T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ

VERİ TABANI YÖNETİM SİSTEMLERİ İLE GEMİ SEFERLERİ YÖNETİM UYGULAMASI

ŞEVVAL ÖZEREN
220501028

DR. ÖĞR. ÜYESİ ERCAN ÖLÇER

05.05.2024
https://github.com/sevvalozrn

1 GİRİŞ
1.1 Projenin amacı

- Bu projede gemi seferlerini yönetmek üzere veri tabanı tasarımı ve arayüz kullanılarak bir yazılım geliştirilmesi istenmektedir. Herhangi bir programlama dili kısıtlaması yoktur.
- Projede gerçekleştirilmesi beklenenler:
    - SQL ile bir veri tabanı oluşturularak veri yönetimi sağlanması.
    - Yazılımın bir arayüz kullanılarak görselleştirilmesi.
    - Gemiler, seferler, kaptanlar, mürettebat ve limanlar varlıkları ile ilgili veri yönetimi yapılmalı.
    - Verilen varlıkların özelliklerine uygun olarak tablolar oluşturulmalı.
    - Her varlık için bir sınıf oluşturularak varlığa veri ekleme, düzenleme, silme gibi işlemlerin yapılacağı fonksiyonlar da tanımlanmalı.
    - Tüm bu veri tabanı ile ilgili işlemler yazılım kullanılarak yapılmalıdır.

2 GEREKSİNİM ANALİZİ
2.1 Arayüz gereksinimleri

- Kullanıcı arayüz gereksinimleri, kullanıcıların veriler üzerindeki ekleme, silme, düzenleme gibi çeşitli işlemleri kolaylıkla gerçekleştirebilmesini sağlamak amacıyla aşağıdaki maddelerde yazıldığı şekilde belirlenmiştir:
-

1. Ana Menü Ekranı: Kullanıcıya Gemi, Sefer, Liman, Kaptan ve Mürettebat gibi temel veri kategorileri üzerinde değişiklik yapmak veya görüntülemek için erişim sağlayacak bir ana menü oluşturulmuştur.

2. Veri Yönetim Ekranları: Varlıkların üzerinde ekleme, değiştirme, silme gibi işlemlerin yapılmasını sağlayacak ekran tasarımları yapılmıştır.
    - Gemi Yönetim Ekranı İçin: Kullanıcıların gemi bilgilerini görüntüleyebileceği; gemi ekleyebileceği, düzenleyebileceği ve silebileceği bir ekran oluşturulmuştur. Ayrıca geminin türüne göre (yolcu kapasitesi, petrol kapasitesi, konteyner kapasitesi vb.) farklı ekstra özellikler olduğundan ekleme ve düzenleme kısımlarının tasarımları daha komplike tutulmuştur.
    - Sefer Yönetim Ekranı İçin: Kullanıcıların sefer bilgilerini yönetebileceği; yeni seferler ekleyebileceği, düzenleyebileceği ve silebileceği bir ekran tasarlanmıştır.
    - Liman Yönetim Ekranı İçin: Kullanıcıların liman bilgilerini yönetebileceği; yeni limanlar ekleyebileceği, düzenleyebileceği ve silebileceği bir ekran tasarımı olmuştur.
    - Kaptan ve Mürettebat Yönetim Ekranı İçin: Kullanıcıların kaptan ve mürettebat bilgilerini ekleyebileceği, düzenleyebileceği ve silebileceği bir ekran tasarımı oluşturulmuştur.

3. Arama ve Filtreleme: Kullanıcıların varlık tabloları üzerinde herhangi bir değişiklik yapması durumunda varlığın özellikleri arasında hızlıca arama yapılması ve belirli kriterlere göre filtreleme yapılmasıdır.

