Chat Sistemi | Yazılı Sohbet Sistemi

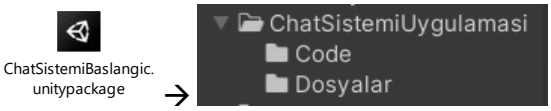
1)Chat sistemi anlatım | Kurulum | Paketleri Dahil Etme | İlk işlemler

Bu bölümümüzde Chat sisteminin nasıl gerçekleştiğini inceleyelim. Chat sisteminin yapısı Photon sisteminden farklı. Chat sisteminin mantığı Photon'daki gibi aynı işlevi yaparken yapısı ve kodları bakımından farklılık içeriyor.

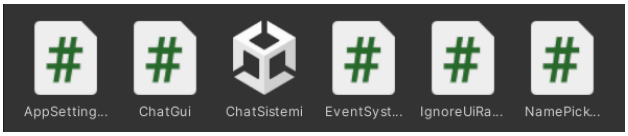
Chat sisteminin yapımına başlayalım:

1-Photon klasörü içerisinde örnek bir chat sistemi mevcut. Buradaki kodlar ve yöntemleri kullanarak kendi sistemimizi kuracağımız için kaldırmanı tavsiye ederim.

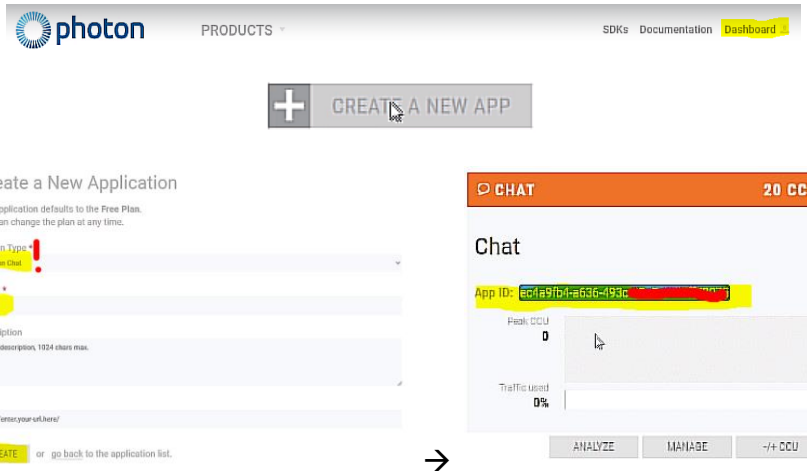
2-Silem işlemini tamamladıktan sonra, aşağıdaki package dosyasını projemize import ediyoruz.



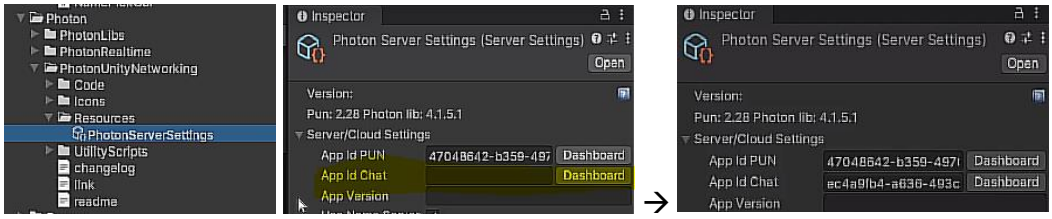
3- Örnek chat scene açıyoruz.



4- PhotonNetwork'e bağlanmak için sistem üzerinden bir api key almıştık. Şimdi Chat için ise aynı sistemden bir api key almamız daha gerekiyor. Chat işlemlerini gerçekleştirebilmek için.



5-Chat için aldığımız key aşağıdaki ayarlar kısmına eklememiz gerekiyor. Ekledikten sonra direkt oyunu çalıştırdığımızda herhangi bir hata ile consol kısmında karşılaşmıyorsak, projemiz sorunsuz çalışıyor demektir.



6-Photon Chat sisteminin bize sundukları.

!Photon sisteminde kanal sistemi mevcut. İster oyuncuları tek bir kanal üzerinden konuşturmak mümkünken, istersek de genel sohbet, oyun teknik sohbet gibi farklı kanallardan konuşmayı da photon chat sistemi bize sağlar.

!!Photon chat ücretli sürümünde sınırsız sayıda kullanıcıya sınırsız sayıda kanala izin veriyor.

