

# DOKUNMATİK PANEL KALİBRASYON VE BAKIM ARAYÜZÜ

HAKAN AKGÜN

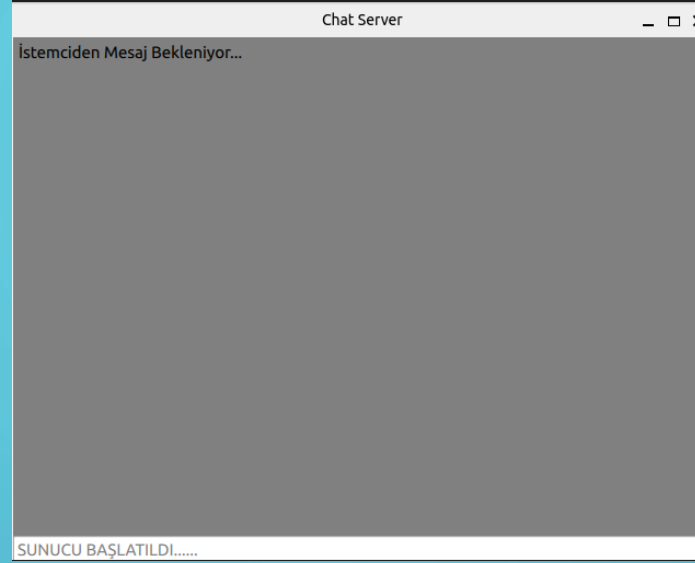
# PROJE AMACI



- Bu proje , yer istasyonunda bulunan bilgisayar ile İHA kullanımına yardımcı arayüze sahip tabletler arasındaki kalibrasyonu sağlayan ve bazı teknik işlemleri gerçekleştirilmesi sağlayan bir arayüz projesidir.
- Böylelikle tablete erişim sağlamadan yer istasyonu bilgisayarından tablet ile ilgili işler gerçekleştirilebilecektir.



Şekil 1. Windows C# uygulaması



Şekil 2. Linux C++ uygulaması

- Windows bilgisayarda çalışacak olan proje C# WPF MVVM pattern ve Material Design ile oluşturulmuştur. Tableti simüle edecek sanal makineye(LINUX) bağlanarak işlemleri gerçekleştirir.
- Tablette yani sanal makinede de olan arayüz projesi ise mesajlaşma uygulaması olarak, QT C++ QML ile gerçekleştirilmiştir.



# MVVM PATTERN

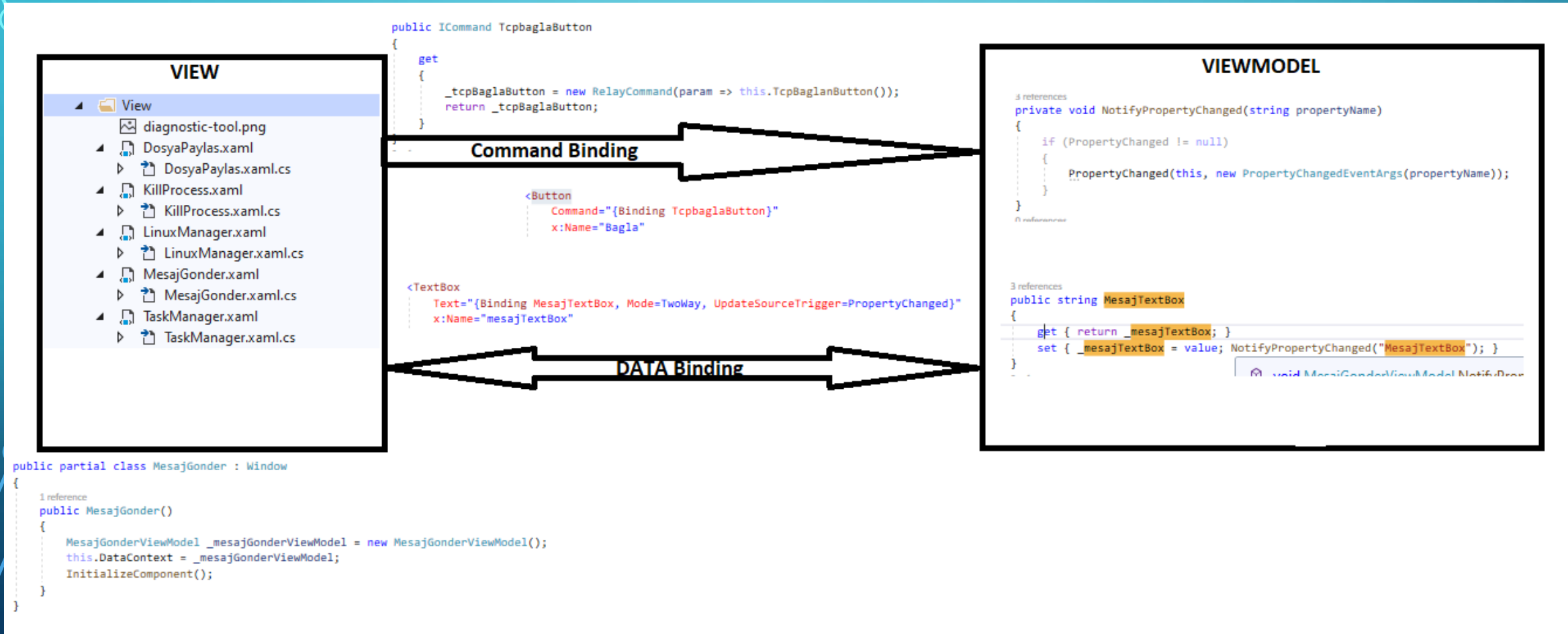
- **MVVM:** Presentation Model mimarisinin WPF ve Silverlight teknolojileri için özelleştirilmiş bir halidir.(Presentation model presenter ile karşılaştırıldığında View ile çok daha fazla konuşkan iletişime sahiptir.) Arayüz(View) ile logic işlemlerin(model) birbirlerinden bağımsız çalışmasını sağlayan yapıdır. Amacı geliştirmeyi kolaylaştırmak ve frontend developer ile backend developerların bağımsız çalışabilmesidir. Temel parçaları;
- **Model:** Model, uygulama içinde kullanacağımız verilerdir.
- **View:** View, datanın sunulduğu katmandır. Tüm görsellikler viewda yer alır. Kısaca verinin sunulduğu yerdir(Ekran). Bu sınıfların amacı ViewModelden istediğimiz verileri Observe etmektir. Burada lojik işlemler gerçekleştirilmez
- **ViewModel:** ViewModel ise model ve viewı bağlayan yapıdır. View ile model arasında bir yapııştırıcı görevi görür. View doğrudan ViewModel yardımıyla modele erişir ve bazı işlemleri gerçekleştirir. Teknolojik olarak WPFden konuşursak ViewModel aslında viewın DataContext idir. Bu sınıflarda lojik işlemler gerçekleştirilir.

**DataContext:** *Veri bağlama işlemlerinde kullanılmaktadır.* Bu bileşenler sayesinde XAML içerisinde, ilgili kaynaklara bağlanma, tek yönlü ve çift yönlü olarak veri transfer etme(Data Binding) işlemlerini gerçekleştirebiliriz.

**Neden MVVM:**

- \* Commanding, Binding gibi özellikleri kullanarak hızlı ve etkin uygulama geliştirme.
- \* View arkasında bulunan kod dosyasına sıfır kod yazarak bağımlılıkları minimuma indirmek ve uygulama içerisinde katmanların bağımlılıklarını minimuma indirmek.
- \* Daha genişletilebilir, bakımı kolay ve test edilebilir uygulama geliştirme.
- \* Arayüzü kolay bir şekilde taşınabilen uygulamalar geliştirmek.

