



**T.C.  
İSTANBUL ATLAS ÜNİVERSİTESİ**

**Mühendislik ve Doğa Bilimleri Fakültesi  
Bilgisayar Mühendisliği Bölümü**

**Lisans Bitirme Projesi**

**Barınaklardan Hayvan Sahiplendirme Web Sitesi  
Web Site İsmi: Yuvamı Bul**

Remzican SOKUR  
210501017

Koray Işık ALTUN  
210501014

Halil Can GÜREL  
210501019

Bitirme Projesi Danışmanı: Dr. Öğr.Üyesi Osman Durmaz

İstanbul, Mayıs 2025  
T.C.



**T.C.  
İSTANBUL ATLAS ÜNİVERSİTESİ**

**Mühendislik ve Doğa Bilimleri Fakültesi**

**Bilgisayar Mühendisliği Bölümü**

**Lisans Bitirme Projesi**

**Barınaklardan Hayvan Sahiplendirme Web Sitesi  
Web Site İsmi: Yuvamı Bul**

Remzican SOKUR  
210501017

Koray Işık ALTUN  
210501014

Halil Can GÜREL  
210501019

Bitirme Projesi Danışmanı: Dr. Öğr.Üyesi Osman Durmaz

<u>Jüri Üyeleri:</u>	<u>İmza:</u>
Dr. Öğr.Üyesi Osman Durmaz	.....
Dr. Öğr.Üyesi Osman Durmaz	.....
Prof.Dr. Afif Sıddıki	.....

## TEŞEKKÜR

---

Bitirme projesi danışmanımız olmayı kabul eden ve projeyi geliştirme sürecinde bize yol gösterici olan değerli hocamız Dr.Öğr. Üyesi Osman Durmaz’a sevgilerimizi ve saygılarımızı sunarız .

Üniversite hayatımız boyunca desteğini hiçbir zaman bizden esirgemeyen, her türlü konuda yardımcı olan değerli hocamız Dr.Öğr. Üyesi Recep DURANAY’a sevgilerimizi ve saygılarımızı sunarız.

MAYIS, 2025

Koray Işık Altun

Remzican Sokur

Halil Can Gürel

## ÖZET

---

Bu projede , sahipsiz hayvanların barınaklardan sahiplenilmesini kolaylaştırmak amacıyla geliştirilen web tabanlı bir hayvan sahiplendirme platformu sunulmuştur. Platform; kullanıcı kayıt ve girişi, sahipsiz hayvanlar için ilan oluşturma ve görüntüleme, blog içerikleri ile bilgilendirme, harita üzerinden barınak konumlarını görüntüleme ve bağış yapma gibi işlevleri içermektedir .

Sistem, genel kullanıcılar ve barınak yöneticileri için farklı roller tanımlayarak kullanıcı ihtiyaçlarına uygun arayüzler geliştirilmiştir. Genel kullanıcılar, sahipsiz hayvan ilanlarını inceleyip görüntüleyebilir ; barınak yöneticileri, barınakta bulunan hayvanlar için ilan ekleyebilmekte ve bu ilanları düzenleyebilmektedir. Yönetici kullanıcı ise tüm sistem bileşenlerini yönetebilme yetkisine sahiptir.

Web sitesi , kullanıcı dostu tasarımı ile geniş bir kullanıcı kitlesine hitap edecek şekilde hazırlanmıştır. Bu proje, teknolojiyi sosyal sorumlulukla birleştirerek sahipsiz hayvanların sahiplenme süreçlerini dijital ortamda daha etkin ve erişilebilir hale getirmeyi amaçlamaktadır.

### **Anahtar Kelimeler:**

Hayvan sahiplendirme, Web tabanlı platform, Barınak yönetimi

## ABSTRACT

---

This study presents a web-based animal adoption platform developed to facilitate the adoption of stray animals from shelters. The platform includes various functionalities such as user registration and login, creation and viewing of adoption listings for stray animals, informative blog content, viewing shelter locations via maps, and making donations.

The system is designed with different user roles, including general users and shelter managers, to meet specific user needs through dedicated interfaces. General users can browse and view adoption listings, while shelter managers can add and edit listings for animals in their shelters. Additionally, an admin user has full authority to manage all components of the system.

The website has been designed with a user-friendly interface to appeal to a wide audience. This project combines technology with social responsibility to make the adoption process for stray animals more efficient and accessible in a digital environment.

**Keywords:**

Animal adoption, Web-based platform, Shelter management

## İçindekiler

TEŞEKKÜR .....	iv
ÖZET .....	iii
ABSTRACT.....	iv
BÖLÜM 1: GİRİŞ.....	1
1.1 Literatür Taraması :.....	1
1.2 Projenin Amacı :.....	2
1.3 Projenin Tanımı: .....	2
1.4 Projenin Kapsamı :.....	4
1.5 Kullanıcı Roller ve Yetkileri :.....	6
BÖLÜM 2: Proje Planı.....	7
2.1 Kullanılan Teknolojiler ve Araçlar :.....	7
2.2 Proje Planı :.....	8
BÖLÜM 3: SİSTEM ANALİZİ VE TASARIMI.....	10
3.1 Kullanıcı Gereksinimleri :.....	10
3.2 Sistem Gereksinimleri :.....	10
3.3 Backend (Sunucu Tarafı).....	10
3.4 Veri Akış Diyagramları (DFD):.....	11
3.5 Sistem Mimarisi : .....	11
Avantajları:.....	12
Dezavantajları: .....	12
3.6 Veritabanı Tasarımı.....	13
3.6.1. Kullanıcı Tablosu.....	13
3.6.2 İlan Tablosu:.....	14
3.6.3. Blog Tablosu: .....	15
BÖLÜM 4 :API TASARIMI.....	16
4.1 API Tasarımı Nedir? .....	16

<b>4.2 Swagger Nedir?.....</b>	<b>16</b>
4.2.1. Swagger Dökümanı Oluşturma Süreci .....	17
<b>4.3 Kullanıcı Yönetimi.....</b>	<b>22</b>
<b>4.4 Kullanıcı İşlemleri.....</b>	<b>22</b>
4.4.1 Teknik Detaylar .....	23
<b>4.5 İlan Yönetimi .....</b>	<b>24</b>
4.5.1 İlan Yönetimi İşlevleri .....	24
4.5.2 Teknik Detaylar .....	25
<b>4.6 Blog Yönetimi .....</b>	<b>26</b>
4.6.1 Blog Yönetimi İşlevleri .....	26
4.6.2 Teknik Detaylar .....	27
<b>BÖLÜM 5: Kullanılan Teknolojiler.....</b>	<b>28</b>
<b>5.1. HTML Nedir? .....</b>	<b>28</b>
<b>5.2. CSS Nedir?.....</b>	<b>28</b>
<b>5.3. JavaScript Nedir? .....</b>	<b>29</b>
<b>5.4. Backend Teknolojileri .....</b>	<b>30</b>
5.4.1 Node.js .....	30
5.4.2 Express.js.....	31
<b>5.5 Veritabanı Teknolojisi.....</b>	<b>32</b>
5.5.1. Diğer Araçlar .....	33
<b>BÖLÜM 6: UYGULAMA MODÜLLERİ.....</b>	<b>35</b>
<b>6.1 Kullanıcı Yönetimi.....</b>	<b>35</b>
6.1.1 Kayıt Olma ve Giriş Yapma.....	35
6.1.2 Profil Güncelleme.....	36
6.1.3 İlan Oluşturma.....	37
6.1.4 Blog Oluşturma.....	38
6.1.5 Kullanıcı Yetkilendirme .....	39
<b>BÖLÜM 7: Kullanıcı Arayüzü .....</b>	<b>41</b>
<b>7.1. Giriş Yap Arayüzü .....</b>	<b>41</b>
7.1.1 Arayüzde Bulunan Öğeler: .....	41
7.1.2 Kullanıcı Deneyimi Açısından Katkılar: .....	41
7.1.3 Teknik Özellikler: .....	42

7.1.4 Ekran Görüntüsü: .....	42
<b>7.2. Kayıt Ol Arayüzü .....</b>	<b>42</b>
7.2.1 Arayüzde Bulunan Öğeler: .....	43
7.2.2. Kullanıcı Deneyimine Katkıları.....	43
7.2.3. Teknik Uygulama Detayları.....	43
7.2.4. Ekran Görüntüsü .....	44
<b>7.3. İlanlar Arayüzü.....</b>	<b>44</b>
7.3.1 Arayüzde Yer Alan Bileşenler: .....	44
7.3.2 İlan Kartları: .....	45
7.3.3 Kullanıcı Deneyimi Açısından Katkılar: .....	45
7.3.4 Ekran Görüntüsü 1 : .....	46
7.3.5 Ekran Görüntüsü 2 : .....	46
<b>7.4 Blog Sayfası Arayüzü .....</b>	<b>47</b>
7.4.1. Arayüz Öğeleri.....	47
7.4.2. Kullanıcı Deneyimine Katkıları.....	47
7.4.3 Teknik Uygulama Detayları.....	48
7.4.4. Ekran Görüntüsü 1: (Kullanıcı ve Barınak Görünümü).....	48
7.4.5. Ekran Görüntüsü 2: (Admin Görünümü).....	49
<b>7.5 Bağış Yap Sayfası Arayüzü .....</b>	<b>49</b>
7.5.1. Arayüz Öğeleri.....	49
7.5.2. Ekran Görüntüsü: .....	50
<b>7.6.Haritalar Arayüzü.....</b>	<b>50</b>
7.6.1 Ekran Görüntüsü: .....	51
<b>BÖLÜM 8: Sonuç ve Değerlendirme .....</b>	<b>51</b>
<b>8.1 Projenin Başarı Durumu.....</b>	<b>52</b>
<b>8.2 Gelecek Çalışmalar.....</b>	<b>52</b>
<b>BÖLÜM 9 : Kaynakça.....</b>	<b>55</b>
<b>ÖZGEÇMİŞLER.....</b>	<b>57</b>



# BÖLÜM 1: GİRİŞ

---

Bu proje, sahihsiz hayvanların barınaklardan sahiplenilmesini kolaylaştırmak amacıyla geliştirilen kullanıcı dostu bir web platformudur. Sistem; kullanıcı kayıt ve giriş işlemleri, ilan oluşturma ve görüntüleme, blog içerikleri, harita üzerinden barınak keşfi ve bağış yapma gibi işlevleri bir araya getirerek, hem barınak görevlilerine hem de hayvanseverlere pratik çözümler sunar.

## 1.1 Literatür Taraması :

### Benzer uygulamalar

- Haytap (Hayvan Hakları Federasyonu)
  - Türkiye'deki barınaklarla iş birliği yaparak sahiplendirme ilanları yayınlayan bir platformdur.
  - Web sitesi üzerinden sahiplendirme ilanları paylaşılır.
  - Ayrıca hayvan hakları konusunda bilgilendirici içerikler, bağış kampanyaları ve gönüllülük imkanları sunar. [1]
- Petfinder (ABD)
  - Amerika merkezli en büyük sahiplendirme platformlarından biridir.
  - Barınaklar, hayvanları tür, yaş, konum ve sağlık durumuna göre detaylı biçimde ilan olarak ekler.
  - Kullanıcılar ilanlara başvurarak doğrudan barınakla iletişime geçebilir.
  - Mobil uygulama desteğiyle geniş kullanıcı kitlesine ulaşır. [2]
- Adopt A Pet (ABD/Kanada)
  - Hayvan sahiplendirme alanında önemli bir diğer platformdur.
  - Kullanıcıların bulunduğu konuma göre en yakın sahiplendirme ilanlarını listeler.
  - Kullanıcı dostu arayüzü ve e-posta bildirim sistemiyle, kullanıcılara uygun hayvanlar önerilir. [3]

## 1.2 Projenin Amacı :

Bu projenin amacı, barınaklarda yaşayan sahipsiz hayvanların kolay ve güvenli bir şekilde sahiplendirilmesini sağlamaktır. Geliştirdiğimiz web sitesi sayesinde barınak görevlileri, hayvanların bilgilerini (fotoğraf, yaş, cinsiyet, sağlık durumu gibi) sisteme yükleyebilecek. Böylece hayvanseverler, site üzerinden hayvanları inceleyerek sahiplenme yapabilecekler.

Türkiye'de pek çok barınak, kaynak yetersizliği nedeniyle hayvanlara yeterli bakımı sunmakta zorlanıyor. Bu platform, sahiplendirme sürecini hızlandırarak barınaklardaki yoğunluğu azaltmayı ve daha fazla hayvanın sıcak bir yuvaya kavuşmasını amaçlıyor.

Ayrıca sitemizde, hayvan bakımıyla ilgili bilgilendirici blog yazıları, barınakların konumları, hayvanlar için bulunan sivil toplum kuruluşları için bağış sayfası da yer alacak. Böylece sadece sahiplendirme değil, toplumda hayvan sevgisi ve sorumluluğu da yaygınlaştırılacak.

Kısacası, bu proje sayesinde teknoloji ile hayvanlar arasında bir köprü kuruyor, onların daha iyi bir hayat yaşamalarına katkı sağlamayı hedefliyoruz.

## 1.3 Projenin Tanımı:

Bu proje, sahipsiz hayvanların barınaklardan sahiplenilmesini kolaylaştırmak amacıyla geliştirilmiş bir hayvan sahiplendirme web platformudur. Platform, hem barınak görevlilerine hem de hayvansever kullanıcılara yönelik kullanıcı dostu bir arayüz sunar ve hayvanların daha hızlı, güvenli ve şeffaf bir şekilde yeni yuvalarına kavuşmalarını amaçlar.

Sistem; kullanıcı kayıt ve giriş işlemleri, ilan oluşturma ve görüntüleme, blog içerikleriyle bilgilendirme, harita üzerinden barınak keşfi ve bağış yapma gibi birçok işlevsel bileşeni bir araya getirir. Giriş ve kayıt ekranları, kullanıcıların sisteme kolayca katılımını sağlarken; ilanlar arayüzü, hayvanlara ait bilgilerin (fotoğraf, yaş, cinsiyet, konum ) kullanıcıya sunulmasına olanak tanır. Ayrıca blog sayfası ile kullanıcılar hayvan bakımı ve sahiplendirme süreçlerine dair güncel bilgilere erişebilirler.

Harita arayüzü sayesinde kullanıcılar, bulundukları konuma en yakın barınakları görebilir; bağış sayfası ise sivil toplum kuruluşlarına güvenli şekilde destek olma imkânı sunar. Sistem, kullanıcı deneyimi ön planda tutularak sade ve mobil uyumlu şekilde geliştirilmiştir. Veriler API aracılığıyla dinamik olarak yönetilirken, kullanıcıların kimlik doğrulama süreçleri güvenli bir şekilde gerçekleştirilir.

Bu platform, teknolojiyi kullanarak barınaklar ile toplum arasında köprü kurmakta; sahihsiz hayvanların daha hızlı sahiplendirilmesine katkı sağlamaktadır

## 1.4 Projenin Kapsamı :

Bu projede sahipsiz hayvanların sahiplendirilmesini kolaylaştıran, kullanıcı dostu, bilgi odaklı ve etkileşimli bir web tabanlı platform geliştirilmesini kapsamaktadır. Proje yalnızca bir ilan verme sisteminden ibaret olmayıp; aynı zamanda kullanıcıları bilgilendiren, bağış yapılmasına aracılık eden, fiziksel barınaklarla dijital köprü kuran çok yönlü bir yapıya sahiptir.

