Veterinerlik Otomasyonu Proje Raporu

Veterinerlik Otomasyonu Gereksinim Analizi

Aşağıda sistemin temel bileşenleri ve ihtiyaç duyulan özellikler detaylı olarak açıklanmıştır.

1. Müşteriler

Veterinerlik hizmetlerinden yararlanan bireyler veya kuruluşlarla ilgili bilgileri saklar.
Gereksinimler:

Müşteri ID (PK)
Müşteri Adı
Müşteri Soyadı
Müşteri Telefon Numarası
Müşteri Adresi

---

2. Veterinerler

Veteriner kliniğinde çalışan uzmanların bilgilerini içerir.
Gereksinimler:

Veteriner ID (PK)
Veteriner Adı
Veteriner Soyadı
Veteriner Telefon Numarası

---

3. Hayvanlar

Veterinerlik hizmeti alan hayvanlarla ilgili bilgileri tutar.
Gereksinimler:

Hayvan ID (PK)
Hayvan Türü
Hayvan Yaşı
Hayvan Cinsiyeti

---

4. Hastalıklar

Hayvanlarda teşhis edilen hastalıkların bilgisini saklar.
Gereksinimler:

Hastalık ID (PK)
Hastalık Adı

---

5. Tedaviler

Hastalıkların teşhis ve tedavi süreçlerini kapsayan verileri içerir.
Gereksinimler:

Tedavi ID (PK)
Tedavi Başlangıç Tarihi
Tedavi Bitiş Tarihi

---

6. Tedavi-İlaç İlişkisi

Tedavide kullanılan ilaçların miktarını ve ilişkisini yönetir.
Gereksinimler:

Tedavi ID (FK)
İlaç ID (FK)
İlaç Dozu

---

7. Tedavi-Hastalık İlişkisi

Tedavi ve hastalık arasındaki ilişkiyi yönetir.
Gereksinimler:

Tedavi ID (FK)
Hastalık ID (FK)

---

8. İlaçlar

Veterinerlik hizmetlerinde kullanılan ilaçların veritabanında tutulmasını sağlar.
Gereksinimler:

İlaç ID (PK)
İlaç Adı
İlaç Fiyatı

---

9. Tedavi Sonucu

Tedavi sonucunda hayvanın sağlık durumunu belirten kayıtları tutar.
Gereksinimler:

Sonuç ID (PK)
Sonuç Durumu
Hayvan Durumu (Enum: Ölü, İyileşti, Tedavi Ediliyor)