# DIAGRAM



- **NotifyPropertyChanged:** C# tarafından uygulama ekranında bulunan herhangi bir UI Control'ünün değeri değiştirme işlemlerini sık sık yaparız. İşte bu gibi işlemleri örneğin TextBlock'un Text'ini değiştirme işlemini C# tarafında tbName.Text="Hakan"; yazmak yerine INotifyPropertyChanged interface'ini kullanarak bu gibi işlemleri kolaylıkla ve daha yönetilebilir bir şekilde yapabiliriz. Sadece text için değil buton click eventleri içinde kullanılabilir.
- **RelayCommand.cs Class'ı:** RelayCommand button'a tıklandığında çalışacak olan event gibi düşünebiliriz, Butonun click statelerini aşağıda ki metodlar sayesinde handle edip yönetimini sağlıyoruz

```
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Windows.Input;
7
8 namespace MvvmDeneme.Command
9 {
10     12 references
11     public class RelayCommand : ICommand
12     {
13         private Action<object> execute;
14         private Func<object, bool> canExecute;
15
16         public event EventHandler CanExecuteChanged
17         {
18             add { CommandManager.RequerySuggested += value; }
19             remove { CommandManager.RequerySuggested -= value; }
20         }
21
22         11 references
23         public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)
24         {
25             this.execute = execute;
26             this.canExecute = canExecute;
27         }
28
29         0 references
30         public bool CanExecute(object parameter)
31         {
32             return this.canExecute == null || this.canExecute(parameter);
33         }
34
35         0 references
36         public void Execute(object parameter)
37         {
38             this.execute(parameter);
39         }
40     }
41 }
```

# İlk Ekran Kodları

```
291     }
292 }
293 1 reference
294 public void Baglanti()
295 {
296     if (TextSelect == null || SifreSelect == null || SelectedIps == null)
297     {
298         MessageBox.Show("Bilgileri Eksiksiz Giriniz!");
299     }
300     else
301     {
302         host = SelectedIps.ToString();
303         username = TextSelect.ToString();
304         password = SifreSelect.ToString();
305         string result = SshBaglanti("ps -aux");//ps -aux
306         string[] subStrings = result.Split('\n');
307         MyListItems = subStrings;
308     }
309 }
310 1 reference
311 private string SshBaglanti(string command)
312 {
313     string result = string.Empty;
314     try
315     {
316         AuthenticationMethod method = new PasswordAuthenticationMethod(username, password);
317         ConnectionInfo connection = new ConnectionInfo(host, username, method); // ip: 192.168.139.129 kullaciadi: hkn sifre: 123
318         sshClient = new SshClient(connection);
319
320         sshClient.Connect();
321         SshCommand cmd = sshClient.RunCommand(command);
322         result = cmd.Result;
323     }
324     catch (Exception ex)
325     {
326         MessageBox.Show(ex.Message);
327         if (!sshClient.IsConnected)
328         {
329             TextSelect = "";
330             SifreSelect = "";
331             MessageBox.Show("Bağlanamadı, Kontrol Ediniz!");
332         }
333     }
334     return result;
335 }
336 }
```

Şekil 3. SSH Bağlantısı

```
197 1 reference
198 public void BaglantiKopar()
199 {
200     string[] arr = new string[] { };
201     if (!(this.sshClient == null) && this.sshClient.IsConnected)
202     {
203         this.sshClient.Disconnect();
204         SelectedIps = null;
205         TextSelect = string.Empty;
206         SifreSelect = string.Empty;
207         MyListItems = arr;
208         CpuLabel = "";
209         MemLabel = "";
210         MessageBox.Show("Bağlantı Koptu");
211     }
212     else
213         MessageBox.Show("Bağlantınız yok");
214 }
215 1 reference
216 public void Yenile()
217 {
218     string result = string.Empty;
219     if ((this.sshClient == null) || !(this.sshClient.IsConnected))
220     {
221         MessageBox.Show("Bağlantınız Bulunmamaktadır.");
222     }
223     else
224     {
225         SshCommand cmd = this.sshClient.RunCommand("ps -aux");
226         result = cmd.Result;
227     }
228     string[] subStrings = result.Split('\n');
229     MyListItems = subStrings;
230
231     CpuLabel = "";
232     MemLabel = "";
233 }
```

Şekil 4. Bağlantı Kopar ve Yenile

**SSH:** Bir bilgisayar ile diğer bir bilgisayar arasında güvenli bir bağlantı kurulumunu sağlayan ağ protokolüdür. Bilgisayar/Sunucu, Sunucu/Sunucu arasında bağlantıları sağlar. Ssh iki bilgisayar arasındaki bağlantıyı gizli ve açık olarak iki anahtarla şifreler.

```

232 1 reference
233 public void ProcessSonlandır()
234 {
235     try
236     {
237         string secilen = MyListItems[SelectedIndex].ToString();
238
239         string[] parse = secilen.Split(' ');
240
241         int index = 0;
242         string pid = string.Empty;
243         string cpu = string.Empty;
244         string memory = string.Empty;
245         foreach (string c in parse)
246         {
247             if (c.Length != 0)
248             {
249                 if (index == 1)
250                 {
251                     pid = c;
252                 }
253                 else if (index == 2)
254                 {
255                     cpu = c;
256                 }
257                 else if (index == 3)
258                 {
259                     memory = c;
260                     break;
261                 }
262                 index++;
263             }
264         }
265         if (!(secilen == MyListItems[0]))
266         {
267             if (secilen != null || secilen == "")
268             {
269                 CpuLabel = (pid + " Numaralı Sürecin CPU Kullanımı(%): " + cpu);
270                 MemLabel = (pid + " Numaralı Sürecin Memory Kullanımı(%)" + memory);
271             }
272             if (this.host == "" || this.username == "" || this.password == "")
273             {
274                 MessageBox.Show("Lütfen IP Kullanıcı Ve Şifreyi Giriniz");
275             }
276             else
277             {
278                 KillProcess _killProcessView = new KillProcess(sshClient, pid);
279                 Application.Current.MainWindow = _killProcessView;
280                 _killProcessView.sureclbl.Content = (pid + " Numaralı Süreci Sonlandırmaya Emin misiniz?");
281                 Application.Current.MainWindow.Show();
282             }
283         }
284         else
285         {
286             MessageBox.Show("Yanlış Seçim, Lütfen Bir Süreç Seçiniz.");
287         }
288     }
289     catch (Exception)
290     {
291         MessageBox.Show("Lütfen Bir Süreç Seçiniz");
292     }

```

Şekil 5. Süreç Sonlandırma

```

174 1 reference
175 public void LinuxManage()
176 {
177     LinuxManager _linuxManagerView = new LinuxManager(this.sshClient);
178     Application.Current.MainWindow = _linuxManagerView;
179 }
180 1 reference
181 public void TaskManage()
182 {
183     TaskManager _taskManagerView = new TaskManager();
184     Application.Current.MainWindow = _taskManagerView;
185     //Application.Current.MainWindow.Show();
186 }
187 1 reference
188 public void DosyaGonder()
189 {
190     DosyaPaylas _dosyaGonderView = new DosyaPaylas(host, username, password);
191     Application.Current.MainWindow = _dosyaGonderView;
192     Application.Current.MainWindow.Show();
193 }
194 1 reference
195 public void MesajGonder()
196 {
197     MesajGonder _mesajGonderView = new MesajGonder();
198     Application.Current.MainWindow = _mesajGonderView;
199     Application.Current.MainWindow.Show();
200 }

