

**Proje Adı:**

# Duyuru Panosu

**Projeyi Yapanlar:**

Ali KAZAR – 460115007

Burak AKÇA – 460116823

Muratcan ÜNSAL – 460115505

**Projenin Amacı:** Kullanıcı anlık olarak Raspberry Pi üzerinden duyuru paylaşımı yapabilmesini sağlamaktır.

**Projede Kullanılan Malzemeler:**

- 1 Raspberry Pi 3
- 2 Prolink HDMI A to VGA+3.5mm ST Socket
- 3 Soğutucu
- 4 5V 2A güç kablosu.
- 5 16 gb SD Card
- 6 Ekran

**Proje Hakkında Genel Açıklama**

Öğrencilerin duyuruları okul içinde daha rahat takip edebilmeleri için ve öğrenci işlerindeki yükü azaltmak sürekli sorulan soruları bu duyuru takip sistemi üzerinden öğrencilere sunmak için duyuru takip sistemi yapmaya karar verdik

-Duyuru paylaşımı genel duyuruların yanında o bölüme ve öğrenci işlerine dair duyurularıda içerebilir

Python diliyle yazılmış olan sunucu, olay tabanlı çalışmaktadır. Bu sayede Raspberry Pi 3 üzerindeki arayüzden sunucuya sürekli istek(request) gönderilmez. Yeni veri geldiği zaman otomatik olarak arayüzdeki dönen duyurulara eklenir.

C# diliyle yazdığımız ve kullanıcının etkileşime geçeceği uygulamanın adı Duyuru Takip Sistemidir. Bu uygulamayla kullanıcı;

- Raspberry Pi üzerinde dönmekte olan duyurulara anında yeni duyuru ekleyebilir veya duyurulardan istediğini silebilir.
- Ayarlar sekmesini kullanarak dönmekte olan duyuruların görsel özelliklerini değiştirebilir.

Raspberry Pi üzerinde dönmekte olan '.html' uzantılı sayfada, html, css3, javascript gibi diller kullanılmıştır.

Projede SQLite veritabanı kullanılmıştır. Kullanılmasının sebebi hafif olmasına rağmen bir çok özelliğe sahip olmasıdır. Ayrıca Python dili içinde gömülü olarak gelen kütüphanesi vasıtasıyla SQLite veritabanıyla çok rahat bir şekilde bağlantı kurulabilmektedir.

Proje 3 ana kısımdan oluşmaktadır.

1. Windows işletim sistemli bilgisayar üzerinde çalışacak olan C# dilinde yazılmış kullanıcı uygulaması.
2. Raspberry Pi 3 üzerinde çalışacak olan Python diliyle yazılmış sunucu.
3. Raspberry Pi 3 üzerinde çalışacak olan browser da tüm duyuruların döndüğü arayüz ekranı.

## Kullanıcı Programında Kullanılan Önemli Kütüphaneler

1. System.Net.Sockets: C# dilinde gömülü olarak gelen bir kütüphane. 'TcpClient' sınıfını kullanarak sunucuya soket vasıtasıyla bağlanmayı sağlıyor.
2. Json.Net: C# diliyle yazılmış olan programda yarattığımız list, dictionary gibi veri tiplerini json formatına çevirmek için kullandığımız kütüphanedir. Bu kütüphane sayesinde server tarafıyla veri iletişiminde büyük bir kolaylık elde edildi.

## Sunucu Programında Kullanılan Önemli Kütüphaneler

1. Twisted: Python'da yazılmış ve açık kaynak kodlu MIT lisansı ile lisanslanmış bir olay odaklı ağ motorudur. Server kodunun temelini oluşturmaktadır.
2. Autobahn|Python: 'Web, Mobile, Internet of Things' için açık kaynak kodlu, gerçek zamanlı bir framework. Twisted Framework'ün de kullanır. Projede websocket'i kullanmamıza olanak sağladı.
3. Sqlite3: Python üzerinde gömülü olarak gelen bir kütüphanedir. Sqlite olan bir veritabanı üzerinde CRUD(Create, Read, Update, Delete) işlemlerinin yapılmasını sağlamaktadır.
4. Json: Python üzerinde gömülü olarak gelen bir kütüphanedir. Kullanıcı tarafından soketlerle gönderilen verinin Python veri tipine çevrilmesinde kullanılmaktadır.

## Kullanıcı Programının Çalışma Mantığı

Kullanıcının yapacağı 'planlanmış duyuru' ekleme işlemi üzerinden anlatılacaktır.

Kullanıcı, gerekli verileri girdikten sonra ekle butonuna bastıktan sonra aşağıdaki işlemler gerçekleşmektedir.

1-'duyuruEkleDict' adında bir 'Dictionary' oluşmakta ve bu Dictionary e kullanıcının girdiği verilerek ek olarak "komut": "ekle" verisi eklenmektedir. Bu veri server a gittiğinde yapılmak işlemi belirtmek için kullanılmıştır.

2- Oluşturulan Dictionary json formatına dönüştürülür.

3-'requestGonderAl' methodu kullanılarak raspberry pi üzerinde çalışmakta olan server a gönderilir.

'requestGonderAl' methodu tüm C# kodunun temelini oluşturmaktadır.

```
private void duyuruEkle()
{
    string duyuru = duyuruTextBox.Text;
    string baslik = duyuruBaslikTb.Text;
    string giris_tarih;
    string duyuru_tur;
    if (aktifRadioButton.Checked == true)
    {
        duyuru_tur = "aktif";
        giris_tarih = DateTime.Now.ToShortDateString();
    }
    else
    {
        duyuru_tur = "plan";
        giris_tarih = girisDateTimePicker.Value.ToShortDateString();
    }
    string cikis_tarih = planDateTimePicker.Value.ToShortDateString();
    Dictionary<string, string> duyuruEkleDict = new Dictionary<string, string>();
    duyuruEkleDict.Add("komut", "ekle");
    duyuruEkleDict.Add("kullanici", kullanici);
    duyuruEkleDict.Add("baslik", baslik);
    duyuruEkleDict.Add("duyuru", duyuruTextBox.Text);
    duyuruEkleDict.Add("giris_tarih", giris_tarih);
    duyuruEkleDict.Add("cikis_tarih", cikis_tarih);
    duyuruEkleDict.Add("duyuru_tur", duyuru_tur);
    string output = JsonConvert.SerializeObject(duyuruEkleDict);
    requestGonderAl(output);
}
5 references
private string requestGonderAl(string strToSend)
{
    //veri gönderiliyor...

    NetworkStream serverStream = clientSocket.GetStream();
    byte[] outputStream = Encoding.UTF8.GetBytes(strToSend);
    serverStream.Write(outputStream, 0, outputStream.Length);
    serverStream.Flush();

    //response alınıyor...
    byte[] inputStream = new byte[4096];
    int bytesRead = serverStream.Read(inputStream, 0, inputStream.Length);
    string returnData = Encoding.ASCII.GetString(inputStream, 0, bytesRead);
    Console.WriteLine(returnData);
    return returnData;
}
```

