Üzleti intelligencia labor jegyzőkönyv

A jegyzőkönyvet készítette:

Csáky Richárd, BNLUAG

A gyakorlat ideje, helye:

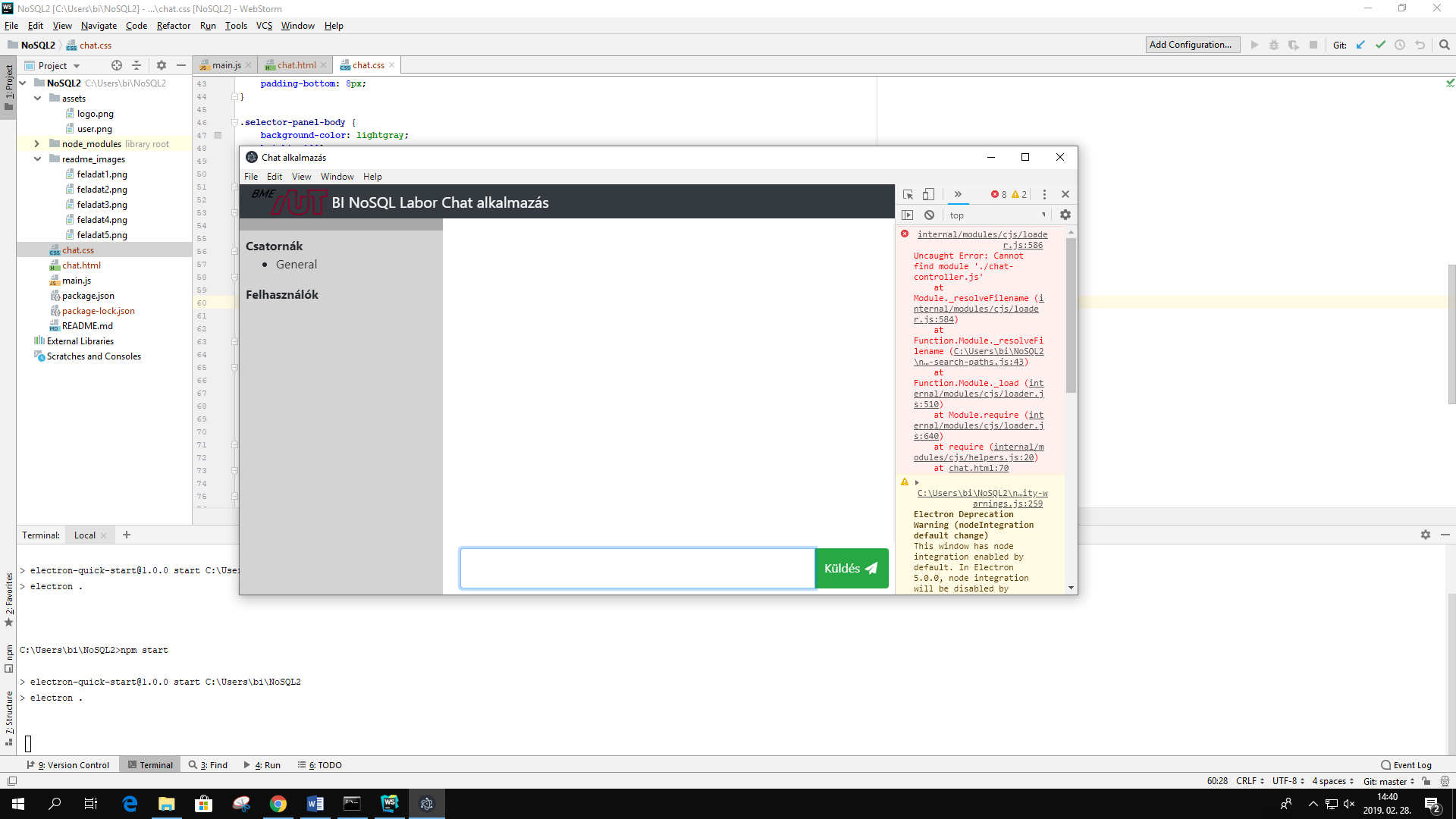
2019.02.28, IL207

Gyakorlatvezető:

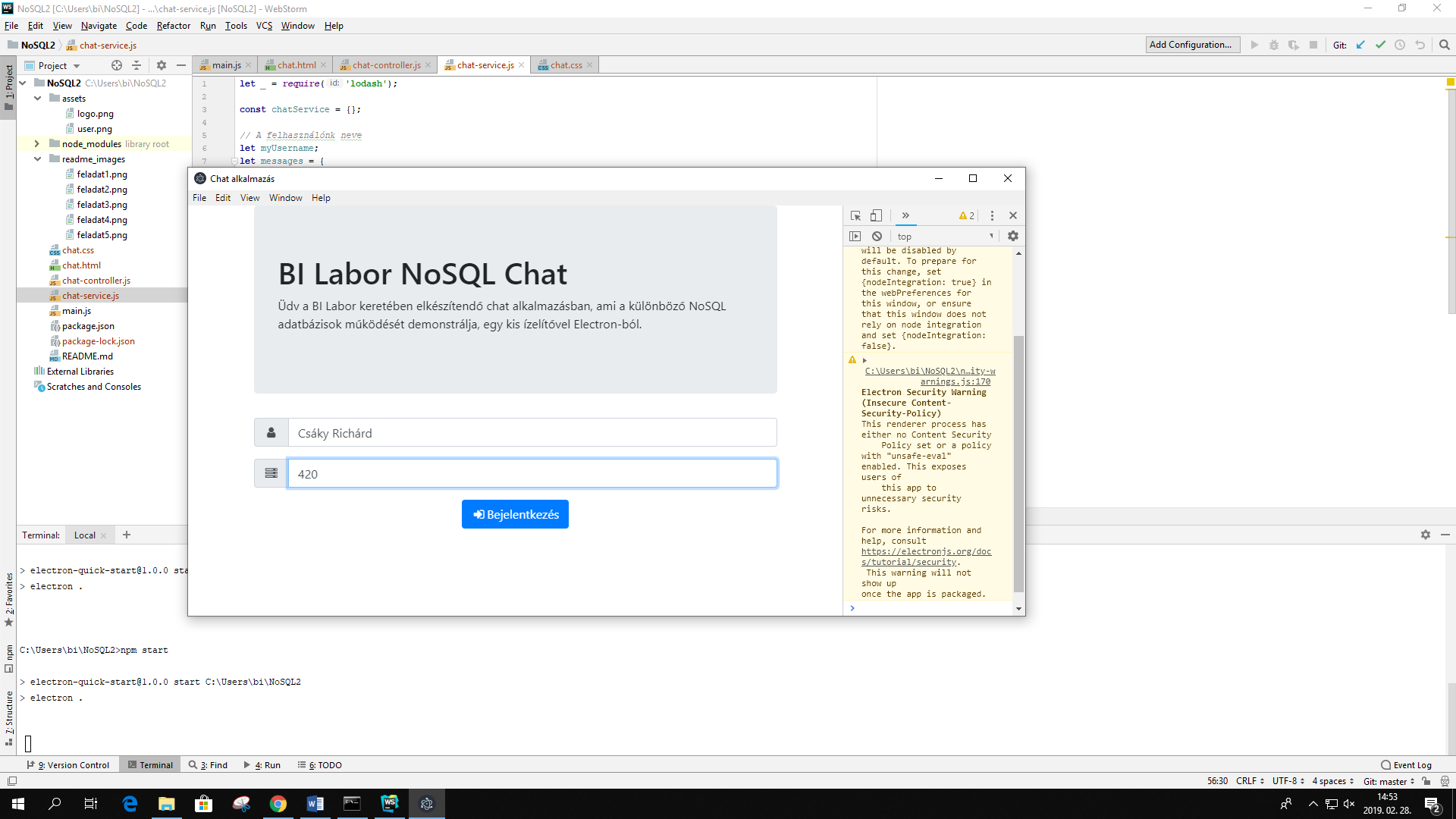
Sík Dávid

# Vezetett rész

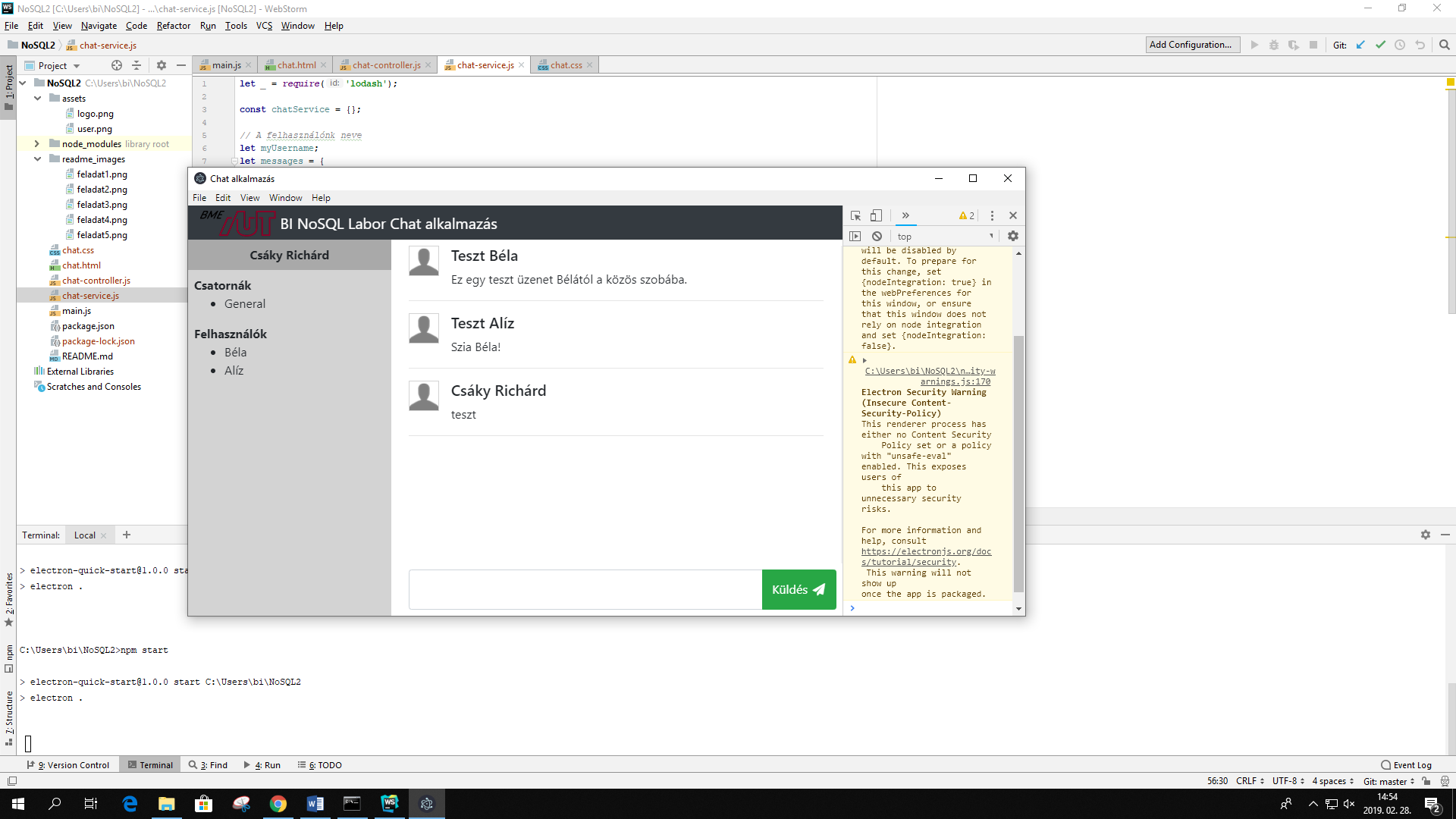
## Első feladat