```

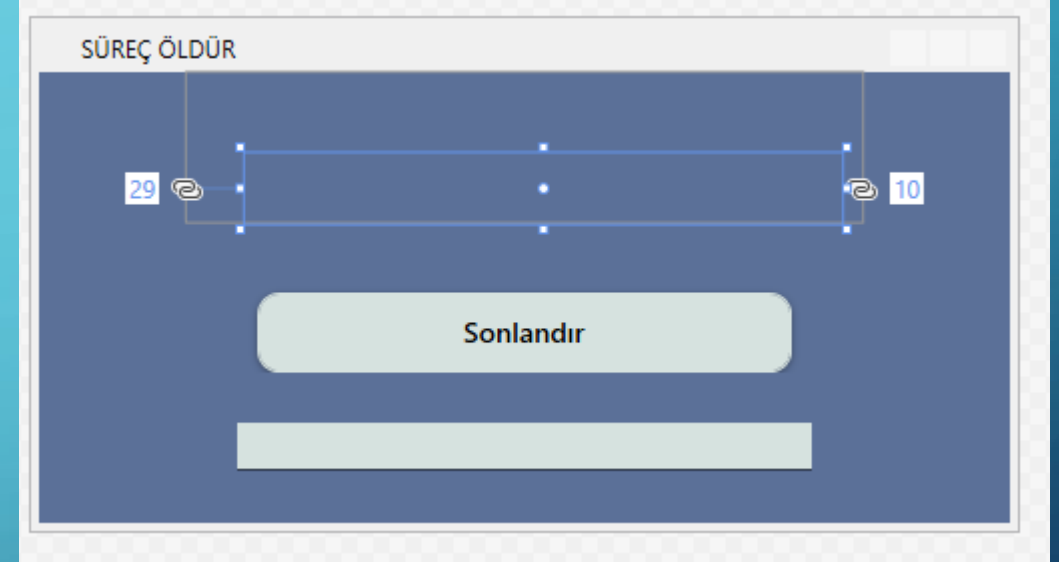
Şekil 6. İlk Ekranda Olan Diğer Butonlar



# Süreç Sonlandır Ekranı

```
13 namespace MvvmDeneme.ViewModel
14 {
15     3 references
16     public partial class KillProcessViewModel : INotifyPropertyChanged
17     {
18         private ICommand _pidSonlandirButton;
19         private SshClient sshClient;
20         private string pidSayisi = string.Empty;
21         private string _tp;
22         public event PropertyChangedEventHandler PropertyChanged;
23         1 reference
24         private void NotifyPropertyChanged(string propertyName)
25         {
26             if (PropertyChanged != null)
27             {
28                 PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
29             }
30         }
31         1 reference
32         public string TextSelect
33         {
34             get { return _tp; }
35             set { _tp = value; NotifyPropertyChanged("TextSelect"); }
36         }
37         1 reference
38         public KillProcessViewModel(SshClient sshClient, string pidSayisi)
39         {
40             this.sshClient = sshClient;
41             this.pidSayisi = pidSayisi;
42         }
43         0 references
44         public ICommand pidSonlandirButton
45         {
46             get
47             {
48                 _pidSonlandirButton = new RelayCommand(param => this.ProcessSonlandirPid());
49                 return _pidSonlandirButton;
50             }
51         }
52         1 reference
53         public void ProcessSonlandirPid()
54         {
55             string command = ("kill -9 " + this.pidSayisi);
56             SshCommand cmd = sshClient.RunCommand(command);
57             if (cmd.Error == null)
58             {
59                 TextSelect = (this.pidSayisi + " numaralı süreç sonlandırılmıştır.");
60             }
61             else
62             {
63                 MessageBox.Show(cmd.Error);
64                 Application.Current.MainWindow.Close();
65             }
66         }
67     }
68 }
```

Şekil 7. Süreç Sonlandır Butonu



Şekil 8. KillProcess.xaml

# Dosya Paylaşımı Ekranı

```
276 private void comboBox_SelectionChanged()  
277 {  
278     if (host == "" || username == "" || password == "")  
279     {  
280         MessageBox.Show("Bağlantı Yok!");  
281     }  
282     else  
283     {  
284         using (SftpClient client = new SftpClient(this.host, 22, this.username, this.password))  
285         {  
286             client.KeepAliveInterval = TimeSpan.FromSeconds(60);  
287             client.ConnectionInfo.Timeout = TimeSpan.FromMinutes(180);  
288             client.OperationTimeout = TimeSpan.FromMinutes(180);  
289             client.Connect();  
290             if (SelectServerClient == "SUNUCU/İSTEMCİ")  
291             {  
292                 ButtonGizleB = Visibility.Hidden;  
293                 ButtonGizleA = Visibility.Visible;  
294                 LblFilePath = "";  
295                 treeView.Clear();  
296                 ListDirectory(treeView, "C:\\Users\\PC_1770\\Desktop");//C:\\Users\\PC_1770\\Desktop  
297             }  
298             if (SelectServerClient == "İSTEMCİ/SUNUCU")  
299             {  
300                 ButtonGizleA = Visibility.Hidden;  
301                 ButtonGizleB = Visibility.Visible;  
302                 LblFilePath = "";  
303                 treeView.Clear();  
304                 DownloadDirectoryForLinux(client, "/home/hkn/Desktop");  
305                 //RemoteFileButton_Click(this.sftpclient);  
306             }  
307         }  
308     }  
309 }
```

Şekil 9. SFTP Bağlantısı

**SFTP:** FTP bilgisayarlar arasında güvenli dosya alışverişi standart yöntemidir. SFTP ise daha güvenli halidir. SSH protokolünü kullanır. Paketler şifrelenir ve paketler halinde taşınır. SFTP SSH ile aynı port üzerinde çalışır(22). SFTPde log kayıtları tutulur.

```
117 private void ListDirectory(ObservableCollection<TreeViewItem> treeView, string path)  
118 {  
119     //treeView.Clear();  
120     var rootDirectoryInfo = new DirectoryInfo(path);  
121     treeView.Add(CreateDirectoryNode(rootDirectoryInfo));  
122 }  
123  
124 2 references  
125 private static TreeViewItem CreateDirectoryNode(DirectoryInfo directoryInfo)  
126 {  
127     var directoryNode = new TreeViewItem { Header = directoryInfo.Name };  
128     foreach (var directory in directoryInfo.GetDirectories())  
129         directoryNode.Items.Add(CreateDirectoryNode(directory));  
130     foreach (var file in directoryInfo.GetFiles())  
131         directoryNode.Items.Add(new TreeViewItem { Header = file.Name });  
132     return directoryNode;  
133 }  
134  
135 2 references  
136 private void DownloadDirectoryForLinux(SftpClient client, string remotePath)  
137 {  
138     try  
139     {  
140         IEnumerable<SftpFile> files = client.ListDirectory(remotePath);//LblFilePath2.Content.ToString()  
141         foreach (SftpFile file in files)  
142         {  
143             if ((file.Name != ".") && (file.Name != ".."))  
144             {  
145                 string sourceFilePath = remotePath + "/" + file.Name;  
146                 if (file.IsDirectory)  
147                 {  
148                     var directoryNode = new TreeViewItem { Header = "***" + file.Name };  
149                     treeView.Add(directoryNode);  
150                     DownloadDirectoryForLinux(client, sourceFilePath);  
151                 }  
152                 else  
153                 {  
154                     var directoryNode = new TreeViewItem { Header = file.Name };  
155                     treeView.Add(directoryNode);  
156                 }  
157             }  
158         }  
159     }  
160     catch (Exception ex)  
161     {  
162         throw;  
163     }  
164 }
```

