
**T.C**

**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ**

**PROJE KONUSU:**

# ARAYÜZ İLE VERİTABANI YÖNETİMİ

**ÖĞRENCİ ADI:**

**AHMET CAN BOSTANCI**

**DİLA SERAY TEGÜN**

**ÖĞRENCİ NUMARASI:**

**220501031**

**220501022**

**GİTHUB LİNKLERİ :** https://github.com/Bozokhalat https://github.com/dilaseray

**DERS SORUMLUSU:**

**DR. ÖĞR. ÜYESİ NURBANU ALBAYRAK**

**TARİH: 5.05.2024**

# 1. Giriş

## 1.1 Projenin Amacı

Bu projenin amacı, bir gemi yönetim sistemi geliştirmektir. Bu sistem, çeşitli gemi türleri (petrol, konteyner, yolcu gemileri), limanlar, seferler ve kaptanlarla ilgili kayıtları yönetmek için kullanılacaktır. Projenin başlıca hedefleri şunlardır:

- Gemi kayıtlarını yönetmek: Gemi ekleme, güncelleme, silme.
- Liman ve sefer bilgilerini takip etmek: Liman ve sefer kayıtlarını eklemek, güncellemek ve silmek.
- Kaptan ve mürettebat kayıtlarını yönetmek: Kaptan ve mürettebat bilgilerini eklemek, güncellemek ve silmek.
- Kullanıcı dostu bir arayüz sağlamak: Gemi yönetimi için kullanıcı dostu bir arayüz oluşturmak.
- Veritabanı ile etkileşim kurmak: SQLite kullanarak güvenilir bir veri depolama sağlamak.

# 2. Gereksinim Analizi

## 2.1 Arayüz Gereksinimleri

Kullanıcı arayüzü gereksinimleri şunları içerir:

- Kullanıcıların kayıtları ekleyebileceği, güncelleyebileceği ve silebileceği basit bir arayüz.
- Seferlerin, gemilerin ve limanların bir tabloda görüntülenmesi.
- Farklı gemi türlerine (petrol, konteyner, yolcu) özgü formlar.
- Radyobutonlar, metin kutuları, düğmeler ve tablolardan oluşan kullanıcı arayüz bileşenleri.

Donanım arayüzü gereksinimleri:

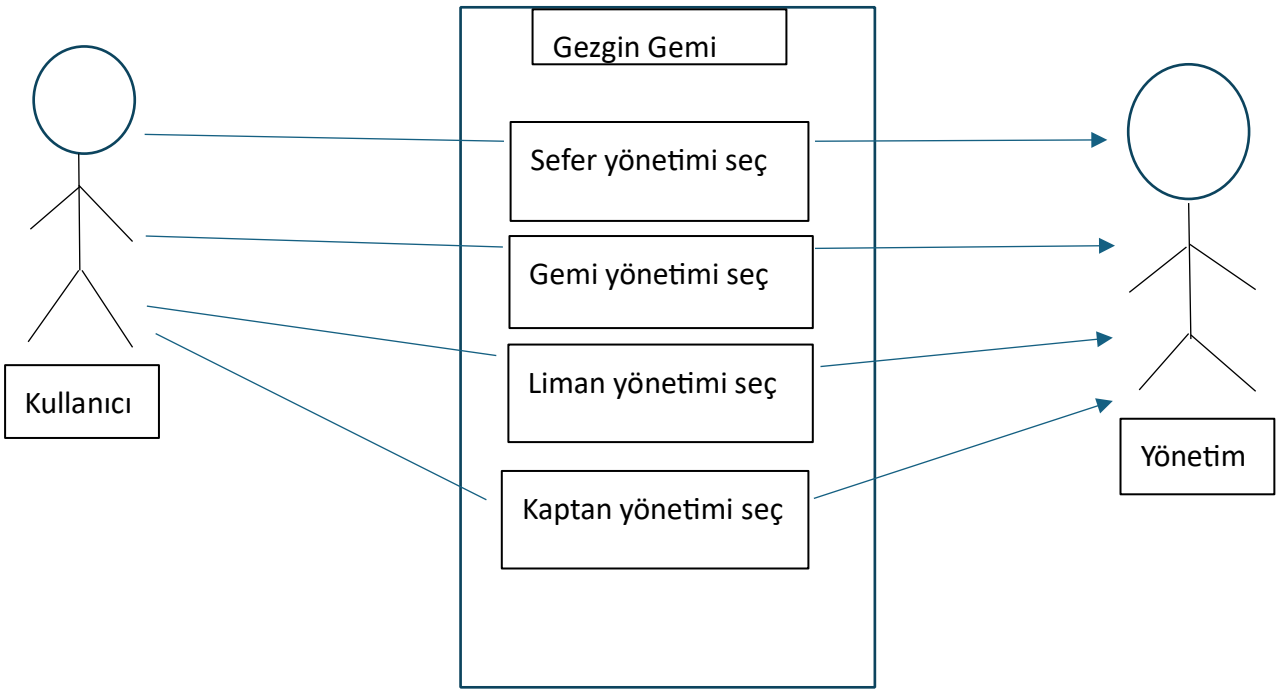
- Geliştirme ve çalıştırma için bir bilgisayar.
- Çalıştırmak için uygun işletim sistemi ve PyQt5 kurulumu.

## 2.2 Fonksiyonel Gereksinimler

Fonksiyonel gereksinimler şunlardır:

- Kullanıcıların gemi kayıtlarını eklemesine, güncellemesine ve silmesine olanak tanıyan fonksiyonlar. • Kullanıcıların liman ve sefer bilgilerini güncellemesine ve görüntülemesine izin veren fonksiyonlar.
- Veritabanı bağlantısı ve işlemlerini gerçekleştirebilen fonksiyonlar. • Veri girişinin doğru yapılmasını sağlamak için doğrulama kontrolleri.

## 2.3 Use-Case Diyagramı




- **Kullanıcı**: Sistemle etkileşime giren kişi.