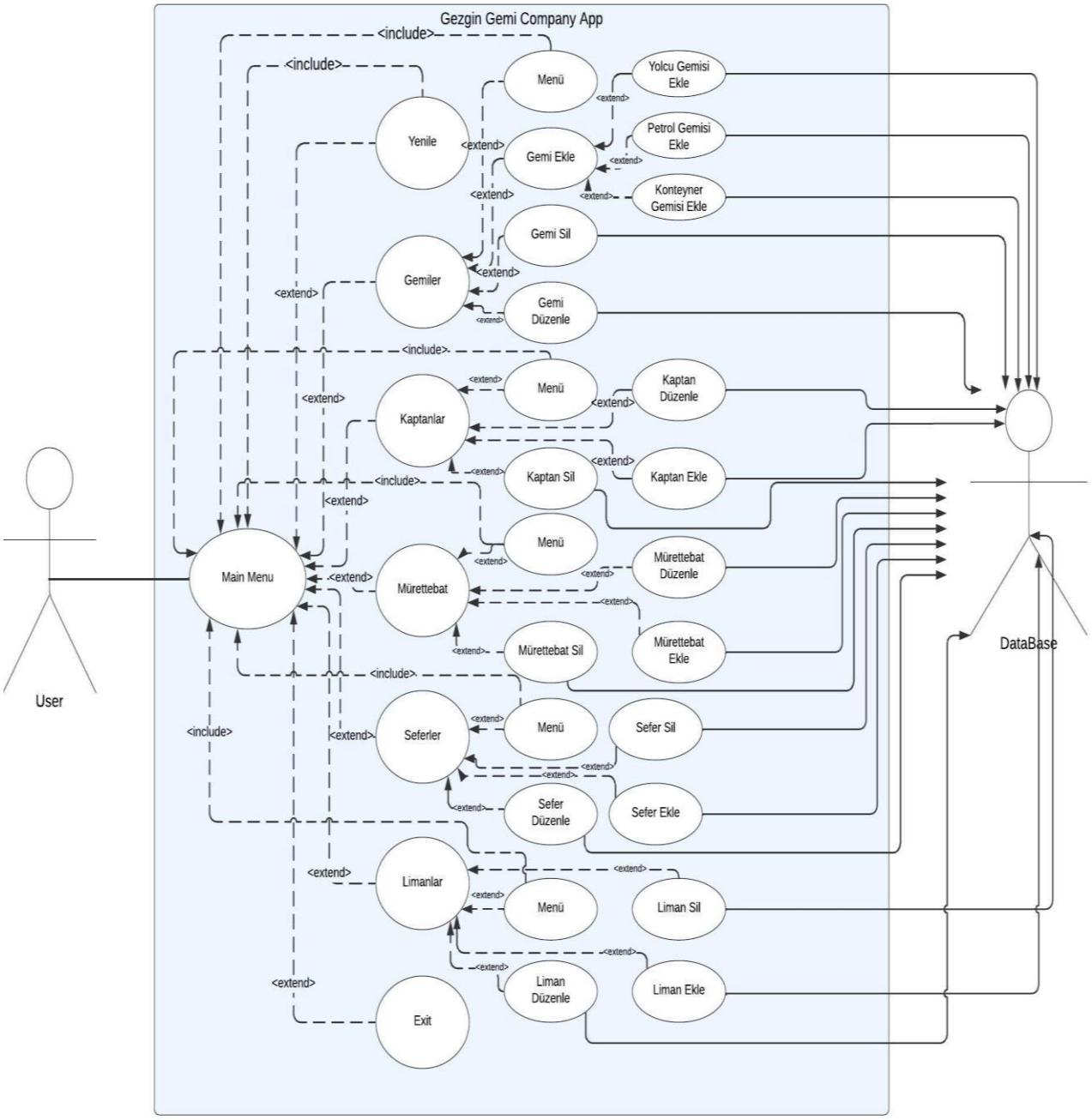
4. Kullanım Kolaylığı: Arayüzün kullanıcılar için kolayca anlaşılacak ve kullanabilecek bir tasarımı vardır.

2.2 Fonksiyonel gereksinimler

- **Gemi Yönetimi Fonksiyonları:** Yeni gemi ekleyebilme. Mevcut gemileri görüntüleme, düzenleme ve silme. Her gemi için türüne uygun bilgilerin girilmesi (yolcu kapasitesi, petrol kapasitesi, konteyner kapasitesi gibi).
- **Sefer Yönetimi Fonksiyonları:** Yeni sefer oluşturma. Mevcut seferleri görüntüleme, düzenleme ve silme. Seferler için gerekli bilgilerin girilmesi (başlangıç tarihi, bitiş tarihi, gemi bilgileri, kaptanlar ve mürettebat bilgileri gibi).
- **Liman Yönetimi Fonksiyonları:** Yeni liman ekleyebilme. Mevcut limanları görüntüleme, düzenleme ve silme. Her liman için gerekli bilgilerin girilmesi (ülke, nüfus, pasaport gerekliliği gibi).
- **Kaptan ve Mürettebat Yönetimi Fonksiyonları:** Yeni kaptan ve mürettebat ekleyebilme. Mevcut kaptan ve mürettebatları görüntüleme, düzenleme ve silme. Her personel için gerekli bilgilerin girilmesi (ad, soyad, adres, doğum tarihi, işe giriş tarihi, lisans bilgisi gibi).
- **Arama ve Filtreleme Fonksiyonları:** Veriler arasında hızlı arama yapabilme. Belirli kriterlere göre verileri filtreleme.
- **Hata ve Geri Bildirim İşlemleri:** Kullanıcıya hata durumlarında bilgilendirme yapabilme. Geri bildirim verme mekanizmaları.