---

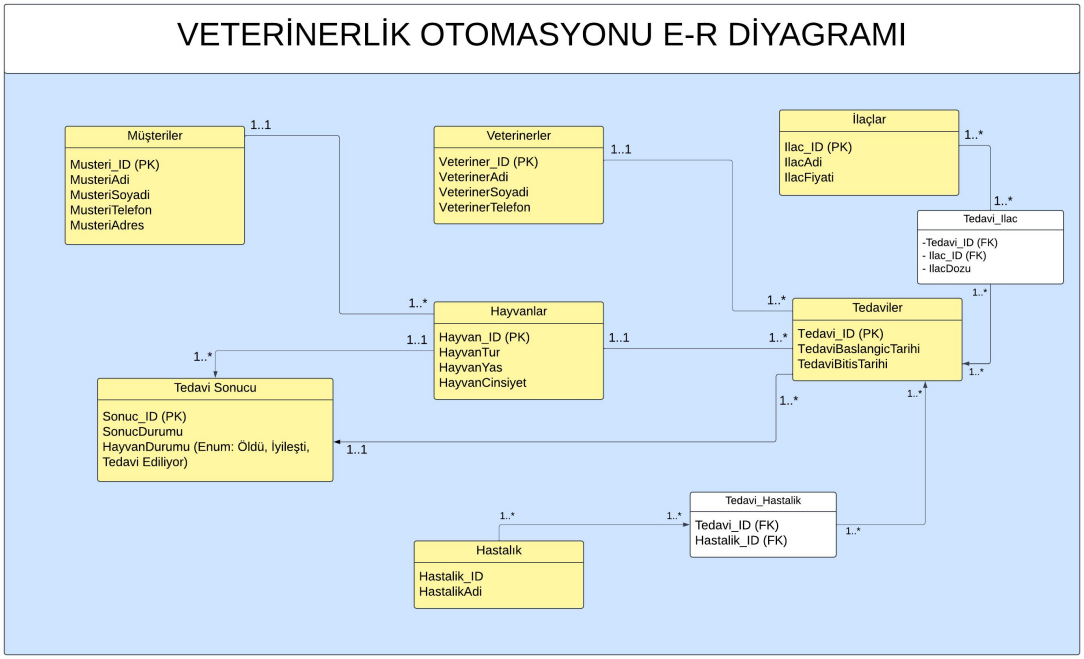
10. İlişkiler ve Kardinaliteler

İlişki Detayları:

Müşteriler ve Hayvanlar: 1 müşterinin birden fazla hayvanı olabilir, bir hayvanın yalnızca bir sahibi olabilir (1..*).
Veterinerler ve Tedaviler: 1 veteriner birden fazla tedavide yer alabilir, her hayvanın tedavisini üstlenen yalnız bir veteriner vardır (1..*).
Hayvanlar ve Tedaviler: 1 hayvana birden fazla tedavi uygulanabilir, her tedavi yalnız bir hayvana özgüdür (1..*).
Tedaviler ve İlaçlar: 1 tedavide birden fazla ilaç kullanılabilir, bir ilaç birden fazla tedavide kullanılabilir (*..*).
Tedaviler ve Hastalıklar: 1 tedavide birden fazla hastalık görülebilir, bir hastalık birden fazla tedavide görülebilir (*..*).
Tedavi Sonucu ve Hayvanlar: Her tedavi sonucu bir hayvana özgüdür, bir hayvanın birden fazla tedavi sonucu olabilir. (1..*).
Tedavi Sonucu ve Tedaviler: Her tedavinin yalnız bir tedavi sonucu vardır (1...*).

UML Diyagramı

Oluşturulan tabloların ve ara tabloların UML diyagramı aşağıda verilmiştir:



Veri tabanı şemasının SQL Server'de oluşturulması

CREATE TABLE ile UML diyagramındaki varlıkları tablo olarak eklediğimiz kodlar:

```
--ilk tablomuz olan müşteriler tablomuz
CREATE TABLE Musteriler (
   Musteri_ID INT PRIMARY KEY IDENTITY(1,1),
```

```
    MusteriAdi VARCHAR(50) NOT NULL,
    MusteriSoyadi VARCHAR(50) NOT NULL,
    MusteriTelefon VARCHAR(15),
    MusteriAdres VARCHAR(MAX)
);

-- ikinci tablomuzda müşterilerin hayvanları olacak
CREATE TABLE Hayvanlar (
    Hayvan_ID INT PRIMARY KEY IDENTITY(1,1),
    HayvanTur VARCHAR(50) NOT NULL,
    HayvanYas INT,
    HayvanCinsiyet VARCHAR(10),
    Musteri_ID INT NOT NULL,
    FOREIGN KEY (Musteri_ID) REFERENCES Musteriler(Musteri_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

-- Veterinerlerin tutulduğu tablo
CREATE TABLE Veterinerler (
    Veteriner_ID INT PRIMARY KEY IDENTITY(1,1),
    VeterinerAdi VARCHAR(50) NOT NULL,
    VeterinerSoyadi VARCHAR(50) NOT NULL,
    VeterinerTelefon VARCHAR(15)
);

-- yapılan tedavinin özellikleri
CREATE TABLE Tedaviler (
    Tedavi_ID INT PRIMARY KEY IDENTITY(1,1),
    TedaviBaslangicTarihi DATE NOT NULL,
    TedaviBitisTarihi DATE,
    Hayvan_ID INT NOT NULL,
    Veteriner_ID INT NOT NULL,
    FOREIGN KEY (Hayvan_ID) REFERENCES Hayvanlar(Hayvan_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (Veteriner_ID) REFERENCES Veterinerler(Veteriner_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

--kullanılan Ilaclar
CREATE TABLE Ilaclar (
    Ilac_ID INT PRIMARY KEY IDENTITY(1,1),
    IlacAdi VARCHAR(50) NOT NULL,
    IlacFiyati DECIMAL(10, 2)
);
```

```
-- o tedavide kullanılan ilaçlar (Ara Tablo)
CREATE TABLE Tedavi_Ilac (
   Tedavi_ID INT NOT NULL,
   Ilac_ID INT NOT NULL,
   IlacDozu VARCHAR(50),
   PRIMARY KEY (Tedavi_ID, Ilac_ID),
   FOREIGN KEY (Tedavi_ID) REFERENCES Tedaviler(Tedavi_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (Ilac_ID) REFERENCES Ilaclar(Ilac_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

-- hayvanın hastalığı için
CREATE TABLE Hastalik (
   Hastalik_ID INT PRIMARY KEY IDENTITY(1,1),
   HastalikAdi VARCHAR(50) NOT NULL
);

-- hastalık için tedavi (Ara Tablo)
CREATE TABLE Tedavi_Hastalik (
   Tedavi_ID INT NOT NULL,
   Hastalik_ID INT NOT NULL,
   PRIMARY KEY (Tedavi_ID, Hastalik_ID),
   FOREIGN KEY (Tedavi_ID) REFERENCES Tedaviler(Tedavi_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (Hastalik_ID) REFERENCES Hastalik(Hastalik_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

-- hayvan öldü mü iyileşti mi
CREATE TABLE TedaviSonucu (
   Sonuc_ID INT PRIMARY KEY IDENTITY(1,1),
   SonucDurumu VARCHAR(50),
   HayvanDurumu NVARCHAR(50) NOT NULL CHECK (HayvanDurumu IN ('Olmustur',
'Iyilesmiştir', 'Tedavi Ediliyor')),
   Tedavi_ID INT NOT NULL,
   FOREIGN KEY (Tedavi_ID) REFERENCES Tedaviler(Tedavi_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

Veri Tabanı Tablolarına Örnek Veriler Eklenmesi

INSERT INTO kullanarak proje gereksinimlerini göstermek amacıyla veri tabanı tablolarına örnek veriler eklediğimiz kodlar:

```
-- kullanici konumundaki müşteriler
INSERT INTO Musteriler (MusteriAdi, MusteriSoyadi, MusteriTelefon, MusteriAdres)
VALUES
('Ahmet', 'Dağistanli', '0551641235', 'Ankara'),
('Emre', 'Altundağ', '05522167548', 'Istanbul'),
('Yusuf', 'Şakir', '05341737542', 'Mardin');

-- müsterilerin hayvanları
INSERT INTO Hayvanlar (HayvanTur, HayvanYas, HayvanCinsiyet, Musteri_ID)
VALUES
('Eşşek', 10, 'Disi', 1),
('zurafa', 2, 'Erkek', 2),
('kaplumbağa', 1, 'Disi', 3);

-- tedavi ediciler
INSERT INTO Veterinerler (VeterinerAdi, VeterinerSoyadi, VeterinerTelefon)
VALUES
('Ayşe', 'Celik', '05902312334'),
('Zülfikar', 'Er', '06107239284'),
('Fatih', 'Dağkiran', '03322342354');

-- lazım olan tarihler
INSERT INTO Tedaviler (TedaviBaslangicTarihi, TedaviBitisTarihi, Hayvan_ID,
Veteriner_ID)
VALUES
('2025-01-01', '2024-01-05', 1, 1),
('2025-04-10', '2024-01-15', 2, 2),
('2025-02-20', '2024-01-22', 3, 3);