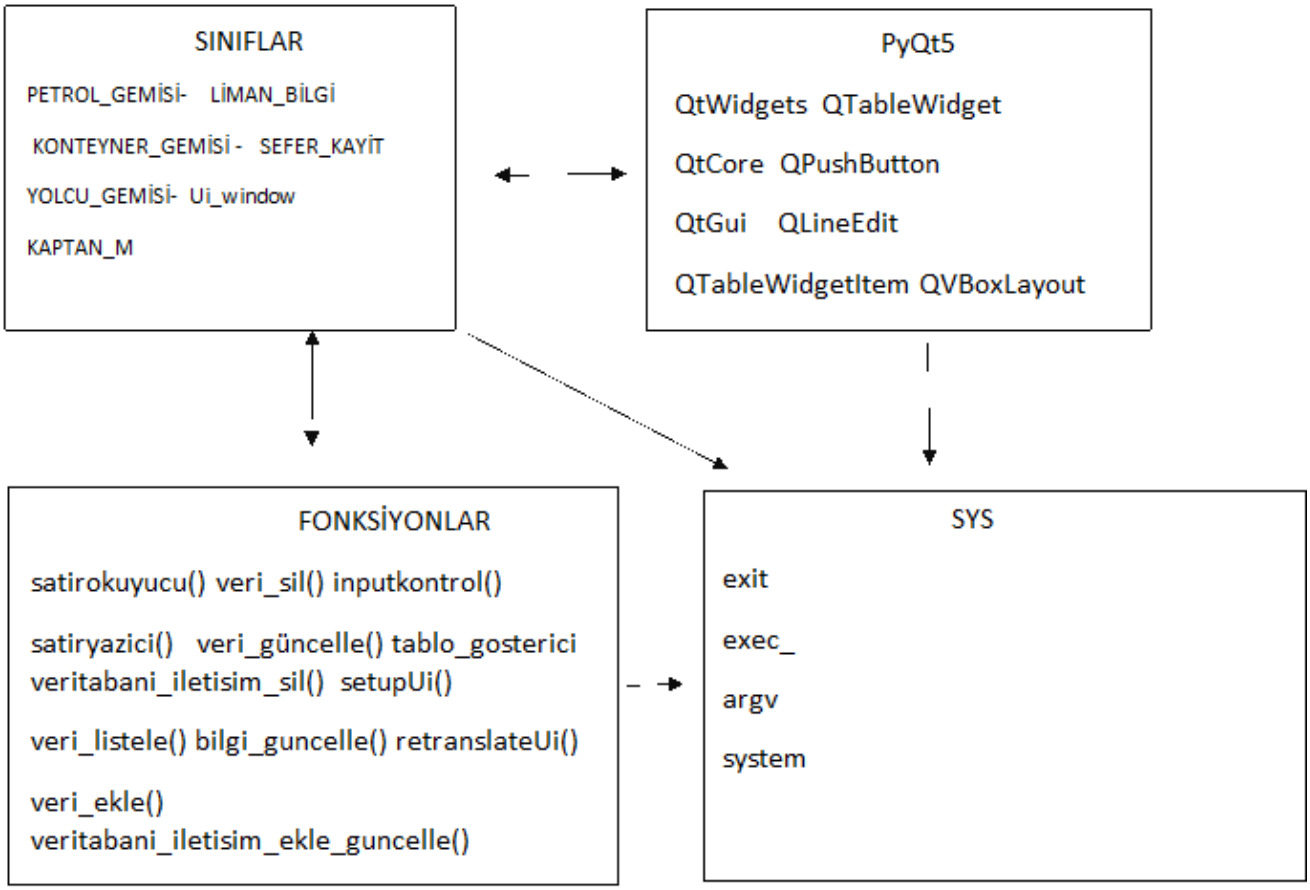
- **Yönetim Use Case'leri**: Kullanıcı, çeşitli yönetim görevlerini gerçekleştirebilir:
    - **Sefer Yönetimi**: Sefer eklemek, güncellemek ve silmek. ○ **Gemi Yönetimi**: Gemi eklemek, güncellemek ve silmek.
    - **Liman Yönetimi**: Liman eklemek, güncellemek ve silmek.
    - **Kaptan Yönetimi**: Kaptan eklemek, güncellemek ve silmek.

# 3. Tasarım

## 3.1 Mimari Tasarım

Mimari tasarımda modüller ve sınıflar kullanılarak sistemin ana bileşenleri açıklanır. Modüller şunlardır:

- Gemi Yönetimi: Petrol, konteyner ve yolcu gemileri için ayrı sınıflar içerir.
- Kaptan ve Mürettebat Yönetimi: Kaptan ve mürettebat kayıtlarını yönetmek için sınıflar içerir.
- Liman Yönetimi: Liman bilgilerini yönetmek için sınıflar içerir.
- Sefer Yönetimi: Sefer kayıtlarını yönetmek için sınıflar içerir.
- Arayüz: PyQt5 kullanılarak arayüz bileşenlerini oluşturur.




## 3.2 Kullanılacak Teknolojiler

Yazılımın yazılacağı dil ve diğer teknolojiler:

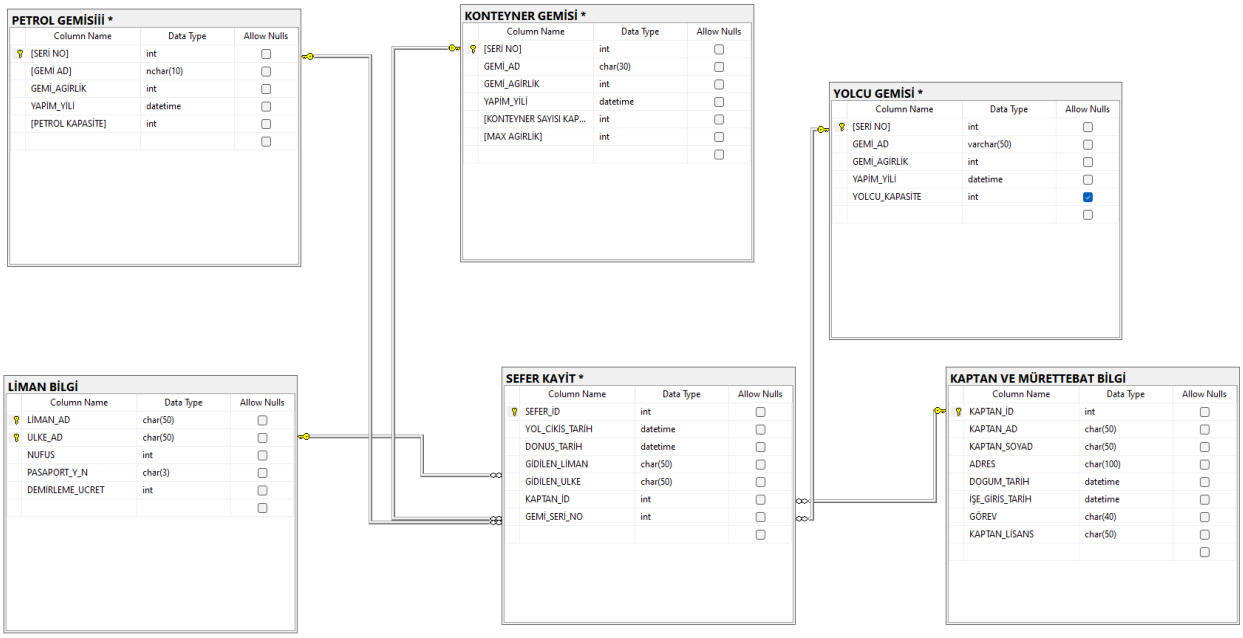
- Programlama dili: Python
- Kullanılan kütüphaneler: PyQt5, sqlite3
- Veritabanı: SQLite
- Kullanılan platform: Windows/Linux/macOS

## 3.3 Veritabanı Tasarımı

Veritabanı tasarımı ve ilişkiler hakkında bilgi:

- ER Diyagramı: Gemi türleri, limanlar, seferler ve kaptanlar için bir ER diyagramı içerir.