Şekil 10. Treeview Linux Bilgisayar Ve Windows Bilgisayar için Veri Yükleme

```

35 public DosyaPaylas(string host, string username, string password)
36 {
37     this.host = host;
38     this.username = username;
39     this.password = password;
40
41     DosyaPaylasViewModel _dosyaPaylasViewModel = new DosyaPaylasViewModel(host, username, password);
42     this.DataContext = _dosyaPaylasViewModel;
43     InitializeComponent();
44 }
45
46 1 reference
47 private void DownloadFileButton_Click(object sender, RoutedEventArgs e)
48 {
49     string localDir = @"C:\Users\PC_1770\Desktop";
50     var remoteDir = "/home/hkn/Desktop";
51
52     using (SftpClient client = new SftpClient(this.host, this.username, this.password))
53     {
54         client.KeepAliveInterval = TimeSpan.FromSeconds(60);
55         client.ConnectionInfo.Timeout = TimeSpan.FromMinutes(180);
56         client.OperationTimeout = TimeSpan.FromMinutes(180);
57         client.Connect();
58         bool connected = client.IsConnected;
59         if (connected == true)
60             MessageBox.Show("SFTP Bağlantısı Sağlandı");
61
62         client.ChangeDirectory(remoteDir);
63
64         DownloadDirectory(client, remoteDir, localDir);
65
66         MessageBox.Show("Dosyanız Server ile Paylaşılmıştır.");
67
68         client.Disconnect();
69     }
70
71 1 reference
72 private void ShareFileButton_Click(object sender, RoutedEventArgs e)
73 {
74     string localDir = @"C:\Users\PC_1770\Desktop";
75     var remoteDir = "/home/hkn/Desktop";
76
77     using (SftpClient client = new SftpClient(this.host, 22, this.username, this.password))
78     {
79         client.KeepAliveInterval = TimeSpan.FromSeconds(60);
80         client.ConnectionInfo.Timeout = TimeSpan.FromMinutes(180);
81         client.OperationTimeout = TimeSpan.FromMinutes(180);
82         client.Connect();
83         bool connected = client.IsConnected;
84         if (connected == true)
85             MessageBox.Show("SFTP Bağlantısı Sağlandı.");
86
87         client.ChangeDirectory(remoteDir);
88         //MessageBox.Show(client.WorkingDirectory);
89
90         UploadDirectory(client, localDir, remoteDir);
91         MessageBox.Show("Dosyanız Client ile Paylaşılmıştır.");
92
93         client.Disconnect();
94     }
95 }

```

Şekil 11. Xaml Üzerindeki Buton Click Eventleri

```

private void DownloadDirectory(SftpClient client, string remotePath, string localPath)
{
    if (lblFilePath.Content.ToString() == "")
    {
        MessageBox.Show("Lütfen Paylaşım Yapacağınız Dosyayı Seçiniz!!!");
    }
    try
    {
        Directory.CreateDirectory(localPath);
        IEnumerable<SftpFile> files = client.ListDirectory(remotePath); //lblFilePath2.Content.ToString()
        string destFilePath = System.IO.Path.Combine(localPath, lblFilePath.Content.ToString());

        string a = "/home/hkn/Desktop/" + lblFilePath.Content.ToString();
        using (Stream fileStream = File.Create(destFilePath))
        {
            if (lblFilePath.HasContent)
                client.DownloadFile(a, fileStream);
        }
        this.Close();
    }
    catch (Exception ex)
    {
        throw;
    }
}

```

Şekil 12. Treeview Seçilene Göre Dosya İndirme Yükleme

```

2 references
private void UploadDirectory(SftpClient client, string localPath, string remotePath)
{
    if (lblFilePath.Content.ToString() == "")
    {
        MessageBox.Show("Lütfen Paylaşım Yapacağınız Dosyayı Seçiniz!!!");
    }
    try
    {
        IEnumerable<FileSystemInfo> infos = new DirectoryInfo(localPath).EnumerateFileSystemInfos(lblFilePath.Content.ToString());
        foreach (FileSystemInfo info in infos)
        {
            if (info.Attributes.HasFlag(FileAttributes.Directory))
            {
                string subPath = remotePath + "/" + info.Name;
                if (!client.Exists(subPath))
                {
                    client.CreateDirectory(subPath);
                }
                UploadDirectory(client, info.FullName, remotePath + "/" + info.Name);
            }
            else
            {
                using (Stream fileStream = new FileStream(info.FullName, FileMode.Open))
                {
                    Console.WriteLine("Uploading {0} ({1:N0} bytes)", info.FullName, ((FileInfo)info).Length);

                    client.UploadFile(fileStream, remotePath + "/" + info.Name);
                }
            }
        }
        this.Close();
    }
    catch (Exception ex)
    {
        throw;
    }
}

```

```

1 reference
private void treeView_SelectedItemChanged(object sender, RoutedPropertyChangedEventArgs<object> e)
{
    selected_item = treeView.SelectedItem as TreeViewItem;
    lblFilePath.Content = selected_item.Header.ToString();
}

```

Şekil 13. Treeviewda Seçilenin Header İsmine Göre İşlemler Gerçekleştiriliyor



# MESAJ GÖNDER EKRANI

```
0 references
public ObservableCollection<string> Ips { get; set; } = new ObservableCollection<string>()
{
    "192.168.139.129",
    "IP2",
    "IP3",
    "IP4"
};
3 references
public string SelectedIps
{
    get { return _sp; }
    set { _sp = value; NotifyPropertyChanged("SelectedIps"); }
}
3 references
public string PortSelect
{
    get { return _tp; }
    set { _tp = value; NotifyPropertyChanged("PortSelect"); }
}
3 references
public string MesajTextBox
{
    get { return _mesajTextBox; }
    set { _mesajTextBox = value; NotifyPropertyChanged("MesajTextBox"); }
}
0 references
public ICommand TcpbaglaButton
{
    get
    {
        _tcpBaglaButton = new RelayCommand(param => this.TcpBaglanButton());
        return _tcpBaglaButton;
    }
}
0 references
public ICommand MesajgonderButton
{
    get
    {
        _mesajGonderButton = new RelayCommand(param => this.MesajGonderButton());
        return _mesajGonderButton;
    }
}
```

Şekil 14. Button Command Ve Data Binding İşlemleri

```
1 reference
private void TcpBaglanButton()
{
    if (PortSelect == null || SelectedIps == null)
    {
        MessageBox.Show("Bilgileri Eksiksiz Giriniz!");
    }
    else
    {
        int PORT = Convert.ToInt32(PortSelect);
        string uzakBilgisayarIp = SelectedIps.ToString();

        try
        {
            this.soket.Connect(new IPEndPoint(IPAddress.Parse(uzakBilgisayarIp), PORT));
            MessageBox.Show("Başarıyla Bağlanıldı. Mesajınızı Girebilirsiniz.");
        }
        catch (Exception ex)
        {
            MessageBox.Show("\n (X) -> Bağlanmaya çalışırken hata oluştu: " + ex.Message);
            Application.Current.MainWindow.Close();
        }
    }
}
1 reference
private void MesajGonderButton()
{
    if (this.soket.Connected)
    {
        string gonder = MesajTextBox.ToString();
        this.soket.Send(Encoding.UTF8.GetBytes(gonder));
        MesajTextBox = "";
        if (gonder == "exit")
        {
            this.soket.Shutdown(SocketShutdown.Both);
            this.soket.Close();
            MessageBox.Show("Bağlantı Koparıldı.");
            MesajTextBox = "";
            PortSelect = "";
            SelectedIps = "";
            Application.Current.MainWindow.Close();
        }
    }
    else
    {
        MessageBox.Show("Bağlantınız Yok");
    }
}
```

