

# Kişisel Web Sitesi

211307077-Arda Muhammet Ünsalan

*Kocaeli Üniversitesi  
Bilişim Sistemleri Mühendisliği*

**Abstract**—Bu proje, Django, Docker, Nginx ve Gunicorn gibi modern teknolojiler kullanılarak kişisel bir web sitesi geliştirilmesini kapsamaktadır. Amacımız, güçlü bir backend altyapısı ve ölçeklenebilir bir dağıtım yöntemi sağlayarak, kullanıcı dostu, güvenli ve dinamik bir platform oluşturmaktır.

Proje kapsamında Django, hızlı ve güvenli bir web uygulama geliştirme çerçevesi olarak seçilmiş; Docker ise bağımlılıkların yönetimi ve platform bağımsızlığı sağlamak için kullanılmıştır. Nginx, ters proxy sunucu olarak yapılandırılmış ve statik dosyaların hızlı bir şekilde sunulmasını sağlamıştır. Uygulamanın performansını artırmak ve çoklu istemci desteği sağlamak amacıyla Gunicorn kullanılmıştır.

Bu raporda, geliştirme sürecinde kullanılan araçların seçimi, uygulanan mimari, karşılaşılan sorunlar ve bunlara getirilen çözümler detaylı bir şekilde açıklanmaktadır. Proje, modern web uygulamalarında sürdürülebilirlik, taşınabilirlik ve güvenilirlik ilkelerine uygun bir çözüm sunmayı hedeflemektedir.

**Index Terms**—Django, Docker, Python, Nginx, Sanallaştırma

## I. GİRİŞ

Günümüzde kişisel web siteleri, bireylerin profesyonel kimliklerini sergilemeleri, yeteneklerini tanıtmaları ve çevrimiçi bir varlık oluşturmaları açısından büyük önem taşımaktadır. Bu tür sitelerin kullanıcı deneyimi odaklı, güvenli, hızlı ve kolayca yönetilebilir olması gerekmektedir. Bu gereksinimleri karşılamak amacıyla modern yazılım teknolojilerinden yararlanarak bir kişisel web sitesi geliştirilmiştir.

Bu projede, Django web çerçevesi kullanılarak siteye güçlü bir backend altyapısı sağlanmış, içerik yönetimi ve veri işleme süreçleri kolaylaştırılmıştır. PostgreSQL, verilerin güvenli, hızlı ve ölçeklenebilir bir şekilde saklanması sağlanmak için tercih edilmiş ve veritabanı işlemleri Docker kullanılarak konteynerize edilmiştir. Bu sayede, veritabanı yönetimi daha esnek ve taşınabilir bir hale getirilmiştir.

Docker, projenin bağımlılıklarını yönetmek ve farklı platformlarda aynı çalışma ortamını sağlamak için kullanılmıştır. Nginx, statik dosyaların hızlı bir şekilde sunulması ve güvenli bir ters proxy sunucu işlevi görmesi amacıyla yapılandırılmıştır. Performansı artırmak ve çoklu istemci desteği sunmak için Gunicorn kullanılarak WSGI uygulaması çalıştırılmıştır.

Son olarak, proje DijitalOcean platformunda yayınlanarak, bulut tabanlı bir ortamda erişilebilir hale getirilmiştir. DijitalOcean'un sunduğu altyapı ve yönetim araçları, projenin kolay bir şekilde ölçeklendirilmesine ve sürdürülebilir bir şekilde yönetilmesine olanak tanımıştır. Bu giriş bölümünde, kullanılan teknolojilerin seçilme nedenleri ve projenin genel

amacı hakkında bilgi verilmekte olup, sonraki bölümlerde geliştirme süreci ve mimari detaylar ayrıntılı olarak ele alınacaktır.

## II. PROJE BAŞLANGICI

### A. Gerekli Araçların ve Ortamın Hazırlanması

Bu bölümde, Django çerçevesinin kurulumu ve proje oluşturma süreci açıklanmaktadır. Django, hızlı geliştirme süreci ve temiz bir mimari sunması nedeniyle projede tercih edilmiştir.

İlk olarak, bağımlılıkların yönetimi ve proje dosyalarının izole edilmesi için bir Python sanal ortamı oluşturulmuştur. Bunun için Python'un kurulu olduğundan emin olmuş ve ardından python -m venv env komutu çalıştırılarak sanal ortam hazırlanmıştır. Bu ortam, Windows işletim sisteminde env, Linux ve MacOS'ta ise source env/bin/activate komutlarıyla aktif hale getirilmiştir.

