KAYDET', 'Kaydet ve başka birini ekle', and 'Kaydet ve düzenlemeye devam et'.

# Yazarlar Sayfası



**Yazarlarımız (6)**  
Blogumuza katkı sağlayan değerli yazarlarımız

|  |   |  |
|--|---|--|
|  <p><b>Furkan Durmuş</b><br/>Bilgisayar Mühendisi<br/>Python ve Django ile web uygulamaları geliştiriyor.<br/><a href="#">Profili Gör</a></p>               |  <p><b>Ahmet Ekim</b><br/>Yazılım Mühendisi<br/>iOS ve Android platformları için uygulamalar geliştiriyor.<br/><a href="#">Profili Gör</a></p>               |  <p><b>Tuncay Altun</b><br/>Elektrik Elektronik Mühendisi<br/>Hem frontend hem de backend geliştirme konusunda deneyimli.<br/><a href="#">Profili Gör</a></p> |
|  <p><b>Ahmet Faruk DURSUN</b><br/>Backend Developer<br/>Veri analizi ve makine öğrenimi projeleri üzerinde çalışıyor.<br/><a href="#">Profili Gör</a></p> |  <p><b>Ayşe YILMAZ</b><br/>Mobil Uygulama Geliştiricisi<br/>iOS ve Android platformları için uygulamalar geliştiriyor.<br/><a href="#">Profili Gör</a></p> |  <p><b>Elif GÖKÇE</b><br/>DevOps Mühendisi<br/>Sürekli entegrasyon ve dağıtım süreçlerini yönetiyor.<br/><a href="#">Profili Gör</a></p>                    |

[f](#) [t](#) [G](#) [i](#) [in](#) [r](#)

Yazarlar sayfamız artık hem db den besleniyor, hem de admin panelinden müdahale edilebiliyor. 48



```
require 'capybara/re Rails'
```

```
1 Capybara.javascript_driver = :webkit
```

```
2 Category.delete_all; Category.create
```

```
3 Shoulda::Matchers.configure do |config|
```

```
4   config.integrate do |with|
```

```
5     with.test_framework :rspec
```

```
6     with.library :rails
```

```
7   end
```

```
8 end
```

```
9 # Add additional requires below this line. This
```

```
10 # Requires supporting ruby files with support/
```

```
11 # spec/support/ and its subdirectories. These
```

```
12 # run as spec files by default. This will
```

```
13 # in _spec.rb will both be required and
```

```
14 # It is recommended that you configure
```

# Yazar Detay Sayfası



# Yazar Detay Sayfası



**Furkan Durmuş**

Bilgisayar Mühendisi

Python ve Django ile web uygulamaları geliştiriyor.

[Profili Gör](#)

Yazarlar sayfasında Profili gör'e tıklandığında detay sayfasının açılmasını sağlayalım.

# Yazar Detay Sayfası

1- core / templates altına yazar\_detay.html sayfası oluşturalım.

```
✓ core
  > __pycache__
  > migrations
  ✓ templates
    <> anasayfa.html
    <> hakkimizda.html
    <> iletisim.html
    <> yazar_detay.html
    <> yazarlar.html
```

# Yazar Detay Sayfası

- 1- core / templates altına yazar\_detay.html sayfası oluşturalım.
- 2- **Yazar\_detay.html sayfasının içeriğini block içerik kısmı boş olacak şekilde base html extent ederek dolduralım.**

```
core > templates > <> yazar_detay.html
1  {% extends "base.html" %}
2  {% load static %}
3
4  {% block baslik %}{{ title }}{% endblock %}
5  {% block icerik %}
6
7
8  {% endblock %}
9
```

# Yazar Detay Sayfası

- 1- core / templates altına yazar\_detay.html sayfası oluşturalım.
- 2- Yazar\_detay.html sayfasının içeriğini block içerik kısmı boş olacak şekilde base html extent ederek dolduralım.
- 3- **block içerik kısmını geçen ders öğrenmiş olduğumuz şekilde ai kullanarak doldurunuz.**

Nereden başlayalım?

```
from django.db import models
class Author(models.Model):
    full_name = models.CharField(max_length=100)
    title = models.CharField(max_length=100)
    bio = models.TextField()
    profile_url = models.URLField(blank=True, null=True)
    def __str__(self):
        return self.full_name
```

Bu modelime uygun Bootstrap içerikli HTML kodunu istiyorum. Base.html zaten extend edilmiş durumda, sadece content block içine eklenecek kısmı ver. Profil URL alanı ekranda gösterilmeyecek. Bu sayfa yazarlar listesi değil, yazar detay sayfasıdır.