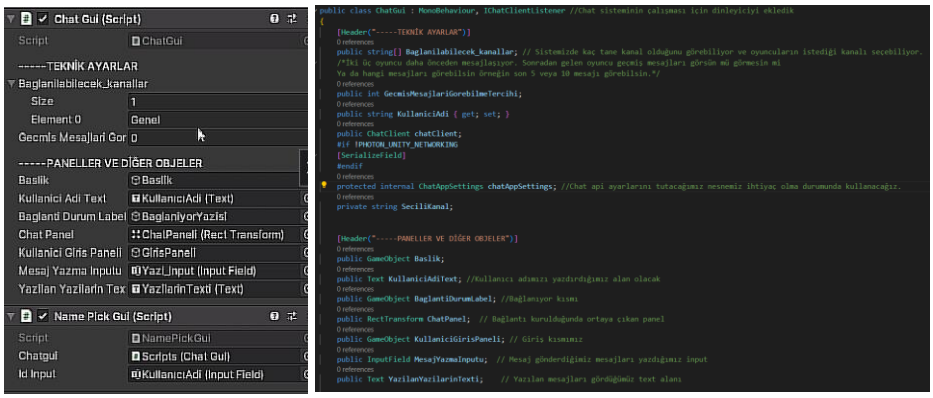
!!!Gönderilen mesajlar şifreli olarak gitmekte böylece güvenlik açısından da daha karlı bir işlem.

!!!!Mesajlarda ara bellek sistemi mevcut böylece senkronizasyonu sağlamada ve daha hızlı iletişim kurmayı sağlar.

!!!!!Photon chat sisteminde arkadaşlık sistemi mevcut. İstediğimiz oyuncuyu arkadaş olarak ekleyerek istediğimiz bilgileri ondan alabiliyoruz.

!!!!!!Bu sistemde kötü kelime filtre de mevcut. Ve çok dilli olarak destekliyor. Kötü kelimeleri alabiliyoruz. Ayrıca görüntüleri ve html tagları (linkleri) de filtreleme imkanı sağlar.

7- Sonrasında ise boş bir klasörün içerisine ChatGui scriptimizi atalım.



8-NamePickGui script ise mesaj bağlantısını ilk tetikleyen alan, bağlantıyı kurduktan sonra kendisini kapatacak. Bu dosyamızı da oluşturduğumuz objeye atıyoruz.

9-Bazı fonksiyon bilgileri

```
/*Mevcut olan default olarak kullanılan bir fonksiyon. Çok önemli bir fonksiyon.
DebugReturn: Mevcut olan hatalarda bağlanma kesilmesi vs. ya da konuşma bilgilerini
almak için burası önemlidir.
İçeriğine harhangi bir müdahalede bulunamaya gerek yoktur.*/
50 references
public void DebugReturn(ExitGames.Client.Photon.DebugLevel level, string message)
{
if (level == ExitGames.Client.Photon.DebugLevel.ERROR)
    {
        Debug.LogError(message);
    }
    else if (level == ExitGames.Client.Photon.DebugLevel.WARNING)
    {
        Debug.LogWarning(message);
    }
    else
    {
        Debug.Log(message);
    }
}
```

```
/*OnChatStateChange: Oyuncuların durumlarında offline olma online olma veya meşgul olma
durumlarını yakaladığımız ve işlem yapabileceğimiz alan */
4 references
public void OnChatStateChange(ChatState state)
{
    // sohbet durumu değiştiğinde buradan yakalanabilir.

}
```

```
/*Bir kanala bağlandığımızda bu fonksiyonumuz çalışacak*/
1 reference
public void OnSubscribed(string[] channels, bool[] results)
{
    Debug.Log("Bir Kanala Bağlanıldı : " + string.Join(", ", channels));

    ShowChannel(channels[0]);
}
```

```
/*OnSubscribed: Burada da bir kanala bağlandığımızda çalışan fonksiyondur. Bir öncekiden farkı
daha detaylı bilgi alabiliyoruz.*/
0 references
public void OnSubscribed(string channel, string[] users, Dictionary<object, object> properties)
{
    Debug.LogFormat("Kanal : {0}, Kişi sayısı: {1} Kanal Özellikleri: {2}.", channel, users.Length, properties.ToStringFull())
}
```

```
/*OnUnsubscribed: hehangi bir kanaldan ayrıldığımızda çalışan fonksiyondur.
!Ayrıca genel olarak biz listener eklediğimiz için bazı fonksiyonların içerisini doldurma
zorunluluğu olmasa bile bulunmak zorundadır.*/
1 reference
public void OnUnsubscribed(string[] channels)
{


    foreach (string channelName in channels)
    {

        Debug.Log("Kanaldan Ayrıldın :" + channelName + "'.");

    }
}
```

```
/*Eğer arkadaş listemiz var ise arkadaş listesi durumlarını buradan alabiliyoruz.*/
1 reference
public void OnStatusUpdate(string user, int status, bool gotMessage, object message)
{
    /// Arkadaş listemizde ki kişilerin durumlarla ilgili güncellenmelerini alabiliriz.
}
```

```
/*Kanala dahil olan kişileri alabiliyoruz.*/
1 reference
public void OnUserSubscribed(string channel, string user)
{
    Debug.LogFormat("Kanala Giren kişiler : channel=\"{0}\" userId=\"{1}\"", channel, user);
}
```

```
/*Kanaldan çıkan kişileri alabiliyoruz..*/
1 reference
public void OnUserUnsubscribed(string channel, string user)
{
    Debug.LogFormat("Kanaldan Çıkan kişiler : channel=\"{0}\" userId=\"{1}\"", channel, user);
}
```

```
/*Kanalın özelliği değişirse.*/
0 references
public void OnChannelPropertiesChanged(string channel, string userId, Dictionary<object, object> properties)
{
    Debug.LogFormat("Kanal Özellikleri Değiştirildi : {0} by {1}. Props: {2}.", channel, userId, Extensions.ToStringFull(properties));
}
```

