Raspberry Pi

**settings.py**

Ayar dosyası oluşturuldu.

Değişken bilgiler toplu olarak karışıklığı (bulamama sorununu) engellemek için bu dosyada tutulmaktadır.

DATABASE\_FILE\_NAME = "data/database.csv"

DATABASE\_INDEX\_NAME = 0

DATABASE\_INDEX\_EMAIL\_ADDRESS = 1

DATABASE\_INDEX\_PHONE\_NO = 2

WEBAPI\_URL = "http://10.201.1.187:37001/api/data"

WEBAPI\_URL\_ALARM = "http://10.201.1.187:37001/api/alarm"

WEBAPI\_URL\_ALARM\_DISABLE = "http://10.201.1.187:37001/api/disablealarm"

WEBAPI\_URL\_ALARM\_ENABLE = "http://10.201.1.187:37001/api/enablealarm"

EMAIL\_API\_STMP\_HOST = "smtp-mail.outlook.com"

EMAIL\_API\_SMTP\_PORT = 587

EMAIL\_API\_EMAIL\_ADDRESS = "muhammedtiftikci@outlook.com"

EMAIL\_API\_EMAIL\_PASSWORD = ""

SMS\_API\_USERNAME = ""

SMS\_API\_ORIGINATOR = ""

SMS\_API\_PASSWORD = ""

**utils.py**

Yardımcı işlemlerin tanımlanacağı dosyadır.

Gerekli paket komut satırına ‘pip install requests’ yazılarak kurulum yapıldı.

Bu paket ile http istekleri yapılabilmektedir. Http isteği sms gönderiminde kullanılmaktadır.

Bu modülde aşağıdaki fonksiyonlar tanımlanmıştır.

**Sms gönderimi:**

<http://www.mutlucell.com/api/index.html?hm_xmlin_formati_nasil_olmalidir_.htm> adresinden gönderilmesi gereken xml yapısı incelendi.

<http://www.mutlucell.com/api/index.html?hm_xml_kodunu_post_etme.htm> adresinden gönderilmesi gereken adres ve yapısı incelendi.

Daha sonra python kodu aşağıdaki gibi python kodu tasarlandı.

def send\_sms\_mutlucell(username, password, originator, message, numbers):

url = "https://smsgw.mutlucell.com/smsgw-ws/sndblkex"

xml = '''<?xml version="1.0" encoding="UTF-8"?>

<smspack ka="{}" pwd="{}" org="{}">

<mesaj>

<metin>{}</metin>

<nums>{}</nums>

</mesaj>

</smspack>'''.format(username, password, originator, message, ",".join(numbers))

headers = { 'Content-Type': 'text/xml; charset=utf-8' }

response = requests.post(url, data=xml, headers=headers)

return response.text

**Email gönderimi:**

Email gönderimi araştırılarak aşağıdaki fonksiyon oluşturuldu.

def send\_mail(host, port, from\_addr, password, to\_addr, subject, body):

server = smtplib.SMTP(host, port)

server.ehlo()

server.starttls()

server.login(from\_addr, password)

server.sendmail(from\_addr, to\_addr, "Subject: {}\n\n{}".format(subject, body))

server.quit()

**Kullanıcı bilgilerinin web servis üzerinden alınması:**

İleride anlatılacak olan yazdığım web servis üzerinden sms ve email gönderilecek olan kişi bilgilerin alınması için bir fonksiyon oluşturuldu.

Bu fonksiyon verilen url üzerine GET türünde istek yaparak, aldığı veriyi 3 kolonu bulunan 2 boyutlu bir diziye çevirerek döndürür.

GET türü: Http isteklerinde belirli istekler kullanılmaktadır. Web servislerde genel olarak GET, POST, PUT ve DELETE türleri kullanılmaktadır. Veri almak için kullanılan GET ve POST türünden biri olan GET kullanılmıştır.

Fonksiyon aşağıdaki gibidir:

def get\_data\_from\_api(url):

response = requests.get(url)

print(response.json())

return [[x["adSoyad"].encode("utf-8"), x["email"], x["telefon"]] for x in response.json()]

**speechrecognition.py**

Bu modülün amacı mikrofonu dinleyerek algıladığı sesi yazıya çevirip döndüren bir fonksiyon yazılmasıdır.

Sesin yazıya çevrilmesi için SpeechRegonition çatısı kullanılmıştır. Modül kurulumu için aşağıdaki komutlar, komut satırında çalıştırılarak paketler yüklenmiştir.

pip install speechrecognition

pip install pocketsphinx

pip install pyaudio

Daha sonra dökümanından incelenen kod aşağıdaki gibi bir fonksiyona atanmıştır:

import speech\_recognition as sr

r = sr.Recognizer()

with sr.Microphone() as mic:

print("Say something:")

audio = r.listen(mic)

print("Time over, thanks.")

try:

print("Text: " + r.recognize\_sphinx(audio))

except Exception as ex:

print(ex.message)

**main.py**

Ana çalıştırılacak olan dosyadır.