2.3 Use-Case diyagramı

Use Case Diyagramı:



3 TASARIM

3.1 Mimari tasarım

- Veri Modeli Tasarımı:

  Ödev veri tabanı için sistemdeki tüm varlıkların (Gemiler, Seferler, Kaptanlar, Mürettebat, Limanlar) verilerini tutmak için kullanılacaktır bunun için her varlığa yönelik ayrı bir tablo oluşturulmuştur. Her tabloda da ilgili varlığın özelliklerini temsil eden sütunlar eklenmiştir.

- Sınıf Tasarımı:

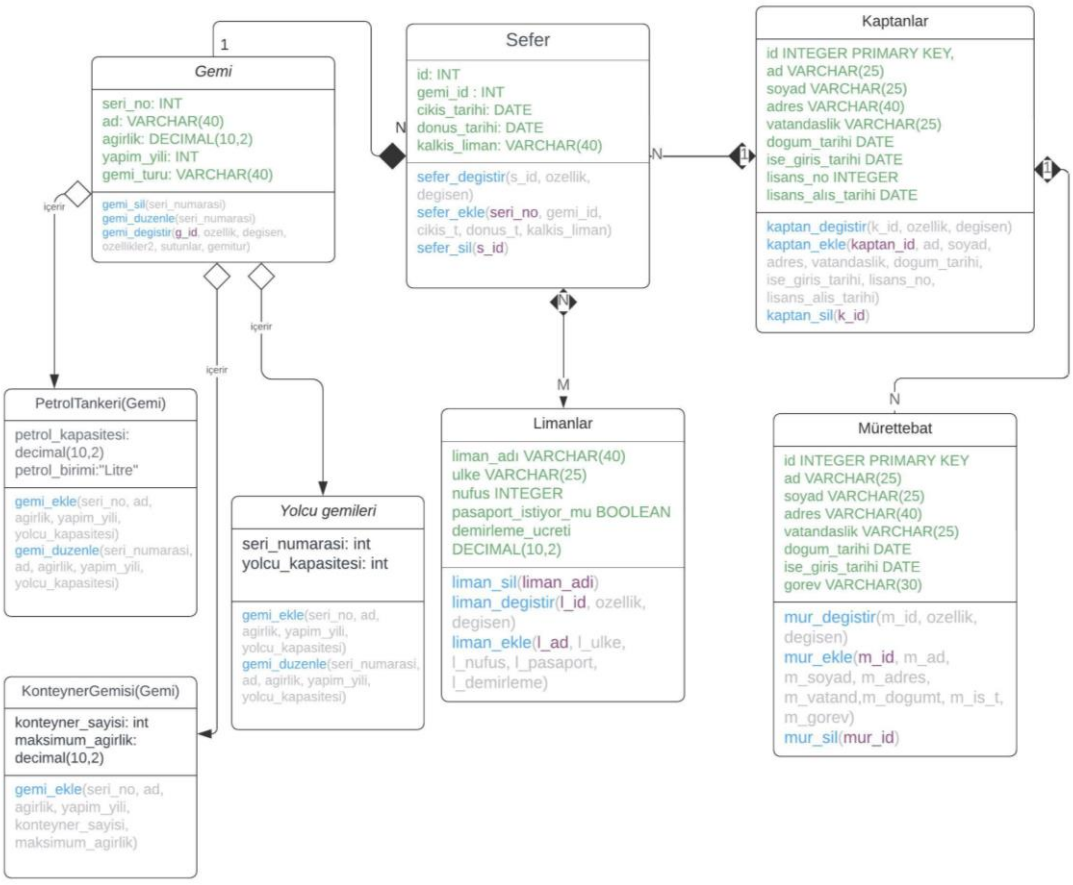
  Önceden oluşturulan her varlık için yazılım üzerinden düzenlemeler yapabilmek amacıyla birer sınıf oluşturulmuştur. her sınıf, ilgili varlığın

özelliklerini ve bu özelliklere erişmek için gerekli metodları içermektedir.

- Form Ekranı Tasarımı:
  Verilerin eklenmesi, silinmesi, düzenlenmesi için tkinter ile kullanıcı arayüzü (formlar) oluşturulmuştur. Formlar ile kullanıcıların ilgili varlıkla ilgili verileri girmesi, düzenlemesi ve silmesine olanak sağlanmıştır.

