

## Dokumentáció

Türk Viktor

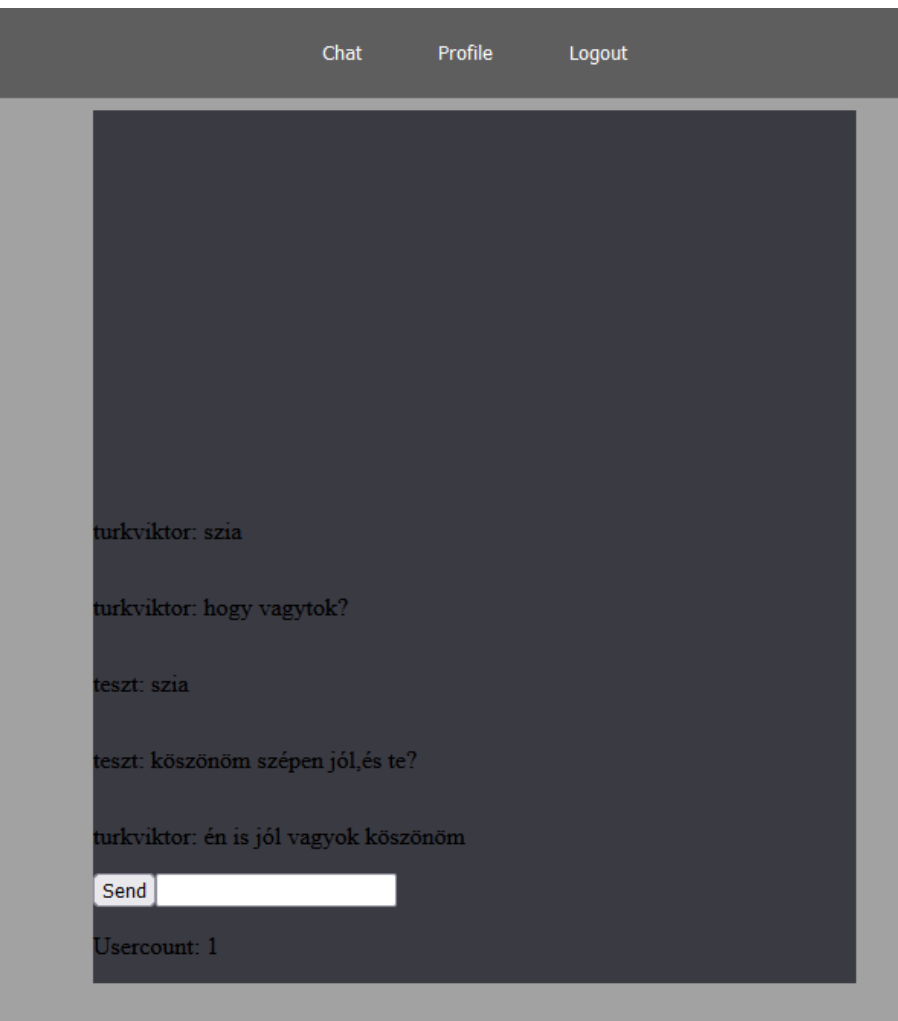
F5HV4G

Chat/Social app

# A weboldal célja

A weboldal célja egy privátabb chatet/szociális teret létrehozni

A weboldalon főként megtalálható a chat



Ez a chat a socket.io websocket library felhasználásával készült mert a cél az volt,hogy másodpercre pontosan tudjanak egymással kommunikálni az emberek

# A chat működése

A chat az előbb említett websocket segítségével működik:

Amint valaki belép a chat felületére létrehoz egy websocket kapcsolatot amit a backend egyből érzékel, megnöveli a felhasználók számát majd üzenetet küld minden felhasználó részére, hogy frissüljön a felhasználók száma mindenkinél

```
io.on('connection', (socket) => {  
  console.log(socket.id + ' connected');  
  userCount++;  
  io.emit('user-count', userCount);  
});
```