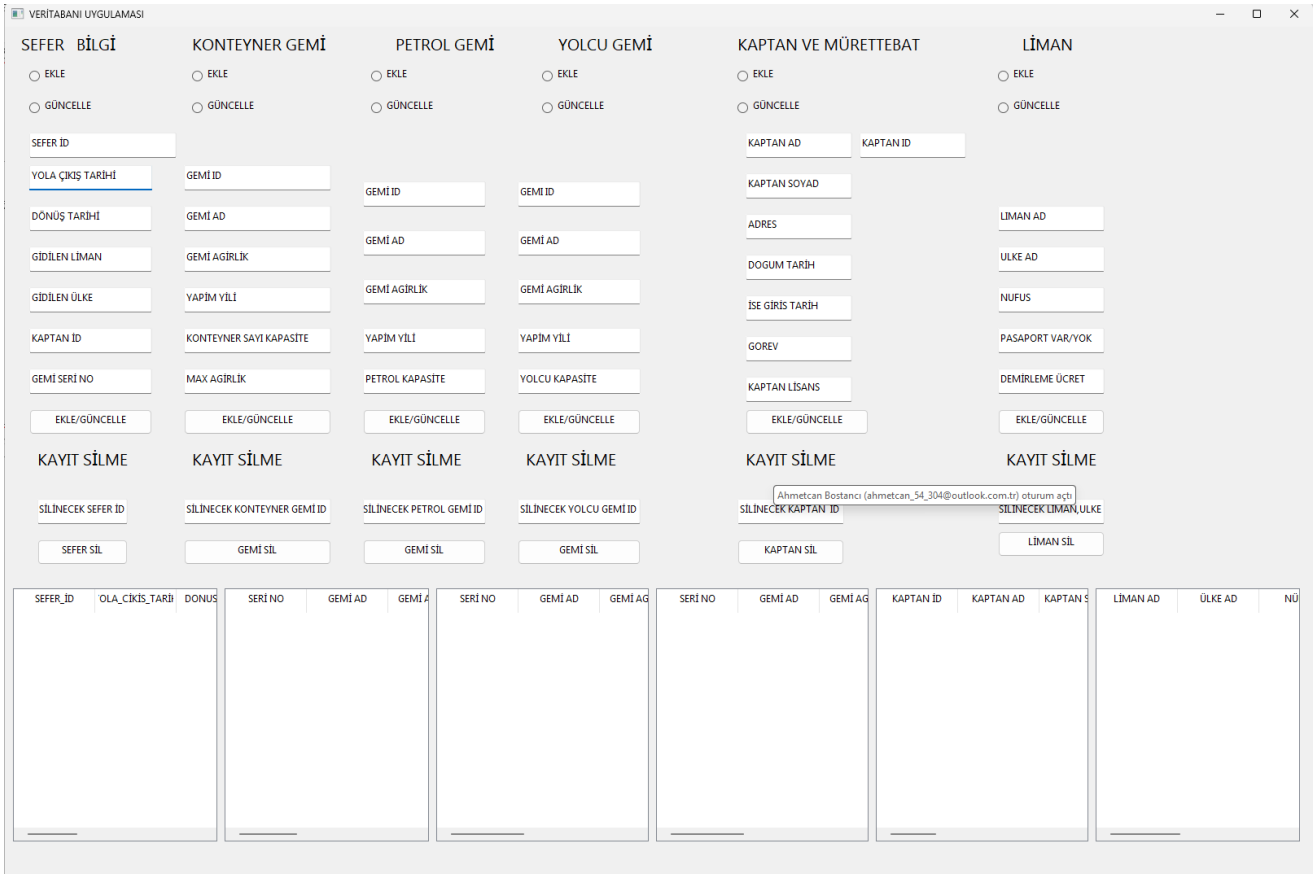
- Tablolar: Gemi türleri, limanlar, kaptanlar ve seferler için tablolar.
- Birincil ve yabancı anahtar ilişkileri.




# 3.4 Kullanıcı Arayüzü Tasarımı

Kullanıcı arayüzü tasarımı için önemli unsurlar:

- Arayüz bileşenleri: QLineEdit, QPushButton, QTableWidget gibi bileşenler kullanılmıştır.




**ArayüzÇalıştırma açıklaması** :

Arayüzden veri tabanına veri ekleme /güncelleme ve silme işlemleri yapmak için çeşitli opsiyonel düğmeleri kullanabilirsiniz. Örnek olarak petrol gemi eklemek için ekle seçeneğini seçip gerekli değerleri girip EKLE/GÜNCELLE butonuna basıldıktan sonra eklenecektir. EKLE seçeneğinde iken ID kendiniz giremezsin ID otomatik olarak atanmaktadır. EKLE seçeneğini seçtiğinizde ID girişi otomatik pasif hale gelecektir. GÜNCELLE seçeneği gelince ID girmek aktif hale gelir ID girerek ve değerleri girerek güncelleme yapabilirsiniz bu işlemler tüm bölmeler için aynıdır.

SEFER ekleme tablosu tüm tablolarla iletişim halindedir girilen gemi seri numaraları, kaptan id, gidilen liman, gidilen ülke  ilgi tablolarda olursa sefer tablosu ekleme işlemi gerçekleştirir. Mevcut olmayan gidilen liman, gidilen ulke,kaptan id, gemi seri no sefer tablosuna kaydedilemez. Sefer tablosunda kaptan id, gemi seri numaraları farklı satırlarda aynı olamaz.

# 4. Uygulama

## 4.1 Kodlanan Bileşenlerin Açıklamaları

```python
import sqlite3
import os

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QTableWidget, QTableWidgetItem
```

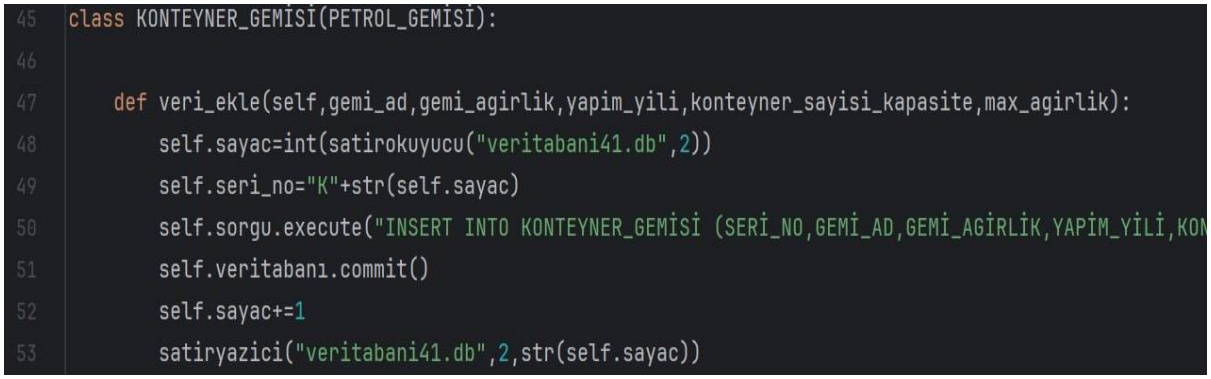
Bu kod snippet'i, PyQt5 ve SQLite kullanılarak oluşturulmuş bir grafiksel kullanıcı arayüzü (GUI) uygulamasının başlangıç kısmını açıklamaktadır.

```python
def satirokuyucu(file,satir):

    db=sqlite3.connect(file)
    islemci = db.cursor()
    veri=islemci.execute("SELECT * FROM İNDEX_TABLO ").fetchall()
    return veri[0][satir-1]

def satiryazici(file, satir, veri):

    db=sqlite3.connect(file)
    islemci=db.cursor()
    if satir==1:
        islemci.execute("UPDATE İNDEX_TABLO SET P=?",(veri,))
    elif satir==2:
        islemci.execute("UPDATE İNDEX_TABLO SET K=?",(veri,))
    elif satir==3:
        islemci.execute("UPDATE İNDEX_TABLO SET Y=?",(veri,))
    elif satir==4:
        islemci.execute("UPDATE İNDEX_TABLO SET KA=?",(veri,))
    elif satir==5:
        islemci.execute("UPDATE İNDEX_TABLO SET SF=?",(veri,))
    db.commit()
```

Bu kod, SQLite veritabanında bulunan bir "INDEX_TABLO" tablosundan veri okumak ve güncellemek için iki fonksiyon içerir.