INSERT INTO Ilaclar (IlacAdi, IlacFiyati)
VALUES
('gripin', 50.00),
('ateş düşürücü', 30.00),
('kas gevşetici', 20.00);

-- ara tablomuz
INSERT INTO Tedavi_Ilac (Tedavi_ID, Ilac_ID, IlacDozu)
VALUES
(6, 1, '1x1'), -- 1. günde 1 tane
(7, 3, '2x1'),
```

```
(3, 3 , '3x3'), -- 2. günde 1 tane
(10, 2, '1x2'); -- 1. 2 tane

-- tedavi_id = 1 ve ilac_id = 3 olan veriyi eklemek için
INSERT INTO Tedavi_Ilac (Tedavi_ID, Ilac_ID , IlacDozu)
VALUES (3, 3 , '3x3');
SELECT * FROM Tedavi_Ilac;

-- rahatsızlığı
INSERT INTO Hastalik (HastalikAdi)
VALUES
('kolera'),
('kirilma'),
('tansiyon');


INSERT INTO Tedavi_Hastalik (Tedavi_ID, Hastalik_ID)
VALUES
(1, 1), -- 1. kolera  icindi
(2, 2), -- 2. tedavi kirilma icindi
(3, 3); -- 3. tedavi tansiyon icindi

-- hayvana ne olduğu burada
INSERT INTO TedaviSonucu (SonucDurumu, HayvanDurumu, Tedavi_ID)
VALUES
('olumlu', 'Iyilesmistir', 1), -- Ilk tedavi basarili oldu
('Devam Ediyor', 'Tedavi Ediliyor', 2), -- Ikinci tedavi devam ediyor
('olumsuz', 'Olmustur', 3); -- Ucuncu tedavi basarisiz oldu
```

Stored Procedure Kullanımı

CREATE PROCEDURE ile tablolara veri ekleme ve güncelleme işlemlerini otomatikleştirdiğimiz saklı yordam kodları:

```
-- 1. Musteri Ekle
CREATE PROCEDURE MusteriEkle
   @MusteriAdi VARCHAR(50),
   @MusteriSoyadi VARCHAR(50),
   @MusteriTelefon VARCHAR(15),
   @MusteriAdres VARCHAR(MAX)
AS
BEGIN
   INSERT INTO Musteriler (MusteriAdi, MusteriSoyadi, MusteriTelefon,
MusteriAdres)
```

```
    VALUES (@MusteriAdi, @MusteriSoyadi, @MusteriTelefon, @MusteriAdres);
END;
GO

-- 2. Hayvan Ekle
CREATE PROCEDURE HayvanEkle
    @HayvanTuru VARCHAR(50),
    @HayvanYasi INT,
    @HayvanCinsiyeti VARCHAR(10),
    @Musteri_ID INT
AS
BEGIN
    INSERT INTO Hayvanlar (HayvanTur, HayvanYas, HayvanCinsiyet, Musteri_ID)
    VALUES (@HayvanTuru, @HayvanYasi, @HayvanCinsiyeti, @Musteri_ID);
END;
GO

-- 3. Veteriner Ekle
CREATE PROCEDURE VeterinerEkle
    @VeterinerAdi VARCHAR(50),
    @VeterinerSoyadi VARCHAR(50),
    @VeterinerTelefon VARCHAR(15)
AS
BEGIN
    INSERT INTO Veterinerler (VeterinerAdi, VeterinerSoyadi, VeterinerTelefon)
    VALUES (@VeterinerAdi, @VeterinerSoyadi, @VeterinerTelefon);
END;
GO

-- 4. Tedavi Ekle
CREATE PROCEDURE TedaviEkle
    @BaslangicTarihi DATE,
    @BitisTarihi DATE,
    @Hayvan_ID INT,
    @Veteriner_ID INT
AS
BEGIN
    INSERT INTO Tedaviler (TedaviBaslangicTarihi, TedaviBitisTarihi, Hayvan_ID,
Veteriner_ID)
    VALUES (@BaslangicTarihi, @BitisTarihi, @Hayvan_ID, @Veteriner_ID);
END;
GO

