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/rails'
Capybara.javascript_driver
Category.delete_all;
Shoulda::Matchers.configure
config.integrate
with.test_framework
with.library :rails
end
end
Add additional requires
Requires supporting
spec/support/ and its
run as spec files by default.
in _spec.rb will both be required
It is recommended

Admin Panel

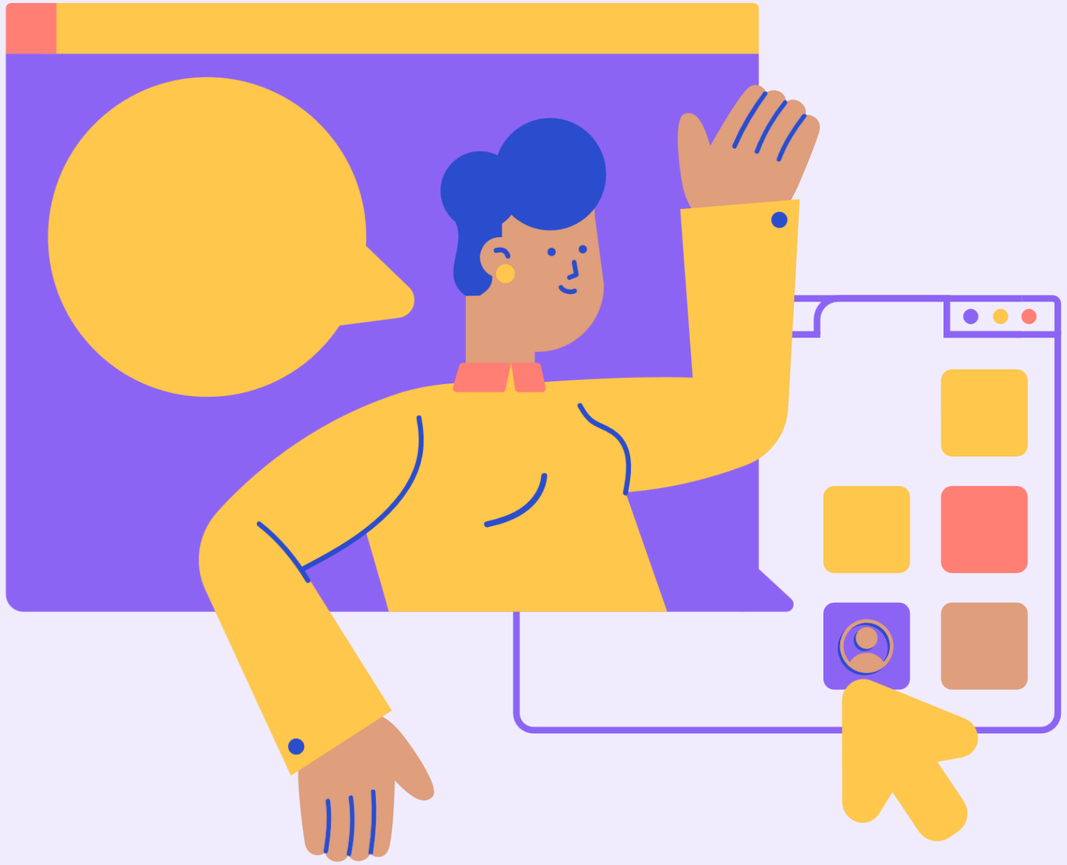
Admin Paneli Nedir ?

Admin Paneli Nedir ?

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

Yani geliştirici veya site yöneticisi:

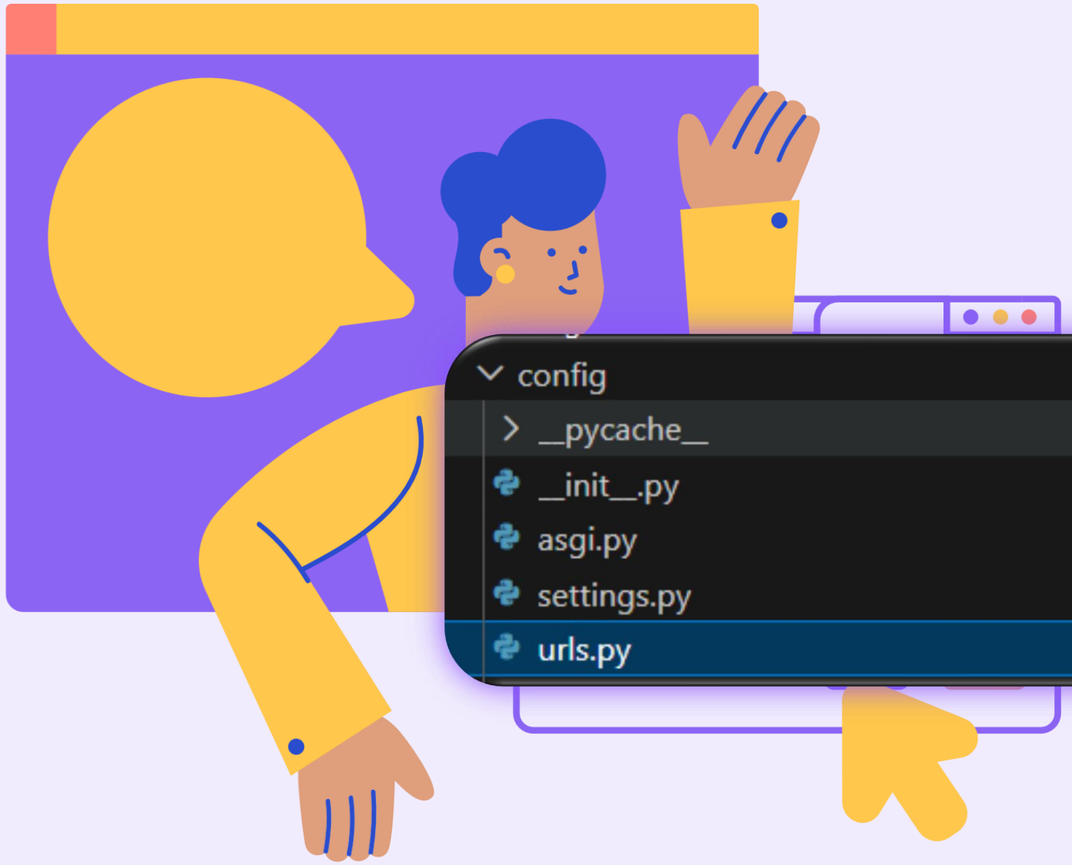
- veri ekleyebilir,
- silebilir,
- güncelleyebilir,
- kullanıcıları yönetebilir,
- modelleri kontrol edebilir.



Admin Paneli

Admin panelinin URL'ini projenizde bulabilir misiniz ?