# Yazar Detay Sayfası

1- core / templates altına yazar\_detay.html sayfası oluşturulalım.

2- Yazar\_detay.html sayfasının içeriğini block içerik kısmı boş olacak şekilde base html extent ederek dolduralım.

3- block içerik kısmını geçen ders öğrenmiş olduğumuz şekilde ai kullanarak doldurunuz.

4- İngilizce varsa düzeltiniz, Geri butonu eklediyse url name'i yazarlar yapınız. Bizim yazarlar sayfamızın url ismi 'yazarlar'.

```
{% block icerik %}
<div class="container py-5">

  <div class="row justify-content-center">
    <div class="col-lg-8">

      <div class="card shadow-sm border-0">
        <div class="card-body p-4">
          <h2 class="card-title mb-2">{{ author.full_name }}</h2>
          <h6 class="text-muted mb-4">{{ author.title }}</h6>
          <hr>
          <div class="mt-3">
            <h5 class="mb-3">Hakkında</h5>
            <p class="card-text" style="line-height: 1.7;">{{ author.bio }}</p>
          </div>
        </div>
      </div>
      <div class="mt-4 text-center">
        <a href="{% url 'yazarlar' %}" class="btn btn-outline-secondary">← Geri Dön</a>
      </div>
    </div>
  </div>
{% endblock %}
```

# Yazar Detay Sayfası

```
<h2 class="card-title mb-2">{{ author.full_name }}</h2>  
<h6 class="text-muted mb-4">{{ author.title }}</h6>
```

```
def yazar_detay(request, id):  
    yazar = core.models.Author.objects.get(id=id)  
    context = {  
        "author": yazar  
    }  
    return render(request, "yazar_detay.html", context)
```

5- core / views.py içerisinde yazar\_detay fonksiyonunu tanımlayınız. Html içerisinde veriyi hangi isimle kullandı iseniz burda da context içinde aynı ismi kullanınız.

# Yazar Detay Sayfası



```
urlpatterns = [  
    path('', views.anasayfa, name='anasayfa'),  
    path('iletisim/', views.iletisim, name='iletisim'),  
    path('hakkimizda/', views.hakkimizda, name='hakkimizda'),  
    path('yazarlar/', views.yazarlar, name='yazarlar'),  
    path('yazar_detay/<int:id>/', views.yazar_detay, name='yazar_detay'),  
]
```

6- core / urls.py içerisinde yazar\_detay path tanımlamasını yapınız.



# Yazar Detay Sayfası



```
</p>  
<a href="{% url 'yazar_detay' id=yazar.id %}" class="btn btn-outline-primary btn-sm">Profili Gör</a>  
</div>
```

7- *'Yazarlar.html'* içerisindeki Profili gör butonunda üstteki gibi güncelleme yapınız.



# Yazar Detay Sayfası



## Genel Yapı :

1-

```
</p>  
<a href="{% url 'yazar_detay' id=yazar.id %}" class="btn btn-outline-primary btn-sm">Profili Gör</a>  
</div>
```

2-

```
path('yazar_detay/<int:id>/', views.yazar_detay, name='yazar_detay'),
```

3-

```
def yazar_detay(request, id):  
    yazar = core.models.Author.objects.get(id=id)  
    context = {  
        "author": yazar  
    }  
    return render(request, "yazar_detay.html", context)
```