-- 5. Musteriye Gore Hayvanlari Listele
CREATE PROCEDURE MusteriyeGoreHayvanlariListele
    @Musteri_ID INT
AS
```

```
BEGIN
   SELECT Hayvan_ID, HayvanTur, HayvanYas, HayvanCinsiyet
   FROM Hayvanlar
   WHERE Musteri_ID = @Musteri_ID;
END;
GO

-- 6. Tedavi Detaylarini Listele
CREATE PROCEDURE TedaviDetaylariniListele
   @Tedavi_ID INT
AS
BEGIN
   SELECT t.TedaviBaslangicTarihi, t.TedaviBitisTarihi, h.HayvanTur, h.HayvanCinsiyet,
v.VeterinerAdi, v.VeterinerSoyadi
   FROM Tedaviler t
   JOIN Hayvanlar h ON t.Hayvan_ID = h.Hayvan_ID
   JOIN Veterinerler v ON t.Veteriner_ID = v.Veteriner_ID
   WHERE t.Tedavi_ID = @Tedavi_ID;
END;
GO

-- 7. Tedaviye Ilac Ekle
CREATE PROCEDURE TedaviyeIlacEkle
   @Tedavi_ID INT,
   @Ilac_ID INT,
   @IlacDozu VARCHAR(50)
AS
BEGIN
   INSERT INTO Tedavi_Ilac (Tedavi_ID, Ilac_ID, IlacDozu)
   VALUES (@Tedavi_ID, @Ilac_ID, @IlacDozu);
END;
GO

-- 8. Tedaviye Hastalik Ekle
CREATE PROCEDURE TedaviyeHastalikEkle
   @Tedavi_ID INT,
   @Hastalik_ID INT
AS
BEGIN
   INSERT INTO Tedavi_Hastalik (Tedavi_ID, Hastalik_ID)
   VALUES (@Tedavi_ID, @Hastalik_ID);
END;
GO

-- 9. Tedavi Sonucu Guncelle
CREATE PROCEDURE TedaviSonucuGuncelle
   @Tedavi_ID INT,
```

```
    @SonucDurumu VARCHAR(50),
    @HayvanDurumu NVARCHAR(50)
AS
BEGIN
    UPDATE TedaviSonucu
    SET SonucDurumu = @SonucDurumu, HayvanDurumu = @HayvanDurumu
    WHERE Tedavi_ID = @Tedavi_ID;
END;
GO
```

Trigger Kullanımı

CREATE TRIGGER ile veri tabanı tablolarında değişiklik yapıldığında tetiklenmeyi sağlayan kodlar:

```
-- Trigger: ilacFiyatıGüncelleme
DROP TRIGGER IF EXISTS trgAfterUpdateIlacFiyat;
GO

CREATE TRIGGER trgAfterUpdateIlacFiyat
ON Ilaclar
AFTER UPDATE
AS
BEGIN
    DECLARE @IlacID INT;
    DECLARE @YeniFiyat DECIMAL(10, 2);

    SELECT TOP 1 @IlacID = Ilac_ID, @YeniFiyat = IlacFiyati FROM INSERTED;

    PRINT 'İlaç_ID nin fiyati ' + CAST(@IlacID AS NVARCHAR) + ' şöyle güncellendi: ' +
CAST(@YeniFiyat AS NVARCHAR);
END;
GO

-- Trigger: TedaviDurumGüncelleme
DROP TRIGGER IF EXISTS trgAfterUpdateTedaviDurum;
GO

CREATE TRIGGER trgAfterUpdateTedaviDurum
ON TedaviSonucu
AFTER UPDATE
AS
BEGIN
    DECLARE @TedaviID INT;
```

```
    DECLARE @YeniDurum NVARCHAR(50);