- Nesne Yönelimli Programlama (OOP):
  Proje, gerektiği gibi nesne yönelimli programlama prensiplerine uygun olarak tasarlanmıştır. Her varlık için bir sınıf oluşturulmuş ve ilgili işlemler bu sınıflar üzerinden gerçekleştirilmiştir. Sınıfların bazı özelliklerinde Polymorphism(çok biçimlilik) kullanılmıştır.

- Veri Tabanı ile Bağlantı:
  Projede, SQLite veri tabanı ile bağlantı kurulmuş; tablolar oluşturulmuş ve bağlantı sayesinde veri tabanındaki verileri okuyabilmiş, düzenleyebilmiş ve silebilmesi sağlanmıştır.

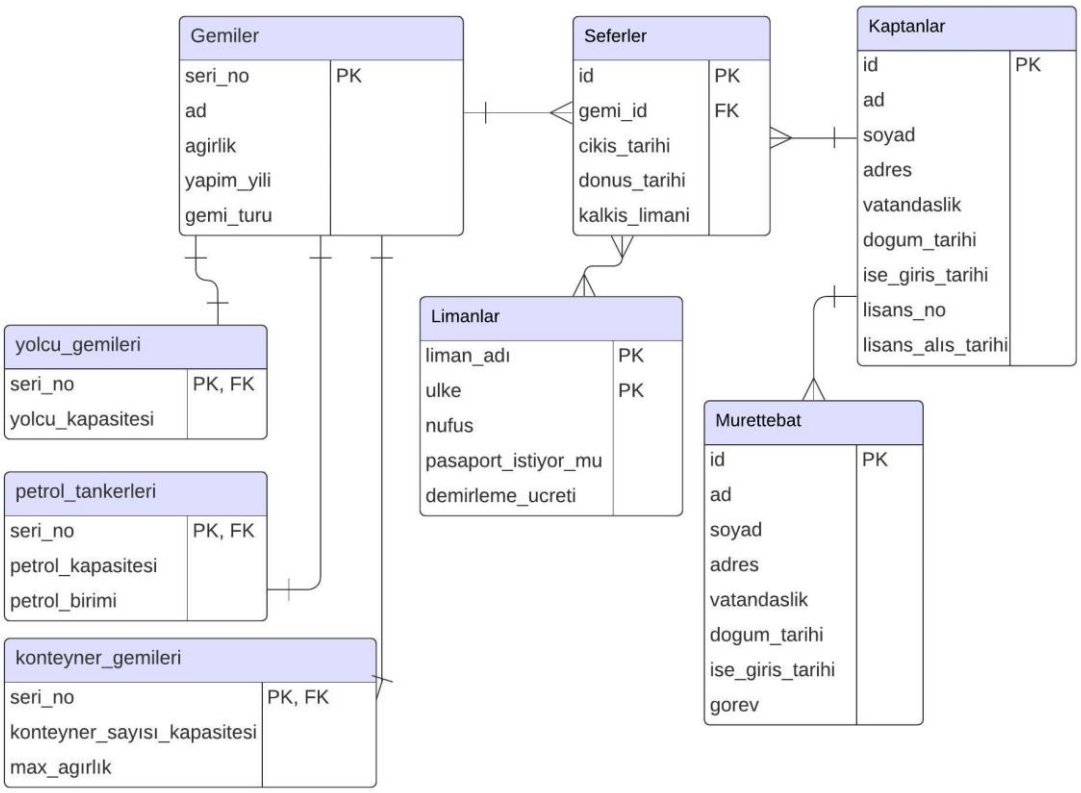


3.2 Kullanılacak teknolojiler

- Yazılım için Python programlama dili kullanılmıştır. Veri tabanı oluştururken ise SQLite kullanılmıştır.
- Arayüz için “tkinter” kütüphanesi kullanılmıştır. Python ve SQLite arasındaki bağlantıyı sağlamak amacıyla da “sqlite3” kütüphanesi dahil edilmiştir. Tkinter kütüphanesinden extra olarak “Combobox” ve “Messagebox” gibi modüller de dahil edilmiştir.