- **satirokuyucu** fonksiyonu, belirtilen bir veritabanı dosyasını (**file**) açar ve "INDEX_TABLO" tablosundaki tüm satırları alarak, belirli bir satırda bulunan değeri döndürür (**satir** parametresi ile belirlenen sütun index'i).

- **satiryazici** fonksiyonu, belirlenen bir satırdaki değeri güncellemek için kullanılır. Güncellenecek satır, **satir** parametresine göre belirlenir ve yeni değer **veri** parametresiyle sağlanır. Güncellemeler, uygun sütun için **UPDATE** sorgusu kullanılarak yapılır ve ardından değişiklikler **db.commit()** ile veritabanına kaydedilir.



```
class KONTEYNER_GEMİSİ(PETROL_GEMİSİ):

    def veri_ekle(self,gemi_ad,gemi_agirlik,yapim_yili,konteyner_sayisi_kapasite,max_agirlik):
        self.sayac=int(satirokuyucu("veritabani41.db",2))
        self.seri_no="K"+str(self.sayac)
        self.sorgu.execute("INSERT INTO KONTEYNER_GEMİSİ (SERİ_NO,GEMİ_AD,GEMİ_AGİRLİK,YAPİM_YİLİ,KON
        self.veritabanı.commit()
        self.sayac+=1
        satiryazici("veritabani41.db",2,str(self.sayac))
```

Bu kod, bir SQLite veritabanında "KONTEYNER_GEMİSİ" adlı tabloya yeni bir konteyner gemisi kaydı eklemek için bir sınıf ve fonksiyon tanımlar.

**veri_ekle**: Yeni bir konteyner gemisi kaydı ekler.
**Parametreler**: Gemi adı, gemi ağırlığı, yapım yılı, konteyner kapasitesi ve maksimum ağırlık.

**İşleyişi**:

**satirokuyucu** fonksiyonunu kullanarak bir sayaç değeri alır, ardından yeni seri numarasını oluşturur ("K" ile başlayarak sayacı ekler).
Veritabanına **INSERT INTO** sorgusuyla yeni konteyner gemisi kaydını ekler.
Değişiklikleri **commit** ile veritabanına kaydeder.
Sayaç değerini artırır, böylece sonraki gemi için benzersiz bir seri numarası sağlar.

Bu kod, belirli verilerle yeni bir konteyner gemisi kaydı oluşturmak için kullanılır. Benzersiz bir seri numarası oluşturur ve yeni gemi bilgisini veritabanına ekler, ardından değişiklikleri kalıcı hale getirir.

```
    def veri_sil(self,id):
        self.sorgu.execute("DELETE FROM KONTEYNER_GEMİSİ WHERE SERİ_NO=? ",(id,))
        self.veritabanı.commit()
    def veri_güncelle(self, id,gemi_ad,gemi_agirlik,yapim_yili,konteyner_sayisi_kapasite,max_agirlik):
        self.sorgu.execute("UPDATE KONTEYNER_GEMİSİ SET GEMİ_AD=?,GEMİ_AGİRLİK=?,YAPİM_YİLİ=?,KONTEYNER_SAYISI_KAPASİTE = ?
        self.veritabanı.commit()
```

Bu fonksiyonlar, "KONTEYNER_GEMİSİ" tablosunda kayıt silme ve güncelleme işlemleri yapar:

- **veri_sil**: Belirtilen seri numarasını kullanarak bir kaydı siler.
- **veri_güncelle**: Gemi ile ilgili detayları, seri numarasına göre günceller.

Bu işlemler, tablodaki verilerin dinamik olarak yönetilmesini sağlar ve değişikliklerin kalıcı hale getirilmesi için commit işlemini içerir.

```
    def baslat(self):
        self.gemi_idler=[]
        for gemi_id in self.sorgu.execute("SELECT * FROM SEFER_KAYİT").fetchall():
            self.gemi_idler.append(gemi_id[6])
        self.kaptan=[]
        for kaptan_id in self.sorgu.execute("SELECT * FROM SEFER_KAYİT").fetchall():
            self.kaptan.append(kaptan_id[5])
        self.liman=[]
        for liman in self.sorgu.execute("SELECT * FROM SEFER_KAYİT").fetchall():
            self.liman.append([liman[3],liman[4]])
        self.sefer=[]
        for sefer in self.sorgu.execute("SELECT * FROM SEFER_KAYİT").fetchall():
            self.sefer.append(sefer[0])
        self.bilgi_guncelle()
    def bilgi_guncelle(self):
        self.selfkaptan=[]
        for selfkaptan in self.sorgu.execute("SELECT KAPTAN_İD FROM KAPTAN_MÜRETTEBAT ").fetchall():
            self.selfkaptan.append(selfkaptan)
```

- **self.gemi_idler**: "SEFER_KAYİT" tablosundan altıncı sütundaki gemi seri numaralarını alır ve listeye ekler.
- **self.kaptan**: "SEFER_KAYİT" tablosundan beşinci sütundaki kaptan kimliklerini alır ve listeye ekler.
- **self.liman**: "SEFER_KAYİT" tablosundan üçüncü ve dördüncü sütunlardaki liman ve ülke bilgilerini alarak bir listeye ekler.

- **self.sefer**: "SEFER_KAYİT" tablosundan ilk sütundaki sefer kimliklerini alır ve listeye ekler.
- Son olarak, **bilgi_guncelle()** metodunu çağırarak ek verileri günceller
- **selfkaptan**: "KAPTAN_MÜRETTEBAT" tablosundan kaptan kimliklerini alır ve listeye ekler.

Bu kodun işlevi, veritabanından verileri alarak sınıf içinde listeler oluşturmak ve güncellemektir. Bu listeler,sistemdeki seferler, gemiler, kaptanlar ve limanlarla ilgili bilgileri içerir. Böylece, diğer metodlar bu listeleri kullanarak işlem yapabilir. Kod, veri tabanında bulunan belirli tabloları tarayarak ilgili verileri birden fazla listeye atar, böylece işlemler daha hızlı ve verimli hale gelir.

```
217 if not os.path.exists("veritabani41.db"):
218
219     veritabani=sqlite3.connect("veritabani41.db")
220
221     islem=veritabani.cursor()
222
223     islem.execute('''
224
225         CREATE TABLE İNDEX_TABLO(
226
227             P INTEGER NOT NULL,
228             K INTEGER NOT NULL,
229             Y INTEGER NOT NULL,
230             KA INTEGER NOT NULL,
231             SF INTEGER NOT NULL
232         )
233         ''')
```

- **islem.execute('" ... "')** ile SQL **CREATE TABLE** sorgusu çalıştırılır.

- Bu sorgu, "INDEX_TABLO" adlı bir tablo oluşturur ve aşağıdaki sütunlara sahip olmasını sağlar: ○ **P, K, Y, KA, SF**: Her biri **INTEGER** türünde ve **NOT NULL** (boş olamaz).

Bu tablo, uygulamanın diğer bileşenleri için bir çeşit sayaç veya indeks tablosu olarak kullanılır. Örneğin, belirli bir sayacın tutulması veya farklı kategoriler için değerlerin saklanması için kullanılabilir.

