

**T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

BSM 498 BİTİRME ÇALIŞMASI

**ADVENTURE NEST
YAPAY ZEKÂ DESTEKLİ
EMLAK KİRALAMA UYGULAMASI**

**B181210557 - Waasiq MASOOD
B191210078 - Mert SİĞİRCİ**

**Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Danışmanı : Öğr. Gör. Dr. Can YÜZKOLLAR**

2022-2023 Bahar Dönemi

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

ADVENTURE NEST
YAPAY ZEKÂ DESTEKLİ
EMLAK KİRALAMA UYGULAMASI

BSM 498 - BİTİRME ÇALIŞMASI

Waasiq MASOOD

Mert SİĞİRCİ

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Bu tez .. / .. / ... tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

.....
Jüri Başkanı

.....
Üye

.....
Üye

ÖNSÖZ

Emlak web siteleri uzun zamandır oldukça popüler olmuştur, seyahat eden insanlar her zaman tercihlerine göre odaları rezerve edebilmişlerdir. Ayrıca, evlerini misafirlerine sunan insanlar da kiralama yaparak kazanç sağlayabilmişlerdir, bu durum ev sahipleri ve kiracılar için karşılıklı faydalı olmuştur. Biz böyle bir web sitesi oluşturduk, burada insanlar giriş yaparak evlerini isteklerine göre rezerve edebilirler. Ayrıca evlerini de misafirlerine sunabilirler ve evlerinin fotoğraflarını çevrimiçi olarak paylaşabilirler. Evlerini sunarken, fotoğraflar çevrimiçi olarak yüklendiğinde arka planda bir derin öğrenme modülü çalışır ve odadaki nesneler otomatik olarak algılanır ve ön uca geri döndürülür.

İÇİNDEKİLER

ÖNSÖZ.....	iii
İÇİNDEKİLER.....	iv
SİMGELER VE KISALTMALAR LİSTESİ.....	vi
ŞEKİLLER LİSTESİ.....	vii
ÖZET.....	ix

BÖLÜM 1.

GİRİŞ	11
1.1. Kullanılan Teknolojiler.....	11
1.2. Donanım ve Yazılım Arayüzleri	12
1.3. Güvenilirlik, Güvenlik ve Kullanılabilirlik	12

BÖLÜM 2.

FRONTEND TASARIMI VE GELİŞTİRMESİ.....	13
2.1. Anasayfa	13
2.2. Giriş Sayfası	14
2.3. Host Adventure	14
2.4. Hosting – Alt Başlıklar.....	15
2.4.1. Birinci Bileşen.....	15
2.4.2. İkinci Bileşen.....	15
2.4.3. Üçüncü Bileşen.....	16
2.4.4. Dördüncü Bileşen.....	16
2.4.5. Beşinci Bileşen.....	17
2.4.6. Altıncı Bileşen.....	17
2.4.7. Yedinci Bileşen.....	18
2.4.8. Sekizinci Bileşen.....	18
2.5 İlan Sayfası	19
2.6 Yardım Sayfası	20

BÖLÜM 3.

DERİN ÖĞRENME MODÜLÜ	21
3.1. Giriş.....	21
3.2. Veri Seti	21
3.3. Veri Çıkarma.....	24
3.4. Algoritmalar İçin Veri Hazırlama.....	25
3.5. YOLOv5.....	26
3.6. Kaggle’da Model Eğitmek.....	28

BÖLÜM 4.

BACKEND TASARIMI VE GELİŞTİRME	32
4.1. Mimarının Tanıtımı.....	32
4.2. Core Katmanı.....	34
4.2.1. Models ve DTOs Klasörü.....	35
4.2.2. Repositories, UnitOfWorks ve Services Klasörü.....	36
4.2.3. Configuration Klasörü.....	37
4.3. Repository Katmanı.....	38
4.3.1. Repositories Klasörü.....	38
4.3.2. Seeds Klasörü.....	39
4.3.3. UnitOfWorks Klasörü.....	39
4.3.4. Configuring Klasörü.....	39
4.3.5. Migrations Klasörü.....	40
4.3.6. ApplicationDbContext Sınıfı.....	41
4.4. Service Katmanı.....	42
4.4.1. Mapping Klasörü.....	42
4.4.2. Services Klasörü.....	43
4.5 API Katmanı.....	44
4.6 Veritabanı.....	45
4.7 Uygulama Kodları.....	45

BÖLÜM 5.

SONUÇLAR VE ÖNERİLER.....	46
KAYNAKLAR.....	47
ÖZGEÇMİŞ.....	48
BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI.....	49

SİMGELER VE KISALTMALAR LİSTESİ

JS	: JavaScript
ASP.NET	: Active Server Pages Network Enabled Technologies
py	: Python File
ReactJS	: React JavaScript
JSX	: JavaScript X
TSX	: TypeScript X
TS	: TypeScript
DL	: Deep Learning
YOLO	: You Only Look Once

ŞEKİLLER LİSTESİ

Şekil 2.1	Ana Sayfa	13
Şekil 2.2	Giriş Sayfası	14
Şekil 2.3	Ev Hosting	14
Şekil 2.4.1	İkinci Bileşen	14
Şekil 2.4.2	Üçüncü Bileşen	15
Şekil 2.4.3	Dördüncü Bileşen	15
Şekil 2.4.5	Beşinci Bileşen	16
Şekil 2.4.6	Altıncı Bileşen	16
Şekil 2.4.7	Yedinci Bileşen.....	17
Şekil 2.4.8	Sekizinci Bileşen	17
Şekil 2.5	Yardım Sayfası	18
Şekil 3.2.1	Veri Seti Örnek 1,.....	22
Şekil 3.2.2	Veri Seti Örnek 2	22
Şekil 3.2.3	Veri Seti Örnek 3	23
Şekil 3.3.1	FiftyOne Yazılım	24
Şekil 3.3.2	CMD Komutu	24
Şekil 3.4	YOLO Dosya Formatı	25
Şekil 3.5.1	YOLOv5 Mantığı	27
Şekil 3.5.2	YOLOv5 Versiyonları Kıyaslaması	27
Şekil 3.6.1	Derin Öğrenme Kodu 1	28
Şekil 3.6.2	Derin Öğrenme Kodu 2	28
Şekil 3.6.3	Derin Öğrenme Kodu 3.....	29
Şekil 3.6.4	Eğitim Sonucu	29
Şekil 3.6.5	Model Testi	29
Şekil 3.6.6	Graik 1	30
Şekil 3.6.7	Grafik 2	30
Şekil 3.6.8	Grafik 3	31

Şekil 4.1.1	Mimari tanıtımı.....	28
Şekil 4.2.1	Core katmanı içeriği.....	22
Şekil 4.2.1.1	Modellerin listesi.....	23
Şekil 4.2.1.2	DTO'ların listesi.....	24
Şekil 4.2.1.3	Örnek model içeriği.....	25
Şekil 4.2.2.3	IGenericRepository içeriği.....	26
Şekil 4.2.2.4	IGenericService içeriği.....	27
Şekil 4.3.1	Repository katmanı içeriği.....	28
Şekil 4.3.1.1	Repositories sınıf listesi.....	29
Şekil 4.3.1.2	Örnek repository sınıf içeriği.....	30
Şekil 4.3.2	Örnek seed sınıf içeriği.....	31
Şekil 4.3.4	Örnek configuration sınıf içeriği.....	32
Şekil 4.3.6	AppDbContext sınıf içeriği.....	33
Şekil 4.4.1	Service katmanı sınıf içeriği.....	34
Şekil 4.4.2.1	Service klasörü listesi.....	35
Şekil 4.4.2.2	IGenericService sınıf içeriği.....	36
Şekil 4.5.1	API katmanı listesi.....	37
Şekil 4.5.2	UsersController içeriği.....	
Şekil 4.6.1	Veritabanı ilişkisi.....	

ÖZET

Anahtar Kelimeler: Derin Öğrenme, Emlak Kiralama, Web Sitesi, Görüntü İşleme.

Günümüzde, dünya genelinde giderek daha fazla insan seyahat ettiği için, insanların kendi yaşam tarzlarına göre yaşamaları ihtiyacı daha yaygın hale gelmiştir. Bazıları otel deneyimi yerine yerel yaşam tarzını deneyimlemek istiyor. Adventure Nest, evlerini kiralamak isteyen insanları belirli yerlerde konaklama arayan insanlarla buluşturan bir çevrimiçi pazar yeridir.

Ev sahipliği yapmak isteyenler için de oda içindeki nesneleri algılayan bir derin öğrenme modülü bulunduğu için ev sahiplerinin odalarındaki nesneleri manuel olarak girmelerine gerek yoktur.

BÖLÜM 1. GİRİŞ

Adventure Nest, insanların ek bir derin öğrenme modülüyle çevrimiçi deneyimlerini iyileştirebilecek bir emlak kiralama web sitesidir.

1.1. Kullanılan Teknolojiler

Adventure Nest; C#, Python ve TypeScript ekosisteminde geliştirilen web tabanlı bir uygulamadır. Kullanılan teknolojiler aşağıdaki gibidir:

- Frontend:
 - React JS
 - Tailwind CSS
 - Typescript
 - Prettier + ESLint
- Backend:
 - .NET Core
 - Flask
 - Cloudinary
- Veritabanı:
 - MS SQL Server
- Derin Öğrenme:
 - PyTorch
 - Test edilen algoritmalar: Yolov5M[1], Yolov5L[2]
 - Kaggle
 - OpenImages, COCO Dataset
 - OID v4 Toolkit

İstenen sonuçları elde etmek için ReactJS'teki birkaç kütüphane kullanıldı. API'leri test etmek için Postman ve Swagger UI kullanıldı. Derin öğrenme kısmı için, nesnelerin veri setleri OpenImages'tan OID v4 aracı kullanılarak çıkarıldı, eğitim Kaggle ve Google Colab'da çevrimiçi olarak gerçekleştirildi. Ağırlıklar ve Biaslar aracı model analizi için kullanıldı.