3.3 Veri tabanı tasarımı

- Ödevde SQLite veri tabanı tercih edilmiş ve işlemler onun üzerinden yapılmıştır. Ödevde istenen varlıkların tabloları oluşturulmuş, aralarındaki bağlantılar eklenmiş, tabloların özellikleri eklenmiş ve tablolar arasındaki ilişkilere göre yabancı anahtar, birincil anahtar vs. üzerinde düzenlemeler yapılmıştır.
- ER Diyagramı:

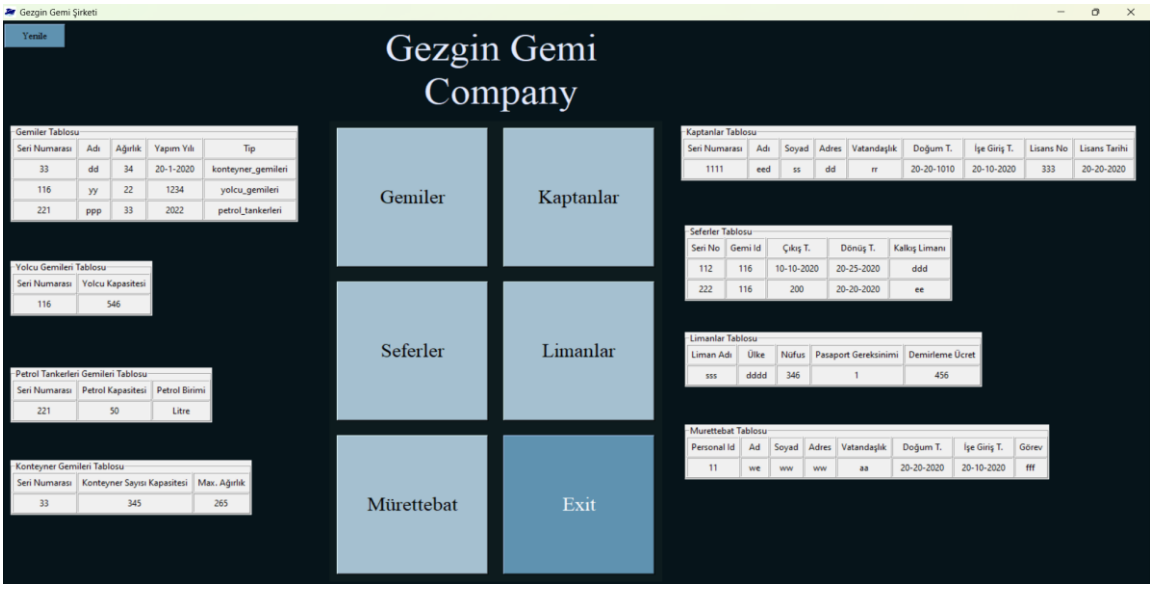


3.4 Kullanıcı arayüzü tasarımı

- Kullanıcı arayüzü tasarımında özellikle kolay kullanım için basit dizaynlar üzerinden tasarım yapılmıştır. Kullanıcıyı öncelikle bir ana menü ekranı karşılamaktadır buradan hangi varlık üzerinde düzenleme yapacağını seçebilir, sayfayı yenileyebilir (tabloların güncel gözükmesi için), isterse "Exit" butonundan çıkış yapabilir ve de varlıklarını güncel haliyle tablolarını görebilir.

- **Yazılım Üzerinden Görsel Açıklama:**

Uygulama açıldığında kullanıcıyı ilk olarak aşağıdaki ekran karşılıyor. (Tablolardaki veriler test amaçlı eklenmiş verilerdir. Bir veri olmadığında sadece sütunlar gözükmektedir.)



Sağ üstteki “Yenile” tuşu: Ekranı yenileyerek tabloların güncel halini göstermektedir.
Merkezdeki Varlık İsimleri Butonları: Herhangi bir butona tıklandığında o veri üzerinden ekleme, silme, düzenleme işlemlerinin yapılacağı bir ekrana yönlendirmektedir.
“Exit” Butonu: Bu butona basıldığında iste pencereyi kapatmaktadır.

Eğer Gemiler varlığı seçilirse:

Gemiler varlığı üzerinden düzenleme yapmak için bu ekran çıkar. Orta sütundaki gemi ekleme kısmında farklı gemi tipleri eklemek için üç farklı buton verilmiştir. Bu her gemi tipinin kendine has özellikleri olduğundan veri girilmesini kolaylaştırmak için yapılmıştır.
Örneğin Yolcu Gemisi Ekle Butonuna basarsak:

Bu şekilde bir ekranla karşılaşırız.

Ayrıca Gemi Düzenleme sütununda da düzenlenecek geminin ID’sine göre şöyle bir ekran oluşmaktadır:

Ben burada 221 Seri No’lu geminin bilgilerini çağırdım. Görüldüğü gibi gem gemi tablosu hem de gemi tipine özel tablosu ekrana verilerek kullanıcının bunlara göre özellik değişimi yapmasına olanak tanınmıştır. Tabloların hemen altındaki Combobox’a tıklanarak geminin özellikleri çıkıyor ve seçilen özelliğe göre hemen aşağısındaki entry box’tan değişim yapılıyor ve ”Gemi Değiştir” butonu ile de değişim kaydediliyor.
Ayrıca her sekmede sol üstte bir “Menü” tuşu var ona basıldığında da ekran kapatılıyor ve kullanıcının ana menüye dönmesi sağlanıyor.
Bu ekran diğer varlıklar için de aynı şekilde işliyor sadece tek fark diğer varlıklarda ekstra atablolar olmadığından orta sütunda direkt veri girişi alınıyor.

