



T.C
KOCAELİ SAęLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOęA BİLİMLERİ FAKÜLTESİ
YAZILIM MÜHENDİSLİęİ

PROJE KONUSU:
VERİ TABANI YÖNETİM SİSTEMLERİ PROJESİ

ÖęRENCİ ADI:
ZEHRA YARDIMCI

ÖęRENCİ NUMARASI:
220502038

DERS SORUMLUSU:
PROF. DR. NEVCİHAN DURU

TARİH: 04.05.2024

1 GİRİŞ

1.1 Projenin amacı

- Projenin amacı, Gezgin Gemi Şirketi'nin seferlerini etkin bir şekilde yönetebilmek için bir yazılım geliştirmektir.

1.2 Projede Gerçekleştirilmesi Beklenenler:

- Gemilerin verilerinin tutulması ve yönetilmesi:
 - Seri numarası, adı, ağırlığı, yapım yılı gibi bilgilerin saklanması.
 - Yolcu gemileri için yolcu kapasitesi, petrol tankerleri için petrol kapasitesi, konteyner gemileri için konteyner sayısı kapasitesi gibi özel bilgilerin kaydedilmesi.
- Seferlerin yönetilmesi:
 - Her sefer için gerekli olan bilgilerin saklanması: ID, yola çıkış tarihi, dönüş tarihi, yola çıkış limanı vb.
 - Bir seferde birden fazla limanda demirleme imkanının sağlanması.
 - Geçmiş, gelecek ve olası seferlerin bilgilerinin tutulması.
- Limanların verilerinin yönetilmesi:
 - Her liman için gerekli olan bilgilerin saklanması: liman adı, ülke, nüfus, pasaport gerekliliği, demirleme ücreti.
 - Henüz uğranmamış ancak potansiyel olarak uğranabilecek limanların kaydedilmesi.
- Kaptan ve mürettebat bilgilerinin yönetilmesi:
 - Kaptan ve mürettebat için gerekli kişisel bilgilerin saklanması: ID, ad, soyad, adres, vatandaşlık, doğum tarihi, işe giriş tarihi.
 - Kaptanların lisans bilgilerinin, mürettebatın çalıştığı görevin tutulması.
- Veri tabanı ve form ekranı arasındaki yönetimin nesneler aracılığı ile gerçekleştirilmesi.
- Sınıfların oluşturulması ve verilerin doğru bir şekilde tutulmasının sağlanması.

Veri tabanı işlemlerinin Python kullanılarak gerçekleştirilmesi.

2 GEREKSİNİM ANALİZİ

2.1 Arayüz gereksinimleri

Giriş Ekranı:

- Kullanıcıların sisteme giriş yapabilmeleri için kullanıcı adı ve şifre alanları olmalıdır.

- Giriş yap butonuyla sisteme erişim sağlanmalıdır.
Ana Menü:
- Kullanıcıların gemiler, seferler, kaptanlar, mürettebat ve limanlar gibi temel veri tabanı öğelerine erişebilecekleri bir ana menü olmalıdır.
- Ana menüde ilgili modüllere yönlendiren butonlar veya bağlantılar bulunmalıdır.
Gemiler Modülü:
- Gemilerin listelendiği, eklediği, düzenlediği ve sileceği bir arayüz olmalıdır.
- Her gemi için detaylı bilgilerin görüntülenebileceği bir detay ekranı bulunmalıdır.
Seferler Modülü:
- Seferlerin listelendiği, eklediği, düzenlediği ve sileceği bir arayüz olmalıdır.
- Her sefer için detaylı bilgilerin görüntülenebileceği bir detay ekranı bulunmalıdır.
Kaptanlar Modülü:
- Kaptanların listelendiği, eklediği, düzenlediği ve sileceği bir arayüz olmalıdır.
- Her kaptan için detaylı bilgilerin görüntülenebileceği bir detay ekranı bulunmalıdır.
Mürettebat Modülü:
- Mürettebatın listelendiği, eklediği, düzenlediği ve sileceği bir arayüz olmalıdır.
- Her mürettebat üyesi için detaylı bilgilerin görüntülenebileceği bir detay ekranı bulunmalıdır.
Limanlar Modülü:
- Limanların listelendiği, eklediği, düzenlediği ve sileceği bir arayüz olmalıdır.
- Her liman için detaylı bilgilerin görüntülenebileceği bir detay ekranı bulunmalıdır.