1.2. Donanım ve Yazılım Arayüzler

Proje herhangi bir donanım arabirimi içermez, kullanıcının uygulamaya erişmesi için internet erişimine sahip bir tarayıcı gereklidir. Yazılım arayüzleri şunlardır:

- ➔ Name: React JavaScript
Mnemonic: React JS
Version Number: React – 18
Source: <https://reactjs.org/>
- ➔ Name: Flask
Mnemonic: Node
Version Number: v 2.2.3
Source: <https://pypi.org/project/Flask/>
- ➔ Name: NET Core
Mnemonic: Express JS
Version Number: v6
Source: <https://learn.microsoft.com/en-us/dotnet/>
- ➔ Name: MS SQL Server
Mnemonic: MongoDB
Version Number: 2019
Source: <https://www.microsoft.com/en-us/sql-server/>

1.3. Güvenilirlik, Güvenlik ve Kullanılabilirlik

Uygulama her zaman kullanılabilir olmalıdır, arka plandaki kullanılabilirlik DB bağlantısına bağlıdır.

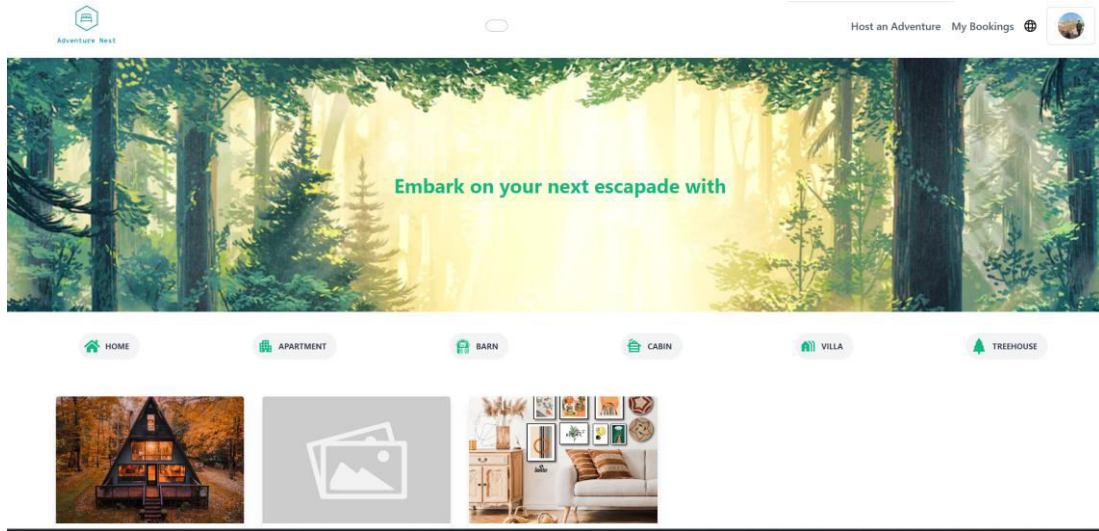
Kullanıcının e-posta, şifre ve kişisel bilgilerini girdiği için uygulama güvenli olmalıdır. Kişisel görüntüler de güvenli bir şekilde depolanmalıdır.

BÖLÜM 2. ARAYÜZ TASARIMI VE GELİŞTİRME

Proje, önyüzün tasarlanması ve sıfırdan geliştirilmesiyle başlatıldı.

2.1. Anasayfa

Uygulamanın anasayfası aşağıdaki gibidir; bir banner ve navbar ile liste görüntülenir. Şu anda yer tutucu bir tarih seçici ve sayfa bulunmaktadır. Ayrıca, üst simge anasayfaya geri dönerken, Host an Adventure'a tıkladığında hosting sayfasına yönlendirir. Kullanıcı simgesine tıklayarak Google oturumu açılabilir veya JWT authentication ile siteye giriş yapabilir.

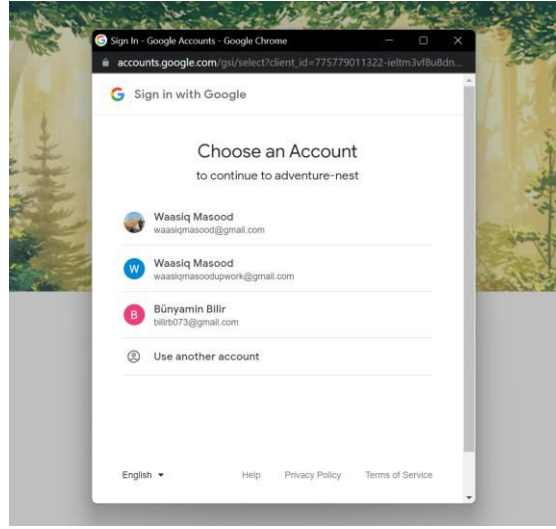


Şekil 2.1. Anasayfa

Alt Navigasyon backend'den alınan ikonlar ve alt kategorileri içerir. Şu anda uygulanmamıştır. Uygulandığında, alt kategoriler için kartlar kullanıcıda gösterilecektir. Şu anda mülkler için 6 kategori bulunmaktadır:

- 1) Home
- 2) Apartment
- 3) Barn
- 4) Cabin
- 5) Villa
- 6) Treehouse

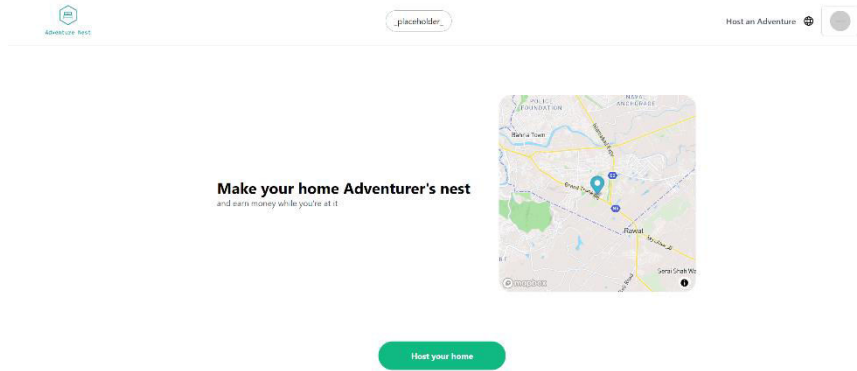
2.2. Giriş Sayfası



Şekil 2.2. Giriş Sayfası

2.3. Host Adventure

Hosting macerası sayfası önce kullanıcının konumunu gösteren bir sayfayla başlar ve Airbnb benzeri bir tasarıma sahiptir. Evini barındır düğmesine tıkladığında, kullanıcı başka bir sayfaya yönlendirilir.



Şekil 2.3. Ev host etme

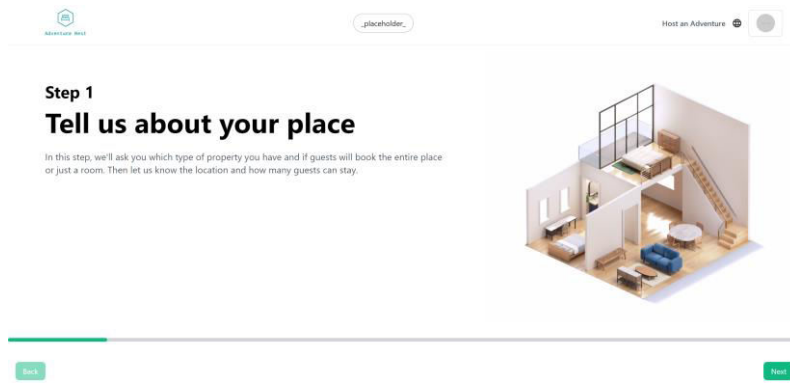
Projede bir ışıık tasarım modu kullanıldı. Navbar bileşeni, bir logo ile birlikte ev düğmesi, bildirim düğmesi, isim ve açılır menü çubuğunu içeriyordu. Harita bileşeni için Mapbox GL kullanıldı.

2.4. Hosting-Alt Başlıklar

Mevcut uygulamanın önemli bir parçası olan barındırma kategorisi, derin öğrenme modülünün entegre edildiği bir kategoridir. Barındırma alt kategorileri ve ekran görüntüleri aşağıda açıklanmaktadır.

2.4.1. Birinci Bileşen

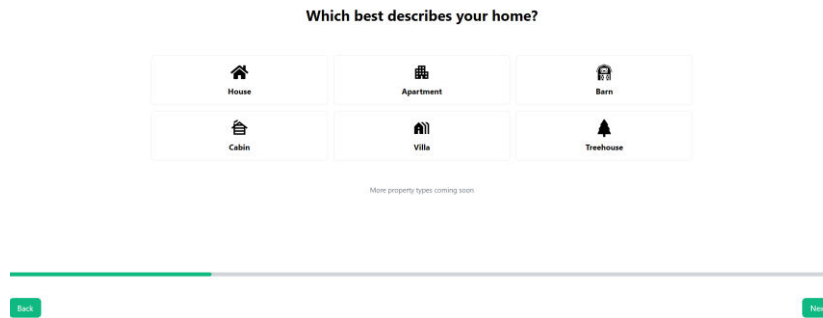
İlk bileşen pek bir şey içermiyor ve sadece bir giriş ekranıdır:



Şekil 2.4.1. Birinci bileşen

2.4.2. İkinci Bileşen

İkinci bileşen, kullanıcının ev tercihleri hakkında bilgi isteyen bir sayfadır.



Şekil 2.4.2. İkinci bileşen

2.4.3. Üçüncü Bileşen

Üçüncü bileşen, misafirlerin hangi tür yerde kalacaklarını sorduğu bir bileşendir.

Şekil 2.4.3. Üçüncü bileşen

2.4.4. Dördüncü Bileşen

Dördüncü bileşen, kullanıcının adresini sormaktadır.

Şekil 2.4.4. Dördüncü bileşen

2.4.5. Beşinci Bileşen

Beşinci bileşen, kullanıcıların evlerinde kaç misafir, yatak odası ve yatak olduğu konusunda bilgi istemektedir.

Share some basics about your place

Guests

- 1 +

Bedrooms

- 1 +

Beds

- 1 +

Bathrooms

- 1 +

Back

Next

Şekil 2.4.5. Beşinci bileşen


2.4.6. Altıncı Bileşen

Bir sonraki adım için yer tutucu bileşenidir.