Şekil 15. TCP Bağlantısı ve Mesaj Gönderme

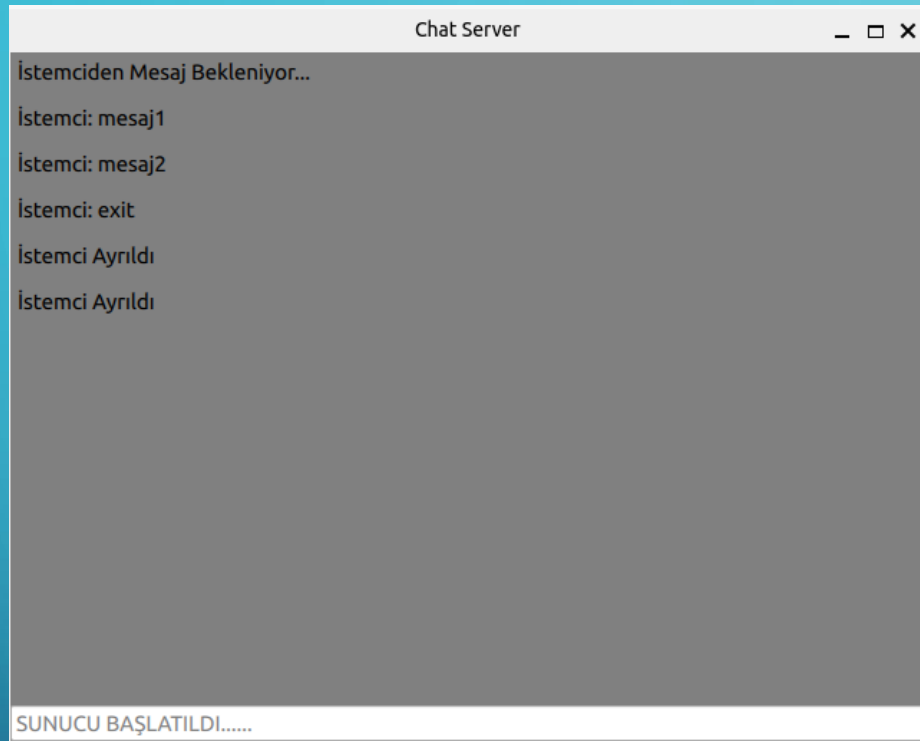
**TCP (Transmission Control Protocol)** : Bilgisayarlar arasındaki iletişimin, küçük paketler hâlinde ve kayıpsız olarak gerçekleştirilmesini sağlayan bir protokoldür. Aslında TCP protokolünün en önemli özelliği veriyi karşı tarafa gönderirken veya alırken verinin bütünlüğünü sağlamasıdır. Projede kullanılan SSH ve SFTP protokoller TCP kullanır. Bu projede sunucu Linux tarafı olarak yapıldı çünkü mesajı gönderecek olan Windows bilgisayardır. C++ uygulaması sunucuyu başlatır ve listen() ile mesajı bekler.

```
public MesajGonderViewModel()
{
    Socket soket2 = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    this.soket = soket2;
}
```

```
#include "TcpServer.hpp"

TcpServer::TcpServer(QObject *parent) : QObject(parent)
{
    connect(&_amp;_server, &QTcpServer::newConnection, this, &TcpServer::onNewConnection);
    connect(this, &TcpServer::newMessage, this, &TcpServer::onNewMessage);
    if(_server.listen(QHostAddress::Any, 45000)) {
        qInfo() << "İstemci Bekleniyor ...";
    }
}
```

Şekil 16. C++ Uygulaması Port Üzerinden Sunucuyu Başlatıp Dinlemeye Başlar



```
10 }
11
12 void TcpServer::onNewConnection()
13 {
14     const auto client = _server.nextPendingConnection();
15     if(client == nullptr) {
16         return;
17     }
18
19     qInfo() << "Yeni İstemci Bağlı.";
20
21     //_clients.insert(this->getClientKey(client), client);
22
23     connect(client, &QTcpSocket::readyRead, this, &TcpServer::onReadyRead);
24     connect(client, &QTcpSocket::disconnected, this, &TcpServer::onClientDisconnected);
25 }
26
27 void TcpServer::onReadyRead()
28 {
29     const auto client = qobject_cast<QTcpSocket*>(sender());
30
31     if(client == nullptr) {
32         return;
33     }
34
35     const auto message = "İstemci: " + client->readAll();
36
37     emit newMessage(message);
38 }
39
40 void TcpServer::onClientDisconnected()
41 {
42     const auto client = qobject_cast<QTcpSocket*>(sender());
43
44     if(client == nullptr) {
45         return;
46     }
47     const auto message = "İstemci Ayrıldı";
48     emit newMessage(message);
49     qInfo() << "İstemci Ayrıldı";
50     //_clients.remove(this->getClientKey(client));
51 }
52
53 void TcpServer::onNewMessage(const QByteArray &ba)
54 {
55     for(const auto &client : qAsConst(_clients)) {
56         client->write(ba);
57         client->flush();
58     }
59 }
60
```

# WINDOWS PERFORMANS EKRANI

Wpf proje içerisinde winform çağırılarak oluşturulan ekrandır. Windows bilgisayarın Performansını göstermek için performanceCounter() ve timer() kullanılmıştır.

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10 using MetroFramework.Forms;
11 using MetroFramework;
12
13 namespace MvvmDeneme
14 {
15     5 references
16     public partial class Manage : MetroFramework.Forms.MetroForm
17     {
18         1 reference
19         public Manage()
20         {
21             InitializeComponent();
22         }
23         1 reference
24         private void timer_Tick(object sender, EventArgs e)
25         {
26             float fcpu = pCPU.NextValue();
27             float fram = pRAM.NextValue();
28             metroProgressBar1CPU.Value = (int)fcpu;
29             metroProgressBar2RAM.Value = (int)fram;
30             lblCPU.Text = string.Format("{0:0.00}%", fcpu);
31             lblRAM.Text = string.Format("{0:0.00}%", fram);
32             chart1.Series["CPU"].Points.AddY(fcpu);
33             chart1.Series["RAM"].Points.AddY(fram);
34         }
35         1 reference
36         private void Manage_Load(object sender, EventArgs e)
37         {
38             timer.Start();
39         }
40     }
41 }
```

Şekil 17. WinForm Uygulaması

**PerformanceCounter:** sistem üzerinde çalışan tüm işlemlerin (process), işlemcinin (cpu), belleğin (ram), servislerin vs. durumunu öğrenebileceğimiz ve anlık olarak verilerini alabileceğimiz bileşendir.

**Timer:** Zamanlayıcı olarak kullanılan .NET komponentidir. Programlamada zamanlama ölçümü gerektiren her durumda kullanılabilir. İstenen komutların belirlenen zaman aralığında tekrarlanmasını sağlamak için kullanılmaktadır. Tekrarlanmasını istediğimiz kodları **Timer\_Tick** olayına yazarız. İnterval özelliği kodlarımızın ne kadar sürede tekrarlanacağını belirlediğimiz özelliktir. Milisaniye cinsinden değer verilir.

Properties	
pCPU System.Diagnostics.PerformanceCounter	
[ApplicationSettings]	
(Name)	pCPU
CategoryName	Processor
CounterName	% Processor Time
GenerateMember	True
InstanceLifetime	Global
InstanceName	_Total
MachineName	.
Modifiers	Private
ReadOnly	True

Properties	
timer System.Windows.Forms.Timer	
[ApplicationSettings]	
(Name)	timer
Enabled	False
GenerateMember	True
Interval	200
Modifiers	Private
Tag	



# SANAL MAKİNE(LINUX) PERFORMANS EKRANI