Sanal ortam etkinleştirildikten sonra, pip install django komutu ile Django'nun en son sürümü yüklenmiştir. Django yüklenikten sonra, django-admin startproject myproject komutu kullanılarak bir Django projesi oluşturulmuştur. Bu komut ile proje dizini oluşturulmuş ve temel dosyalar (settings.py, urls.py, vb.) hazırlanmıştır. Bu adımlar sonucunda Django projesi başarıyla kurulmuş ve geliştirme sürecine hazır hale getirilmiştir. Projeyi oluşturduktan sonra bir sanal ortam (env) oluşturduğum ve kişisel bilgileri gitignore dosyasına ekleyerek GitHub'a göndermedim. Bu sayede, hassas verilerin korunmasını ve uygulama güvenliğini sağlamış oldum.

Ayrıca, proje için bir requirements.txt dosyası oluşturduğum ve projede kullanılan bağımlılıkların sürümlerini bu dosyada not ettim. Bu, projeyi farklı ortamlarda kolayca yeniden oluşturmayı ve bağımlılıkların doğru sürümleriyle çalışmayı mümkün kıldı.

### B. Django Projelerinin Hazırlanması ve Modelleme

Bu aşamada, projeyi daha modüler ve yönetilebilir hale getirebilmek için iki adet Django uygulaması oluşturduğum: core ve contact.

İlk olarak, core adlı uygulamayı oluşturup, bu uygulama içinde metin, ikon gibi bilgileri modellemek amacıyla gerekli modelleri hazırladım. Bu modeller, projenin genel yapısına ait temel bilgileri ve görsel öğeleri düzenlemeye olanak tanındı. Core uygulaması, proje için genel yapı taşlarını oluşturdu ve ilerleyen aşamalarda başka uygulamalarla entegrasyon sağlamak için temel bir altyapı sundu.

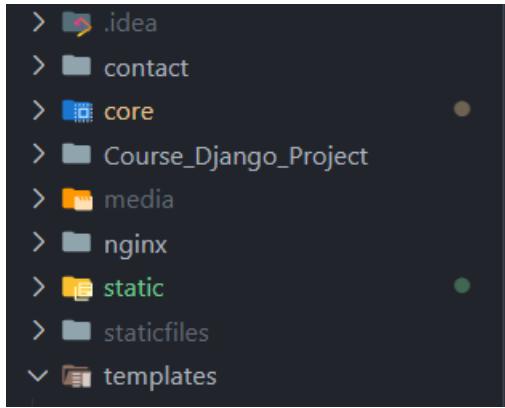


Fig. 1. Dosya Yapısı

Ardından, contact adlı başka bir Django uygulaması oluşturuldu. Bu uygulama, kullanıcılarından gelen iletişim taleplerini almak amacıyla bir Django formu oluşturmayı amaçladı. İletişim formu, kullanıcıların adı, e-posta adresi ve mesaj gibi bilgilerini alacak şekilde tasarlandı. Bu form verilerinin modellemesi, kullanıcılarından gelen taleplerin veritabanına kaydedilmesini ve yönetilmesini sağladı. Bu sayede kullanıcı geri bildirimlerini toplamak ve bu verileri organize bir şekilde saklamak mümkün oldu.

Bu iki uygulamanın entegrasyonu ve yapılandırılması, projede esneklik ve sürdürülebilirlik sağlamış oldu. Detaylı model sınıflarını burada açıklayamamış olsak da, her iki uygulamanın tasarımını ve işlevi projenin temel yapı taşlarını oluşturarak, gelecekteki geliştirmelere olanak sağlamaktadır.

### C. Proje Veritabanı Modelleme

a) *Contact Modeli* : Bu model, kullanıcılarından gelen iletişim taleplerini almak ve bu verileri veritabanında saklamak için tasarlanmıştır. Django'nun models.Model sınıfından türetilen Message adlı model, core uygulamasındaki AbstractModel sınıfını temel alır. Model içerisinde kullanıcıdan "name", "email", "subject" ve "message" içerikleri istenir.

b) *AbstractModel Model* : AbstractModel sınıfı, bir üst sınıf olarak tanımlanmıştır ve tüm diğer modeller bu sınıfından türetilmiştir. updated date ve created date gibi alanlar, her modelin tarihsel verilerini takip etmek için kullanılır. Bu alanlar otomatik olarak güncellenir veya eklenir. Meta sınıfı: abstract = True olarak ayarlanmış ve böylece bu model doğrudan veritabanına yansıtılmaz; sadece türetilen sınıflar veritabanına kaydedilir.