Step 2

Make your place stand out

In this step, you'll add 5 or more photos. Then, you'll create a title and description.



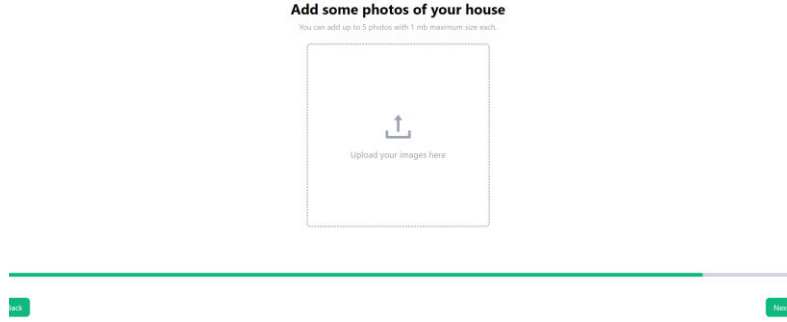
Back

Next

Şekil 2.4.6. Altıncı bileşen

2.4.7. Yedinci Bileşen

Bu bileşen, backend'den gelen yanıtın kullanıcıya geri gönderileceği bir bileşendir. Görsel olarak henüz uygulanmamıştır.



Şekil 2.4.7. Yedinci bileşen

2.4.8. Sekizinci Bileşen

Bu bileşen kullanıcının ev sahipliği için yaptığı başvurunun durumunu gösterir. Arka planda çalışan bir işlem olduğu için henüz görsel olarak uygulanmamıştır.

Final Step

AI is detecting the amenities in your house. Meanwhile, can you fill in this information?

The name you would like to put on your listing:

A cheesy description:

Country:

City:

The detected objects in your house are:

Back

Next

Şekil 2.4.8. Sekizinci bileşen


2.5. İlan Sayfası

Kullanıcı listeye tıkladığında, onunla ilgili bilgi gösteren bir sayfa da kullanıcıya gösterilecektir. Şu anda yer tutucu veriler gösteriliyor. Daha sonra bağlantı kurulduğunda, veriler arka uçtan alınarak bu sayfaya yüklenecektir.

Samujana Twenty-Four
Koh Samui, Thailand
4 guests · 2 bedrooms · 2 beds · 2 baths



Hosted by Waasiq

 Dedicated workspace

₺9,499
per night

4.9 ★

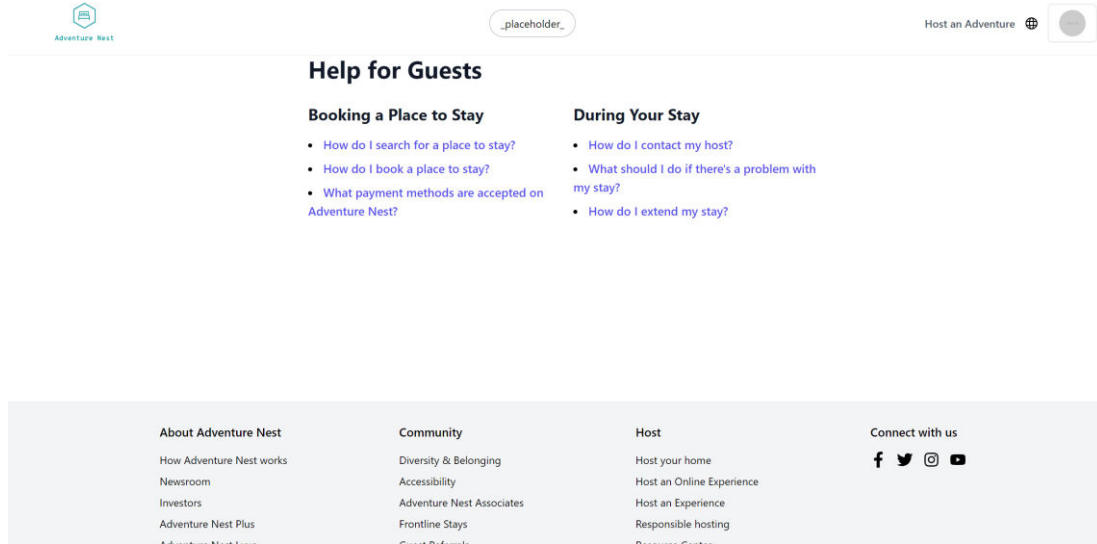
15/04/2023

15/04/2023

Şekil 2.5. İlan sayfası

2.6. Yardım Sayfası

Yardım sayfası çok fazla işlevselliğe sahip değildir ve aşağıdaki gibidir.



Şekil 2.6. Yardım sayfası

BÖLÜM 3. DERİN ÖĞRENME MODÜLÜ

Pytorch kütüphanesi kullanılarak derin öğrenme modülü uygulandı. Uygulama süreci çok uzun sürmese de, veri kümesi bulmak ve ihtiyaçlarımıza göre yeniden düzenlemek ana zorlu görevlerdi.

3.1. Giriş

Görevimiz, odadaki nesnelerin resimlerini tespit etmek; bunlar günlük hayatta kullanılan nesneler olabilir, kanepe, buzdolabı, mikrodalga fırın, televizyon gibi şeyler bizim algoritmamız tarafından tespit edilmelidir.

3.2. Veri Seti

Bu görevde veri seti elde etmek ana engeldi, şu veri setleri araştırmak için kısa listeye alındılar ve bizim verimizde kullanışlı olup olmayacaklarını görmek için incelendi:

- 1) ADE20K
- 2) COCO
- 3) OpenImages
- 4) SUNRGBD

İlk olarak, tüm bu veri setleri yerel olarak indirildi veya çevrimiçi olarak görselleştirildi, aşağıdaki örnekler her bir veri setinden nesneleri göstermektedir.

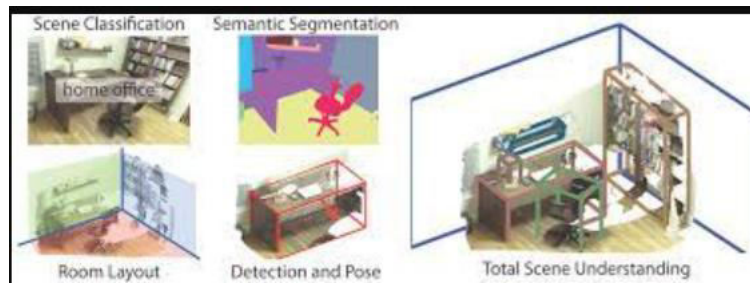


Şekil 3.2.1. Veri seti örnek 1

Bu görüntüler ayrılmış görüntüleri de içerir ve etiketler ayrılmış görüntülerdir. Bu veri kümesi için Mask RCNN kullanılması gerekiyor, ancak bu eski bir yöntemdir ve ayrıca görüntüler daha önceden bölümlere ayrılması gerekiyor.

COCO veri kümesi, genellikle önceden eğitilmiş birçok algoritmanın kullanıldığı genel bir veri kümesidir.

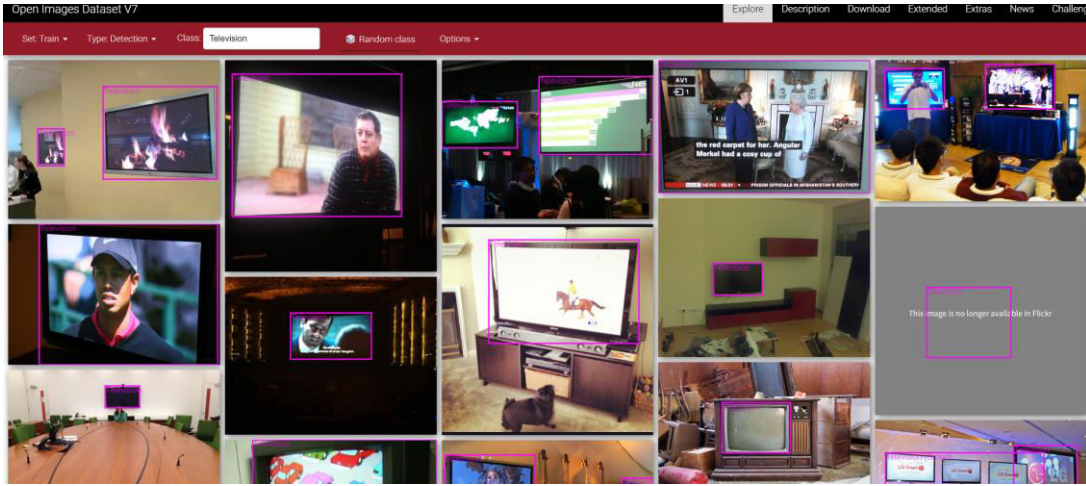
SUNRGBD veri kümesi, sahnelerin derinlik fotoğraflarını içerir ve görüntülerimiz derinlik içermediği için kullanılması gereksizdir.



Şekil 3.2.2. Veri seti örnek 2

Seçilen veri kümesi OpenImages v6 idi. Open Images V6, nesne tespiti, görsel ilişki tanıma ve görüntü sınıflandırması için derin öğrenme modellerinin eğitimi için tasarlanmış büyük ölçekli bir etiketli görüntü veri kümesidir. 1,7 milyondan fazla

görüntü içerir, her biri nesne sınırlayıcı kutuları, nesne segmentasyon maskeleri, bounding box ve nesne sınıflandırmalarıyla etiketlenmiştir. Veri kümesi Google tarafından 2020 yılında Open Images projesinin bir parçası olarak yayınlanmıştır. Kümede 600 sınıf bulunmaktadır, aşağıdaki örnek bounding box ile birlikte verilmiştir.