Kullanıcı yönetim modülü, kullanıcıların güvenli şekilde kayıt olması ve sisteme giriş yapmasını sağlar. Kullanıcılar kişisel profillerini yönetebilir; ilanlarını ve blog yazılarını görüntüleyebilir. Sistem, farklı kullanıcı rollerine sahiptir: genel kullanıcı (User), barınak yöneticisi (Shelter) ve yönetici (Admin). Bu roller, kullanıcıların sisteme olan erişim ve yetki düzeylerini belirler.

Hayvan sahiplendirme ilan sistemi, hayvan türü, cinsiyeti, yaşı, sağlık durumu gibi bilgileri içeren ilanlar oluşturabilir. Sistemdeki tüm ilanlar listelenebilir, filtrelenebilir ve detaylı şekilde görüntülenebilir. Bu sayede kullanıcılar, uygun sahiplendirme ilanlarını kolayca bulabilir ve iletişime geçebilir.

Bilgilendirici blog modülü hayvan bakımı, sahiplendirme süreçleri ve hayvan hakları gibi konular hakkında içerikler sunulmasını amaçlayan bu modülde, yalnızca yetkili kullanıcılar (Admin) içerik ekleyebilir ve düzenleyebilir. Böylece kullanıcıların bilinç düzeyi artırılarak daha sağlıklı sahiplendirme süreçleri teşvik edilir.

Bağış yapma arayüzü, kullanıcıların güvenilir sivil toplum kuruluşlarına (STK) kolayca bağış yapabilmesini sağlayan bir yönlendirme sayfası sunulacaktır. Her STK için doğrudan bağış bağlantıları yer alacak ve bu sayede sistem, sosyal fayda odaklı katkılar sunmaya devam edecektir.

Barınak konum haritası İstanbul'daki barınaklar, harita üzerinde konumlarıyla birlikte gösterilecektir. Kullanıcılar bu sayede kendilerine en yakın barınağı tespit edebilir ve doğrudan iletişime geçebilir. Bu özellik, dijital sistem ile fiziksel barınaklar arasında etkili bir bağlantı kurmayı amaçlamaktadır.

Güvenlik ve veri yönetimi sistemdeki kullanıcı oturumları JSON Web Token (JWT) teknolojisi ile güvenli şekilde yönetilecektir. Aynı zamanda form doğrulama sistemleri entegre edilerek, hatalı girişlere karşı kullanıcıların doğru şekilde bilgilendirilmesi sağlanacaktır.

Proje yalnızca teknik bir web uygulaması geliştirme süreci değil; aynı zamanda sosyal sorumluluk bilinciyle şekillendirilmiş, toplum yararını gözeten bir yazılım geliştirme çalışmasıdır. Hem bireysel hem de kurumsal kullanıcıları bir araya getirerek, sahihsiz hayvanların daha hızlı, sağlıklı ve bilinçli bir şekilde sahiplenilmesini hedeflemektedir.

## 1.5 Kullanıcı Roller ve Yetkileri :

Platformdaki kullanıcı rolleri ve yetkileri üç ana gruptan oluşmaktadır. Kullanıcı (User), Barınak Yöneticisi (Shelter Admin) ve Yönetici (Admin).

Kullanıcı, platformun en genel ve temel kullanıcısıdır. Sisteme kayıt olabilir, giriş yapabilir ve hayvan ilanlarını görüntüleyip çeşitli filtreleme seçenekleriyle inceleyebilir. Ayrıca blog içeriklerini okuyabilir, bağış sayfası aracılığıyla sivil toplum kuruluşlarına yönlendirilerek bağış yapabilir ve harita üzerinden barınakların konumlarını inceleyebilir. Ancak ilan oluşturma ya da blog yazısı paylaşma gibi içerik üretme işlemleri kullanıcılar için kısıtlanmıştır.

Barınak Yöneticisi (Shelter) rolü, hayvan barınaklarını temsil eden kullanıcılardan oluşur. Bu kullanıcılar, yeni ilanlar oluşturabilir, mevcut ilanlarını düzenleyebilir veya silebilir. Kendi profillerinde yalnızca kendilerine ait ilanların listesini görüntüleyebilirler. Ancak blog içerikleri üzerinde herhangi bir yetkileri bulunmamaktadır ve yalnızca kendi oluşturdukları içerikler üzerinde işlem yapabilirler.

Yönetici (Admin), platformun en yetkili kullanıcısıdır ve tüm sistemin yönetiminden sorumludur. Tüm kullanıcıları ve ilanları görüntüleyebilir, düzenleyebilir veya silebilir. Ayrıca blog yazıları oluşturabilir, mevcut yazıları düzenleyip silebilir. Platform genelindeki moderasyon işlemleri de yöneticiler tarafından gerçekleştirilir. Giriş yapan kullanıcıların rollerini kontrol edebilme yetkisi olan adminler, güvenlik gereği kullanıcı parolalarına erişemezler.

## BÖLÜM 2: PROJE PLANI

---

### 2.1 Kullanılan Teknolojiler ve Araçlar:

#### Frontend Teknolojileri:

- HTML
- CSS
- JavaScript
- 

#### Backend Teknolojileri:

- Node.js
- Express.js
- MongoDB
- JWT (JSON Web Token)
- Swagger (API dokümantasyonu için)

## 2.2 Proje Planı:

Bu proje, hayvan barınaklarında bulunan sahipsiz hayvanların, web tabanlı bir platform aracılığıyla daha kolay ve etkili bir şekilde sahiplendirilmesini amaçlamaktadır. Proje kapsamında geliştirilen sistem; kullanıcıların hayvan ilanlarını görüntülemesi, sahiplendirme, bağış yapması, blog içeriklerine ulaşması ve harita üzerinden barınakları incelemesi gibi işlevleri barındırmaktadır. Planlama süreci aşağıdaki aşamalara göre gerçekleştirilmiştir:

### Analiz ve İhtiyaç Belirleme:

- Barınaklar ve bireysel kullanıcılar için dijital bir köprü kurularak sahiplendirme süreci kolaylaştırılacaktır.
- Mevcut benzer uygulamalar ( Haytap, Petfinder ) incelenerek eksik ve güçlü yönleri analiz edilmiştir.
- Kullanıcı türleri tanımlanmıştır: User (bireysel kullanıcı), Shelter (barınak yöneticisi) ve Admin (yönetici).
- Sistemin temel işlevleri ve kullanıcı arayüzleri belirlenmiştir.

### Tasarım Süreci:

- Web sitesinin kullanıcı dostu, erişilebilir olması hedeflenmiştir.
- Giriş yap, kayıt ol, ana sayfa, ilanlar, profil, bağış yap, blog ve harita sayfalarının arayüz tasarımları hazırlanmıştır.
- Arayüz bileşenleri ve kullanıcı deneyimi iyileştirmeleri detaylandırılmıştır.



## Geliştirme Aşaması:

### Frontend:

- Arayüz tasarımı için HTML, CSS, JavaScript kullanılmıştır.
- Sayfalar arasında dinamik geçişler sağlanmış ve kullanıcı deneyimi öncelikli tutulmuştur.

### Backend:

- Sunucu tarafı işlemleri Node.js ve Express.js ile gerçekleştirilmiştir.
- Veritabanı işlemleri için MongoDB tercih edilmiştir.
- Kimlik doğrulama işlemleri JWT (JSON Web Token) ile güvenli hale getirilmiştir.
- API dokümantasyonu Swagger ile hazırlanmıştır.

## BÖLÜM 3: SİSTEM ANALİZİ VE TASARIMI

---

### 3.1 Kullanıcı Gereksinimleri :

Sistem, farklı kullanıcı rollerine (Genel Kullanıcı, Barınak Yöneticisi ve Yönetici) özel olarak çeşitli işlevler sunmalıdır. Tüm kullanıcılar sisteme kayıt olup giriş yapabilmelidir. Genel kullanıcılar, profil bilgilerini düzenleyebilmeli, sahipsiz hayvan ilanlarını görüntüleyebilmeli, harita üzerinden barınak konumlarını görebilmeli, blog içeriklerini okuyabilmeli ve bağış yapabilmelidir. Barınak yöneticileri , kendilerine ait hesaplarla sisteme giriş yaparak barınaktaki sahipsiz hayvanlar için ilan oluşturabilmeli, bu ilanları güncelleyip silebilmelidir.

Yönetici (Admin) rolündeki kullanıcılar ise tüm kullanıcıların bilgilerine erişebilmeli, sistemdeki blog içeriklerini yönetebilmelidir. Bu gereksinimler, sistemin kullanıcı dostu, işlevsel ve erişilebilir bir yapıda olmasını sağlamayı amaçlamaktadır.

### 3.2 Sistem Gereksinimleri :

#### Yazılım Gereksinimleri:

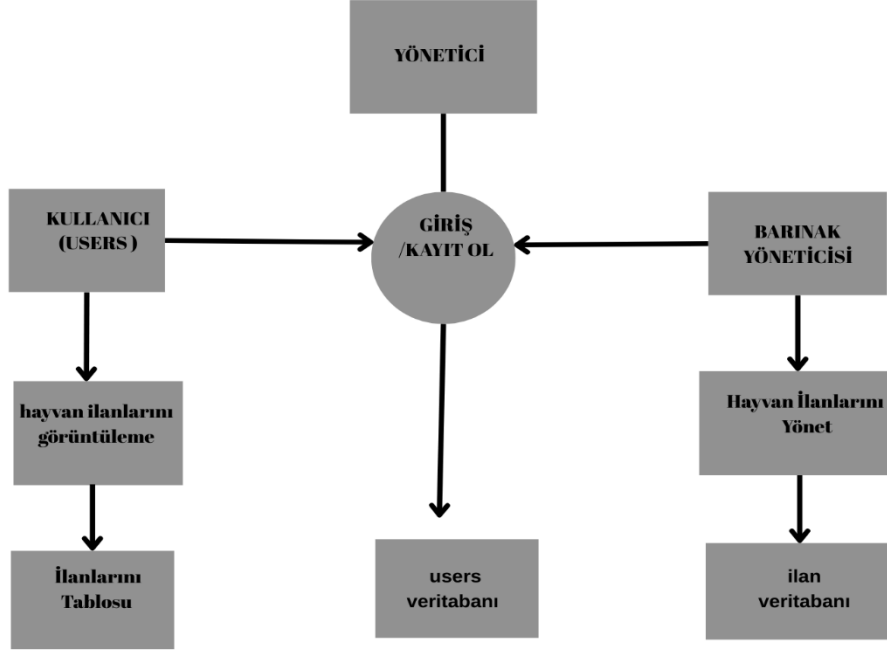
#### Frontend (Kullanıcı Arayüzü)

- HTML
- CSS
- JavaScript

#### 3.3 Backend (Sunucu Tarafı)

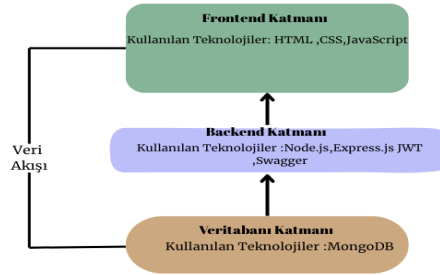
- Express.js - Web sunucu framework'ü
- Node.js – Sunucu tarafı çalıştırma ortamı
- MongoDB- NoSQL veritabanı
- Mongoose- – MongoDB ile çalışmak için ORM kütüphanesi
- JWT (JSON Web Token) - Kimlik doğrulama ve yetkilendirme
- Swagger- API dokümantasyonu için

### 3.4 Veri Akış Diyagramları (DFD):



Şekil 1 – Veri Akış Diyagramı

### 3.5 Sistem Mimarisi :



Şekil 2 – Sistem Mimarisi

**3 Katmanlı Mimari (Three-Tier Architecture)**, yazılım geliştirme sürecinde yaygın olarak kullanılan, uygulama bileşenlerini sunum Frontend, Backend ve Veritabanı katmanlarına ayıran bir mimari modeldir. Bu yapı, uygulamanın modülerliğini artırır, bakımını kolaylaştırır ve geliştirme sürecinde esneklik sağlar.

### **Avantajları:**

- **Modülerlik:** Her katman bağımsız geliştirilebilir ve güncellenebilir.
- **Bakım Kolaylığı:** Hatalar izole katmanlarda bulunabilir.
- **Yeniden Kullanılabilirlik:** Aynı iş mantığı farklı arayüzlerle (web, mobil) kullanılabilir.

### **Dezavantajları:**

- Karmaşık küçük projelerde gereksiz mimari yük oluşturabilir.
- Katmanlar arası geçişler bazen performans kaybı yaratabilir.

#### **1. Frontend Katmanı :**

- Kullanıcı ile doğrudan etkileşime giren katmandır.
- HTML, CSS, JavaScript teknolojileri kullanıldı

#### **2. Backend Katmanı:**


- Uygulamanın kurallarını ve karar mekanizmalarını barındırır.
- Node.js , Express.js JWT ,Swagger kullanıldı .

#### **3. Veritabanı Katmanı:**

- Veritabanı işlemlerini gerçekleştiren katmandır.
- MongoDB veritabanı kullanıldı. [\[22\]](#) [\[23\]](#)

## 3.6 Veritabanı Tasarımı

### 3.6.1. Kullanıcı Tablosu

users
 _id
firstName
lastName
email
password
role
birthDate
isAdmin
createdAt

Şekil 3 -Kullanıcı Tablosu

### 3.6.2 İlan Tablosu:

ilan
 _id
cins
yas
cinsiyet
saglikDurumu
karakterOzellikleri
bulunduguYer
iletisimNo
hikaye

Şekil 4 -İlan Tablosu

### 3.6.3. Blog Tablosu:

blog
Write here
blogNo
title
description
content

Şekil 5 – Blog Tablosu

- Veritabanı tabloları Db Schema kullanılarak yapıldı . DbSchema, görsel veritabanı şeması tasarım aracıdır ve MongoDB veritabanı ile uyumlu çalışır. Şema oluşturma, sorgu yazma ve şema üzerinde değişiklik yapma işlemleri için kullanılır

## BÖLÜM 4: API TASARIMI

---

### 4.1 API Tasarımı Nedir?

API tasarımı, bir yazılımın dış dünyaya hangi veri ve işlevleri nasıl sunacağını planlama sürecidir. Bu süreçte amaç, geliştiricilerin ihtiyaç duyduğu bilgilere ve işlemlere en anlaşılır, tutarlı ve kullanılabilir şekilde ulaşmasını sağlamaktır. İyi planlanmış bir API, sahip olduğu endpoint'ler (erişim yolları), HTTP metodları (GET, POST vb.) ve veri yapılarıyla, ne sunduğunu açıkça belli eder.