2.2 Donanım gereksinimleri

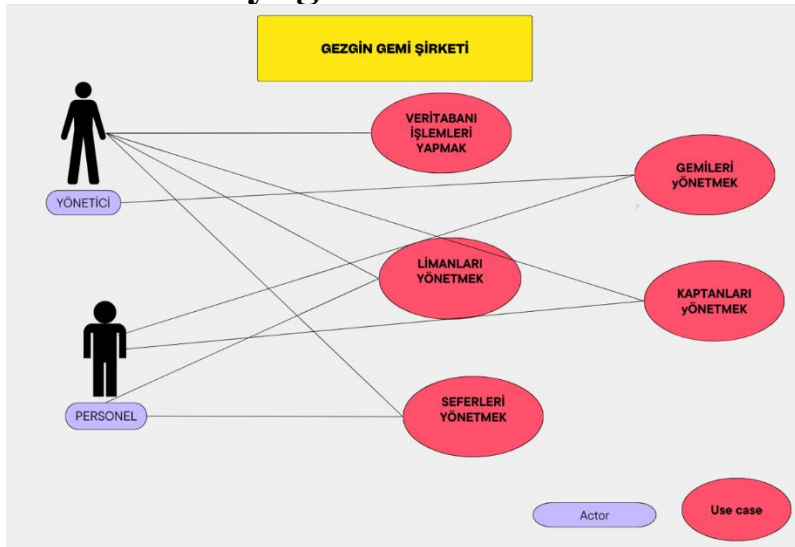
- Yazılımın çalışması için standart bir bilgisayar donanımı yeterlidir.

2.3 Fonksiyonel gereksinimler

- **Giriş ve Oturum İle İlgili İşlemler:**
 - Kullanıcıların sisteme giriş yapabilmesi ve oturum açabilmesi gerekmektedir.
 - Oturumların güvenli bir şekilde yönetilmesi ve oturum süresinin belirlenmesi gerekmektedir.

- **Gemiler İle İlgili İşlemler:**
 - Yeni gemi ekleyebilme, var olan gemiyi güncelleme, gemi bilgilerini görüntüleme ve gemi silme işlemleri yapılabilir olmalıdır.
 - Her gemi için özel özelliklere göre ayrı ayrı işlemler yapılabilir olmalıdır: yolcu kapasitesi, petrol kapasitesi, konteyner sayısı gibi.
- **Sefer İşlemleri:**
 - Yeni sefer oluşturma, var olan seferi güncelleme, sefer detaylarını görüntüleme ve seferi silme işlemleri yapılabilir olmalıdır.
 - Her sefer için liman bilgileri ve gemi bilgileri ilişkilendirilebilir olmalıdır.
- **Kaptan ve Mürettebat Yönetimi:**
 - Yeni kaptan ve mürettebat ekleyebilme, var olanları güncelleme, detaylarını görüntüleme ve silme işlemleri yapılabilir olmalıdır.
 - Kaptanların lisans bilgileri ve mürettebatın görev bilgileri tutulmalı ve yönetilebilmelidir.
- **Liman İşlemleri:**
 - Yeni liman ekleme, var olan limanı güncelleme, liman detaylarını görüntüleme ve limanı silme işlemleri yapılabilir olmalıdır.
 - Her liman için ülke, nüfus, pasaport gerekliliği ve demirleme ücreti gibi özellikler yönetilebilir olmalıdır.
- **Veri Tabanı Yönetimi:**
 - Veri tabanında kayıtlı olan verilerin güvenli bir şekilde saklanması ve erişilmesi sağlanmalıdır.
 - Veri tabanındaki verilerin düzenlenmesi, ekleme, silme ve güncelleme gibi işlemler fonksiyonel olarak sağlanmalıdır.