Özetle, bu kod, SQLite veritabanının varlığını kontrol eder ve yoksa bir tane oluşturur. Ardından, belirli bir tablo ("INDEX_TABLO") oluşturmak için SQL sorgusu çalıştırır. Bu tablo, INTEGER türünde beş sütuna sahiptir ve uygulamada sayaç veya indeksleme için kullanılabilir.

```
234     veritabani.commit()
235     islem.execute("INSERT INTO İNDEX_TABLO (P,K,Y,KA,SF) VALUES (1,1,1,1,1)")
236     islem.execute('''
237     CREATE TABLE KONTEYNER_GEMİSİ(
238 
239             SERİ_NO TEXT PRIMARY KEY ,
240             GEMİ_AD TEXT NOT NULL,
241             GEMİ_AGİRLİK INTEGER NOT NULL,
242             YAPİM_YİLİ TIMESTAMP NOT NULL,
243             KONTEYNER_SAYISI_KAPASİTE INTEGER NOT NULL,
244             MAX_AGİRLİK INTEGER NOT NULL   )
245 
246 ''')
247     islem.execute('''
```

Bu kod, SQLite veritabanında bir veritabanı dosyasının varlığını kontrol eder ve eğer yoksa bir tane oluşturur. Ardından, belirli bir tabloyu ("INDEX_TABLO") oluşturmak için bir SQL sorgusu çalıştırır. İşlevleri şunlardır:

### 1. Veritabanının Var Olup Olmadığını Kontrol Etme

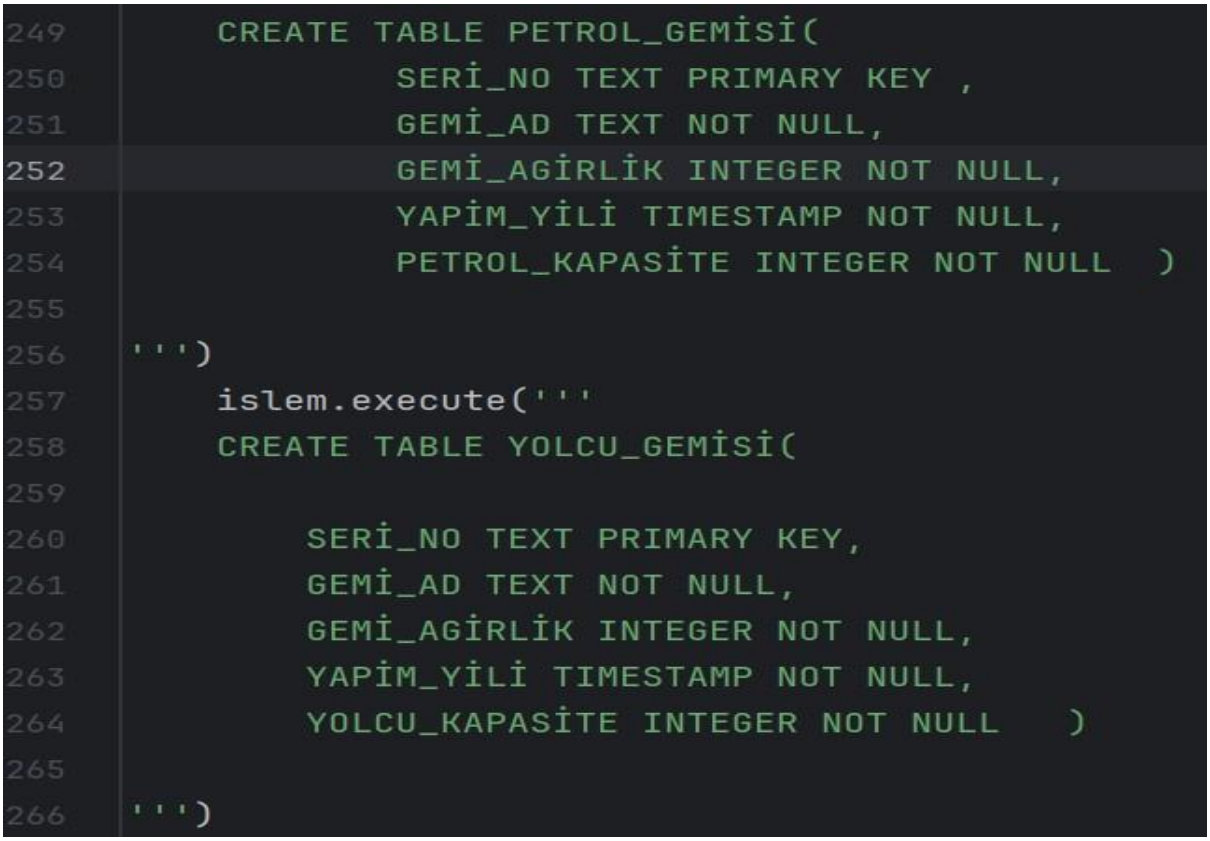
- Kod, **os.path.exists("veritabani41.db")** ifadesi ile "veritabani41.db" adlı bir dosyanın mevcut olup olmadığını kontrol eder.

- Eğer veritabanı dosyası yoksa, aşağıdaki adımları takip eder.

2. **Yeni Bir Veritabanı Dosyası Oluşturma** • Eğer veritabanı dosyası bulunmazsa, **sqlite3.connect("veritabani41.db")** ifadesi ile yeni bir SQLite veritabanı dosyası oluşturulur.
   - **cursor()** metodu ile SQL sorguları yürütmek için bir **cursor** nesnesi (**islem**) oluşturulur.

3. **"INDEX_TABLO" Tablosunu Oluşturma**
   - **islem.execute('" ... "')** ile SQL **CREATE TABLE** sorgusu çalıştırılır.
   - Bu sorgu, "INDEX_TABLO" adlı bir tablo oluşturur ve aşağıdaki sütunlara sahip olmasını sağlar: ○ **P, K, Y, KA, SF**: Her biri **INTEGER** türünde ve **NOT NULL** (boş olamaz).

Bu tablo, uygulamanın diğer bileşenleri için bir çeşit sayaç veya indeks tablosu olarak kullanılır. Örneğin, belirli bir sayacın tutulması veya farklı kategoriler için değerlerin saklanması için kullanılabilir.



```
    CREATE TABLE PETROL_GEMİSİ(
            SERİ_NO TEXT PRIMARY KEY ,
            GEMİ_AD TEXT NOT NULL,
            GEMİ_AGİRLİK INTEGER NOT NULL,
            YAPİM_YİLİ TIMESTAMP NOT NULL,
            PETROL_KAPASİTE INTEGER NOT NULL  )

''')
    islem.execute('''
    CREATE TABLE YOLCU_GEMİSİ(

        SERİ_NO TEXT PRIMARY KEY,
        GEMİ_AD TEXT NOT NULL,
        GEMİ_AGİRLİK INTEGER NOT NULL,
        YAPİM_YİLİ TIMESTAMP NOT NULL,
        YOLCU_KAPASİTE INTEGER NOT NULL   )

''')
```

Bu SQL sorguları, SQLite veritabanında "PETROL_GEMİSİ" ve "YOLCU_GEMİSİ" adında iki tablo oluşturur. Her iki tablo da benzersiz bir seri numarasını birincil anahtar olarak kullanır ve gemi ile ilgili farklı bilgileri tutar. Bu tablolar, ilgili gemi türleri için veritabanında kayıtları saklamak ve yönetmek için kullanılır.

```
338  class Ui_window(object):
339
340      def inputkontrol(self):
341          if self.sefer_ekle_mode.isChecked():
342              self.konteyner_gemi_id_2.setEnabled(False)
343          else:
344              self.konteyner_gemi_id_2.setEnabled(True)
345          if self.konteyner_ekle_mode.isChecked():
346              self.konteyner_gemi_id.setEnabled(False)
347          else:
348              self.konteyner_gemi_id.setEnabled(True)
349          if self.petrol_guncelle_ekle.isChecked():
350              self.petrol_gemi_id.setEnabled(False)
351          else:
352              self.petrol_gemi_id.setEnabled(True)
353          if self.yolcu_ekle_mode.isChecked():
354              self.yolcu_gemi_id.setEnabled(False)
355          else:
356              self.yolcu_gemi_id.setEnabled(True)
357          if self.kaptan_ekle_mode.isChecked():
358              self.mkaptan_id.setEnabled(False)
359          else:
360              self.mkaptan_id.setEnabled(True)
361
```