Başarılı bir API tasarımı yalnızca kullanıcıya değil, geliştirici ekipler açısından da büyük kolaylık sağlar. Çünkü doğru tasarlanmış bir API; hem iş ihtiyaçlarına uyum sağlar hem de kullanım kolaylığı, esneklik, test edilebilirlik ve güçlü dokümantasyon gibi yazılım kalitesini artıran unsurları beraberinde getirir. Bu nedenle API'nin tasarımı, projenin ilk aşamalarında ele alınmalı; böylece ekip içerisindeki tüm paydaşlar ortak bir anlayışta buluşabilir ve ileride çıkabilecek sorunların önüne erkenden geçilebilir.

Ayrıca, API tasarımı büyük projelerde ya da kurumsal yapılarda sürdürülebilirliği sağlamak açısından da önemlidir. Tekrarlanabilir, standart bir yapı ortaya koymak; farklı projelerde yeniden kullanılacak ortak desenlerin (pattern'ların) oluşmasına katkı sağlar. Bu da kurumsal API stratejilerinin daha düzenli, tutarlı ve yönetilebilir olmasını mümkün kılar. [4]

### 4.2 Swagger Nedir?

Swagger, bir API'nin yapısını açık bir şekilde tanımlamanıza olanak sağlayan bir araçtır. Temel amacı, API'lerin nasıl çalıştığını hem insanlar hem de makineler için anlaşılır hale getirmektir. Bu yapı sayesinde, bir API'nin sunduğu tüm işlevler ve veri yapıları standart bir biçimde tanımlanabilir ve bu bilgiler otomatik olarak kullanılabilir hale gelir.

Ayrıca Swagger, sadece dokümantasyon oluşturmakla kalmaz; API'ye özel istemci kütüphanelerini (client libraries) farklı programlama dillerinde otomatik olarak üretebilir. Bunun yanında, test süreçlerinde kullanılacak altyapılar oluşturmak veya API'yi keşfetmek gibi birçok farklı alanda da fayda sağlar.



Swagger bunu, API'nizden aldığı **JSON** ya da **YAML** formatındaki detaylı bir yapı tanımı (specification) sayesinde yapar. Bu dosya, API'nin sunduğu tüm endpoint'leri, veri tiplerini ve işlevleri sistemli bir şekilde içerir. Kısacası Swagger, bir API'yi sadece tanımlamakla kalmaz; onunla çalışmayı da çok daha verimli hale getirir. [5]

#### 4.2.1. Swagger Dökümanı Oluşturma Süreci

- Swagger kurulumu: Projede Swagger kullanabilmek için swagger-ui-express ve swagger-jsdoc gibi kütüphaneleri yükledik.

```
Problems 10 Output Debug Console Terminal Ports Spell Checker 10
PS C:\Users\remzi\OneDrive\Masaüstü\Sahiplendirme-FullStack-main> npm install swagger-ui-express swagger-jsdoc
```

Şekil 6 -Swagger Kodu 1

- Projeye Swagger entegrasyonu yapmak :Öncelikle swagger.js adında bir dosya oluşturduk.Bu dosya API 'mizin ayrıntılarını belirttiği bir dosyadır. Bu dosya API sürümü, başlık, açıklama ve istek ve yanıt parametreleriyle API endpointleri gibi bilgileri içerir. Daha sonra app.js dosyamızın içerisine swagger eklentilerini yaptık. [20]

```
import swaggerJsdoc from 'swagger-jsdoc';

const options = {
  definition: {
    openapi: '3.0.0',
    //BİLGİLER
    info: {
      title: 'PawFinder API',
      version: '1.0.0',
      description: 'Hayvan sahiplendirme platformu API dokümantasyonu',
    },
    //Sunucu bilgileri
    servers: [
      {
        url: 'http://localhost:4000',
        description: 'Geliştirme sunucusu',
      },
    ],
    //API'de kullanılacak kimlik doğrulama yöntemlerini belirtiyoruz
    components: {
      securitySchemes: {
        bearerAuth: {
          type: 'http', //HTTP kimlik doğrulama yöntemi
          scheme: 'bearer', //Bearer token kimlik doğrulama yöntemi
          bearerFormat: 'JWT', //Token formatı JWT
        },
      },
    },
  },
  apis: ['./app.js', './docs/api-docs.js'], // API rotalarının ve dokümantasyonun bulunduğu dosyalar
};

const specs = swaggerJsdoc(options); //Yukarıda belirtilen options ile swagger yapısını oluşturunuz.
export default specs; //swagger yapısını dışarıdan kullanabilmek için export ediyoruz.
```

Şekil 7 – Swagger Kodu 2

```
import swaggerUi from 'swagger-ui-express';
import specs from './docs/swagger.js'; //swagger.js dosyasındaki swagger yapısını kullanıyoruz
```

Şekil 8 -Swagger Kodu 3

```
// Swagger UI
app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(specs));
```

Şekil 9 – Swagger Kodu 4

```
// Sunucuyu başlat
const server = app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
  console.log(`API Dökümanı http://localhost:\${PORT}/api-docs`);
})
```

Şekil 10 – Swagger Kodu 5

- Swagger dokümantasyonunu özelleştirmek: Swagger entegrasyonu oluşturduktan sonra, API'mizi daha anlaşılır hale getirebilmek için aşağıdaki öğeleri özelleştirdik:
  - **Schema:** API'nin veri yapısını tanımladık. Bu sayede, API'ye gelen ve giden veriler açıkça tanımlandı ve kullanıcılar hangi formatlarda veri göndermeleri gerektiğini kolayca görebiliyorlar. [20]

```

/**
 * @swagger
 * components:
 *   schemas:
 *     User:
 *       type: object
 *       required:
 *         - firstName
 *         - lastName
 *         - email
 *         - password
 *         - birthDate
 *       properties:
 *         firstName:
 *           type: string
 *         lastName:
 *           type: string
 *         email:
 *           type: string
 *           format: email
 *         password:
 *           type: string
 *           minLength: 6
 *         birthDate:
 *           type: string
 *           format: date
 *
 * Ad:
 *   type: object
 *   required:
 *     - cins
 *     - yas
 *     - cinsiyet
 *     - saglik_durumu
 *     - karakter_ozellikleri
 *     - bulundugu_yer
 *     - iletisim_no
 *     - hikaye
 *     - resim_url
 *   properties:
 *     cins:
 *       type: string
 *     yas:
 *       type: number
 *     cinsiyet:
 *       type: string
 *       enum: [Erkek, Dişi]
 *     saglik_durumu:
 *       type: string
 *     karakter_ozellikleri:
 *       type: string
 *     bulundugu_yer:
 *       type: string
 *     iletisim_no:
 *       type: string
 *     hikaye:
 *       type: string
 *     resim_url:
 *       type: string
 *
 * Blog:
 *   type: object
 *   required:
 *     - title
 *     - description
 *     - content
 *     - imageUrl
 *     - olusturan
 *   properties:
 *     title:
 *       type: string
 *     description:
 *       type: string
 *     content:
 *       type: string
 *     imageUrl:
 *       type: string
 *     olusturan:
 *       type: string
 *       description: Blog yazısını olusturan kullanıcının adı ve soyadı

```

Şekil 11 – Swagger Kodu 6

**Tags:** API'nin bölümlerini belirginleştirdik. Örneğin, kullanıcı işlemleri, admin işlemleri gibi farklı kategorilere ayırarak, her kategorinin altında ilgili endpoint'leri topladık.

```

/**
 * @swagger
 * tags:
 *   - name: Auth
 *     description: Kimlik doğrulama işlemleri
 *   - name: Admin
 *     description: Admin işlemleri
 *   - name: Kullanıcı
 *     description: Kullanıcı işlemleri
 *   - name: İlanlar
 *     description: İlan işlemleri
 *   - name: Blog
 *     description: Blog işlemleri
 */

```

Şekil 12 – Swagger Kodu 7

- **SecuritySchemes:** Bu adımda API'yi kullanacak kullanıcıların güvenli bir şekilde kimlik doğrulaması yapabilmesi için **JWT (JSON Web Token)** kullanıldı. Swagger dökümantasyonunda güvenlik şeması tanımlayarak, API'nin kimlik doğrulama mekanizmasını açıkça belirttik

```
/**
 * @swagger
 * components:
 *   securitySchemes:
 *     bearerAuth:
 *       type: http
 *       scheme: bearer
 *       bearerFormat: JWT
 */
```

Şekil 13 – Swagger Kodu 8

- Swagger ile API endpoint'lerini dökümanete etmek :Bu adımda, tüm API endpoint'lerinin Swagger üzerinden doğru şekilde dökümanete edilmesini sağladık. Her bir endpoint'in açıklamaları, parametreleri ve dönüş değerleriyle ilgili bilgiler Swagger şemalarında yer alıyor.

```
/**
 * @swagger
 * /api/register:
 *   post:
 *     summary: Yeni kullanıcı kaydı
 *     tags: [Auth]
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/User'
 *     responses:
 *       201:
 *         description: Kullanıcı başarıyla oluşturuldu
 *       400:
 *         description: Gecersiz giriş
 */

/**
 * @swagger
 * /api/login:
 *   post:
 *     summary: Kullanıcı girişi
 *     tags: [Auth]
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             type: object
 *             required:
 *               - email
 *               - password
 *             properties:
 *               email:
 *                 type: string
 *                 format: email
 *               password:
 *                 type: string
 *     responses:
 *       200:
 *         description: Başarılı giriş
 *       401:
 *         description: Gecersiz kimlik bilgileri
 */
```

Şekil 14 – Swagger Kodu 9 (Örnek endpoint)

### 4.3 Kullanıcı Yönetimi

Kullanıcı yönetimi modülü, sistemdeki kullanıcıların kimlik doğrulama, yetkilendirme ve profil yönetimi işlemlerini gerçekleştirmek için tasarlanmıştır. Bu modül, kullanıcıların sisteme güvenli bir şekilde erişmesini sağlar ve farklı rollerle (örneğin, admin, shelter, user) yetkilendirme işlemlerini destekler. Kullanıcı yönetimi, sistemin temel yapı taşlarından biri olup, kullanıcıların yetkilerine göre farklı işlemleri gerçekleştirmelerine olanak tanır.

### 4.4 Kullanıcı İşlemleri

Kayıt Olma:

Kullanıcılar sisteme kayıt olduklarında /api/register endpoint'i kullanılır. Gerekli bilgiler: ad, soyad, e-posta, şifre ve doğum tarihidir. Şifreler, güvenlik amacıyla bcrypt kütüphanesi kullanılarak hashlenir ve veritabanında bu şekilde saklanır. Kayıt işlemi sırasında e-posta adresinin benzersizliği kontrol edilir; aynı e-posta adresiyle birden fazla kayıt yapılamaz.

Giriş Yapma:

Kullanıcılar sisteme giriş yaptıklarında /api/login endpoint'i kullanılır. Giriş işlemi sırasında e-posta ve şifre doğrulanır. Doğrulama başarılı olursa, kullanıcıya bir JWT (JSON Web Token) döndürülür. JWT, kullanıcının kimliğini doğrulamak ve yetkilendirme işlemlerinde kullanılmak üzere istemci tarafında saklanır.

Profil Görüntüleme ve Güncelleme:

Kullanıcıların kendi profil bilgileri /api/profile endpoint'i üzerinden görüntülenir. Profil bilgileri, şifre hariç tüm kullanıcı bilgilerini içerir. Kullanıcılar, profil bilgilerini güncellemek istediklerinde /api/users/:id endpoint'ini kullanabilir; ancak yalnızca kendi bilgilerini güncelleme yetkisine sahiptirler.

Şifre Değiştirme:

Kullanıcılar, mevcut şifrelerini doğruladıktan sonra yeni bir şifre belirlemek için `/api/change-password` endpoint'ini kullanılır. Yeni şifre, güvenlik amacıyla hashlenerek veritabanında saklanır.

**Kullanıcı Silme:**

Kullanıcılar, kendi hesaplarını silebilir veya admin kullanıcılar diğer kullanıcıları silebilir. Bu işlem, `/api/users/:id` endpoint'i üzerinden gerçekleştirilir.

#### 4.4.1 Teknik Detaylar

##### **Kimlik Doğrulama:**

JWT, yani JSON Web Token, web uygulamalarında kullanıcı doğrulamasını güvenli ve pratik bir şekilde sağlamak için kullanılan bir yöntemdir. Sunucu, kullanıcıyı doğruladıktan sonra ona bir token (jeton) verir ve bu token, sonraki isteklerde kullanılarak kimlik doğrulaması yapılır. Token, üç parçadan oluşur: başlık (header), içerik (payload) ve imza (signature). Bu yapı sayesinde hem veriler taşınabilir hem de güvenliği sağlanabilir. JWT'nin en büyük avantajlarından biri, stateless (durumsuz) olmasıdır; yani sunucunun oturum bilgisi tutmasına gerek kalmaz. Bu da ölçeklenebilir sistemler için oldukça kullanışlıdır.

- Tüm korumalı endpoint'lerde `Authorization: Bearer <token>` başlığı gereklidir.
- Token, kullanıcının kimliğini doğrulamak için sunucu tarafında `jsonwebtoken` kütüphanesi ile doğrulanır. [6]

##### **Yetkilendirme:**

- Kullanıcıların belirli işlemleri gerçekleştirebilmesi için yetkilendirme kontrolü yapılır.
- Örneğin, admin yetkisi gerektiren işlemler `isAdmin` middleware'i ile kontrol edilir.
- Kullanıcıların kendi ilanlarını veya hesaplarını yönetebilmesi için ek kontroller yapılır.

##### **Veri Güvenliği:**

- Şifreler, `bcrypt` kütüphanesi ile hashlenerek saklanır. Bu, şifrelerin düz metin olarak saklanmasını engeller ve güvenliği artırır.
- Kullanıcı bilgileri, yalnızca gerekli alanlar döndürülerek istemciye gönderilir. Örneğin, şifre alanı hiçbir zaman istemciye gönderilmez.

## 4.5 İlan Yönetimi

İlan yönetimi modülü, hayvan barınaklarının sahiplendirme ilanlarını oluşturmaya, düzenlemesine ve silmesine olanak tanır. Bu modül, kullanıcıların hayvan sahiplenme sürecini kolaylaştırmak ve barınaklarla iletişim kurmalarını sağlamak için tasarlanmıştır. İlan yönetimi, sistemin temel işlevlerinden biri olup, kullanıcıların hayvanlar hakkında bilgi edinmesini ve sahiplendirme sürecine katılmasını sağlar.

### 4.5.1 İlan Yönetimi İşlevleri

İlan yönetimi modülü, aşağıdaki temel işlevleri sağlar:

- İlan Oluşturma:

- Barınak sahipleri (shelter rolüne sahip kullanıcılar), `/api/ads` endpoint'i üzerinden yeni bir ilan oluşturabilir.
- İlan oluşturma işlemi sırasında hayvanın cinsi, yaşı, cinsiyeti, sağlık durumu, iletişim bilgileri ve görsel URL gibi bilgiler sağlanır.
- İlanlar, veritabanında `Ad` modeli kullanılarak saklanır.

- İlan Listeleme:

- Tüm kullanıcılar, `/api/ads` endpoint'i üzerinden mevcut ilanları listeleyebilir.
- Barınak sahipleri, yalnızca kendi ilanlarını listelemek için `/api/ads/my-ads` endpoint'ini kullanabilir.