2.4 Use-Case diyagramı

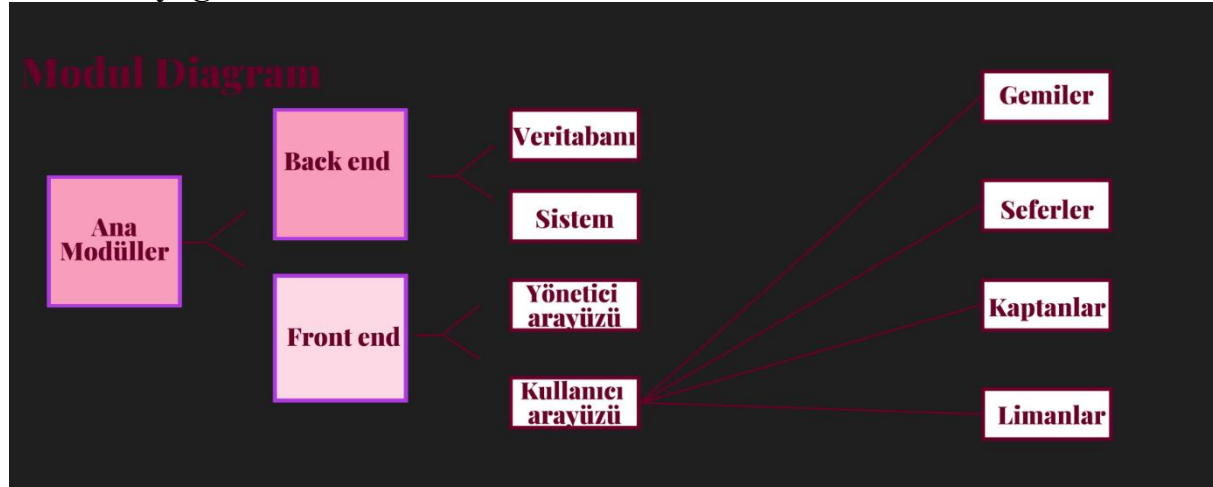


3 TASARIM

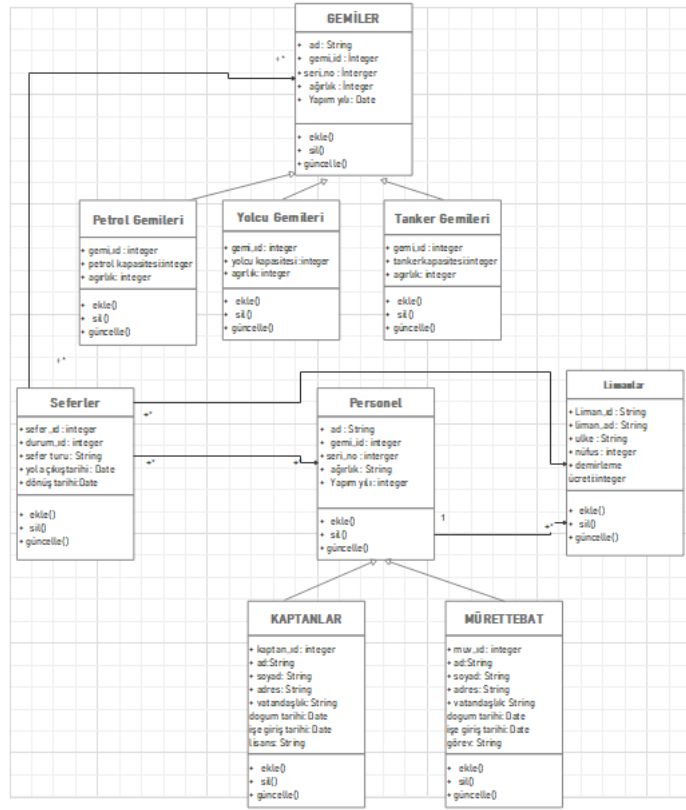
3.1 Mimari tasarım

- **Veri Tabanı Yapısı:**

- Veri tabanı, ilişkisel bir yapıda olmalıdır.
 - Gemiler, seferler, kaptanlar, mürettebat ve limanlar gibi temel varlıklar için ayrı tablolar oluşturulmalıdır.
 - İlişkisel veri tabanı yönetim sistemi olarak SQL Server kullanılacaktır.
- **Nesne Tabanlı Programlama:**
- Python kullanılarak nesne tabanlı programlama yapılmıştır.
 - Her varlık için ayrı bir sınıf oluşturulmuştur: Gemiler, Seferler, Kaptanlar, Mürettebat, Limanlar gibi.
- **Kullanıcı Arayüzü Tasarımı:**
- Kullanıcı dostu ve erişilebilir bir arayüz tasarlanmalıdır.
 - Arayüz, kullanıcıların veri girişi yapabilecekleri, verileri görüntüleyebilecekleri ve düzenleyebilecekleri form ekranları içermelidir.
- **Güvenlik ve Yetkilendirme:**
- Kullanıcıların güvenli bir şekilde oturum açmaları ve erişim yetkilerinin yönetilmesi sağlanmalıdır.
 - Kullanıcıların yalnızca yetkilendirildikleri verilere erişebilmeleri ve yetkilendirilmedikleri verilere erişimleri engellenmelidir.
 - **Modül diyagramı:**