Göndereceğimiz veri (strToSend) UTF-8 formatında byte a çevriliyor. Daha sonra oluşturmuş olduğumuz NetworkStream nesnesi üzerinden yazılıyor(büyüküğü, başlayacağı index değeri belli) Daha Sonra '.flush()' methodu kullanılarak stream üzerindeki bu veriler temizleniyor Geri gelen veri(response) için byte array tanımlanıyor. NetworkStream sınıfının 'read()' methodu kullanılarak stream üzerinden veri byte olarak alınıyor. En sonunda bu byte veri 'string' e dönüştürölüyor.

'requestGonderAI' methodu bu 'string' değerini döndürüyor.

4-Server gelen veriyi byte olarak alır ve önce UTF-8 formatında 'string' veri tipine çevririr. Daha sonra Json Kütüphanesinin bir methodu olan 'json.loads(veri)' kullanılarak Python veri tipine çevrilir.

Örneğin C# da bir dictionary olarak kullanılan veri, gerekli işlemler yapıldıktan sonra byte olarak server a gönderildi. Server daki işlemlerden sonra biz bu veriyi Python'un Dictionary veri tipinde kullanabileceğiz.

5- Tüm işlemler bittikten sonra elimizde C# tarafında oluşturmuş olduğumuz verileri içeren bir Dictionary bulunmaktadır. 'komut' kısmı kontrol edilir. 'ekle' işlemi yapılacaktır.

6- Gerekli method çağrılarak veritabanına yeni veri ekleme işlemi gerçekleşir.

7- Tüm bu işlemlerde bir hata çıkmazsa C# programına 'onay' mesajı gönderilir.

8- Ekleme işlemi gerçekleştikten sonra, server ın başka bir methodu kullanılarak, Raspberry Pi üzerinde çalışmakta olan ve servera websocket ile bağlanmış 'client.html' adlı sayfaya "kontrol" uyarısını içeren bir mesaj gönderilir. Bu mesajı alan client.html server a bir başka istek göndererek tüm aktif duyuru listesini tekrar ister. Böylece yeni eklenen duyuru anında client.html üzerinde belirmiş olur.

Yukarda yapılan işlemler, diğer tüm 'komut' işlemlerinde benzer şekilde gerçekleştiği için ek olarak bir başka komuta değinilmeyecektir.

### **Sunucunun Çalışma Mantığı**

'server.py' scripti çalıştırıldığında iki farklı port dinlenmeye başlanır. Bu portlardan biri kullanıcı uygulamasını dinleyen '9000' portu diğeri ise Raspberry Pi üzerindeki görsel arayüz sayfasından gelecek olan websocketin bağlanabileceği '9001' portu.

Herhangi biri bağlandığı zaman port numarasına ait Protocol sınıfının nesnesi oluşturulur.

Kullanıcı uygulaması için protokolün adı 'CsharpProtocol' , Duyuru Arayüzü için protokolün adı 'ClientHtmlProtocol' dür.

Daha detaylı bir anlatım için aşağıdaki main methodunu inceleyelim.

Öncelikle gerekli olan kütüphaneler scripte eklenir.

```

def main():

    import sys
    from twisted.python import log
    from twisted.internet import reactor

    log.startLogging(sys.stdout)

    csharpFactory = BroadcastServerFactory(u"ws://127.0.0.1:9000")
    csharpFactory.protocol = CsharpProtocol

    clientHtmlFactory = BroadcastServerFactory(u"ws://127.0.0.1:9001")
    clientHtmlFactory.protocol = ClientHtmlProtocol
    reactor.listenTCP(9000, csharpFactory)
    reactor.listenTCP(9001, clientHtmlFactory)
    reactor.run()

if __name__ == '__main__':
    main()

```

Hata ayıklamada yardımcı olması için 'log.startLogging(sys.stdout)' komutuyla loglama işlemi başlatılır.

Daha sonra dinlemek istediğimiz portlar için Factory nesneleri oluşturulur. Oluşturmuş olduğumuz bu factory lere ise protokollerini belirtiriz.

'reactor' ise tüm 'döngü' yü gerçekleştirir. listenTcp methodu 'reactor' e belli bir port numarasına ve belli bir protokole gelen bağlantıları işlemesini söyler.

'reactor.run' dedikten sonra birden fazla port numarasına gelecek olan bağlantıları dinleyen sunucu başlatılmış olur.

Aşağıda kodunun sadece bir parçası verilen BroadcastServerFactory sınıfına değinilecektir.

```

class BroadcastServerFactory(WebSocketServerFactory):
    """
    Gelen tüm mesajları bağlı olan tüm clientlara yollayan Broadcast Server
    """

    clients = []
    def __init__(self, url):
        WebSocketServerFactory.__init__(self, url)
        #self.clients = []
        self.veriGüncelle()

    def register(self, client):...
    def unregister(self, client):...

    def broadcast(self, msg):
        print("broadcasting message '{}' ..".format(msg))
        for c in self.clients:
            c.sendMessage(msg.encode('utf8'))
            print("message sent to {}".format(c.peer))

    def veriGüncelle(self):...

    def duyuruGuncelSil(self, rowsList):...

    def duyuruGuncelAktif(self, rowsList):...

```

## **'Factory' Sınıfı**

'Factory' lerin temel görevi dinlenen port numarasına denk düşen Protocol sınıfının nesnesini oluşturmaktır. Bu nesneler her yeni bir bağlantıda oluşturulur ve bağlantı koptuğunda yok edilir. Ama kalıcı ayarlar 'Factory' sınıfının nesnesinde saklanmaya devam eder. Bu da server kapanıncaya kadar sürer.

Bizim yazdığımız 'Factory' sınıfının en dikkat çeken özelliği 'clients' adındaki liste niteliği ve 'broadcast' adındaki methodudur.

'9001' portuna gelen her bir bağlantı 'clients' listesine eklenmektedir. Ve ne zaman 'broadcast' methodu çağrıldığında methoda gelen veri bu clients listesinde tüm clientlara dağıtılır. Bu method CsharpProtocol ü içinde yeni veri eklendiği zaman veya veri silindiği zaman çağrılmaktadır. Bu yöntemle kullanıcı bir duyuru girdiğinde bağlantıya bağlı olarak duyuru anında arayüze gönderilir.