- İlan Detay Görüntüleme:

- Kullanıcılar, `/api/ads/:ilan\_no` endpoint'i üzerinden belirli bir ilan detayını görüntüleyebilir.
- İlan detayları, hayvanın özelliklerini ve iletişim bilgilerini içerir.

- İlan Güncelleme:

- Barınak sahipleri, `/api/ads/:ilan\_no` endpoint'i üzerinden mevcut bir ilanı güncelleyebilir.
- Güncelleme işlemi sırasında yalnızca ilanı oluşturan kullanıcı işlem yapabilir.

- İlan Silme:

- Barınak sahipleri, `/api/ads/:ilan\_no` endpoint'i üzerinden bir ilanı silebilir.
- Silme işlemi, yalnızca ilanı oluşturan kullanıcı tarafından gerçekleştirilebilir.



#### 4.5.2 Teknik Detaylar

- Veri Modeli:

- İlanlar, `Ad` modeli kullanılarak MongoDB'de saklanır. Modelin temel alanları şunlardır:

- `cins`: Hayvanın cinsi.
- `yas`: Hayvanın yaşı.
- `cinsiyet`: Hayvanın cinsiyeti.
- `saglik\_durumu`: Hayvanın sağlık durumu.
- `bulundugu\_yer`: Hayvanın bulunduğu yer.
- `iletisim\_no`: İletişim numarası.
- `resim\_url`: Hayvanın görsel URL'si.
- `olusturan`: İlanı oluşturan kullanıcının e-posta adresi.

- Yetkilendirme:

- İlan oluşturma, güncelleme ve silme işlemleri yalnızca shelter rolüne sahip kullanıcılar tarafından yapılabilir.

- Yetkilendirme işlemleri, `checkAdPermissions` middleware'i ile kontrol edilir. Bu middleware, kullanıcının shelter rolüne sahip olup olmadığını doğrular.

- Kimlik Doğrulama:

- İlan yönetimi işlemleri için JWT (JSON Web Token) kullanılarak kimlik doğrulama yapılır.
- Tüm korumalı endpoint'lerde `Authorization: Bearer <token>` başlığı gereklidir.

- Frontend İşlevselliği

- İlanlar, kullanıcı arayüzünde dinamik olarak görüntülenir.
- İlan detay sayfası, `/ad-detail.html` dosyasında dinamik olarak oluşturulur ve API'den alınan verilerle doldurulur.

## 4.6 Blog Yönetimi

Blog yönetimi modülü, sistemdeki admin kullanıcıların bilgilendirici içerikler oluşturmaya, düzenlemesine ve silmesine olanak tanır. Bu modül, kullanıcıların hayvan sahiplenme süreçleri, barınaklar ve hayvan bakımı gibi konularda bilgi edinmelerini sağlar. Blog yönetimi, kullanıcıların sisteme olan bağlılığını artırmak ve bilgilendirici içerikler sunmak için önemli bir rol oynar.

### 4.6.1 Blog Yönetimi İşlevleri

Blog yönetimi modülü, aşağıdaki temel işlevleri sağlar:

- Blog Oluşturma:

- Admin kullanıcılar, `/api/blogs` endpoint'i üzerinden yeni bir blog yazısı oluşturabilir.
- Blog oluşturma işlemi sırasında başlık, açıklama, içerik ve görsel URL gibi bilgiler sağlanır.
- Blog yazıları, veritabanında `Blog` modeli kullanılarak saklanır.

- Blog Listeleme:

- Tüm kullanıcılar, `/api/blogs` endpoint'i üzerinden mevcut blog yazılarını listeleyebilir.
- Bloglar, oluşturulma tarihine göre sıralanır ve en güncel yazılar en üstte gösterilir.

- Blog Detay Görüntüleme:

- Kullanıcılar, `/api/blogs/:blogNo` endpoint'i üzerinden belirli bir blog yazısının detaylarını görüntüleyebilir.

- Blog detayları, başlık, açıklama, içerik ve görsel URL gibi bilgileri içerir.

- Blog Güncelleme:

- Admin kullanıcılar, `/api/blogs/:blogNo` endpoint'i üzerinden mevcut bir blog yazısını güncelleyebilir.

- Güncelleme işlemi sırasında yalnızca yetkili kullanıcılar işlem yapabilir.

- Blog Silme:

- Admin kullanıcılar, `/api/blogs/:blogNo` endpoint'i üzerinden bir blog yazısını silebilir.
- Silme işlemi, yalnızca yetkili kullanıcılar tarafından gerçekleştirilebilir.

#### 4.6.2 Teknik Detaylar

- Veri Modeli:

- Blog yazıları, `Blog` modeli kullanılarak MongoDB'de saklanır. Modelin temel alanları şunlardır:

- `title`: Blog başlığı.
- `description`: Blog açıklaması.
- `content`: Blog içeriği.
- `imageUrl`: Blog görselinin URL'si.
- `createdAt`: Blogun oluşturulma tarihi.
- `updatedAt`: Blogun güncellenme tarihi.

- Yetkilendirme:

- Blog oluşturma, güncelleme ve silme işlemleri yalnızca admin kullanıcılar tarafından yapılabilir.
- Yetkilendirme işlemleri, `checkBlogPermissions` middleware'i ile kontrol edilir. Bu middleware, kullanıcının admin olup olmadığını doğrular.

- Kimlik Doğrulama:

- Blog yönetimi işlemleri için JWT (JSON Web Token) kullanılarak kimlik doğrulama yapılır.
- Tüm korumalı endpoint'lerde `Authorization: Bearer <token>` başlığı gereklidir.

## BÖLÜM 5: KULLANILAN TEKNOLOJİLER

---

### 5.1. HTML Nedir?

HTML (HyperText Markup Language), web tabanlı sistemlerin kullanıcıya sunulan ön yüz yapısını oluşturan temel işaretleme dilidir. 1990'lı yıllarda Tim Berners-Lee tarafından geliştirilen HTML, günümüzde tüm web sayfalarının iskeletini oluşturan standart teknolojidir. HTML, bir sayfanın yapısal bileşenlerini (başlık, paragraf, liste, görsel, bağlantı vb.) tanımlamak amacıyla kullanılır ve tarayıcılar tarafından yorumlanarak kullanıcıya görsel bir arayüz sunar. HTML'in semantik yapısı sayesinde, sayfa içeriği hem insanlar hem de makineler (arama motorları, ekran okuyucular) tarafından anlaşılabilir hâle gelir. Projemizde HTML, kullanıcı arayüzünün iskeletini oluşturmak amacıyla kullanılmıştır. Sayfa başlıkları, butonlar, görseller, formlar ve listelemeler gibi tüm yapısal öğeler HTML ile tanımlanmış, böylece kullanıcıya sade, anlaşılır bir arayüz sunulmuştur. Ayrıca HTML'in semantik yapısı sayesinde, içerikler arama motorları ve yardımcı teknolojiler tarafından da kolaylıkla erişilebilir hâle getirilmiştir.

#### Avantajları:

- Yaygın destek ve kolay öğrenilebilirlik
- Tüm tarayıcılarla uyumludur
- Semantik yapı ile SEO ve erişilebilirlik avantajı sağlar

#### Dezavantajları:

Yalnızca yapısal öğeleri tanımlar, etkileşim veya görsellik için CSS/JS desteğine ihtiyaç duyar [7] [8]

### 5.2. CSS Nedir?

CSS (Cascading Style Sheets), HTML ile oluşturulan yapısal içeriklerin görsel olarak düzenlenmesini sağlayan stil tanımlama dilidir. 1996 yılında W3C tarafından standartlaştırılan CSS, bir web sayfasının renk, yazı tipi, hizalama, kenar boşlukları, geçiş efektleri ve ekran boyutuna duyarlılık gibi tüm görsel

özelliklerini tanımlar. Gelişen CSS3 ile birlikte animasyonlar, medya sorguları ve responsive (duyarlı) tasarımlar mümkün hâle gelmiştir. Projemizin ön yüzünde, kullanıcı arayüzü anlaşılabilir bir şekilde olması adına stil bütünlüğü, okunabilirlik ve mobil uyumluluk CSS aracılığıyla sağlanmıştır. CSS,HTML yapısının görsel açıdan düzenlenmesi, responsive ve modern arayüz tasarımları için kullanıldı.

#### **Avantajları:**

- Sayfa tasarımını kolayca değiştirme imkânı
- Medya sorguları ile mobil uyumlu (responsive) tasarım
- Animasyon ve geçiş efektleriyle zengin kullanıcı deneyimi

#### **Dezavantajları:**

- Büyük projelerde karmaşık hale gelebilir
- Tarayıcı uyumsuzlukları yaşanabilir [9] [10]

### **5.3. JavaScript Nedir?**

JavaScript, web sayfalarına dinamiklik ve etkileşim kazandırmak amacıyla kullanılan istemci taraflı bir programlama dilidir. 1995 yılında Brendan Eich tarafından geliştirilen JavaScript, zamanla yalnızca basit animasyonları değil, büyük ölçekli web uygulamalarının tüm mantıksal süreçlerini yönetebilecek seviyeye ulaşmıştır. JavaScript sayesinde sayfa içi olaylara (buton tıklamaları, form kontrolleri, API veri çekme işlemleri) tepki verilebilir. React Native gibi modern kütüphaneler de JavaScript temelli olduğu için, projede hem ön yüz bileşenlerinin yönetimi hem de Spotify API ile olan etkileşimlerin sağlanmasında JavaScript aktif olarak kullanılmıştır. Böylece, kullanıcıların uygulama içi deneyimi hem görsel hem de işlevsel açıdan zenginleştirilmiştir. Projemizde JavaScript tercih edilmesinin temel nedeni, web arayüzüne dinamiklik kazandırma ve kullanıcı etkileşimini artırma yeteneğidir.

#### **Avantajları:**

- Gerçek zamanlı etkileşim sağlar (form doğrulama, API çağrıları).
- Yaygın destek, büyük topluluk ve zengin kütüphane ekosistemi.

#### **Dezavantajları:**

- Hatalı kodlar tarayıcıda sorun yaratabilir.
- Güvenlik açıklarına karşı dikkatli kullanılmalıdır [11] [12]

## 5.4. Backend Teknolojileri

Backend teknolojileri, bir web veya mobil uygulamanın sunucu tarafında çalışan; istemciden gelen istekleri karşılayan, veritabanı ile iletişim kurarak verileri yöneten ve iş mantığını yürüten bileşenler bütünüdür. Projemizde, yüksek performans, ölçeklenebilirlik ve JavaScript ekosistemine uyumluluk kriterleri göz önünde bulundurularak **Node.js** ve **Express.js** tercih edilmiştir. Bu ikili, hem geliştirme sürecinin hızlanmasını sağlamış hem de modüler mimari yaklaşımları destekleyerek ekip çalışmasına uygun bir altyapı sunmuştur.

### 5.4.1 Node.js

**Node.js**, Chrome'un V8 JavaScript motoru üzerine kurulmuş, açık kaynaklı ve asenkron bir çalışma ortamıdır. Tek iş parçacıklı yapısı ile olay tabanlı işlem modelini birleştirerek aynı anda çok sayıda isteği verimli şekilde işleyebilir. Bu sayede, yüksek trafik senaryolarında bile düşük gecikme süreleri elde edilmesine olanak tanır (Node.js Foundation, 2024). [\[13\]](#)

Projede, API uç noktalarının yönetimi, iş mantığının uygulanması ve veri işleme görevleri Node.js ile hayata geçirilmiştir. JavaScript'in sunucu ve istemci tarafında ortak kullanılabilmesi, kod paylaşımını ve hata ayıklamayı kolaylaştırmış, ekip içi bilgi transferini hızlandırmıştır. Ayrıca zengin modül ekosistemi sayesinde şifreleme, kimlik doğrulama ve dosya işlemleri gibi sık tekrarlanan ihtiyaçlar kısa sürede çözümlenebilmiştir.

Bu sebeple, uygulamanın hem hızlı yanıt verebilmesi hem de aynı dil altyapısı üzerinden geliştirme ekibinin verimliliğinin artması adına Node.js tercih edilmiştir.

#### Avantajları:

- Yüksek performans (asenكرون yapısı sayesinde)
- Tek dil (JavaScript) ile hem istemci hem sunucu tarafı geliştirilebilir
- Geniş modül desteği (npm)

### **Dezavantajları:**

- CPU yoğun görevlerde verim düşebilir
- Tek iş parçacıklı yapı karmaşık işlemlerde dikkat gerektirir

### **5.4.2 Express.js**

Express.js, Node.js üzerinde çalışan, minimal ve genişletilebilir bir web çatısıdır. HTTP isteği yönetimi, yönlendirme (routing) ve middleware katmanları aracılığıyla RESTful API geliştirmeyi kolaylaştırır. Modüler yapısı, farklı özelliklerin ayrı katmanlarda ele alınmasını sağlar ve kod tekrarını azaltır (Express.js Foundation, 2024). [14]

Projemizde kullanıcı yönetimi, ilan ve blog modülleri için API rotaları Express.js ile tanımlanmıştır. express.Router kullanımı ile modüller arası ayırım yapılmış; kimlik doğrulama, veri doğrulama ve hata yönetimi gibi işlemler middleware ile aracılık edilmiştir. jsonwebtoken, bcrypt ve express-validator gibi popüler paketler, güvenlik ve veri bütünlüğü konularında standart çözümler sunmuştur.

Bu sebeple, RESTful tasarım prensiplerine uygun, esnek ve genişletilebilir bir API yapısı oluşturmak için Express.js seçilmiştir.