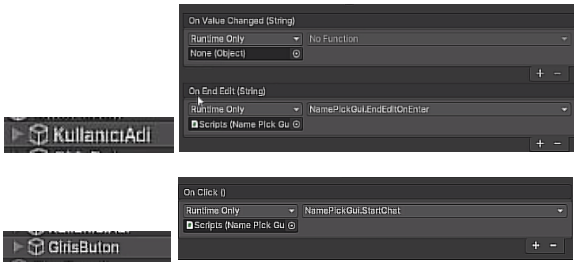
```
/*Kanalın içerisindeki oyuncuların özellikleri değişirse*/
0 references
public void OnUserPropertiesChanged(string channel, string targetUserId, string senderUserId, Dictionary<object, object> properties)
{
    Debug.LogFormat("Kullanıcı Özellikleri Değiştirildi : (channel:{0} user:{1}) by {2}. Props: {3}.",
     channel, targetUserId, senderUserId, Extensions.ToStringFull(properties));
}
```

```
/*Eğer hata alırsak, hangi kanalda ne tür hata var bunu alabiliyoruz.*/
0 references
public void OnErrorInfo(string channel, string error, object data)
{
    Debug.LogFormat("OnErrorInfo for channel {0}. Error: {1} Data: {2}", channel, error, data);
}
```

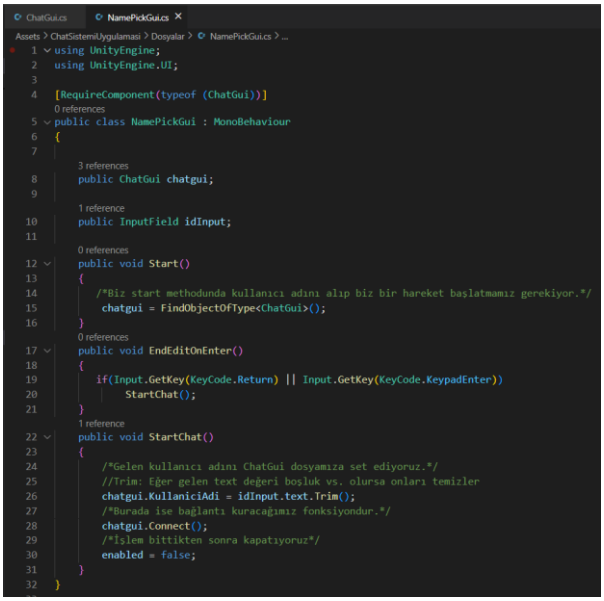
2)Chat Script dosyasının yazılması

1- Sistemin nasıl çalıştığı hakkında genel bir bilgi sahibi olduk. Şimdi ise, script dosyalarını doldurma işlemini yapacağız. Ama öncelikle, NamePickGui.cs script dosyamızı dolduralım. İlk tetiklemeyi yapalım.

Biz inputfield içerisinde ismimizi yazdıktan sonra ister entere basalım istersek butona basalım tetikleme işlemi yapmasını istediğimizi söylüyoruz.

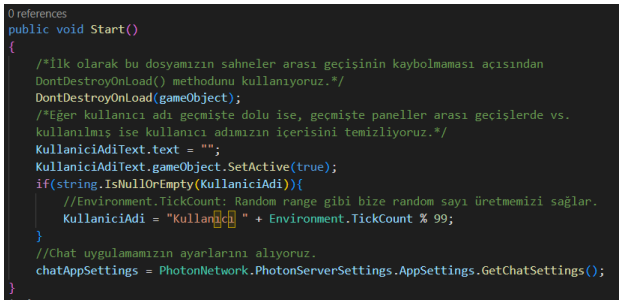


Sonrasında script dosyamıza geçelim.