Bu özelliğe ek olarak sunucu her baştan başlatıldığında veritabanına kontrol ederek günü gelen duyuruları eklemesini veya silmesini sağlayan veriGuncelle methodu 'Factory' nin constructor ında çağrılmaktadır. Bu sayede güncelleme yapılmış olur.

## **'Protocol' Sınıfı**

'Protocol' sınıfının nesneleri ise, dinlenen porta her yeni bağlantı geldiğinde oluşturulur. Bizim yazdığımız iki protokolden 'ClientHtmlProtocol' raspberry üzerindeki arayüzden gelecek websocket bağlantısını beklerken diğeri 'CsharpProtocol' kullanıcı programından gelecek olan bağlantıyı beklemektedir.

'ClientHtmlProtocol' protokolü methodları sayesinde aktif duyuruları arayüze iletimini sağlar.

'CsharpProtocol' ise gelen duyuruları veritabanına kaydetme, silme, güncelleme gibi işlemlerden sorumludur.

Aşağıda 'BroadcastServerFactory' nin server ın her başlatıldığına çalışan duyuruları güncellenmesini sağlayan methodu verilmiştir.

```

class BroadcastServerFactory(WebSocketServerFactory):
    """
    Gelen tüm mesajları bağlı olan tüm clientlara yollayan Broadcast Server
    """

    clients = []

    def __init__(self, url):...

    def register(self, client):...
    def unregister(self, client):...

    def broadcast(self, msg):...

    def veriGüncelle(self):
        try:
            conn = sqlite3.connect('duyuruDB.db')
            c = conn.cursor()
            c.execute('select rowid, * from duyuru')
            rows = c.fetchall()
            aktifHaleGelecekDuyurular = []
            silinecekDuyurular = []
            for row in rows:
                giris_tarih = datetime.strptime(row[4], "%d.%m.%Y")
                cikis_tarih = datetime.strptime(row[5], "%d.%m.%Y")
                if row[6] == "plan":
                    if giris_tarih <= datetime.now():
                        aktifHaleGelecekDuyurular.append(row[0])
                        print("{}' aktif hale gelecek. rowId = {}".format(row[3], row[0]))
                    if datetime.now() > cikis_tarih:
                        silinecekDuyurular.append(row[0])
                        print("{}' silinecek. rowId = {}".format(row[3], row[0]))

            self.duyuruGüncelAktif(aktifHaleGelecekDuyurular)
            self.duyuruGüncelSil(silinecekDuyurular)
        except sqlite3.OperationalError:
            print("Operational Error:", sys.exc_info()[1])

    def duyuruGüncelSil(self, rowslist):...

```

## Raspberry Pi Arayüzünün Çalışma Mantığı

'html' uzantılı bir dosyadır. CSS kullanılarak renkler ve duyuruların konumları düzenlenmiştir. Javascript kullanılarak ise websocket bağlantısı sağlanmış ayrıca bir duyuruların belli bir süre aralığında değişmesini sağlayan bir 'slideshow' eklenmiştir.

Aşağıdaki kod parçası arayüze socket vasıtasıyla bir mesaj gönderildiğinde çalışmaktadır.



```

socket.onmessage = function (e) {

    if (typeof e.data == "string") {
        if (e.data == "kontrol")
        {
            sendText();
        }
        else
        {
            //else if haline getir...
            currentSlide = 0;
            clearInterval(myInterval);
            var aktif_uyurular = JSON.parse(e.data);
            //try
            var node = document.getElementById('duyurular');
            while (node.hasChildNodes()) {
                node.removeChild(node.firstChild);
            }
            console.log(aktif_uyurular);
            for (ind = 0; ind < aktif_uyurular.length; ++ind) {

                var element = document.createElement('p');
                if (ind == 0)
                {
                    element.className = "duyuru showing";
                }
                else
                {
                    var element = document.createElement('p');
                    element.className = "duyuru";
                }

                element.innerHTML = aktif_uyurular[ind];
                document.getElementById("duyurular").appendChild(element);
            }
            myInterval = setInterval(nextSlide, 5000);
        }
    }
}

```

Öncelikle gelen mesajın veri tipi kontrol edilmektedir. String ise bir sonraki kontrol e geçiliyor. Gelen mesaj 'kontrol' ise yeni bir duyuru eklendiği anlaşıyor ve sunucudan 'sendText' methodu kullanılarak canlı duyuruları göndermesini istiyor. Eğer gelen mesaj 'kontrol' değilse sırasıyla dönmekte olan slide'ın indexini 0 lıyor. Duyuru döndürmeyi durduruyor. Gelen tüm duyuruları sayfaya ekliyor. En sonunda ise tekrar duyuru döndürmeyi 'setInterval' methoduyla başlatıyor.

## YARDIM

### Raspberry Pi 3 IP Bulma

1. 'Ctrl+Esc' ya da varsa 'Başlat' tuşuyla menüyü açın.
2. 'Terminal' i açın.
3. 'ifconfig' yazıp enter a basın.
4. Aşağıda ekran görüntüsünde yuvarlak içine alınmış olan 'inet addr' in karşılığı ip adresiyle windows makinanızdan sunucuya bağlanabilirsiniz.

```
pi@raspberrypi:~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:cd:ab:09
          inet6 addr: fe80::3f83:10fe:ef0f:4239/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:200 errors:0 dropped:0 overruns:0 frame:0
          TX packets:200 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:16656 (16.2 KiB)  TX bytes:16656 (16.2 KiB)

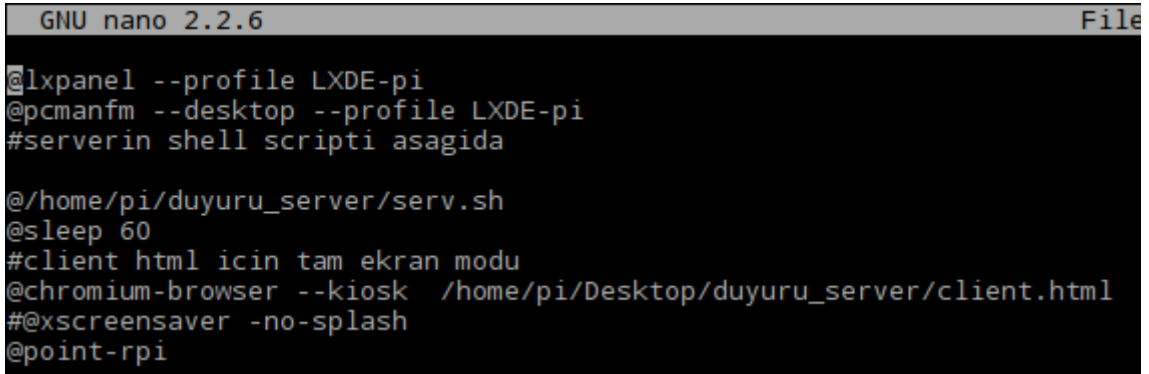
wlan0     Link encap:Ethernet  HWaddr b8:27:eb:98:fe:5c
          inet addr:192.168.0.15  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::9b8a:63c3:920b:cbdf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:553 errors:0 dropped:160 overruns:0 frame:0
          TX packets:111 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:144454 (141.0 KiB)  TX bytes:15539 (15.1 KiB)

pi@raspberrypi:~ $ scrot -s
```