- **Radyo Düğmeleri Kontrolü**:
  - Radyo düğmeleri, bir grup seçenekten birini seçmek için kullanılır. ○ Bu kodda, birden fazla radyo düğmesi ("mode") ve giriş alanları bulunmaktadır.
- **Koşullar**:
  - Her **if-else** bloğu, belirli bir radyo düğmesinin seçili olup olmadığını kontrol eder (**isChecked()** yöntemi ile).
  - Eğer bir radyo düğmesi seçilmişse (**isChecked()** True), ilgili giriş alanı devre dışı bırakılır (**setEnabled(False)**). Aksi takdirde, etkinleştirilir (**setEnabled(True)**).
  - Bu mekanizma, radyo düğmesinin seçimine bağlı olarak kullanıcı girişini sınırlandırmak veya izin vermek için kullanılır.
- **Değişkenler**:
  - **self.konteyner_gemi_id_2**, **self.konteyner_gemi_id**, **self.petrol_gemi_id**, **self.yolcu_gemi_id**, **self.mkaptan_id**: Bu bileşenler, radyo düğmelerinin durumuna bağlı olarak etkinleştirilen veya devre dışı bırakılan giriş alanlarını temsil eder.

- **Radyo Düğmeleri**: Örneğin, **self.sefer_ekle_mode**, **self.konteyner_ekle_mode**, **self.petrol_guncelle_ekle**, **self.yolcu_ekle_mode**, **self.kaptan_ekle_mode**.

```
406     def veritabani_iletisim_sil(self,no):
407         if no==1:
408             SEFER.veri_sil(self.silinecek_sefer_id.text())
409         elif no==2:
410             KONTEYNER.veri_sil(self.silinecek_konteyner_id.text())
411         elif no==3:
412             PETROL.veri_sil(self.silinecek_petrol_id.text())
413         elif no==4:
414             YOLCU.veri_sil(self.silinecek_yolcu_id.text())
415         elif no==5:
416             KAPTAN.veri_sil(self.silenecek_kaptan_id.text())
417         elif no==6:
418             liman_silincek_veri=self.silinecek_liman_id.text().split(",")
419             LİMAN.veri_sil(liman_silincek_veri[0],liman_silincek_veri[1])
420         self.tablo_gosterici()
```

- **Parametreler**:
  - **no**: Bir sayı olup, hangi kategorideki kaydın silineceğini belirtir.
- **Koşullar ve Eylemler**:
  - **if-elif** blokları, **no** parametresinin değerine göre hangi kayıtların silineceğini belirler:
    - ✦ **no == 1**: "SEFER" tablosundan bir sefer kaydı silinir. Silinecek seferin ID'si **self.silinecek_sefer_id.text()** ile alınır.
    - ✦ **no == 2**: "KONTEYNER" tablosundan bir konteyner gemisi silinir. ID, **self.silinecek_konteyner_id.text()** ile alınır.
    - ✦ **no == 3**: "PETROL" tablosundan bir petrol gemisi silinir. ID, **self.silinecek_petrol_id.text()** ile alınır.
    - ✦ **no == 4**: "YOLCU" tablosundan bir yolcu gemisi silinir. ID, **self.silinecek_yolcu_id.text()** ile alınır.
    - ✦ **no == 5**: "KAPTAN" tablosundan bir kaptan silinir.

ID, **self.silenecek_kaptan_id.text()** ile alınır.

✦ **no == 6**: "LİMAN" tablosundan bir liman kaydı silinir. Burada silinecek ID, virgülle ayrılmış bir değer olarak alınır, ardından **.split(",")** ile ayrılır.

- **Tablo Güncelleme**:
  - Kayıt silme işlemi tamamlandıktan sonra, tabloyu güncellemek için **self.tablo_gosterici()** metodu çağrılır. Bu, kullanıcı arayüzündeki tabloyu güncelleyerek değişiklikleri yansıtır.

```
486     def setupUi(self, window):
487         window.setObjectName("window")
488         window.setEnabled(True)
489         window.resize(1607, 1070)
490         font = QtGui.QFont()
491         font.setFamily("Myanmar Text")
492         font.setPointSize(9)
493         font.setBold(False)
494         font.setItalic(False)
495         font.setUnderline(False)
496         font.setWeight(50)
497         font.setKerning(True)
498         font.setStyleStrategy(QtGui.QFont.PreferDefault)
499         window.setFont(font)
500         window.setTabletTracking(False)
```

- **Pencere Tanımı ve Özellikleri**: ○ **setObjectName("window")**: Pencere için benzersiz bir nesne adı ayarlar. Bu, PyQt5'in içinde belirli bileşenlere referans vermek için kullanılır.
  - **setEnabled(True)**: Pencerenin etkin olduğunu belirtir. ○ **resize(1607, 1070)**: Pencerenin boyutlarını (genişlik ve yükseklik) piksel cinsinden ayarlar.
- **Font Ayarları**:
  - **QtGui.QFont()**: PyQt5'te bir yazı tipi (font) oluşturur. ○ **setFamily("Myanmar Text")**: Yazı tipi ailesini

○

"Myanmar Text" olarak ayarlar. ○ **setPointSize(9)**: Yazı tipinin nokta boyutunu 9 olarak ayarlar.

**setBold(False), setItalic(False), setUnderline(False)**: Yazı tipinin kalın, italik veya altı çizili olmadığını belirtir. ○ **setWeight(50)**: Yazı tipinin ağırlığını ayarlar. 50, normal ağırlık anlamına gelir. ○ **setKerning(True)**: Yazı tipinin harf aralığını (kerning) etkinleştirir.

○ **setStyleStrategy(QtGui.QFont.PreferDefault)**: Yazı tipi stili için öncelikli stratejiyi belirtir. ○ **window.setFont(font)**: Yukarıda yapılandırılan fontu pencereye uygular. ○ **setTabletTracking(False)**: Tablet cihazlarında izleme özelliğini devre dışı bırakır

```
964     def retranslateUi(self, window):
965         _translate = QtCore.QCoreApplication.translate
966         window.setWindowTitle(_translate("window", "VERİTABANI UYGULAMASI"))
967         self.label.setText(_translate("window", "SEFER   BİLGİ  "))
968         self.yol_cikis_tarih.setText(_translate("window", "YOLA ÇIKIŞ TARİHİ"))
969         self.donus_tarih.setText(_translate("window", "DÖNÜŞ TARİHİ"))
970         self.gidilen_liman.setText(_translate("window", "GİDİLEN LİMAN"))
971         self.gidilen_ulke.setText(_translate("window", "GİDİLEN ÜLKE"))
972         self.kaptan_id.setText(_translate("window", "KAPTAN İD"))
973         self.gemi_seri_no.setText(_translate("window", "GEMİ SERİ NO"))
974         self.sefer_kayit_buton.setText(_translate("window", "EKLE/GÜNCELLE"))
975         self.label_2.setText(_translate("window", "  KAYIT SİLME"))
976         self.silinecek_sefer_id.setText(_translate("window", "SİLİNECEK SEFER İD"))
977         self.sefer_kayit_sil_buton.setText(_translate("window", "SEFER SİL"))
978         item = self.sefer_tablo.horizontalHeaderItem(0)
979         item.setText(_translate("window", "SEFER_İD"))
980         item = self.sefer_tablo.horizontalHeaderItem(1)
981         item.setText(_translate("window", "YOLA_CİKİS_TARİH"))
982         item = self.sefer_tablo.horizontalHeaderItem(2)
983         item.setText(_translate("window", "DONUS_TARİH"))
```

- **QtCore.QCoreApplication.translate**: Bu fonksiyon, uygulamanın yerelleştirilmiş metinlerini çevirmek için kullanılır. Dil değişikliği veya güncelleme işlemlerinde esneklik sağlar.
- **Pencere Başlığı**:
  - **window.setWindowTitle(...)** ile pencere başlığı belirlenir veya güncellenir.

