

**BURDUR MEHMET AKİF ERSOY ÜNİVERSİTESİ   
TEKNİK BİLİMLER MESLEK YÜKSEKOKULU**  
BİLGİSAYAR TEKNOLOJİLERİ BÖLÜMÜ   
BİLGİSAYAR PROGRAMCILIĞI PROGRAMI

**GÖRSEL PROGRAMLAMA  
ÜYE KAYIT OTOMASYONU  
FİNAL SINAVI PROJESİ**

2313409024 – Alper Tığrak  
alpertigrak1@icloud.com

2024 – BURDUR

# İçindekiler

[İçindekiler 2](#_Toc185797391)

[Veri Tabanı Oluşturma Sorguları 3](#_Toc185797392)

[Python Programlama Dili İle Gerçekleştirilen Üye Kayıt Detayları 5](#_Toc185797393)

[Projeyi Gerçekleştirme Aşamaları 6](#_Toc185797394)

[Arayüz Görseli 25](#_Toc185797395)

[Kaynaklar 26](#_Toc185797396)

# Veri Tabanı Oluşturma Sorguları

--create database KutuphaneDB4

--KutuphaneDB veritabanını kullanır.

use KutuphaneDB1

----uye taplosu oluşturur.

--create table uye(

-- UyeID INT IDENTITY (1,1) PRIMARY KEY, --Otomatik artan benzersiz ID

-- TCKimlikNo NVARCHAR(11) NOT NULL UNIQUE, --TCkimlik numarası benzersiz ve zorunlu

-- Ad NVARCHAR(50) NOT NULL, --Üyenin Adı(zorunlu)

-- Soyad NVARCHAR(50) NOT NULL, --Üyenin Soyadı(zorunlu)

-- Meslek NVARCHAR(50) NOT NULL,

-- CepTel NVARCHAR(15) NOT NULL, --Cep telefonu rakam içermeli ve zorunlu

-- Eposta NVARCHAR(50) NOT NULL, --Eposta Adresi

-- Adres NVARCHAR(50) NOT NULL,

-- --Ceptel için kısıtlama(+90 ile başlayacak ve 10 rakam olacak)

-- CONSTRAINT chk\_CepTel CHECK(

-- CepTel LIKE '+90%' AND LEN(CepTel) = 13 AND ISNUMERIC(SUBSTRING(CepTel,4,LEN(CepTel)))=1),

-- --TCKimlik no için kısıtlama(11 haneli rakam)

-- CONSTRAINT chk\_TCKimlikNo CHECK(LEN(TCKimlikNo) = 11 AND ISNUMERIC(TCKimlikNo)=1),

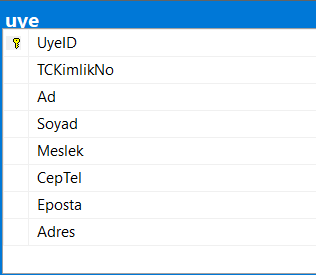
-- CONSTRAINT chk\_Eposta CHECK(

-- CHARINDEX('@', Eposta) > 1 AND

-- CHARINDEX('.', Eposta, CHARINDEX('@', Eposta)) > CHARINDEX('@', Eposta) + 1)

--);

Veri Tabanı Görseli



Şekil Diagram

# Python Programlama Dili İle Gerçekleştirilen Üye Kayıt Detayları

Aşağıdaki gibi dosya isimlerine dikkate ederek (sınıf) proje oluşturunuz. Detaylar aşağıda yer almaktadır.

**Kodlama yaparken mutlaka açıklama (yorum) satırı kullanınız!**

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil

# Projeyi Gerçekleştirme Aşamaları

1.Veritabanı oluşturdum ve kodları yazdım.

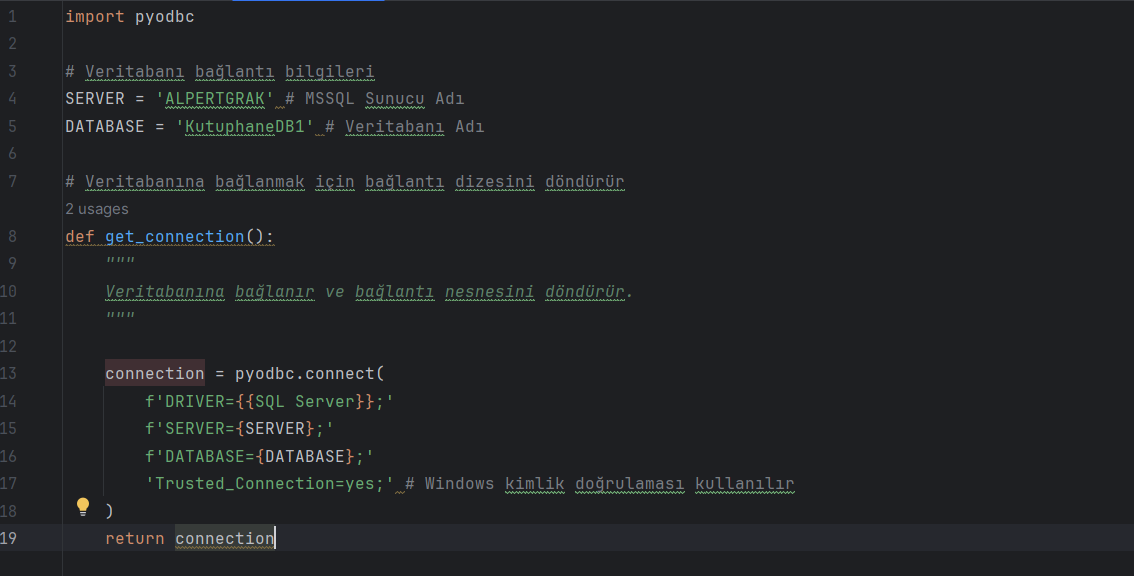
2.config.py sayfasını oluşturup kodlarını yazdım. veritabanı ve Python sayfalarına bağlantı kurdum.

3.VT\_Sorgular.py sayfası oluşturdum ve gerekli kodları yazdım. projedeki ekle,güncelle,sil,listele,bul alanlarını yazdım.

4.Arayuz.py sayfası oluşturdum ve kodları yazdım. proje için arayüz oluşturdum.

config.py

import pyodbc  
  
# Veritabanı bağlantı bilgileri  
SERVER = 'ALPERTGRAK' # MSSQL Sunucu Adı  
DATABASE = 'KutuphaneDB1' # Veritabanı Adı  
  
# Veritabanına bağlanmak için bağlantı dizesini döndürür  
def get\_connection():  
 *"""  
 Veritabanına bağlanır ve bağlantı nesnesini döndürür.  
 """* connection = pyodbc.connect(  
 f'DRIVER={{SQL Server}};'  
 f'SERVER={SERVER};'  
 f'DATABASE={DATABASE};'  
 'Trusted\_Connection=yes;' # Windows kimlik doğrulaması kullanılır  
 )  
 return connection



Şekil Config.py

**VT\_Sorgulari.py**

# VT\_SORGULAR  
  
import pyodbc  
from config import get\_connection # config.py'den get\_connection fonksyonunu alıyoruz  
  
  
class Uye:  
 @staticmethod  
 def connect\_db():  
 *"""  
 veritabanına bağlanr ve bağlantı nesnesi döndürür.  
 """* connection = get\_connection() # config.py'dengelen bağlantı fonksiyonunu kullanıyoruz.  
 return connection # bağlantı nesnesini döndürür.  
  
 @staticmethod  
 def ekle(TCKimlikNo, Ad, Soyad, Meslek, CepTel, Eposta, Adres):  
 *"""  
 Yeni bir üye ekler  
  
 parametreler:  
 -TCKimlikNo(str):Üyenin tc kimlik no  
 -Ad(str):Üyenin adı  
 -Soyad(str):Üyenin Soyadı  
 -CepTel(str):Üyenin cep telefonu  
 -Eposta(str):Üyenin e-posta adresi  
 """* conn = Uye.connect\_db()  
 cursor = conn.cursor()  
  
 try:  
 # önce tckimlik numarası kayıtlımı kontrol et.  
 cursor.execute("SELECT COUNT(\*) FROM uye WHERE TCKimlikNo=?", (TCKimlikNo))  
 count = cursor.fetchone()[0]  
  
 if count > 0:  
 raise ValueError("Bu TC Kimlik Numarası Zaten Kayıtlı.")  
  
 # eğer kayıtlı değilse ekle.  
 cursor.execute(  
 "INSERT INTO uye (TCKimlikNo,Ad,Soyad,Meslek,CepTel,Eposta,Adres) VALUES(?,?,?,?,?,?,?)",  
 (TCKimlikNo, Ad, Soyad, Meslek, CepTel, Eposta, Adres)  
 )  
 conn.commit()  
 except pyodbc.Error as e:  
 error\_message = str(e)  
 if "chk\_CepTel" in error\_message:  
 raise ValueError(  
 "Cep telefonu formatı hatalı! Lütfen yalnızca rakam girin ve 10-15 haneli olduğundan emin olun.")  
 elif "TCKimlikNo" in error\_message:  
 raise ValueError("TCKimlik Numarası geçersiz veya zaten kayıtlı.")  
 elif "chk\_Eposta" in error\_message:  
 raise ValueError("E-posta adresi geçersiz! Lütfen '@' ve '.' karakterlerini doğru şekilde kullanın.")  
 else:  
 raise ValueError(f'Veeritabanı hatası: {e}')  
 finally:  
 conn.close()  
  
 @staticmethod  
 def guncelle(UyeID, TCKimlikNo, Ad, Soyad, Meslek, Ceptel, Eposta, Adres):  
 *"""  
 tc kimlik numarasına göre bilgileri günceller  
  
 parametreler:  
 -TCKimlikNo (str):Güncellenecek üyenin TC Kimlik Numarası  
 -Ad (str): Yeni ad  
 -Soyad (str): Yeni Soyad  
 -CepTel(str):Yeni cep telefonu  
 -Eposta (str):Yeni e-posta adresi  
 """* conn = Uye.connect\_db()  
 cursor = conn.cursor()  
  
 try:  
 # Güncelleme işlemi  
 cursor.execute(  
 "UPDATE uye SET TCKimlikNo=?,Ad=?, Soyad=?, Meslek=?, CepTel=?, Eposta=?, Adres=? WHERE UyeID=?",  
 (TCKimlikNo, Ad, Soyad, Meslek, Ceptel, Eposta, Adres, UyeID)  
 )  
 conn.commit()  
  
 # Güncelleme sonrası veri kontrolü  
 cursor.execute(  
 "SELECT \* FROM uye WHERE TCKimlikNo=?",  
 (TCKimlikNo,)  
 )  
 result = cursor.fetchone()  
  
 if result:  
 print("Güncellenen Kayıt: ", result)  
 else:  
 raise ValueError("Veritabanında bu TC Kimlik Numarası ile kayıt bulunamadı.")  
  
 except pyodbc.Error as e:  
 error\_message = str(e)  
 if "chk\_CepTel" in error\_message:  
 raise ValueError(  
 "Cep telefonu formatı hatalı! Lütfen yalnızca rakam girin ve 10-15 haneli olduğundan emin olun.")  
 elif "TCKimlikNo" in error\_message:  
 raise ValueError("TCKimlik numarası geçersiz veya zaten kayıtlı.")  
 elif "chk\_Eposta" in error\_message:  
 raise ValueError("E-posta adresi geçersiz! Lütfen '@' ve '.' karakterlerini doğru şekilde kullanın.")  
 else:  
 raise ValueError(f'Güncelleme Hatası: {e}')  
 finally:  
 conn.close()  
  
 @staticmethod  
 def sil(TCKimlikNo):  
 *"""  
 Tc kimlik numarasına göre üye siler:  
  
 Parametreler:  
 TCKimlikNo(str):Silinecek üyenin TcKimlikNumarası  
 """* conn = Uye.connect\_db()  
 cursor = conn.cursor()  
 try:  
 cursor.execute("DELETE FROM uye WHERE TCKimlikNo=?", (TCKimlikNo))  
 conn.commit()  
 except pyodbc.Error as e:  
 raise ValueError(f"Silme Hatası:{e}")  
 finally:  
 conn.close()  
  
 @staticmethod  
 def listele():  
 *"""  
 Tüm Üyeleri listeler  
  
 Geri Dönüş:  
 -list[truple]: Tüm üyeler bilgilerini içeren bir liste  
 """* conn = Uye.connect\_db()  
 cursor = conn.cursor()  
 try:  
 cursor.execute("SELECT \* FROM uye")  
 return cursor.fetchall() # tüm kayıtrları döndürür  
 finally:  
 conn.close()  
  
 @staticmethod  
 def bul(TCKimlikNo):  
 *"""  
 TC kimlik numarasına göre üye arar.  
  
 parametreler:  
 -TCKimlikNo(str): Aranacak üyenin TC Kimlik numarası  
  
 Geri Dönüş:  
 tuple:Eğer üye bulunursa, üye bilgilerini içeren bir truple döndürür.  
  
 - None: eğer üye bulunmazsa None döndürür.  
 """* conn = Uye.connect\_db()  
 cursor = conn.cursor()  
  
 try:  
 cursor.execute("SELECT \* FROM uye WHERE TCKimlikNo=?", (TCKimlikNo,))  
 uye = cursor.fetchone() # Sadece bir sonuç döndürürlür.  
 return uye # eğer üye varsa,bilgilerini döndür.  
 finally:  
 conn.close()

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil vt-sorgular

metin, ekran görüntüsü içeren bir resim

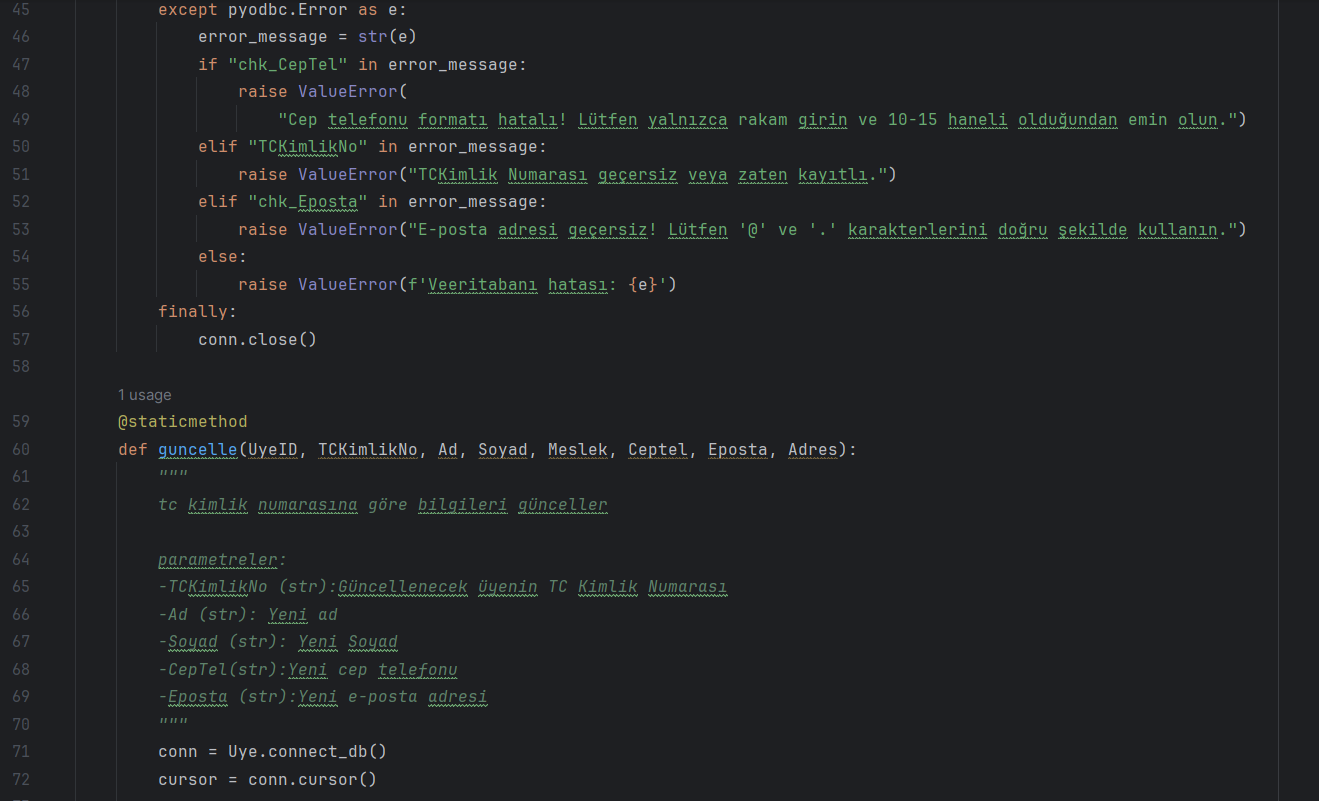
Açıklama otomatik olarak oluşturuldu

Şekil vt-sorgular

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

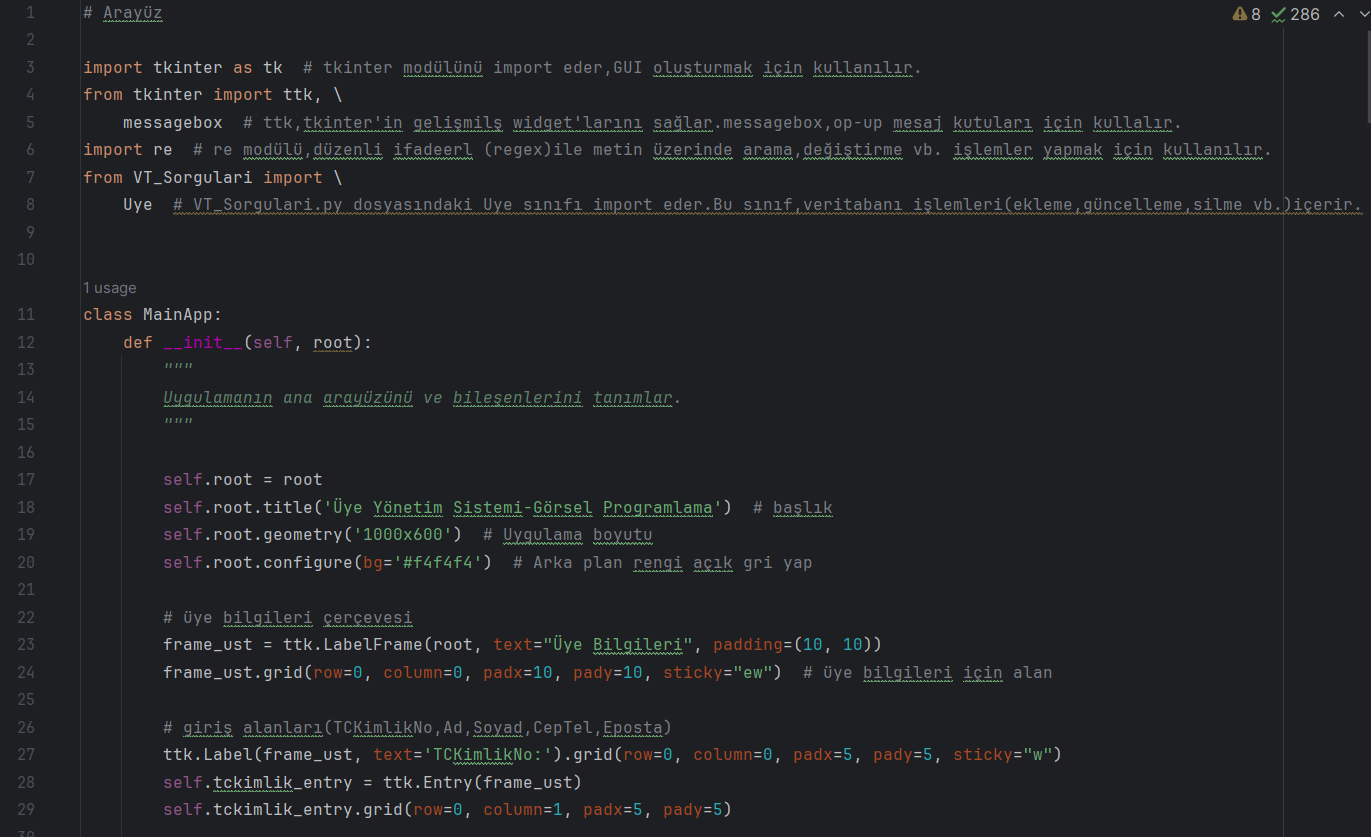
Şekil vt-sorgular



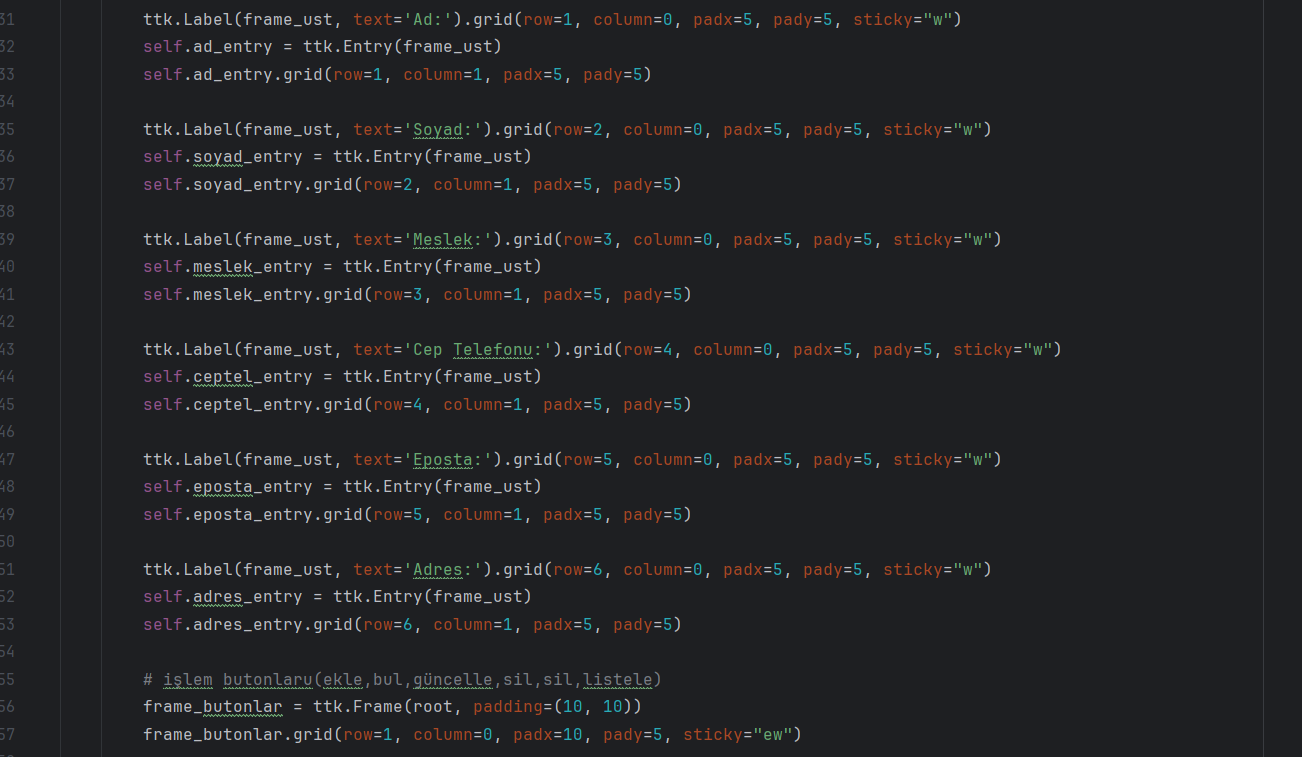
Şekil vt-sorgular

**Arayuz.py**

# Arayüz  
  
import tkinter as tk # tkinter modülünü import eder,GUI oluşturmak için kullanılır.  
from tkinter import ttk, \  
 messagebox # ttk,tkinter'in gelişmilş widget'larını sağlar.messagebox,op-up mesaj kutuları için kullalır.  
import re # re modülü,düzenli ifadeerl (regex)ile metin üzerinde arama,değiştirme vb. işlemler yapmak için kullanılır.  
from VT\_Sorgulari import \  
 Uye # VT\_Sorgulari.py dosyasındaki Uye sınıfı import eder.Bu sınıf,veritabanı işlemleri(ekleme,güncelleme,silme vb.)içerir.  
  
  
class MainApp:  
 def \_\_init\_\_(self, root):  
 *"""  
 Uygulamanın ana arayüzünü ve bileşenlerini tanımlar.  
 """* self.root = root  
 self.root.title('Üye Yönetim Sistemi-Görsel Programlama') # başlık  
 self.root.geometry('1000x600') # Uygulama boyutu  
 self.root.configure(bg='#f4f4f4') # Arka plan rengi açık gri yap  
  
 # üye bilgileri çerçevesi  
 frame\_ust = ttk.LabelFrame(root, text="Üye Bilgileri", padding=(10, 10))  
 frame\_ust.grid(row=0, column=0, padx=10, pady=10, sticky="ew") # üye bilgileri için alan  
  
 # giriş alanları(TCKimlikNo,Ad,Soyad,CepTel,Eposta)  
 ttk.Label(frame\_ust, text='TCKimlikNo:').grid(row=0, column=0, padx=5, pady=5, sticky="w")  
 self.tckimlik\_entry = ttk.Entry(frame\_ust)  
 self.tckimlik\_entry.grid(row=0, column=1, padx=5, pady=5)  
  
 ttk.Label(frame\_ust, text='Ad:').grid(row=1, column=0, padx=5, pady=5, sticky="w")  
 self.ad\_entry = ttk.Entry(frame\_ust)  
 self.ad\_entry.grid(row=1, column=1, padx=5, pady=5)  
  
 ttk.Label(frame\_ust, text='Soyad:').grid(row=2, column=0, padx=5, pady=5, sticky="w")  
 self.soyad\_entry = ttk.Entry(frame\_ust)  
 self.soyad\_entry.grid(row=2, column=1, padx=5, pady=5)  
  
 ttk.Label(frame\_ust, text='Meslek:').grid(row=3, column=0, padx=5, pady=5, sticky="w")  
 self.meslek\_entry = ttk.Entry(frame\_ust)  
 self.meslek\_entry.grid(row=3, column=1, padx=5, pady=5)  
  
 ttk.Label(frame\_ust, text='Cep Telefonu:').grid(row=4, column=0, padx=5, pady=5, sticky="w")  
 self.ceptel\_entry = ttk.Entry(frame\_ust)  
 self.ceptel\_entry.grid(row=4, column=1, padx=5, pady=5)  
  
 ttk.Label(frame\_ust, text='Eposta:').grid(row=5, column=0, padx=5, pady=5, sticky="w")  
 self.eposta\_entry = ttk.Entry(frame\_ust)  
 self.eposta\_entry.grid(row=5, column=1, padx=5, pady=5)  
  
 ttk.Label(frame\_ust, text='Adres:').grid(row=6, column=0, padx=5, pady=5, sticky="w")  
 self.adres\_entry = ttk.Entry(frame\_ust)  
 self.adres\_entry.grid(row=6, column=1, padx=5, pady=5)  
  
 # işlem butonlaru(ekle,bul,güncelle,sil,sil,listele)  
 frame\_butonlar = ttk.Frame(root, padding=(10, 10))  
 frame\_butonlar.grid(row=1, column=0, padx=10, pady=5, sticky="ew")  
  
 ttk.Button(frame\_butonlar, text="Ekle", command=self.ekle).grid(row=0, column=0, padx=10, pady=10)  
 ttk.Button(frame\_butonlar, text="Bul", command=self.bul).grid(row=0, column=1, padx=10, pady=10)  
 self.guncelle\_btn = ttk.Button(frame\_butonlar, text="Güncelle", command=self.guncelle, state='disabled')  
 self.guncelle\_btn.grid(row=0, column=2, padx=10, pady=10)  
 self.sil\_btn = ttk.Button(frame\_butonlar, text="Sil", command=self.sil, state='disabled')  
 self.sil\_btn.grid(row=0, column=3, padx=10, pady=10)  
 ttk.Button(frame\_butonlar, text="Listele", command=self.listele).grid(row=0, column=4, padx=10, pady=10)  
  
 # listeleme alanı(treewiew)  
 frame\_liste = ttk.LabelFrame(root, text="Üye Listesi", padding=(10, 10))  
 frame\_liste.grid(row=2, column=0, padx=10, pady=10, sticky="nsew")  
  
 self.tree = ttk.Treeview(frame\_liste,  
 columns=("UyeID", "TCKimlikNo", "Ad", "Soyad", "Meslek", "CepTel", "Eposta", "Adres"),  
 show="headings")  
 self.tree.grid(row=0, column=0, sticky="nsew")  
 scrollbar = ttk.Scrollbar(frame\_liste, orient="vertical", command=self.tree.yview)  
 self.tree.configure(yscrollcommand=scrollbar.set)  
 scrollbar.grid(row=0, column=1, sticky="ns")  
  
 # treewiew sütun başlıkları  
 self.tree.heading("UyeID", text="Üye ID")  
 self.tree.heading("TCKimlikNo", text="TC Kimlik No")  
 self.tree.heading("Ad", text="Ad")  
 self.tree.heading("Soyad", text="Soyad")  
 self.tree.heading("Meslek", text="Meslek")  
 self.tree.heading("CepTel", text="Cep Telefonu")  
 self.tree.heading("Eposta", text="E-posta")  
 self.tree.heading("Adres", text="Adres")  
  
 # treewiew sütun genişliklerini manuel olarak ayarlıyoruz  
 self.tree.column("UyeID", width=100, anchor="center")  
 self.tree.column("TCKimlikNo", width=150, anchor="center")  
 self.tree.column("Ad", width=150, anchor="w")  
 self.tree.column("Soyad", width=150, anchor="w")  
 self.tree.column("Meslek", width=150, anchor="w")  
 self.tree.column("CepTel", width=150, anchor="center")  
 self.tree.column("Eposta", width=200, anchor="w")  
 self.tree.column("Adres", width=200, anchor="w")  
  
 root.grid\_rowconfigure(2, weight=1) # listeleme alanı için esneklik sağlıyoruz  
  
 root.grid\_columnconfigure(0, weight=1) # listeleme alanı için esneklik sağlıyoruz  
 frame\_liste.grid\_rowconfigure(0, weight=1) # treewiew alanı için esneklik sağlıyoruz  
 frame\_liste.grid\_columnconfigure(0, weight=1) # treewiew sutünları giriş alanına aktar  
  
 self.tree.bind("<<TreeviewSelect>>", self.kayit\_secildi)  
  
 def ekle(self):  
 *"""  
 Yeni üyeler ekler  
 """* tckimlik = self.tckimlik\_entry.get()  
 ad = self.ad\_entry.get()  
 soyad = self.soyad\_entry.get()  
 meslek = self.meslek\_entry.get()  
 ceptel = self.ceptel\_entry.get()  
 eposta = self.eposta\_entry.get()  
 adres = self.adres\_entry.get()  
  
 # TC Kimlik No ve Ceptel formatı kontrolü  
 if not self.gecerli\_tckimlik(tckimlik):  
 messagebox.showerror("Hata", "Geçersiz TC Kimlik Numarası. 11 haneli olmalıdır")  
 return  
 if not self.gecerli\_ceptel(ceptel):  
 messagebox.showerror("Hata", "Geçersiz cep telefonu numarası. +90 ile başlayan ve 10 haneli olmalıdır.")  
 return  
 try:  
 Uye.ekle(tckimlik, ad, soyad, meslek, ceptel, eposta, adres) # Veritabanı ekleme işlemi  
 messagebox.showinfo("Başarı", "Üye başarıyla eklendi.")  
 self.temizle()  
 self.listele() # Listeyi güncelle  
 except Exception as e:  
 messagebox.showerror("Hata", str(e))  
  
 def gecerli\_tckimlik(self, tckimlik):  
 *"""  
 TC Kimlik Numarasının geçerli olup olmadığını kontrol eder (11 haneli ve rakam olabilir).  
 """* return tckimlik.isdigit() and len(tckimlik) == 11  
  
 def gecerli\_ceptel(self, ceptel):  
 *"""  
 Cep Telefonunun geçerli olup olmadığını kontrol eder (+90 ile başlayacak ve 10 rakam içermeli)  
 """* return bool(re.match(r"^\+90\d{10,15}$", ceptel))  
  
 def gecerli\_eposta(self, eposta):  
  
 if '@' not in eposta or '.' not in eposta:  
 return "Geçersiz e-posta adresi: '@' ve '.' karakterleri eksik."  
 if eposta.index('@') > eposta.index('.'):  
 return "Geçersiz e-posta adresi: '@' karakteri '.' karakterinden önce olamaz."  
 return True  
  
 def bul(self):  
 *"""  
 TC Kimlik Numarasına göre üye arar ve giriş alanlarını doldurur.  
 """* uye = Uye.bul(self.tckimlik\_entry.get()) # Veritabanına arama yapıyoruz  
 if uye:  
 # Giriş alanlarını bulduğumuz üye bilgileriyle dolduruyoruz  
 self.ad\_entry.delete(0, tk.END)  
 self.ad\_entry.insert(0, uye[2])  
 self.soyad\_entry.delete(0, tk.END)  
 self.soyad\_entry.insert(0, uye[3])  
 self.meslek\_entry.delete(0, tk.END)  
 self.meslek\_entry.insert(0, uye[4])  
 self.ceptel\_entry.delete(0, tk.END)  
 self.ceptel\_entry.insert(0, uye[5])  
 self.eposta\_entry.delete(0, tk.END)  
 self.eposta\_entry.insert(0, uye[6])  
 self.adres\_entry.delete(0, tk.END)  
 self.adres\_entry.insert(0, uye[7])  
  
 self.guncelle\_btn.config(state='normal') # Güncelle butonunu aktif yapıyoruz  
 self.sil\_btn.config(state='normal') # Sil butonunu aktif yapıyoruz  
 else:  
 messagebox.showinfo("Bilgi", "Üye Bulunamadı.") # Eğer üye bulunamazsa bilgi veriyoruz  
  
 def guncelle(self):  
 try:  
 selected\_item = self.tree.selection()[0]  
 selected\_data = self.tree.item(selected\_item, 'values')  
 Uye.guncelle(  
 selected\_data[0],  
 self.tckimlik\_entry.get(),  
 self.ad\_entry.get(),  
 self.soyad\_entry.get(),  
 self.meslek\_entry.get(),  
 self.ceptel\_entry.get(),  
 self.eposta\_entry.get(),  
 self.adres\_entry.get()  
 )  
  
 except Exception as e:  
 messagebox.showerror("Hata", f"Güncelleme Başarısız{(e)}")  
  
 def sil(self):  
 *"""  
 Seçilen üyenin kaydını siler.  
 """* try:  
 Uye.sil(self.tckimlik\_entry.get()) # Veritabanında silme işlemi  
 messagebox.showinfo("Başarı", "Üye başarıyla silindi.")  
 self.temizle()  
 self.listele() # Listeyi Güncelle  
 except Exception as e:  
 messagebox.showerror("Hata", f"Silme işlemi başarısız: {e}")  
  
 def listele(self):  
 *"""  
 Tüm üyeleri listeler.  
 """* for row in self.tree.get\_children(): # Önceden eklenmiş tüm verileri temizle  
 self.tree.delete(row)  
 uyeler = Uye.listele() # Veritabanındaki tüm verileri listele  
 for uye in uyeler:  
 # Veriyi düzgün formatta Treeview'e ekleriz  
 self.tree.insert("", tk.END, values=(  
 str(uye[0]), str(uye[1]), str(uye[2]), str(uye[3]), str(uye[4]), str(uye[5]), str(uye[6]), str(uye[7])))  
  
 def temizle(self):  
 *"""  
 Tüm giriş alanlarını temizler ve butonları devre dışı bırakır.  
 """* self.tckimlik\_entry.delete(0, tk.END)  
 self.ad\_entry.delete(0, tk.END)  
 self.soyad\_entry.delete(0, tk.END)  
 self.meslek\_entry.delete(0, tk.END)  
 self.ceptel\_entry.delete(0, tk.END)  
 self.eposta\_entry.delete(0, tk.END)  
 self.adres\_entry.delete(0, tk.END)  
 self.guncelle\_btn.config(state='disabled')  
 self.sil\_btn.config(state='disabled')  
  
 def kayit\_secildi(self, event):  
 *"""  
 Treeview'dan seçilen kaydı giriş alanlarına doldurur  
 """* try:  
 selected\_item = self.tree.selection()[0] # Seçilen kaydı alıyoruz  
 selected\_data = self.tree.item(selected\_item, 'values') # Kaydın verilerini alıyoruz  
 # Verileri giriş alanlarına yerleştiriyoruz  
 self.tckimlik\_entry.delete(0, tk.END)  
 self.tckimlik\_entry.insert(0, selected\_data[1])  
 self.ad\_entry.delete(0, tk.END)  
 self.ad\_entry.insert(0, selected\_data[2])  
 self.soyad\_entry.delete(0, tk.END)  
 self.soyad\_entry.insert(0, selected\_data[3])  
 self.meslek\_entry.delete(0, tk.END)  
 self.meslek\_entry.insert(0, selected\_data[4])  
 self.ceptel\_entry.delete(0, tk.END)  
 self.ceptel\_entry.insert(0, selected\_data[5])  
 self.eposta\_entry.delete(0, tk.END)  
 self.eposta\_entry.insert(0, selected\_data[6])  
 self.adres\_entry.delete(0, tk.END)  
 self.adres\_entry.insert(0, selected\_data[7])  
  
 self.guncelle\_btn.config(state='normal') # Güncelle ve sil butonlarını aktif yapıyoruz  
 self.sil\_btn.config(state='normal')  
 except IndexError:  
 pass # Eğer seçim yapılmamışsa herhangi bir işlem yapmayalım  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = MainApp(root)  
 root.mainloop()



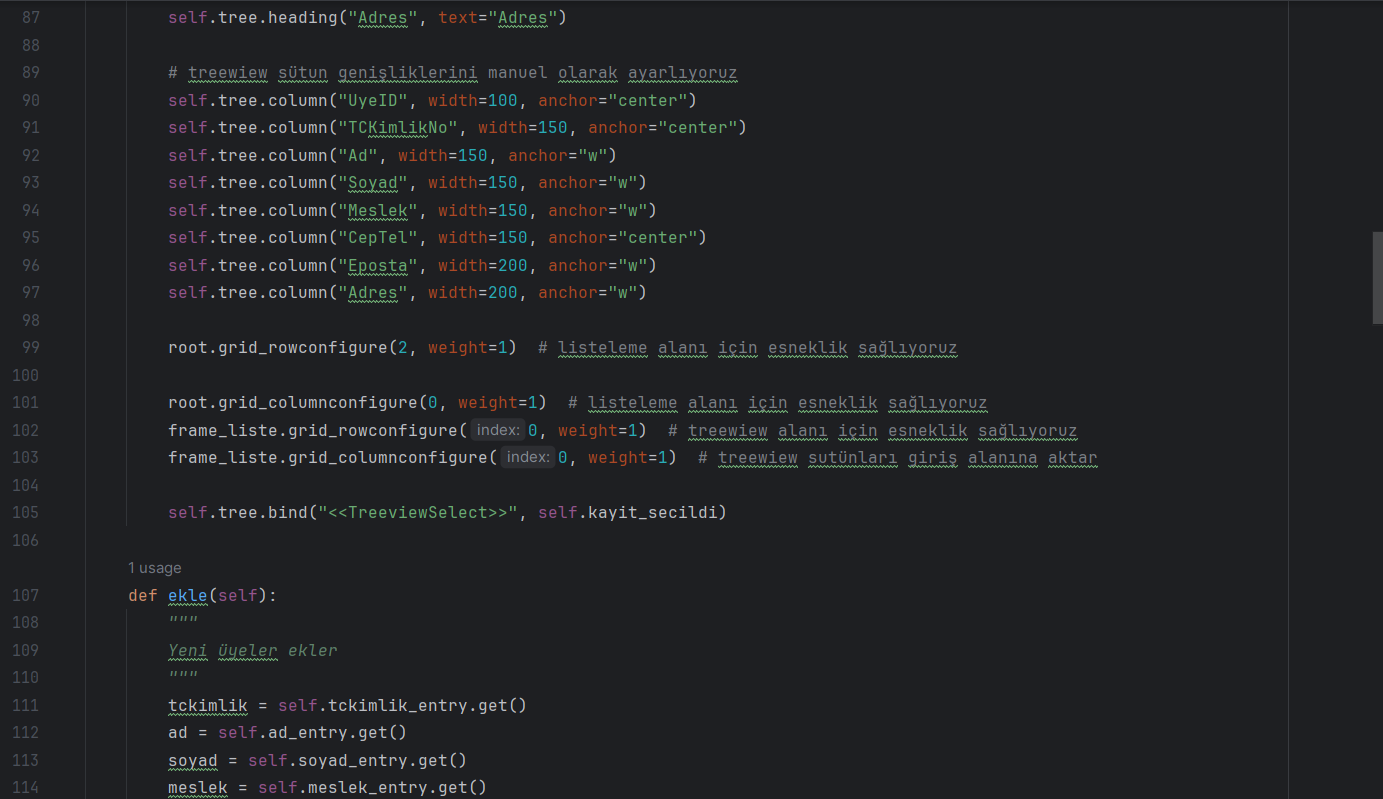
Şekil arayüz



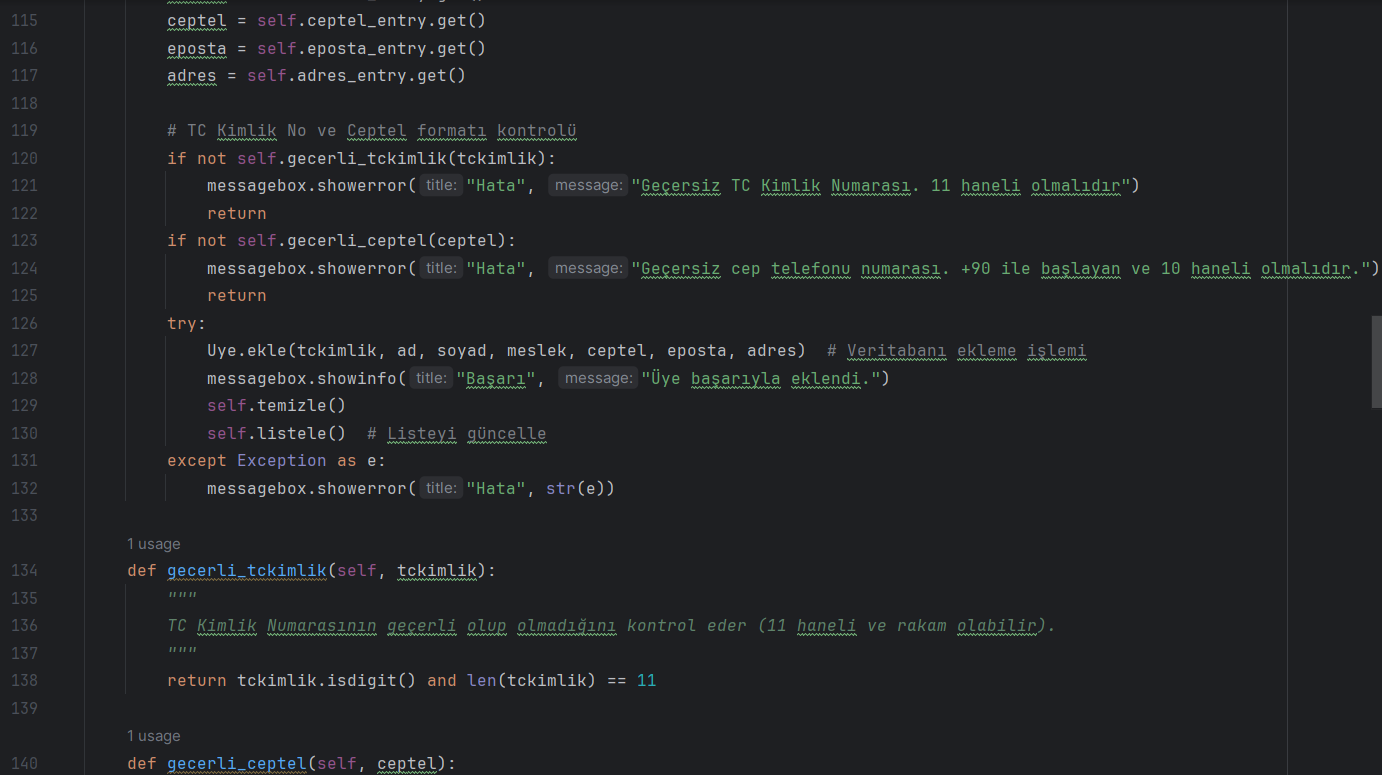
Şekil arayüz



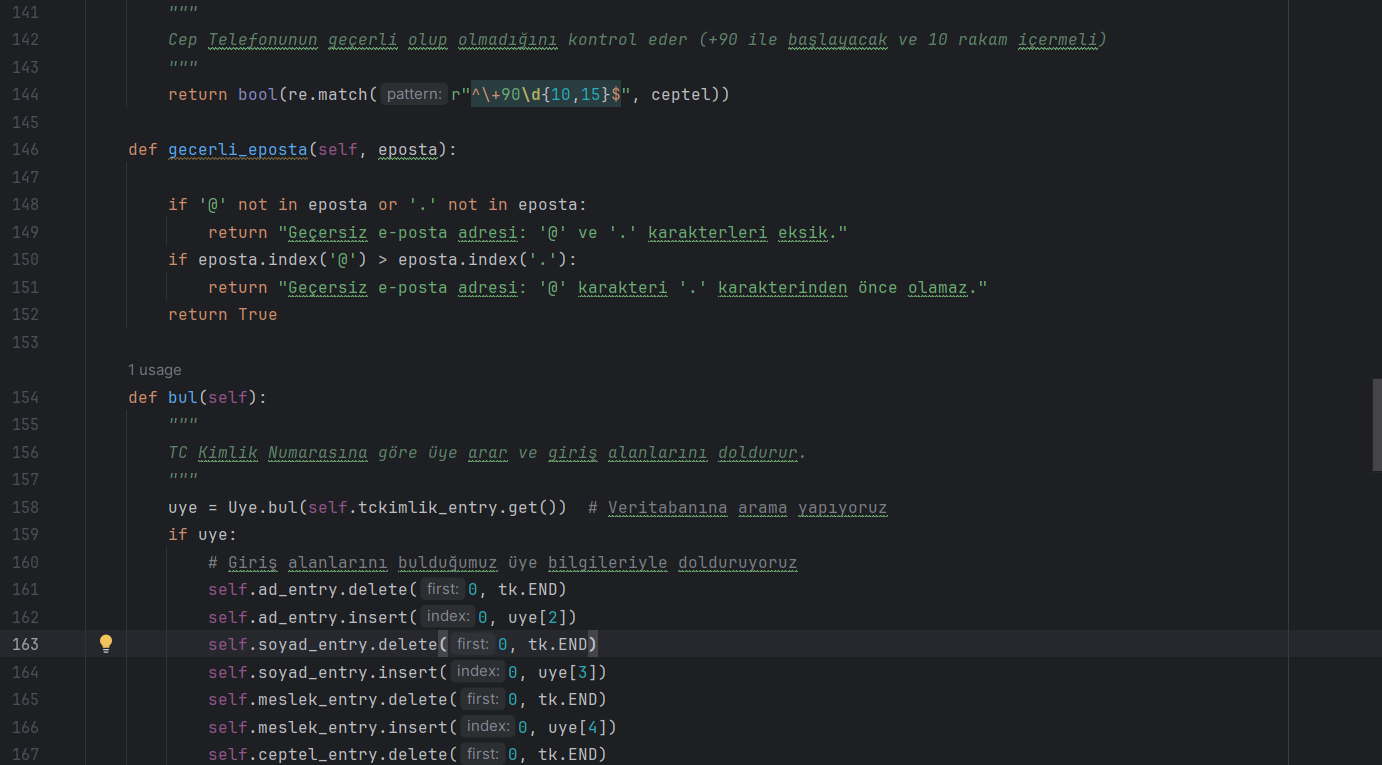
Şekil arayüz



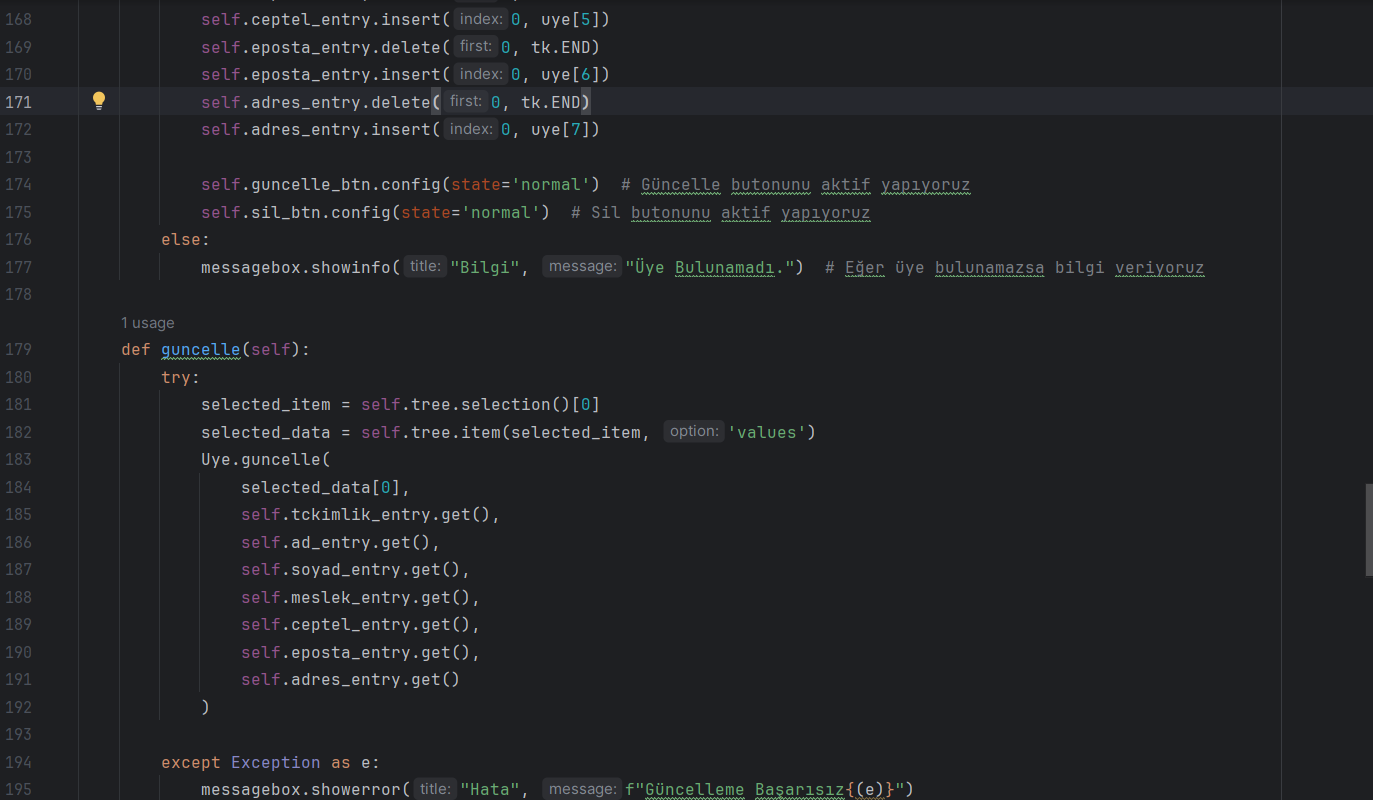
Şekil arayüz



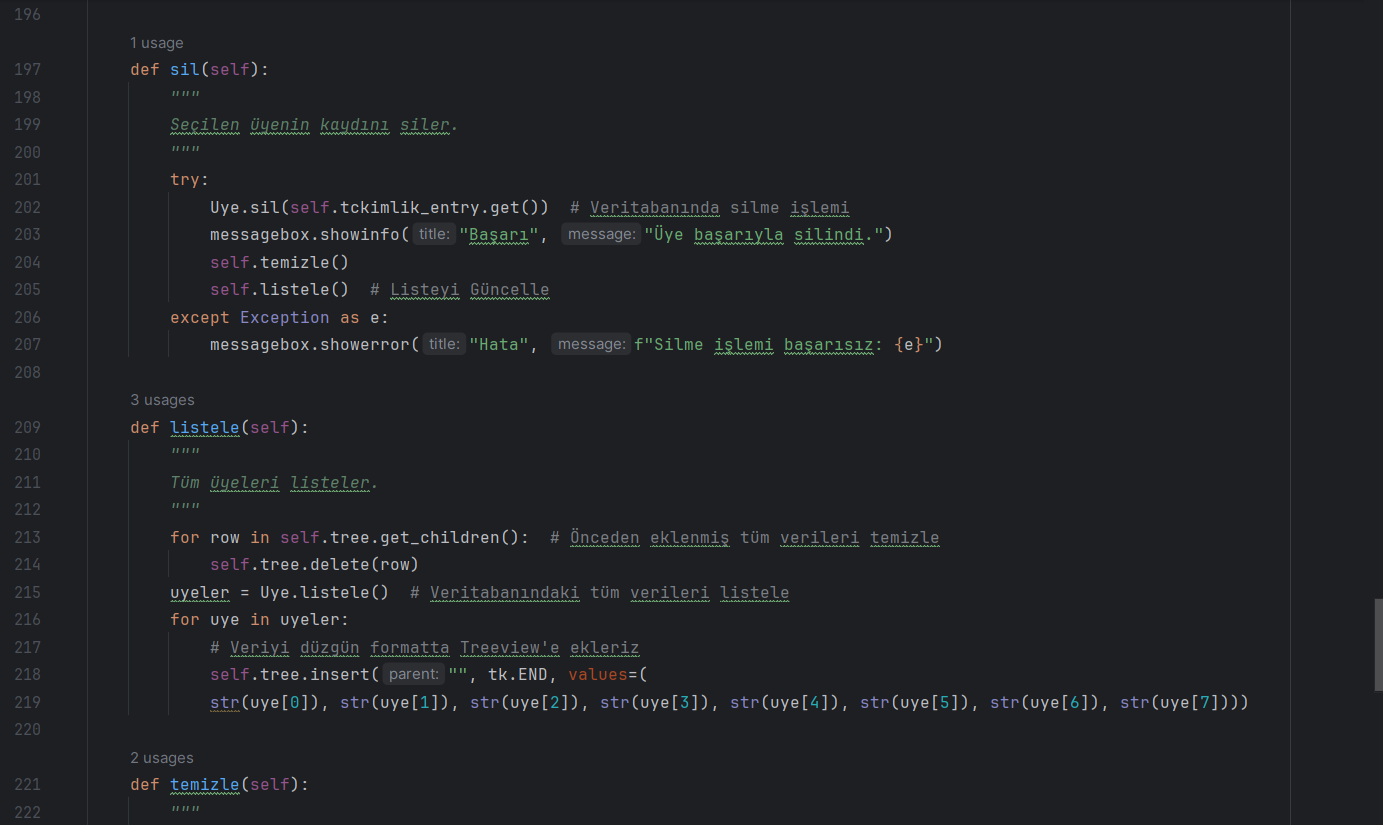
Şekil arayüz



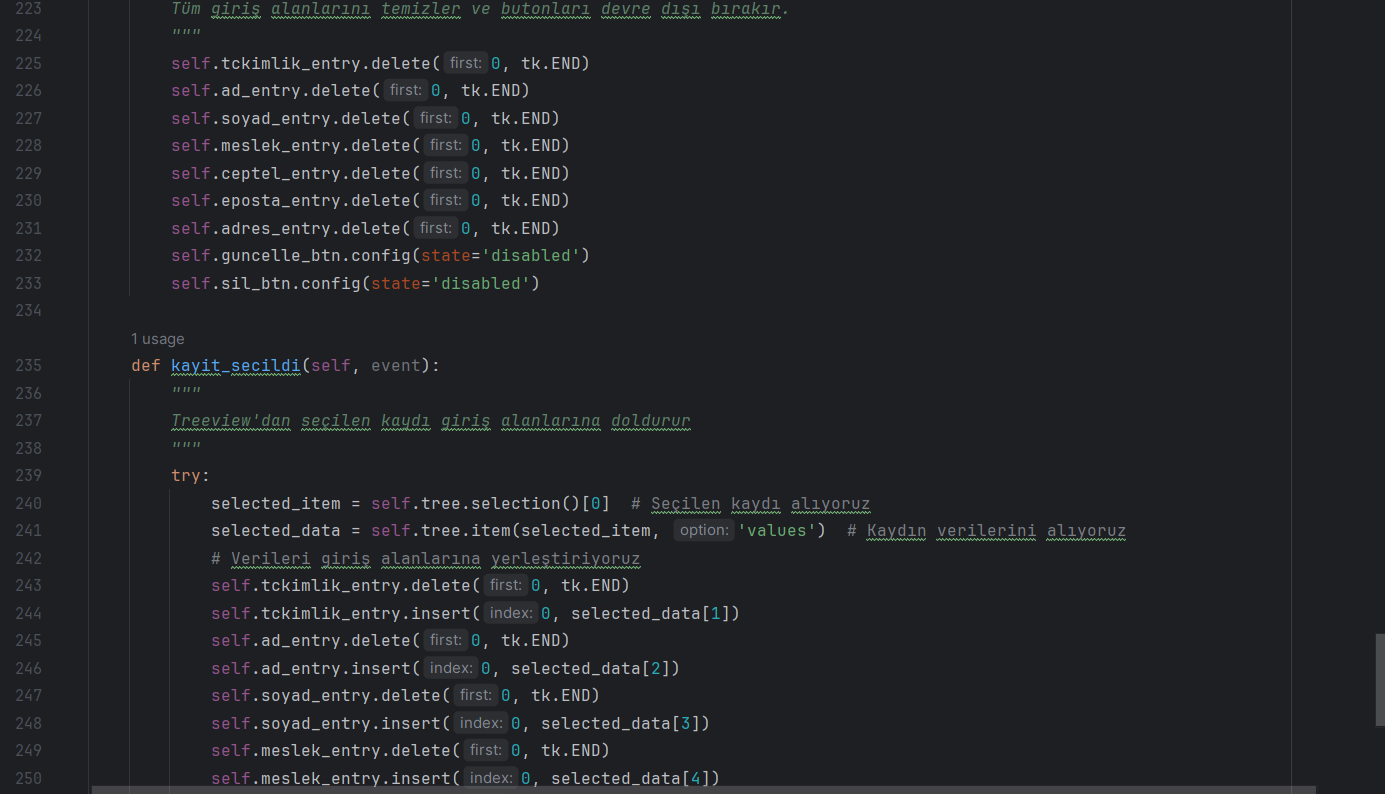
Şekil arayüz



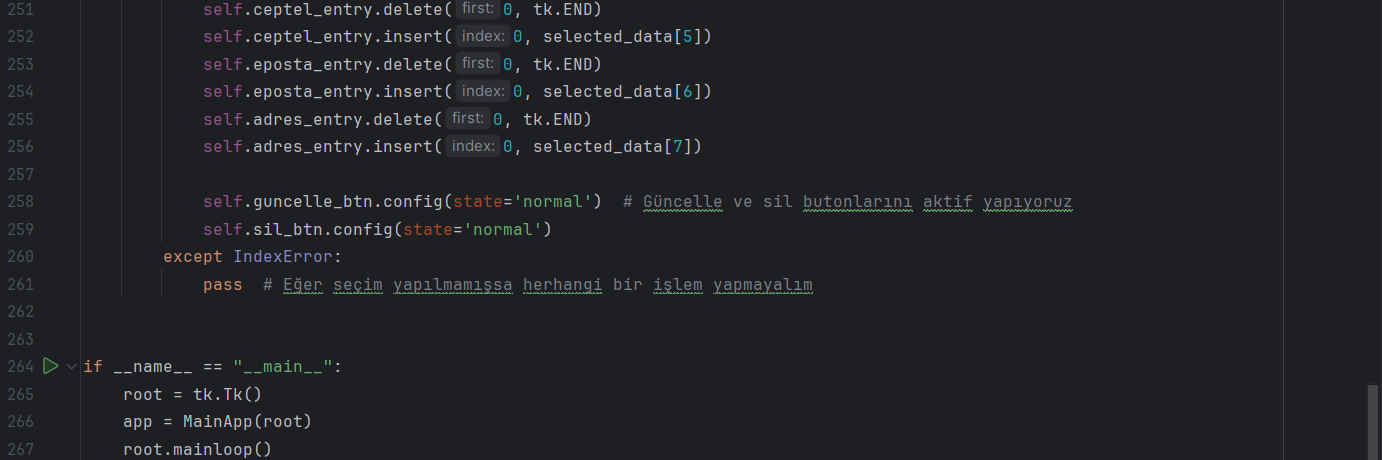
Şekil arayüz



Şekil arayüz

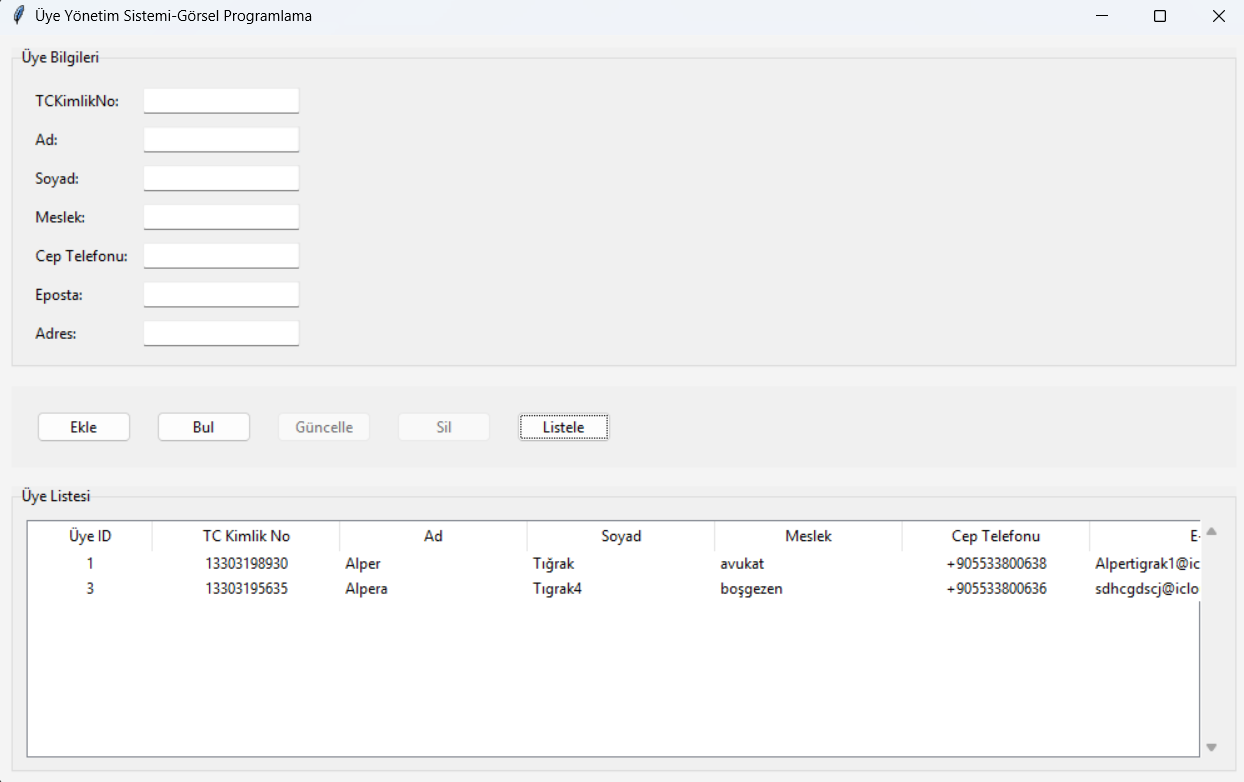


Şekil arayüz



Şekil arayüz

# Arayüz Görseli



# Kaynaklar

* Nesne Tabanlı Programlama Ders Dokümanı, Öğr. Gör. Gökhan TURAN, 2024
* Görsel Programlama Ders Dokümanı, Öğr. Gör. Gökhan TURAN, 2024