## Genel İşlemler

Raspberry Pi açıldığı zaman sunucu kendiliğinden başlatılacak ve duyuruları içeren Chromium-Browser ise hemen ardından açılacaktır. Bu durumu durdurmak için aşağıdaki ayarları yapmak gerekmektedir.

1. 'Terminal' açılır.
2. 'sudo nano /.config/lxsession/lxde-pi/autostart' komutu girilir.
3. Açılan dosyanın ekran görüntüsü aşağıdadır.
4. '@/home/pi/duyuru\_server/serv.sh' ve '@chromium-browser...' ile başlayan satırlar '#' konarak yorum içine alınır.cmd
5. 'Ctrl+O' ya peşinide 'Enter' a basılarak dosya saklanıp kapatılır.



```
GNU nano 2.2.6 File
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
#serverin shell scripti asagida

@/home/pi/duyuru_server/serv.sh
@sleep 60
#client html icin tam ekran modu
@chromium-browser --kiosk /home/pi/Desktop/duyuru_server/client.html
#@xscreensaver -no-splash
@point-rpi
```

Sistem baştan başlatıldığında Sunucu ve Web Browser açılmayacaktır. Tekrar eski haline döndürmek için yukardaki işlemler 4. Madde hariç aynen tekrarlanır. 4. Madde de bahsedilen satırlardaki '#' ler silinerek kaydedilip baştan başlatıldığında eski haline dönmüş olacaktır.

## Sunucu Kullanıcı Nasıl Başlatılır?

Sunucu elle başlatmak için 'Terminal' üzerinden '/home/pi' klasörü içinde bulunan 'duyuru\_server' klasörüne gidilir. Bu klasör içindeyken aşağıdaki komut çalıştırılır.

'python3 server.py'

## KULLANICI PROGRAMINDAN RESİMLER

### Canlı Duyurular Sekmesi

Canlı Duyurular

Bekleyen Duyurular

Duyuru Ekle

Ayarlar

2.01.2017 01:55:38

	Duyuru Başlığı	Duyuru...	Bkendiği Tarih	Silineceği Tarih
▶	Final Sınavları	Final sınavları 02.01.2017 - 15.01.2017 tarihleri arasında yapılacaktır.	2.01.2017	9.01.2017
*				

Yenile

Sil

Duyuru Görüntüle

Bağlandı

IP: 127.0.0.1

Port: 9000

Bağlan

### Bekleyen Duyurular Sekmesi

Canlı Duyurular

Bekleyen Duyurular

Duyuru Ekle

Ayarlar

2.01.2017 01:56:47

	Duyuru Başlığı	Duyuru...	Bkendiği Tarih	Silineceği Tarih
*				

Yenile

Sil

Duyuru Görüntüle

Bağlandı

IP: 127.0.0.1

Port: 9000

Bağlan

## Duyuru Ekleme Sekmesi

Canlı DuyurularBekleyen DuyurularDuyuru EkleAyarlar

2.01.2017 01:57:54

☒ Aktif Duyuru☐ Planlanmış Duyuru

Duyuru Başlığı:  'Final Sınavları' başlıklı duyurunuz eklenmiştir.

Duyuru:

Ekleme Tarihi: Şimdi

Silinme Tarihi: 9 Ocak 2017 Pazartesi

Ekle

Bağlandı

IP:  127.0.0.1

Port:  9000

Bağlan

## Ayarlar Sekmesi

Canlı DuyurularBekleyen DuyurularDuyuru EkleAyarlar

2.01.2017 01:58:35

☒ Varsayılan Görünüm☐ Özel Görünüm

Sayfa Arka Plan Rengi:  Seç

Duyuru Arka Plan Rengi:  Seç

Duyuru Metin Rengi:  Seç

Duyuru Metin Fontu:  Seç Font

Duyuru Değişme Sıklığı:  5 sn

Uygula

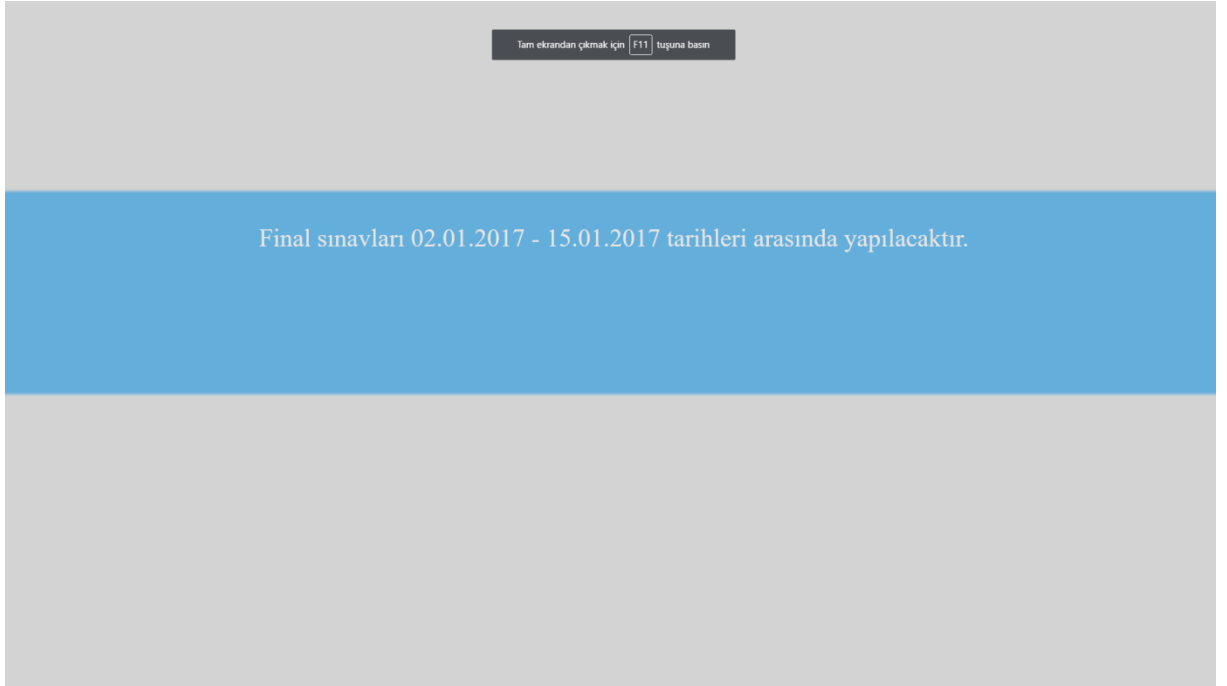
Bağlandı

IP:  127.0.0.1

Port:  9000

Bağlan

## RASPBERRY PI ÜZERİNDEKİ ARAYÜZ EKRANI



## KODLAR

### KULLANICI PROGRAMI KODLARI

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.Net.Sockets;

using Newtonsoft.Json;

namespace DuyuruProjesi
{
    3 references
    public partial class Form1 : Form
    {
        TcpClient clientSocket = new TcpClient();
        public Dictionary<string, string> defaultAyarlar = new Dictionary<string, string>();
        public Dictionary<string, string> ozelAyarlarDict = new Dictionary<string, string>();
        string kullanıcı = "Admin";
        1 reference
        public Form1()
        {
            InitializeComponent();
        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {
            timer1.Start();
            //defaultAyarYap();
            try
            {
                clientSocket.Connect("127.0.0.1", 9000);
                duyuruYenile();
                durumLabel.Text = "Bağlandı";
                durumLabel.ForeColor = Color.Green;
            }
            catch(SocketException)
            {
                durumLabel.Text = "Bağlanamadı";
                durumLabel.ForeColor = Color.Red;
            }
        }
        1 reference
        private void duyuruEkleButton_Click(object sender, EventArgs e)
        {
            DialogResult dlg = MessageBox.Show("Eklemek istediğimize emin misiniz?", "", MessageBoxButtons.YesNo);
            if (dlg == DialogResult.Yes)
            {

```

```

        if (dlg == DialogResult.Yes)
        {
            duyuruEkle();
            string bilgi = String.Format("'{}' başlıklı duyurunuz eklenmiştir.", duyuruBaslikTb.Text);
            ekleBilgiLabel.Text = bilgi;
            duyuruYenile();
            duyuruBaslikTb.Clear();
            duyuruTextBox.Clear();
            girisDateTimePicker.ResetText();
            planDateTimePicker.ResetText();
        }
    }

    1 reference
    private void duyuruEkle()
    {
        string duyuru = duyuruTextBox.Text;
        string baslik = duyuruBaslikTb.Text;
        string giris_tarih;
        string duyuru_tur;
        if (aktifRadioButton.Checked == true)
        {
            duyuru_tur = "aktif";
            giris_tarih = DateTime.Now.ToShortDateString();
        }
        else
        {
            duyuru_tur = "plan";
            giris_tarih = girisDateTimePicker.Value.ToShortDateString();
        }
        string cikis_tarih = planDateTimePicker.Value.ToShortDateString();
        Dictionary<string, string> duyuruEkleDict = new Dictionary<string, string>();
        duyuruEkleDict.Add("komut", "ekle");
        duyuruEkleDict.Add("kullanici", kullanici);
        duyuruEkleDict.Add("baslik", baslik);
        duyuruEkleDict.Add("duyuru", duyuruTextBox.Text);
        duyuruEkleDict.Add("giris_tarih", giris_tarih);
        duyuruEkleDict.Add("cikis_tarih", cikis_tarih);
        duyuruEkleDict.Add("duyuru_tur", duyuru_tur);
        string output = JsonConvert.SerializeObject(duyuruEkleDict);
        requestGonderAl(output);
    }

    1 reference
    private void aktifYenileButton_Click(object sender, EventArgs e)
    {
        duyuruYenile();
    }

```



```

    {
        duyuruYenile();
    }
    1 reference
    private void planDuyuruYenileButton_Click(object sender, EventArgs e)
    {
        duyuruYenile();
    }

    7 references
    private void duyuruYenile()
    {
        aktifGrid.Rows.Clear();
        planGrid.Rows.Clear();
        List<Dictionary<string, List<string>>>> countAndRowIdS = getCountAndRowId();
        Dictionary<string, string> yenileDuyuruDict = new Dictionary<string, string>();
        foreach (string rowId in countAndRowIdS[0]["rowIdS"])
        {
            yenileDuyuruDict.Add("komut", "yenile");
            yenileDuyuruDict.Add("rowid", rowId);
            string giden = JsonConvert.SerializeObject(yenileDuyuruDict);
            string gelen = requestGonderAl(giden);
            List<Dictionary<string, string>> rows = JsonConvert.DeserializeObject<List<Dictionary<string, string>>>(gelen);

            if (rows[0]["duyuru_tur"] == "aktif")
            {
                aktifGrid.Rows.Add(rows[0]["baslik"], rows[0]["duyuru"], rows[0]["giris_tarih"], rows[0]["cikis_tarih"], rows[0]["rowid"]);
            }
            else
            {
                planGrid.Rows.Add(rows[0]["baslik"], rows[0]["duyuru"], rows[0]["giris_tarih"], rows[0]["cikis_tarih"], rows[0]["rowid"]);
            }
            yenileDuyuruDict.Clear();
        }
    }
    1 reference
    private void planDuyuruSilButton_Click(object sender, EventArgs e)
    {
        DialogResult dlg = MessageBox.Show("Seçtiğiniz duyuruyu silmek istediğinize emin misiniz?", "", MessageBoxButtons.YesNo);
        if (dlg == DialogResult.Yes)
        {
            string rowid = planGrid.SelectedRows[0].Cells[4].Value.ToString();
            duyuruSil(rowid);
            duyuruYenile();
        }
    }
    1 reference
    private void aktifDuyuruSilButton_Click(object sender, EventArgs e)
    {
        DialogResult dlg = MessageBox.Show("Seçtiğiniz duyuruyu silmek istediğinize emin misiniz?", "", MessageBoxButtons.YesNo);
        if (dlg == DialogResult.Yes)
        {
            string rowid = aktifGrid.SelectedRows[0].Cells[4].Value.ToString();
            duyuruSil(rowid);
            duyuruYenile();
        }
    }
}

```

```

    }
    2 references
    private void duyuruSil(string rowid)
    {
        // a gidecek dict...
        Dictionary<string, string> duyuruSilDict = new Dictionary<string, string>();
        duyuruSilDict.Add("komut", "sil");
        duyuruSilDict.Add("rowid", rowid);
        //silme islemi...
        string giden = JsonConvert.SerializeObject(duyuruSilDict);
        string gelen = requestGonderAl(giden);
        //silmeden sonra tekrar update....
    }
    5 references
    private string requestGonderAl(string strToSend)
    {
        //veri gönderiliyor...