c) *GeneralSetting Modeli* : Bu model, GeneralSetting modeli, temel ayar bilgilerini saklamak için kullanılır. Bu modeldeki her bir örnek, bir ayar adı (name), açıklama (description) ve parametre (parameter) bilgilerini içerir. Alanlar: name: Ayar adını belirtir. description: Ayarın açıklamasını tutar. parameter: Ayar için parametreyi içerir. Meta sınıfı: Bu model, name alanına göre sıralanır ve verbose name değerleri "General Setting" ve "General Settings" olarak tanımlanır.

d) *Image Modeli* : ImageSetting modeli, görsellerle ilgili ayarları saklar. Bu modelde bir isim, açıklama ve bir

```
class Experience(AbstractModel):
    company_name = models.CharField(
        default='',
        max_length=254,
        blank=True,
        verbose_name='Company Name',
    )
    job_title = models.CharField(
        default='',
        max_length=254,
        blank=True,
        verbose_name='Job Title',
    )
    job_location = models.CharField(
        default='',
        max_length=254,
        blank=True,
        verbose_name='Job Location',
    )
    start_date = models.DateField(
        verbose_name='Start Date',
    )
    end_date = models.DateField(
        default=None,
        blank=True,
        null=True,
        verbose_name='End Date',
    )
```

Fig. 2. Experience Modeli Yapısı

dosya alanı (resim) bulunur. Alanlar: name: Görsel ayarının adını tutar. description: Görsel ayarının açıklamasını tutar. file: Görsel dosyasının yükleneceği alan. uploadto='images/' parametresiyle dosyalar belirtilen dizine kaydedilir. Meta sınıfı: Görseller, name alanına göre sıralanır ve "Image Setting" ve "Image Settings" olarak adlandırılır.

e) *Skill Modeli* : Skill modeli, kullanıcının becerilerini tanımlar ve her beceri için bir sıralama (order) ve beceri yüzdesi (percentage) belirler. Alanlar: order: Beceri sıralamasını belirtir. name: Beceri adını belirtir. percentage: Beceri yüzdesini belirtir. Yüzde değeri, MinValueValidator ve MaxValueValidator ile 1 ile 100 arasında sınırlıdır.

f) *Experience Modeli* : Experience modeli, kullanıcıya ait iş deneyimlerini saklar. Her bir iş deneyimi için şirket adı, görev unvanı, görev yeri ve başlangıç ve bitiş tarihleri içerir. Alanlar: companyname: Şirket adı. jobtitle: Görev unvanı. joblocation: Görev yerinin adı. startdate ve enddate: Deneyimin başlangıç ve bitiş tarihlerini belirtir. Meta sınıfı: Deneyimler, başlangıç tarihine (startdate) göre sıralanır

g) *Education Modeli* : Education modeli, kullanıcının eğitim geçmini içerir. Bu modelde okul adı, bölüm adı, ana dal adı ve eğitim başlangıç ve bitiş tarihleri yer alır. Alanlar: schoolname: Okul adı. major: Ana dal adı. department: Bölüm adı. startdate ve enddate: Eğitim tarihlerini belirtir. Meta sınıfı: Eğitim bilgileri, eğitim başlangıç tarihine (startdate) göre sıralanır.

h) *SocialMedia Modeli* : SocialMedia modeli, kullanıcının sosyal medya hesap bilgilerini içerir. Her bir sosyal medya kaydı, bir bağlantı ve simge (ikon) içerir. Alanlar: order: Sosyal medya bağlantısının sıralama numarasını belirtir.

```

class Education(AbstractModel):
    school_name = models.CharField(
        default='',
        max_length=254,
        blank=True,
        verbose_name='School Name',
    )
    major = models.CharField(
        default='',
        max_length=254,
        blank=True,
        verbose_name='Major',
    )
    department = models.CharField(
        default='',
        max_length=254,
        blank=True,
        verbose_name='department',
    )

    start_date = models.DateField(
        verbose_name='Start Date',
    )

    end_date = models.DateField(
        default=None,
        blank=True,
        null=True,
        verbose_name='End Date',
    )

```

Fig. 3. Education Modeli Yapısı

link: Sosyal medya platformunun URL’si. icon: Sosyal medya platformunun simgesi. Meta sınıfı: Sosyal medya bilgileri, sıralama numarasına (order) göre sıralanır.

i) *Document Modeli* : Document modeli, belgeleri saklamak için kullanılır. Bu modelde bir slug, belgeye ait bir buttoncontext (buton metni) ve belge dosyasının yerini belirten file alanları bulunur. Alanlar: order: Belgelerin sıralama numarasını belirtir. slug: URL’lerde kullanılacak kısa metin. buttoncontext: Belgeler için buton metni. file: Belge dosyasının yeri.