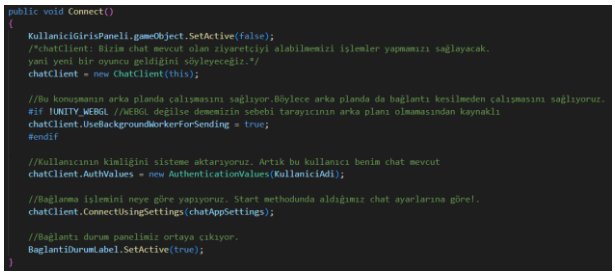


2-Artık ana script dosyamızdaki işlemleri yapıyoruz. (ChatGui)

1.İlk olarak Start fonksiyonu ile script dosyamızı yazmaya başlıyoruz.



2.Sonrasında Connect yani bağlantı işlemlerimize geçiyoruz.

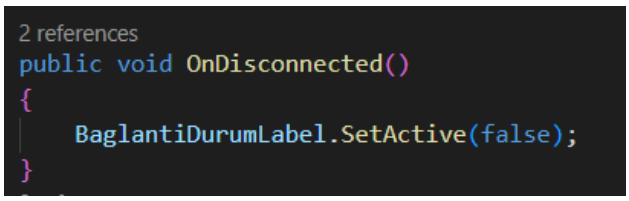


```
public void Connect()
{
    KullaniciGirisPaneli.gameObject.SetActive(false);
    /*chatClient: Bizim chat mevcut olan ziyaretçiyi alabilmemizi işlemler yapmamızı sağlayacak.
    yani yeni bir oyuncu geldiğini söyleyeceğiz.*/
    chatClient = new ChatClient(this);

    //Bu konuşmanın arka planda çalışmasını sağlıyor.Böylece arka planda da bağlantı kesilmeden çalışmasını sağlıyoruz.
    #if !UNITY_WEBGL //WEBGL değilse dememizin sebebi tarayıcının arka planı olmamasından kaynaklı
    chatClient.UseBackgroundWorkerForSending = true;
    #endif

    //Kullanıcının kimliğini sisteme aktarıyoruz. Artık bu kullanıcı benim chat mevcut
    chatClient.AuthValues = new AuthenticationValues(KullaniciAdi);

    //Bağlanma işlemini neye göre yapıyoruz. Start methodunda aldığımız chat ayarlarına göre!.
    chatClient.ConnectUsingSettings(chatAppSettings);

    //Bağlantı durum panelimiz ortaya çıkıyor.
    BaglantiDurumLabel.SetActive(true);
}
```

3.Evet sisteme bağladık şimdi ise kanala bağlama işlemi yapacağız.

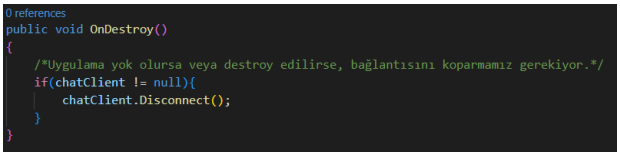
```
public void OnConnected()
{
    /*Eğer tek kanalım var ise*/
    //Eğer en az bir tane kanalım var ise kanalıma bağla
    if(BaglanilabilecekKanallar != null && BaglanilabilecekKanallar.Length > 0){
        chatClient.Subscribe(BaglanilabilecekKanallar);
        //Eklemeye devam edeceğiz.
    }
    BaglantiDurumLabel.SetActive(false);
    KullaniciAdiText.text = "Bağlanan Kişi: " + KullaniciAdi;
    ChatPanel.gameObject.SetActive(true);
    /*Bizimle arkadaş olan oyuncuların bizim durumlarımızı görebilmeleri için SetOnlineStatus diyerek
    online olduğumuzu belirtiyoruz. yanına ise bir mesaj bırakarak, arkadaşlarımız bizim durumlarımızı
    ve mesajlarımızı görebilecek*/
    chatClient.SetOnlineStatus(ChatUserStatus.Online,"Ben geldim beyler");
}
```

4.Eğer bağlantımız koparsa OnDisconnected,



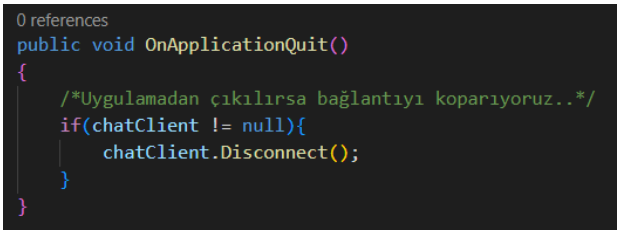
```
2 references
public void OnDisconnected()
{
    BaglantiDurumLabel.SetActive(false);
}
```