- **Etiketler ve Diğer Metinler**:
  - **self.label.setText(...)**: Belirli bir etiketin metnini ayarlar.

- 

  **self.yol_cikis_tarih.setText(...)**: Metin kutusundaki varsayılan metni ayarlar. ○ **self.donus_tarih, self.gidilen_liman, self.gidilen_ulke, self.kaptan_id, self.gemi_seri_no**: Bu gibi öğelerin metinlerini günceller.
  - **Düğme Metinleri**: **self.sefer_kayit_buton**, **self.sefer_kayit_sil_buton** gibi öğelerin metinlerini ayarlar.
  - **Tablo Başlıkları**:
    - **item.setText(...)**: Tablonun sütun başlıklarını ayarlar.
  - Burada, bir sefer tablosu ve ilgili öğelerin başlıkları güncellenmektedir.

```
1014        item = self.petrol_tablo.horizontalHeaderItem(0)
1015        item.setText(_translate("window", "SERİ NO"))
1016        item = self.petrol_tablo.horizontalHeaderItem(1)
1017        item.setText(_translate("window", "GEMİ AD"))
1018        item = self.petrol_tablo.horizontalHeaderItem(2)
1019        item.setText(_translate("window", "GEMİ AGİRLİK"))
1020        item = self.petrol_tablo.horizontalHeaderItem(3)
1021        item.setText(_translate("window", "YAPİM YİLİ"))
1022        item = self.petrol_tablo.horizontalHeaderItem(4)
1023        item.setText(_translate("window", "PETROL KAPASİTE"))
1024        self.label_5.setText(_translate("window", "PETROL GEMİ"))
1025        self.petrol_gemi_ad.setText(_translate("window", "GEMİ AD"))
```

- **QtCore.QCoreApplication.translate**: Bu fonksiyon, uygulamanın yerelleştirilmiş metinlerini çevirmek için kullanılır. Dil değişikliği veya güncelleme işlemlerinde esneklik sağlar.
- **Pencere Başlığı**:
  - **window.setWindowTitle(...)** ile pencere başlığı belirlenir veya güncellenir.
- **Etiketler ve Diğer Metinler**:
  - **self.label.setText(...)**: Belirli bir etiketin metnini ayarlar. ○ **self.yol_cikis_tarih.setText(...)**: Metin kutusundaki varsayılan metni ayarlar.

**self.donus_tarih, self.gidilen_liman, self.gidilen_ulke, self.kaptan_id, self.gemi_seri_no**: Bu gibi öğelerin metinlerini günceller.

- **Düğme Metinleri**: **self.sefer_kayit_buton**, **self.sefer_kayit_sil_buton** gibi öğelerin metinlerini ayarlar.
- **Tablo Başlıkları**:
  - **item.setText(...)**: Tablonun sütun başlıklarını ayarlar.
- Burada, bir sefer tablosu ve ilgili öğelerin başlıkları güncellenmektedir.

Bu kod parçası, PyQt5 GUI'sindeki çeşitli metinleri ayarlamak ve güncellemek için kullanılır. Kullanıcı arayüzü öğelerinin başlıklarını, etiketlerini, düğmelerini ve tablo başlıklarını yeniden tanımlamak veya yeniden çevirmek için kullanılabilir. Bu, uygulamanın dinamik yapıda çalışmasını ve gerektiğinde metin değişikliklerini desteklemesini sağlar.

```
1120  if __name__ == "__main__":
1121      import sys
1122      app = QtWidgets.QApplication(sys.argv)
1123      window = QtWidgets.QMainWindow()
1124      ui = Ui_window()
1125      ui.setupUi(window)
1126      window.show()
1127      sys.exit(app.exec_())
```

Bu kod parçası, bir PyQt5 uygulamasını başlatır ve kullanıcıya ana pencereyi gösterir. PyQt5 GUI uygulamalarında, bu kod bloğu genellikle ana başlatma noktasıdır ve uygulama yaşam döngüsünü kontrol eder. Uygulama penceresini oluşturur, yapılandırır ve gösterir;

- 

ardından kullanıcı etkileşimlerini işlemek için uygulama döngüsünü başlatır.

## 4.2 Görev Dağılımı

Görev dağılımı ve geliştirme aşamaları:

- Yazılım tasarımı ve geliştirme: Ahmet Can BOSTANCI, Dila Seray TEGÜN
- Kodlama ve test: Ahmet Can BOSTANCI, Dila Seray TEGÜN
- Rapor hazırlama ve belgeler: Dila Seray TEGÜN. Ahmet Can BOSTANCI

## 4.3 Karşılaşılan Zorluklar ve Çözüm Yöntemleri

Geliştirme sürecinde karşılaşılan problemler ve çözümler: •

**Zorluk**:Algoritma oluşturma zorluğu.

- **Çözüm**:Daha efektif classlar yazarak sorunun çözülmesi.

- **Zorluk**: Arayüz bileşenlerinde uyumsuzluk.
  - **Çözüm**: PyQt5 kütüphane güncellemeleri ve bileşenlerin yeniden yerleşimi.

# 4.4 Proje İsterlerine Göre Eksik Yönler

Eksik kalan veya tamamlanamayan özellikler:

Proje isterine göre bir gemide birden fazla kaptan bulunabilirken. Eksik yön olarak bu projede bir gemide sadece bir kaptan olucak şekilde ayarlanmıştır.

Proje isterine göre bir mürettebat tablosu veri tabanında tutulması gerekirken. Bu projede mürettebat tablosu yer almamaktadır.