        NetworkStream serverStream = clientSocket.GetStream();
        byte[] outStream = Encoding.UTF8.GetBytes(strToSend);
        serverStream.Write(outStream, 0, outStream.Length);
        serverStream.Flush();

        //response alınıyor...
        byte[] inStream = new byte[4096];
        int bytesRead = serverStream.Read(inStream, 0, inStream.Length);
        string returnData = Encoding.ASCII.GetString(inStream, 0, bytesRead);
        Console.WriteLine(returnData);
        return returnData;
    }
    1 reference
    private List<Dictionary<string, List<string>>>> getCountAndRowId()
    {
        //update etmeden önce veritabanındaki verilerin rowidleri çekiliyor.
        Dictionary<string, string> getCountAndRowIdDict = new Dictionary<string, string>();
        getCountAndRowIdDict.Add("komut", "getRowIds");
        string giden = JsonConvert.SerializeObject(getCountAndRowIdDict);
        string gelen = requestGonderAl(giden);
        List<Dictionary<string, List<string>>>> countAndRowIds = JsonConvert.DeserializeObject<List<Dictionary<string, List<string>>>>>(gelen);
        return countAndRowIds;
    }
    1 reference
    private void aktifGrid_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)
    {
        aktifRichTb.Clear();
        aktifBaslikTb.Clear();
        aktifBaslikTb.Text = aktifGrid.Rows[e.RowIndex].Cells[0].Value.ToString();
        string richBox = aktifGrid.Rows[e.RowIndex].Cells[1].Value.ToString();
        aktifRichTb.AppendText(richBox);
    }
    1 reference
    private void planGrid_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)
    {

```

```

1 reference
private void planGrid_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    planBaslikTb.Clear();
    planRichTb.Clear();
    planBaslikTb.Text = planGrid.Rows[e.RowIndex].Cells[0].Value.ToString();
    string richBox = planGrid.Rows[e.RowIndex].Cells[1].Value.ToString();
    planRichTb.AppendText(richBox);
}

public int mousex, mousey, move;

1 reference
private void label1_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Minimized;
}

1 reference
private void label2_Click(object sender, EventArgs e)
{
    this.Close();
}

1 reference
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    move = 1;
    mousex = e.X;
    mousey = e.Y;
}

1 reference
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    if (aktifRadioButton.Checked == true)
    {
        girisDateTimePicker.Visible = false;
        label12.Visible = true;
        planDateTimePicker.Value = DateTime.Today.AddDays(+7);
    }
}

1 reference
private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    if (planRadioButton.Checked == true)
    {
        girisDateTimePicker.Visible = true;
        label12.Visible = false;
    }
}

1 reference
private void button2_Click(object sender, EventArgs e)
{
    try
    {

```

```

}
1 reference
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        string ip = textBox2.Text;
        int port = Convert.ToInt32(textBox3.Text);
        clientSocket.Connect(ip, port);
        duyuruYenile();
        durumLabel.Text = "Bağlandı";
        durumLabel.ForeColor = Color.Green;
    }
    catch (SocketException)
    {
        durumLabel.Text = "Bağlanamadı";
        durumLabel.ForeColor = Color.Red;
    }
}

1 reference
private void timer1_Tick(object sender, EventArgs e)
{
    tarihSaatLabel.Text = DateTime.Now.ToString();
}

1 reference
private void label10_MouseDoubleClick(object sender, MouseEventArgs e)
{
}

1 reference
private void panel1_MouseUp(object sender, MouseEventArgs e)
{
    move = 0;
}

1 reference
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    if (move == 1)
    {
        this.SetDesktopLocation(MousePosition.X - mousex, MousePosition.Y - mousey);
    }
}

//AYARLAR
1 reference
private void ozelGorunumRb_CheckedChanged(object sender, EventArgs e)
{
    if (ozelGorunumRb.Checked == true)
    {
        arkaPlanDialogButton.Enabled = true;
        duyuruArkaPlanDialogButton.Enabled = true;
        metinRenkDialogButton.Enabled = true;
        metinFontDialogButton.Enabled = true;

        ayarUygulaButton.Enabled = true;
    }
}

```

```

1 reference
private void varsayilanGorunumRb_CheckedChanged(object sender, EventArgs e)
{
    if (varsayilanGorunumRb.Checked == true)
    {
        arkaPlanDialogButton.Enabled = false;
        duyuruArkaPlanDialogButton.Enabled = false;
        metinRenkDialogButton.Enabled = false;
        metinFontDialogButton.Enabled = false;

        //-----//

        sayfaArkaRenkTb.BackColor = Color.FromArgb(211, 211, 211);
        duyuruArkaRenkTb.BackColor = Color.FromArgb(100, 174, 219);
        metinRenkTb.BackColor = Color.FromArgb(230, 230, 230);
        metinFontLbl.Font = new Font("Sans-Serif", metinFontLbl.Font.Size);
    }
}

0 references
private void defaultAyarYap()
{
    sayfaArkaRenkTb.BackColor = Color.FromArgb(211, 211, 211);
    duyuruArkaRenkTb.BackColor = Color.FromArgb(100, 174, 219);
    metinRenkTb.BackColor = Color.FromArgb(230, 230, 230);
    metinFontLbl.Font = new Font("Sans-Serif", metinFontLbl.Font.Size);
    defaultAyarlar.Add("sayfaArkaPlan", "#d3d3d3");
    defaultAyarlar.Add("duyuruArkaPlan", "#64aedb");
    defaultAyarlar.Add("metinRengi", "#e6e6e6");
    defaultAyarlar.Add("metinFontu", "Sans-Serif");
    defaultAyarlar.Add("donusSure", "5");
}