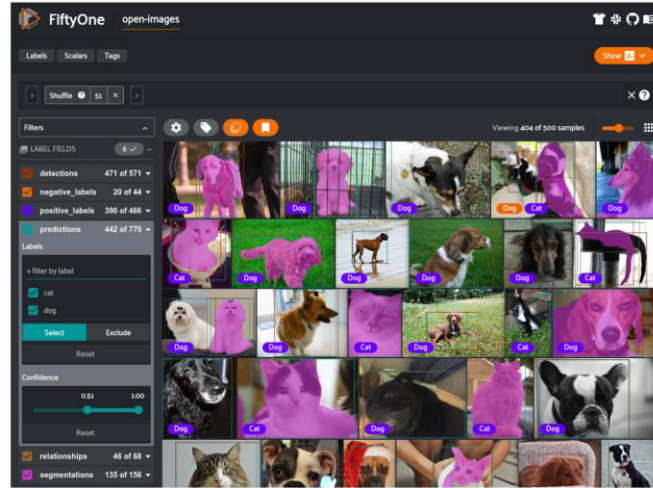


Şekil 3.2.3. Veri seti örnek 3

Bu veri kümesiyle ilgili sorun, verilerin kullanımımız için çıkarılmasının gerçekten zor olmasıydı. Tüm veri kümesi 600 GB'den fazlaydı ve indirilemedi, bu nedenle verileri indirmek için başka bir yol bulunması gerekiyordu.

3.3. Veri Çıkarma

Veri çıkarımı için ilk yaklaşım, Fifty-one adlı bir yazılım kullanılarak yapıldı. Fifty-one, özel veri kümelerini indirmek, görselleştirmek ve indirmek için kullanılan bir yazılımdır.



Şekil 3.3.1. FiftyOne yazılımı

Ancak arayüz istediğim şekilde çalışmıyordu. OIDv4 toolkit adlı başka bir araç bulundu (https://github.com/EscVM/OIDv4_ToolKit), bu yazılım CMD'den çalışır ve Open Images v4 + v6'yı kullanarak görüntüleri indirir. Aşağıda gösterildiği gibi ilk önce beş görüntü kategorisi aşağıdaki gibi CMD kullanılarak indirildi.

```
PS E:\Bitirme\ai\code\OIDv4_ToolKit-master> python main.py downloader --classes 'Washing machine Home appliance Couch Refrigerator Television' --type_csv all --limit 500 --multiclass 1
```

Şekil 3.3.2. CMD komutu

Seçilen sınıflar:

- 1) Couch
- 2) Refrigerator
- 3) Bed
- 4) Washing machine
- 5) Television
- 6) Gas Stove
- 7) Mixing Bowl
- 8) Blender
- 9) Coffeemaker
- 10) Microwave

500 resim sınırı bu resimler için seçildi. Open Images v4 veri kümesinde etiketler aşağıdaki formatta yer alır:

LabelName, Confidence, XMin, YMin, XMax, YMax, IsOccluded, IsTruncated, IsGroupOf, IsDepiction, IsInside

The labels were first tested and found to be correct.

3.4. Algoritmalar İçin Veri Hazırlama

Etiketler, YOLO[1] algoritması için dönüştürülmek üzere "convert_annotation i" adlı özel bir script kullanıldı. Etiket doğru formata dönüştürüldükten sonra, belirli bir dosya formatı takip edildi, aşağıdaki gibidir.

name	Date Modified	Type	Size
test	4/7/2023 10:12 AM	File folder	
train	4/7/2023 10:13 AM	File folder	
validation	4/7/2023 10:14 AM	File folder	
weights	4/7/2023 10:20 AM	File folder	
dataset.yaml	4/7/2023 10:20 AM	Yaml Source File	1 KB
yolov5.yaml	4/7/2023 10:20 AM	Yaml Source File	2 KB

Şekil 3.4. YOLO dosya formatı

Test, eğitim ve doğrulama kümeleri, "images" alt klasöründeki görüntüleri ve "labels" alt klasöründeki etiketleri içerir. "Weights" klasörü önceden eğitilmiş ağırlıkları içerir, "yolov5.yaml" dosyası YOLO[2] yapılandırma dosyalarını içerir, "dataset.yaml" dosyası ise özel veri kümemizin yapılandırmasını içerir.

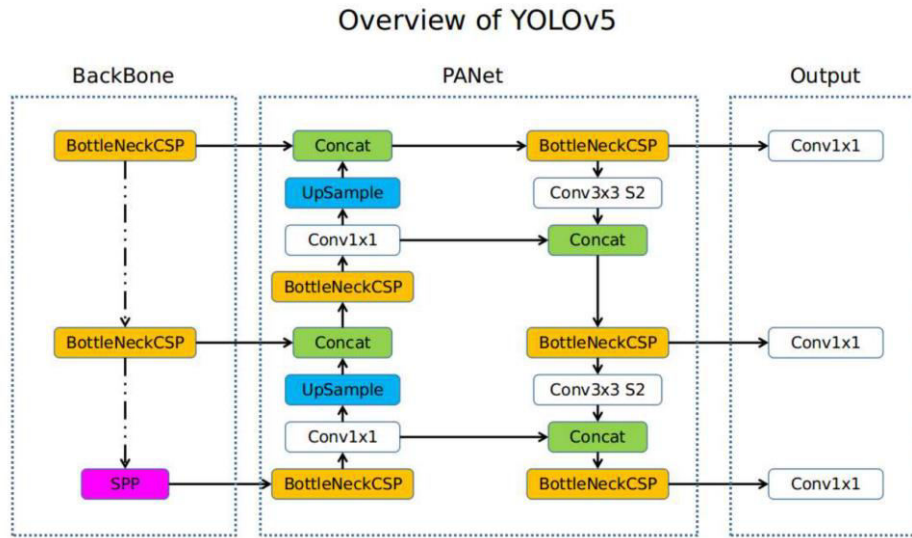
Veri seti hazırlandıktan sonra online eğitimlerde kullanılmak üzere Kaggle'a yüklenmiştir.

3.5. YOLOv5

YOLOv5[1], geliştirilmiş bir nesne tespit algoritmasıdır. Ultralytics tarafından geliştirilen YOLO (You Only Look Once) ailesi nesne tespit modellerinin geliştirilmiş bir sürümüdür ve PyTorch altyapısı üzerine inşa edilmiştir. YOLOv5[2], çeşitli nesne tespit ölçütlerinde en son performansı elde etmek için yeteneklidir ve öncülerine göre daha hızlı ve verimlidir.

YOLOv5[1], girdiği görüntüyü hücrelerin bir karesi olarak bölerek, her hücre için sınırlayıcı kutuları ve sınıf olasılıklarını tahmin ederek çalışan bir nesne tespit algoritmasıdır. Bu algoritma, PyTorch çatısı üzerine inşa edilmiş, YOLO (You Only Look Once) ailesi nesne tespit modellerinin geliştirilmiş bir sürümüdür. YOLOv5[2], çeşitli nesne tespit benchmarklarında üstün performans gösterme yeteneğine sahiptir ve öncülerine göre daha hızlı ve verimlidir.

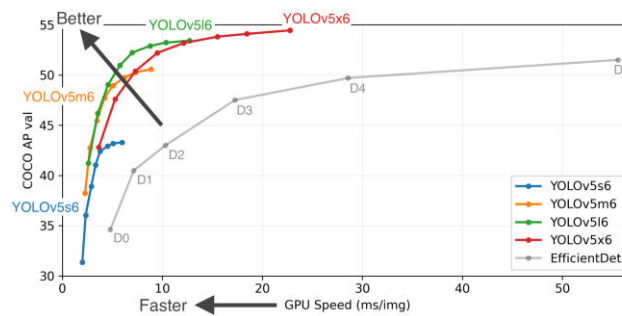
Algoritma, özellikleri giriş görüntüsünden çıkaran bir arka uç mimarisi ile başlayan, ardından sınırlayıcı kutular, nesne olasılık skorları ve sınıf olasılıkları için tahminler yapan bir dizi tespit başlığı ile devam eden bir evrimsel sinir ağı (CNN) kullanır. Algoritma, genelleştirme kabiliyetini artırmak için etiketli verilerin yanı sıra veri artırma tekniklerinin bir kombinasyonunu kullanarak uçtan uca eğitilir. Genel olarak, YOLOv5 hızlı ve doğru olacak şekilde tasarlanmıştır ve çeşitli donanım platformlarında çalışmasına izin veren küçük bir bellek ayak izine sahiptir.



Şekil 3.5.1. YOLOv5 mantığı

YOLOv5 modellerinin dört ana türü vardır:

- 1) YOLOv5s/2/: Daha az parametre ve daha düşük doğruluk ancak daha hızlı çıkarım hızı olan küçük bir versiyon.
- 2) YOLOv5m/2/: Orta sayıda parametreye sahip, hız ve doğruluk arasında iyi bir denge sağlayan orta boyutlu bir versiyon.
- 3) YOLOv5l/2/: Daha fazla parametreye ve daha iyi doğruluğa sahip büyük bir versiyon, ancak daha yavaş çıkarım hızı.
- 4) YOLOv5x/2/: En yüksek sayıda parametreye ve en yüksek doğruluğa sahip ekstra büyük bir versiyon, ancak en yavaş çıkarım hızı.