Ezek után egy üzenetek tömbön végig iterál és elküldi az adott felhasználónak azt összes korábbi üzenetet

```
for(let i = 0; i < messages.length; i++){  
  socket.emit('new-message', messages[i]);  
}
```

Ezek után ha üzenetet szeretnénk küldeni egy input box segítségével megtehetjük:

```
<input  
  [(ngModel)]="message.message"  
  (keydown.enter)="sendMessage()"  
  type="text"  
>
```

Ez meghívja a sendMessage()-t

```
sendMessage() {  
  this.chatService.sendMessage(this.message);  
  this.message.message = '';  
}
```

Ami küld a backendre websocketen keresztül egy üzenetet

```
sendMessage(message: { user: string | undefined, message: string }){  
  this.socket.emit('new-message', message);  
}
```

Az üzenetben a felhasználó neve illetve az üzenete kerül elküldésre amit a "socket.emit" segítségével meg is kap a backend.

A backend ezután feldolgozza az üzenetet, megnézi, hogy ne legyen üres majd az üzenetek tömbbe belerakja az érkezett üzenetet, és továbbküldi a többi felhasználónak is, hogy lehessen látni.

```
socket.on('new-message', (message: { user: string, message: string }) => {  
  if(message.message.trim() == "") return;  
  console.log(message);  
  messages.push(message);  
  io.emit('new-message', { "user": message.user, "message": message.message });  
});
```

Ha egy felhasználó kilépne/megszakadna a kapcsolat a backend rögtön érzékeli és csökkenti a felhasználók számát.

```
socket.on('disconnect', () => {  
  console.log(socket.id + ' disconnected');  
  userCount--;  
  io.emit('user-count', userCount);  
});
```

# Profil oldal

Az oldalon megtalálható egy saját profil oldal

Itt megtudjuk adni a leírásunkat és azokat a szociális felületet ahol fent vagyunk.

Az edit profile gombra kattintva már szerkeszthető is a profilunk.