### **Avantajları:**

- Hızlı geliştirme süreci
- Basit yapısı sayesinde öğrenmesi kolay
- Orta katman (middleware) desteği ile güçlü yapılandırma

### **Dezavantajları:**

- Büyük ölçekli projelerde yapı karmaşıklaşabilir
- Tam özellikli frameworklere göre bazı eksiklikler olabilir

#### Projedeki Kullanım Detayları

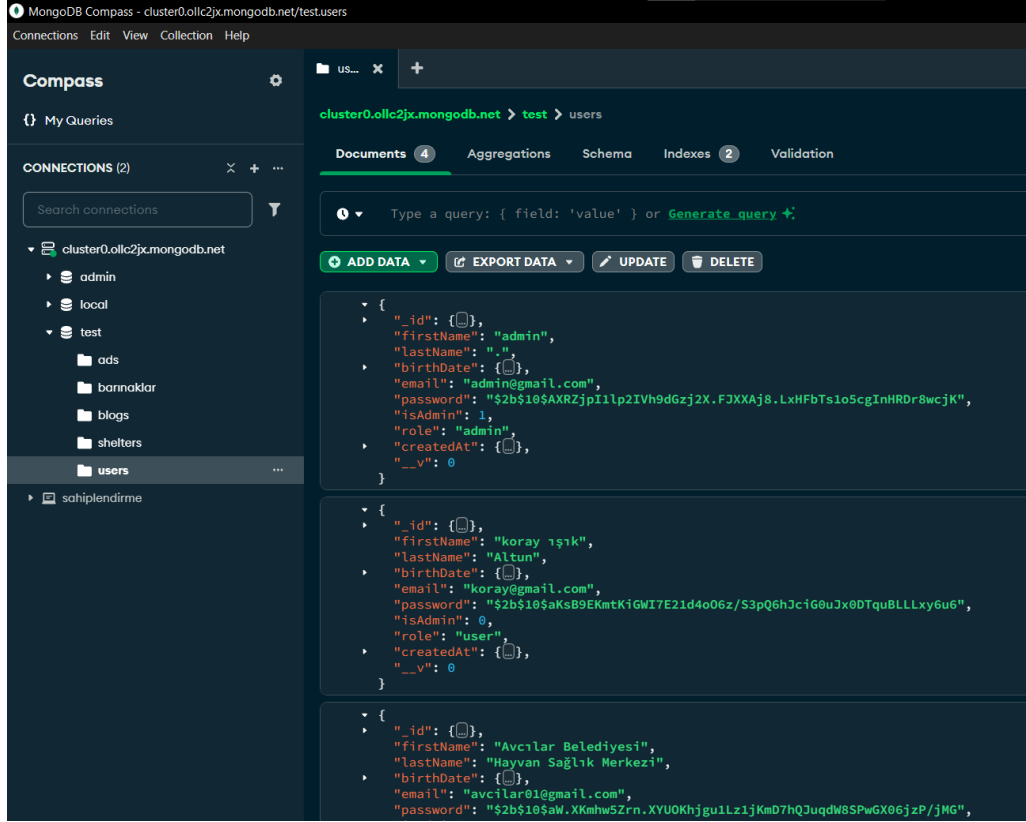
- Kimlik Doğrulama ve Yetkilendirme: JWT (JSON Web Token) kullanılarak kullanıcı oturumları yönetilmiş, farklı roller (admin, barınak, kullanıcı) middleware aracılığıyla denetlenmiştir.
- Veri Yönetimi: MongoDB veritabanı, mongoose kütüphanesi ile modellenmiş ve CRUD işlemleri RESTful API uç noktaları üzerinden gerçekleştirilmiştir.
- Hata ve Veri Doğrulama: Express'in hata yakalama mekanizmaları ile anlamlı geri bildirimler oluşturulmuş, express-validator ile istemci kaynaklı hatalı veri girişlerinin önüne geçilmiştir. .

[17]

### 5.5 Veritabanı Teknolojisi

Projede **MongoDB**, doküman odaklı NoSQL veritabanı olarak kullanılmıştır. MongoDB; esnek şema yapısı, JSON benzeri BSON formatında veri saklama imkânı ve yüksek yatay ölçeklenebilirlik özellikleri sayesinde, veri modellerimizi hızlıca evrimleştirmemize ve büyüyen veri hacmini etkin biçimde yönetmemize olanak tanımıştır. Ayrıca, Node.js ekosistemi ile sıkı entegrasyonu, mongoose gibi ODM kütüphaneleri aracılığıyla kolay model tanımlamaları ve gelişmiş sorgu yetenekleri sunmuştur. Bu sebeple, dinamik veri yapısına uyumluluk ve performans odaklı ölçeklenebilirlik ihtiyaçlarımız doğrultusunda MongoDB tercih edilmiştir.





Şekil 15 – MongoDB Görseli

MongoDB, projemizde veri yönetimi ve depolama ihtiyaçlarını karşılamak üzere tercih edilmiştir çünkü esnek ve ölçeklenebilir bir NoSQL veritabanı çözümü sunmaktadır. Belirli ve sabit bir şemaya bağlı kalma zorunluluğu olmadan, dinamik olarak değişebilen ilan verilerimizi kolaylıkla modellememize olanak tanımıştır. Bu durum, barınaklar tarafından oluşturulan sahiplendirme ilanlarının içerik ve yapılarındaki çeşitliliği esnek biçimde yönetmemizi sağlar .MongoDB'nin en büyük avantajlarından biri, JSON benzeri BSON formatında veri saklamasıdır; bu sayede JavaScript tabanlı uygulamalarla kullanılır. Bunun yanında MongoDB'nin dezavantajları da mevcuttur. Örneğin, çok karmaşık ilişkisel veri yapıları söz konusu olduğunda, klasik ilişkisel veritabanlarına göre daha fazla işlem yükü oluşturabilir. Ancak sahiplendirme ilanları gibi daha az kritik veri türleri için MongoDB'nin esnekliği ve hız avantajı, projeye büyük katkı sağlamıştır. [15] [18]

### 5.5.1. Diğer Araçlar

- **JWT (JSON Web Token):** Kimlik doğrulama ve yetkilendirme süreçlerinde stateless yapıda, güvenli token bazlı bir yöntem sunar. Sunucu tarafında oturum bilgisini saklamadan, imzalanmış ve doğrulanabilir token'lar kullanarak kullanıcı kimliğini güvenli biçimde yönetmemizi sağlamıştır.

Bu sebeple, ölçeklenebilir ve bakımı kolay bir kimlik doğrulama altyapısı oluşturmak için JWT kullanılmıştır. . [16]

- **Swagger (OpenAPI):** RESTful API'lerin otomatik olarak belgelenmesini ve interaktif bir test arayüzü sunmasını sağlayan açık kaynaklı bir araç setidir. Swagger sayesinde, backend'de tanımlı tüm uç noktalar ek ekip üyeleri tarafından kolayca keşfedilmiş, testler dokümana entegre edilerek hatalar hızlıca tespit edilmiştir. Bu sebeple, API uç noktalarının belgelendirilmesi ve ekip içi iş akışının verimliliğini artırmak için Swagger tercih edilmiştir. . [19]

## BÖLÜM 6: UYGULAMA MODÜLLERİ

### 6.1 Kullanıcı Yönetimi

#### 6.1.1 Kayıt Olma ve Giriş Yapma

Yuvamı Bul

ilanlar Blog Donate Harita Giriş Yap Kayıt Ol

**Kayıt Ol**

Ad

Soyad

E-posta

Şifre

gg.aa.yyyy

Kayıt Ol

Zaten hesabınız var mı? [Giriş Yap](#)

Şekil 16 – Kayıt Olma Arayüzü

```
143 // Register endpoint
144 app.post('/api/register', [
145   body('firstName').notEmpty().withMessage('Ad alanı boş bırakılamaz').trim(),
146   body('lastName').notEmpty().withMessage('Soyad alanı boş bırakılamaz').trim(),
147   body('email').isEmail().withMessage('Geçerli bir e-posta adresi giriniz'),
148   body('password').isLength({ min: 6 }).withMessage('Şifre en az 6 karakter olmalıdır'),
149   body('birthDate').notEmpty().withMessage('Doğum tarihi boş bırakılamaz')
150 ], async (req, res) => {
151   try {
152     const errors = validationResult(req);
153     if (!errors.isEmpty()) {
154       return res.status(400).json({ errors: errors.array() });
155     }
156
157     const { firstName, lastName, email, password, birthDate } = req.body;
158
159     // Email kontrolü
160     let user = await User.findOne({ email });
161     if (user) {
162       return res.status(400).json({ message: 'Bu email zaten kayıtlı' });
163     }
164
165     // Şifre hashleme
166     const salt = await bcrypt.genSalt(10);
167     const hashedPassword = await bcrypt.hash(password, salt);
168
169     // Yeni kullanıcı oluşturma
170     user = new User({
171       firstName,
172       lastName,
173       email,
174       password: hashedPassword,
175       birthDate: new Date(birthDate)
176     });
```

Şekil 17 – Kayıt Olma Kodu

Bu kod, kullanıcıların sisteme kayıt olmasını sağlar. Giriş verileri doğrulandıktan sonra e-posta kontrolü yapılır, şifre güvenli şekilde hashlenir ve kullanıcı bilgileri veritabanına kaydedilir.

## 6.1.2 Profil Güncelleme

The image shows a web interface for updating a user profile. It consists of three main panels. The top panel, titled 'Ad', 'Soyad', and 'E-posta', contains three rows of input fields with labels 'Ad', 'Soyad', and 'E-posta'. The values are 'Avcılar Belediyesi', 'Hayvan Sağlık Merkezi', and 'avcilar@gmail.com' respectively. Below this is a section for changing the password, titled 'Şifre Değiştir', which has three input fields: 'Mevcut Şifre', 'Yeni Şifre', and 'Yeni Şifre (Tekrar)'. To the right of this is a section for changing the email, titled 'E-posta Değiştir', which also has three input fields: 'Mevcut E-posta', 'Yeni E-posta', and 'Yeni E-posta (Tekrar)'. Each of these two sections has a blue button at the bottom: 'Şifreyi Güncelle' for the password section and 'E-postayı Güncelle' for the email section. At the bottom left of the entire interface is a red button labeled 'Çıkış Yap'.

Şekil 18 – Profil Güncelleme Arayüzü

```
102 // Şifre değiştirme
103 app.post('/api/change-password', authenticateToken, [
104   body('currentPassword').notEmpty().withMessage('Mevcut şifre gerekli'),
105   body('newPassword').isLength({ min: 6 }).withMessage('Yeni şifre en az 6 karakter olmalı'),
106 ], async (req, res) => {
107   try {
108     const errors = validationResult(req);
109     if (!errors.isEmpty()) {
110       return res.status(400).json({ errors: errors.array() });
111     }
112
113     const { currentPassword, newPassword } = req.body;
114
115     // Kullanıcıyı bul
116     const user = await User.findById(req.user.userId);
117
118     if (!user) {
119       return res.status(404).json({ message: 'Kullanıcı bulunamadı' });
120     }
121
122     // Mevcut şifreyi kontrol et
123     const isMatch = await bcrypt.compare(currentPassword, user.password);
124
125     if (!isMatch) {
126       return res.status(400).json({ message: 'Mevcut şifre yanlış' });
127     }
128
129     // Yeni şifreyi hashle
130     const salt = await bcrypt.genSalt(10);
131     const hashedPassword = await bcrypt.hash(newPassword, salt);
132
133     // Şifreyi güncelle
134     user.password = hashedPassword;
135     await user.save();
136
137     res.json({ message: 'Şifre başarıyla değiştirildi' });
138   } catch (err) {
139     console.error(err);
140     res.status(500).json({ message: 'Sunucu hatası' });
141   }
142 });
143
```

Şekil 19 - Profil Güncelleme Kodu

Bu kod, kullanıcı şifresini değiştirmek için kullanılır, mevcut şifreyi doğruladıktan sonra yeni şifreyi şifreler ve veritabanına kaydeder. Hata durumlarında uygun mesajlarla kullanıcıya geri dönüş yapılır ve şifre değiştirme başarılı olduğunda bir onay mesajı gönderilir.

### 6.1.3 İlan Oluşturma

The screenshot shows a web application interface for creating a new ad. The title is 'Yeni İlan Oluştur'. The form contains several input fields: 'Hayvan Cinsi', 'Yaş', 'Cinsiyet' (with a dropdown menu currently showing 'Erkek'), 'Sağlık Durumu', 'Karakter Özellikleri', 'Bulunduğu Yer', 'İletişim Numarası', and 'Hikaye'. The application name 'Yuvamı Bul' is visible in the top left, and navigation links for 'İlanlar', 'Blog', 'Donate', and 'Harita' are in the top right.

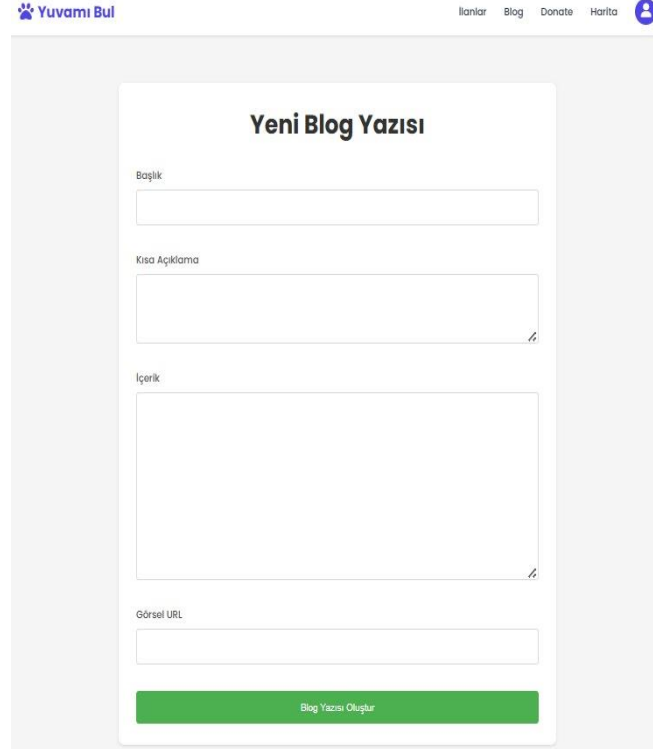
Şekil 20 – İlan Oluşturma Arayüzü

```
281 // İlan oluşturma
282 app.post('/api/ads', authenticateToken, checkAdPermissions, async (req, res) => {
283   try {
284     console.log('İlan oluşturma isteği:', req.body);
285     console.log('Kullanıcı bilgileri:', req.user);
286
287     // Kullanıcı bilgilerini middleware'den al
288     const userData = req.user;
289
290     if (!userData) {
291       return res.status(404).json({
292         message: 'Kullanıcı bulunamadı',
293         error: 'Kullanıcı bilgileri alınamadı'
294       });
295     }
296
297     const ad = new Ad({
298       ...req.body,
299       userId: userData._id,
300       olusturan: userData.email,
301       olusturanAd: `${userData.firstName} ${userData.lastName}`
302     });
303
304     console.log('Oluşturulan ilan:', ad);
305
306     const savedAd = await ad.save();
307     console.log('Kaydedilen ilan:', savedAd);
308
309     res.status(201).json(savedAd);
310   } catch (error) {
311     console.error('İlan oluşturma hatası:', error);
312     res.status(400).json({
313       message: 'İlan oluşturulurken bir hata oluştu',
314       error: error.message
315     });
316   }
317 });
```

### Şekil 21 - İlan Oluşturma Kodu

Bir kullanıcının sisteme ilan eklemesini sağlayan bir API uç noktasını göstermektedir. Kullanıcının kimlik doğrulaması yapıldıktan sonra gelen ilan verisi, veritabanına kaydedilerek başarılı şekilde oluşturulan ilan JSON formatında geri döndürülmektedir.

#### 6.1.4 Blog Oluşturma



The screenshot shows the 'Yeni Blog Yazısı' (New Blog Post) form in the Yuvamı Bul application. The form is titled 'Yeni Blog Yazısı' and contains four input fields: 'Başlık' (Title), 'Kısa Açıklama' (Short Description), 'İçerik' (Content), and 'Görsel URL' (Image URL). Each field has a corresponding label and a text input area. The 'İçerik' field is a larger text area. At the bottom of the form is a green button labeled 'Blog Yazısı Oluştur' (Create Blog Post). The form is set against a light gray background with a white border. The top of the page shows the 'Yuvamı Bul' logo and navigation links: 'İlanlar', 'Blog', 'Donate', 'Harita', and a user profile icon.