Aşağıdaki gibi:

4 UYGULAMA

4.1 Kodlanan bileşenlerin açıklamaları

- **Sınıflar:**

Temel metotlarıyla sınıflar aşağıdaki gibidir. Her bir varlık için ayrı bir sınıf oluşturulacağı belirtilmişti. Gemi varlığı için aşağıdaki sınıf oluşturulmuştur. Her gemi tipi varlığı için ise bir altındaki sınıflar oluşturulmuştur. Gemi varlıklarında gemi ekle ve gemi düzenle fonksiyonlarının işleyişleri aynı olduğundan sadece Yolcu Gemisi sınıfındaki işlemler açıkça gösterilmiş diğerleri tamamiyle aynıdır.

```
class Gemi:

    def __init__(self, seri_numarasi, ad, agirlik, yapim_yili, gemi_turu,
yolcu_kapasitesi=None, petrol_kapasitesi=None, konteyner_sayisi=None,
maksimum_agirlik=None):

    @staticmethod
    def gemi_sil(seri_numarasi):

    @staticmethod
    def gemi_duzenle(seri_numarasi):
```

```
class YolcuGemi(Gemi):
    def __init__(self, seri_numarasi, ad, agirlik, yapim_yili,
yolcu_kapasitesi):

    @staticmethod
    def gemi_ekle(seri_no, ad, agirlik, yapim_yili, yolcu_kapasitesi):
        cursor.execute('''INSERT INTO gemiler(seri_no, ad, agirlik,
yapim_yili, gemi_turu) VALUES (?, ?, ?, ?, ?)''',
                       (seri_no, ad, agirlik, yapim_yili,
"yolcu_gemileri"))

        cursor.execute('''INSERT INTO yolcu_gemileri(seri_no,
```

```
yolcu_kapasitesi) VALUES (?, ?)''',
                        (seri_no, yolcu_kapasitesi))
        baglanti.commit()
        print("Yolcu gemisi eklendi.")


    @staticmethod
    def gemi_duzenle(seri_numarasi, ad, agirlik, yapim_yili,
yolcu_kapasitesi):
        cursor.execute('''UPDATE gemiler SET ad = ?, agirlik = ?,
yapim_yili = ? WHERE seri_no = ?''',
                        (ad, agirlik, yapim_yili, seri_numarasi))
        cursor.execute('''UPDATE yolcu_gemileri SET yolcu_kapasitesi = ?
WHERE seri_no = ?''',
                        (yolcu_kapasitesi, seri_numarasi))
        baglanti.commit()
        print("Yolcu gemisi düzenlendi.")


class PetrolTankeri(Gemi):
    def __init__(self, seri_numarasi, ad, agirlik, yapim_yili,
petrol_kapasitesi, petrol_birimi):


    @staticmethod
    def gemi_ekle(seri_no, ad, agirlik, yapim_yili, petrol_kapasitesi,
petrol_birimi):



class KonteynerGemisi(Gemi):
    def __init__(self, seri_numarasi, ad, agirlik, yapim_yili,
konteyner_sayisi, maksimum_agirlik):

    @staticmethod
    def gemi_ekle(seri_no, ad, agirlik, yapim_yili, konteyner_sayisi,
maksimum_agirlik):
```

Aynı şekilde sefer, kaptan, mürettebat ve limanlar için de aşağıdaki sınıfın aynı temelinde sınıflar oluşturulmuştur.

```
class Sefer:

    def __init__(self, s_id, gemi_id, cikis_tarihi, donus_tarihi,
kalkis_liman):

    def sefer_sil(s_id):

    def sefer_ekle(seri_no, gemi_id, cikis_t, donus_t, kalkis_liman):
```

```
    @staticmethod
    def sefer_degistir(s_id, ozellik, degisen):
```

Veritabanını oluşturmak için de bağlantı kurulmuştur:

```
baglanti = sql.connect("proje.db")

# sql komutlarını python içinde de çalıştırabilmek için
cursor = baglanti.cursor()
```

Bu bağlantı kullanılarak da aşağıda bir varlık üzerinden örneği verilmiş tablolar oluşturulmuştur:

```
# Gemiler için tablo oluşturma
cursor.execute('''CREATE TABLE IF NOT EXISTS gemiler (

                    seri_no INTEGER PRIMARY KEY,

                    ad VARCHAR(40),

                    agirlik DECIMAL(10,2),

                    yapim_yili INTEGER,

                    gemi_turu VARCHAR(40)

                )''')
```

(Aynı şekilde her varlık için tablo oluşturulmuştur.)

