Bilgisayar Ağları Projesi

Websocket Tabanlı Mesajlaşma Uygulaması

```python
connected = set()
kisiler = []

async def server(websocket, path):
    connected.add(websocket)
    i = 0
    try:
        async for veriler in websocket:  # veriler send ile gelen her veri(client ve message)
            if veriler[0] == "_":  # Client ise
                kisiler.append(veriler)  # listeye ekle
                for conn in connected:  # Client list
                    for i in range(len(kisiler)):
                        await conn.send(kisiler[i])
                        print(kisiler[i])
                        i = i + 1
                i = 0
            else:  # Message ise
                for conn in connected:
                    if conn != websocket:
                        await conn.send(veriler)
                        print(veriler)
    finally:
        connected.remove(websocket)

start_server = websockets.serve(server, "localhost", 5000)

asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
```

Öncelikle

import asyncio

import websockets kullanılarak import edilir. Websockets kütüphanesi python terminalinde -pip install websockets yazılarak indirilip kullanılmaya başlanır.

start_server = websockets.serve(server, “localhost”, 5000) şeklinde server değişkeni tanımlanır. Burada kullanılan fonksiyon 3 parametre alır. 1. Parametre çalışacağı fonksiyonu belirtir, 2.parametre çalışacağı url’yi ve 3.parametre çalışacağı portu belirtir.

1.parametredeki fonksiyon websocketsden gelen websocket ve path parametrelerini alır. Önceden tanımlanan connected ifadesine gelen bağlantının websocket türünde verisi add() fonksiyonu ile eklenir. Burada connected ifadesi set() fonksiyonu ile küme olarak tanımlanmıştır. Çünkü gelen veriler websocket türündedir. Liste, sözlük ve demet veri türü gibi birden çok veri türünü birlikte barındıran veri tipidir. Kümeler ile ilgili yaptığınız her türlü işlevi(birleşme,kesişim vs.) bu veri tipi ile yapabilirsiniz.

Ancak indexleme bulunmamaktadır.

Aktif bulunan clientlerin isimlerini kisiler isminde sözlük türünde bir sözlükte sonra kullanılmak üzere tutulur. Bu sözlükte tutulan verilere erişmek için i isminde int değişkeni tanımlanır. Daha sonra bu değişken ile sözlükte indexleme yaparak istediğimiz veriye ulaşabiliriz.

try ifadesi ile server client arasında ki veri alışverişi bu ifade arasında gerçekleşir. İlk önce verinin mesaj mı yoksa kullanıcı adı mı olduğu kontrol edilir. Mesaj ise sunucuya bağlı diğer istemcilere yollanır, değilse aktif kişi listesi gönderilier

finally kısmında client kapatıldığında sunucuyla bağlantısı kesilir ve istemci kaldırılır.

```
<script>


  var vue = new Vue({
    el: "#app",
    data: {
      veriler: {
        message: "",
        kisiler: ["Client1"]
      }
    },
    methods: {
      mesajyaz: function(event) {
        var vm = this;
        if (vm.veriler.message != "") {
          //mesaj gönderilir
          document.getElementById("messagebox").innerHTML +=
           "<h5>   kullanıcı_1: " + vm.veriler.message + " </h5> ";
          socket.send(vm.veriler.message);
          vm.veriler.message = "";
        }
        },


    }
  });
```

Biz tasarım kısmında html & css kullandık DOM işlemini vue.js ile yaptık. Burda mesajyaz () fonksiyonu enter veya gönder butonuna basılığında tetiklenir ve mesajı socket.send() ile servere yollar.

```
const socket = new WebSocket('ws://localhost:5000');

socket.addEventListener('open',  function(event) {
   console.log('Connected to the WS Server!');
   socket.send("_Client1");
 });
 socket.addEventListener('close', function(event) {
   console.log('Disconnected from the WS Server!')
   var index=vue._data.veriler.kisiler.indexOf("_Client1");
   vue._data.veriler.kisiler.splice(index, 1);
   console.log(vue._data.veriler.kisiler);
});
  socket.addEventListener('message', function (event) {
   //mesaj geliyor

 if (event.data.substring(0,1)!="_") {
  document.getElementById("messagebox").innerHTML+="<h5>   kullanıcı_2: "+event.data+"</h5>";

   }
   else {
     console.log(event.data)
         for (var kisi  of vue._data.veriler.kisiler) {

               if (kisi=="Client1") {
                 vue._data.veriler.kisiler[0]=(event.data);

                  }  else{
                 if (kisi!=event.data) {
                   vue._data.veriler.kisiler.push(event.data);

               }
```

Burada websocket ile belirtilen url'ye socket bağlanır. Socketinde addEventListener fonksiyonları ile açma, kapama ve mesaj alma işlemleri yapılır.

Açma işleminde bulunduğu clientin kullanıcı adını sunucuya gönderir.

Mesajda ise sunucudan gelen cevabın kullanıcı adı mı mesaj mı olduğunun kontrolü yapılır.

Close da ise client kapatıldığı durumda sunucuya bilgi yollanır ve aktif kullanıcılardan ismi silinir.

Oğuz Eren ACAR

İsmail Furkan TÜRKOĞLU

Mehmet Nazım KÖROĞLU

Mehmet Can DEMİR