Şekil 22 – Blog Oluşturma Arayüzü

```


439 // Blog routes
440 app.post('/api/blogs', authenticateToken, checkBlogPermissions, async (req, res) => {
441   try {
442     const blog = new Blog(req.body);
443     await blog.save();
444     res.status(201).json(blog);
445   } catch (error) {
446     res.status(400).json({ message: error.message });
447   }
448 });
449
450 app.get('/api/blogs', async (req, res) => {
451   try {
452     const blogs = await Blog.find().sort({ createdAt: -1 });
453     res.json(blogs);
454   } catch (error) {
455     console.error('Blog listeleme hatası:', error);
456     res.status(500).json({ message: 'Bloglar listelenirken bir hata oluştu' });
457   }
458 });
459
460 app.get('/api/blogs/:blogNo', async (req, res) => {
461   try {
462     const blog = await Blog.findOne({ blogNo: req.params.blogNo });
463     if (!blog) {
464       return res.status(404).json({ message: 'Blog bulunamadı' });
465     }
466     res.json(blog);
467   } catch (error) {
468     console.error('Blog detay hatası:', error);
469     res.status(500).json({ message: 'Blog detay alınırken bir hata oluştu' });
470   }
471 });
472

```


Şekil 23 – Blog Oluşturma kodu

Bu kod, blog verilerini eklemek için kullanılır .Hata durumlarında, işlemle ilgili uygun hata mesajları döndürülerek kullanıcıya bilgi verilir


### 6.1.5 Kullanıcı Yetkilendirme











İlanlar Blog Donate Harita



## Kullanıcı Yetkilendirme



Ad Soyad	E-posta	Mevcut Rol	İşlem
admin .	admin@gmail.com	 Admin	 Yetkilendir
koray ışık Altun	koray@gmail.com	 Admin	 Yetkilendir
Avcılar Belediyesi Hayvan Sağlık Merkezi	avcilar01@gmail.com	 Barınak Sahibi	 - Yetkiyi Düşür
Çeşme Belediyesi Hayvan Barınağı	cesme@gmail.com	 Barınak Sahibi	 - Yetkiyi Düşür

Şekil 24 – Kullanıcı Yetkilendirme Arayüzü

```

184 // Kullanıcıyı yetkilendir
185 async function authorizeUser(userId) {
186   try {
187     const response = await fetch(`/api/users/${userId}/authorize`, {
188       method: 'PUT',
189       headers: {
190         'Authorization': `Bearer ${localStorage.getItem('token')}`,
191         'Content-Type': 'application/json'
192       },
193       body: JSON.stringify({ role: 'shelter' })
194     });
195
196     if (response.ok) {
197       notifications.success('Kullanıcı başarıyla yetkilendirildi!', 'İşlem Başarılı');
198       loadUsers(); // Tabloyu yenile
199     } else {
200       const error = await response.json();
201       notifications.error(error.message || 'Yetkilendirme işlemi başarısız!', 'İşlem Başarısız');
202     }
203   } catch (error) {
204     console.error('Yetkilendirme hatası:', error);
205     notifications.error('Yetkilendirme işlemi sırasında bir hata oluştu!', 'Sistem Hatası');
206   }
207 }

```

Şekil 25 – Kullanıcı Yetkilendirme Kodu

Bu kod, belirtilen kullanıcıyı yetkilendirmek için kullanılır ve işlem başarılı olursa kullanıcıyı yetkilendirilir . Eğer işlemde bir hata oluşursa, hata mesajı gösterilir ve kullanıcıya bir sistem hatası bildirilir



## BÖLÜM 7: KULLANICI ARAYÜZÜ

---

Kullanıcı deneyimini ön planda tutarak geliştirilen arayüzler, sistemin kolay ve etkili bir şekilde kullanılmasını sağlamaktadır. Bu bölümde kullanıcıların sistemle etkileşimde bulunduğu ana sayfa, profil, ilanlar, bağış yap, blog, harita ve profilim gibi sayfalar görsellerle birlikte tanıtılmış, her bir sayfanın işlevi açıklanmıştır.

### 7.1. Giriş Yap Arayüzü

Kullanıcıların sistemdeki hesaplarına güvenli bir şekilde erişim sağlamalarını mümkün kılan bir ekran olarak tasarlanmıştır. Kullanıcılar, sisteme giriş yapmak için geçerli kimlik bilgilerini (e-posta ve şifre) girerler. Bu ekran, kullanıcı deneyimini iyileştirmek amacıyla basit ve kullanıcı dostu bir arayüz ile geliştirilmiştir. Giriş yapma işlemi, hızlı ve güvenli bir şekilde gerçekleştirilmesi hedeflenerek tasarlanmıştır.

#### 7.1.1 Arayüzde Bulunan Öğeler:

- **E-posta Alanı:** Kullanıcıların, kayıt sırasında belirttikleri e-posta adreslerini girmeleri için sağlanan metin kutusudur. Bu alan, e-posta formatına uygun veri kabul eder.
- **Şifre Alanı:** Kullanıcıların, hesap güvenliğini sağlamak amacıyla şifrelerini gizli bir şekilde girmelerine olanak tanıyan metin kutusudur. Karakterler gizlenir ve yalnızca şifre uzunluğu görüntülenir.
- **Giriş Yap Butonu:** Kullanıcıların giriş işlemini başlatmalarını sağlayan, formu gönderme işlevi gören butondur. Bu buton tıklandığında, sunucuya kullanıcı bilgileri gönderilir ve doğrulama işlemi yapılır.
- **Kayıt Ol Linki:** Sistemde hesabı bulunmayan kullanıcıların yeni bir hesap oluşturabilmeleri için yönlendirme sağlayan bir bağlantıdır. Bu linke tıklayarak kullanıcılar, kayıt ekranına geçiş yapabilirler.

#### 7.1.2 Kullanıcı Deneyimi Açısından Katkıları:

**Basit ve Hızlı İşlem:** Kullanıcı dostu bir tasarım ile, giriş işlemi kullanıcılar için hızlı ve verimli bir şekilde tamamlanabilir.

**Hata Geri Bildirimi:** Hatalı girişlerde, kullanıcıya doğru bir şekilde geri bildirim sağlanarak, hatalı bilgiler (örneğin "E-posta veya şifre hatalı") hakkında bilgilendirme yapılır.

**Başarılı Giriş:** Giriş işlemi başarıyla tamamlandığında, kullanıcıya hoş geldin mesajı gösterilir ve sistemdeki diğer sayfalara yönlendirilir.

### 7.1.3 Teknik Özellikler:

**Kimlik Doğrulama:** Kullanıcı giriş bilgileri, JavaScript aracılığıyla sunucuya gönderilir ve burada doğrulama işlemi yapılır. Bu işlem için /api/login endpoint'i kullanılmaktadır.

**Token ve Kullanıcı Bilgilerinin Saklanması:** Başarılı bir girişin ardından, kullanıcıya ait JWT token ve kullanıcı bilgileri (ad, soyad, e-posta, rol vb.) localStorage'a kaydedilir. Bu sayede kullanıcının kimliği doğrulandıktan sonra, diğer işlemler için bu bilgiler kullanılabilir.

### 7.1.4 Ekran Görüntüsü:

Şekil 26 - Projedeki Giriş Yapma Ekranı

## 7.2. Kayıt Ol Arayüzü

Kayıt ol arayüzü, platforma yeni kullanıcıların katılımını sağlamak amacıyla geliştirilmiş bir kullanıcı arayüzüdür. Bu ekran aracılığıyla kullanıcılar, sisteme giriş yapabilmek ve sunulan hizmetlerden faydalanabilmek için temel bilgilerini sisteme tanımlarlar. Arayüz, sade ve erişilebilir bir tasarım diliyle oluşturulmuş olup kullanıcı deneyimini ön planda tutacak şekilde yapılandırılmıştır. Bu ekran, kullanıcıların güvenli, hızlı ve sorunsuz bir şekilde kayıt işlemlerini gerçekleştirebilmesini hedefler.

### 7.2.1 Arayüzde Bulunan Öğeler:

- **Ad Alanı:** Kullanıcının adını veya barınak ismini girdiği metin kutusu.
- **Soyad Alanı:** Kullanıcının soyadını veya barınak ismini girdiği metin kutusu.
- **Posta Alanı:** Kullanıcının veya barınak yetkilisinin e-posta adresi girdiği metin kutusu.
- **Şifre Alanı:** Kullanıcının güvenli bir şifre belirlediği metin kutusu.
- **Doğum Tarihi Alanı:** Kullanıcının doğum tarihini veya barınak yetkilisinin barınak kuruluş tarihini seçtiği tarih alanı.
- **Kayıt Ol Butonu:** Kullanıcının kayıt işlemini tamamlamak için tıkladığı buton.
- **Giriş Yap Linki:** Zaten hesabı olan kullanıcıların giriş sayfasına yönlendirilmesini sağlayan bağlantı.

### 7.2.2. Kullanıcı Deneyimine Katkıları

Kapsayıcı ve Anlaşılır Form Alanları sayesinde kullanıcılar, hangi bilgiyi nereye yazmaları gerektiğini kolaylıkla kavrayabilmektedir.

Doğrudan Geribildirimler ile hatalı ya da eksik girişlerde kullanıcılar hızlı şekilde bilgilendirilmekte ve işlem yeniden denenebilmektedir.

Başarılı Kayıt Yönlendirmesi, kullanıcıyı işlem sonunda otomatik olarak giriş sayfasına aktarmakta ve zaman kaybının önüne geçmektedir.

Bildirim Mekanizması, kullanıcıya kayıt işleminin sonucu hakkında (başarı veya hata durumunda) açık ve anlaşılır mesajlarla bilgi vermektedir.

### 7.2.3. Teknik Uygulama Detayları

Bu arayüzdeki form, JavaScript kullanılarak dinamik şekilde yönetilmektedir. Form gönderimi, /api/register uç noktasına HTTP POST isteği aracılığıyla gerçekleştirilir. Gelen yanıt doğrultusunda kullanıcıya uygun mesaj gösterilir:

Kayıt başarılıysa: Kullanıcı bilgilendirilir ve giriş ekranına yönlendirilir.

Kayıt başarısızsa: Hata mesajı gösterilerek kullanıcı gerekli düzeltmeleri yapmaya yönlendirilir.

## 7.2.4. Ekran Görüntüsü

**Kayıt Ol**

Ad

Soyad

E-posta

Şifre

gg.aa.yyyy

Kayıt Ol

Zaten hesabınız var mı? [Giriş Yap](#)

Şekil 27 – Projedeki Kayıt Olma Ekranı

## 7.3. İlanlar Arayüzü

İlanlar arayüzü, sistemde mevcut olan tüm hayvan ilanlarının sunulduğu temel ekranlardan biridir. Bu sayfa barınak yetkilileri tarafından oluşturulan ilanların listelenmesini sağlar. Her ilan, kullanıcıya hayvanın türü, yaşı, bulunduğu konum gibi temel bilgileri gösteren bir kart yapısında görüntülenir.

### 7.3.1 Arayüzde Yer Alan Bileşenler:

- **Başlık:** Sayfanın üst kısmında "Hayvan İlanları" başlığı yer almakta ve sayfanın genel amacını kullanıcıya açıkça belirtmektedir.

#### Arama ve Filtreleme Alanı:

- **İlan Arama:** Kullanıcının, hayvanın cinsi gibi temel niteliklere göre metin bazlı arama yapmasını sağlayan giriş kutusu.
- **Şehir Filtresi:** İstanbul, Ankara ve İzmir gibi şehirler için seçim yapılmasına olanak tanıyan açılır menü.
- **Ara Butonu:** Belirtilen kriterlere uygun ilanların listelenmesini sağlayan buton.

### 7.3.2 İlan Kartları:

Her bir ilan kartı, aşağıdaki öğeleri içerecek şekilde kullanıcıya sunulur:

- **Resim:** Hayvana ait bir görsel, kartın üst kısmında yer alır.
- **Cins Bilgisi:** İlan edilen hayvanın türü/cinsi (örneğin: "Labrador", "Tekir Kedi").
- **Konum:** Hayvanın bulunduğu şehir veya ilçe bilgisi.
- **Yaş:** Hayvanın yaşı, kullanıcıya bilgi vermesi açısından belirtilir.
- **Cinsiyet:** Hayvanın erkek ya da dişi olduğu bilgisi.
- **Oluşturan :** İlanı oluşturan barınağın adı.
- **İlan Tarihi:** İlanın sisteme ne zaman eklendiği bilgisi.
- **Detay Butonu:** Kullanıcının, seçilen ilana ait daha detaylı bilgi almasını sağlayan yönlendirme bağlantısı.

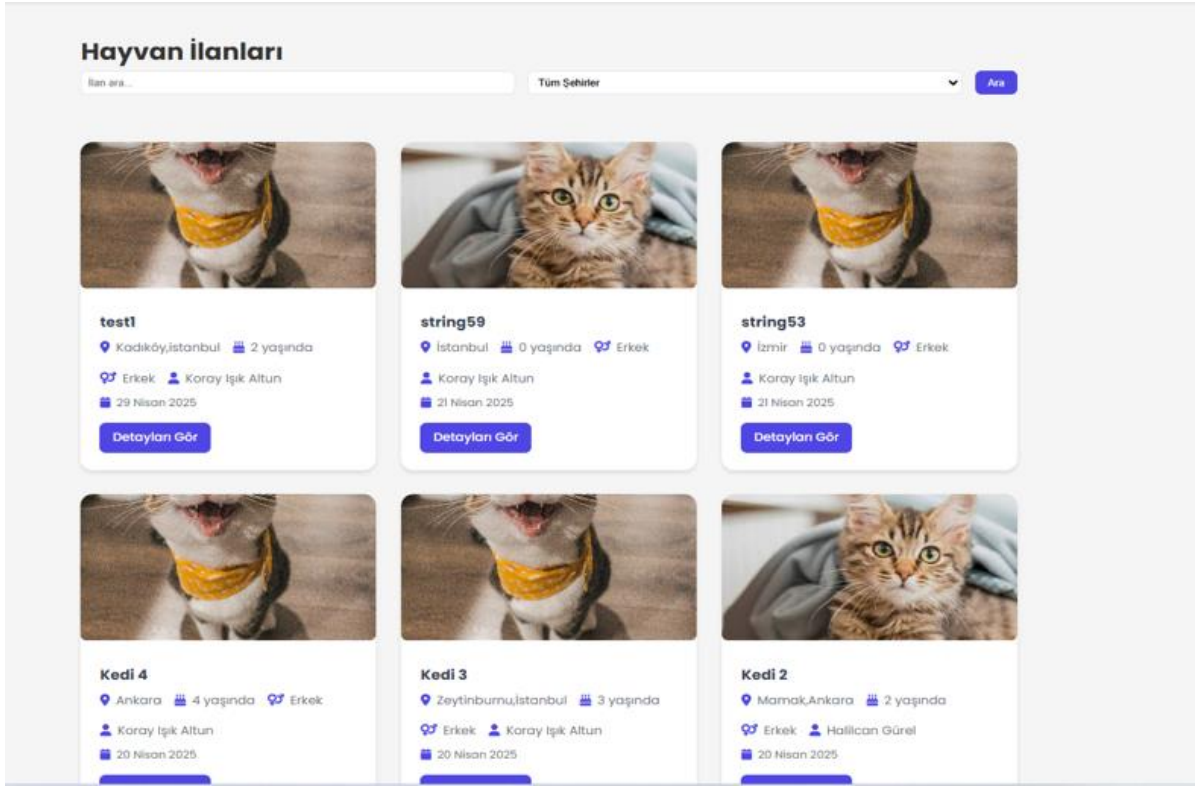
### 7.3.3 Kullanıcı Deneyimi Açısından Katkılar:

Tüm ilanlar, sayfa her yüklendiğinde sistemden dinamik olarak çekilmekte ve güncel bilgilerle kullanıcıya sunulmaktadır.