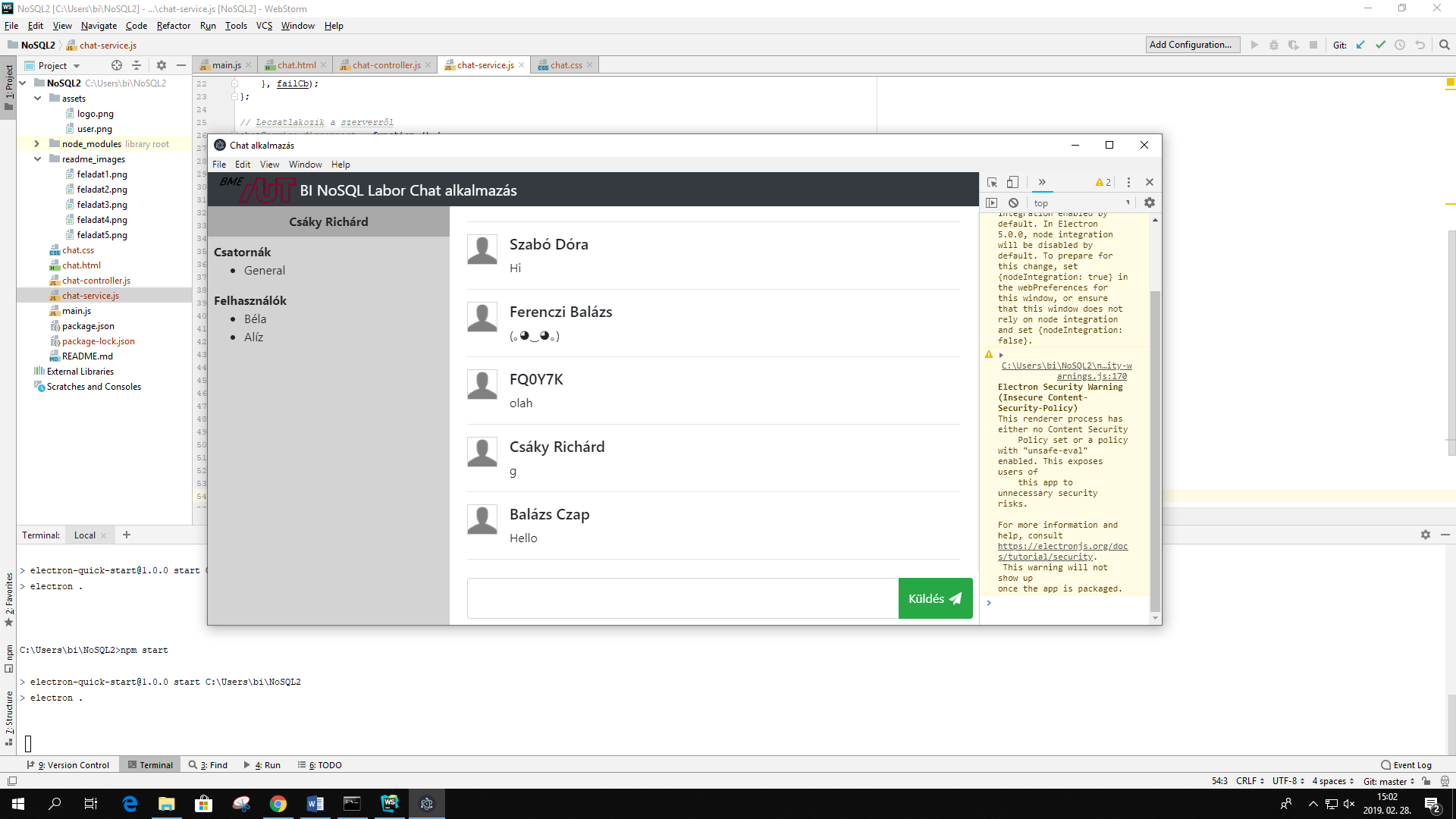
## Bejelentkezés



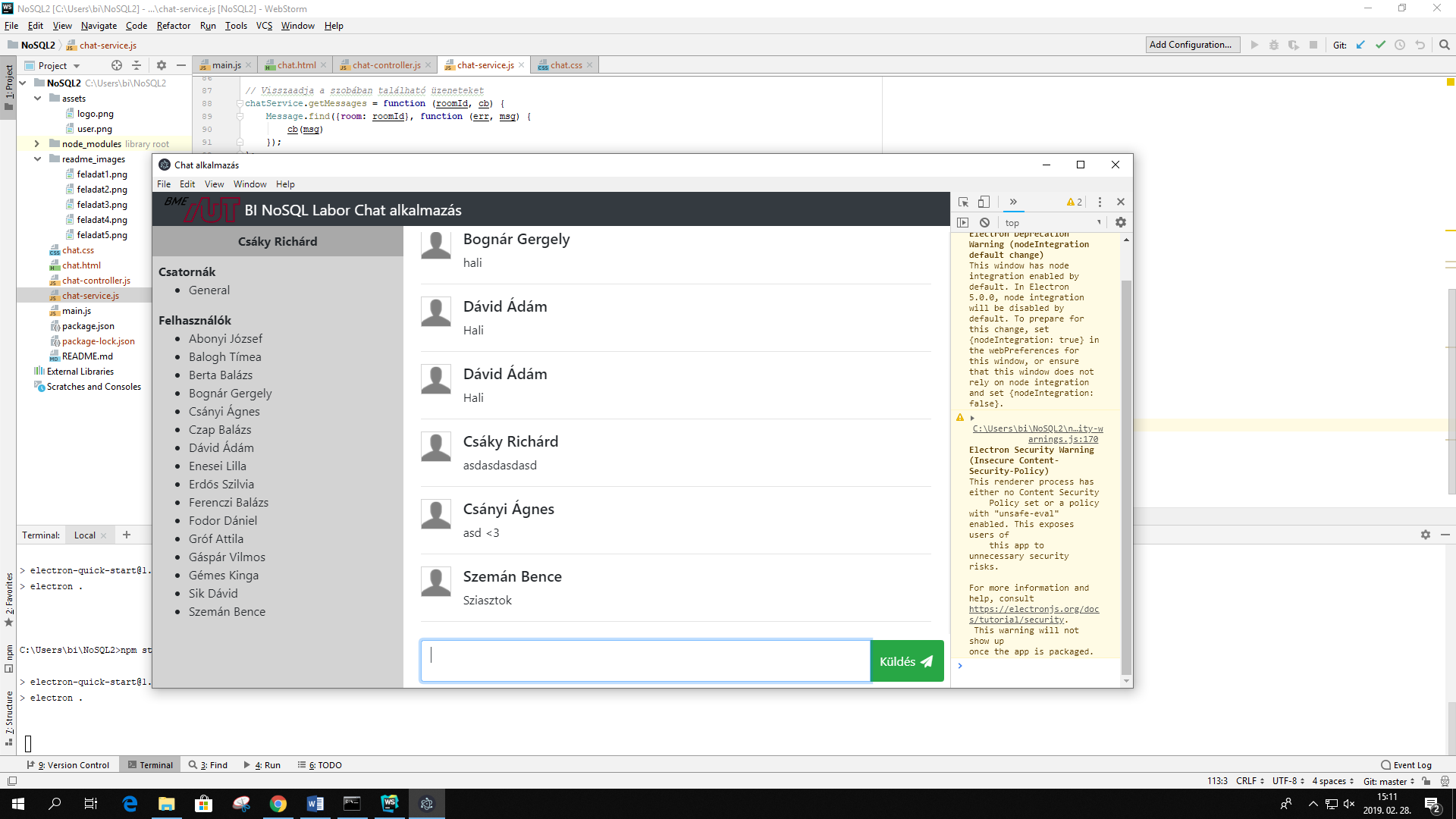
## Teszt üzenet küldése



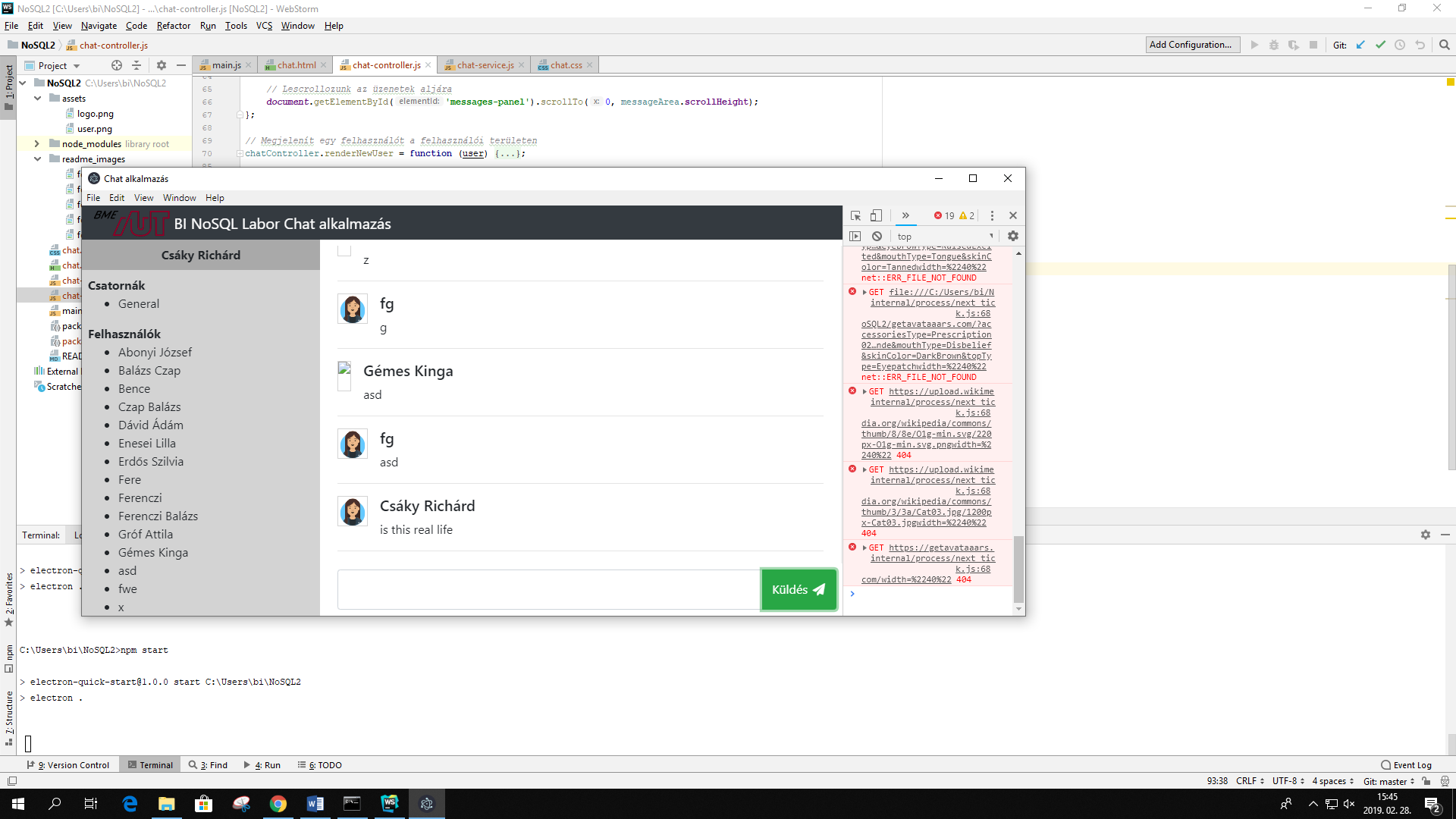