Şekil 3.5.1.1 YOLOv5 versiyonları kıyaslaması

3.6. Kaggle'da Model Eğitmek

Kaggle, veri bilimi yarışmaları için bir platformdur ve veri bilimcilerin birbirleriyle çalışarak yarışabilecekleri çeşitli makine öğrenimi zorlukları ve veri kümeleri sunar. Ayrıca topluluk üyeleri arasında veri keşfi, analizi ve işbirliği için araçlar ve kaynaklar sağlar. Kaggle 2017 yılında Google tarafından satın alındı. Ayrıca, modellerimizi çevrimiçi olarak eğitmek için kullanabileceğimiz defterler ve GPU'lar gibi araçlar sunar. Derin öğrenme için Python'da aşağıdaki kütüphaneler kullanıldı:

- 1) Numpy
- 2) Pandas
- 3) cv2
- 4) matplotlib

Derin öğrenme kısmı için kod aşağıdaki gibidir:

```
!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
!pip install -gr requirements.txt # install

import torch
from yolov5 import utils
display = utils.notebook_init() # checks
```

Şekil 3.6.1. Derin öğrenme kodu 1



Şekil 3.6.2. Derin öğrenme kodu 2

```
python train.py --img 640 --batch 16 --epochs 100 --data
{custom_cfg_vam1} --weights {custom_weight} --cfg {yolo_cfg_vam1} -
-cache --project {project} --name {name}
```

Şekil 3.6.3. Derin öğrenme kodu 3

100 epoch için eğittikten sonra sonuç:

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size	
	Class	Images	Instances	P	R	mAP50	
	all	269	346	0.816	0.755	0.798	0.642
	Blender	269	10	0.856	0.8	0.962	0.778
	Coffemaker	269	18	0.861	0.889	0.936	0.783
	Microwave oven	269	20	0.937	0.739	0.724	0.56
	Gas stove	269	17	0.5	0.471	0.448	0.217
	Mixing bowl	269	25	0.899	0.713	0.871	0.783
	Couch	269	44	0.718	0.58	0.592	0.453
	Refrigerator	269	20	0.848	0.95	0.94	0.8
	Washing machine	269	39	0.686	0.842	0.745	0.659
	Bed	269	114	0.945	0.747	0.881	0.639
	Television	269	39	0.914	0.817	0.879	0.747

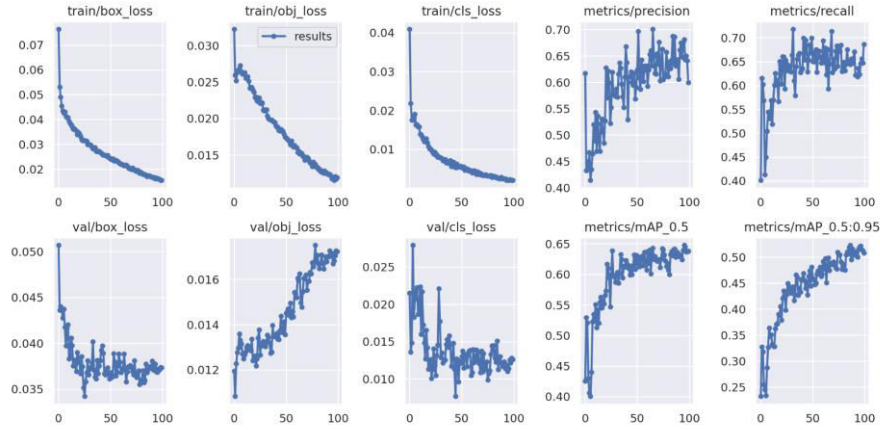
Şekil 3.6.4. Eğitim sonucu

Burada görüldüğü gibi, genel olarak mAP düşük ve ev aletleri en düşük mAP'ye sahipti, daha fazla inceleme yapıldığında Ev Aletleri sınıfı çeşitli farklı görüntüler içeriyor ve bu nedenle makine bunlar hakkında net bir şekilde öğrenemedi.



Şekil 3.6.5. Model testi

Analytic graphs:



Şekil 3.6.6. Grafik 1

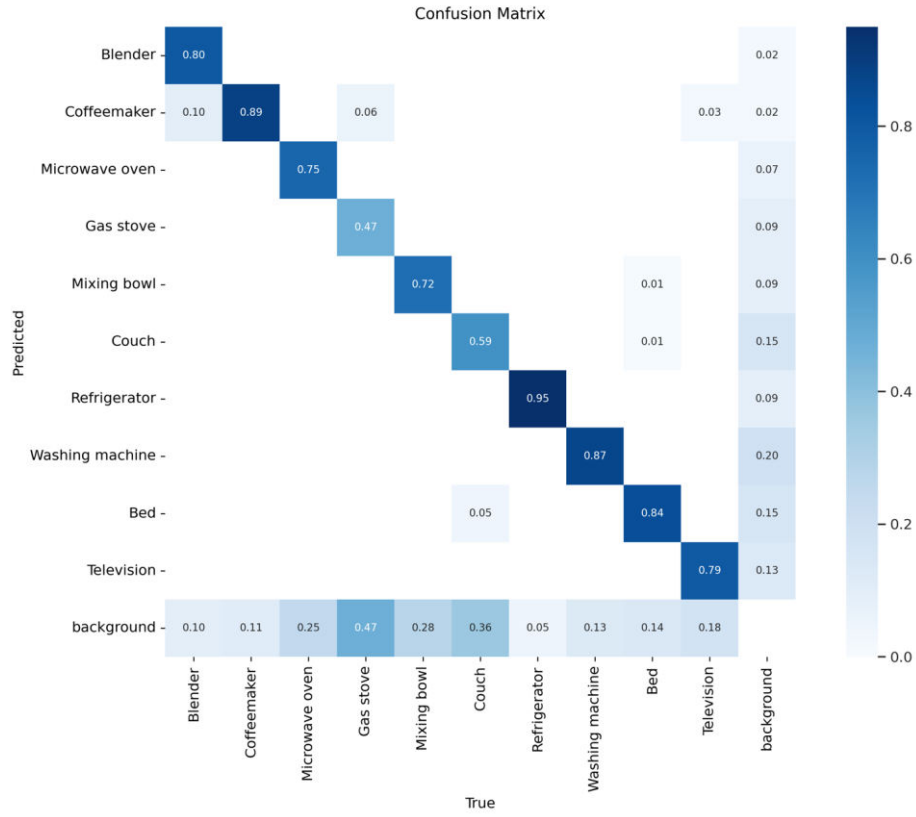


Şekil 3.6.7. Grafik 2

Analizin ardından, ev aletlerinin değiştirilmesinin en iyi çözüm olduğu ve ayrıca görüntü kalitelerinin artırılması ve YOLOv5L kullanılmasının en iyi çözüm olabileceği görülmektedir. İkinci çalışmada kullanılan sınıflar aşağıdaki gibidir.

- 1) Couch
- 2) Refrigerator
- 3) Bed
- 4) Washing machine
- 5) Television
- 6) Gas stove
- 7) Mixing bowl
- 8) Blender
- 9) Coffee maker
- 10) Microwave

Aynı kod çalıştırıldı ve aşağıdaki sonuçlar elde edildi:

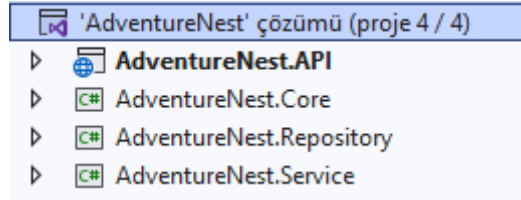


Şekil 3.6.8. Grafik 3

Bu sürüm için algoritmanın sonuçları umut verici görünmektedir.

BÖLÜM 4. BACKEND TASARIMI VE GELİŞTİRME

4.1. Mimarinin Tanıtımı



Şekil 4.1. Mimari tanıtımı

Adventure Nest projesi çok katmanlı olarak REST mimaride geliştirilmiştir. REST(Representational State Transfer), 2000 yılında Roy Fielding tarafından doktora tezinde tanıtılmış ve tanımlanmıştır.REST, dağıtık sistemler tasarlamak için kullanılan bir mimari tarzıdır. REST, istemci-sunucu arasındaki haberleşmeyi sağlayan HTTP protokolü üzerinden çalışan bir mimaridir.

REST ,servis yönelimli mimari üzerine oluşturulan yazılımlarda kullanılan bir transfer yöntemidir. İstemci ve sunucu arasında XML ve JSON verilerini taşıyarak uygulamanın haberleşmesini sağlar. REST mimarisini kullanan servislere ise RESTful servis denir. HTTP protokolü demişken HTTP, (Hyper Text Transfer Protocol) Köprü metin aktarım protokolü anlamına gelmekle beraber dünyada en yaygın kullanılan protokoldür ve internetteki web sayfalarını görüntülemek için kullanılan bir protokoldür. [3]

HTTP protokolünün GET, PUT, POST, DELETE başta olmak üzere 39 tane metodu vardır. Uygulamada bu 4 HTTP metodu kullanıldı.

- 1) GET metodu, veri listeleme ,veri görüntülemek için kullanılır. GET talepler güvenli ve aynı parametrelerle kaç kez tekrar ettiğine bakılmaksızın sonuçlar aynı olmalıdır.
- 2) POST metodu, Veri eklemek için kullanılır ancak mevcut olan bir veriyi güncellemek için de kullanılabilir.
- 3) PUT metodu, Veri güncellemek için kullanılır.
- 4) DELETE metodu, Kaynaktan veriyi silmek için kullanılır.

RESTful servis basit olmalarının yanında oldukça da esnek ve yeteneklidir. Aslında tipik Web Servislerle yapabilen her şey RESTful servislerle yapılabilir. Ayrıca mimari olarak nasıl olması, ne gibi özelliklere sahip olması hakkında belli yönergeler olsa da SOAP gibi keskin standartları olan bir mimari değildir. Üzerine çoğu

platformda (C#,JAVA vs.), bir sürü framework yazılmıştır durumda, fakat birçok platformun standart kütüphaneleri kullanılarak, hızlıca RESTful Servis geliştirebilir.

Rest servisler ;

- 1) Platform bağımsızlar.
- 2) Dil bağımsızlar.
- 3) HTTP üzerinden çalışırlar.
- 4) Esnekler ve çok kolay genişletilebilirler.

Ayrıca belli kısıtları da var bunlara kısıttan daha çok REST mimarisinin hangi sınırlar içerisinde yer almasını belirleyen prensipler diyebiliriz. Bunlardan bazıları;

Client-Sunucu Mimarisi, burada anlatılmak istenilen aslında Separation of Concerns prensibi. İstemcinin sunucu tarafındaki veri kaynağı hakkında hiç birşey bilmemesi, sunucunun da doğru istekler geldiği sürece doğru yanıt vermesi. İstemci ile sunucunun birbirlerinden bağımsız olması. Amaç aslında platform bağımsız çalışmayı ve scalability’i arttırmak. Ayrıca aralarındaki interface ortak kaldığı sürece birbirlerinden bağımsız bir şekilde gelişmeleri de sağlanmış oluyor.