4-

```
<> yazar_detay.html
```



# Yazar Detay Sayfası



The screenshot displays a web browser window with the URL [127.0.0.1:8000/yazar\\_detay/1/](http://127.0.0.1:8000/yazar_detay/1/). The browser's address bar shows the URL and standard navigation icons. The page content includes a navigation menu with links for 'Kampus Blog', 'Ana Sayfa', 'İletişim', 'Hakkımızda', 'Yazarlar', 'Blog', and 'Kullanıcı'. A search bar with the placeholder text 'yazı ara...' and a green 'Ara' button is located in the top right. The main content area features a profile card for 'Furkan Durmuş', identified as a 'Bilgisayar Mühendisi'. Below the name is a section titled 'Hakkında' with the text 'Python ve Django ile web uygulamaları geliştiriyor.'. A 'Geri Dön' button is positioned below the profile card. At the bottom of the page, there is a row of social media icons for Facebook, Twitter, Google+, Instagram, LinkedIn, and GitHub. The footer contains the text '© 2026 Copyright'.

Sonuç



# Yazar Detay Sayfası



Yazar detay sayfasına mevcut 404.html sayfasını entegre ediniz.

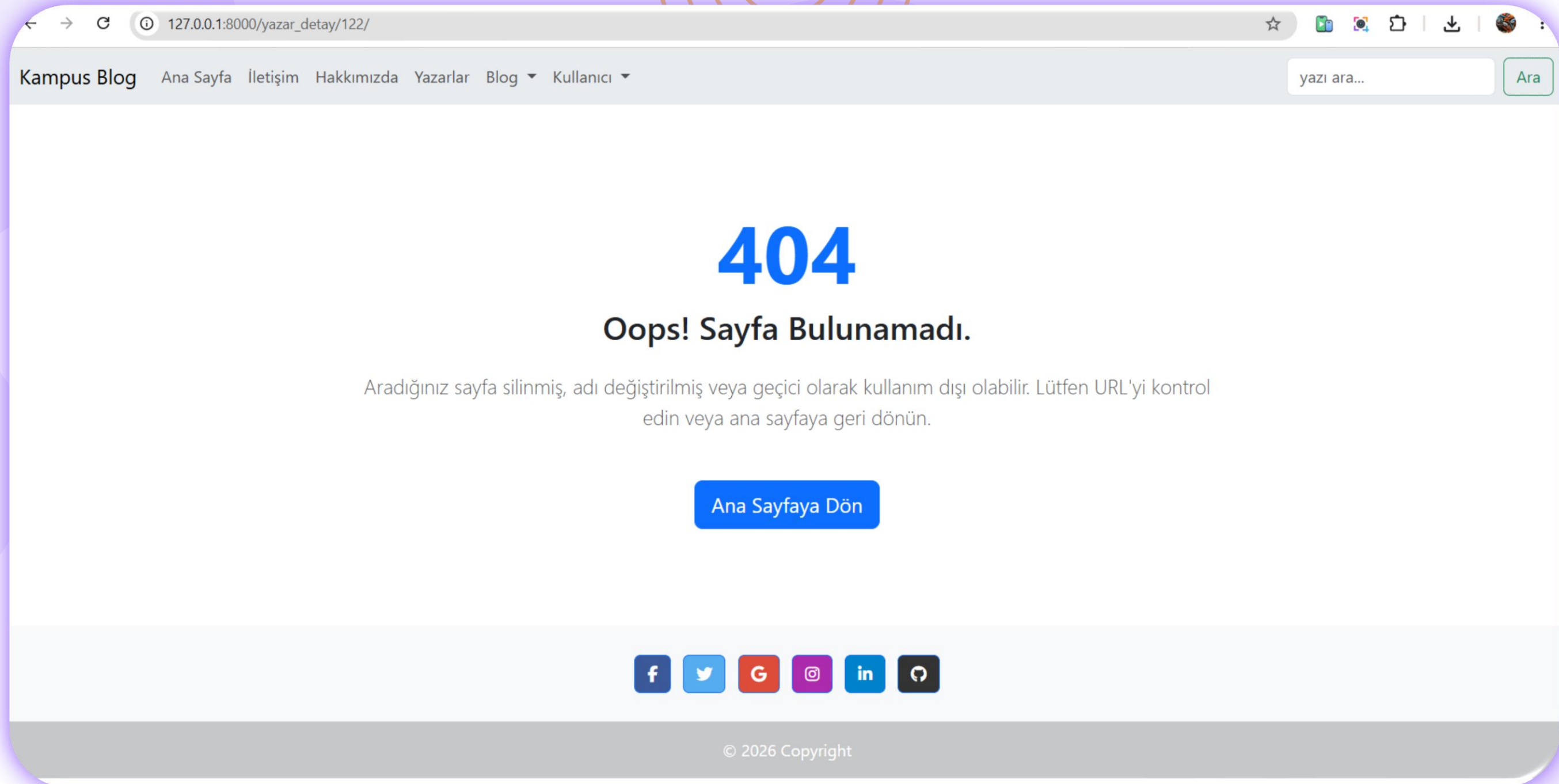
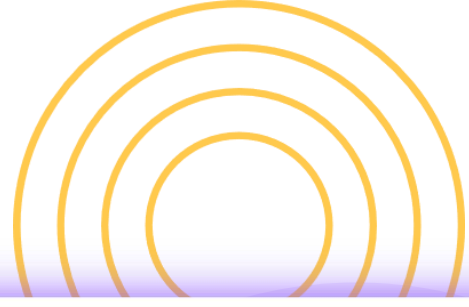