Arama ve filtreleme özelliği sayesinde kullanıcılar ihtiyaç duydukları hayvanlara daha kısa sürede erişebilmektedir.

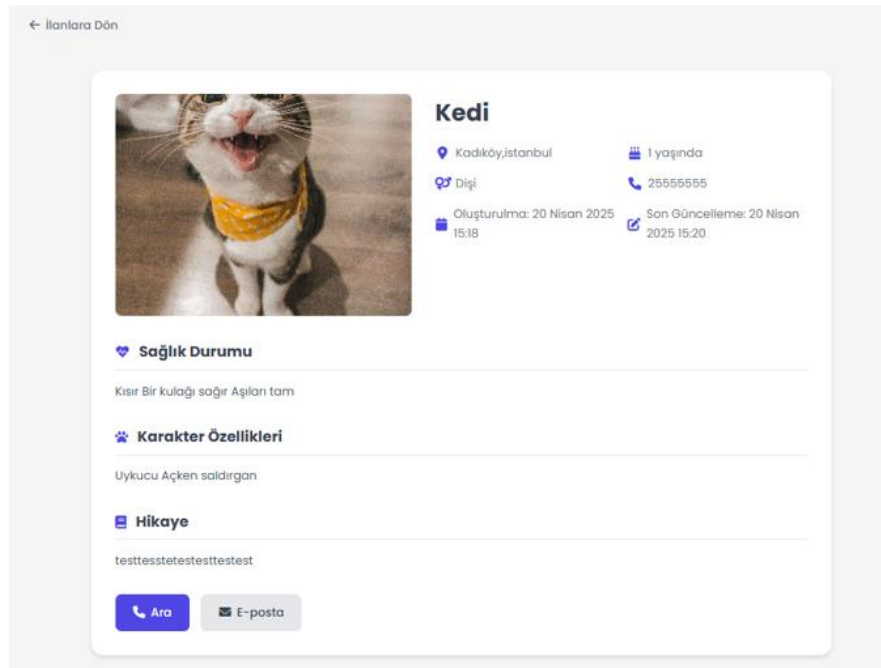
Temel düzeyde mobil uyumluluğu destekleyecek şekilde yapılandırılmıştır. Ancak kullanıcı deneyimini daha da artırmak adına, gelecekte farklı ekran boyutlarına tam uyum sağlayacak şekilde kapsamlı bir mobil optimizasyon süreci planlanabilir.

### 7.3.4 Ekran Görüntüsü 1 :



Şekil 28 – Projedeki İlanlar Sayfası

### 7.3.5 Ekran Görüntüsü 2 :



Şekil 29 – Projedeki Detaylı İlan Görüntüleme Sayfası

## 7.4 Blog Sayfası Arayüzü

Blog sayfası arayüzü, kullanıcıların blog içeriklerine kolayca erişebilmesini ve yöneticilerin içerik ekleyip yönetebilmesini sağlayacak şekilde tasarlanmıştır. Modern bir görsel estetikle oluşturulmuş bu arayüz, kullanıcıların platforma entegre olmuş blog yazılarını kolayca gezebileceği ve etkileşimde bulunabileceği bir deneyim sunar. Ayrıca, admin kullanıcıları için içerik yönetimi seçenekleri de içerir, böylece yazılar üzerinde değişiklik yapabilirler. Tasarım, her cihazda uyumlu çalışacak şekilde responsive (duyarlı) olarak geliştirilmiştir.

### 7.4.1. Arayüz Öğeleri

- **Başlık ve Yönetici Kontrol Paneli:** Sayfanın üst kısmında, platformun blog içeriklerine odaklanıldığı bir başlık yer alır. Yönetici kullanıcıları için, "Yeni Blog Oluştur" butonu sağ üst köşede bulunan kontrol panelinde mevcuttur.
- **Blog Kartları:** Blog yazılarına dair görsellerin ve başlıkların yer aldığı kartlardan oluşan ızgara yapısına sahiptir. Her bir kart, başlık, kısa açıklama ve oluşturulma tarihini içerir. Her bir kartta ayrıca "Devamını Oku" butonu bulunur.
- **Blog Detayına Erişim:** Kullanıcılar, blog kartlarındaki "Devamını Oku" butonuna tıklayarak yazının detaylarına ulaşabilirler. Detaylar bir modal pencere içinde gösterilir ve yazının tamamı, görseli, oluşturulma ve güncellenme tarihleri gibi bilgiler yer alır.
- **Yönetici Butonları:** Admin kullanıcıları, blog kartlarının üzerinde yer alan "Düzenle" ve "Sil" butonları aracılığıyla içerik yönetimini yapabilirler. (Görüntü2.2)

### 7.4.2. Kullanıcı Deneyimine Katkıları

Erişilebilir ve Anlaşılır Yapı sayesinde kullanıcılar, blog içeriklerini hızlı bir şekilde bulabilir ve ilgili yazıya kolayca geçiş yapabilirler.

Admin Yetkileri ve İçerik Yönetimi ile yalnızca admin kullanıcılarının içerik üzerinde düzenleme yapabilmesi sağlanır, böylece içerik yönetimi güvenli hale gelir.

Modal Pencere ile Detaylı Görünüm kullanıcılara, içeriklerin tamamını ve ilgili bilgileri kolayca gösterir. Ayrıca bu pencere kapanabilir, böylece kullanıcılar dilediklerinde yazıya geri dönebilirler.

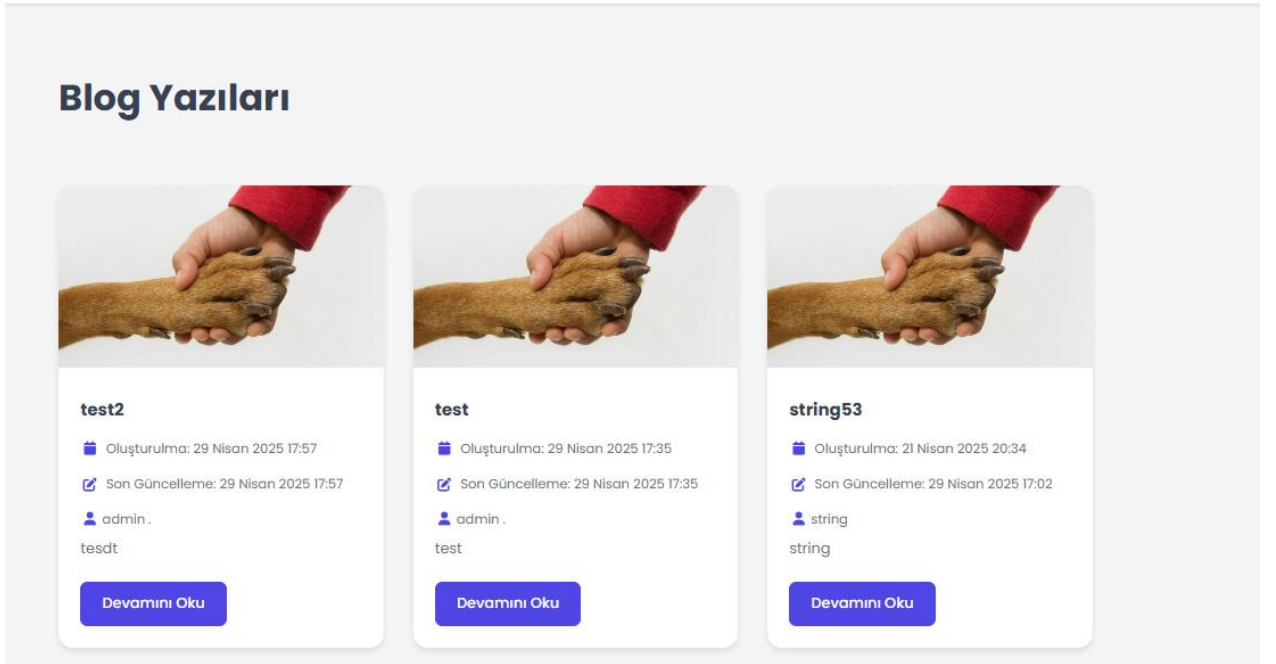
### 7.4.3 Teknik Uygulama Detayları

Blog sayfası, JavaScript kullanılarak dinamik bir yapıda oluşturulmuştur. Sayfa, API üzerinden veri alır ve yazılar blog kartları olarak kullanıcıya gösterilir. Admin kullanıcılarının içerik yönetim işlemleri ise aşağıdaki şekilde gerçekleştirilir:

- Düzenleme ve Silme: Admin kullanıcıları, blog yazılarının üzerine tıklayarak içeriklerini düzenleyebilir ya da silebilirler. Bu işlemler API aracılığıyla gerçekleştirilir ve sayfa, güncel verilere göre yeniden render edilir.
- Modal Pencere İle Blog Detayı: Kullanıcılar, blog kartına tıkladıklarında, API üzerinden çekilen yazı içeriği, görseli ve tarih bilgileri bir modal pencerede gösterilir.

Sayfa yüklenmeden önce, oturum kontrolü yapılır. Eğer kullanıcı daha önce giriş yapmışsa, admin kontrol paneli ve düzenleme butonları yalnızca admin yetkilerine sahip kullanıcılara gösterilir.

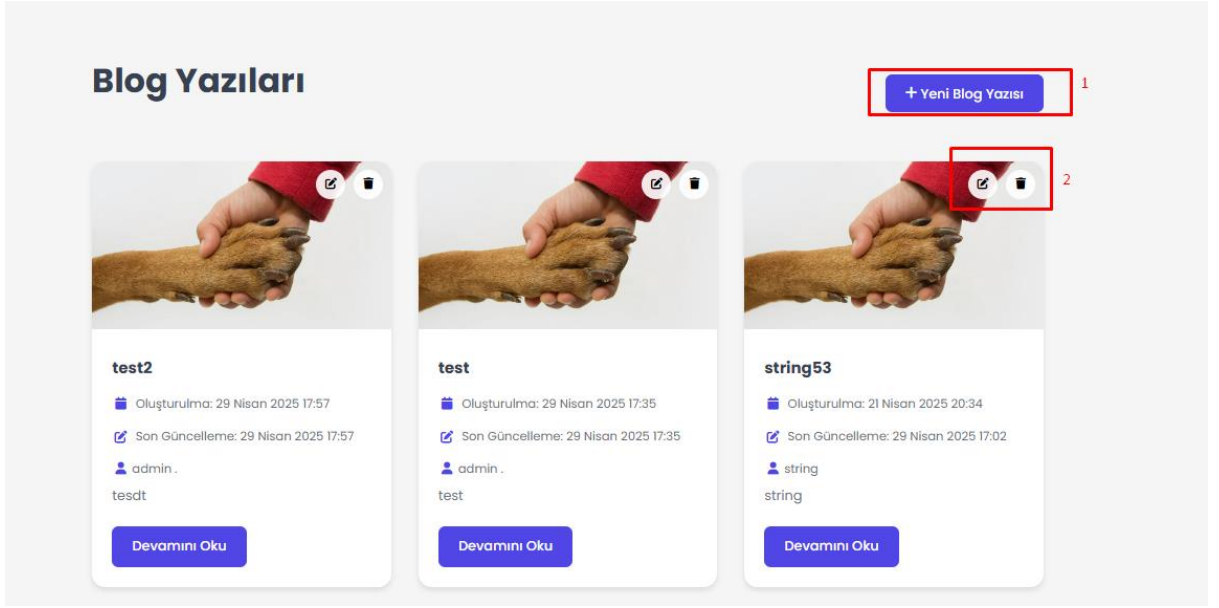
### 7.4.4. Ekran Görüntüsü 1: (Kullanıcı ve Barınak Görünümü)



Şekil 30 – Projedeki Blog Yazıları Sayfası



#### 7.4.5. Ekran Görüntüsü 2: (Admin Görünümü)



Şekil 31 – Projedeki Blog Yazıları Ekleme ,Silme ,düzenleme Sayfası

#### 7.5 Bağış Yap Sayfası Arayüzü

Bağış Yap sayfası, kullanıcıların güvenilir kuruluşlar aracılığıyla hayvanlar için maddi destek sağlamasına imkan tanır. Sayfa, kullanıcıya bilgilendirici bir girişle başlayarak, onları bilinçli bir şekilde bağış yapmaya yönlendirir.

##### 7.5.1. Arayüz Öğeleri

Bağış sayfası aşağıdaki temel bileşenlerden oluşur:

- **Sayfa Başlığı ve Alt Başlık:** Sayfanın üst kısmında yer alan başlık ("Hayvanların Yanında Ol!") kullanıcıyı duyarlı bir bağış yapma davranışına yönlendirmeyi amaçlar. Alt başlık ise kullanıcıyı motive edici kısa bir açıklama sunar.
- **Bağış Bilgilendirme Bölümü:** Kullanıcıya neden bağış yapması gerektiğini açık bir şekilde anlatır.
- **Bağış Kartları (donation-grid):** Üç farklı sivil toplum kuruluşuna (STK) ait bağış bağlantılarını içeren görsel ve metin destekli kartlar kullanıcıya sunulur. Her kart bir logo, başlık, açıklama ve "Bağış Yap" butonundan oluşur . butonlara tıklandığında ilgili kuruluşun resmi bağış sayfası yeni sekmede açılır.

