



**UNIVERZITET U ZENICI**  
**Politehnički fakultet**  
**Softversko inženjerstvo**  
**Multimedijalni sistemi i aplikacije**



**VIDEO / AUDIO / TEKST CHAT SOBA - PRIČALICA**

***Profesor: Doc. Dr. Nermin Goran***

***Student: Ahmed Mujić i Amer Musić***

**Zenica, 2023**

## **SADRŽAJ**

<b>DOKUMENTACIJA ZA KORISNIKA .....</b>	<b>3</b>
<b>PRIMJER KORIŠTENJA APLIKACIJE.....</b>	<b>4</b>
<b>DOKUMENTACIJA ZA DEVELOPERE .....</b>	<b>7</b>
<b>DOKUMENTACIJA KLIJENTA.....</b>	<b>10</b>
<b>WIRESHARK ANALIZA.....</b>	<b>14</b>

## **DOKUMENTACIJA ZA KORISNIKA**

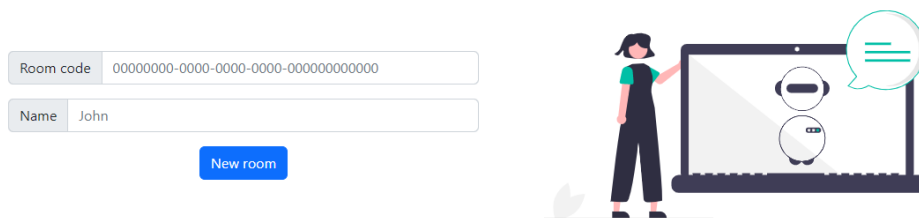
Pričalica je besplatna aplikacija koja se može koristiti putem računara mobitela i tableta, a omogućava komunikaciju između dvije ili više osoba. Putem ove aplikacije moguće je kreirati razne događaje kao što su sastanci, treninzi, webinar i slično. Osim prethodno navedenih mogućnosti koje nudi aplikacija omogućeno je i dijeljenje sadržaja kao što su dokumenti, fotografije i videozapisi. Aplikacija je besplatna, jednostavna za korištenje te olakšava komunikaciju na daljinu.

Budući da u posljednjih nekoliko godina tempo života postaje sve ubrzniji, velika većina se trudi uštedjeti na vremenu što više može jer “vrijeme je novac. Upravo je iz tog razloga ova aplikacija idealno rješenje za sve koji žele održati komunikaciju na daljinu bilo iz edukativnih, poslovnih ili privatnih razloga. Također, u toku pandemije korona virusa online aplikacije za komunikaciju su u mnogome olakšale komunikaciju.

## PRIMJER KORIŠTENJA APLIKACIJE

Prilikom pristupa linku <https://pricalicafrontend.azurewebsites.net/> korisniku se otvara sljedeći prozor kao na slici ispod:

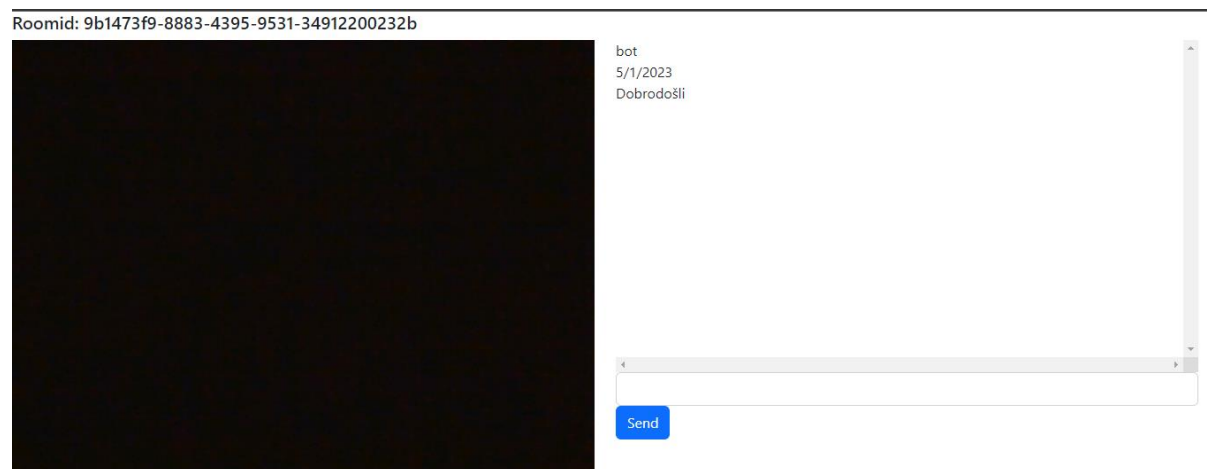
---



Slika 1: Izgled početnog ekrana

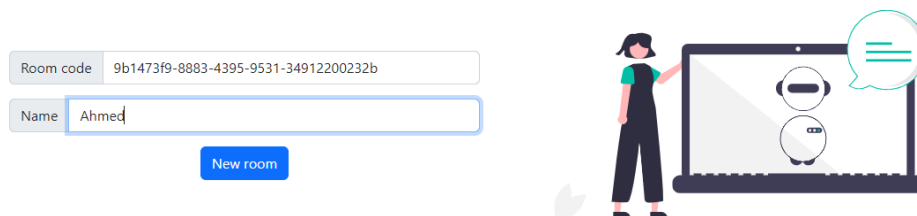
Ukoliko je korisnik koji je pristupi linku host sobe za poziv on će da popuni samo *Name* polje i potom kliknuti na dugme *New room*. Nakon što se priključi sobi za poziv otvorit će mu se sljedeći ekran:

---



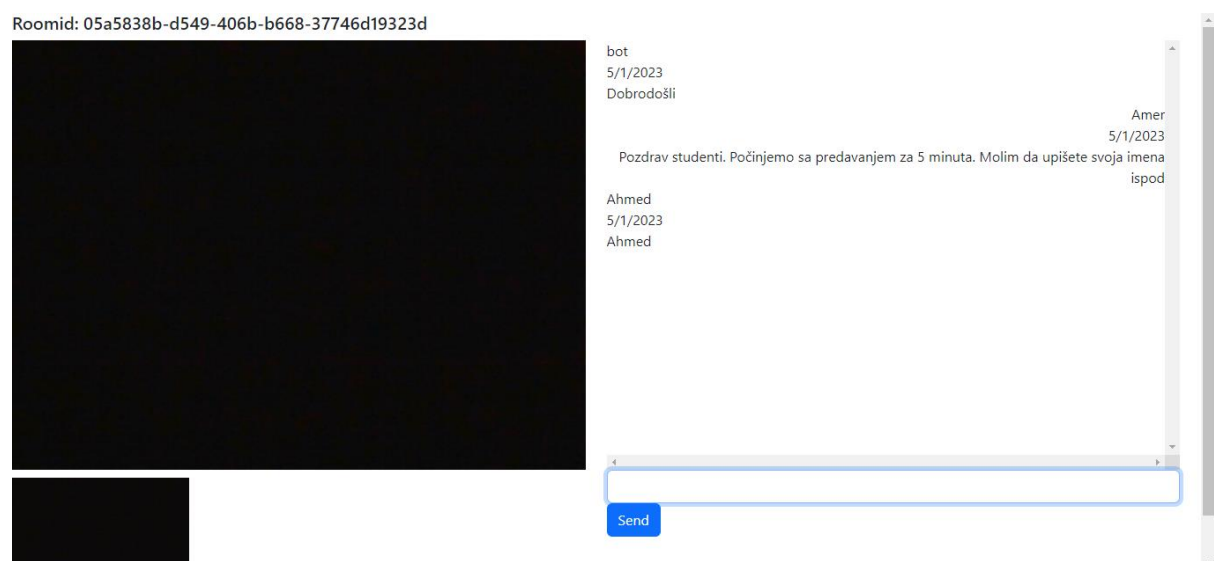
Slika 2: Izgled početnog ekrana sobe

U samom vrhu ekrana host sobe može da vidi RoomId koji je potrebno poslati drugim korisnicima kako bi pristupili istoj sobi. Nakon što pošalje kod drugi korisnik može da pristupi sobi upotrebom RoomId. Izgled prikaza ekrana početnog ekrana je također isti kao i kod ulaza host korisnika samo što u ovom slučaju korisnik pored imena unosi i RoomId.