# Yazar Detay Sayfası

Yazar detay sayfasına mevcut 404.html sayfasını entegre ediniz.

```
def yazar_detay(request, id):  
  
    try:  
        yazar = core.models.Author.objects.get(id=id)  
    except:  
        return render(request, "404.html", status=404)  
  
    context = {  
        "author": yazar  
    }  
    return render(request, "yazar_detay.html", context)
```

# Yazar Detay Sayfası



Sonuç



# İletişim Sayfası

```
require 'capybara/reils'
```

```
1 Capybara.javascript_driver = :webkit
```

```
2 Category.delete_all; Category.create
```

```
3 Shoulda::Matchers.configure do |config|
```

```
4   config.integrate do |with|
```

```
5     with.test_framework :rspec
```

```
6     with.library :rails
```

```
7   end
```

```
8 end
```

```
9 # Add additional requires below this line
```

```
10 # Requires supporting ruby files with support/ directory
```

```
11 # spec/support/ and its subdirectories. These files
```

```
12 # run as spec files by default. This will not work if
```

```
13 # in _spec.rb will both be required and the
```

```
14 It is recommended that you configure the
```



# İletişim Sayfası



## İletişim

Bizimle iletişime geçmek için aşağıdaki bilgileri kullanabilirsiniz.

### Mesaj Gönder

Ad Soyad

Email

Mesaj

Gönder

İletişim sayfasında bulunan form doldurulduğunda bunu admin panelinde göstermeliyiz.



# İletişim Sayfası

Core / models.py içerisine Author modelinin altına ContactMessage class'ını yandaki gibi ekleyiniz.

```
core > models.py > ...  
11  
12  
13 class ContactMessage(models.Model):  
14  
15     full_name = models.CharField(max_length=100)  
16     email = models.EmailField()  
17     message = models.TextField()  
18     created_at = models.DateTimeField(auto_now_add=True)  
19  
20     def __str__(self):  
21         return self.full_name  
22  
23
```



# İletişim Sayfası



Yaptığımız bu modelin Database'e yansması için projeyi durdurup terminalde sırasıyla ;

```
python manage.py makemigrations
```

```
python manage.py migrate
```

Kodlarını çalıştırınız.