5.Uygulama yok olma veya kapatılma durumunda



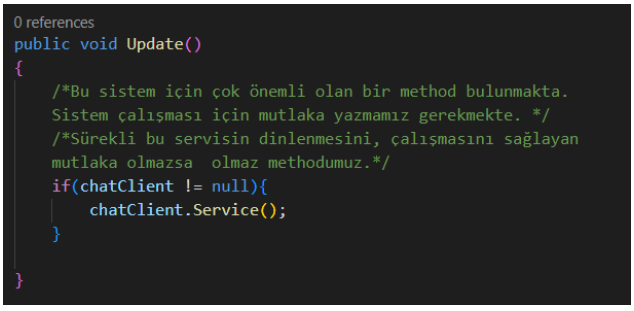
```
0 references
public void OnDestroy()
{
    /*Uygulama yok olursa veya destroy edilirse, bağlantısını koparmamız gerekiyor.*/
    if(chatClient != null){
        chatClient.Disconnect();
    }
}
```

6.Uygulama kapatıldığında



```
0 references
public void OnApplicationQuit()
{
    /*Uygulamadan çıkılırsa bağlantıyı koparıyoruz..*/
    if(chatClient != null){
        chatClient.Disconnect();
    }
}
```

7.Dikkat!

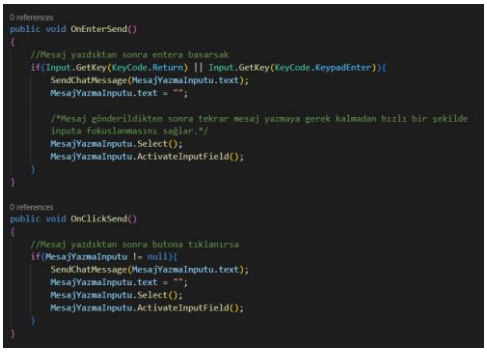


```
0 references
public void Update()
{
    /*Bu sistem için çok önemli olan bir method bulunmakta.
    Sistem çalışması için mutlaka yazmamız gerekmekte. */
    /*Sürekli bu servisin dinlenmesini, çalışmasını sağlayan
    mutlaka olmazsa  olmaz methodumuz.*/
    if(chatClient != null){
        chatClient.Service();
    }

}
```

3)Chat Script dosyasının yazılması 2

Bir önceki dersimizde bağlantı işlemlerini yaptık. Şimdi mesaj sistemimizi de yazıyoruz.

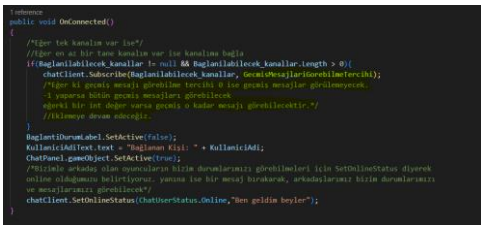
1-Mesaj yazma



2-Mesaj gönderme

3-Mesaj alma

4-Geçmiş mesajları görebilecek mi?



5- ChatPanelimizi bir tuşa atayıp, açıp kapama işlemi yapıyoruz.

4)Birden fazla kanal oluşturma | Kanal yönetimi

Genelde oyun içlerinde tek bir oda oluyor ve tek bir alanda konuşabiliyorlar. İhtiyaca göre farklı kanalların oluşturulabilmesi ve yönetebilmesini göreceğiz.

1-Bu bölümümüzde ilk olarak, kanalları ayarlayabilmek için toggle sistemini yapacağız.

Toggle Sistemi:

1.Yeni bir panel oluşturuyoruz.

2.Panelin içerisine bir Toggle Butonu oluşturuyoruz.

3.Toggle butonu içerisindeki Label'ı Bacground içerisine alıyoruz.

4.Lable = Text, Checkmark = Toggle isimlerini veriyoruz.

5.Background Toggle dışarısına yeni oluşturduğumuz panelin içerisine alıp, Toggle objesini siliyoruz. !Toggle silmeden önce toggle componentini kopyalayıp bacground içerisine atıyoruz.

6.Bacground içerisine eklediğimiz komponentin Group kısmına yeni oluşturduğumuz paneli atıyoruz.

7. KanalSecim.cs isminde yeni bir script oluşturup, background objesi içerisine atıyoruz.

8.Panel içerisindeki yerlerini konumlarını ayarlıyoruz. Toggle içerisindeki image kaldırıp sadece rengi bıraktık.