Class Diagramı:



3.2 Kullanılacak teknolojiler

- **Yazılım Dili:**
 - Yazılım Python programlama dili kullanılarak geliştirilecektir.
- **Harici Kütüphaneler:**
 - **Tkinter:** Python'da masaüstü uygulamaları için kullanılan standart bir arayüz kütüphanesidir. Tkinter, kullanıcı arayüzü oluşturmak için kullanılacaktır.
 - **pyodbc:** Python'da veri tabanlarına bağlanmak için kullanılan bir kütüphanedir. pyodbc, SQL Server veya diğer veri tabanlarına erişim sağlamak için kullanılacaktır.
- **Diğer Teknolojiler:**
 - Veri tabanı yönetimi için **SQL Server** kullanılacaktır. Python'da SQL Server ile etkileşim sağlamak için **pyodbc** kullanılacaktır.

3.3 Veri tabanı tasarımı

- Veri tabanı tasarımı, Gezin Gemi Şirketi'nin seferlerini yönetebilmesi için gereken verilerin yapılandırılmasını ve ilişkilendirilmesini içerir.
- **Gemiler tablosu**, şirketin farklı türdeki gemilerinin özelliklerini içerir. Her gemi için bir dizi temel bilgi saklanır, bu bilgiler gemi adı, seri numarası, ağırlık, yapım yılı ve türü gibi detayları içerir.

Seferler tablosu, şirketin geçmiş, gelecek ve olası seferlerinin detaylarını içerir. Her sefer için başlangıç ve bitiş tarihleri, sefer türü, seferin gerçekleştiği gemilerin bilgileri ve seferin uğradığı limanlar gibi bilgiler bu tabloda saklanır. Ayrıca, seferlerin durumunu belirten bir durum kodu da bulunur.


Limanlar tablosu, seferlerin uğrayabileceği limanların detaylarını içerir. Her liman için liman adı, ülke, nüfus, pasaport gerekliliği ve demirleme ücreti gibi bilgiler saklanır. Ayrıca, potansiyel olarak uğranabilecek limanlar da ayrı bir tabloda saklanır.

Kaptanlar ve Mürettebat tabloları, gemilerde görev yapacak personelin detaylarını içerir. Her personel için ad, soyad, adres, vatandaşlık, doğum tarihi, işe giriş tarihi ve görev gibi bilgiler bu tablolarda saklanır. Ayrıca, kaptanlar için lisans bilgisi de kaydedilir.



- Kullanıcı arayüzü tasarımı, Gezgin Gemi Şirketi personelinin gemileri, seferleri, limanları, kaptanları ve mürettebatıyla ilgili verilere erişimini ve bu verileri yönetimini kolaylaştırmak için yapılandırılmıştır. Arayüz, kullanıcıların veri girişi yapabileceği,

mevcut verileri görüntüleyebileceği, düzenleyebileceği ve yeni veriler ekleyebileceği form ve ekranı içerir.

 Gemi Bilgi Formu

Seri Numarası:	<input type="text"/>	
Adı:	<input type="text"/>	
Ağırlık:	<input type="text"/>	
Yapım Yılı:	<input type="text"/>	
Tür:	<input type="text"/>	
Maksimum Ağırlık:	<input type="text"/>	
Gemi ID:	<input type="text"/>	
<input type="button" value="Ekle"/>	<input type="button" value="Güncelle"/>	<input type="button" value="Sil"/>

- **Uygulamanın nasıl çalıştırılacağı ile ilgili açıklama:**
 - Uygulamayı başlatmak için Python doysası çalıştırılmalıdır. Bu dosyayı çalıştırdığınızda, kullanıcı arayüzü belirecektir. Arayüz üzerinden gemileri, seferleri, limanları, personeli ve diğer ilgili verileri görüntüleyebilir, düzenleyebilir ve ekleyebilirsiniz.