# İletişim Sayfası

Core / admin.py'de ContactMessage'ı register ediniz.

```
core > admin.py
1  from django.contrib import admin
2  from .models import Author, ContactMessage
3
4  admin.site.register(Author)
5  admin.site.register(ContactMessage)
6
7
```

# İletişim Sayfası

Core / views.py'de import kısmına bu 2 satırı ekleyiniz.

```
core > views.py > ...  
1 from django.shortcuts import render  
2 import core.models  
3 import blog.models  
4 from .models import ContactMessage  
5 from django.shortcuts import redirect  
6  
7
```

# İletişim Sayfası

Core / views.py'de de mevcutta olan def iletişim'i yandaki gibi güncelleyiniz.

```
8  def iletişim(request):
9  if request.method == "POST":
10     ad_soyad = request.POST.get('full_name')
11     email = request.POST.get('email')
12     mesaj = request.POST.get('message')
13     ContactMessage.objects.create(
14         full_name=ad_soyad,
15         email=email,
16         message=mesaj
17     )
18     return redirect('iletisim')
19     return render(request, 'iletisim.html')
```

# İletişim Sayfası



Core / templates/ iletisim.html de mevcutta olan  
<form> etiketini yandaki gibi güncelleyiniz.

```
48  
49  
50 <form method="POST">  
51   {% csrf_token %}  
52   <div class="mb-3">
```



# İletişim Sayfası

```
form method="POST">
  {% csrf_token %}
  <div class="mb-3">
    <label class="form-label">Ad Soyad</label>
    <input name="full_name" type="text" class="form-control" placeholder="Adınızı giriniz">
  </div>

  <div class="mb-3">
    <label class="form-label">Email</label>
    <input name="email" type="email" class="form-control" placeholder="Email adresiniz">
  </div>

  <div class="mb-3">
    <label class="form-label">Mesaj</label>
    <textarea name="message" class="form-control" rows="4" placeholder="Mesajınızı yazınız"></textarea>
  </div>
```

Core / templates/ iletisim.html de mevcutta olan alanlara üstteki gibi name veriniz.

# İletişim Sayfası



Çalışma mantığı :

1 - iletişim.html formu post edecek

2- iletişim.def bu postu yakalayacak ve bunu db ye kayıt edecek.

3- Admin paneline bu mesajlar gözükecek.

```
<form method="POST">
  {% csrf_token %}
  <div class="mb-3">
    <label class="form-label">Ad Soyad</label>
    <input name="full_name" type="text" class=
  </div>
```

```
def iletisim(request):
    if request.method == "POST":
        ad_soyad = request.POST.get('full_name')
        email = request.POST.get('email')
        mesaj = request.POST.get('message')
        ContactMessage.objects.create(
            full_name=ad_soyad,
            email=email,
            message=mesaj
        )
        return redirect('iletisim')
    return render(request, 'iletisim.html')
```

```
core > admin.py
1 from django.contrib import admin
2 from .models import Author, ContactMessage
3
4 admin.site.register(Author)
5 admin.site.register(ContactMessage)
6
7
```



# İletişim Sayfası



| CORE                              |        |
|-----------------------------------|--------|
| Authors                           | + Ekle |
| Contact messages                  | + Ekle |
| KİMLİK DOĞRULAMA VE YETKİLENDİRME |        |
| Gruplar                           | + Ekle |
| Kullanıcılar                      | + Ekle |

Eylem: -----  0 / 1 seçildi

- CONTACT MESSAGE
- [Furkan Durmuş](#)