Admin Paneli

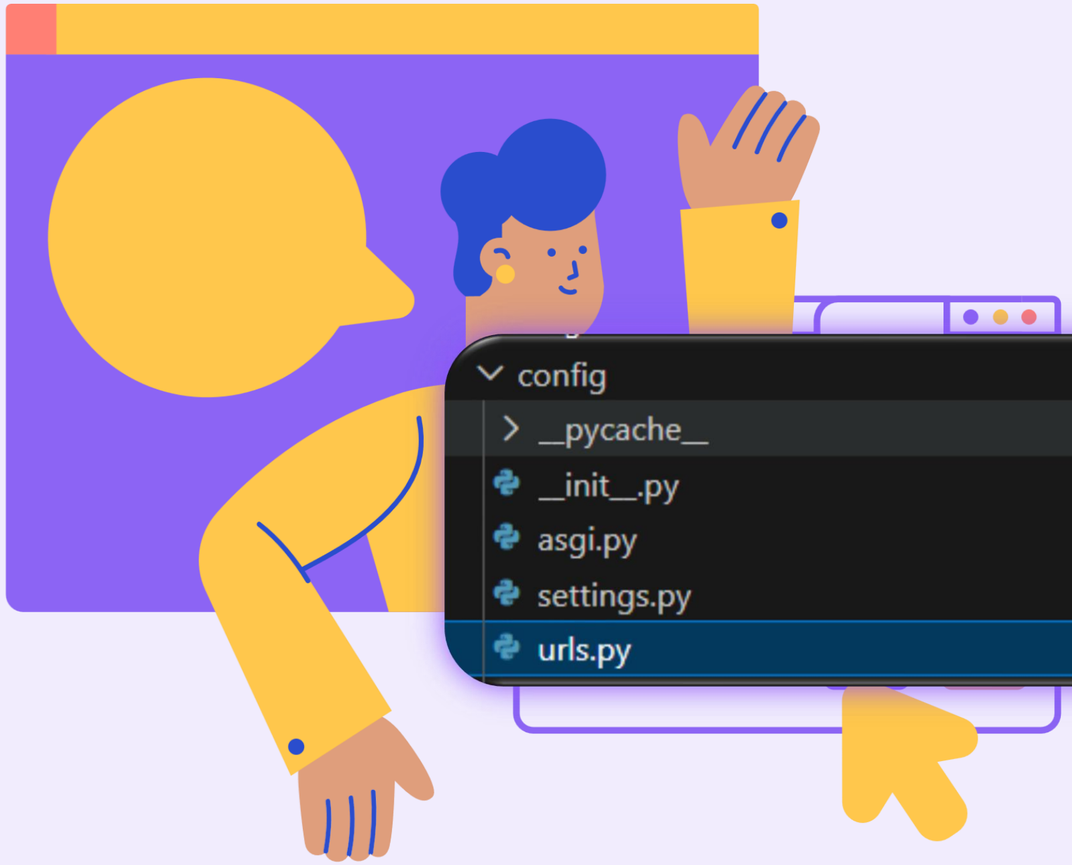


config
- __pycache__
- __init__.py
- asgi.py
- settings.py
- urls.py

```
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

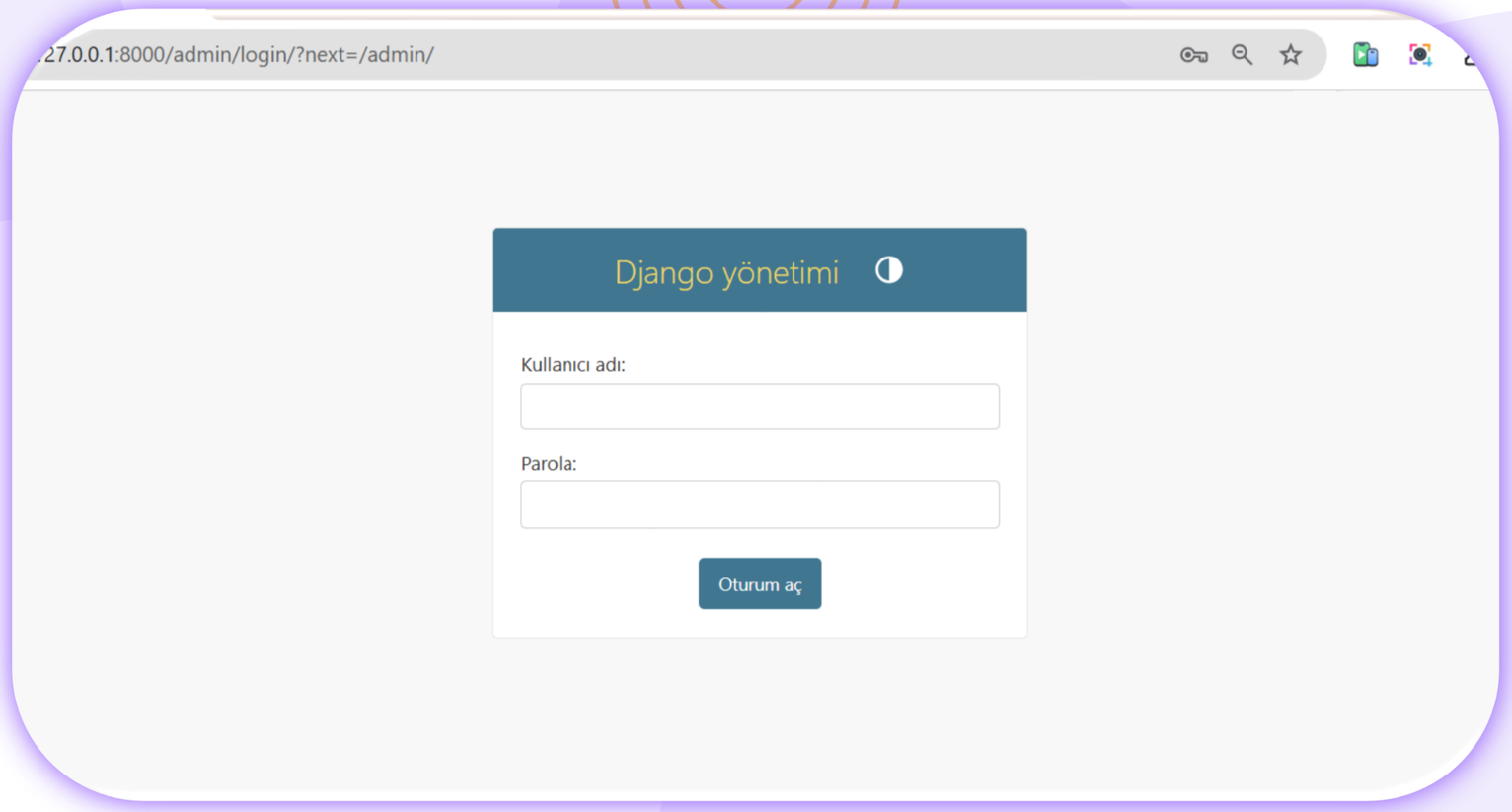


config
- __pycache__
- __init__.py
- asgi.py
- settings.py
- urls.py

```
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



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.

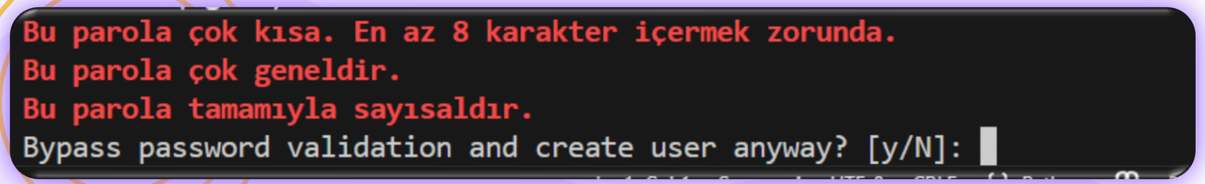
4- ’E-posta Adresi ‘ kısmını **boş** bırakabilirsiniz.

5- Şifre girdiğinizde ekranda güvenlik sebebiyle gözükmez! Ama arka planda yazar.

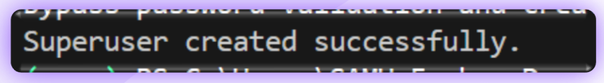
6- ‘Password’ kısmını istediğiniz şifre yapabilirsiniz. Şimdilik **1234** yazabilirsiniz.

7- Şifreyi tekrar giriniz.

Admin Paneli



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

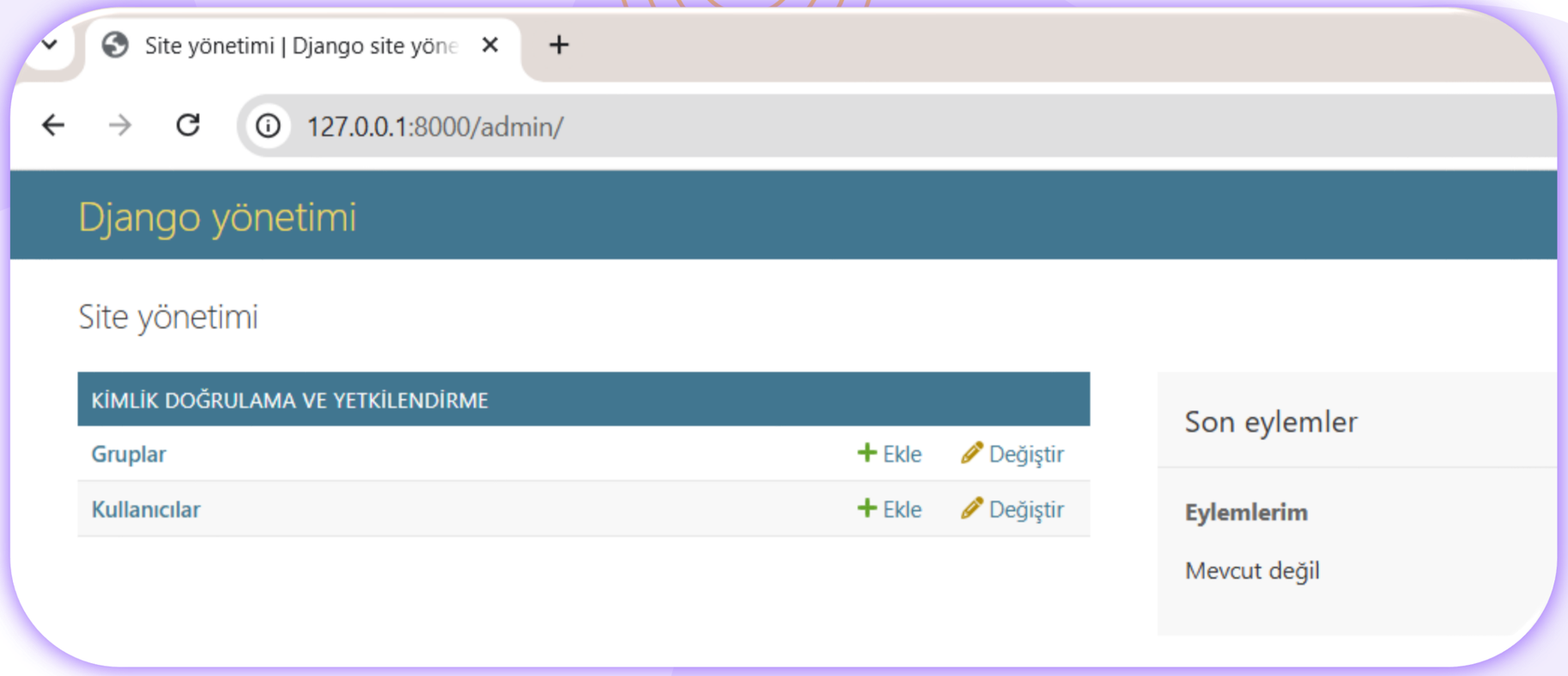
9- 

yazısını gördüyseniz başarılı bir şekilde süper user oluştu.

10- Projeyi tekrar başlatın.