9.Background = ToggleButton, Toggle = Tercih olarak isimleri değiştirdik.

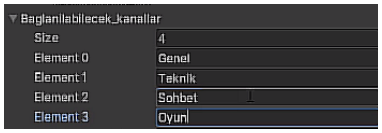
10.Ana script dosyamızda dışarıdan oluşturduğumuz ilk butonu alıyoruz.

```
public Toggle KanalToggleButton;  // Ilk butonu alıyoruz. Çünkü e kadar kanal var ise dışarıdan aldığımız butonu ürettiriceğiz.
```

2-Bir liste oluşturuyoruz. Çünkü ne kadar kanal var ise o listede o kanallar bulunacak

```
0 references
private readonly Dictionary<string, Toggle> Kanallar = new Dictionary<string, Toggle>();
```

3-Bağlanabilecek kanallar listesine 4 kanal ismini söyledik



4-Artık kodlarımıza başlayalım

```
/*Bir kanala bağlandığımızda bu fonksiyonumuz çalışacak*/
1 reference
public void OnSubscribed(string[] channels, bool[] results)
{
    /*Ilk kanala bağlandığımızda kanalları sunuyoruz*/
    foreach (string channel in channels)
    {
        if(KanalToggleButton != null){
            KanalButonuOlustur(channel);
        }
    }
    Debug.Log("Bir Kanala Bağlandık : " + string.Join(", ", channels));
    ShowChannel(channels[0]);
}
1 reference
void KanalButonuOlustur(string channelName){
    /*Gelen kanllar liste içerisinde varsa durduruyoruz.
    yok ise örnekleme işlemi yapacağız.*/
    if(Kanallar.ContainsKey(channelName)){
        return;
    }
    Toggle tgbtn = Instantiate(KanalToggleButton);
    tgbtn.gameObject.SetActive(true);
    tgbtn.GetComponentInChildren<KanalSecim>().KanalSetEt(channelName);
    //false = world position
    tgbtn.transform.SetParent(KanalToggleButton.transform.parent,false);
    Kanallar.Add(channelName,tgbtn);
}
```

AND

```
1 reference
public class KanalSecim : MonoBehaviour, IPointerClickHandler
{
    3 references
    public string _kanal;
    1 reference
    public void KanalSetEt(string Kanal){
        _kanal = Kanal;
        Text text = GetComponentInChildren<Text>();
        text.text = _kanal;
    }
    0 references
    public void OnPointerClick(PointerEventData eventData){
        ChatGui handler = FindObjectOfType<ChatGui>();
        handler.ShowChannel(_kanal);
    }
}
```

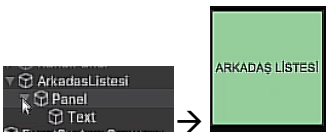
```
[Header("-----KANAL OBJELERİ")]
3 references
public Toggle KanalToggleButton;  // Ilk butonu alıyoruz. Çünkü e kadar kanal var ise dışarıdan aldığımız butonu ürettiriceğiz.
2 references
private readonly Dictionary<string, Toggle> Kanallar = new Dictionary<string, Toggle>();
```

5)Arkadaşlık sisteminin yapılması | Arkadaş ekleme & silme | Arkadaş durumlarını görme

1-Arkdadaş Listesi Oluşturma:

1.Kanal Paneli kopyalıyoruz. Ve arkadaşlar listesi isminde paneli oluşturuyoruz. Bu panelde arkadaşlarımızı göreceğiz. !Kanal panelin içerisindeki Toggle butonu sileceğiz. Çünkü bu bir liste butona şu an ihtiyaç duymuyoruz.

-Başlığımız



-Bir image oluşturuyoruz. Ve içerisine arkadaşının durumunu ve ismini görebileceği iki adet text oluşturuyoruz.

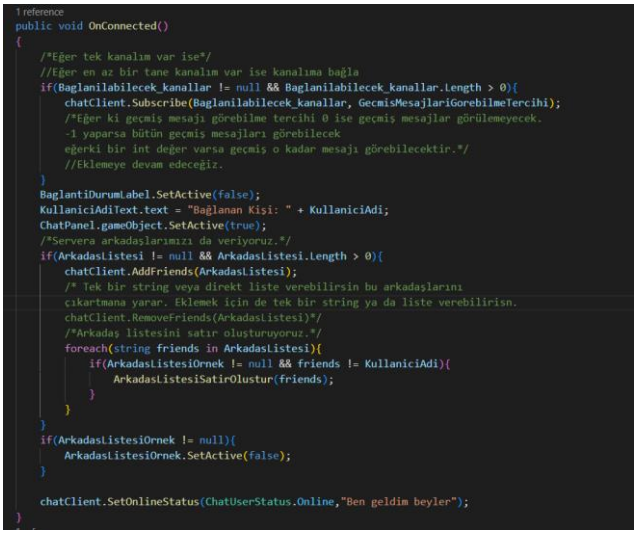
-<u>Image = ArkadaşListem</u> olarak ismini değiştiriyoruz. Ve pasif yapıyoruz. Bu bizim örnekleyeceğimiz objemiz.