## Szerverre küldés



## Felhasználók lista



## Első egyéni Feladat



### chat.html-ben:

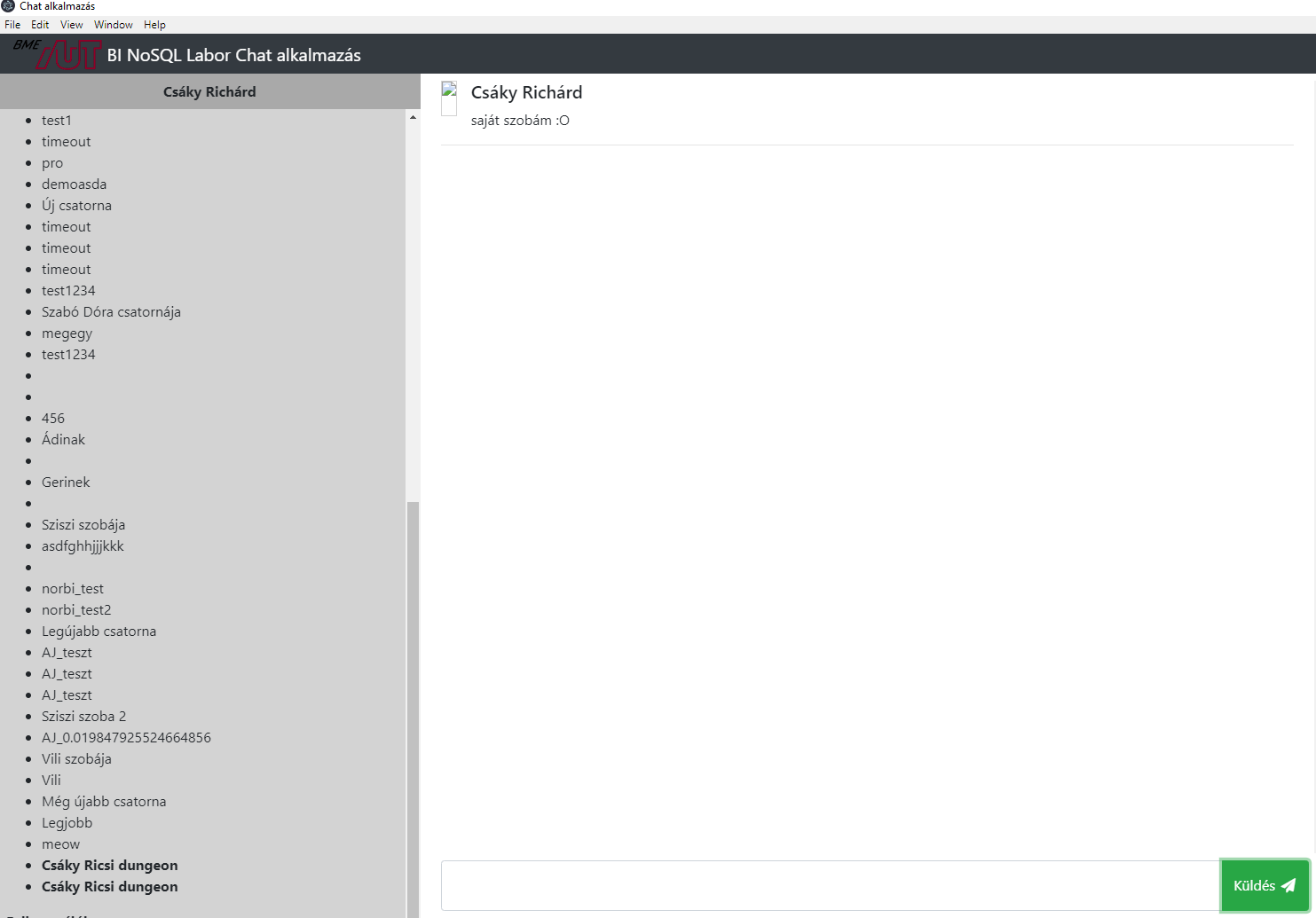
<**div class="input-group mb-3"**>  
 <**div class="input-group-prepend"**>  
 <**span class="input-group-text"**><**i class="fa fa-fw fa-server"**></**i**></**span**>  
 </**div**>  
 <**input id="avatarURL" type="text" class="form-control" placeholder="Avatar URL"**>  
</**div**>