```
public partial class LinuxCpuMem : MetroFramework.Forms.MetroForm
{
    Thread t1;
    string result = string.Empty;
    List<DateTime> TimeList = new List<DateTime>();
    //reference
    public LinuxCpuMem(SshClient sshClient)
    {
        InitializeComponent();
        t1 = new Thread(() => KomutCalistir(sshClient));
        t1.Start();
        myChart.ChartAreas[0].AxisX.LabelStyle.Format = "hh:mm";
        panelChart.AutoScroll = true;
    }
    //reference
    public void KomutCalistir(SshClient sshClient)
    {
        while ((sshClient.IsConnected))
        {
            //Thread.Sleep(500);
            SshCommand cmd = sshClient.RunCommand("top -n 1 -b");
            result = cmd.Result;

            string cpuDegeri = ParseCpu(result);
            string cpu = cpuDegeri.Remove(0, 9);
            double cpuInt = Double.Parse(cpu, CultureInfo.InvariantCulture);

            int width = myChart.Width;
            int height = myChart.Height;

            DateTime now = DateTime.Now;
            TimeList.Add(now);
            myChart.Size = new Size(width++, height++);

            if (InvokeRequired)
            {
                BeginInvoke(new Action(() =>
                {
                    lblCpu.Text = cpuDegeri;
                    lblRam.Text = ParseMemory(result);
                    //myChart.ChartAreas[0].AxisX.Maximum = 100;
                    myChart.Series["CPU"].Points.AddXY(now, cpuInt/10);
                }));
            }
            else
            {
                t1.Abort();
            }
        }
    }
}
```

Şekil 18. Thread ile Komut Çalıştır Metodu

Ssh bağlantısı ile komut, thread ile çalıştırılarak CPU ve RAM bilgileri alınır.

WPFde ve Winformda temel olarak geri planda 2 thread bulunmakta; birisi rendering için diğeri UI'ı yönetmek için kullanılır. Rendering thread UI'a veri getirirken, ekrana çizim yaparken, eventler'i handle ederken devreye girer. UI Thread'i ise genellikle 1 tane olur, bazı durumlarda birden fazla da olabilir.

Yeni açılan ekranda grafik güncellemesi ve labellerin güncellenmesi için UI thread'inden farklı yeni bir thread açıldı. Dispatcher yani threadların senkron problemini çözmek için de winformda kullanılan InvokeRequired ve BeginInvoke kullanıldı label ve grafiğin sürekli güncellenmesi sağlandı.

InvokeRequired: Forma gelen talebin farklı bir iş parçacığından gelip gelmediğini kontrol eder.

Eğer farklı bir iş parçacığından talep gelmişse aşağıdaki BeginInvoke metoduyla işlem gerçekleştiriliyor.

```

public string ParseMemory(string contentOfOutput)
{
    var line = contentOfOutput.Split('\n');
    string cpuLine;
    cpuLine = line[3];
    var wordCpu = cpuLine.Substring(0, 54);
    return wordCpu;
}
1 reference
public string ParseCpu(string contentOfOutput)
{
    var line = contentOfOutput.Split('\n');
    string cpuLine;
    cpuLine = line[2];
    var wordCpu = cpuLine.Substring(0, 13);
    return wordCpu;
}

```

Şekil 19. Komutun Parse Edilmesi

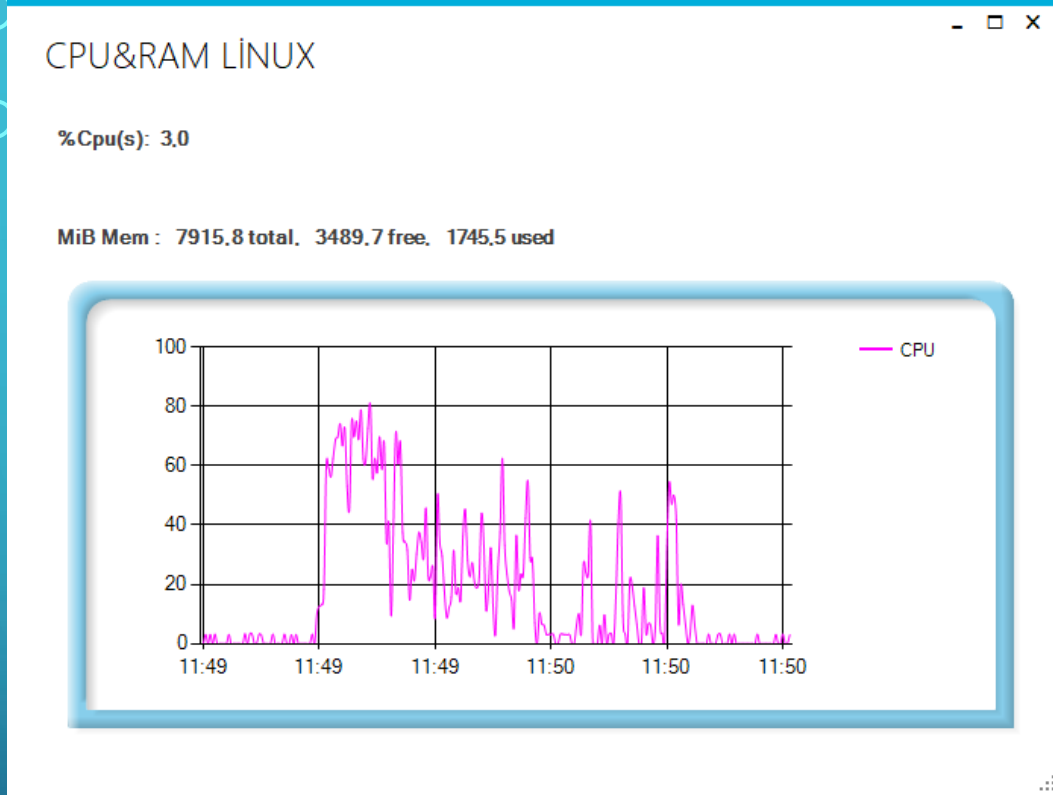
```

hkn@root: ~
hkn@root:~$ top -n 1 b
top - 11:45:21 up 1 day, 2:53, 1 user, load average: 0,07, 0,02, 0,00
Tasks: 297 total, 1 running, 296 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,0 us, 3,1 sy, 0,0 ni, 96,9 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 7915,8 total, 3485,9 free, 1756,6 used, 2673,3 buff/cache
MiB Swap: 2048,0 total, 2048,0 free, 0,0 used. 5817,5 avail Mem