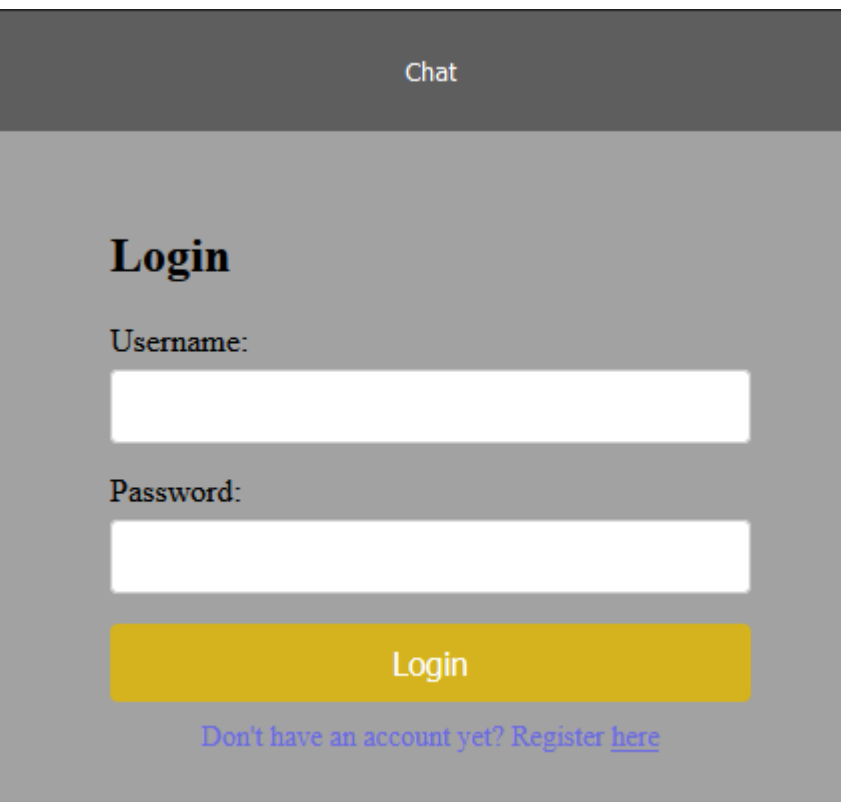


A backendre küldött kérés után egyből frissül a MongoDB-s adatbázisunk.

```
"description": "teszt",  
"twitterhandle": "kukac5",  
"linkedinhandle": "lockedin",  
"githubhandle": "hitgub"  
},  
{  
  "_id": "667a409e337dc0ae410f72f2",  
  "username": "turkviktor",  
  "password": "asdasd",  
  "description": "új leírás",  
  "twitterhandle": "masiktwitter",  
  "linkedinhandle": "másiklindedin",  
  "githubhandle": "másikgithub"  
}
```

# Bejelentkezés oldal

Regisztrálás után be is tudunk jelentkezni

A screenshot of a web application's login page. At the top, there is a dark gray header with the word "Chat" in white. Below the header, the main content area has a light gray background. The word "Login" is displayed in a large, bold, black font. Underneath, there are two white input fields with gray borders. The first field is labeled "Username:" and the second is labeled "Password:". Below these fields is a yellow button with the word "Login" in black text. At the bottom of the form, there is a link that says "Don't have an account yet? Register [here](#)".

Bejelentkezéskor a backend egy titkos szó segítségével general egy JWT-t amit visszaküld frontendre és eltárolja a Local Storagebe

```
async Login(user: DTO.Login): Promise<string | null> {
  const userCollection = await this.getUserCollection();
  const userFound = await userCollection.findOne({ username: user.username, password: user.password });
  if (!userFound) {
    return null;
  }

  const token = jwt.sign({ username: user.username,
    exp: Math.floor(Date.now() / 1000) + (3600 * 3600),
    user_id: userFound._id
  }, this.jwtSecret);

  return token;
}
```

Ez a JWT fontos szerepet játszik a validálásban, mivel ennek a segítségével tudjuk megerősíteni a felhasználónkat, hogy tényleg ő az

Erre egy validateToken() Függvényt használ a backend

```
function validateToken(req: Request, res: Response, next: NextFunction) {  
  const token = req.get("Authorization")  
  if (!token) {  
    res.status(401).json({ error: 'Unauthorized' });  
    return;  
  }  
  const toVerify = token.split(' ')[1];  
  const tokenVerified = jwt.verify(toVerify, mongo.jwtSecret);  
  next();  
}
```

Ez a függvényt a backend sokszor felhasználva mint itt látható:

```
app.get("/user", validateToken, async (req: Request, res: Response) => {
```

Frontenden minden releváns oldalon(mint pl. a profil) elküldi ezt a kérést és csak akkor enged tovább ha megbizonyosodott a backend a token érvényességéről

## Regisztrációs oldal

Végezetül találkozhatunk a regisztrációs oldallal:

Itt amennyi fiókot létre szeretnék hozni annyit létre lehet.

Hasonlóan a bejelentkezéshez ez is egy kérést küld a szerverre és a kérés ellenőrzése után fel is viszi az adatbázisba

```
app.post("/register", DTO.validateRegister, (req: Request, res: Response) => {  
  const registerData: DTO.Register = req.body;  
  registerData.description = "No description provided";  
  registerData.twitterhandle = "No twitter handle provided";  
  registerData.linkedinhandle = "No linkedin handle provided";  
  registerData.githubhandle = "No github handle provided";  
  
  mongo.Register(registerData)  
    .then((result) => {  
      res.json(result);  
    })  
    .catch((err) => {  
      res.status(500).json({ error: err });  
    });  
});
```

```
async Register(user: DTO.Register): Promise<InsertOneResult> {  
  const userCollection = await this.getUserCollection();  
  const result = await userCollection.insertOne(user);  
  return result;  
}
```

Chat

### Register

Username:

Email:

Password:

Password again:

Already have an account? [Login here](#)