### chat-controller.js-ben:

*// Új üzenetet küldünk a felhasználónkkal*chatController.sendMessage = **function** () {  
 **let** textInput = ***document***.getElementById(**'new-message-text'**);  
 **if** (!*\_*.*isEmpty*(textInput.**value**)) {  
 **let** message = {  
 **avatarUrl**: myAvatar,  
 **user**: myUsername,  
 **content**: textInput.**value**,  
 **date**: **new *Date***()  
 };  
 chatController.renderNewMessage(message);  
 chatService.sendMessage(selectedRoom, message);  
 }  
 textInput.**value** = **''**;  
};

chatController.renderNewMessage = **function** (message) {  
 *// Megkeressük a DOM-ban a "messages" ID-val rendelkező üzenő területet, ami egy rendezetlen lista (<ul>).* **let** messageArea = ***document***.getElementById(**'messages'**);  
  
 *// Kitöltünk és hozzáadunk egy új üzenetet a HTML sablon alapján* messageArea.insertAdjacentHTML(**'beforeEnd'**,  
 **'<div class="media messages">'** +  
 **'<img src='** + *\_*.*escape*(message.**avatarUrl**) + **'width="40" height="40" class="mr-3 message-avatar">'** +  
 **'<div class="media-body">'** +  
 **'<h5 class="mt-0">'** + *\_*.*escape*(message.**user**) + **'</h5>'** + *\_*.*escape*(message.**content**) +  
 **'</div>'** +  
 **'</div>'** +  
 **'<hr>'** );  
  
 *// Lescrollozunk az üzenetek aljára* ***document***.getElementById(**'messages-panel'**).scrollTo(0, messageArea.**scrollHeight**);  
};

### Második és harmadik egyéni feladata



chat-controller.js-ben:

chatController.login = **function** () {  
 **let** usernameInput = document.getElementById('usernameInput');  
 **let** serverInput = document.getElementById('serverInput');  
 **let** avatarInput = document.getElementById('avatarURL');  
  
 **if** (\_.isEmpty(usernameInput.value) || \_.isEmpty(serverInput.value)) {  
 alert('Kérlek add meg az összes adatot!');  
 } **else** {  
 myUsername = \_.escape(usernameInput.value);  
 myAvatar = \_.escape(avatarInput.value);  
 chatService.connect(usernameInput.value, serverInput.value, **function** () {  
 //Sikeres csatlakozás esetén  
 // Screen-t váltunk (szegényember SPA-ja)  
 document.getElementById('login-window').style.display = 'none';  
 document.getElementById('main-window').style.display = 'flex';  
  
 // Kiírjuk a bejelentkezett felhasználó nevét  
 document.getElementById('username').innerText = myUsername;  
 chatController.refreshUsers();  
 chatController.refreshRooms();  
 },  
 **function** (err) {  
 alert("Nem sikerült csatlakozni az adatbázishoz: " + err)  
 },  
 // Új üzenet érkezett valahova (esemény a room\_channel-ben)  
 **function** (roomName) {  
 chatController.refreshRoom(roomName);  
 },  
 // Változott a felhasználók száma  
 **function** () {  
 chatController.refreshUsers();  
 },  
 **function** () {  
 chatController.refreshRooms();  
 });  
 }  
};

chatController.renderNewRoom = **function** (room) {  
 **let** roomList = document.getElementById('channel-list');  
 **let** listedRoom = \_.escape(room);  
  