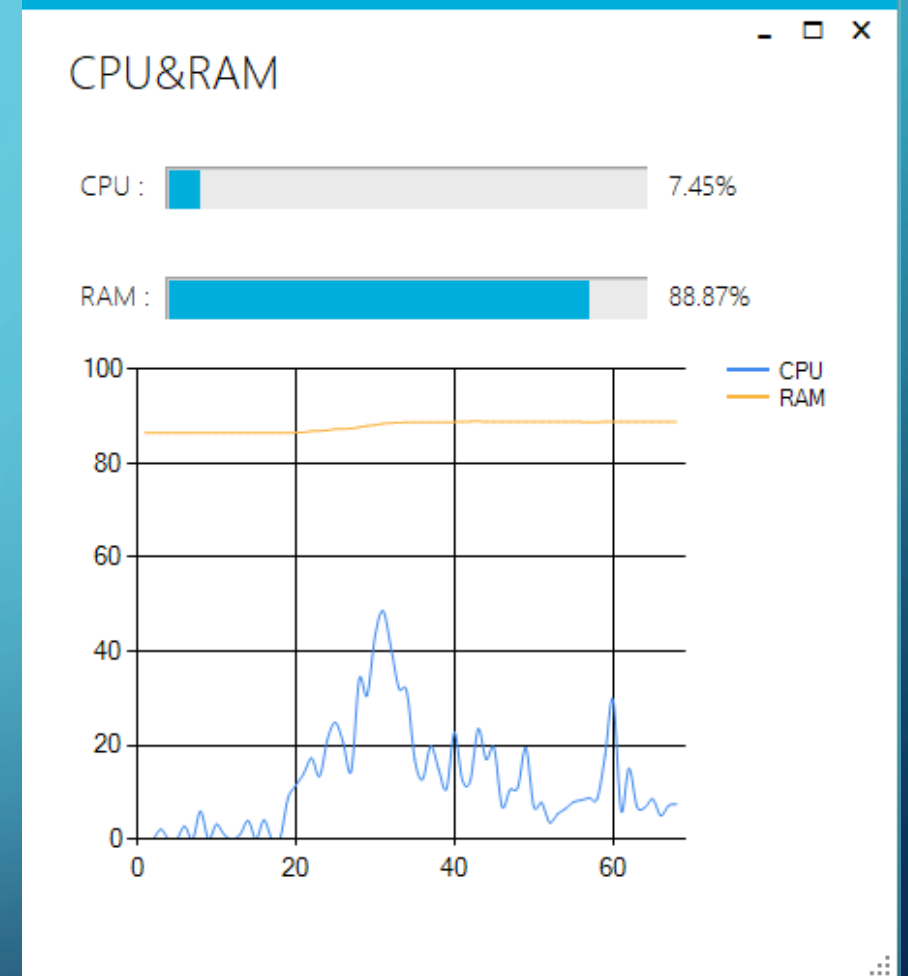
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 2021 hkn        20   0 4411244 290172 125548 S   13,3   3,6   3:02.45 gnome-s+
 2230 hkn        20   0 350556 28892  18180 S    6,7   0,4   0:00.99 ibus-ex+
    1 root        20   0 168084  13200   8152 S    0,0   0,2   0:05.11 systemd
    2 root        20   0      0      0      0 S    0,0   0,0   0:00.04 kthreadd
    3 root         0 -20      0      0      0 I    0,0   0,0   0:00.00 rcu_gp
    4 root         0 -20      0      0      0 I    0,0   0,0   0:00.00 rcu_par+
    5 root         0 -20      0      0      0 I    0,0   0,0   0:00.00 netns
    7 root         0 -20      0      0      0 I    0,0   0,0   0:00.00 kworker+
    9 root         0 -20      0      0      0 I    0,0   0,0   0:02.13 kworker+