### III. FRONTEND TASARIMININ PROJELYE EKLENMESI

Django, genellikle backend (sunucu tarafı) geliştirme için kullanılan güçlü bir web framework’üdür. Ancak, frontend (kullanıcı arayüzü) ile entegrasyonu da oldukça kolaydır. Django’nun sağladığı özelliklerle, HTML, CSS ve JavaScript dosyalarını kullanarak projenizi görsel olarak zenginleştirebilir ve dinamik hale getirebilirsınız.

#### A. Template Dosyaların Dahil Edilmesi

Django projelerinde şablon yönetimi, TEMPLATES ayarlarıyla yapılır. Bu ayarlar, şablonların nasıl işleneceğini ve hangi dizinlerden yükleyeceğini belirtir. Yukarıda verdığınız TEMPLATES ayarları, Django’nun şablon motoru olan DjangoTemplates’i kullanarak şablonları yüklemeyi ve işlemi yapılandırır. Bu ayarları modüler ve verimli bir şekilde kullanmak için birkaç önemli noktayı gözden

a) *DIRS Ayarı* : ‘DIRS’: [BASEDIR / ‘templates’] Bu satır, şablonların bulunduğu dizinlerin listesini belirtir. Django, burada belirtilen dizinleri kontrol eder ve şablonları bu dizinlerden yükler. BASE-DIR projenizin kök dizinini

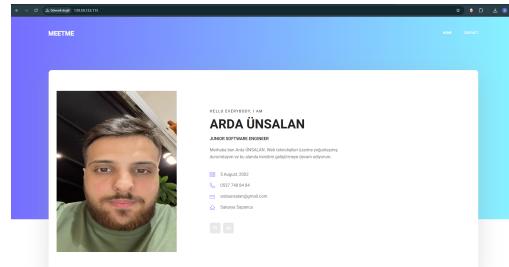


Fig. 4. Home Page Tasarımı

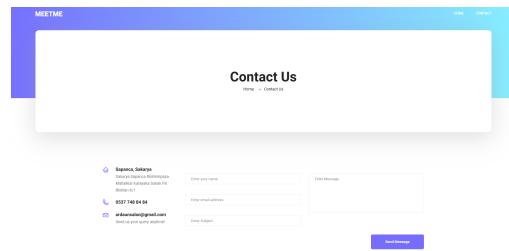


Fig. 5. Contact Page Tasarımı

temsil eder, ve burada templates adlı bir klasör kullanarak şablon dosyalarını projeye eklemişsiniz. Bu dizin yaplandırmas, şablonlar ana projede merkezi bir konumda tutarak, dier şablonlara erişimi kolaylaştırr.

b) *Backend Ayarı*: ‘BACKEND’: ‘django.template.backends.django.DjangoTemplates’ Bu satır, Django’nun şablon motorunun hangi backend ile çalışacağını belirler. Burada, Django’nun yerleşik şablon motoru DjangoTemplates kullanılıyor.

Bu TEMPLATES ayarları, Django’nun şablon motorunun şablonları nasıl yükleyeceğini, hangi dizinleri kontrol edeceğini ve hangi context verilerinin şablonlara dahil edileceğini belirler. Ayrca, modüler yapı desteklemek için, ortak yapılar (navbar, footer gibi) şablonlara dahil edebilmek adına contextprocessors kullanın büyük bir kolaylık salar. core.views.layout gibi özelleştirilmiş context processor’lar, projede kullanılacak ortak verileri şablonlara entegre etmek için oldukça kullanışlıdır.

#### B. Static Dosyaların Dahil Edilmesi

Django projelerinde, statik dosyaların (CSS, JavaScript, gorseller vb.) yönetimi, STATICFILES\_DIRS ve STATIC\_URL gibi ayarlarla yapılır. Bu dosyalar, şablonlar ve sayfalar üzerinde görsel ve işlevsel iyileştirmeler yapmak için kullanılır.

Statik dosyaların projeye dahil edilmesi için aşağıdaki adımlar takip edilebilir:

- STATICFILES\_DIRS ayarı, projenizdeki statik dosyaların bulunduğu dizinleri belirtir. Bu dizinler, Django’nun statik dosyalarını toplaması için kullanılır.
- STATIC\_URL ayarı, tarayıcıda erişilebilecek statik dosyaların URL’sini belirler. Örneğin, STATIC\_URL = ‘/static/’ olarak ayarlanabilir.