 **if** (selectedRoom === listedRoom) {  
 // Ha már itt vagyunk nem kell linket készíteni.  
 roomList.insertAdjacentHTML('beforeEnd', '<li class="selector-panel-item selected"><b>' + listedRoom + '</b></li>');  
 } **else** {  
 roomList.insertAdjacentHTML('beforeEnd', '<li class="selector-panel-item" onclick="chatController.changeRoom(\'' + listedRoom + '\')">' + listedRoom + '</li>');  
 }  
};

chatController.changeRoom = **function** (roomName) {  
 //console.log(roomName);  
 selectedRoom = roomName;  
 chatController.refreshRooms();  
 chatController.refreshRoom();  
 chatController.refreshUsers();  
};

chatController.refreshRooms = **function** () {  
 document.getElementById('channel-list').innerHTML = '';  
 chatService.getRooms( **function**(rooms) {  
 \_.forEach(rooms, **function** (room) {  
 chatController.renderNewRoom(room.name);  
 })  
 });  
};

chat-service.js-ben:

**const** Room = mongoose.model('channel', **new** mongoose.Schema({  
 name: String  
}));

chatService.connect = **function** (username, serverAddress, successCb, failCb, messageCallback, userCallback, channelCallback) {  
 myUsername = username;  
 **let** dbReady = **false**;  
 **let** mqReady = **false**;  
  
 **let** db = mongoose.connect('mongodb://bi-chat:bi-chat@' + serverAddress + ':27017/bi-chat?authSource=admin', {useNewUrlParser: **true**});  
 redisClient = redis.createClient({  
 host: serverAddress, retry\_strategy: **function** () {  
 }  
 });  
  
 // Ha minden kapcsolat felépült  
 **function** connectionSuccesfull() {  
 // Felvesszük magunkat az online user listára  
 redisClient.zadd(usersChannel, 0, username);  
  
 // Szólunk a channelen hogy bejelentkeztünk  
 redisClient.publish(usersChannel, username);  
  
  
 // Feliratkozunk az eseményekre amiket figyelnünk kell  
 // A subscribehoz külön kliens kell, ezért lemásoljuk az eredetit  
 redisSubscriberClient = redisClient.duplicate();  
 redisSubscriberClient.subscribe(roomsChannel);  
 redisSubscriberClient.subscribe(usersChannel);  
 redisSubscriberClient.subscribe('channels\_channel');  
 redisSubscriberClient.on('message', **function** (channel, message) {  
 **if** (channel === roomsChannel) {  
 // Ha a szoba channel-be érkezik üzenet azt jelenti valamelyik szobába frissíteni kell az üzeneteket  
 console.log('message');  
 messageCallback(message);  
 } **else if** (channel === usersChannel) {  
 // Ha a user channelbe érkezik üzenet azt jelenti változott a user lista  
 console.log('user');  
 userCallback();  
  
 } **else if** (channel === 'channels\_channel') {  
 console.log("THIS");  
 channelCallback();  
 }  
 });  
  
 successCb();  
 }  
  
 // Nem tudjuk a kettő CB közül melyik hívódik meg előszőr, így a második után fogunk csak visszahívni  
 db.then(**function** () {  
 dbReady = **true**;  
 **if** (mqReady === **true**) {  
 connectionSuccesfull();  
 }  
 }, failCb);  
  
 // Redis kliens eseményei  
 redisClient.on('ready', **function** () {  
 mqReady = **true**;  
 **if** (dbReady === **true**) {  
 // Ha a DB kapcsolatot is felépítettük bejelentkezünk  
 connectionSuccesfull();  
 }  
 });  
 redisClient.on('error', failCb);  
  
 chatService.sendRoom();  
};

chatService.getRooms = **function** (cb) {  
 Room.find({}, **function** (err, msg) {  
 cb(msg)  
 });  
};

chatService.sendRoom = **function** () {  
 **let** room = **new** Room({  
 name: 'Csáky Ricsi dungeon'  
 });  
 room.save().then(**function** () {  
 // Szólunk hogy frissítettük a szobában az üzeneteket  
 redisClient.publish('channels\_channel', 'Csáky Ricsi dungeon');  
 })  
};