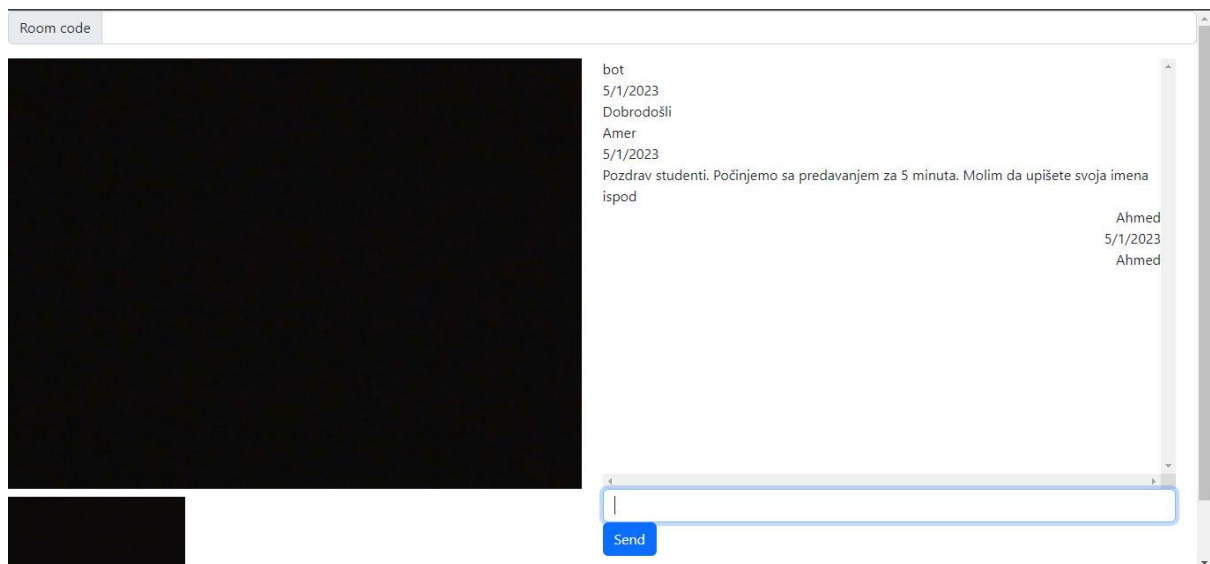


Slika 3: Izgled početnog ekrana korisnika koji pristupa sobi upotrebom linka

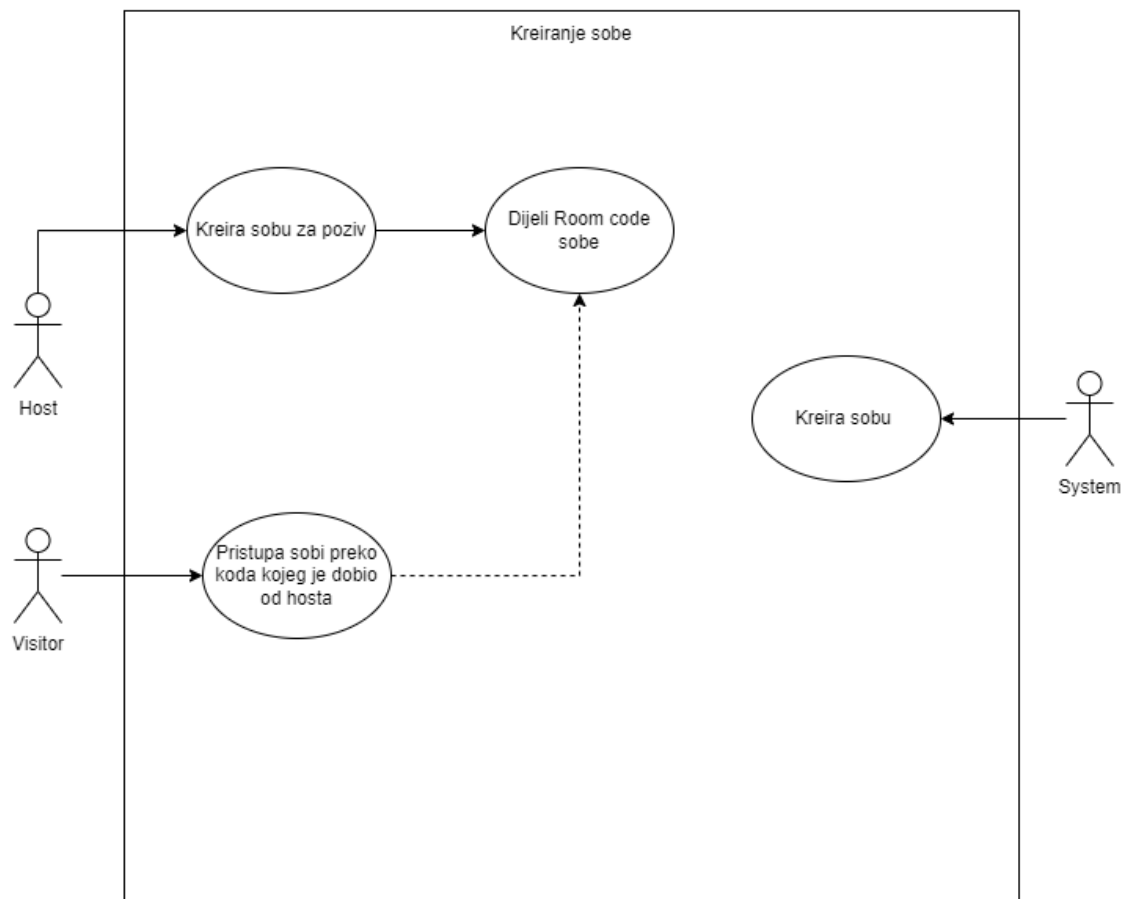
Nakon što je korisnik dodao potrebne podatke i pridružio se sobi ima pristup chatu kao i kameriehost korisnika koja zauzima većinski dio ekrana. Desno od kamere korisnika nalazi se chat polje u kojem je moguće razmijenjivati poruke između korisnika.



Slika 4: Prikaz ekrana host korisniku sa vidljivim porukama drugih korisnika



Slika 5: Prikaz ekrana korisniku sa prikazom poruka host korisnika



Ono što je objašnjeno u tekstu iznad možemo vidjeti na ovom diagram. Naime, host korisnik kreira sobu i dijeli link sa drugim korisnicima kako bi isti imali mogućnost da se pridružeistoj sobi.

## DOKUMENTACIJA ZA DEVELOPERE

Prilikom izrade projekta korištene su tehnologije:

- Angular 11
- .NET Core 6
- SignalR
- PeerJS

Kreiranje soba i chat se vrše preko SignalR-a, dok se video i audio poziv vrše preko PeerJS (peer to peer) konekcija.

Da bi se soba kreirala potrebno je da korisnik unese svoje ime i da istu napravi, nakon čega dobija id te sobe.