Daha sonra Menü penceresi aşağıdaki özellikler ile oluşturulmuştur:

```
penc = Tk()
penc.title("Gezgin Gemi Şirketi")
penc.iconbitmap("gemilogo2.ico")
penc.configure(bg="#06141A")
penc.geometry("1530x750+0+0")
```

Pencereye butanlar ve tablolar eklenmiştir. Buton eklenişi:

```
buton0 = Button(label_frame, text="Gemiler", font=("Times New Roman", 20),
bg="#A5C0D0", padx=60, pady=10,
                width=uniform_buton_boyut, height=uniform_buton_boyut,
command=lambda: Gemi_Pencereleri.gemiler_pencere())
buton0.grid(row=0, column=0, sticky="nsew", padx=10, pady=10)
```

Tabloların eklenişi:

```
cursor.execute("SELECT * FROM gemiler WHERE seri_no = ?",
(seri_numarasi,))
gemi = cursor.fetchone()

if gemi:
    # Tek satırlık tabloyu oluştur
    table_frame = LabelFrame(gemiler_p, text="Gemi Özellikleri")
    table_frame.grid(row=7, column=2, sticky="n", padx=0, pady=(0,10))
```

```
    # Başlıklar
    columns = ["Seri No", "Ad", "Ağırlık", "Yapım Yılı", "Gemi Türü"]

    # Başlıkları ekleme
    for col_index, col_name in enumerate(columns):
        label = Label(table_frame, text=col_name, padx=10, pady=5,
relief="ridge")
        label.grid(row=0, column=col_index, sticky="nsew")

    # Verileri ekleme
    for col_index, col_data in enumerate(gemi):
        label = Label(table_frame, text=col_data, padx=10, pady=5,
relief="ridge")
        label.grid(row=1, column=col_index, sticky="nsew")
else:
    messagebox.showerror("Hata", "Belirtilen Id sahip gemi bulunamadı.")
```

Daha sonra her varlık için seçim ekranları oluşturulmuş aynı yollar izlenerek tablolar, butonlar, ComboBoxlar ve Entry Boxlar eklenmiştir.

4.2 Görev dağılımı

4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

Başta veri tabanı oluşturma, python ile bağlantı kurma vb. Durumlarda fazlaca zorluk yaşadım çünkü daha önce hiç veri tabanı ile bir proje hazırlamamıştım. Bu konuda tecrübesiz olduğumdan internetteki çeşitli kaynaklardan yardım aldım. Kurulum, bağlama vb. işlemleri video eğitimlerden izleyerek hallettim. Arayüz geliştirme konusunda daha tecrübeli olduğumdan pek bir sorun yaşamadım sadece görsel dizaynı ayarlamak yorucuydu.

4.4 Proje isterlerine göre eksik yönler

Projede yapılması beklenen tüm işlemler yapılmıştır.

5 TEST VE DOĞRULAMA

5.1 Yazılımın test süreci

Yazılım testi için aşağıda verilen kod yazılmıştır ve gemiler tablosuna önceden eklenen bir gemi varlığının seri numarası girilerek işleyişi test edilmiştir(Örnek olarak konteyner gemileri sınıfı kullanılmıştır):

```
def konteyner_gemisi_test():
    # Konteyner gemisi bilgileri
    seri_no = 154
    ad = "Konteyner Gemisi 1"
    agirlik = 3000
    yapim_yili = 2012
    konteyner_sayisi = 2000
    maksimum_agirlik = 500000

    # Konteyner gemisi ekleme işlemi
    KonteynerGemisi.gemi_ekle(seri_no, ad, agirlik, yapim_yili,
konteyner_sayisi, maksimum_agirlik)
```

```
    # Konteyner gemisinin özelliklerini sorgulama ve ekrana yazdırma
    Gemi.gemi_duzenle(seri_no)

    # Konteyner gemisi düzenleme işlemi
    yeni_ad = "Yeni Konteyner Gemisi Adı"
    yeni_agirlik = 3200
    yeni_yapim_yili = 2014
    yeni_konteyner_sayisi = 2500
    yeni_maksimum_agirlik = 600000
    KonteynerGemisi.gemi_duzenle(seri_no, yeni_ad, yeni_agirlik,
yeni_yapim_yili, yeni_konteyner_sayisi,
                                 yeni_maksimum_agirlik)

    # Konteyner gemisi silme işlemi
    KonteynerGemisi.gemi_sil(seri_no)

# KonteynerGemisi test fonksiyonunu çağır
konteyner_gemisi_test()
```