Ve içerisine Arkadas.cs isminde bir script dosyası oluşturuyoruz.

2-Ana script dosyama geçiyorum, dışarıdan bu oluşturduğum yapıdaki elemanları alıyorum. Veri tabanı bağlantısı olmadığı için manuel olarak arkadaş listemi giriyorum. Ayrıca panelimizi de istersek alıp açma kapatma yapabiliriz.

```
[Header("-----ARKADAŞ LİSTESİ OBJELERİ")]
0 references
public string [] ArkadasListesi;
0 references
public GameObject ArkadasListesiOrnek;
//Arkadaş script dosyamı alıyorum.
0 references
private readonly Dictionary<string, Arkadas> ArkadasListemDC = new Dictionary<string, Arkadas>();
```

3-Akadaş listesi oluşturma işlemine bakalım.



```
1 reference
public void OnConnected()
{
    /*Eğer tek kanalım var ise*/
    //Eğer en az bir tane kanalım var ise kanalıma bağla
    if(BaglanilabilecekKanallar != null && BaglanilabilecekKanallar.Length > 0){
        chatClient.Subscribe(BaglanilabilecekKanallar, GecmisMesajlariGorebilmeTercihi);
        /*Eğer ki geçmiş mesajı görebilme tercihi 0 ise geçmiş mesajlar görülemeyecek.
        -1 yaparsa bütün geçmiş mesajları görebilecek
        eğerki bir int değer varsa geçmiş o kadar mesajı görebilecektir.*/
        //Eklemeye devam edeceğiz.
    }
    BaglantiDurumLabel.SetActive(false);
    KullaniciAdiText.text = "Bağlanan Kişi: " + KullaniciAdi;
    ChatPanel.gameObject.SetActive(true);
    /*Servera arkadaşlarımızı da veriyoruz.*/
    if(ArkadasListesi != null && ArkadasListesi.Length > 0){
        chatClient.AddFriends(ArkadasListesi);
        /* Tek bir string veya direkt liste verebilirsin bu arkadaşlarını
        çıkartmana yarar. Eklemek için de tek bir string ya da liste verebilirsn.
        chatClient.RemoveFriends(ArkadasListesi)*/
        /*Arkadaş listesini satır oluşturuyoruz.*/
        foreach(string friends in ArkadasListesi){
            if(ArkadasListesiOrnek != null && friends != KullaniciAdi){
                ArkadasListesiSatirOlustur(friends);
            }
        }
    }
    if(ArkadasListesiOrnek != null){
        ArkadasListesiOrnek.SetActive(false);
    }

    chatClient.SetOnlineStatus(ChatUserStatus.Online,"Ben geldim beyler");
}
```

```
1 reference
void ArkadasListesiSatirOlustur(string gelenarkadas){
    GameObject friendsCreate = Instantiate(ArkadasListesiOrnek);
    friendsCreate.gameObject.SetActive(true);
    Arkadas arkadasItem = friendsCreate.GetComponent<Arkadas>();
    arkadasItem.FriendsId = gelenarkadas;
    arkadasItem.DurumGuncelle(0);
    friendsCreate.transform.SetParent(ArkadasListesiOrnek.transform.parent,false);
    ArkadasListemDC[gelenarkadas] = arkadasItem;
}
```

Arkadas.cs

```
using UnityEngine.UI;
using UnityEngine.EventSystems;

5 references
public class Arkadas : MonoBehaviour
{
    [HideInInspector]
    1 reference
    public string FriendsId{
        set{
            Name.text = value;
        }
        get{
            return Name.text;
        }
    }
    /*Bu değereri ise dışarıdan alıyoruz.*/
    2 references
    public Text Name;
    1 reference
    public Text State;
    2 references
    string _status;

    2 references
    public void DurumGuncelle(int state){
        _status = state switch
        {
            1 => "Invisible",
            2 => "Online",
            3 => "Away",
            5 => "Do not disturb",
            6 => "Looking for game",
            7 => "Playing",
            _ => "Offline"
        };
        State.text = _status;
    }

}
```

```
/*Eğer arkadaş listemiz var ise arkadaş listesi durumlarını buradan alabiliyoruz.*/
1 reference
public void OnStatusUpdate(string user, int status, bool gotMessage, object message)
{
    /// Arkadaş listemizde ki kişilerin durumlarla ilgili güncellenmelerini alabiliriz.
    if(ArkadasListemDC.ContainsKey(user)){
        /*Eğer o kullanıcıya erişir isek, aşağıdaki gibi biz Arkadaş komponentini vereniliyoruz.*/
        Arkadas arkadasItem = ArkadasListemDC[user];
        if(arkadasItem != null) arkadasItem.DurumGuncelle(status);
    }
}
```