4 UYGULAMA

4.1 Kodlanan bileşenlerin açıklamaları

- **Sınıf Tanımı (Gemiler):**
 - **Gemiler** sınıfı, gemilerle ilgili veritabanı işlemlerini yönetir.
 - **__init__** metodu, sınıfın başlatılması sırasında çağrılır ve bir veritabanı bağlantı dizesini kabul eder.
 - **connection_string** sınıf özelliği, veritabanına bağlanmak için kullanılacak bağlantı dizesini tutar.
- **Gemi Ekleme (ekle):**
 - **ekle** metodu, veritabanına yeni bir gemi eklemek için kullanılır.
 - Parametreler: **seri_numarasi**, **adi**, **agirlik**, **yapim_yili**, **tur**, ve isteğe bağlı olarak **maks_agirlik**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, SQL komutu çalıştırılır ve bağlantı kapatılır. Herhangi bir hata durumunda **except** bloğu çalışır.
- **Gemi Silme (sil):**
 - **sil** metodu, belirli bir gemiyi veritabanından silmek için kullanılır.
 - Parametre: **gemi_id**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, gemi silme

SQL komutu çalıştırılır ve bağlantı kapatılır.

- **Gemi Güncelleme (güncelle):**
 - **güncelle** metodu, belirli bir geminin bilgilerini güncellemek için kullanılır.
 - Parametreler: **gemi_id** ve güncellenecek diğer alanlar (****kwargs** ile esneklik sağlanır).
 - **try-except** bloğu içinde, güncelleme SQL komutu çalıştırılır.
- **Belirli Bir Gemi Getirme (getir):**
 - **getir** metodu, belirli bir gemiyi veritabanından getirmek için kullanılır.
 - Parametre: **gemi_id**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, sorgu çalıştırılır ve sonuç alınır.
- **Sınıf Tanımı (YolcuGemileri):**
 - **YolcuGemileri** sınıfı, yolcu gemileriyle ilgili veritabanı işlemlerini yönetir.
 - **__init__** metodu, sınıfın başlatılması sırasında çağrılır ve bir veritabanı bağlantı dizesini kabul eder.
 - **connection_string** sınıf özelliği, veritabanına bağlanmak için kullanılacak bağlantı dizesini tutar.
- **Yolcu Gemisi Ekleme (ekle):**
 - **ekle** metodu, veritabanına yeni bir yolcu gemisi eklemek için kullanılır.
 - Parametreler: **gemi_id**, **yolcu_kapasitesi**, **maks_agirlik**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, SQL komutu çalıştırılır ve bağlantı kapatılır.
- **Yolcu Gemisi Silme (sil):**
 - **sil** metodu, belirli bir yolcu gemisini veritabanından silmek için kullanılır.
 - Parametre: **gemi_id**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, gemi silme SQL komutu çalıştırılır ve bağlantı kapatılır.
- **Yolcu Gemisi Güncelleme (güncelle):**
 - **güncelle** metodu, belirli bir yolcu gemisinin bilgilerini güncellemek için kullanılır.
 - Parametreler: **gemi_id** ve güncellenecek diğer alanlar (**yolcu_kapasitesi** ve/veya **maks_agirlik**).
 - **try-except** bloğu içinde, güncelleme SQL komutu çalıştırılır.
- **Belirli Bir Yolcu Gemisi Getirme (getir):**
 - **getir** metodu, belirli bir yolcu gemisini veritabanından getirmek için kullanılır.
 - Parametre: **gemi_id**.