1 reference
private void ayarUygulaButton_Click(object sender, EventArgs e)
{
    ozelAyarlarDict.Clear();
    ozelAyarlarDict.Add("sayfaArkaPlan", sayfaArkaRenkTb.BackColor.Name);
    ozelAyarlarDict.Add("duyuruArkaPlan", "#64aedb");
    ozelAyarlarDict.Add("metinRengi", "#e6e6e6");
    ozelAyarlarDict.Add("metinFontu", metinFontLbl.Font.Name);
    ozelAyarlarDict.Add("donusSure", "5");
}

0 references
private Dictionary<string, string> getAyarlar()
{
    Dictionary<string, string> ayarlarDict = new Dictionary<string, string>();
    ayarlarDict.Add("komut", "ayarlar");
    string giden = JsonConvert.SerializeObject(ayarlarDict);
    string gelen = requestGonderAl(giden);
    Dictionary<string, string> rows = JsonConvert.DeserializeObject<Dictionary<string, string>>(gelen);
    return rows;
}

```

## RASPBERRY PI ARAYÜZ EKRANI KODLARI

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <div class="mainDiv">
      <ul id="duyurular">
      </ul>
    </div>
    <script type="text/javascript">

      var socket = null;
      var isopen = false;
      var currentSlide = 0;
      var myInterval = null;
      window.onload = function() {
        //socket = new WebSocket("ws://192.168.0.17:9001");
        socket = new WebSocket("ws://127.0.0.1:9001");
        socket.binaryType = "arraybuffer";

        socket.onopen = function () {
          console.log("Connected!");
          isopen = true;
          sendText();
        }

        socket.onmessage = function (e) {

          if (typeof e.data == "string") {
            if (e.data == "kontrol")
            {
              console.log("Yeni veri girişi!!!");
              console.log("Bağlantı açıksa güncel listeyi almaya gidiyorum");
              sendText();
            }
            else
            {
              //else if haline getir...
              currentSlide = 0;
              clearInterval(myInterval);
              var aktif_duyurular = JSON.parse(e.data);
              //try
              var node = document.getElementById('duyurular');
              while (node.hasChildNodes()) {
                node.removeChild(node.firstChild);
              }
            }
          }
        }
      }
    </script>
  </body>
</html>
```

```

    }
    console.log(aktif_uyurular)
    for (ind = 0; ind < aktif_uyurular.length; ++ind) {

        var element = document.createElement('p');
        if (ind == 0)
        {
            element.className = "duyuru showing";
        }
        else
        {
            var element = document.createElement('p');
            element.className = "duyuru";
        }

        element.innerHTML = aktif_uyurular[ind];
        document.getElementById("duyurular").appendChild(element);
    }
    myInterval = setInterval(nextSlide, 5000);
}

}

socket.onclose = function (e) {
    //console.log("Connection closed.");
    socket = null;
    isopen = false;
}

};

function nextSlide() {
    //console.log(currentSlide)
    var slides = document.querySelectorAll("#duyurular .duyuru");

    slides[currentSlide].className = "duyuru";
    currentSlide = (currentSlide+1)%slides.length;
    slides[currentSlide].className = "duyuru showing";
}

function sendText() {
    if (isopen) {
        socket.send("aktif");
        console.log("Aktif duyuruları almak için mesaj gönderildi.");
    } else {
        console.log("Connection not opened.")
    }
}

};

```

```

<style>

    #duyurular {
        position: relative;
        height: 300px;
        padding: 0px;
        margin: 0px;
        list-style-type: none;
    }

    .duyuru {
        position: absolute;
        left: 0px;
        top: 0px;
        width: 100%;
        height: 100%;
        opacity: 0;
        z-index: 1;

        -webkit-transition: opacity 1s;
        -moz-transition: opacity 1s;
        -o-transition: opacity 1s;
        transition: opacity 1s;
    }

    .showing {
        opacity: 1;
        z-index: 2;
    }

```

```

    .duyuru {
        word-break: break-all;
        font-size: 40px;
        padding: 40px;
        box-sizing: border-box;
        box-shadow: 0 0 5px 10px rgb(100, 174, 219);
        background: rgb(100, 174, 219);
        color: rgb(230, 230, 230);
        text-align: center;
    }

    body {
        background: rgb(211, 211, 211);
    }

    .mainDiv
    {
        /*yeni div ekleyince benzer css kullan*/
        width: 100%;
        height: 50%;
        position: absolute;
        top: 0;
        bottom: 0;
        left: 0;
        right: 0;
        margin: auto;
    }

```

```

</style>

```

```

</body>

```



## SUNUCU KODLARI

```
__author__ = 'Muratcan Unsal'

import json
import sqlite3
import sys
from autobahn.twisted.websocket import WebSocketServerProtocol, WebSocketServerFactory
from datetime import datetime

class CsharpProtocol(WebSocketServerProtocol):

    def connectionMade(self):
        print("C# baglandı")

    def dataReceived(self, data):

        self.data_json = self.getJson(data)
        komut = self.data_json["komut"]

        if komut == "getRowIdS":
            dataToGo = bytes(json.dumps([{"count": self.getCount(), "rowIdS": self.getRowId()}])+"\n", "utf-8")
            self.transport.write(dataToGo)

        elif komut == "ekle":
            self.duyuruEkle(self.data_dict["kullanici"], self.data_dict["baslik"], self.data_dict["duyuru"],
                           self.data_dict["giris_tarih"], self.data_dict["cikis_tarih"], self.data_dict["duyuru_tur"])
            dataToGo = bytes(json.dumps([{"status": "ok"}]) + "\n", "utf-8")

            self.factory.broadcast("kontrol")
            self.transport.write(dataToGo)

        elif komut == "yenile":
            self.reqRowId = int(self.data_dict["rowid"])
            dataToGo = bytes(json.dumps(self.yenileDuyuru(self.reqRowId)) + "\n", "utf-8")
            self.transport.write(dataToGo)

        elif komut == "sil":
            self.duyuruSil(self.data_dict["rowid"])
            dataToGo = bytes(json.dumps([{"status": "ok"}]) + "\n", "utf-8")
            self.factory.broadcast("kontrol")
            self.transport.write(dataToGo)

        elif komut == "yolla":
            msg = "{} from {}".format(self.data_dict["msg"], self.peer)
            self.factory.broadcast(msg)

    def connectionLost(self, reason):
        print(reason)
```

```

def getJson(self, data):
    self.data_dict = str(data, "utf-8")
    self.data_dict = json.loads(self.data_dict)
    return self.data_dict

def duyuruEkle(self, kullanıcı, baslik, duyuru, giris_tarih, cikis_tarih, duyuru_tur):
    #DUYURU EKLEME
    try:
        conn = sqlite3.connect('duyuruDB.db')
        c = conn.cursor()
        c.execute('insert into duyuru values (?, ?, ?, ?, ?, ?)', [kullanıcı, baslik, duyuru, giris_tarih, cikis_tarih, duyuru_tur])
        conn.commit()
        conn.close()
    except sqlite3.OperationalError:
        print("Operational Error:", sys.exc_info()[1])

def yenileDuyuru(self, reqRowId):
    #tablonun son halini alıp, json a çevirip, sonra byte a çeviriyor.
    #PATLAYACAK MÜHÜRMELEN VERİ SİLİNİNCİ....
    try:
        conn = sqlite3.connect('duyuruDB.db')
        c = conn.cursor()
        c.execute('select rowid, * from duyuru where rowid = (?)', [reqRowId])
        rows = c.fetchall()
        rowsListed = [{"rowid": row[0], "kullanıcı": row[1], "baslik": row[2], "duyuru": row[3],
                        "giris_tarih": row[4], "cikis_tarih": row[5], "duyuru_tur": row[6]} for row in rows]

        #dataToGO = bytes(json.dumps(rowsListed) + "\n", "utf-8")
        return rowsListed

    except sqlite3.OperationalError:
        print("Operational Error:", sys.exc_info()[1])

def duyuruSil(self, reqRowId):
    try:
        conn = sqlite3.connect('duyuruDB.db')
        c = conn.cursor()
        c.execute('delete from duyuru where rowid = (?)', [reqRowId])
        conn.commit()
        conn.close()
    except sqlite3.OperationalError:
        print("Operational Error:", sys.exc_info()[1])

def getCount(self):
    conn = sqlite3.connect('duyuruDB.db')
    c = conn.cursor()
    c.execute('select count(*) from duyuru')