   10 root         0 -20      0      0      0 I    0,0   0,0   0:00.00 mm_perc+
   11 root        20   0      0      0      0 S    0,0   0,0   0:00.00 rcu_tas+
   12 root        20   0      0      0      0 S    0,0   0,0   0:00.00 rcu_tas+
   13 root        20   0      0      0      0 S    0,0   0,0   0:00.83 ksoftir+
   14 root        20   0      0      0      0 I    0,0   0,0   0:13.36 rcu_sch+
   15 root         rt   0      0      0      0 S    0,0   0,0   0:00.38 migrati+
   16 root       -51   0      0      0      0 S    0,0   0,0   0:00.00 idle_in+

```

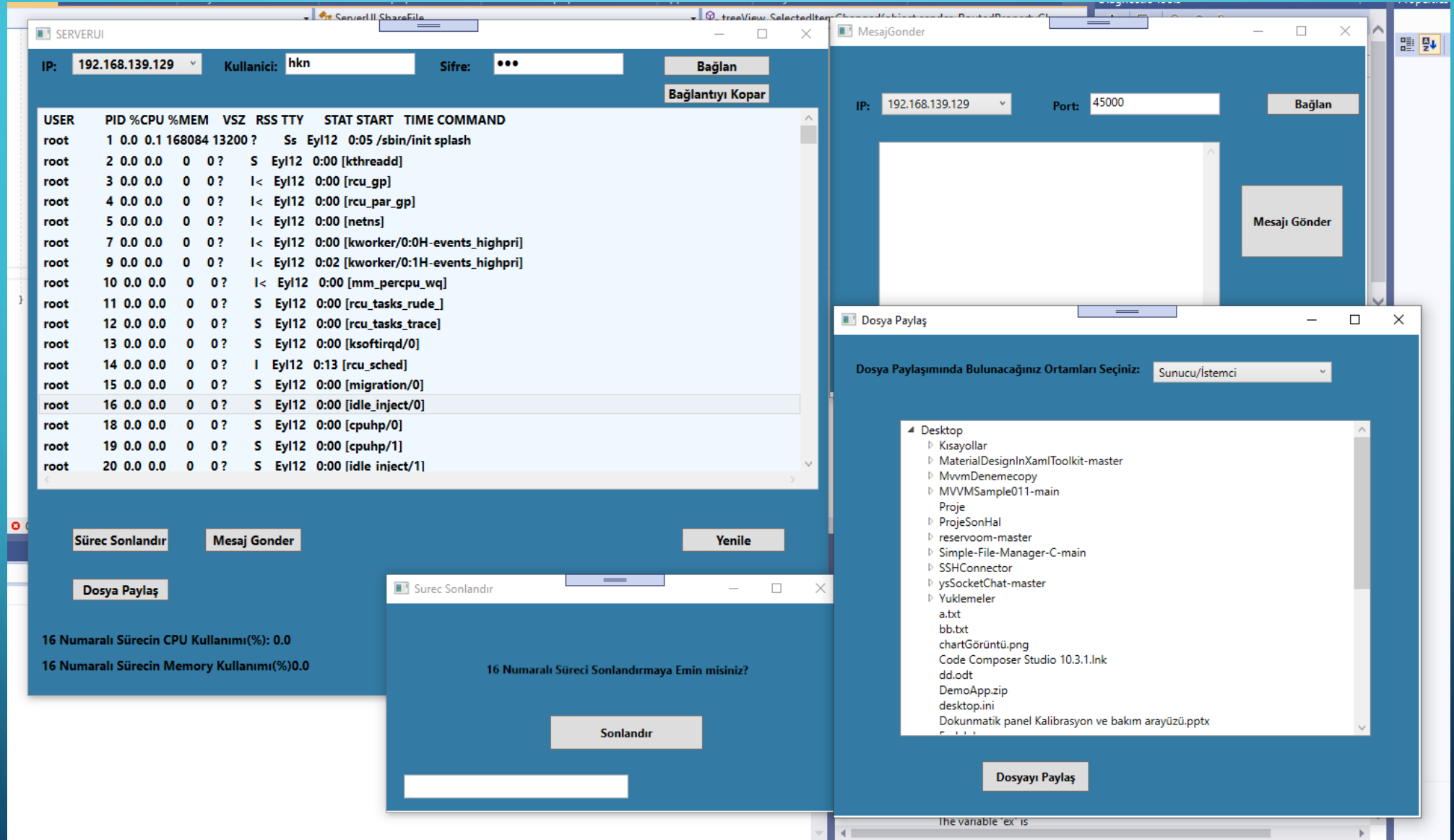
Şekil 20. top -n 1 b Komutu



Şekil 21. Tablet Performans Grafiği

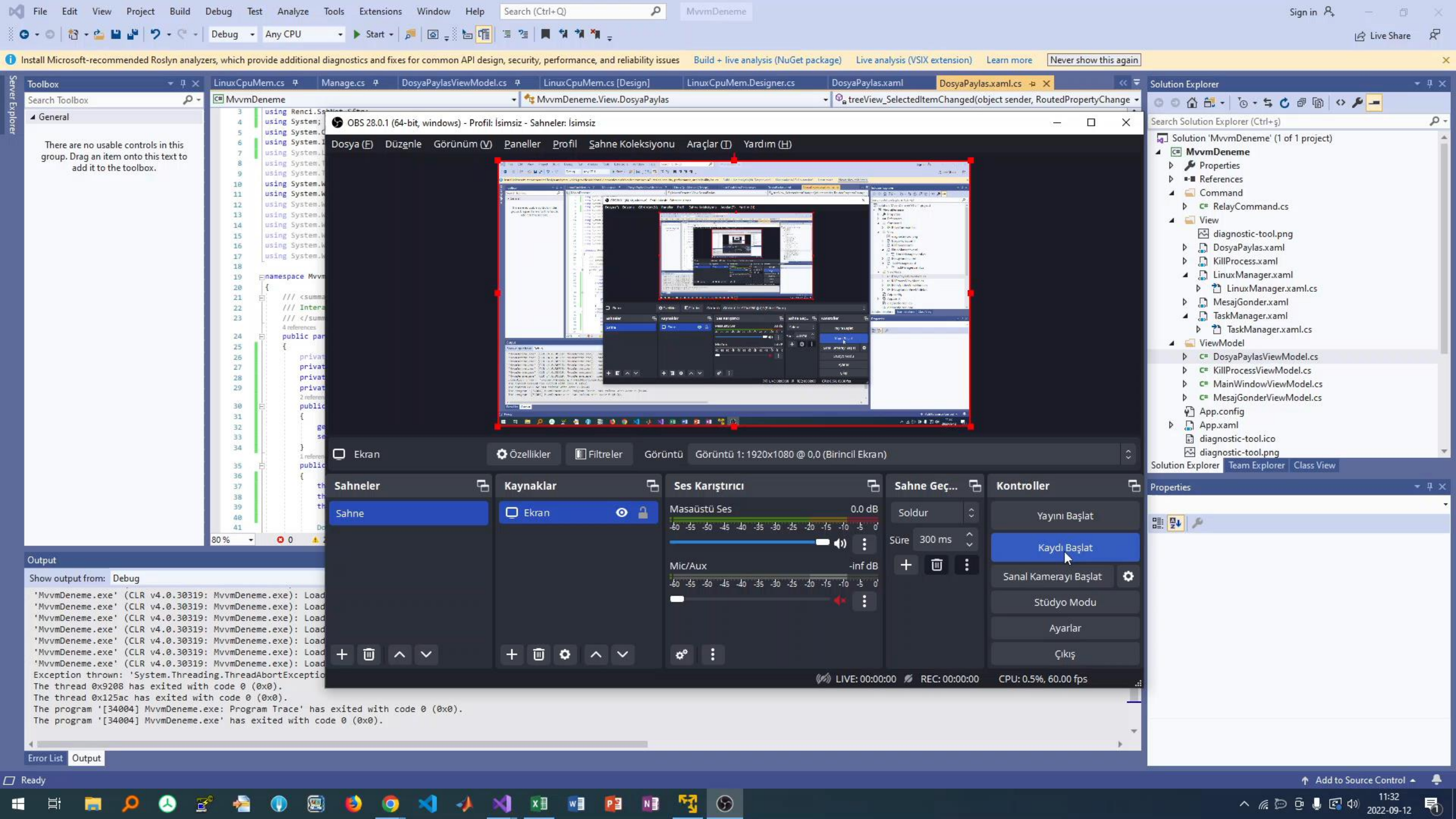


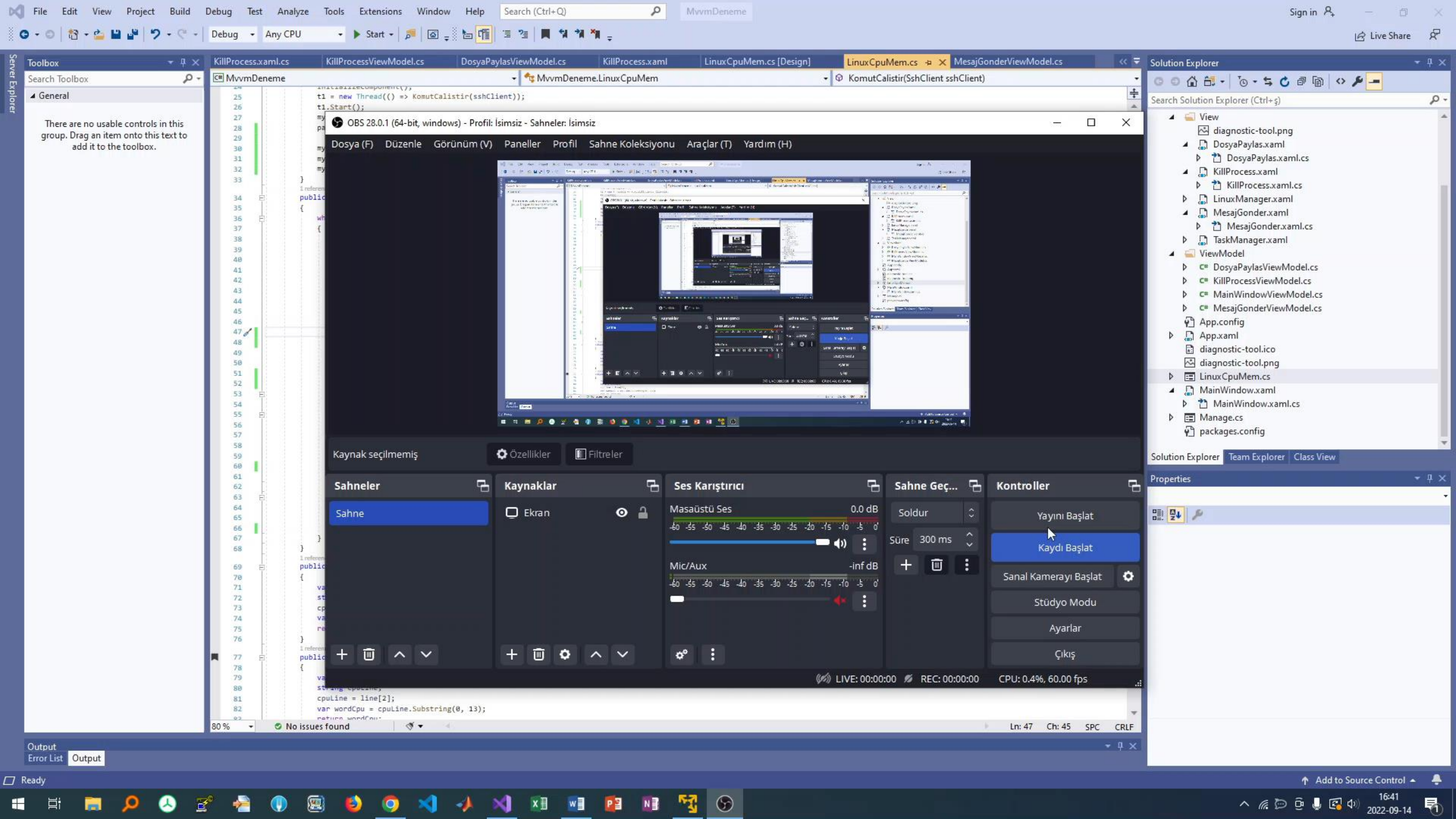
Şekil 22. Windows Performans Grafiği



Şekil 23. Proje İlk Hali







TEŞEKKÜRLER