- `collectstatic` komutunu kullanarak statik dosyaları projede uygun şekilde toplamanız gereklidir.
- Geliştirme ortamında, statik dosyalar genellikle `runserver` komutıyla otomatik olarak sunulur. Ancak, üretim ortamında bunların uygun bir şekilde sunulması için Nginx veya Apache gibi sunucular kullanılabilir.

#### IV. DOCKER KULLANIMI VE DJANGO ILE ENTEGRASYONU

Django projelerinde Docker kullanımı, uygulamanın taşınamasını artırmak ve geliştirme, test etme, üretim ortamları arasında tutarlılığı sağlamak için oldukça yararlıdır. Docker, uygulamaların gerekli tüm bağımlılıkları ve yapılandırmaları ile birlikte konteynerler içinde çalışmasını sağlar. Bu sayede, farklı ortamlarda (geliştirme, test, üretim) aynı ortamda çalışmayı sağlayarak hataları minimize eder. Ayrıca, Docker sayesinde projeyi birden fazla geliştiriciye daha kolay paylaşılabilir ve CI/CD (Sürekli Entegrasyon/Sürekli Dağıtım) süreçlerini daha verimli hale getirebilirsiniz.

##### A. Django ve Docker Entegrasyonu

Django projelerinde Docker kullanmak, uygulamanın yapılandırmasını ve bağımlılıklarını kapsülleyerek daha stabil ve yönetilebilir bir ortam yaratır. Docker, özellikle veritabanı, önbellek, web sunucusu ve uygulama konteynerlerinin bir arada çalışmasını sağlayan bir yapı sunar. Django ile entegrasyonu genellikle şu bileşenlerden oluşur:

- **Veritabanı (PostgreSQL gibi):** Django projelerinde veri yönetimi için veritabanı kullanılır. Docker ile PostgreSQL gibi veritabanları, gerekli tüm bağımlılıklar ve yapılandırmalarla birlikte konteyner içerisinde çalıştırılır.
- **Uygulama Sunucusu (Gunicorn):** Django uygulaması genellikle Gunicorn gibi bir WSGI sunucusu ile çalıştırılır. Docker konteyneri içinde uygulama ve sunucu birleştir.
- **Web Sunucusu (Nginx):** Nginx, statik dosyaların sunulmasında ve uygulamanın trafiğini yönlendirmede kullanılır. Docker içinde Nginx de ayrı bir konteyner olarak çalışır.
- **Bağımlılıklar ve Yapılandırma Dosyaları:** Docker, tüm bağımlılıkların (örneğin, Python paketleri) ve yapılandırma dosyalarının (örneğin, .env dosyaları) izole bir ortamda çalışmasını sağlar.

Aşağıda, Docker Compose ve Dockerfile kullanılarak Django uygulaması için yapılandırılmış bir ortamın adımları ve nasıl çalıştığı açıklanmaktadır.

##### B. Docker Compose ve Dockerfile Yapılandırması

Docker Compose, birden fazla konteynerin aynı anda başlatılmasını ve yönetilmesini sağlayan bir araçtır. Aşağıda verilen Docker Compose konfigürasyonu, Django projesi için PostgreSQL veritabanı, Gunicorn uygulama sunucusu ve Nginx web sunucusunun nasıl çalıştığını göstermektedir.

```

# docker-compose.yml
1
2 services:
3   postgres:
4     image: postgres:latest
5     env_file:
6       - Course_Django_Project/docker.env
7     ports:
8       - "5432:5432"
9     volumes:
10      - "postgresql-data:/var/lib/postgresql/data"
11     command: -p 5432
12
13
14 app:
15   container_name: app_resume
16   hostname: app_resume
17   build:
18     context: .
19     dockerfile: Dockerfile
20   depends_on:
21     - postgres
22   env_file:
23     - Course_Django_Project/docker.env
24   volumes:
25     - ./srv/app # Tüm proje klasörünü bağla, böylece dosya değişikliklerini otomatik olarak uygula
26     - static-data:/srv/app/static # Statik dosyalar için ayrı bir volume
27     - media-data:/srv/app/media # Media dosyalar için ayrı bir volume
28   ports:
29     - "8000:8000"
30   command: gunicorn Course_Django_Project.wsgi:application --bind 0.0.0.0:8000
31
32 nginx:
33   build:
34     context: ./nginx
35     dockerfile: Dockerfile
36   restart: unless-stopped
37   depends_on:
38     - app
39   ports:
40     - "80:80"
41
42 volumes:
43   postgresql-data:
44   static-data:
45   media-data:
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
605
606
607
607
608
609
609
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448