```

```

def getCount(self):
    conn = sqlite3.connect('duyuruDB.db')
    c = conn.cursor()
    c.execute('select count(*) from duyuru')
    count = c.fetchone()
    return [count[0]]

def getRowId(self):
    conn = sqlite3.connect('duyuruDB.db')
    c = conn.cursor()
    c.execute('select rowid from duyuru')
    rowIdS = c.fetchall()
    rowIdS = [k[0] for k in rowIdS]

    return rowIdS

```

```

class ClientHtmlProtocol(WebSocketServerProtocol):

```

```

    def onOpen(self):

        if len(self.factory.clients) != 0:
            self.factory.clients = []

        self.factory.register(self)

    def onMessage(self, payload, isBinary):
        komut = payload.decode('utf8')
        if komut == "aktif":
            duyurular = self.aktifDuyurular()
            duyurular = json.dumps(duyurular)
            self.factory.broadcast(duyurular)

    def connectionLost(self, reason):

        WebSocketServerProtocol.connectionLost(self, reason)
        print("Losing my mind")
        self.factory.unregister(self)

    def aktifDuyurular(self):
        conn = sqlite3.connect('duyuruDB.db')
        c = conn.cursor()

        c.execute('select * from duyuru where duyuru_tur = (?)', ['aktif'])
        rows = c.fetchall()
        rowsListed = [row[2] for row in rows]
        return rowsListed

```

```

class BroadcastServerFactory(WebSocketServerFactory):
    """
    Gelen tüm mesajları bağlı olan tüm clientlara yollayan Broadcast Server
    """

    clients = []

    def __init__(self, url):
        WebSocketServerFactory.__init__(self, url)
        #self.clients = []
        self.veriGüncelle()

    def register(self, client):

        if not client in self.clients:
            print("registered client {}".format(client.peer))
            self.clients.append(client)
            print(self.clients)

    def unregister(self, client):
        if client in self.clients:
            print("unregistered client {}".format(client.peer))
            self.clients.remove(client)

    def broadcast(self, msg):
        print("broadcasting message '{}' ..".format(msg))
        for c in self.clients:
            c.sendMessage(msg.encode('utf8'))
            print("message sent to {}".format(c.peer))

    def veriGüncelle(self):

        try:
            conn = sqlite3.connect('duyuruDB.db')
            c = conn.cursor()
            c.execute('select rowid, * from duyuru')
            rows = c.fetchall()
            aktifHaleGelecekDuyurular = []
            silinecekDuyurular = []
            for row in rows:
                giris_tarih = datetime.strptime(row[4], "%d.%m.%Y")
                cikis_tarih = datetime.strptime(row[5], "%d.%m.%Y")
                if row[6] == "plan":
                    if giris_tarih <= datetime.now():
                        aktifHaleGelecekDuyurular.append(row[0])
                        print("{}' aktif hale gelecek. rowId = {}".format(row[3], row[0]))
                if datetime.now() > cikis_tarih:
                    silinecekDuyurular.append(row[0])
                    print("{}' silinecek. rowId = {}".format(row[3], row[0]))

            self.duyuruGüncelAktif(aktifHaleGelecekDuyurular)

```

```

        self.uyuruGüncelAktif(aktifHaleGelecekUyurular)
        self.uyuruGüncelSil(silinecekUyurular)
    except sqlite3.OperationalError:
        print("Operational Error:", sys.exc_info()[1])

def uyuruGüncelSil(self, rowsList):
    try:
        conn = sqlite3.connect('uyuruDB.db')
        c = conn.cursor()
        for rowId in rowsList:
            c.execute('delete from uyuru where rowid = (?)', [rowId])
            print("{0} rowID li uyuru silindi".format(rowId))
        conn.commit()
        conn.close()
    except sqlite3.OperationalError:
        print("Operational Error:", sys.exc_info()[1])

def uyuruGüncelAktif(self, rowsList):
    try:
        conn = sqlite3.connect('uyuruDB.db')
        c = conn.cursor()
        for rowId in rowsList:
            c.execute('update uyuru set uyuru_tur = (?) where rowid = (?)', ['aktif', rowId])
            print("{0} rowID li uyuru aktif hale geldi".format(rowId))
        conn.commit()
        conn.close()
    except sqlite3.OperationalError:
        print("Operational Error:", sys.exc_info()[1])

def main():

    import sys
    from twisted.python import log
    from twisted.internet import reactor

    log.startLogging(sys.stdout)

    csharpFactory = BroadcastServerFactory(u"ws://127.0.0.1:9000")
    csharpFactory.protocol = CsharpProtocol

    clientHtmlFactory = BroadcastServerFactory(u"ws://127.0.0.1:9001")
    clientHtmlFactory.protocol = ClientHtmlProtocol
    reactor.listenTCP(9000, csharpFactory)
    reactor.listenTCP(9001, clientHtmlFactory)
    reactor.run()

if __name__ == '__main__':
    main()

```

## Proje Resimleri