Stateless, sunucu tarafında istemci ile ilgili bir context veya session tutulmaz. İstemci tarafından yapılan her istek sunucunun cevap verebilmesi için gerekli bilgiyi taşır yani her türlü state istemci tarafında tutulur, ihtiyaç duyulursa istek içerisinde sunucuya bildirilir. Bu scalability açısından da önemlidir, çünkü sunucunun istekler arasında herhangi bir state’i saklamasını gereksiz kılar ve kaynak yönetimini kolaylaştırır. Visibility açısından önemlidir, çünkü isteğin amacını anlamak için tek bir isteğin içerdiği bilgiler yeterlidir. Tabi stateless olmasının bazı dezavantajları da var. İstemci her istekte gerekli bilgileri eklemek zorundadır bu da network trafiğini artırır. Bu ayrıca sunucunun uygulamanın davranışlarındaki tutarlılığı kontrol etmesini zorlaştırır, çünkü birçok farklı istemciden farklı içerikli istekler gelebilir, sunucuya validasyon açısından daha fazla yük binebilir.

Cacheable, HTTP cevapları istemci tarafından “cache”lenebilir, o yüzden sunucu gönderdiği cevapların cacheable olup olmadığını belirtmek durumundadır, bu performans açısından önemlidir.

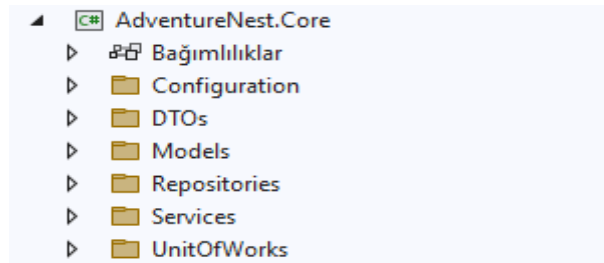
Uniform Interface, istemciler-sunucular arasında ortak, tek biçimli arayüzlerin olması REST’in en önemli prensiplerinden biridir. Bu hem iletişim yöntemini basitleştiriyor hem de ortak bir interface olması sayesinde her parçanın birbirinden bağımsız bir şekilde evrimleşmesine olanak sağlıyor. Bu konu daha önce bahsettiğim HTTP metotları ile de alakalıdır.

Layered System, burada kastedilen aslında istemcinin son sunucuya mı yoksa bir aracı sunucuya mı bağlandığını bilmiyor olması, yani her katman aslında tek bir katmanı biliyor. Bu tür bir yapıya olanak sağlamasıyla birlikte, aracı sunucular load-balancing yaparak scalability arttırabiliyor ve istemcileri belli security policy'lerine zorlayabiliyorlar. Bu yapı ayrıca encapsulation yapılması gereken yerlerde kullanılabilirler.

Code on Demand, sunucu belli durumlarda istemci tarafındaki fonksiyonalityi arttırmak veya değiştirmek için, istemci tarafına executable scriptler gönderebilir. Böyle bir kullanım bazı durumlarda Visibility'i düşürdüğü için, Code on Demand tek opsiyonel kısıttır.[4]

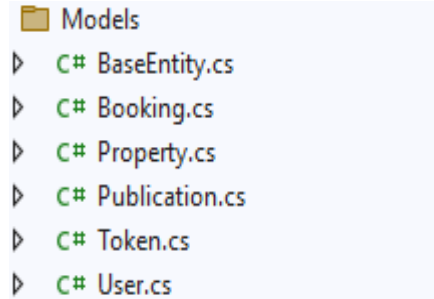
4.2. Core Katmanı

Projenin Core katmanında uygulamanın diğer katmanlarında kullanılacak temel sınıflar ve arayüzler yazılmıştır.

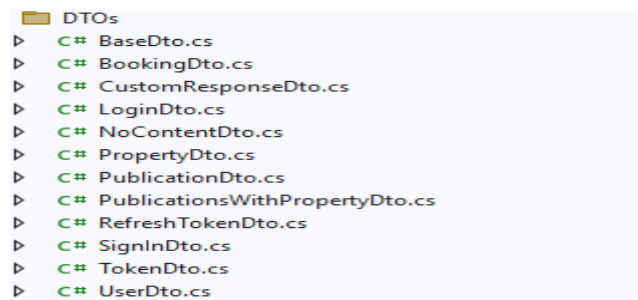


Şekil 4.2.1. Core katmanı içeriği

4.2.1. Models ve DTOs Klasörü



Şekil 4.2.1.1. Modellerin listesi



Şekil 4.2.1.2. DTO'ların listesi

Uygulama nesne yönelimli programlama prensibi ile yazılmıştır. Models katmanında uygulamada var olan nesnelerin(Kullanıcı,mülk gibi) benzetimi yapılmıştır. Örneğin uygulamayı kullanacak kullanıcıyı modellemek, bilgilerini saklamak amacıyla User.cs sınıfı oluşturulmuş yanı özel yapılı değişken tanımlanmıştır.

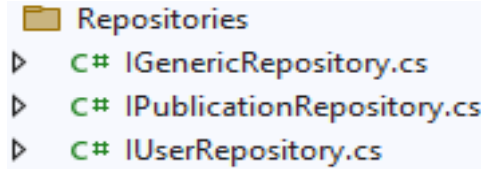
DTO(Data Transfer Object) klasörü ise kullanıcıya sunulmak istenilen verilerin tutulduğu katmandır. Bu sayede kullanıcıya gereksiz ve kritik bilgiler aktarılmamış olur.

Diğer modelleri ve DTO'ları ayrıntılı olarak inceleyebileceğiniz Github linkine aşağıdan ulaşabilirsiniz.

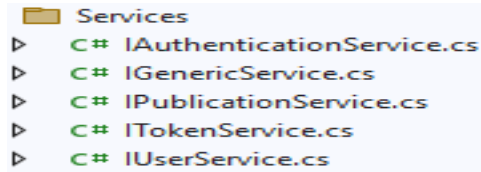
```
namespace AdventureNest.Core.Models
{
    public class User : BaseEntity
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Email { get; set; }
        public byte[] PasswordHash { get; set; }
        public byte[] PasswordSalt { get; set; }
        public DateTime Birthday { get; set; }
        public string ProfilePhotoPath { get; set; }
        public ICollection<Property> Properties { get; set; }
    }
}
```

Şekil 4.2.1.3. Örnek model içeriği

4.2.2. Repositories, UnitOfWorks ve Services Klasörü



Şekil 4.2.2.1. Repositories arayüz listesi



Şekil 4.2.2.2. Services arayüz listesi

Repositories klasörünün içindeki arayüzlerde, veritabanı işlemleri için gerekli fonksiyonları bulunmaktadır. Bu tanımlı fonksiyonlar, Repository katmanındaki ilgili sınıflara implemente edilir. Örneğin IGenericRepository içinde, Models klasörü içindeki sınıfların veritabanına temel CRUD işlemlerini yaptıracak fonksiyonlar bulunmaktadır. Diğer iki arayüzde ise isimlendirilmiş oldukları model sınıfları ile ilgili spesifik fonksiyonlar bulunmaktadır.

```

namespace AdventureNest.Core.Repositories
{
    public interface IGenericRepository<TEntity> where TEntity:class
    {
        //GetAll
        IQueryable<TEntity> GetAllAsync();

        //GetById
        Task<TEntity> GetByIdAsync(int id);

        //Add
        Task AddAsync(TEntity entity);

        //AddRange
        Task AddRangeAsync(IEnumerable<TEntity> entities);

        //Remove
        void Remove(TEntity entity);

        //RemoveRange
        void RemoveRange(IEnumerable<TEntity> entities);

        //Update
        void Update(TEntity entity);

        //Any
        Task<bool> AnyAsync(Expression<Func<TEntity, bool>> expression);

        //Where
        IQueryable<TEntity> Where(Expression<Func<TEntity, bool>> expression);
    }
}
  
```

Şekil 4.2.2.3. IGenericRepository içeriği

Services klasörünün içindeki arayüzler, Service katmanındaki ilgili sınıflara implemente edilir. Arayüzlerin içindeki fonksiyonların içeriğine uygulamanın iş kodları ve mantığı yazılır. Services klasörü içindeki arayüzler, Repositories klasörünün içindeki arayüzler ile ilişkilidir. Örneğin IGenericService, Repositories klasörü içindeki IGenericRepository arayüzünü ile ilişkilidir. Bu iki klasör arasındaki ilişki Service ve Repository katmanları tanıtıldıktan sonra daha iyi anlaşılacaktır.

```
namespace AdventureNest.Core.Services
{
    public interface IGenericService<TEntity, TDto> where TEntity : class
        where TDto : class
    {
        // GetAll
        Task<CustomResponseDto<IEnumerable<TDto>>> GetAllAsync();

        // GetById
        Task<CustomResponseDto<TDto>> GetByIdAsync(int id);

        // Add
        Task<CustomResponseDto<TDto>> AddAsync(TDto dto);

        // AddRange
        Task<CustomResponseDto<IEnumerable<TDto>>> AddRangeAsync(IEnumerable<TDto> dtos);

        // Remove
        Task<CustomResponseDto<NoContentDto>> RemoveAsync(int id);

        // RemoveRange
        Task<CustomResponseDto<NoContentDto>> RemoveRangeAsync(IEnumerable<TDto> dtos);

        // Update
        Task<CustomResponseDto<NoContentDto>> Update(TDto dto, int id);

        // Any
        Task<bool> AnyAsync(Expression<Func<TDto, bool>> expression);

        // Where
        Task<CustomResponseDto<IEnumerable<TDto>>> Where(Expression<Func<TEntity, bool>> expression);
    }
}
```

Şekil 4.2.2.4. IGenericService içeriği

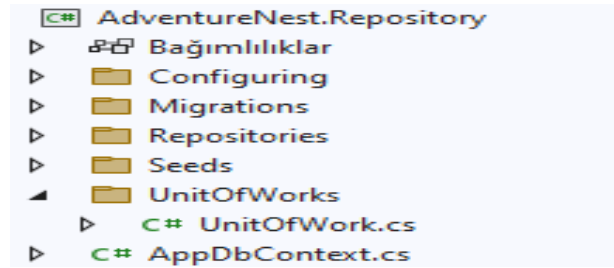
UnitOfWorks klasörünün içinde ise IUnitOfWork adında bir arayüz vardır. Bu arayüzün içinde tanımlı olan fonksiyonlar kullanıcı tarafından ekleme, güncelleme ve silme işlemleri yapıldıktan sonra değişiklikleri veritabanına yansıtmak için kullanılır. IUnitOfWorks arayüzü Repositories katmanı içinde implemente edilip içine gerekli kodlar yazılır.