İçerisinde yapılan işlemler şu şekilde sıralanabilir:

* Bilgi verilecek olan kişi bilgilerini web servis üzerinden al
* Sonsuz bir döngü oluştur:
  + Mikrofon ile ortamı dinle
  + Dinlenen sesleri boşluk karakterine göre parçala
  + Parçalardan herhangi biri ‘imdat’ ise:
    - Web servise alarmın aktif olduğunu bildir
    - Tüm kullanıcılara sms ve email gönder
    - Web servisten alarm kapalı bilgisi alınana kadar alarmı aktif et

Kodlar ise şu şekildedir:

…

**Web Api**

<https://dotnet.microsoft.com/download> adresinden ‘dotnet core’ programı indirildi ve kurulum yapıldı.

Yeni bir web api projesi oluşturmak için

dotnet new webapi -o .\src\WebApi

komutu kullanıldı.

Proje klasörü içine model sınıflarının bulunacağı Models klasörü eklendi.

**Bilgi verilecek kişilerin listesi için:**

Data adında bir sınıf Models klasörünün içerisine eklendi. Data sınıfının AdSoyad, Email ve Telefon özellikleri string türünden tanımlandı.

public class Data

{

public string AdSoyad { get; set; }

public string Email { get; set; }

public string Telefon { get; set; }

}

Yeni bir DataController adında sınıf Controller klasörüne eklendi. **Bu sınıf istekleri karşılamaktadır.**

İlgili Get metodu aşağıdaki gibi yazıldı:

[Route("api/[controller]")]

[ApiController]

public class DataController : ControllerBase

{

[HttpGet]

public ActionResult<IEnumerable<Data>> Get()

{

List<Data> data = new List<Data>()

{

new Data() {

AdSoyad = "Muhamemd Emin TİFTİKÇİ",

Telefon = "+90 506 140 92 83",

Email = "muhammedtiftikci@outlook.com"

},

new Data() {

AdSoyad = "Keml AKYOL",

Telefon = "+90 530 341 1309",

Email = "kemalakyol48@gmail.com"

},

};

return data;

}

}

Alarm değerini tutan boolean türünde static bir değişken oluşturuldu.

**Alarm durumunun alınması için:**

AlarmController sınıfı oluşturuldu.

[Route("api/[controller]")]

[ApiController]

public class AlarmController : ControllerBase

{

[HttpGet]

public bool Get()

{

return Program.ALARM\_IS\_ACTIVE;

}

}

**Alarmın aktif edilmesi için:**

EnableAlarmController sınıfı oluşturuldu.

[Route("api/[controller]")]

[ApiController]

public class EnableAlarmController : ControllerBase

{

[HttpGet]

public ActionResult Get()

{

Program.ALARM\_IS\_ACTIVE = true;

return Ok();

}

}

**Alarmin deaktif edilmesi için:**

DisableAlarmController sınıfı oluşturuldu.

[Route("api/[controller]")]

[ApiController]

public class DisableAlarmController : ControllerBase

{

[HttpGet]

public ActionResult Get()

{

Program.ALARM\_IS\_ACTIVE = false;

return Ok();

}

}

Projenin port adresi Program.cs dosyasından ilgili metoda aşağıdaki gibi eklendi:

public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>

WebHost.CreateDefaultBuilder(args)

.UseStartup<Startup>()

.UseUrls("http://10.201.1.187:37001");

Projenin çalışması için

dotnet run -p .\src\WebApi\WebApi.csproj

komutu çalıştırıldı.

<http://192.168.1.7:37001/api/data> adresi kontrol edilerek sistem test edildi.

**Mobil**

nodejs programı <https://nodejs.org> adresinden indirildi ve kurulum yapıldı.

React Native uygulama geliştirmek için tasarlanmış yardımcı araç olan Expo eklentisi nodejs package manager (npm) ile

npm install -g expo-cli

komutu kullanılarak yüklendi.

Yeni bir React Native projesi oluşturmak için

expi init .\src\Mobile

komutu kullanıldı.

App.js dosyasının render metodu aşağıdaki gibi değiştirilerek alarmı kapatması için bir buton eklendi.

render() {

return (

<View style={styles.container}>

<TouchableHighlight onPress={this.onPress}>

<Text>Alarmı Kapat</Text>

</TouchableHighlight>

</View>

);

}

Üstteki belirtilen onPress metodu yazıldı. Bu metot ilgili adreese istek göndererek alarmın kapanmasını sağlıyor.

onPress() {

fetch("http://127.0.0.1:37001/api/disablealarm", { method: 'GET' })

.then((response) => console.log(response))

.catch((error) => console.log(error));

}