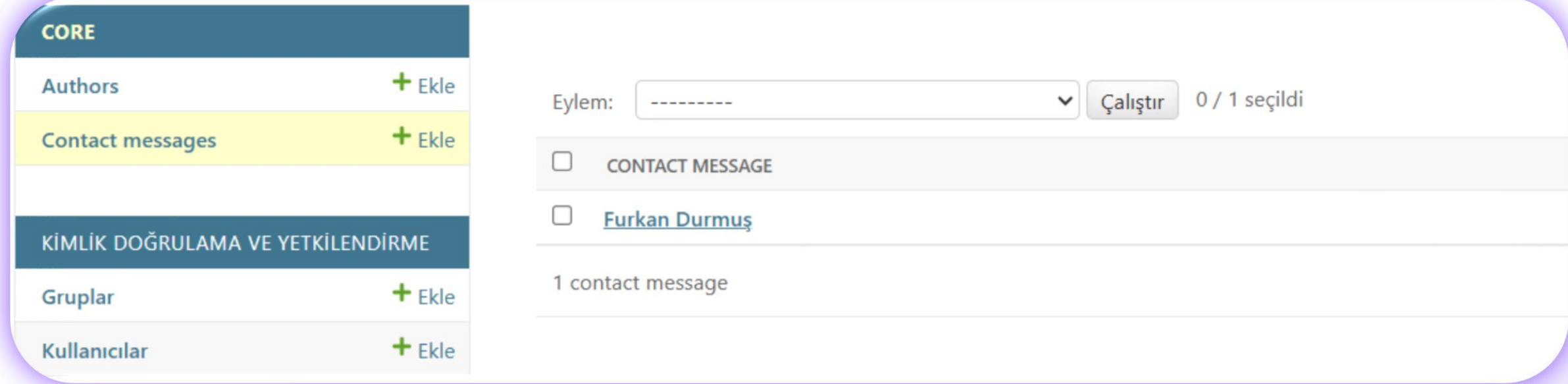
1 contact message

Projeyi çalıştırıp test edebilirsiniz.



# İletişim Sayfası

Admin panelinde contact message tablosu çok kullanışlı değil. Bunun için admin panelini özelleştirebiliriz.



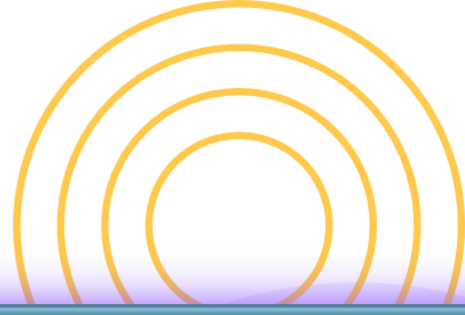
The screenshot displays the admin panel interface. On the left is a sidebar menu with the following items: CORE, Authors (+ Ekle), Contact messages (+ Ekle), KİMLİK DOĞRULAMA VE YETKİLENDİRME, Gruplar (+ Ekle), and Kullanıcılar (+ Ekle). The main content area shows a table with one row: CONTACT MESSAGE. Above the table, there is a search bar with the text "Eylem: -----" and a "Çalıştır" button. Below the table, it indicates "1 contact message".

# İletişim Sayfası

core/admin.py de  
ContactMessage register kısmını  
silip yandaki kodu yazınız.

```
admin.py > ...
1 from django.contrib import admin
2 from .models import Author, ContactMessage
3
4 admin.site.register(Author)
5
6 class ContactMessageAdmin(admin.ModelAdmin):
7
8     list_display = (
9         'full_name',
10        'email',
11        'message',
12        'created_at'
13    )
14
15
16 admin.site.register(ContactMessage, ContactMessageAdmin)
17
```

# İletişim Sayfası



Django yönetimi

HOŞ GELDİNİZ, ADMIN. SİTEYİ GÖSTER / PAROLAYI DEĞİŞTİR / OTURUMU KAPAT

Giriş > Core > Contact messages

Süzmek için yazmaya başlayın...

BLOG

Blog posts + Ekle

CORE

Authors + Ekle

Contact messages + Ekle

KİMLİK DOĞRULAMA VE YETKİLENDİRME

Gruplar + Ekle

Kullanıcılar + Ekle

Değiştirmek için contact message seçin

CONTACT MESSAGE EKLE +

Eylem: ----- Çalıřtır 0 / 1 seçildi

| <input type="checkbox"/> | FULL NAME                     | EMAIL                       | MESSAGE      | CREATED AT          |
|--------------------------|-------------------------------|-----------------------------|--------------|---------------------|
| <input type="checkbox"/> | <a href="#">Furkan Durmuş</a> | furkan.durmus@samsun.edu.tr | Test mesajı. | 13 Mayıs 2026 21:47 |

1 contact message

Sonuç

**Son**