- **try-except** bloğu içinde, veritabanı bağlantısı kurulur, sorgu çalıştırılır ve sonuç alınır.
- Sınıf Tanımı (PetrolTankerleri):
 - **PetrolTankerleri sınıfı**, petrol tankeri gemileriyle ilgili veritabanı işlemlerini yönetir.
 - **__init__ metodu**, sınıfın başlatılması sırasında çağrılır ve bir veritabanı bağlantı dizesini kabul eder.
 - **connection_string** sınıf özelliği, veritabanına bağlanmak için kullanılacak bağlantı dizesini tutar.
- Petrol Tankeri Ekleme (ekle):
 - **ekle metodu**, veritabanına yeni bir petrol tankeri gemisi eklemek için kullanılır.
 - **Parametreler**: gemi_id, petrol_kapasitesi, maks_agirlik.
 - try-except bloğu içinde, veritabanı bağlantısı kurulur, SQL komutu çalıştırılır ve bağlantı kapatılır.
- Petrol Tankeri Silme (sil):
 - **sil metodu**, belirli bir petrol tankeri gemisini veritabanından silmek için kullanılır.
 - Parametre: gemi_id.
 - try-except bloğu içinde, veritabanı bağlantısı kurulur, gemi silme SQL komutu çalıştırılır ve bağlantı kapatılır.
- Petrol Tankeri Güncelleme (güncelle):
 - **güncelle metodu**, belirli bir petrol tankeri gemisinin bilgilerini güncellemek için kullanılır.
 - **Parametreler**: gemi_id ve güncellenecek diğer alanlar (petrol_kapasitesi ve/veya maks_agirlik).
 - try-except bloğu içinde, güncelleme SQL komutu çalıştırılır.
- Belirli Bir Petrol Tankeri Getirme (getir):
 - **getir metodu**, belirli bir petrol tankeri gemisini veritabanından getirmek için kullanılır.
 - **Parametre**: gemi_id.
 - try-except bloğu içinde, veritabanı bağlantısı kurulur, sorgu çalıştırılır ve sonuç alınır.
- Sınıf Tanımı (Seferler):
 - **Seferler** sınıfı, seferlerle ilgili veritabanı işlemlerini gerçekleştirir.
 - **__init__ metodu**, sınıfın başlatılması sırasında çağrılır ve bir veritabanı bağlantı dizesini kabul eder.
 - **connection_string** sınıf özelliği, veritabanına bağlanmak için kullanılacak bağlantı dizesini tutar.
- Sefer Ekleme (ekle):
 - **ekle metodu**, veritabanına yeni bir sefer eklemek için kullanılır.
 - Parametreler: durum_id, sefer_turu, yola_cikis_tarihi,

donus_tarihi, limanlar.

- **try-except** bloğu içinde, veritabanı bağlantısı kurulur, sefer eklenir ve ilgili liman bilgileri de eklenir.
- **Sefer Silme (sil):**
 - **sil** metodu, belirli bir seferi ve ilgili liman bilgilerini veritabanından silmek için kullanılır.
 - Parametre: **sefer_id**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, sefer ve ilgili liman bilgileri silinir.
- **Sefer Güncelleme (güncelle):**
 - **güncelle** metodu, belirli bir seferin bilgilerini güncellemek için kullanılır.
 - Parametreler: **sefer_id** ve güncellenecek diğer alanlar (****kwargs** ile esneklik sağlanır).
 - **try-except** bloğu içinde, güncelleme SQL komutu çalıştırılır.
- **Belirli Bir Seferi Getirme (getir):**
 - **getir** metodu, belirli bir seferin bilgilerini veritabanından getirmek için kullanılır.
 - Parametre: **sefer_id**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, sorgu çalıştırılır ve sonuç alınır.
- **Sınıf Tanımı (Limanlar):**
 - **Limanlar** sınıfı, limanlarla ilgili veritabanı işlemlerini yönetir.
 - **__init__** metodu, sınıfın başlatılması sırasında çağrılır ve bir veritabanı bağlantı dizesini kabul eder.
 - **connection_string** sınıf özelliği, veritabanına bağlanmak için kullanılacak bağlantı dizesini tutar.
- **Liman Ekleme (ekle):**
 - **ekle** metodu, veritabanına yeni bir liman eklemek için kullanılır.
 - Parametreler: **liman_adi, ulke, nufus, pasaport_gerekli_mi, demirleme_ucreti**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, liman eklenir ve işlem tamamlanır.
- **Liman Silme (sil):**
 - **sil** metodu, belirli bir limanı veritabanından silmek için kullanılır.
 - Parametre: **liman_id**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, liman silinir ve işlem tamamlanır.
- **Liman Güncelleme (güncelle):**
 - **güncelle** metodu, belirli bir limanın bilgilerini güncellemek için kullanılır.
 - Parametreler: **liman_id** ve güncellenecek diğer alanlar