4.2.3. Configuration Klasörü

Configuration klasörünün içinde CustomTokenOption adında bir sınıf bulunur bu sınıfın amacı uygulamada kullanılacak olan güvenlik anahtarının(Json Web Token) bilgilerini(Payload) tutmasıdır. Bu bilgiler API katmanındaki appsettings.json içinde tanımlanmıştır ve kod tarafında güvenlik anahtarı oluşturulurken bu sınıf, bilgileri appsettings.json dosyasından alır ve anahtara eklenir.

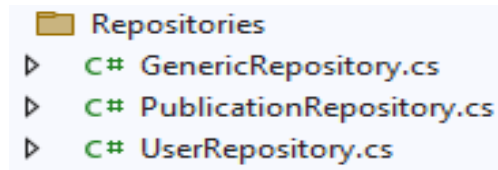
4.3. Repository Katmanı

Projenin Repository katmanında uygulamanın veritabanı işlemlerini içeren sınıflar bulunur.



Şekil 4.3. Repository katmanı içeriği

4.3.1. Repositories Klasörü



Şekil 4.3.1.1. Repositories sınıf listesi

Bu klasör içerisinde bulunan sınıflar, Core katmanındaki repository arayüzlerini implemente eder ve Entity Framework paketini kullanarak gerekli veritabanı işlemlerini yapar.

```
namespace AdventureNest.Repository.Repositories
{
    public class GenericRepository<TEntity> : IGenericRepository<TEntity> where TEntity : class
    {
        protected readonly AppDbContext _context;
        protected readonly DbSet<TEntity> _dbSet;

        public GenericRepository(AppDbContext context)
        {
            _context = context;
            _dbSet = _context.Set<TEntity>();
        }

        public async Task AddAsync(TEntity entity)
        {
            await _dbSet.AddAsync(entity);
        }

        public async Task AddRangeAsync(IEnumerable<TEntity> entities)
        {
            await _dbSet.AddRangeAsync(entities);
        }

        public async Task<bool> AnyAsync(Expression<Func<TEntity, bool>> expression)
        {
            return await _dbSet.AnyAsync(expression);
        }

        public IQueryable<TEntity> GetAllAsync()
        {
            return _dbSet.AsNoTracking().AsQueryable();
        }

        public async Task<TEntity> GetByIdAsync(int id)
        {
            return await _dbSet.FindAsync(id);
        }
    }
}
```

Şekil 4.3.1.2. Örnek Repository sınıf içeriği

4.3.2. Seeds Klasörü

Bu klasör içerisinde veritabanında oluşacak tablolara örnek veriler eklemek için bazı sınıflar bulunur.

```
namespace AdventureNest.Repository.Seeds
{
    public class PublicationSeed : IEntityTypeConfiguration<Publication>
    {
        public void Configure(EntityTypeBuilder<Publication> builder)
        {
            builder.HasData(
                new Publication
                {
                    Id = 1,
                    Description= "Villa with Sapanca Lake view",
                    Header= "Sakarya Lux Villa House",
                    IsActive= true,
                    Price = 1000,
                    PropertyId = 1,
                    CreatedDate = DateTime.Now
                },
                new Publication
                {
                    Id = 2,
                    Description = "Villa with Miami Beach view",
                    Header = "Ultra Lux Villa in Miami",
                    IsActive = true,
                    Price = 4000,
                    PropertyId = 2,
                    CreatedDate = DateTime.Now
                }
            );
        }
    }
}
```

Şekil 4.3.2. Örnek Seed sınıfı içeriği

4.3.3. UnitOfWorks Klasörü

Bu klasör içerisinde Core katmanındaki UnitOfWorks klasörü içindeki arayüzün implemente edildiği sınıf bulunur.

4.3.4. Configuring Klasörü

Bu klasör içerisinde Core katmanındaki tanımlanan modellerin özelliklerinin kuralları oluşturulur. Oluşturulan kurallar veritabanına kaydedilir.

```

namespace AdventureNest.Repository.Configuring
{
    public class BookingConfiguration : IEntityTypeConfiguration<Booking>
    {
        public void Configure(EntityTypeBuilder<Booking> builder)
        {
            builder.HasKey(x => x.Id);
            builder.Property(x => x.Id).UseIdentityColumn();

            builder.Property(x => x.BookingDate).IsRequired();
            builder.Property(x => x.CheckInDate).IsRequired();
            builder.Property(x => x.DepartureDate).IsRequired();
            builder.Property(x => x.PricePerDay).IsRequired().HasColumnType("decimal(18,2)");
            builder.Property(x => x.BookingStatus).IsRequired().HasMaxLength(30);
            builder.Property(x => x.TotalPrice).IsRequired().HasColumnType("decimal(18,2)");

            builder.ToTable("Bookings");

            builder.HasOne(x => x.Publication).WithMany(x => x.Bookings)
                .HasForeignKey(x => x.PublicationId);
        }
    }
}

```

Şekil 4.3.4. Örnek Configuration sınıfı içeriği

4.3.5. Migrations Klasörü

Uygulamanın veritabanı code first yaklaşımı ile oluşturulur. Bu yaklaşımın amacı, geliştiricinin veritabanına girip manuel olarak tabloları oluşturmasını kolaylaştırmaktır. Code first yaklaşımında veritabanına dair bütün sınıflar tanımlandıktan sonra “add-migration xxx” komutu konsola yazılır, bunun sonucunda Migrations adında klasör oluşturulur. Bu klasörün içinde bizim koymuş olduğumuz kurallar bulunur. “update-database” komutu ile birlikte de yapılan değişiklikler veritabanına kaydedilir.

4.3.6. AppDbContext Sınıfı

Bu sınıf sayesinde uygulama veritabanına modellenir, sınıfın içerisine tanımlanan her özellik bir veritabanı tablosuna tekabül eder.

```
namespace AdventureNest.Repository
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
        {
        }

        public DbSet<Booking> Bookings { get; set; }

        public DbSet<Property> Properties { get; set; }

        public DbSet<Publication> Publications { get; set; }

        public DbSet<Token> Tokens { get; set; }

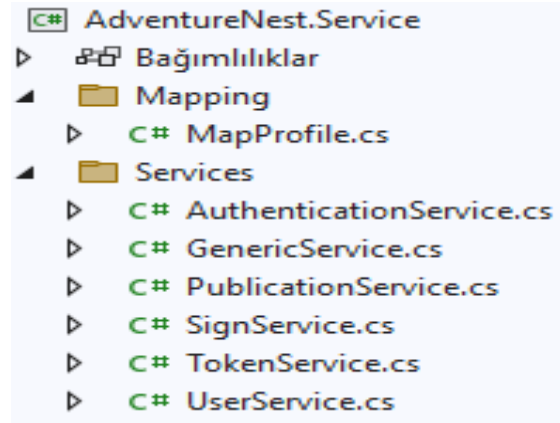
        public DbSet<User> Users { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());
            base.OnModelCreating(modelBuilder);
        }
    }
}
```

Şekil 4.3.6. AppDbContext sınıfı içeriği

4.4. Service Katmanı

Projenin Service katmanında uygulamanın iş kuralları içeren ve mantığını oluşturan sınıflar bulunur.



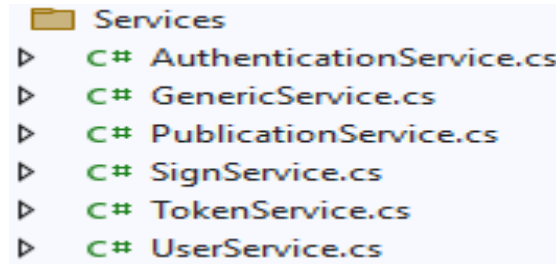
Şekil 4.4. Service katmanı içeriği

4.4.1. Mapping Klasörü

Servis katmanına kullanıcıya gönderilmesi ve kullanıcıdan alınması için DTO nesneleri kullanılır. Kullanıcının talebine göre veritabanından gelen veriler bu nesnelere çevrilip kullanıcıya gönderilir veya veritabanından nesneler alınıp DTO formatına çevrilir kullanıcıya öyle sunulur. Mapping klasörü içerisindeki MapProfile.cs sınıfı sayesinde temel modeller ve DTO'lar tek bir kod satırında birbirine dönüştürülür. Bu işlem için projeye AutoMapper kütüphanesi dahil edilir.

4.4.2. Services Klasörü

Servis klasörü içinde uygulamanın iş akışını, kurallarını gerçekleştiren fonksiyonlar bulunur bu fonksiyonlar, Core katmanında bulunan Services klasörü içerisindeki arayüzleri implement etmiştir.