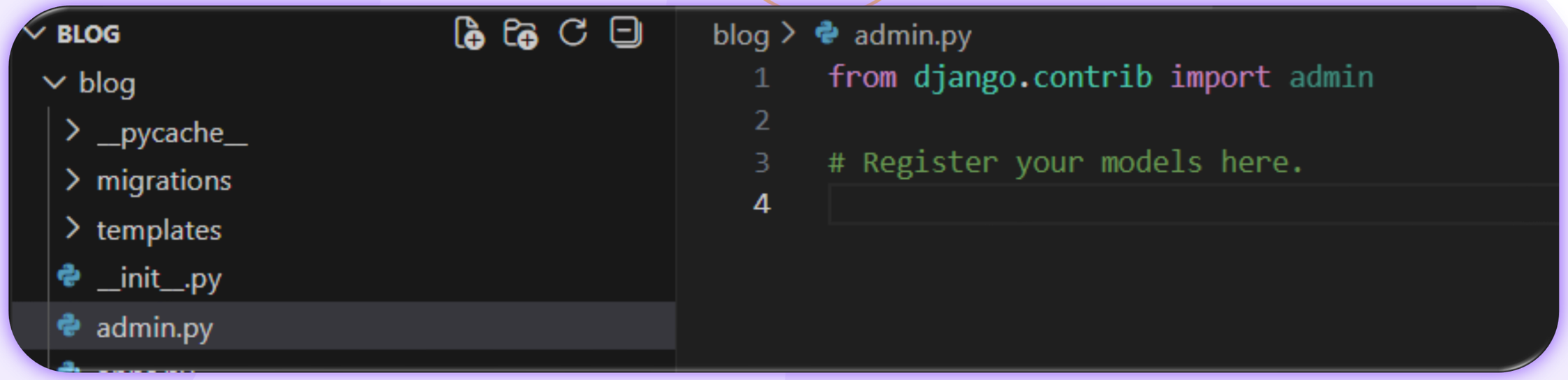
11- Admin paneline oluşturduğunuz bilgilerle giriş yapabilirsiniz.

Admin Paneli



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

Admin Paneli



Blog app'i altındaki admin.py dosyasını açınız.

Admin Paneli

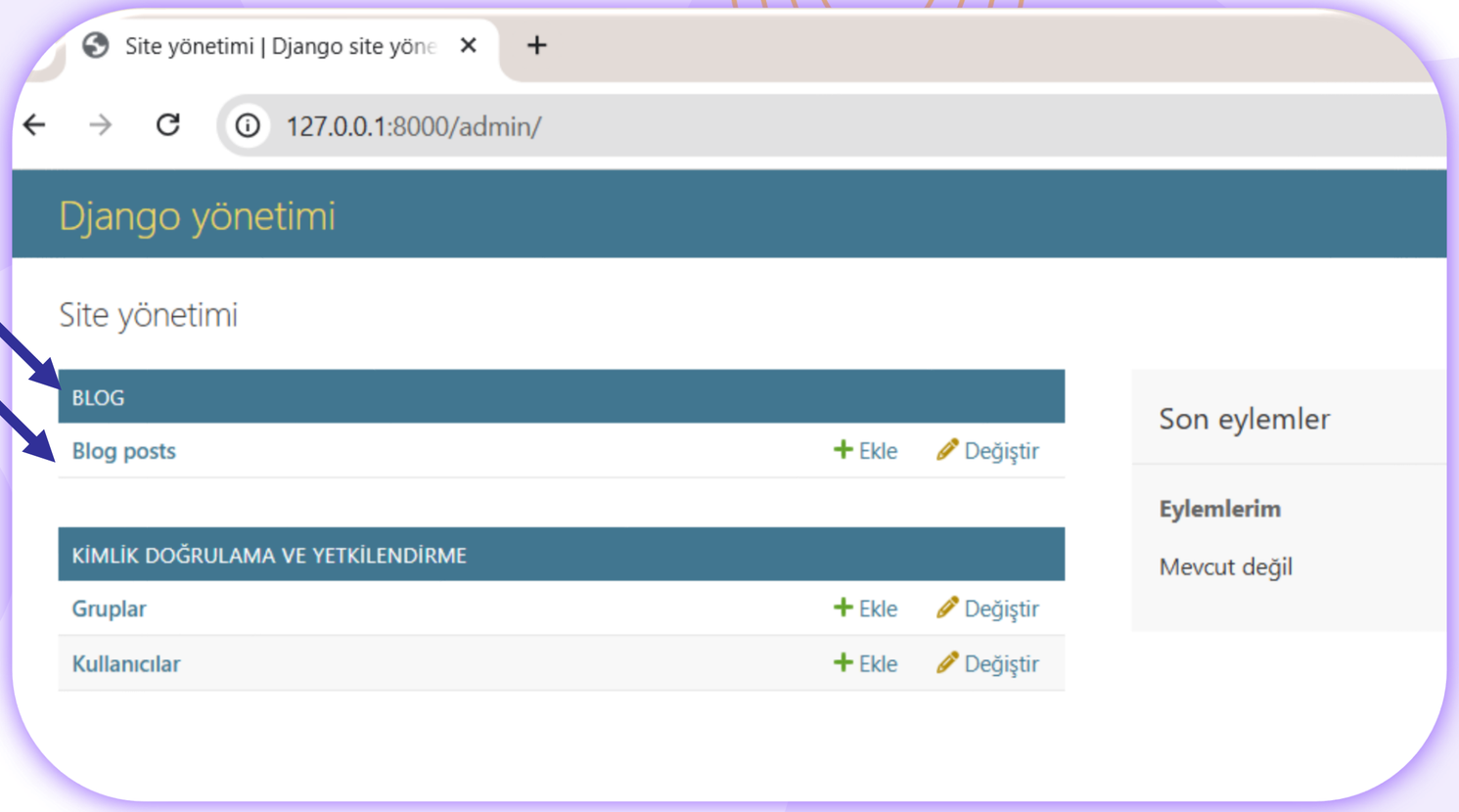
blog > admin.py

```python
from django.contrib import admin
from .models import BlogPost

admin.site.register(BlogPost)

```

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

Admin Paneli

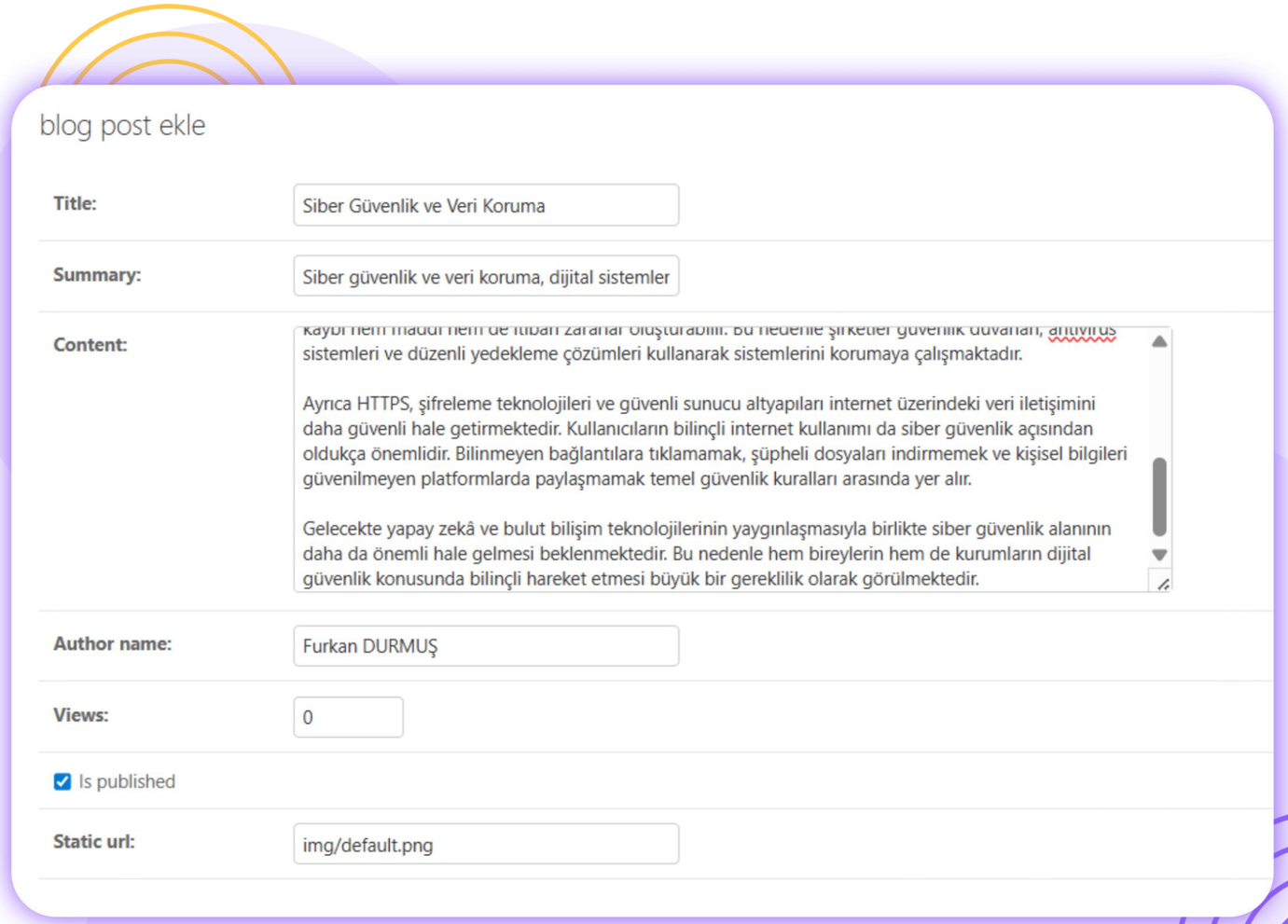
APP adı
Tablo adı



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

Admin Paneli

Default kısımların otomatik doldurulduğunu görebilirsiniz.



Admin Paneli

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

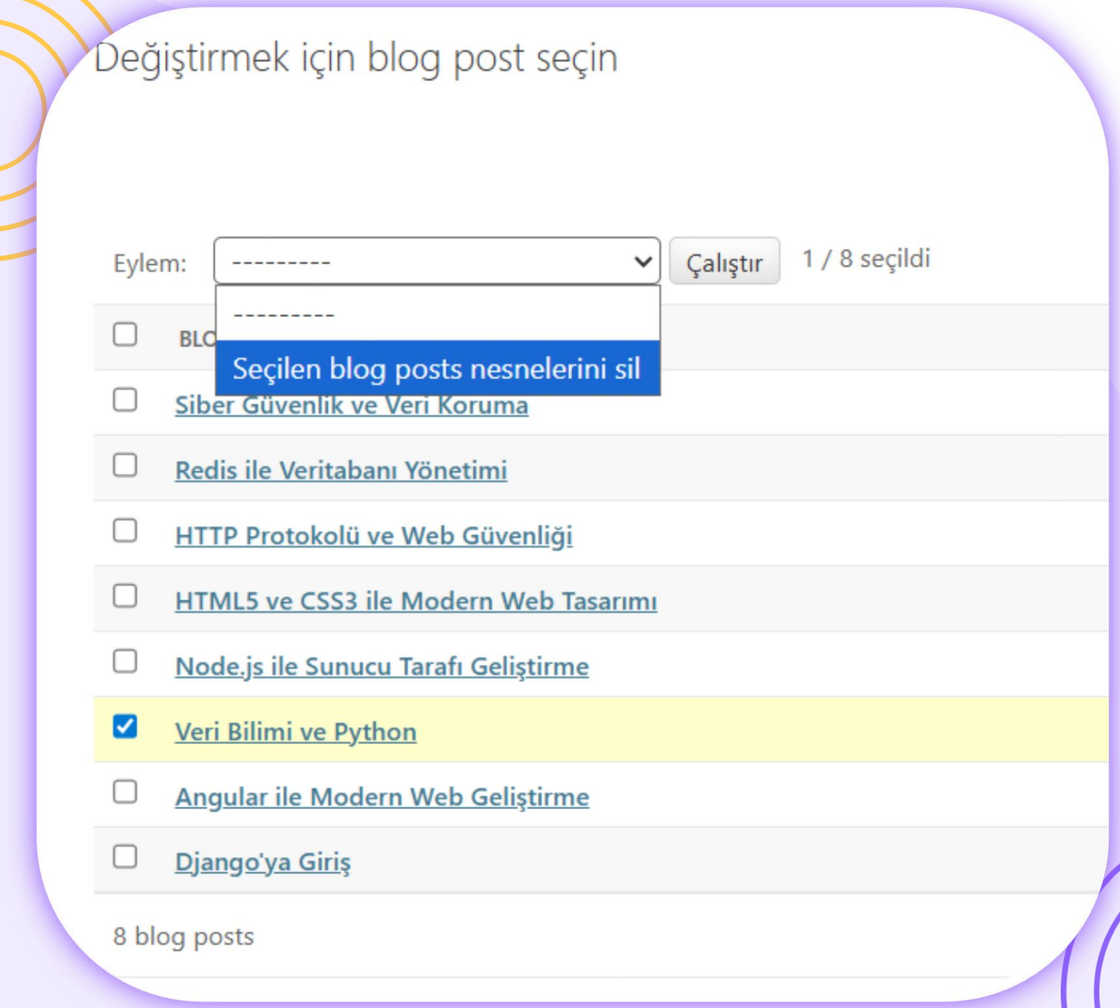
Admin Paneli

```python
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
```

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

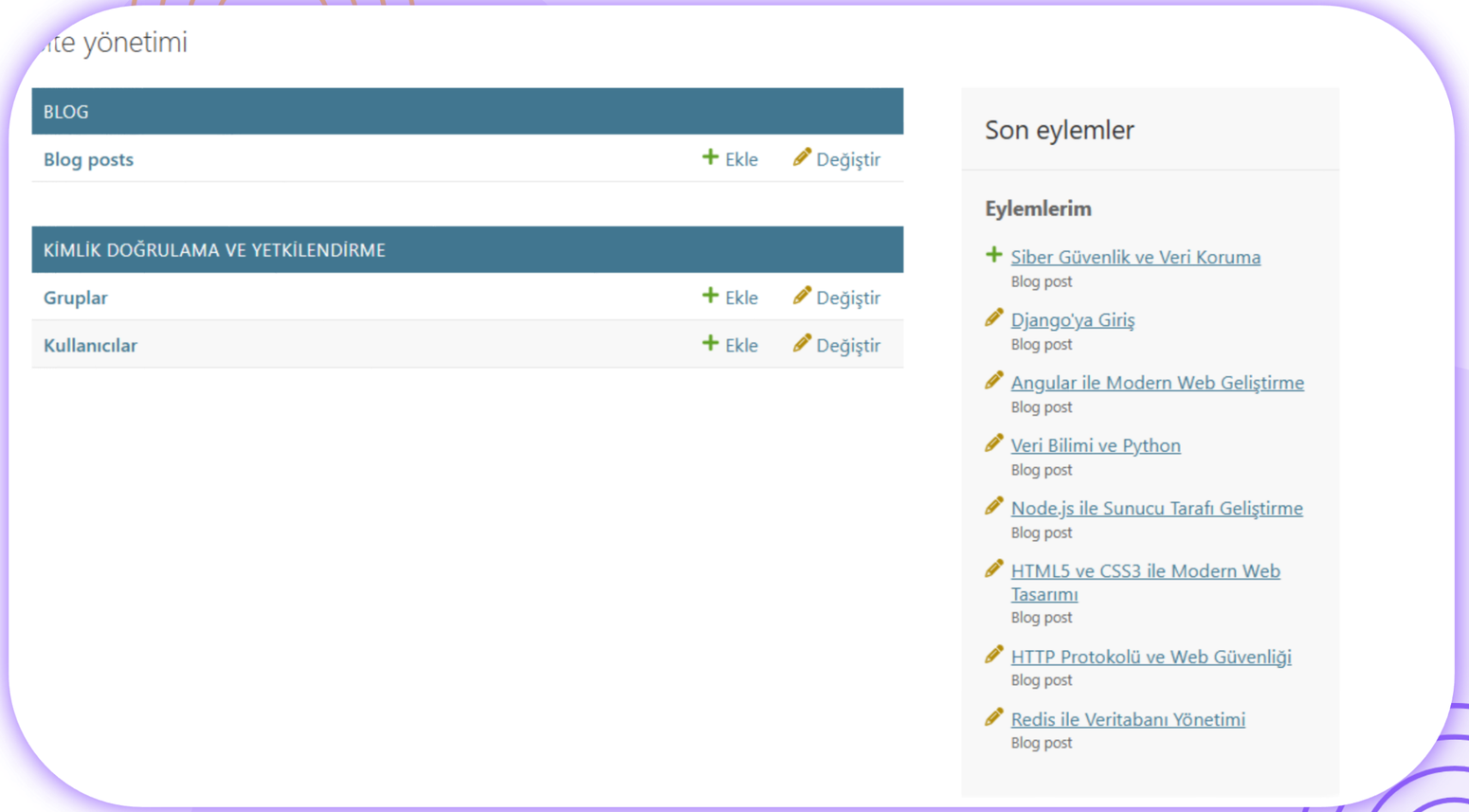
Admin Paneli

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



Admin Paneli

Admin anasayfasında sağ frame'de Son Eylemleri görebilirsiniz.

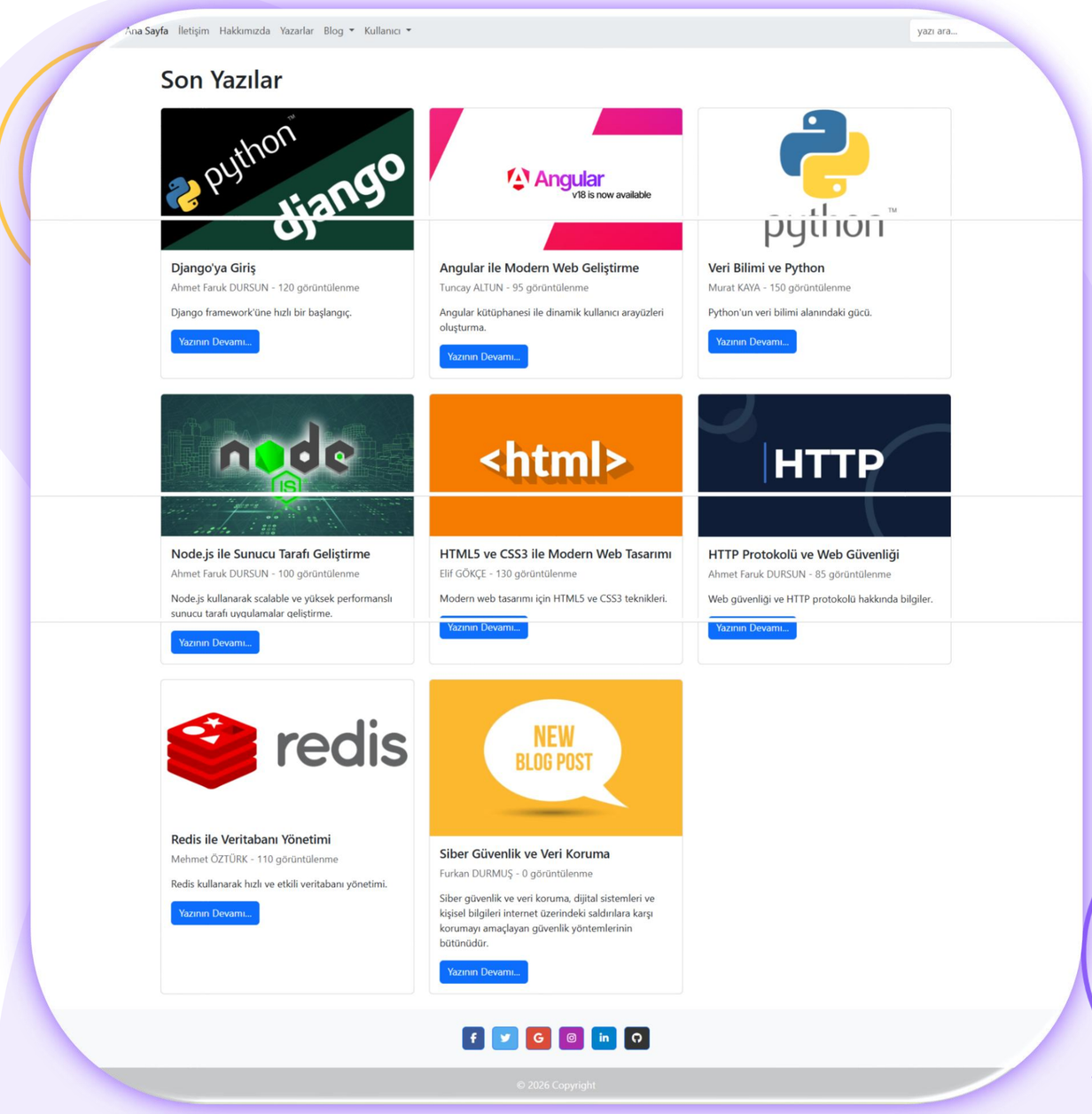


Admin Paneli

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

Admin Paneli

Sonuç



ORM Sorguları

ORM Sorguları

ALL()

Tablodaki tüm kayıtları getirir.

SELECT * FROM blog_blogpost;

```python
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)

```
def blog_detay(request, id):

    try:
        yazi = blog.models.BlogPost.objects.get(id=id)

    except:
        return render(request, "404.html", status=404)
```

ÖNEMLİ!

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

ORM Sorguları

FILTER()

Şarta uyan tüm kayıtları getirir.

BlogPost.objects.filter(is_published=True)

```python
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.

ÖNEMLİ!

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

Filter **büyük küçük harf duyarlıdır.**

Bulamazsa hata vermez.

ORM Sorguları

FILTER(__contains)

```python
def anasayfa(request):

    bloglar = blog.models.BlogPost.objects.filter(title__contains="veri")

    context = {
        "bloglar": bloglar
    }
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

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.

ORM Sorguları

FILTER(__icontains)

```
def anasayfa(request):

    bloglar = blog.models.BlogPost.objects.filter(title__icontains="veri")

    context = {
        "bloglar": bloglar
    }
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

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 duyarsızdır.**

ORM Sorguları

FILTER(__gt)

Sayısal olarak ..'dan büyük olanları getirir.

BlogPost.objects.filter(views__gt=100)

```python
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**'dır.

ORM Sorguları

FILTER(__lt)

Sayısal olarak ..'dan küçük olanları getirir.

BlogPost.objects.filter(views__lt=100)

```
def anasayfa(request):

    bloglar = blog.models.BlogPost.objects.filter(views__lt=50)

    context = {
        "bloglar": bloglar
    }
```

Anasayfa def'te üstteki sorguyu çalıştırınız.

ÖNEMLİ!

lt açılımı **less than**.

Eşit olanları getirmez, **eşit veya küçük** olanlar için **__lte** kullanılmalı.

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

ORM Sorguları

ORDER BY()

Db verilerinde sıralama yapar.

BlogPost.objects.order_by('created_at')

ÖNEMLİ!

Eskiden yeniye, küçükten büyüğe sıralar.

```python
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.

```python
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.

```python
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.

```python
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.

```python
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.

ÖNEMLİ!

ORM den gelen veriyi düzenlememize yarar.

Her zaman en sona yazılır.

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

```python
def anasayfa(request):

    bloglar = blog.models.BlogPost.objects.order_by('-created_at')[:6]

    context = {
        "bloglar": bloglar
    }

    return render(request, "anasayfa.html", context)
```

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 kısmınına 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

```python
from django.contrib import admin
from .models import Author

admin.site.register(Author)

```

Yazarlar Sayfası

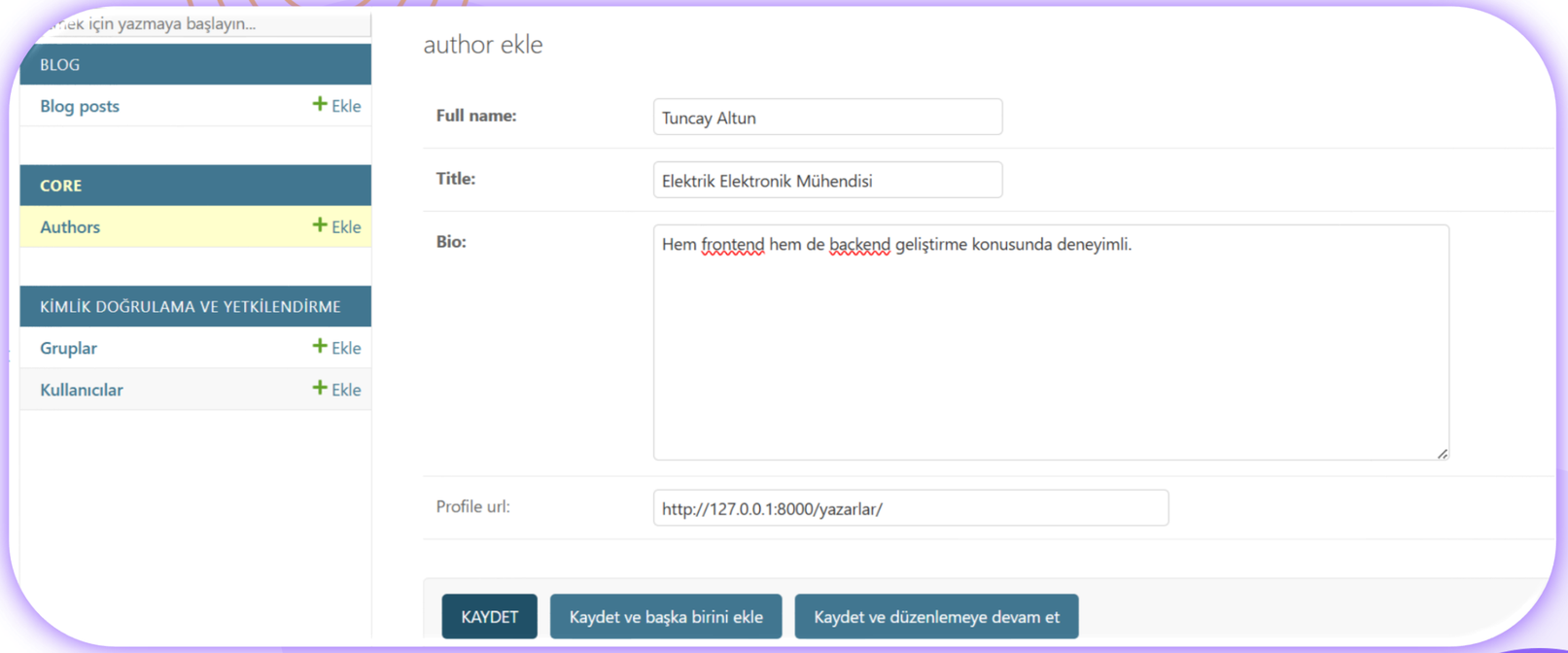
2- Yazarlar def'inde elle girilen veri silinmeli, db deki ilgili tablodaki tüm veri orm ile çekilmeli.

```python
def yazarlar(request):
    yazarlar = core.models.Author.objects.all()

    context = {
        "yazarlar": yazarlar,
        "title": "Yazarlar",
        "toplam_yazar": len(yazarlar),
    }

    return render(request, "yazarlar.html", context)
```

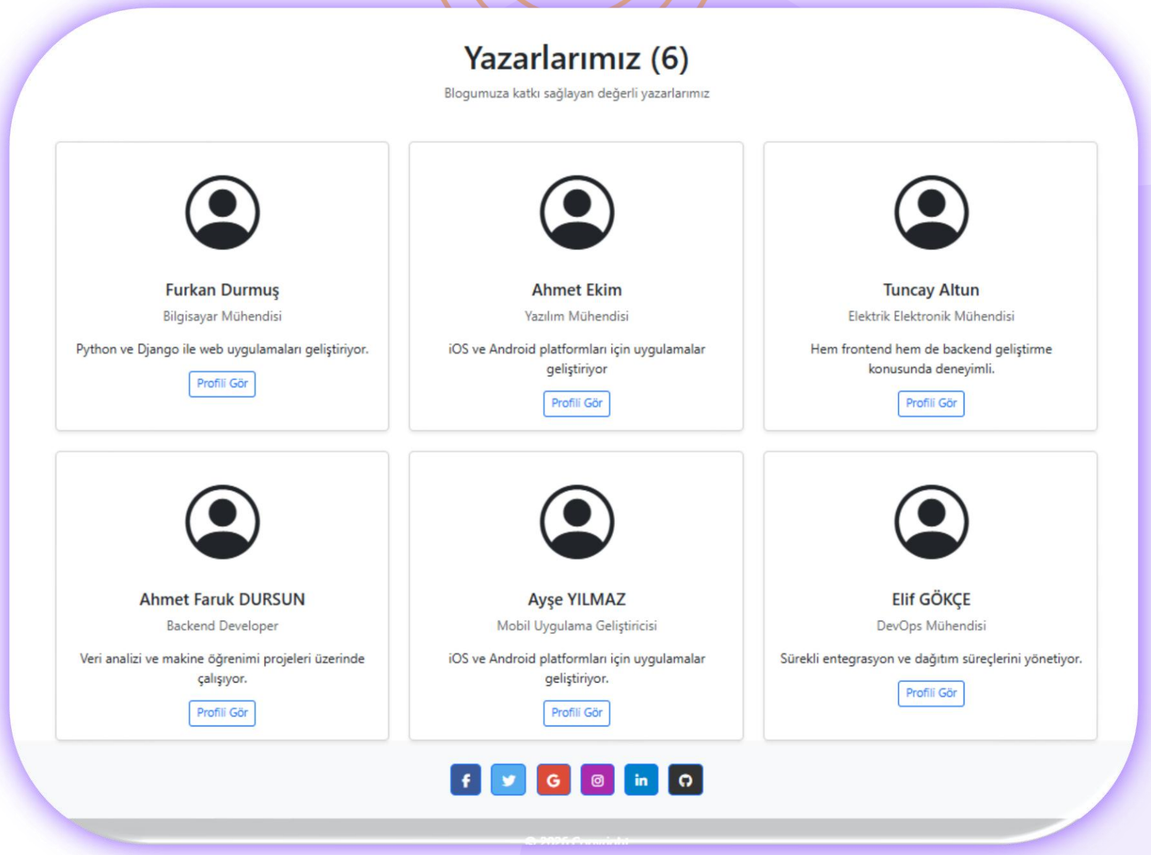
Yazarlar Sayfası

3- Admin panelinden yazar eklenmeli.
(profile url kısmınına şimdilik http://127.0.0.1:8000/yazarlar/ yazınız.)

Toplamda 6 yazar ekleyiniz.



Yazarlar Sayfası



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

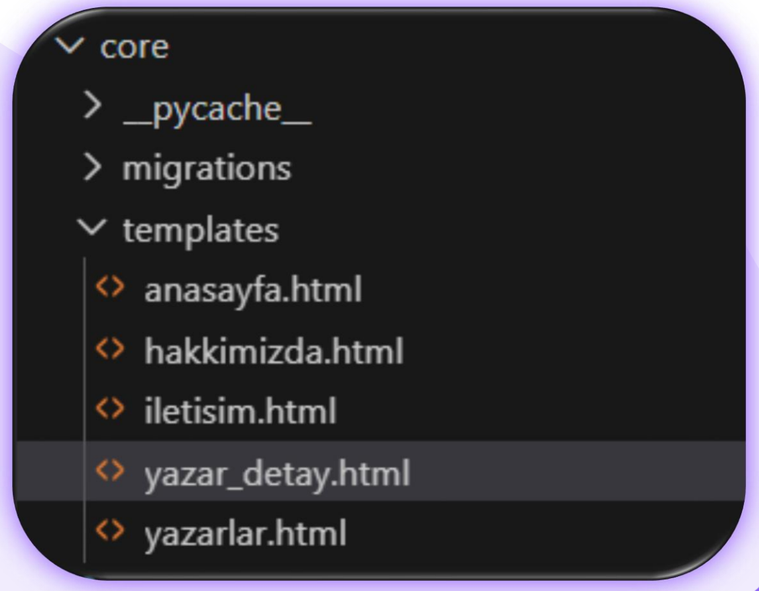
Yazar Detay Sayfası

Yazar Detay Sayfası

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

Yazar Detay Sayfası

1- core / templates altına yazar_detay.html sayfası oluşturalım.



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

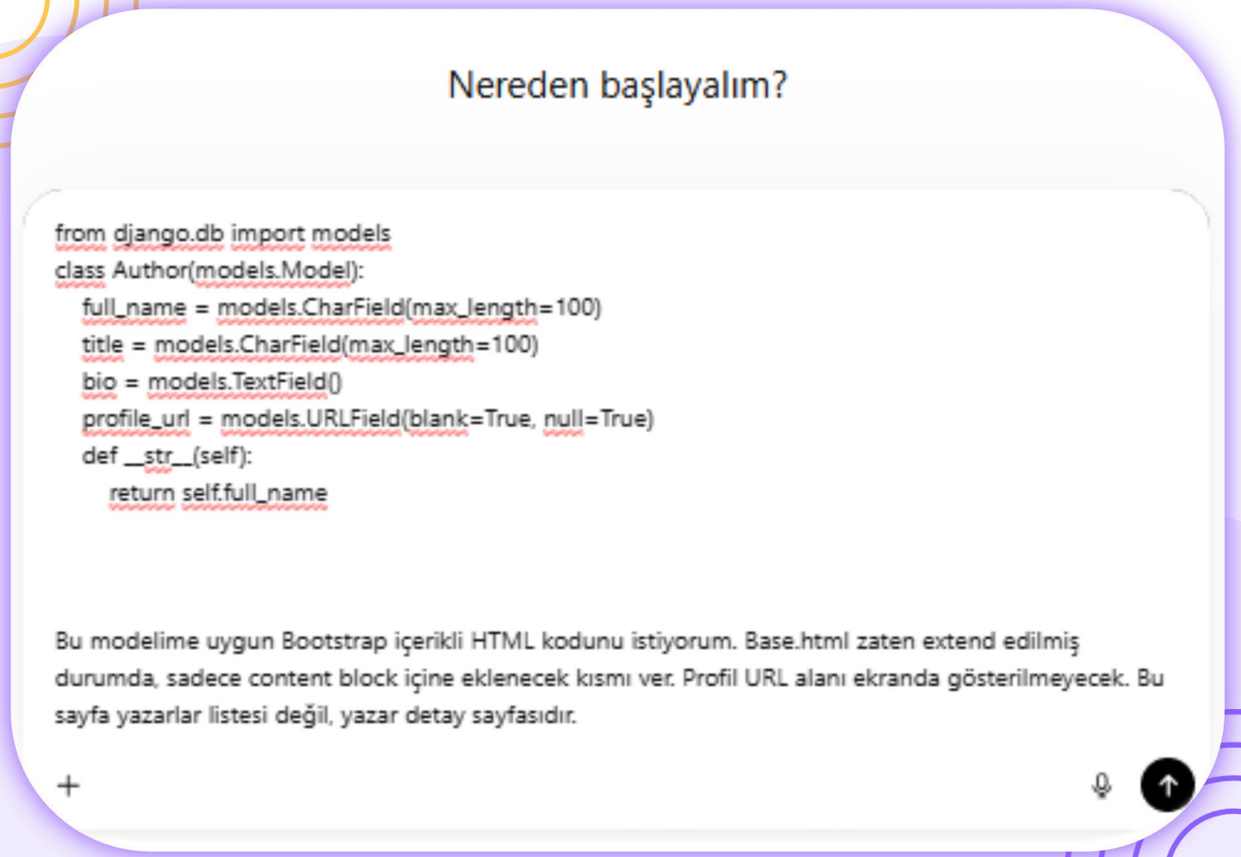
```
1   {% extends "base.html" %}
2   {% load static %}
3
4   {% block baslik %}{{ title }}{% endblock %}
5   {% block icerik %}
6
7
8   {% endblock %}
9
```

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.



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.

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>
</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>
```

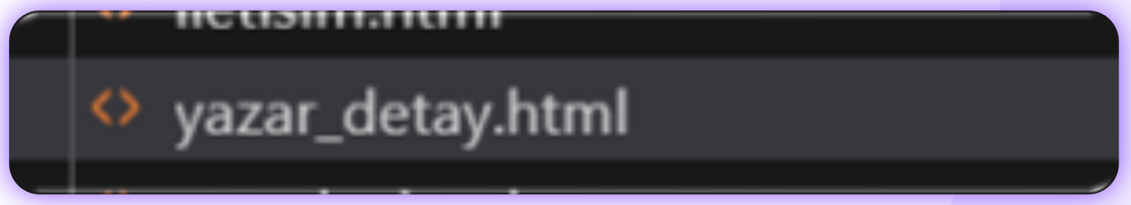
2-

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

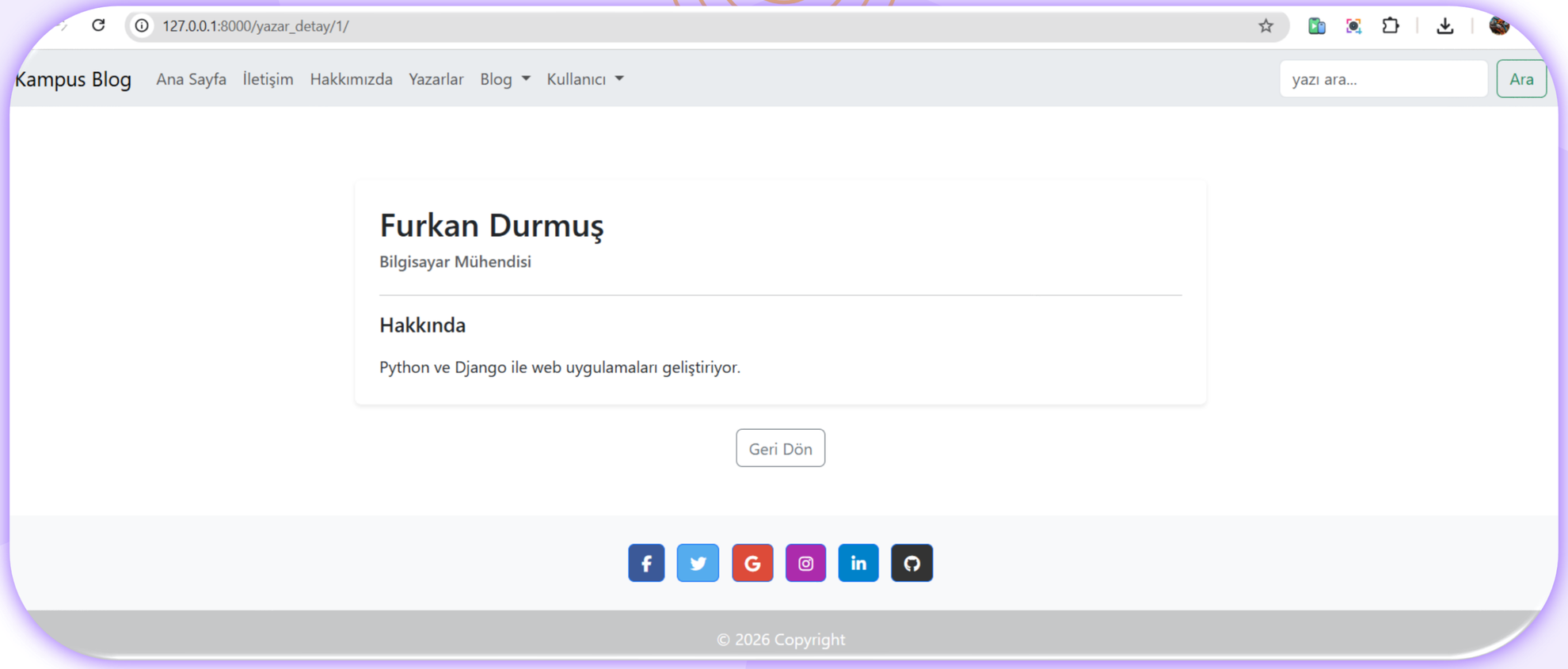
3-

```
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 Sayfası



Sonuç

Yazar Detay Sayfası

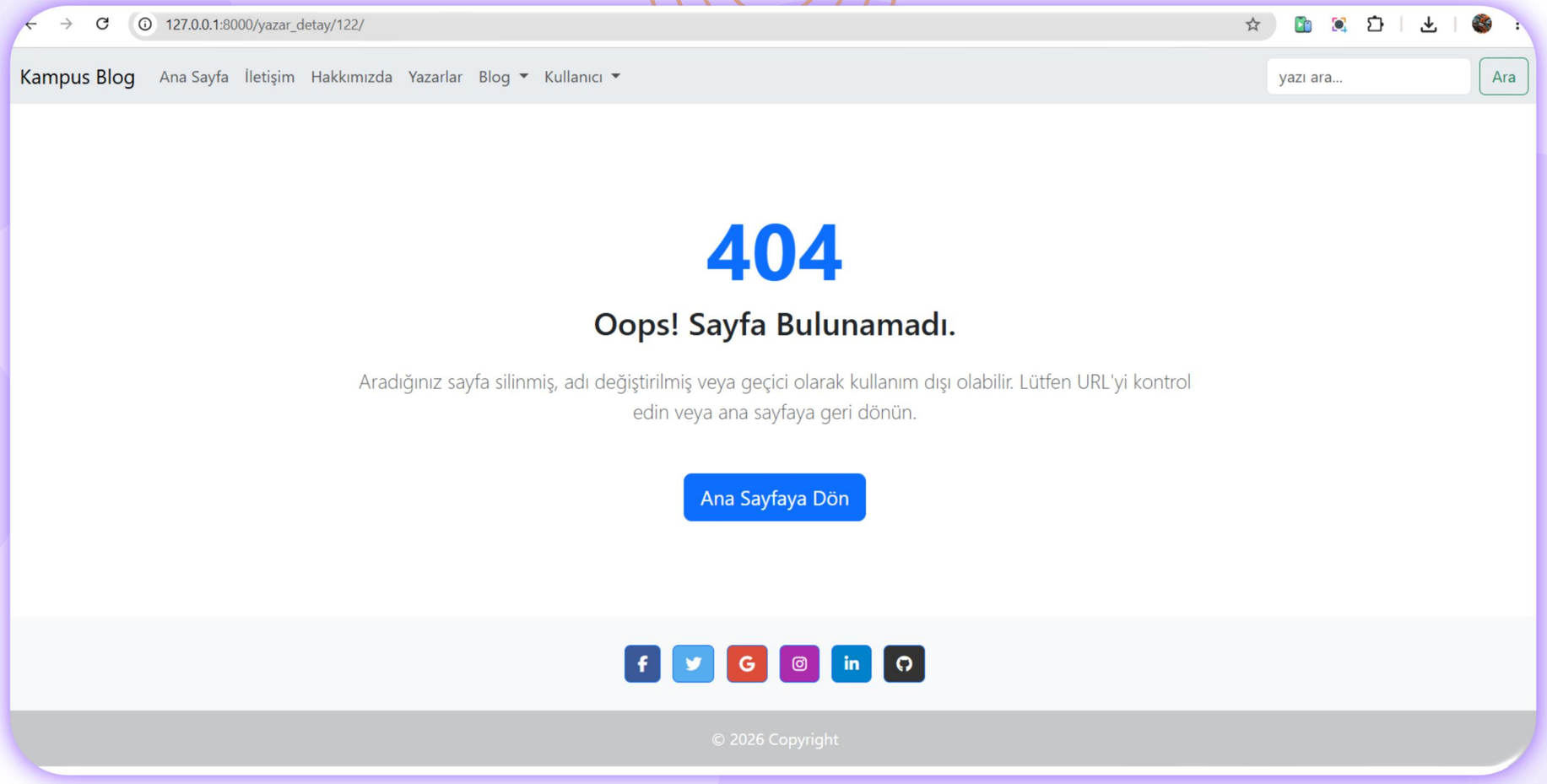
Yazar detay sayfasına mevcut 404.html sayfasını entegre ediniz.

Yazar Detay Sayfası

Yazar detay sayfasına mevcut 404.html sayfasını entegre ediniz.

```python
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)
```

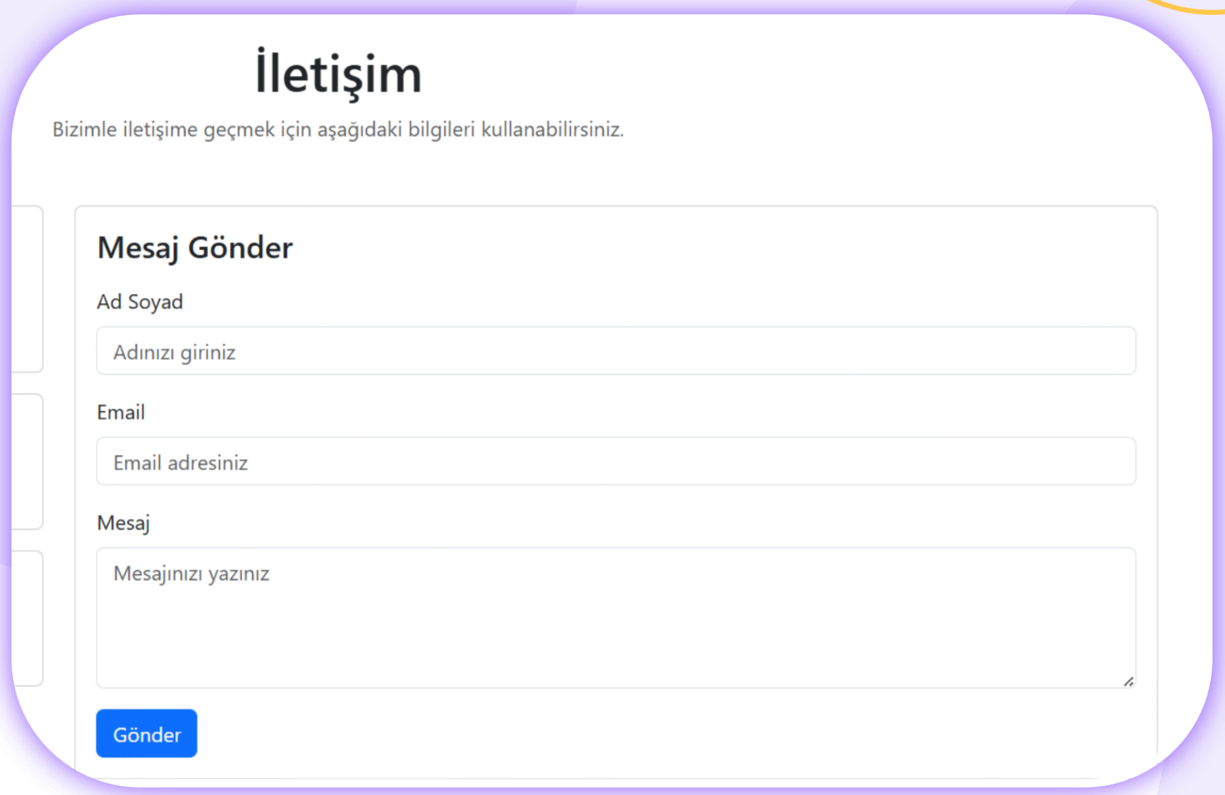
Yazar Detay Sayfası



Sonuç

İletişim Sayfası

İletişim Sayfası



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 > ...

```
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 yansıması için projeyi durdurup terminalde sırasıyla ;

python manage.py makemigrations

python manage.py migrate

Kodlarını çalıştırınız.

İletişim Sayfası

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

core > admin.py

```python
from django.contrib import admin
from .models import Author, ContactMessage

admin.site.register(Author)
admin.site.register(ContactMessage)

```

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
```

İletişim Sayfası

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

```python
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')
```

İletişim Sayfası

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

```
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>
```

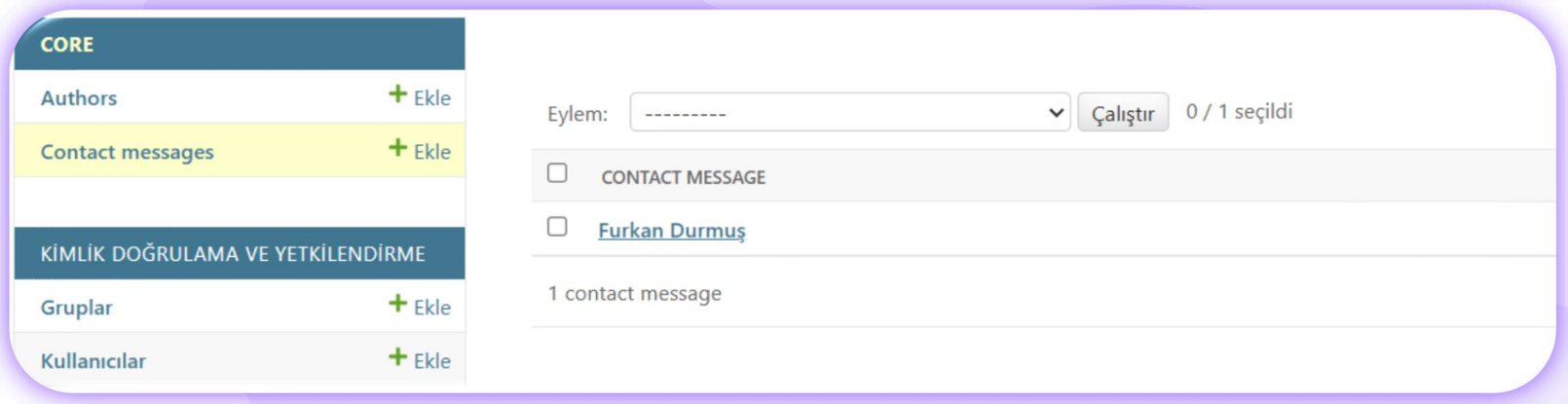
```
def iletisim(request):
    if request.method == "POST":
        ad_soyad = request.POST.get('full_name')
        email = request.POST.get('email')
        mesaj = request.POST.get('message')
        ContactMessage.objects.create(
            full_name=ad_soyad,
            email=email,
         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

core > admin.py

```
from django.contrib import admin
from .models import Author, ContactMessage

admin.site.register(Author)
admin.site.register(ContactMessage)
```

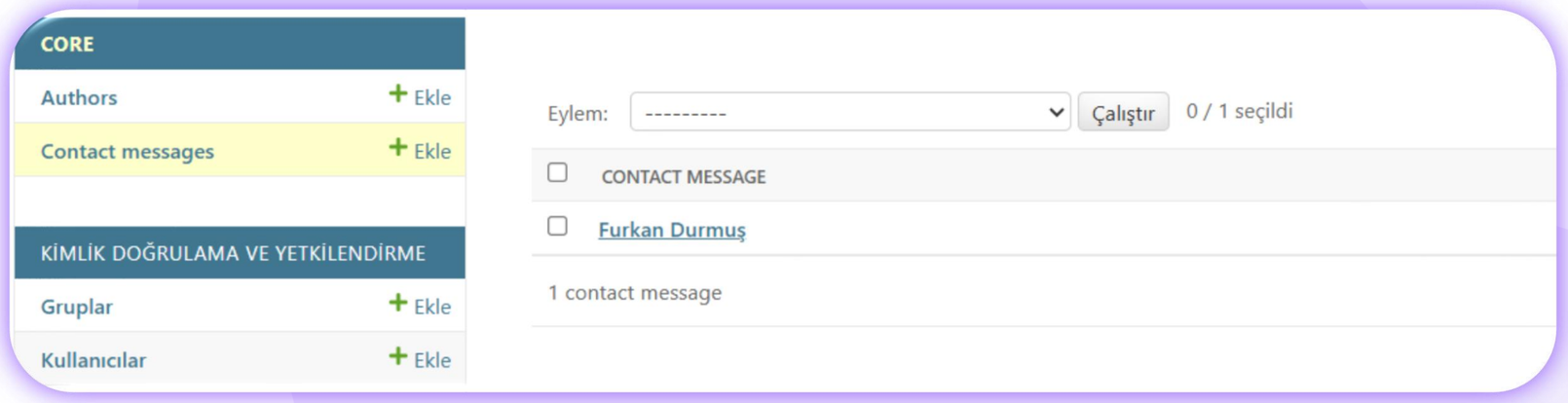
İletişim Sayfası

Projeyi çalıştırıp test edebilirsiniz.



İletişim Sayfası

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



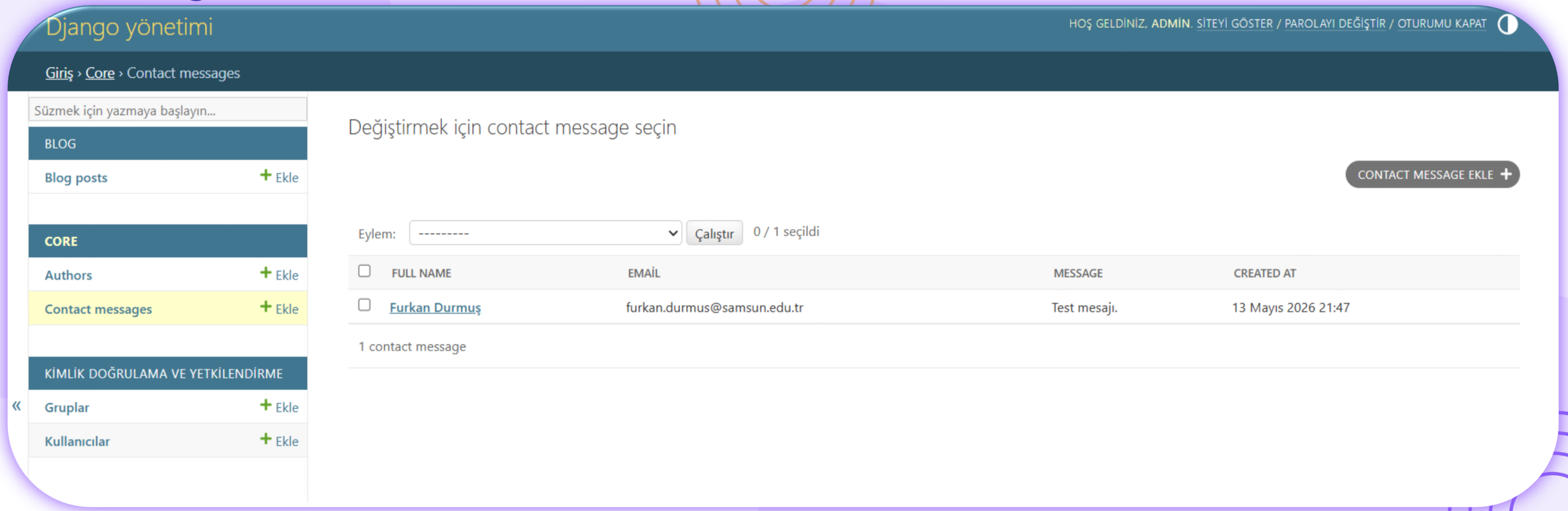
İletişim Sayfası

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

admin.py > ...

```python
from django.contrib import admin
from .models import Author, ContactMessage

admin.site.register(Author)

class ContactMessageAdmin(admin.ModelAdmin):

    list_display = (
        'full_name',
        'email',
        'message',
        'created_at'
    )


admin.site.register(ContactMessage, ContactMessageAdmin)

```

İletişim Sayfası



Sonuç

Son