- **try-except** bloğu içinde, güncelleme SQL komutu çalıştırılır.
- **Belirli Bir Limanı Getirme (getir):**
 - **getir** metodu, belirli bir limanın bilgilerini veritabanından getirmek için kullanılır.
 - Parametre: **liman_id**.
 - **try-except** bloğu içinde, veritabanı bağlantısı kurulur, sorgu çalıştırılır ve sonuç alınır.
- **Sınıf Tanımı (Kaptanlar):**
 - **Kaptanlar** sınıfı, kaptanlarla ilgili veritabanı işlemlerini yönetir.
 - **__init__** metodu, sınıfın başlatılması sırasında çağrılır ve **Personel** sınıfından türetilir.
 - **super().__init__(connection_string, "Kaptanlar")** ifadesiyle, üst sınıfın (**Personel**) **__init__** metoduna bağlantı dizesi ve tablo adı ("Kaptanlar") iletilir.
- **Kaptan Ekleme (ekle):**
 - **ekle** metodu, veritabanına yeni bir kaptan eklemek için kullanılır.
 - Parametreler: **ad, soyad, adres, vatandaslik, dogum_tarihi, ise_giris_tarihi, lisans**.
 - **super().ekle** ifadesiyle, üst sınıfın (**Personel**) **ekle** metoduna parametreler iletilir.
- **Kaptan Güncelleme (güncelle):**
 - **güncelle** metodu, belirli bir kaptanın lisans bilgisini güncellemek için kullanılır.
 - Parametreler: **kaptan_id, lisans**.
 - **try-except** bloğu içinde, güncelleme SQL komutu çalıştırılır. **Sınıf Tanımı (Mürettebat):**
 - **Mürettebat** sınıfı, mürettebatla ilgili veritabanı işlemlerini yönetir.
 - **__init__** metodu, sınıfın başlatılması sırasında çağrılır ve **Personel** sınıfından türetilir.
 - **super().__init__(connection_string, "Mürettebat")** ifadesiyle, üst sınıfın (**Personel**) **__init__** metoduna bağlantı dizesi ve tablo adı ("Mürettebat") iletilir.
- **Mürettebat Ekleme (ekle):**
 - **ekle** metodu, veritabanına yeni bir mürettebat eklemek için kullanılır.
 - Parametreler: **ad, soyad, adres, vatandaslik, dogum_tarihi, ise_giris_tarihi, görev**.
 - **super().ekle** ifadesiyle, üst sınıfın (**Personel**) **ekle** metoduna parametreler iletilir.
- **Mürettebat Güncelleme (güncelle):**
 - **güncelle** metodu, belirli bir mürettebatın görev bilgisini güncellemek için kullanılır.

- Parametreler: **mu_v_id**, **görev**.
- **try-except** bloğu içinde, güncelleme SQL komutu çalıştırılır.
- **Tkinter Kütüphanesi ve Gerekli İçe Aktarma:**
 - **tkinter** kütüphanesi Tkinter olarak içe aktarılır.
 - **messagebox**, iletişim kutuları oluşturmak için içe aktarılır.
 - **Gemiler** sınıfı, veritabanındaki gemilerle ilgili işlemleri gerçekleştirir.
- **Bağlantı Dizesi Tanımlama:**
 - **connection_string**, SQL Server veritabanına bağlanmak için kullanılır.
 - **GemiForm Sınıfı:**
 - **GemiForm** sınıfı, Tkinter penceresini ve gemi bilgi formunu oluşturur.
 - Pencere başlığı "Gemi Bilgi Formu" olarak ayarlanır.
 - Etiketler ve giriş kutuları oluşturularak gemi bilgilerini girmek için arayüz sağlanır.
 - Ekleme, güncelleme ve silme işlemleri için ilgili düğmeler ve bunların işlevleri belirlenir.
- **Gemi Ekleme (gemi_ekle):**
 - Kullanıcının girdiği gemi bilgileri alınır.
 - **Gemiler** sınıfı kullanılarak veritabanına yeni gemi eklenir.
 - İşlem başarılıysa bilgilendirme iletişim kutusu gösterilir, aksi halde hata iletilir.
- **Gemi Güncelleme (gemi_guncelle):**
 - Kullanıcıdan gemi ID'si alınır.
 - Gerekli güncelleme bilgileri alınır ve **Gemiler** sınıfı kullanılarak gemi güncellenir.
 - İşlem başarılıysa bilgilendirme iletişim kutusu gösterilir, aksi halde hata iletilir.
- **Gemi Silme (gemi_sil):**
 - Kullanıcıdan gemi ID'si alınır.
 - **Gemiler** sınıfı kullanılarak gemi silinir.
 - İşlem başarılıysa bilgilendirme iletişim kutusu gösterilir, aksi halde hata iletilir.
- **Ana Fonksiyon ve Uygulamanın Başlatılması:**
 - **main** fonksiyonu, Tkinter uygulamasını başlatır.
 - **GemiForm** sınıfından bir örnek oluşturularak uygulama çalıştırılır.