    SELECT TOP 1 @TedaviID = Tedavi_ID, @YeniDurum = HayvanDurumu FROM
INSERTED;

    PRINT 'Tedavi_IDnin statüsü ' + CAST(@TedaviID AS NVARCHAR) + ' şöyle
güncellendi: ' + @YeniDurum;
END;
GO

-- Trigger: Olum'u Ekledikten Sonra
DROP TRIGGER IF EXISTS trgAfterInsertOlum;
GO

CREATE TRIGGER trgAfterInsertOlum
ON OlenHayvanlar
AFTER INSERT
AS
BEGIN
    DECLARE @AnimalID INT;
    DECLARE @DeathCause NVARCHAR(255);

    SELECT TOP 1 @AnimalID = Hayvan_ID, @DeathCause = HayvanOlumNedeni FROM
INSERTED;

    PRINT 'Bir hayvan öldü ' + CAST(@AnimalID AS NVARCHAR) + ',ölüm sebebi ' +
@DeathCause;
END;
GO

-- Trigger: Iyilesen i ekledikten sonra
DROP TRIGGER IF EXISTS trgAfterInsertIyilesen;
GO

CREATE TRIGGER trgAfterInsertIyilesen
ON IyilesenHayvanlar
AFTER INSERT
AS
BEGIN
    DECLARE @AnimalID INT;
    DECLARE @RecoveryDate NVARCHAR(50);

    SELECT TOP 1 @AnimalID = Hayvan_ID, @RecoveryDate =
CAST(HayvanIyilesmeTarihi AS NVARCHAR) FROM INSERTED;

    PRINT 'Bir hayvan iyileşti' + CAST(@AnimalID AS NVARCHAR) + ', iyilesme tarihi ' +
@RecoveryDate;
```

```
END;
GO

-- Trigger: Musteri'yi Sildikten Sonra
DROP TRIGGER IF EXISTS trgAfterDeleteMusteri;
GO

CREATE TRIGGER trgAfterDeleteMusteri
ON Musteriler
AFTER DELETE
AS
BEGIN
   DECLARE @DeletedMusteriID INT;

   SELECT TOP 1 @DeletedMusteriID = Musteri_ID FROM DELETED;

   PRINT 'Musteri_ID ye sahip müşteri' + CAST(@DeletedMusteriID AS NVARCHAR) + '
silindi';
END;
GO
```

Transaction Kullanımı

Transaction yönetimi için commit ve rollback işlevlerinin kullanıldığı kodlar:

```
USE Veterinerlik;
GO

-- Transaction Yapisi ile COMMIT ve ROLLBACK Ornegi
BEGIN TRANSACTION;

BEGIN TRY
   -- 1. Yeni musteri ekleme
   INSERT INTO Musteriler (MusteriAdi, MusteriSoyadi, MusteriTelefon,
MusteriAdres)
   VALUES ('Test', 'Musteri', '05551234567', 'Test Adresi');

   PRINT 'Müşteri başariyla eklendi.';

   -- 2. Eklenen musterinin ID'sini al
   DECLARE @Musteri_ID INT;
   SET @Musteri_ID = SCOPE_IDENTITY();

   -- 3. Yeni hayvan ekleme
```

```
    INSERT INTO Hayvanlar (HayvanTur, HayvanYas, HayvanCinsiyet, Musteri_ID)
    VALUES ('Tavsan', 3, 'Erkek', @Musteri_ID);

    PRINT 'Hayvan başariyla eklendi.';

    -- Islemi basarili bir sekilde tamamlama
    COMMIT TRANSACTION;
    PRINT 'İşlem başariyla gerçekleştirildi.';

END TRY
BEGIN CATCH
    -- Hata durumunda islemi geri al
    PRINT 'Bir hata oluştu. İşlemi geri ';
    ROLLBACK TRANSACTION;

    -- Hatanin detaylarini yazdir
    PRINT ERROR_MESSAGE();
END CATCH;
GO
```