Prilikom spajanja šaljemo roomId, username i peerId. Ukoliko soba nije napravljena istu pravimo i dodajemo u SignalR grupu. Informacije o korisnicima sobe se spremaju u jednu statičku varijablu koja ostaje takva tokom cijelog vijeka aplikacije. Zatim se poziva metoda SendAsync iz SignalR biblioteke kojoj prosljeđujemo metodu "UserOnlineInGroup" i podatke o svim korisnicima iz te sobe. Ovo će biti korišteno kako bismo mogli prikazati koji su korisnici prisutni.

```
public override async Task OnConnectedAsync()
{
    var httpContext = Context.GetHttpContext();
    var roomId =
httpContext?.Request.Query["roomId"].ToString();
    var username =
httpContext?.Request.Query["username"].ToString();
    var peerId =
httpContext?.Request.Query["peerId"].ToString();
    if (string.IsNullOrEmpty(peerId))
    {
        peerId = Rooms["roomId"].Select(r =>
r.PeerId).FirstOrDefault();
    }
    var userInfo = new UserInfoDto { ConnectionId =
Context.ConnectionId, Username = username, PeerId = peerId };
}
```

```

        if(roomId == null)
        {
            throw new Exception("Room id is not sent");
        }

        if (!Rooms.TryGetValue(roomId, out var user))
        {
            Rooms.Add(roomId, new List<UserInfoDto>() {
userInfo });
        }
        else
        {
            user.Add(userInfo);
        }

        await Groups.AddToGroupAsync(Context.ConnectionId,
roomId);

        await
Clients.Group(roomId).SendAsync("UserOnlineInGroup",
Rooms[roomId].ToList());
    }

```

Za podatke o svim korisnicima možemo koristiti metodu `GetOtherUsersInRoom` koja je dio RTC Hub-a. Ona jednostavno pokupi sve korisnike iz određene sobe na osnovu `roomId` koji se dobije na osnovu `ConnectionId`-a.

```

public async Task GetOtherUsersInRoom()
{
    var roomId = Rooms.Where(r => r.Value.FirstOrDefault(u
=> u.ConnectionId == Context.ConnectionId) !=
null).FirstOrDefault().Key;
    await Clients.Caller.SendAsync("UsersInGroup",
Rooms[roomId].ToList());
}

```



Za slanje poruka koristimo metoda `SendMessage` i prosljedimo `username` i poruku. Na osnovu `ConnectionId`-a se dobije `roomId` i svim klijentima pošaljemo poruku na osnovu metode `"NewMessage"`.

```
public async Task SendMessage(CreateMessageDto messageDto)
{
    var roomId = Rooms.Where(r => r.Value.FirstOrDefault(u
=> u.ConnectionId == Context.ConnectionId) !=
null).FirstOrDefault().Key;

    await Clients.Group(roomId).SendAsync("NewMessage",
new SendMessageDto(messageDto));
}
```

## DOKUMENTACIJA KLIJENTA

```
this.route.queryParams.pipe(takeUntil(this.destroySubject)).subscribe((params) => {
    this.roomId = params['id'] ?? Guid.newGuid(); // ako nema id kreiramo novi
    this.username = params['username']; // dohvatimo username
    this.metadata = {
        displayName: params['username'], // displayName je onako kako će se prikazati drugom korisniku
        userName: params['username'], //username
    };

    this.isHost = params['id'] == null; //ako u ruti nije proslijeđen id onda je taj klijent host
});
```

```
async createLocalStream() {
    this.stream = await navigator.mediaDevices.getUserMedia({
        video: this.enableVideo,
        audio: this.enableAudio,
    }); //dohvatimo audio i video elemente korisnika
    this.LocalVideoPlayer.nativeElement.srcObject = this.stream; //postavimo u DOM naš stream
    this.LocalVideoPlayer.nativeElement.load(); //učitamo video
    this.LocalVideoPlayer.nativeElement.play(); //pokrenemo video, odnosno našu sliku
}
```

```
callUser(callingPeer: string, user: User) {
    let call = this.myPeer.call(callingPeer, this.stream, {
        metadata: this.metadata,
    }); //pozivamo korisnika pri čemu proslijeđujemo peer, stream (naše video i audio podatke i metapodatke)

    call.on('stream', (remoteStream) => { //na uspostavu poziva
        const alreadyExisting = this.videos.findIndex(
            (video) => video.member.userName === user.username
        ); //ako korisnik već postoji i nije potrebno dodati korisnika
        if (alreadyExisting != -1) {
            return;
        }
    });
}
```

```

    }
    this.videos = [ // ažuriranje videos niza koji će rerenderovat DOM
      ...this.videos,
      {
        muted: false,
        srcObject: remoteStream,
        member: {
          userName: user.username,
          displayName: user.username,
        },
      },
    ];
  });
}