4.2 Görev dağılımı

- Grup çalışması yapılmamıştır. Tüm yazılım kodları, veritabanı tasarımı ve diğer gereksinimler bireysel olarak geliştirilmiştir.

4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

- Geliştirme sürecinde en büyük zorluk, Python kodu ile veritabanı arasında sağlıklı bir bağlantı kurmaktı. bu sorunu aşmak için sorunu tekrar gözden geçirdim ve çevrimiçi kaynaklardan yardım aldım. Sonuç olarak, form oluşturma kısmını tamamlamak için gerekli olan bağlantıyı sağla

4.4 Proje isterlerine göre eksik yönler

- Proje gerçekleştirilmesi beklenen görevlerden hiç kodlanamayanlar varsa belirtilmesi Proje isterlerine göre eksik yönler arasında, form oluşturma kısmını tamamlayamadım. Form için yazdığım kod ile veritabanı arasında bağlantı kuramadığım için, form ekranı açılrsa da herhangi bir değişiklik yapılamamaktadır.

5 TEST VE DOĞRULAMA

5.1 Yazılımın test süreci

```
import unittest
from sınıflar import Gemiler

class TestGemiler(unittest.TestCase):

    def setUp(self):
        connection_string = "Driver={SQL Server 16.0.1000};Server=ZehraYARDIMCI;Database=GemiTakipDB;Uid=sa;Pwd=zehra30;"
        self.gemiler = Gemiler(self.connection_string)

    def test_gemi_ekle(self):
        # Gemiler sınıfının ekle() metodunu test etme
        self.gemiler.ekle(seri_numarasi="ABC123", adi="Gemix", agirlik=1000, yapim_yili=2022, tur="Yolcu Gemişi", maks_agirlik=5000)
        # Ekleme işleminden sonra gemi_id'siyle getirme işlemini kontrol etme
        gemi = self.gemiler.getir("ABC123")
        self.assertIsNotNone(gemi)
        self.assertEqual(gemi[1], second="Gemix")
        self.assertEqual(gemi[2], second=1000)
        self.assertEqual(gemi[3], second=2022)
        self.assertEqual(gemi[4], second="Yolcu Gemişi")
        self.assertEqual(gemi[5], second=5000)

    def test_gemi_guncelle(self):
        # Gemiler sınıfının guncelle() metodunu test etme
        self.gemiler.guncelle(gemi_id="ABC123", adi="Gemix 2", agirlik=1500)
        # Güncelleme işleminden sonra gemi_id'siyle getirme işlemini kontrol etme
        gemi = self.gemiler.getir("ABC123")
```

```

self.assertEqual(gemi[5], second: 5000)

def test_gemi_guncelle(self):
    # Gemiler sınıfının guncelle() metodunu test etme
    self.gemiler.guncelle( gemiId: "ABC123", adi="Gemix 2", agirlik=1500)
    # Güncelleme işleminden sonra gemi_id'siyle getirme işlemini kontrol etme
    gemi = self.gemiler.getir("ABC123")
    self.assertIsNotNone(gemi)
    self.assertEqual(gemi[1], second: "Gemix 2")
    self.assertEqual(gemi[2], second: 1500)

def test_gemi_sil(self):
    # Gemiler sınıfının sil() metodunu test etme
    self.gemiler.sil("ABC123")
    # Silme işleminden sonra gemi_id'siyle getirme işleminin yapılamadığını kontrol etme
    gemi = self.gemiler.getir("ABC123")
    self.assertIsNone(gemi)

if __name__ == '__main__':
    unittest.main()

```

5.2 Yazılımın doğrulanması

- Test aşamasında, form ekranı açılrsa da veritabanıyla bağlantı kurulamadığı için herhangi bir değişiklik yapılamamaktadır.

GİTHUB LİNK:

<https://github.com/Zehrayardimci>

5.3 KAYNAKÇA

https://www.youtube.com/watch?v=3_pRrH3E4q0

<https://ilkeronurisik.medium.com/pycharm-ile-mssql-server-ba%C4%9Flant%C4%B1s%C4%B1n%C4%B1n-kurulmas%C4%B1-67ddb7778bda>

<https://www.youtube.com/watch?v=mbf3mmahXTQ&t=55s>

<https://www.youtube.com/watch?v=FurB2AriHjE&t=2s>

<https://www.youtube.com/watch?v=muarMeONMNQ&list=PLURN6mxdcwL-Ame5P9E0uppmbk81USX-A&index=5>