## 7.5.2. Ekran Görüntüsü:



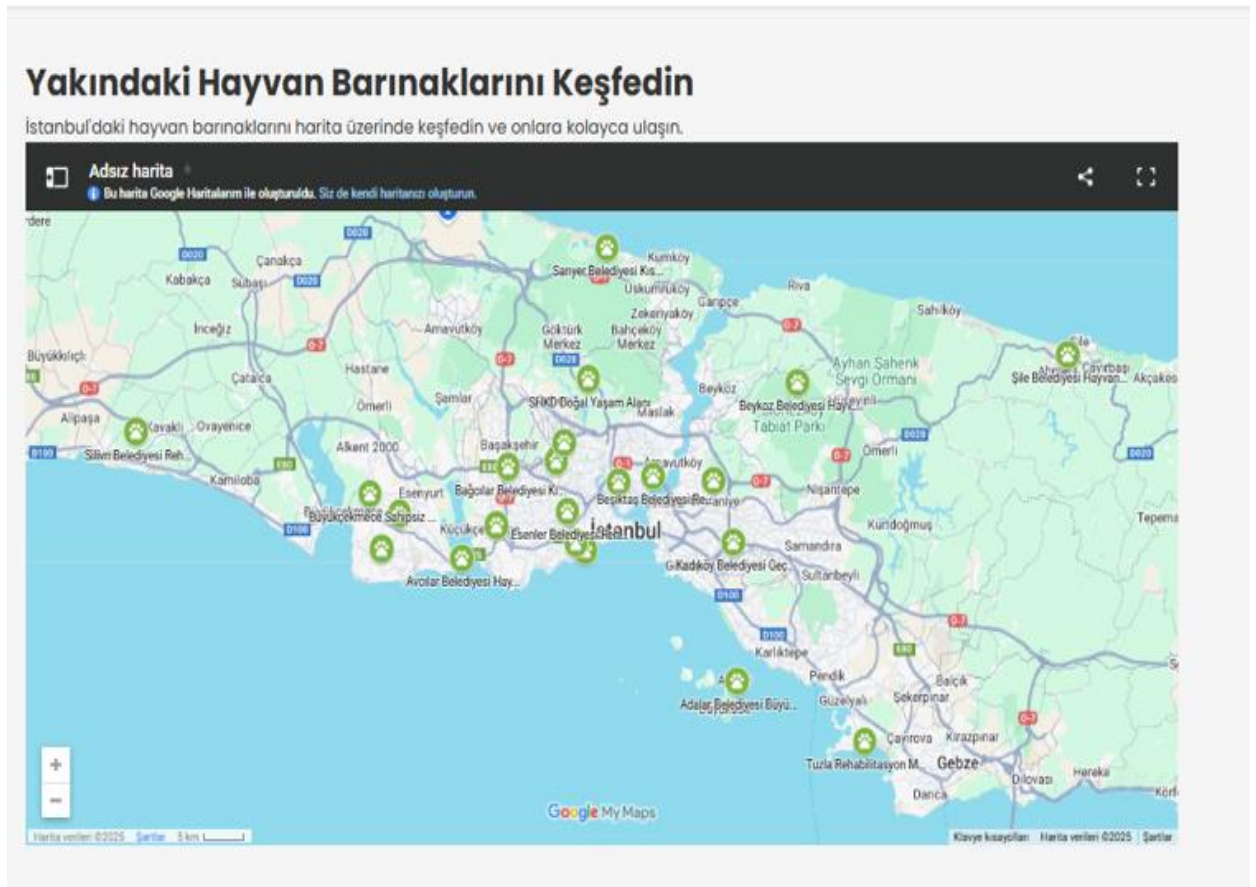
Şekil 32 – Projedeki Bağış Yapma Sayfası

## 7.6. Haritalar Arayüzü

Haritalar sayfası, kullanıcıların bulundukları konuma yakın hayvan barınaklarını kolayca keşfetmelerini sağlamak amacıyla tasarlanmıştır. Sayfa, özellikle İstanbul'daki barınakların konumlarını görsel olarak kullanarak kullanıcıların bu yerlere erişimini kolaylaştırmakta ve fiziksel temas kurma sürecini hızlandırmaktadır.

Harita içerisinde İstanbul'daki çeşitli hayvan barınaklarının konumları kullanıcıya görsel olarak sunulmakta; bu yaklaşım, yalnızca metin tabanlı bilgilendirme yerine mekânsal farkındalık kazandırmayı hedeflemektedir. Ayrıca Google Maps altyapısı kullanılarak harita etkileşimli ve güncel bir yapıda sunulmuş, böylece kullanıcılar harita üzerinde gezinerek detaylı bilgi alabilme imkanına sahip olmuştur.

### 7.6.1 Ekran Görüntüsü:



Şekil 33 – Projedeki Hayvan Barınakları Haritası

## BÖLÜM 8: Sonuç ve Değerlendirme

---

Bu proje, sahipsiz hayvanların sahiplenilmesini kolaylaştırmak, barınakların dijital ortamda görünürlüğünü artırmak ve bireyler ile hayvan barınakları arasında etkili bir iletişim kurmak amacıyla geliştirilmiştir. Sistem; bireysel kullanıcılar, barınak yöneticileri ve yöneticiler olmak üzere farklı kullanıcı rolleriyle çalışmakta ve ilan yayınlama, barınakları harita üzerinde görüntüleme, blog içerikleri oluşturma gibi temel işlevleri desteklemektedir.

Projenin geliştirilmesinde frontend tarafında Html, Css, JavaScript, backend tarafında ise Node.js, Express.js teknolojileri kullanılmıştır. Veritabanı olarak MongoDB, kullanıcı doğrulaması için JWT, API dokümantasyonu için ise Swagger tercih edilmiştir.

Veritabanı, kullanıcı, ilan ve blog gibi temel bileşenleri içerecek şekilde tasarlanmış; sistem mimarisi ise üç katmanlı yapıya göre oluşturulmuştur (Kullanıcı Arayüzü, Sunucu, Veritabanı). Veri akışı, kullanıcı etkileşimlerinin hızlı ve güvenli şekilde gerçekleşmesini sağlayacak biçimde modellenmiştir.

Sonuç olarak geliştirilen bu sistem, hem teknik gereksinimleri başarıyla karşılamış hem de sahipsiz hayvanlar için farkındalık yaratma ve çözüm üretme konusunda önemli bir Web platform olarak işlev görmüştür.

### 8.1 Projenin Başarı Durumu

Bu proje, planlanan hedeflere büyük ölçüde ulaşarak başarılı bir şekilde tamamlanmıştır. Kullanıcı dostu arayüz tasarımı, güçlü veri yönetimi yapısı ve kullanıcı rollerine göre özelleştirilmiş işlevler sayesinde hem bireysel kullanıcıların hem de barınak yöneticilerinin ihtiyaçlarına etkin çözümler sunulmuştur.

Ayrıca benzer uygulamalarla yapılan karşılaştırmalarda sistemin kullanıcı deneyimi, işlevsellik ve teknik altyapı açısından rekabet edebilir düzeyde olduğu gözlemlenmiştir. Bu yönleriyle proje, sahipsiz hayvanların sahiplenilmesini kolaylaştırmak ve bu alandaki sosyal sorumluluğu desteklemek adına etkili bir dijital çözüm sunmaktadır. Kullanıcı deneyimi, teknik altyapı ve işlevsellik bakımından beklentileri karşılayan sistem, sosyal fayda sağlama yönünden de etkili bir web platform olarak değerlendirilmiştir.

### 8.2 Gelecek Çalışmalar

Projenin ilerleyen aşamalarında, sistemin erişilebilirliğini ve kullanıcı deneyimini artırmak amacıyla bir mobil uygulama geliştirilmesi planlanmaktadır. Mobil uygulama sayesinde kullanıcılar; ilanlara, blog

içeriklerine ve barınak bilgilerine daha hızlı ve kolay şekilde ulaşabilecek, bildirimlerle güncel gelişmelerden haberdar olabileceklerdir. Ayrıca mobil platform, sahiplendirme sürecini daha interaktif ve kullanıcı dostu bir hale getirecektir.

#### **a. Sürdürülebilir Kalkınma Bilinci**

**Bu projede**, sahihsiz hayvanların barınaklardan sahiplenilmesini kolaylaştırmak amacıyla geliştirilen web tabanlı bir hayvan sahiplendirme platformu sunulmuştur. Platform; kullanıcı kayıt ve girişi, sahihsiz hayvanlar için ilan oluşturma ve görüntüleme, blog içerikleriyle bilgilendirme, harita üzerinden barınak konumlarını görselleştirme ve bağış yapma gibi işlevleri içermektedir.

Sistem, genel kullanıcılar, barınak yöneticileri ve yönetici kullanıcılar için farklı roller tanımlayarak kullanıcı ihtiyaçlarına uygun arayüzler sunmaktadır. Genel kullanıcılar, ilanları inceleyip görüntüleyebilirken; barınak yöneticileri barındaki hayvanlara ait ilanları ekleyebilmekte ve düzenleyebilmektedir. Yönetici kullanıcı ise tüm sistem bileşenlerini yönetme yetkisine sahiptir.

**Proje, dijitalleşmenin sürdürülebilir kalkınmaya olan katkısını esas alarak** tasarlanmıştır. Yazılım geliştirme sürecinde enerji verimliliği gözetilmiş, sistem gereksinimleri minimum kaynak kullanımı hedefiyle yapılandırılmıştır. Veri işleme ve saklama süreçlerinde israfı azaltacak yöntemler tercih edilmiştir. Kullanıcı dostu arayüzü sayesinde geniş bir kullanıcı kitlesine hitap eden bu platform, teknoloji yoluyla toplumsal farkındalık oluşturarak hem sahihsiz hayvanların sahiplenme süreçlerini dijital ortamda kolaylaştırmakta hem de sürdürülebilirlik bilincine katkı sağlamaktadır.

#### **b. Proje, Risk ve Değişiklik Yönetimi Becerileri**

Proje süreci boyunca kapsam ve zaman yönetimi ilkelerine bağlı kalınmış; görev dağılımı net olarak yapılmış ve proje kilometre taşları belirlenerek takip edilmiştir. Yazılım geliştirme sürecinde teknik ve organizasyonel riskler öngörülerek risk analizleri gerçekleştirilmiş, bu risklere karşı önleyici stratejiler geliştirilmiştir.

Ayrıca proje ilerledikçe kullanıcı geri bildirimleri ve ihtiyaç analizleri doğrultusunda ortaya çıkan gereksinim değişiklikleri sistematik olarak değerlendirilmiş ve değişiklik yönetimi prensiplerine uygun biçimde uygulanmıştır. Böylece proje, yalnızca teknik başarı değil, aynı zamanda etkili proje yönetimi uygulamaları açısından da bir deneyim alanı oluşturmuş; öğrencinin gerçek iş yaşamında karşılaşılabileceği durumlara yönelik yönetim becerileri kazanmasına katkı sağlamıştır.

### c. Mühendislik Çözümlerinin Hukuksal ve Etik Sonuçları

Sistem geliştirilirken, **kişisel verilerin korunması kanunu (KVKK/GDPR)** ve **telif haklarına ilişkin yasal düzenlemeler** dikkate alınmıştır. Kullanıcı bilgilerinin güvenliği ve gizliliği ön planda tutulmuş, veri işleme süreçleri bu yasal çerçeveye uygun şekilde yapılandırılmıştır. Ayrıca projede kullanılan açık kaynak yazılım bileşenlerinin lisans uyumlulukları titizlikle incelenmiş ve uygun kullanım esaslarına bağlı kalınmıştır. Algoritmalar tasarlanırken **etik ilkelere**, özellikle tarafsızlık ve şeffaflık prensiplerine önem verilmiş; ilanların ve kullanıcı etkileşimlerinin adil şekilde işlenmesi sağlanmıştır. Mühendislik kararları, sadece teknik yeterlilik açısından değil, aynı zamanda **toplumsal sorumluluk ve etik farkındalık** çerçevesinde değerlendirilmiş ve uygulanmıştır.

### d. Deney Tasarımı ve Karmaşık Problemlerin Çözümü

Geliştirme sürecinde, sistemin doğruluğu, güvenilirliği ve performansı farklı kullanıcı senaryolarıyla test edilmiştir. Bu kapsamda, **deney tasarımı ilkeleri uygulanmış**, çeşitli test planları oluşturulmuş, olası hata durumları (örneğin ilan ekleme sırasında veri kaybı, erişim sorunları, rol bazlı erişim hataları vb.) simüle edilerek sistemin tepkisi gözlemlenmiştir. Test sonuçları, bilimsel analiz yöntemleriyle değerlendirilerek sistemde gerekli iyileştirmeler yapılmıştır.

Bu süreç, öğrencinin **disipline özgü mühendislik problemlerine sistematik çözüm geliştirme**, deney temelli doğrulama yapma ve kullanıcı ihtiyaçlarına göre optimize edilmiş çözümler üretme becerilerini güçlendirmiştir.

## BÖLÜM 9: KAYNAKÇA

---

- [1] <https://www.haytap.org>
- [2] <https://www.petfinder.com>
- [3] <https://www.adoptapet.com>
- [4] <https://www.postman.com/api-platform/api-design/>
- [5] [https://swagger.io/docs/specification/v2\\_0/what-is-swagger/](https://swagger.io/docs/specification/v2_0/what-is-swagger/)
- [6] <https://supertokens.com/blog/what-is-jwt>
- [7] W3C. (2014). HTML5 Specification. <https://www.w3.org/TR/html5/>
- [8] Berners-Lee, T. (1989). Information Management: A Proposal, CERN.
- [9] W3C. (1996). Cascading Style Sheets, Level 1. <https://www.w3.org/TR/CSS1/>
- [10] Frain, B. (2015). *Responsive Web Design with HTML5 and CSS3*. Packt Publishing.
- [11] Mozilla Developer Network (MDN). (n.d.). JavaScript Documentation. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [12] Erdinç, F. (2024). *Yeni Başlayanlar İçin HTML5, CSS3 ve JavaScript*. Abaküs Kitap.
- [13] Node.js Foundation. (2024). Node.js Documentation. <https://nodejs.org/en/>
- [14] Express.js Foundation. (2024). Express.js Documentation. <https://expressjs.com/>
- [15] Mongoose. (2024). Mongoose Documentation. <https://mongoosejs.com/>
- [16] JWT.io. (2024). JSON Web Token Introduction. <https://jwt.io/>
- [17] Express Validator. (2024). Express Validator Documentation. <https://express-validator.github.io/docs/>
- [18] MongoDB, Inc. (2024). MongoDB Manual. <https://docs.mongodb.com/>
- [19] Swagger.io. (2024). Swagger Documentation. <https://swagger.io/docs/>
- [20] <https://medium.com/@samuelnoye35/simplifying-api-development-in-node-js-with-swagger-a5021ac45742>
- [21] [https://swagger.io/docs/specification/v3\\_0/about/](https://swagger.io/docs/specification/v3_0/about/)
- [22] Sommerville, Ian. *Software Engineering*, 10th Edition, Pearson, 2015

[23] Microsoft Docs. "Three-Tier Architecture"



## ÖZGEÇMİŞLER

---

### Kişisel Bilgiler

**Adı Soyadı:** Halil Can Gürel

**Doğum Tarihi ve Yeri:** 16.09.2001 -Çanakkale

**E-posta:** [halilcangurell@gmail.com](mailto:halilcangurell@gmail.com)

### Öğrenim Durumu

Derece	Alan	Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Müh.	İstanbul Atlas Üniversitesi	Halen Okuyor

### İş Tecrübesi

Yıl	Firma/Kurum	Görevi
2021	DenizBank	Stajyer
2023	Şişli Belediyesi	Stajyer

## Kişisel Bilgiler

**Adı Soyadı:** Remzican Sokur

**Doğum Tarihi ve Yeri:** 20.09.2003 – İstanbul

**E-posta:** remzicansokur5@gmail.com

## Öğrenim Durumu

Derece	Alan	Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Müh	İstanbul Atlas Üniversitesi	Halen Okuyor

## İş Tecrübesi

Yıl	Firma/Kurum	Görevi
2024	Doğuş Yayın Grubu	Stajyer
2025	Veribox Teknoloji	Monitoring Stajyer

## Kişisel Bilgiler

**Adı Soyadı:** Koray Işık Altun

**Doğum Tarihi ve Yeri:** 24.09.2003 - Ordu

**E-posta:** korayaltun607@gmail.com

## Öğrenim Durumu

Derece	Alan	Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Müh	İstanbul Atlas Üniversitesi	Halen Okuyor

## İş Tecrübesi

Yıl	Firma/Kurum	Görevi
2023	Enka System	Stajyer
2024	Smart Info Data Solutions	Stajyer