# 5. Test ve Doğrulama

## 5.1 Yazılımın Test Süreci

Yazılım için test uygulaması:

- Test bileşenleri: [Test edilen bileşenler belirtilir]
- Test senaryoları: [Belirli test senaryoları belirtilir]

## TEST BİLEŞENLERİ

```
testobjesi1=PETROL_GEMİSİ("PETROL_GEMİSİ")

testobjesi2=YOLCU_GEMİSİ("YOLCU_GEMİSİ")

testobjesi3=KONTEYNER_GEMİSİ("KONTEYNER_GEMİSİ")

testobjesi4=KAPTAN_M("KAPTAN_M")

testobjesi5=SEFER_KAYİT("SEFER_KAYİT")

testobjesi6=LİMAN_BİLGİ("LİMAN_BİLGİ")
```

## TEST SENARYOLARI

```
testobjesi1=PETROL_GEMİSİ("PETROL_GEMİSİ")
testobjesi1.veri_ekle("petrol gemisi3",54,2003,1200) # Petrol tablosuna veri ekliyoruz .
# GEMİ ADI, AĞIRLIĞI, YAPIM YILI , PETROL KAPASİTESİ GİRİLİR. GEMİ İD OTOMATİK OLARAK ATANIR .

testobjesi1.veri_güncelle(" gemi id","guncellenicek gemi adı","gullenicek gemi agırlıgı",
                          "guncellenicek gemi yapim yili", "guncellenicek gemi petrol kapasite")
# gemi ID leri tablolarda gösterilir oradan gemi ıd sine bakıp diğer güncellenecek
# değerleri girince değerler güncellencektir.

testobjesi1.veri_sil("silinecek gemi id") # silinecek gemi ıd si girilir ve gemi silinir

print(testobjesi1.veri_listele()) # tablodaki satırları liste içinde tuple olarak listeler  örnek:
# [('P1', 'petrol gemisi', 54, 2003, 1200), ('P2', 'petrol gemisi1', 54, 2003, 1200)]
```

```
testobjesi2=YOLCU_GEMİSİ("YOLCU_GEMİSİ")
testobjesi2.veri_ekle("yolcu gemisi 1",54,2003,1200) # Yolcu tablosuna veri ekliyoruz .
# GEMİ ADI, AĞIRLIĞI, YAPIM YILI , YOLCU KAPASİTESİ GİRİLİR. GEMİ İD OTOMATİK OLARAK ATANIR .

testobjesi2.veri_güncelle(" gemi id","guncellenicek gemi adı","gullenicek gemi agırlıgı",
                          "guncellenicek gemi yapim yili", "guncellenicek gemi yolcu kapasite")
# gemi ID leri tablolarda gösterilir oradan gemi ıd sine bakıp diğer güncellenecek
# değerleri girince değerler güncellencektir.

testobjesi2.veri_sil("silinecek gemi id") # silinecek gemi ıd si girilir ve gemi silinir

print(testobjesi2.veri_listele()) # tablodaki satırları liste içinde tuple olarak listeler  örnek:
# [('Y1', 'yolcu gemisi', 54, 2003, 1200), ('Y2', 'yolcu gemisi1', 54, 2003, 1200)]
```

```
testobjesi3=KONTEYNER_GEMİSİ("KONTEYNER_GEMİSİ")
testobjesi3.veri_ekle("konteyner gemisi 1",54,2003,1200,4364) # konteyner tablosuna veri ekliyoruz .
# GEMİ ADI, AĞIRLIĞI, YAPIM YILI , KONTEYNER SAYISI MAX, MAX AGIRLIK GİRİLİR. GEMİ İD OTOMATİK OLARAK ATANIR .

testobjesi3.veri_güncelle(" gemi id","guncellenicek gemi adı","gullenicek gemi agırlıgı",
                          "guncellenicek gemi yapim yili", "guncellenicek konteyner sayısı max","guncellenicek max agırlık")
# gemi ID leri tablolarda gösterilir oradan gemi ıd sine bakıp diğer güncellenecek
# değerleri girince değerler güncellencektir.

testobjesi3.veri_sil("silinecek gemi id") # silinecek gemi ıd si girilir ve gemi silinir

print(testobjesi3.veri_listele()) # tablodaki satırları liste içinde tuple olarak listeler  örnek:
# [('K1', 'konteyner gemisi', 54, 2003, 1200,1245), ('K2', 'konteyner gemisi2', 54, 2003, 1200,2356)]
```

```
testobjesi4=LİMAN_BİLGİ("KAPTAN_M")
testobjesi4.veri_ekle("kaptan ad","kaptan soyad","kaptan adres",1999,2006,"kaptan gorev","kaptan lisans") # kaptan tablosuna veri ekliyoruz .
# KAPTAN AD ,KAPTAN SOYAD, KAPTAN ADRES, KAPTAN DOGUM TARİN, KAPTAN İSE GİRİS TARİH,KAPTAN GÖREV, KAPTAN LİSANS. KAPTAN İD OTOMATİK OLARAK ATANIR .

testobjesi4.veri_güncelle(" kaptan id","guncellenicek kaptan adı","guncellenicek kaptan soyadı",
                          "guncellenicek kaptan adres", "guncellenicek dogum tarih","guncellenicek ise giris tarih",
                          "guncellenicek kaptan gorev","guncellenicek kaptan lisans")
# kaptan ID leri tablolarda gösterilir oradan kaptan ıd sine bakıp diğer güncellenecek
# değerleri girince değerler güncellencektir.

testobjesi4.veri_sil("silinecek kaptan id") # silinecek kaptan ıd si girilir ve kaptan silinir

print(testobjesi4.veri_listele()) # tablodaki satırları liste içinde tuple olarak listeler  örnek:
# [('KA1', 'AHMETCAN',"BOSTANCI","KÖSEKÖY", 2003, 2024, "KOSTU OGRENCİ","LISANS"), ('KA2', 'DİLA SERAY',"TEGÜN","KÖSEKÖY", 2003, 2024, "KOSTU OGRENCİ","LISANS")]
```

```
testobjesi6=LİMAN_BİLGİ("LİMAN_BİLGİ")
testobjesi6.veri_ekle("LIMAN AD","ULKE AD",44667,"evet",200633) # LIMAN tablosuna veri ekliyoruz .
# LIMAN AD ,ULKE AD, NUFUS, PASAPORT VARMI YOK MU, DEMİRLEME UCRET DEĞERLERİ GİRİLİR .

testobjesi6.veri_güncelle(" LIMAN AD","ULKE AD","guncellenicek nufus",
                "guncellenicek pasaport varlığı", "guncellenicek demirleme ucretı")


testobjesi6.veri_sil("silinecek LIMAN AD,ULKE AD") # LIMAN AD, ULKE AD  FORMATINDA GİRİLİR ARADA VİRGÜL VAR . VE TABLODAN SİLİNİR

print(testobjesi6.veri_listele()) # tablodaki satırları liste içinde tuple olarak listeler  örnek:
# [('AVRASYA', 'ZURİH',2345, "YES", 20242), ('AVRASYA', 'MUNİH',2345, "YES", 20242)]
```

```
testobjesi5=SEFER_KAYİT("SEFER_KAYİT")
testobjesi5.veri_ekle("YOLA CIKIS TARİH","DONUS TARİH","GİDİLEN LİMAN","GİDİLEN ULKE",234,245) # SEFER tablosuna veri ekliyoruz .
# "YOLA CIKIS TARİH","DONUS TARİH","GİDİLEN LİMAN","GİDİLEN ULKE",KAPTAN İD,GEMİ SERİ NO DEĞERLERİ GİRİLİR .

testobjesi5.veri_güncelle(" SEFER İD","guncellenicek yol cıkıs  tarihi","guncellenicek donus tarihi",
                "guncellenicek gidilen liman", "guncellenicek gidilen ulke","guncellenicek kaptan id","guncellenicek gemi seri no")


testobjesi5.veri_sil("silinecek SEFER İD") # SEFER İD  FORMATINDA GİRİLİR ARADA VİRGÜL VAR . VE TABLODAN SİLİNİR

print(testobjesi5.veri_listele()) # tablodaki satırları liste içinde tuple olarak listeler  örnek:
# [('SF1', '11-11-2011',"10-10-2012", "ZURİH", "AVRASYA",12,32),('SF2', '12-11-2031',"10-09-2021", "MUNİH", "AVRASYA",122,332)]
```