```

```

Dockerfile
1  # Pull office base image
2  FROM python:3.10-slim
3
4  # Update and install dependencies
5  RUN apt-get update
6  RUN apt-get install libpq-dev -y
7  RUN apt-get install -y python3-dev build-essential
8  RUN apt-get install postgresql-client -y
9  RUN apt-get clean
10 RUN rm -rf /var/lib/apt/lists/*
11
12 # Set environment variables
13 ENV PYTHONUNWRITEBYTECODE 1
14 ENV VIRTUAL_ENV=/opt/venv
15
16 # Create virtual environment and upgrade pip
17 RUN pip install --upgrade pip && \
18     pip install virtualenv && \
19     python -m virtualenv $VIRTUAL_ENV
20
21 ENV PATH="$VIRTUAL_ENV/bin:$PATH"
22
23 # Add and install requirements
24 ADD ./requirements.txt /tmp/requirements.txt
25 RUN pip install -r /tmp/requirements.txt
26
27 COPY entrypoint.sh /srv/entrypoint.sh
28 RUN sed -i 's/\r$//g' /srv/entrypoint.sh
29 RUN chmod +x /srv/entrypoint.sh
30
31 # Copy the application code
32 COPY . /srv/app
33 WORKDIR /srv/app
34
35 ENTRYPOINT ["/srv/entrypoint.sh"]
36
37 # Run the Django development server (this command
38 CMD ["python", "manage.py", "runserver", "0.0.0.0:
39

```

Fig. 7. DockerFile Dosya İçeriği

*1) Dockerfile Adımları:* Aşağıdaki Dockerfile, Django projesi için bir çalışma ortamı oluşturur, bağımlılıkları yükler ve uygulamayı çalıştmak için gerekli tüm adımları içerir:

İlk olarak, Dockerfile `python:3.10-slim` imajını temel alır. Bu, Python 3.10 sürümünü içeren hafif bir Docker imajıdır.

Bağımlilikler yüklenir. Bu adımda: - PostgreSQL ile bağlantı kurabilmek için gerekli olan `libpq-dev` ve `postgresql-client` paketleri yüklenir. - Python geliştirme için `python3-dev` ve `build-essential` paketleri kurulur. - Ardından, gereksiz dosyalar temizlenir ve disk alanı kazancı sağlanır.

Python sanal ortamı oluşturulur ve `pip` güncellenir. Sanal ortam, projede bağımlılıkların izole bir şekilde kurulmasını sağlar.

Sanal ortamın bin dizini, PATH ortam değişkenine eklenir. Böylece, sanal ortamda yüklü olan Python ve `pip` komutları öncelikli olarak kullanılacaktır.

Projenin bağımlılıkları `requirements.txt` dosyasından okunarak sanal ortamda kurulur.

Bir giriş noktası (`entrypoint.sh`) dosyası kopyalanır ve çalıştırılabilir hale getirilir. Bu dosya, konteyner başlatıldığında çalıştırılacak komutları içerir.

Uygulama dosyaları konteynerin içine kopyalanır ve çalışma dizini `/srv/app` olarak belirlenir.

Konteyner başlatıldığında çalıştırılacak komut (`entrypoint.sh`) belirlenir.

Son olarak, Django geliştirme sunucusu başlatılır. Bu, konteyner başladığında çalıştırılacak komuttur. `0.0.0.0:8000` adresinde sunucu dinleyecek şekilde yapılandırılmıştır, böylece dışarıdan erişilebilir olur.

*2) Sonuç:* Bu Dockerfile, Django projesini Docker ortamında çalıştmak için gerekli tüm yapılandırmayı sağlar. Sanal ortam kullanarak bağımlılıkları izole eder, PostgreSQL ile entegrasyonu mümkün kılar ve uygulamanın başlatılmasını sağlayacak gerekli dosyaları ve komutları içerir. Bu sayede, geliştirme ve üretim ortamlarında aynı yapılandırma ve bağımlılıklar ile uygulama çalıştırılabilir.