Kodda eklenmesi için bilgileri verilen gemi tabloda gözükmektedir:

154	Konteyner Gemisi 1	3000	2012	konteyner_gemileri

Aynı zamanda konteyner gemileri tablosunda da diğer özellikleri ile birlikte gözükmektedir:

Konteyner Gemileri Tablosu

Seri Numarası	Konteyner Sayısı Kapasitesi	Max. Ağırlık
3	2000	500000
154	2000	500000

Aynı şekilde önceden oluşturulan kaptan varlığının ID'si ile Kaptan sınıfı testi aşağıdaki gibidir:

```
def kaptan_test():
    # Örnek kaptan bilgileri
    kaptan_id = 1
    ad = "Ahmet"
    soyad = "Yılmaz"
    adres = "İstanbul, Türkiye"
    vatandaslik = "Türkiye"
    dogum_tarihi = "1990-01-01"
    ise_giris_tarihi = "2010-01-01"
    lisans_no = "123456789"
    lisans_alis_tarihi = "2010-01-01"

    # Kaptan ekleme işlemi
    Kaptan.kaptan_ekle(kaptan_id, ad, soyad, adres, vatandaslik, dogum_tarihi,
ise_giris_tarihi, lisans_no,
                       lisans_alis_tarihi)

    # Kaptan bilgilerini sorgulama
    cursor.execute("SELECT * FROM kaptanlar WHERE id = ?", (kaptan_id,))
```

```
    kaptan = cursor.fetchone()

    # Kaptan bilgilerini ekrana yazdırma
    if kaptan:
        print("Eklenen Kaptan Bilgileri:")
        print(kaptan)
    else:
        print("Belirtilen ID'ye sahip kaptan bulunamadı.")

    # Kaptanın lisans numarasını güncelleme
    yeni_lisans_no = "987654321"
    Kaptan.kaptan_degistir(kaptan_id, "lisans_no", yeni_lisans_no)

    # Güncellenmiş kaptan bilgilerini sorgulama
    cursor.execute("SELECT * FROM kaptanlar WHERE id = ?", (kaptan_id,))
    guncellenmis_kaptan = cursor.fetchone()

    # Güncellenmiş kaptan bilgilerini ekrana yazdırma
    if guncellenmis_kaptan:
        print("Güncellenen Kaptan Bilgileri:")
        print(guncellenmis_kaptan)
    else:
        print("Belirtilen ID'ye sahip kaptan bulunamadı.")

    # Kaptanı silme
    Kaptan.kaptan_sil(kaptan_id)

    # Silinen kaptan bilgilerini sorgulama
    cursor.execute("SELECT * FROM kaptanlar WHERE id = ?", (kaptan_id,))
    silinen_kaptan = cursor.fetchone()

    # Silinen kaptan bilgilerini ekrana yazdırma
    if silinen_kaptan:
        print("Silinen Kaptan Bilgileri:")
        print(silinen_kaptan)
    else:
        print("Belirtilen ID'ye sahip kaptan bulunamadı.")

# Kaptan test fonksiyonunu çağır
kaptan_test()
```

Test çıktıları:

Kaptan eklendi.
Eklenen Kaptan Bilgileri:
(1, 'Ahmet', 'Yılmaz', 'İstanbul, Türkiye', 'Türkiye', '1990-01-01', '2010-01-01', 123456789, '2010-01-01')
lisans_no özelliği güncellendi.
Güncellenen Kaptan Bilgileri:
(1, 'Ahmet', 'Yılmaz', 'İstanbul, Türkiye', 'Türkiye', '1990-01-01', '2010-01-01', 987654321, '2010-01-01')
Belirtilen ID'ye sahip kaptan bulunamadı.

Process finished with exit code 0

Diğer varlık sınıfarı da aynı şekilde test edilmiş ve doğruluğu onaylanmıştır.

5.2 Yazılımın doğrulanması

- Uygulanan testler sonucunda yazılımın doğru biçimde çalıştığı saplanmıştır. Tüm sınıflar ve fonksiyonlar beklendiği gibi birbirleri ile düzgün bir uyum içerisinde çalışmaktadırlar.

5.3 Kaynakça

- https://lucid.app/lucidchart
- https://academy.patika.dev
- https://coolors.co
- https://image.online-convert.com/
- https://www.youtube.com/watch?v=Fv82RX4cWW4