6)Arkadaşa özel mesaj gönderebilme sistemi

1-Bu dersimizde online olan arkadaşlarımıza mesaj atabileceğiz. Burada bizim için en önemli bölüm mesajı gönderdiğimiz bölümdür. Burada ilk yapacağımız işlem SendChatMessage fonksiyonuna yetenekler verip, arkadaşlarımıza mesaj göndermeyi sağlayacağız.

```
private void SendChatMessage(string inputLine)
{
    //Gönderilen mesaj
    if(string.IsNullOrEmpty(inputLine)){
        /*Gönderilen mesaj tanımsız veya boş ise kesip göndermemesini sağlıyoruz.*/
        return;
    }
    //Ekelenecek kanalı ve mesajı gönderiyoruz.
    1 reference
    chatClient.PublishMessage(SeciliKanal, inputLine);
}
```

Arkadaşa mesaj gönderme mantığı: Özel mesajlarda private yani özel korumalı olarak nitelendiriliyor. Ve daha önceden katılmasına gerek kalmıyor.

Özel mesajları SendPrivateMessage ile gönderiyor içerisine kime gönderdiğimizi ve mesajımızı belirtiyoruz. Özel mesajları alabilmek için ise OnPrivateMessage fonksiyonunu ile özel mesajları karşılayabiliriz.

```
chatClient.SendPrivateMessage("Ali","Mesaj");
```

```
1 reference
public void OnPrivateMessage(string sender, object message, string channelName)
{

}
```

2-Özel Mesaj işlemler

```
2 references
private void SendChatMessage(string inputLine)
{
    //Gönderilen mesaj
    if(string.IsNullOrEmpty(inputLine)){
        /*Gönderilen mesaj tanımsız veya boş ise kesip göndermemesini sağlıyoruz.*/
        return;
    }
    //Ekelenecek kanalı ve mesajı gönderiyoruz.
    chatClient.PublishMessage(SeciliKanal, inputLine);
    //Bir hedef isteyecek ve bir mesaj isteyecek
    //Contains: İlgili script ifadesi içerisinde herhangi bir şey aramamızı sağlar.
    bool OzelKonusmaVarmı = chatClient.PrivateChannels.ContainsKey(SeciliKanal);
    string OzelKonusulacakKisi = string.Empty;
    string Mesaj = string.Empty;
    if(OzelKonusmaVarmı){
        string[] parcalanmis = SeciliKanal.Split(new char[] {':'});
        OzelKonusulacakKisi = parcalanmis[1];
    }
    if(inputLine.Contains("\\mesaj")){
        //Mesajı parçalayıp kime gönderildiğini alacağız.
        //\\Mesaj:Ali:Naber \\Mesaj --> 0 Ali --> 1 Naber --> 2
        string[] parcalanmis = inputLine.Split(new char[] {':'});
        OzelKonusulacakKisi = parcalanmis[1];
        Mesaj = parcalanmis[2];
        chatClient.SendPrivateMessage(OzelKonusulacakKisi,Mesaj);
    }
    else{
        //Sadece ilk etapta parçalıyoruz. Sonrasında özel konuşmayı kontrol edip direkt mesajı veriyoruz.
        if(OzelKonusmaVarmı)
            chatClient.SendPrivateMessage(OzelKonusulacakKisi,inputLine);
        else
            chatClient.PublishMessage(SeciliKanal, inputLine);
    }
}
```

```
1 reference
public void OnPrivateMessage(string sender, object message, string channelName)
{
    //Gelen özel mesaj için bir buton oluşturmuş oluyoruz.
    KanalButonuOlustur(channelName);
    //Burada eğer seçili kanalımız gelen özel mesaj kanalı ise mesajları oku diyoruz.
    if(SeciliKanal.Equals(channelName)){
        ShowChannel(channelName);
    }
}
```

7)Bu bölümde kullanılan dosyalar


ChatGui.cs


NamePickGui.cs


Arkdas.cs


KanalSecim.cs