#### D. Sonuç

Docker kullanımı, Django projelerini geliştirmek ve dağıtmak için güçlü bir araçtır. Docker Compose ve Dockerfile kullanarak, Django uygulamanızın bağımlılıkları, veritabanı ve web sunucusu gibi bileşenlerini izole bir şekilde yönetebilir ve tutarlı bir geliştirme ortamı oluşturabilirsiniz. Bu sayede uygulamanın farklı ortamlar arasında taşınabilirliği sağlanır, hatalar azaltılır ve dağıtım süreci hızlanır. .

### V. DJANGO PROJESİNIN DIGITALOCEAN'A YÜKLENMESİ

Django projemi DigitalOcean'a yüklemek için aşağıdaki adımları izledim:

#### A. Sunucuya Bağlanma

İlk olarak, DigitalOcean üzerinde oluşturduğum droplete SSH ile bağlandım. Bağlantı için terminal üzerinden aşağıdaki komutu kullandım:

```
ssh root@<your-droplet-ip>
```

#### B. Gerekli Yazılımların Kurulumu

Sunucuda Python, pip, PostgreSQL ,Gunicorn ve Nginx gibi gerekli yazılımları kurmam gerekiyordu. Bu yazılımların kurulumunu aşağıdaki adımları takip ederek gerçekleştirdim.

*1) Python ve Pip Kurulumu:* Ubuntu sunucusuna Python ve pip yüklemek için aşağıdaki komutları çalıştırıldım:

```
sudo apt update
sudo apt install python3-pip
sudo apt install python3-dev libpq-dev
```

*2) PostgreSQL Kurulumu:* Eğer PostgreSQL kullanıyorsanız, aşağıdaki komutla kurulumu yapabilirsiniz:

```
sudo apt install postgresql
sudo apt install postgresql-contrib
```

*3) Gunicorn Kurulumu:* Django uygulamamı çalıştıracak olan Gunicorn sunucusunu kurmak için şu komutu kullandım:

```
pip install gunicorn
```

4) *Nginx Kurulumu*: Django uygulamamın önünde çalışan ters proxy olarak Nginx'i kurmam gerekti. Nginx'i şu komutla kurdum:

```
sudo apt install nginx
```

#### C. Django Projesinin Sunucuya Aktarılması

Proje dosyalarımı yerel bilgisayarımdan DigitalOcean sunucusuna aktarmak için 'scp' komutunu kullandım.

#### D. Sanallaştırma Ortamı Oluşturma ve Bağımlılıkları Kurma

Proje dosyalarını sunucuya aktardıktan sonra sanal ortam (virtual environment) oluşturdum ve aktif hale getirdim:

```
python3 -m venv venv
source venv/bin/activate
```

Projemdeki bağımlılıkları yüklemek için ise şu komutu kullandım:

```
pip install -r requirements.txt
```

#### E. Gunicorn ve Nginx Konfigürasyonu

Django projemi Gunicorn ile çalıştırıldım ve Nginx'i bir ters proxy sunucusu olarak yapılandırdım.

1) *Gunicorn'u Başlatma*: Gunicorn'u başlatmak için aşağıdaki komutu kullandım:

```
gunicorn --workers 3 myCvProject.wsgi:application
```

2) *Nginx Konfigürasyonu*: Nginx yapılandırma dosyasını düzenledim ve Nginx'i, Gunicorn'a yönlendirecek şekilde ayarladım. Şu komutla Nginx yapılandırma dosyasını düzenledim:

```
sudo nano /etc/nginx/sites-available/myCvProject
```

Aşağıdaki yapılandırma satırlarını ekledim:

```
server{
    listen 80;
    server_name 139.59.133.115;
    server_tokens off;
    client_max_body_size 15M;
    location / {
        proxy_pass http://arda;
        proxy_set_header X-Forwarded-For
        $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
        proxy_redirect off;
    }
}
```

Sonrasında Nginx'i yeniden başlatmak için şu komutu kullandım:

```
sudo systemctl restart nginx
```

3) *DEBUG Modunu Kapatma*: DEBUG modunu kapatarak, üretime ortamına uygun hale getirdim (Bunu hata mesajlarını kullanıcıların görmemesi için yaptım, sitemde bir page hatası olursa 404 fırlatması için):

```
DEBUG = False
```

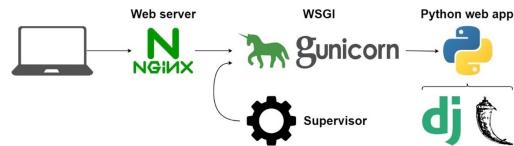


Fig. 8. Proje Çalışma Mantığı

4) *Allowed Hosts Konfigürasyonu*: Allowed Hosts listesine sunucumun IP adresini veya alan adını ekledim:

```
ALLOWED_HOSTS = ['139.59.133.115']
```

5) *Statik Dosyaların Yönetimi*: Statik dosyaların (CSS, JavaScript, görseller) doğru bir şekilde sunulabilmesi için aşağıdaki komutu çalıştırarak statik dosyaları topladım:

```
python manage.py collectstatic
```

6) *Sonuç*: Tüm bu adımları takip ettikten sonra Django projemi başarıyla DigitalOcean sunucusuna yükledim. DigitalOcean, projelerimi bulut ortamında barındırmak için oldukça verimli bir platform sundu ve projemi güvenli bir şekilde yayına almış oldum.