Şekil 3.4.2.1. Service klasörü listesi

```
namespace AdventureNest.Service.Services
{
    public class GenericService<TEntity, TDto> : IGenericService<TEntity, TDto> where TEntity : class
        where TDto : class
    {
        protected readonly IGenericRepository<TEntity> _repository;
        protected readonly IUnitOfWork _unitOfWork;
        protected readonly IMapper _mapper;

        public GenericService(IGenericRepository<TEntity> repository, IUnitOfWork unitOfWork, IMapper mapper) {...}

        public async Task<CustomResponseDto<TDto>> AddAsync(TDto dto)
        {
            var entity = _mapper.Map<TEntity>(dto);

            await _repository.AddAsync(entity);
            await _unitOfWork.CommitAsync();

            var newDto = _mapper.Map<TDto>(entity);

            return CustomResponseDto<TDto>.Success(201, newDto);
        }

        public async Task<CustomResponseDto<IEnumerable<TDto>>> AddRangeAsync(IEnumerable<TDto> dtos)
        {
            var entities = _mapper.Map<IEnumerable<TEntity>>(dtos);

            await _repository.AddRangeAsync(entities);
            await _unitOfWork.CommitAsync();

            var newDtos = _mapper.Map<IEnumerable<TDto>>(entities);

            return CustomResponseDto<IEnumerable<TDto>>.Success(201, newDtos);
        }

        public Task<bool> AnyAsync(Expression<Func<TDto, bool>> expression)
        {
            throw new NotImplementedException();
        }

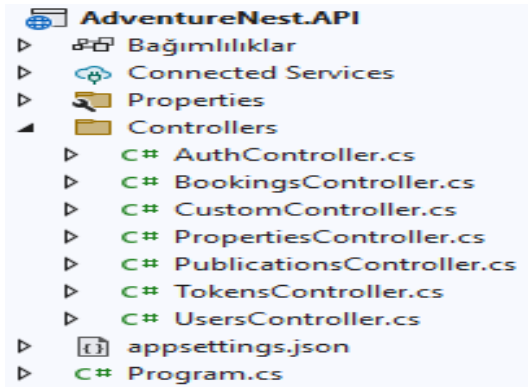
        public async Task<CustomResponseDto<IEnumerable<TDto>>> GetAllAsync()
        {
            var entities = _repository.GetAllAsync();

            var dtos = _mapper.Map<IEnumerable<TDto>>(entities);

            return CustomResponseDto<IEnumerable<TDto>>.Success(200, dtos);
        }
    }
}
```

Şekil 4.4.2.2. IGenericService sınıf içeriği

4.5. API Katmanı



Şekil 3.5.1. API katmanı listesi

Uygulamanın bu katmanı frontend kısmına açılacak ve uygulamanın çalıştırılacağı katmandır. Business katmanından dependency injection yöntemiyle aktarılan `_service` nesnesi yardımıyla gerekli HTTP işlemleri veritabanıyla beraber yapılmaktadır. Burada önceki kısımlarda tanımlanan HTTP metotları kullanılarak veri transfer işlemleri yapılmıştır. Örnek kod aşağıda verilmiştir.

```
namespace AdventureNest.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [Authorize]
    public class UsersController : CustomController
    {
        private readonly IUserService _service;

        public UsersController(IUserService service)
        {
            _service = service;
        }

        [HttpGet]
        public async Task<IActionResult> GetAllUsers()
        {
            var response = await _service.GetAllAsync();
            return await CreateActionResult(response);
        }

        [HttpGet("{id}")]
        public async Task<IActionResult> GetUserById(int id)
        {
            var response = await _service.GetByIdAsync(id);
            return await CreateActionResult(response);
        }

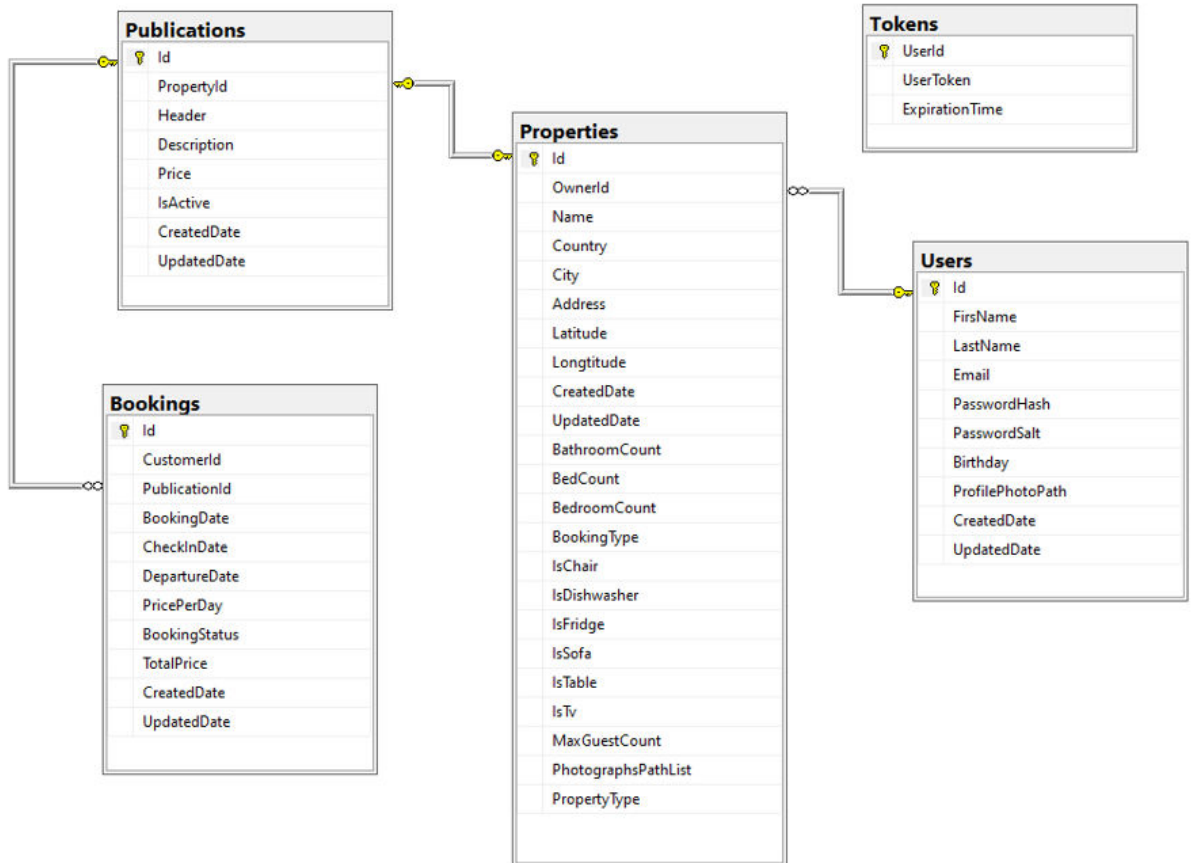
        [HttpPost]
        [AllowAnonymous]
        public async Task<IActionResult> AddUser([FromBody] UserDto userDto)
        {
            var response = await _service.CreateUser(userDto);
            return await CreateActionResult(response);
        }

        [HttpPut("{id}")]
        public async Task<IActionResult> UpdateUser([FromBody] UserDto userDto, int id)
        {
            var response = await _service.Update(userDto, id);
            return await CreateActionResult(response);
        }
    }
}
```

Şekil 4.5.2. UsersController İçeriği

4.6. Veritabanı

Code first yaklaşımı ile veritabanı oluşturulan uygulamanın veritabanı tablo ilişkisi aşağıda verilmiştir.



Şekil 4.6. Veritabanı İlişkisi

4.7. Uygulama Kodları

Uygulamanın kodlarına <https://github.com/mertsigirci11/adventure-nest> adresinden ulaşabilirsiniz.

BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Proje tamamlandıktan sonra, Derin Öğrenme modülünü ve web stack bir arada kullanmanın çok güçlü bir kombinasyon olduğu öğrenildi. Derin öğrenme modülünde %79,5 doğruluk elde edildi. Backend'de modelin hesaplaması için ortalama süre, her bir görüntü için 10 saniyenin altında oldu. Genel olarak, veri boyutu artırılırsa ve mAP değeri düşük nesneler çıkarılırsa mAP'de bir artış gözlemlenebilir. Bizim durumumuzda, gaz ocağında düşük bir mAP tespit edildi, bu yüzden bunu çıkararak doğruluk daha da artabilir.

Uygulamanın Github linki: <https://github.com/waasiq/adventure-nest>

İkinci repo: <https://github.com/mertsigirci11/adventure-nest>

KAYNAKLAR

- [1] A comparative study of YOLOv5 models performance for image localization and classification Marko Horvat., Gordan Gledec., Ljudevit Jelečević. September, 2022.
- [2] YOLOv5 SOTA Realtime Instance Segmentation. Glenn Jocher; Ayush Chaurasia; Alex Stoken; Jirka Borovec; NanoCode012; Yonghye Kwon; Kalen Michael; TaoXie; Jiacong Fang; imyhxy; Lorna; 曾逸夫(Zeng Yifu); Colin Wong; Abhiram V; Diego Montes; Zhiqiang Wang; Cristi Fati; Jebastin Nadar; Laughing; UnglvKitDe; Victor Sonck; tkianai; yxNONG; Piotr Skalski; Adam Hogan; Dhruv Nair; Max Strobel; Mrinal Jain. November 2022.
- [3] <https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/>
- [4] <https://devnot.com/2016/rest-mimarisi-ve-restful-servisler/>
- [5] <https://denizirgin.com/rest-ve-restful-web-servis-kavram%C4%B1-30bc4400b9e0>

ÖZGEÇMİŞ

Waasiq Masood, 22-07-2000 tarihinde Rawalpindi/Pakistan'da doğdu. Lise eğitimini Quetta'dan tamamladıktan sonra 2 yıllık Fsc Pre-Mühendislik eğitimini tamamlamak için Cadet College Hasanabdal'a gitti. Bundan sonra 2019'da Sakarya Üniversitesi'ne Bilgisayar Mühendisliği bölümünde kaydoldu. 2020 yılında Fauji Foundation, Rawalpindi'de bir frontend geliştirici stajyeri olarak çalıştı. 2022 Ocak ayında CENTER SAU'da Makine Öğrenimi stajyeri olarak çalıştı. Temmuz 2022'de İstanbul'da Trampoline'da full stack stajyeri olarak çalıştı.

Mert Sığırcı, 11.08.1999'da Çorum/Türkiye'de doğdu. İlkokul, ortaokul ve lise eğitimini Çorum'da tamamladı. 2018 yılında Çorum Anadolu Lisesi'nden mezun oldu. 2019 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nü kazandı. 2022 yılının Ocak ayından Mayıs ayına kadar Deneyap Atölyeleri'nde yazılım teknolojileri eğitmenliği yaptı. 2022 yılının Haziran ayında bir aylık Sakarya Üniversitesi Bilgi İşlem Daire Başkanlığı'nda donanım stajını ve 2022 yılının Eylül ayında 5 aylık KLN Yazılım ve Bilişim Sistemleri Ltd. Şti. şirketinde yazılım stajını yapmıştır.

BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU : YAPAY ZEKÂ DESTEKLİ FULLSTACK EMLAK KİRALAMA

UYGULAMASI

ÖĞRENCİLER (Öğrenci No/AD/SOYAD): B18121055 Waasiq MASOOD - B191210078

Mert SİĞİRCİ

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problem tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişkisi modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN (JÜRİ ADINA):

DANIŞMAN İMZASI: