T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

NESNE YÖNELİMLİ ANALİZ VE TASARIM PROJE RAPORU

NESNELERİN İNTERNETİ SİSTEMLERİ İÇİN AKILLI CİHAZ TASARIMI

G191210065 - Mehmetali Demirtaş - 2/B

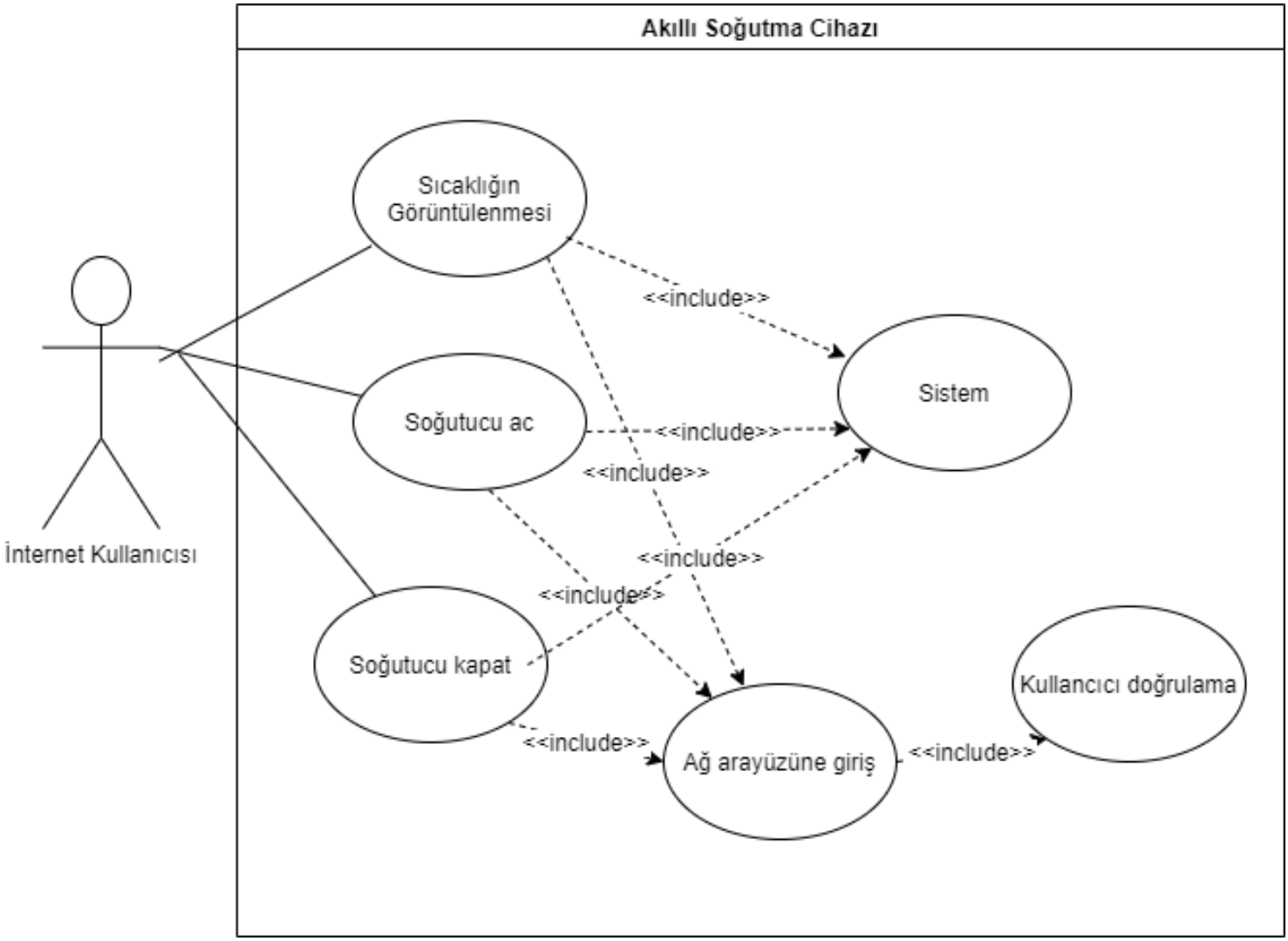
G191210008 – Ahmet Hakan Arı – 2/A

SAKARYA

Mayıs, 2022

Nesne Yönelimli Analiz ve Tasarım Dersi

A. “İnternet kullanıcısı” aktörü için kullanım durumu (Use Case) diyagramı.



B. İnternet üzerinden "Sıcaklığın görüntülenmesi" ve "soğutucunun çalıştırılması" kullanım durumlarına ait metinsel tanımlar

SICAKLIĞIN GÖRÜNTÜLENMESİ:

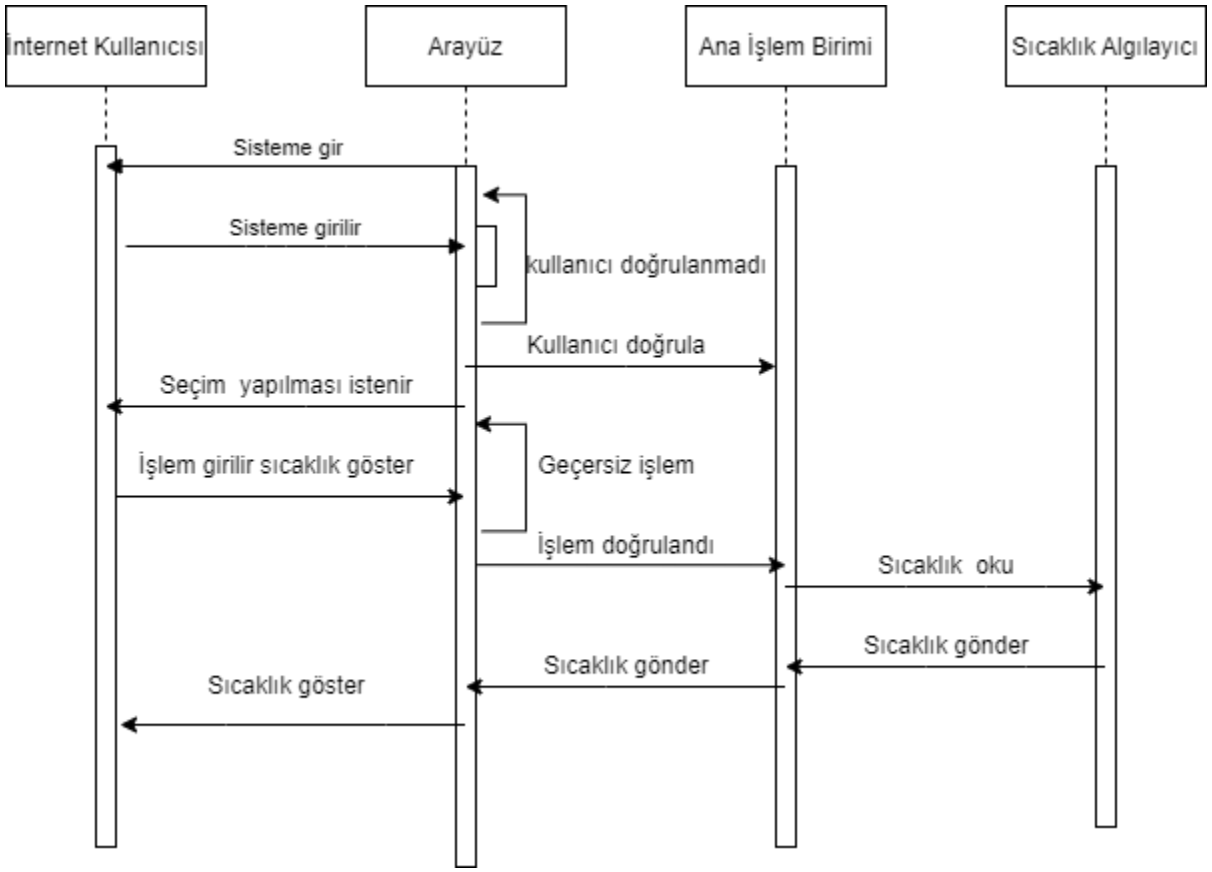
- Kullanım durumunun tanımı:
  - Sıcaklık Görüntüleme
  - İnternet üzerinden "sıcaklığın görüntülenmesi" işlemini tanımlar.
  - 07.05.2022, 10.07.2022 v1.1.1 kullaniciadi1
- İlgili aktörler: İnternet Kullanıcısı
- Giriş koşulu: İnternet kullanıcısının ağ arayüzüne internet üzerinden erişmesi.
- Çıkış koşulu: İnternet kullanıcısı çıkış işlemini seçer.
- Ana Olay Akışı: "Sıcaklık Görüntüle"
  1. İnternet kullanıcısı ağ arayüzünü açar.
  2. Ağ arayüzü giriş ekranında kullanıcıdan kullanıcı adı ve şifresini ister.
  3. İnternet kullanıcısı kendi kullanıcı adı ve şifresini girer.
  4. Girilen kullanıcı bilgileri veritabanında doğrulanır.
  5. Arayüz üzerinden kullanıcıya işlem menüsü açılır ve bir işlem seçmesi istenir.
  6. Kullanıcı sıcaklık görüntüle seçeneğini seçer.
  7. Kullanıcının isteği merkezi işlem birimi üzerinden sıcaklık algılayıcıya bildirilir.
  8. Sıcaklık görüntülenir ve adım 5 e dönülür.
- 1.Alternatif Olay Akışı: "Kullanıcı bilgileri doğrulanmadı"
  4. Kullanıcının bilgileri doğrulanmadı.
  5. Bilgilerin doğrulanmadığı ekrana yazdırılır.
  6. Kullanıcıdan tekrar bilgilerini girmesi istenir ve adım 2 ye dönülür.
- 2.Alternatif olay akışı: "Kullanıcı çıkış seçeneğini seçti"
  6. Kullanıcı çıkış seçeneğini seçer.
  7. Oturum sonlandırılır.

SOĞUTUCUNUN ÇALIŞTIRILMASI:

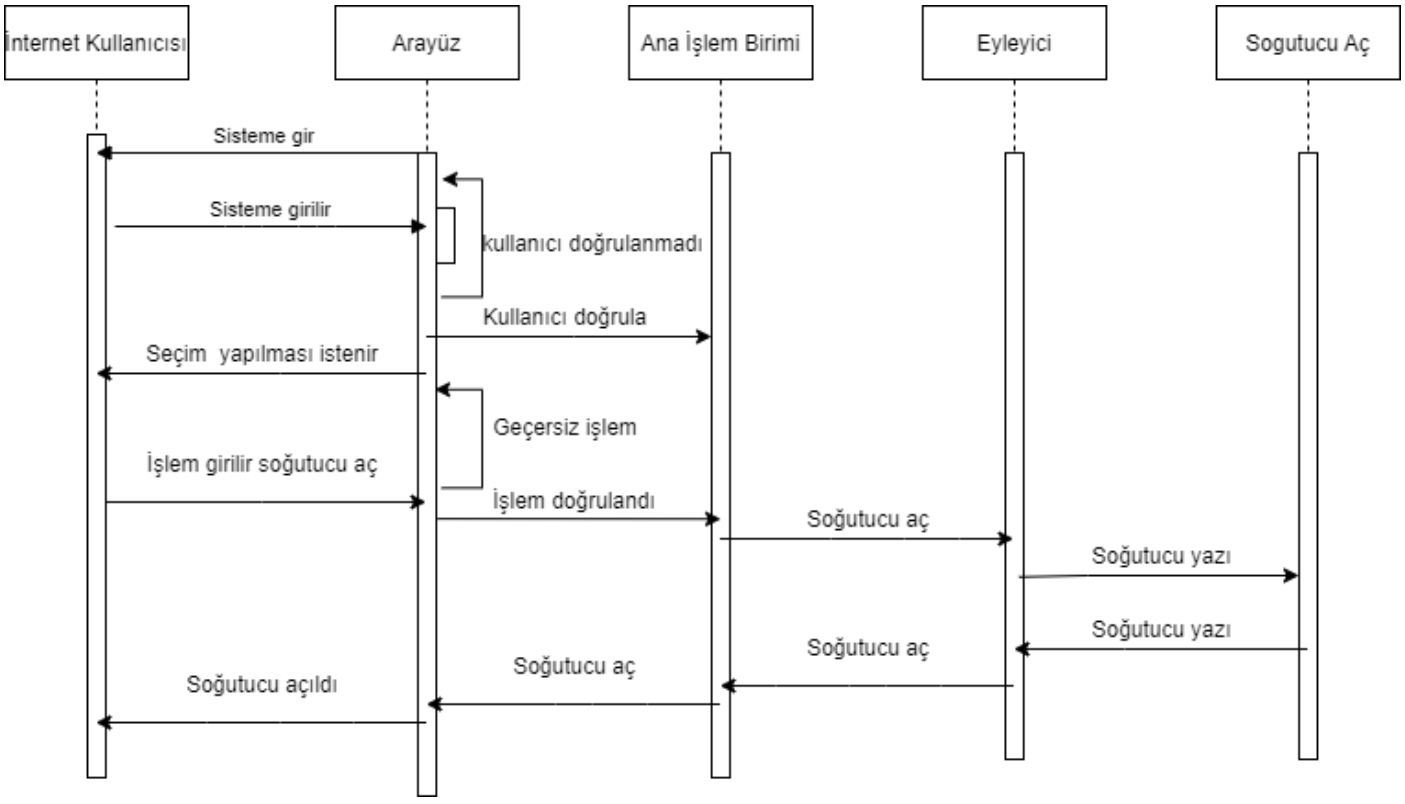
- Kullanım durumunun tanımı:
    - Soğutucu çalıştırma
    - İnternet üzerinden “soğutucunun çalıştırılması” işlemini tanımlar.
    - 07.05.2022, 10.07.2022 v1.1.1 kullaniciadi1
- İlgili aktörler: İnternet Kullanıcısı
- Giriş koşulu: İnternet kullanıcısının ağ arayüzüne internet üzerinden erişmesi.
- Çıkış koşulu: İnternet kullanıcısı çıkış işlemini seçer.
- Ana Olay Akışı: "Soğutucu Aç"
    1. İnternet kullanıcısı ağ arayüzünü açar.
    2. Ağ arayüzü giriş ekranında kullanıcıdan kullanıcı adı ve şifresini ister.
    3. İnternet kullanıcısı kendi kullanıcı adı ve şifresini girer.
    4. Girilen kullanıcı bilgileri veritabanında doğrulanır.
    5. Arayüz üzerinden kullanıcıya işlem menüsü açılır ve bir işlem seçmesi istenir.
    6. Kullanıcı soğutucu açma seçeneğini seçer.
    7. Kullanıcının isteği merkezi işlem birimi üzerinden eyleyiciye bildirilir.
    8. Soğutucu zaten açık değilse eyleyici soğutucuyu açar.
- 1.Alternatif Olay Akışı: “Kullanıcı bilgileri doğrulanmadı”
    4. Kullanıcının bilgileri doğrulanmadı.
    5. Bilgilerin doğrulanmadığı ekrana yazdırılır.
    6. Kullanıcıdan tekrar bilgilerini girmesi istenir ve adım 2 ye dönülür.
- 2.Alternatif olay akışı: “Soğutucu zaten açık”
    9. Soğutucu zaten açık ise soğutucu açılmaz.
    10. Soğutucunun zaten açık olduğu ekrana yazdırılır ve adım 5 e dönülür.
- 3.Alternatif olay akışı: “Kullanıcı çıkış seçeneğini seçti”
    6. Kullanıcı çıkış seçeneğini seçer.
    7. Oturum sonlandırılır.

C. İnternet üzerinden “Sıcaklığın görüntülenmesi” ve “soğutucunun çalıştırılması” kullanım durumlarına ait sıralama şemaları (sequence diagram)

- Sıcaklık Göster Sequence

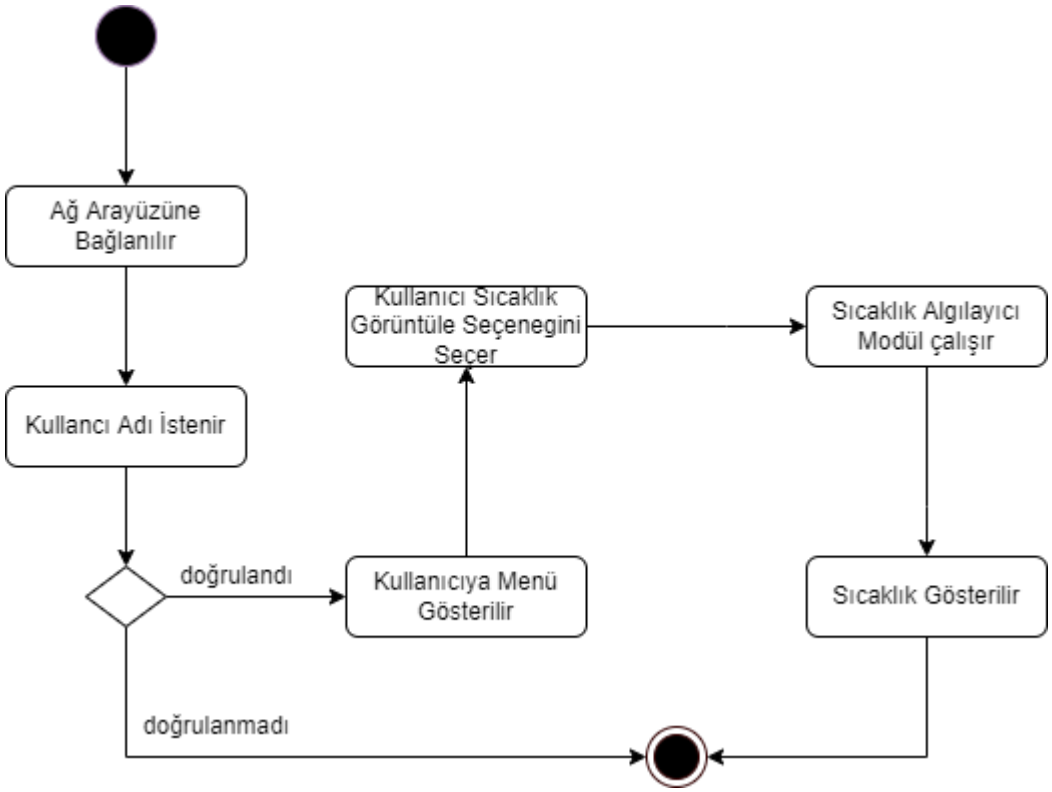


- Soğutucu Aç Sequence

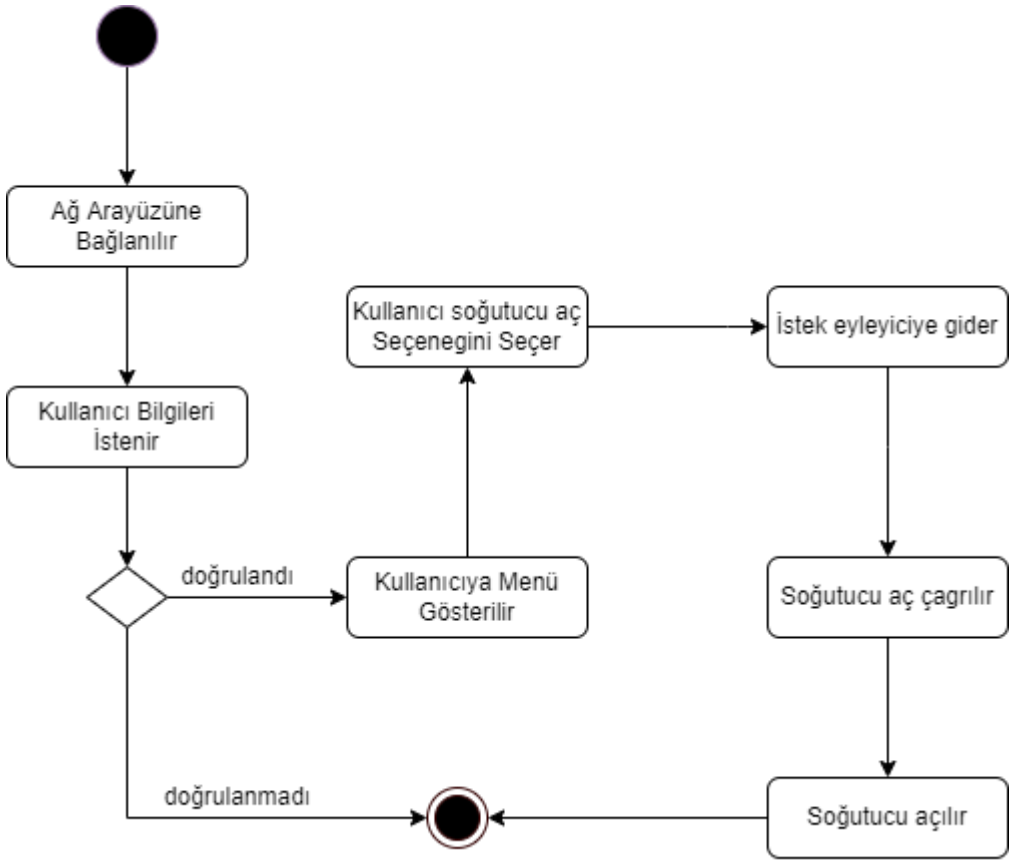


D. İnternet üzerinden “sıcaklığın görüntülenmesi” ve “soğutucunun çalıştırılması” kullanım durumlarına ait etkinlik şemaları (activity diagram).

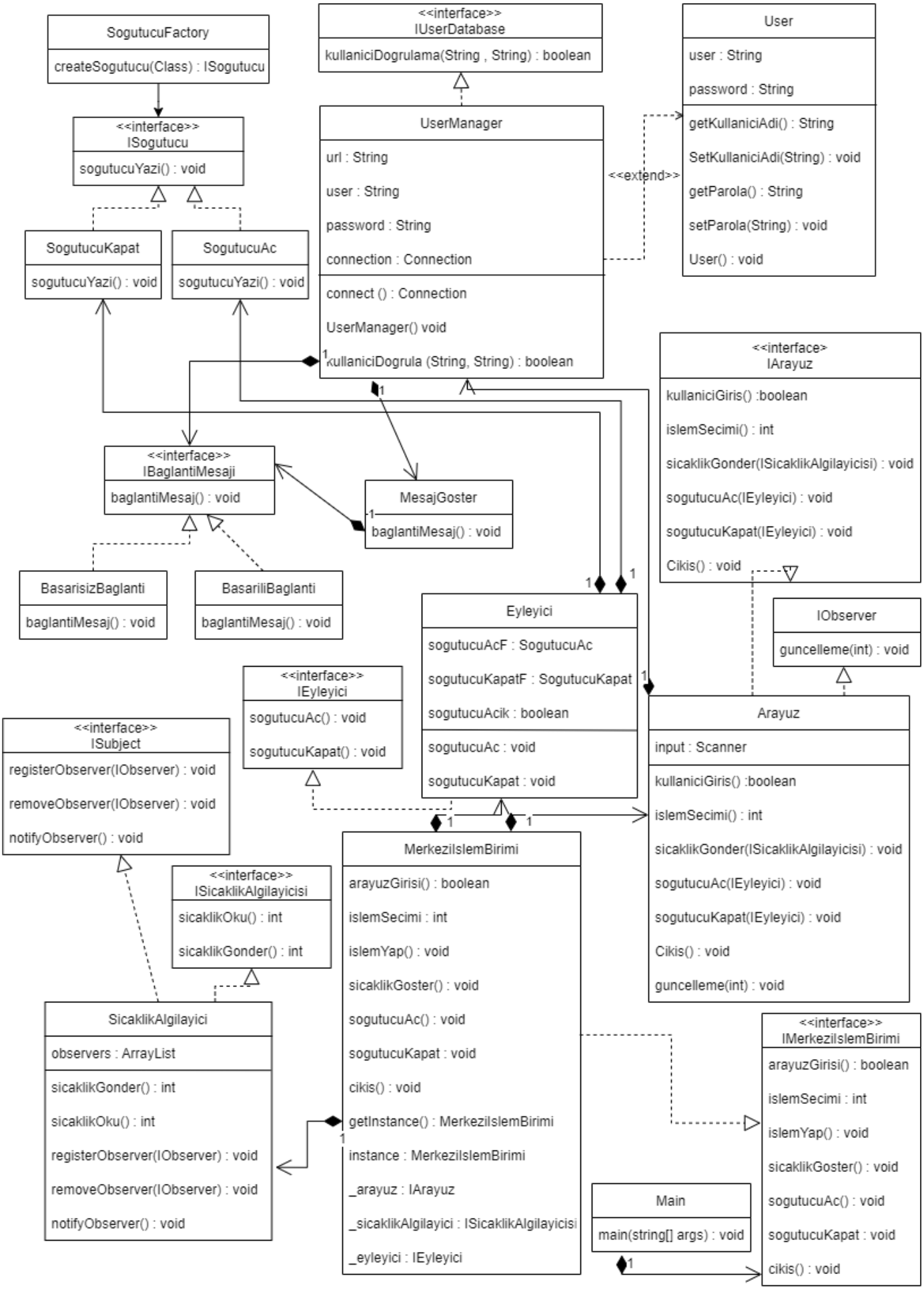
- Sıcaklığın Görüntülenmesi



- Soğutucunun Çalıştırılması



E. Geliştireceğiniz sistemin sınıf şeması

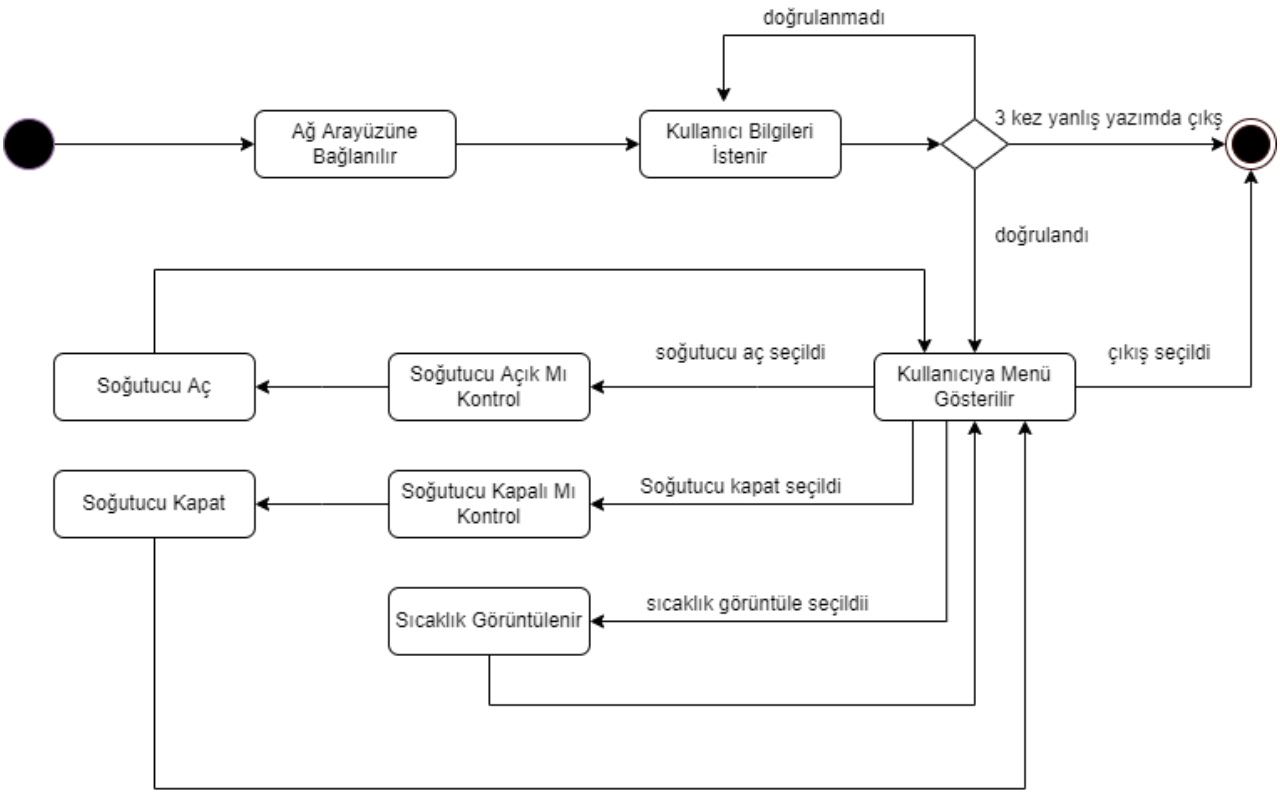


F. CRC Kartları

Arayüz(Kullanıcı ile etkileşime giren sınıf)	
Sorumluluk	**İŞBİRLİĞİ YAPILAN SINIF**
Kullanıcıdan giriş bilgileri ve menü seçim istenmesi	kendisi
Kullanıcı bilgilerini alma	user(SetKullaniciAdi,SetParola,k)
Kullanıcı doğrulama (kullanıcıdoğrulama())	usermanager
Ana menü ve gösterimi	kendisi
eyleyici işlem yapmak	Eyleyici (SoğutucuAÇ, SoğutucuKapa,)
sıcaklık göndermek	Sıcaklık Algılayıcı
çıkış ve güncelleme yazdırmak	kendisi

UserManager(kullanıcı doğrulama işlemlerini gerçekleştirir)	
Sorumluluk	**İŞBİRLİĞİ YAPILAN SINIF**
Veritabanına bağlanma	kendisi
Başarılı Mesajı yazdır	başarılıbaglantı
Kullanıcı doğrulama işlemleri	kendisi
Başarısız mesaj yazdır	başarısızbağlantı

G. Sistemin durum makinası diyagramı



H. Kullanıcı doğrulama ekran görüntüsü

```
KULLANICI GİRİŞ ARAYÜZÜ
Oturum açmak için lütfen kullanıcı adınızı ve parolanızı giriniz
Kullanıcı Adınız: admin1
Parolanız: 12345
PostgreSQL veritabanına başarılı birşekilde bağlantı kuruldu.
Arayüze giriş yapamadınız...!

KULLANICI GİRİŞ ARAYÜZÜ
Oturum açmak için lütfen kullanıcı adınızı ve parolanızı giriniz
Kullanıcı Adınız: admin1
Parolanız: 123456
PostgreSQL veritabanına başarılı birşekilde bağlantı kuruldu.
Arayüze Hoşgeldiniz

İŞLEM MENÜSÜ
1.Sıcaklık Göster
2.Soğutucuyu Aç
3.Soğutucuyu Kapat
4.Çıkış

Yapmak istediğiniz işlemi seçiniz:
```

İ. Sıcaklığın görüntülenmesi ve soğutucunun açılıp kapatılmasıyla ilgili ekran görüntüleri

```
İŞLEM MENÜSÜ
1.Sıcaklık Göster
2.Soğutucuyu Aç
3.Soğutucuyu Kapat
4.Çıkış

Yapmak istediğiniz işlemi seçiniz:
1
Seçtiğiniz İşlem Numarası...: 1
Sıcaklık değeri 13 derecedir.


İŞLEM MENÜSÜ
1.Sıcaklık Göster
2.Soğutucuyu Aç
3.Soğutucuyu Kapat
4.Çıkış

Yapmak istediğiniz işlemi seçiniz:
2
Seçtiğiniz İşlem Numarası...: 2
SOGUTUCU AÇILDI
```

J. Veritabanının görüntüsü (kullanıcı verilerinin saklandığı tablonun, verileri içeren görüntüsü)

akillicihaz/postgres@PostgreSQL 14

Query Editor | Query History

```
1  SELECT * FROM public."User"
2  ORDER BY "userID" ASC
```

Data Output | Explain | Messages | Notifications

	userID [PK] integer	username character varying	password character varying
1	1	admin1	123456
2	2	admin2	123456
3	3	kullanici1	12345
4	4	kullanici2	12345

K. “Dependency Inversion” ilkesinin ne olduğu ve uygulama içerisinde nasıl gerçeklendiği

Yüksek seviye bir sınıfın alt seviye sınıflara olan bağımlılığını ortadan kaldırarak soyut katman üzerinden işlemleri yapmaya dependency inversion denir.

Yani, düşük seviyeli sınıflarda yapılan davranış değişikliği, üst sınıfta bozulma meydana getirmemesi için ikisi arasınnda soyutlama katmanı kullanılmalıdır.

Bu projede ise, veritabanı bağlantısı sonucunda ekrana gerekli yazının yazılması için IBaglantiMesaji adında bir arayüz oluşturduk. Bu arayüzü implement eden BasariliBaglanti ve BasarisizBaglanti adinda iki sınıf oluşturduk. Daha sonra IBaglantiMesaji arayüzünden türetilen nesneyi kullanarak “MesajGoster” adında bir fonksiyon tanımladık. UserManager sınıfında, bağlantı sonucuna göre ekrana yazı yazdırmak için IBaglantiMesaji arayüzü üzerinden BasariliBaglanti ve BasarisizBaglanti nesneleri oluşturup, bu nesneleri MesajGoster sınıfına parametre olarak verdik ve ekrana yazma işlemini gerçekleştirdik.

IBaglantiMesaji Arayüzü:

```
package DependencyInversion;

public interface IBaglantiMesaji {
    public void baglantiMesaj();
}
```

BasariliBaglanti Sınıfı:

```
package DependencyInversion;

public class BasariliBaglanti implements IBaglantiMesaji{
    @Override
    public void baglantiMesaj() {
        System.out.println("Veritabanına başarılı bir şekilde bağlantı kuruldu...");
    }
}
```

BasarisizBaglanti Sınıfı:

```
package DependencyInversion;

public class BasarisizBaglanti implements IBaglantiMesaji{
    @Override
    public void baglantiMesaj() { System.out.println("Veritabanına bağlantı başarısız!!!"); }
}
```

MesajGoster Sınıfı:

```
package DependencyInversion;

public class MesajGoster {
    private  IBaglantiMesaji mesaj;
    public MesajGoster(IBaglantiMesaji mesaj){
        this.mesaj=mesaj;
    }

    public void mesajGoster(){
        mesaj.baglantiMesaj();
    }
}
```

UserManager Sınıfı:

```
IBaglantiMesaji basariliMesaji = new BasariliBaglanti();
IBaglantiMesaji basarisizMesaji = new BasarisizBaglanti();
MesajGoster basariliMesajGoster = new MesajGoster(basariliMesaji);
MesajGoster basarisizMesajGoster = new MesajGoster(basarisizMesaji);
```

```
public Connection connect() {
    Connection connection = null;
    try {
        connection = DriverManager.getConnection(url, user, password);
        basariliMesajGoster.mesajGoster();
    }
```

```
        else {
            basarisizMesajGoster.mesajGoster();
            return false;
        }
```

L. “Factory Method” ve “Observer” desenlerinin ne olduğu ve uygulama içerisinde nasıl gerçeklendiği.

- ### FACTORY METHOD DESENİ

Üst sınıfta nesneler oluşturma işleminde factory sınıfını arabirim olarak kullanma işlemine denir.

Yani; ilk olarak bir interface oluşturulur ve bu interface’yi implement eden sınıflar oluşturulur. Üst sınıfta bu oluşturulan sınıflardan nesne oluşturma işleminde direkt ilgili sınıfları çağırmak yerine, bu görevi bir fabrika sınıfına vererek nesne oluşturma işlemi soyutlaştırılmaktadır.

Projede ise “Sogutucu açıldı” ve “Sogutucu kapatıldı” çıktılarını vermek için ISogutucu adında bir arayüz tasarladık. Bu arayüzü implement eden SogutucuAc ve SogutucuKapat sınıflarını oluşturduk. Daha sonra SogutucuFactory sınıfımızı oluşturduk. Eyleyici sınıfı içerisinde ise SogutucuFactory ile gerekli sınıflardan nesneleri oluşturduk ve çıktı verirken bu nesneleri kullandık.

ISogutucu Arayüzü:

```
package Factory;

public interface ISogutucu {
    public void sogutucuYazi();
}
```

SogutucuAc Sınıfı:

```
package Factory;

public class SogutucuAc implements ISogutucu{
    public void sogutucuYazi() { System.out.println("SOGUTUCU AÇILDI"); }
}
```

SogutucuKapat Sınıfı:

```
package Factory;

public class SogutucuKapat implements ISogutucu{
    @Override
    public void sogutucuYazi() { System.out.println("SOGUTUCU KAPATILDI"); }
}
```

SogutucuFactory Sınıfı:

```
package Factory;

public class SogutucuFactory {
    public static  ISogutucu createSogutucu(Class aclass) throws IllegalAccessException, InstantiationException{
        return (ISogutucu) aclass.newInstance();
    }
}
```

Eyleyici Sınıfı:

```
public class Eyleyici implements IEyleyici {
    SogutucuAc sogutucuAcF = (SogutucuAc) SogutucuFactory.createSogutucu(SogutucuAc.class);
    SogutucuKapat sogutucuKapatF = (SogutucuKapat) SogutucuFactory.createSogutucu(SogutucuKapat.class);
```

```
        else {
            sogutucuAcF.sogutucuYazi();
            sogutucuAcik = true;
        }
    }
    @Override public void sogutucuKapat() {
        if(sogutucuAcik ==false) {
            System.out.println("Soğutucu zaten kapalı");
        }
        else{
            sogutucuKapatF.sogutucuYazi();
            sogutucuAcik =false;
        }
```

- ## OBSERVER DESENİ

Observer tasarım deseni, gözlemlenen nesnede gerçekleşen olaylarla ilgili bilgilendirmeyi sağlayan bir abonelik mekanizması oluşturmayı amaçlar.

Kısacası , elimizdeki mevcut nesnenin durumunda herhangi bir değişiklik olduğunda, bu değişiklerden diğer nesneleri haberdar eden bir tasarımdır. Subject nesnesi içerisinde herhangi bir özellik güncellendiğinde Notify metodu tetiklenecek ve bu metod Subject'e abone olan tüm Observer'ların Update metodunu çalıştıracaktır.

Bu projede ise, IObserver ve ISubject arayüzlerini tasarladıktan sonra, Sicaklik algilayicisi sınıfı ISubject arayüzünü implement ederek Publisher sınıfı olarak kullanıldı. Mesaj olarak kullanıcıya sıcaklık değeri gösterilmektedir.

IObserver Arayüzü:

```
package Observer;

public interface IObserver {
    public void guncelleme(int sicaklik);
}
```

ISubject Arayüzü:

```
package Observer;

public interface ISubject {
    public void registerObserver(IObserver o);
    public void removeObserver(IObserver o);
    public void notifyObservers();
}
```

SicaklikAlgilayici Sınıfı:

```
public class SicaklikAlgilayici implements ISicaklikAlgilayici, ISubject {
    private ArrayList observers = new ArrayList();
    @Override
    public int sicaklikOku() {
        Random random = new Random(); return random.nextInt( bound: 100);
    }
    @Override
    public int sicaklikGonder() {
        System.out.println("Sıcaklık değeri " + sicaklikOku() + " derecedir."); return sicaklikOku();
    }

    @Override
    public void registerObserver(IObserver o) {
        observers.add(o);
    }

    @Override
    public void removeObserver(IObserver o) {
        int i = observers.indexOf(o);
        if (i >= 0) {
            observers.remove(i);
        }
    }


    @Override
    public void notifyObservers() {
            for (int i = 0; i < observers.size(); i++) {
                IObserver observer = (IObserver) observers.get(i);
                observer.guncelleme(sicaklikGonder());
            }
    }
```

Arayüz Sınıfı:

```
@Override
public void guncelleme(int sicaklik) { //IObserver sınıfından guncelleme metodu
    System.out.println("Sıcaklık güncellendi.");
}
```

M.Uygulamanın kaynak kodları

https://github.com/mehmetalidemirtas/NesneYonelimliAnalizVeTasarimProjeOdevi