```

```

this.myPeer = new Peer(); //inicijalizacija peer-a

this.myPeer.on('open', (id: string) => {
  this.peerId = id; //snimanje peerId-a

  this.hubService.createHubConnection( //kreiranje SignalR konekcije
    { username: this.username ?? '',
      roomId: this.roomId ?? '',
      peerId: this.peerId,
      isHost: this.isHost
    }
  );
});

this.myPeer.on('call', (call) => { // na javljanje
  call.answer(this.stream);

  //this.callUser(call.peer);
  this.connectedPeers.push(call.peer);
  call.on('stream', (otherUser: MediaStream) => {
    const alreadyExisting = this.videos.findIndex(
      (video) => video.member.userName === call.metadata.userName
    );
    if (alreadyExisting !== -1) {
      return;
    }
    this.videos = [
      ...this.videos,

```

```

        {
            muted: false,
            srcObject: otherUser,
            member: {
                userName: call.metadata.userName,
                displayName: call.metadata.userName,
            },
        },
    ];
});
});

this.hubService.usersConnected.pipe(takeUntil(this.destroySubject)).subscribe(
(users: User[])=> {
    if(!this.isHost){//ako nismo host
        users.forEach(user => {
            if(this.usersInCall.findIndex(uc => uc.peerId == user.peerId) == -
1 && this.peerId != user.peerId){
                this.callUser(user.peerId as string, user); //pozivamo drugog
koristnika
                this.usersInCall.push(user); //updateovajne liste korisnika u
pozivu
            }
        })
    }
})
})

```

```

createHubConnection(user: User, roomId: string, peerId: string, isHost:
boolean) {
    this.hubConnection = new HubConnectionBuilder()
        .withUrl(
            environment.hubUrl + '?roomId=' + roomId + '&username=' + user.username
+ '&peerId=' + peerId
        )
        .withAutomaticReconnect()
        .build();//kreiranje konekcije

    this.hubConnection.start().then(_=>{

```

```

    this.hubConnection.on('UsersInGroup', (users: User[]) => { // ako SignalR
pošalje ovu metodu mi ćemo update-ovati listu korisnika koji su konektovani
        this.usersConnected.next(users);
    });

    this.hubConnection.on('NewMessage', (message: IncommingMessage) => {
//Ukoliko primimo poruku postavimo subject na zadnju primljenu poruku
        this.incommingMessageSubject.next(message);
    });

    if(!isHost){
        this.hubConnection.invoke('GetOtherUsersInRoom') //ako smo se tek
spojili dohvatamo sve korisnike u sobi
    }
    }).catch((err) => console.log(err));
}

sendMessage(message: Message){
    return this.hubConnection.invoke('SendMessage', message) // slanje poruke
}

```

## WIRESHARK ANALIZA

Da bismo testirali slanje podataka i primanje koristili smo wireshark program za praćenje paketa. Ip adrese koje su uključene u razmjenu podataka su:

- 192.168.0.18 (klijent koji je napravio sobu, odnosno lokalni klijent)
- 85.92.228.99 (klijent koji se pridružio sobi)