#### REFERENCES

- İ. Yılmaz, Python ile Programlamaya Giriş, 2. baskı, Papatya Yayınları, 2019.
- F. Korkmaz, Django ile Web Uygulama Geliştirme, Beta Yayınları, 2020.
- E. Demir, Python Programlamada İleri Seviye, Seçkin Yayıncılık, 2021.

- M. Kılıç, Django ve Python ile Full Stack Web Geliştirme, Nobel Yayıncılık, 2022.
- H. Aksoy, “Python ve Django ile Web Uygulaması Nasıl Geliştirilir?,” Kodlab, 2020. [Online]. Available: <https://www.kodlab.com/python-django-web-uygulamasi/>.
- S. Tuncer, “Django ile Web Uygulaması Geliştirme ve Dağıtım,” Yazılım Akademisi, 2021. [Online]. Available: <https://www.yazilmakademisi.com/django-ile-web-uygulamasi-gelistirme/>.
- A. Okan, “Python ile Web Programlamaya Başlangıç,” TUBITAK Yayınları, 2018.
- O. Aydın, “Django ve Nginx ile Web Uygulama Dağıtımları,” Nginx Türkiye, 2020. [Online]. Available: <https://www.nginxturkiye.com/django-ve-nginx-ile-web-uygulama-dagitimi/>.
- T. Gül, “Python ve Django İle RESTful API Geliştirme,” Akademik Yayınlar, 2021.
- F. Yıldız, “Django ve Nginx Kullanarak Web Uygulaması Yayınlama,” Teknoloji Bilişim, 2021. [Online]. Available: <https://www.teknolojibilisim.com/django-ve-nginx-ile-web-uygulamasi-yayinlama/>.
- M. Yılmaz, “Nginx Nedir ve Nasıl Kullanılır?,” Web Yazılımcı Blogu, 2020. [Online]. Available: <https://www.webyazilimci.com/nginx-nedir-ve-nasil-kullanilir/>.
- R. Güler, “Django ve Nginx ile Python Uygulaması Dağıtımları,” Code Akademi, 2019. [Online]. Available: <https://www.codeakademi.com/django-ve-nginx-ile-python-uygulamasi-dagitimi/>.
- E. Öztürk, “Python Programlamada Veritabanı Kullanımı ve Django,” Seçkin Yayınları, 2020.
- B. Özdemir, “DigitalOcean ile Django Hosting,” Dijital Teknolojiler, 2021. [Online]. Available: <https://www.dijitalteknolojiler.com/digitalocean-ile-django-hosting/>.
- S. Başar, “Django Projelerini DigitalOcean'a Yüklemek,” Yazılım Geliştirme Rehberi, 2020. [Online]. Available: <https://www.yazilimgelistirme.com/django-projelerini-digitalocean-a-yuklemek/>.
- T. Akın, “Nginx ile Yük Dengeleme ve Performans İyileştirme,” Nginx Türkiye, 2021. [Online]. Available: <https://www.nginxturkiye.com/nginx-ile-yuk-dengeleme/>.
- M. Şahin, “Python ve Django ile Uygulama Performansını Artırma Yöntemleri,” Yazılım Teknolojileri, 2019.
- K. Erdoğan, “DigitalOcean'da Django ile Web Uygulaması Yayınlama,” DigitalOcean Blog, 2021. [Online]. Available: <https://www.digitalocean.com/community/tutorials/django-web-uygulamasi-yayinlama>.
- H. Toprak, “Python ve Django ile API Geliştirme ve Yayınlama,” Python Blog, 2021. [Online]. Available: <https://www.pythonblog.com/api-gelistirme-django/>.
- M. Gök, “DigitalOcean VPS ile Django Hosting Rehberi,” DigitalOcean Türkiye, 2020. [Online]. Available: <https://www.digitalocean.com/community/tutorials/django-vps-hosting>.