No.	Time	Source	Destination	Protocol	Length	Info
96203	191.645035	192.168.0.18	66.22.243.154	UDP	130	56112 → 50005 Len=88
96204	191.645041	192.168.0.18	66.22.243.154	UDP	314	56112 → 50005 Len=272
96205	191.650190	192.168.0.18	85.92.228.99	UDP	160	53762 → 57957 Len=118
96206	191.650927	192.168.0.18	35.207.188.57	UDP	253	63462 → 50008 Len=211
96207	191.651169	85.92.228.99	192.168.0.18	UDP	80	57957 → 53762 Len=38
96208	191.652399	85.92.228.99	192.168.0.18	UDP	233	57957 → 53762 Len=191
96209	191.655292	74.125.250.136	192.168.0.18	UDP	176	3478 → 52384 Len=134
96210	191.655922	192.168.0.18	66.22.243.154	UDP	130	56112 → 50005 Len=88
96211	191.655933	192.168.0.18	66.22.243.154	UDP	130	56112 → 50005 Len=88
96212	191.655942	192.168.0.18	66.22.243.154	UDP	130	56112 → 50005 Len=88
96213	191.655948	192.168.0.18	66.22.243.154	UDP	130	56112 → 50005 Len=88
96214	191.655955	192.168.0.18	66.22.243.154	UDP	130	56112 → 50005 Len=88

ip.src == 85.92.228.99    ip.src == 192.168.0.18						
No.	Time	Source	Destination	Protocol	Length	Info
2921...	482.943550	192.168.0.18	66.22.243.154	UDP	1099	63323 → 50005 Len=1057
2921...	482.943556	192.168.0.18	66.22.243.154	UDP	1099	63323 → 50005 Len=1057
2921...	482.943563	192.168.0.18	66.22.243.154	UDP	1099	63323 → 50005 Len=1057
2921...	482.943569	192.168.0.18	66.22.243.154	UDP	1099	63323 → 50005 Len=1057
2921...	482.944503	74.125.250.136	192.168.0.18	UDP	1230	3478 → 52384 Len=1188
2921...	482.946059	192.168.0.18	85.92.228.99	UDP	153	53762 → 57957 Len=111
2921...	482.948503	192.168.0.18	35.207.188.57	UDP	234	61936 → 50008 Len=192
2921...	482.948515	192.168.0.18	66.22.243.154	UDP	1099	63323 → 50005 Len=1057
2921...	482.949032	74.125.250.136	192.168.0.18	UDP	1230	3478 → 52384 Len=1188
2921...	482.949436	74.125.250.136	192.168.0.18	UDP	174	3478 → 52384 Len=132
2921...	482.953261	85.92.228.99	192.168.0.18	UDP	233	57957 → 53762 Len=191
2921...	482.958914	74.125.250.136	192.168.0.18	UDP	164	3478 → 52384 Len=122
2922...	482.962437	192.168.0.18	85.92.228.99	UDP	92	53762 → 57957 Len=50
2922...	482.965195	192.168.0.18	35.207.188.57	UDP	250	61936 → 50008 Len=208
2922...	482.965989	85.92.228.99	192.168.0.18	UDP	80	57957 → 53762 Len=38
2922...	482.966008	192.168.0.18	85.92.228.99	UDP	153	53762 → 57957 Len=111
2922...	482.975016	85.92.228.99	192.168.0.18	UDP	233	57957 → 53762 Len=191
2922...	482.975298	192.168.0.18	74.125.250.136	UDP	94	52384 → 3478 Len=52
2922...	482.976228	192.168.0.18	66.22.243.154	UDP	92	63323 → 50005 Len=50
2922...	482.976250	192.168.0.18	66.22.243.154	UDP	1109	63323 → 50005 Len=1067
2922...	482.976256	192.168.0.18	66.22.243.154	UDP	1109	63323 → 50005 Len=1067
2922...	482.976263	192.168.0.18	66.22.243.154	UDP	1109	63323 → 50005 Len=1067
2922...	482.976273	192.168.0.18	66.22.243.154	UDP	1110	63323 → 50005 Len=1068
2922...	482.976279	192.168.0.18	66.22.243.154	UDP	1110	63323 → 50005 Len=1068
2922...	482.978878	192.168.0.18	162.159.137.234	TLSv1.2	139	Application Data