



# ÜZLETI INTELLIGENCIA LABOR JEGYZŐKÖNYV

A jegyzőkönyvet készítette:

László Tamás, VXR00S

A gyakorlat ideje, helye:

2020.02.19., QBF115

Gyakorlatvezető:

Sik Dávid

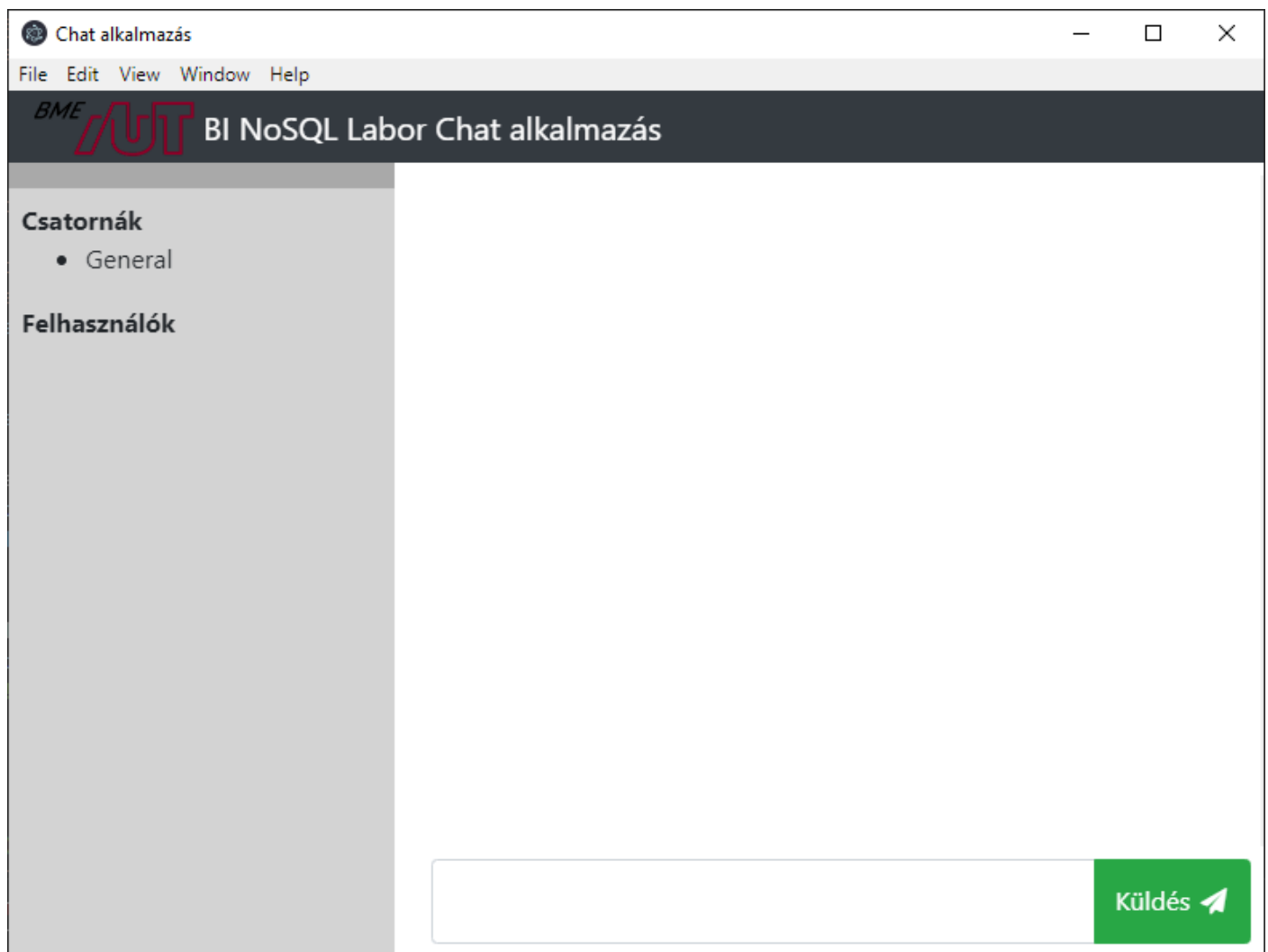


## 1) MEGJELÉNÍTÉS

### FELADAT LEÍRÁSA

Az első lépés az alkalmazás megjelenítésének elkészítése, ez a komplikációk elkerülése végett egyszerű HTML-ben történik, vanilla JavaScript-tel (azaz nem használunk külön keretrendszert). Kezdjük magának a chat felületnek az elkészítésével.

### MEGOLDÁS, MAGYARÁZAT, ÉRTÉKELES



## 2) ÜZLETI LOGIKA MEGVALÓSÍTÁSA

### FELADAT LEÍRÁSA

Építsük tovább az alkalmazásunkat a felületi és üzleti logika megvalósításával!


## MEGOLDÁS, MAGYARÁZAT, ÉRTÉKELES


Chat alkalmazás


File Edit View Window Help

# BI Labor NoSQL Chat

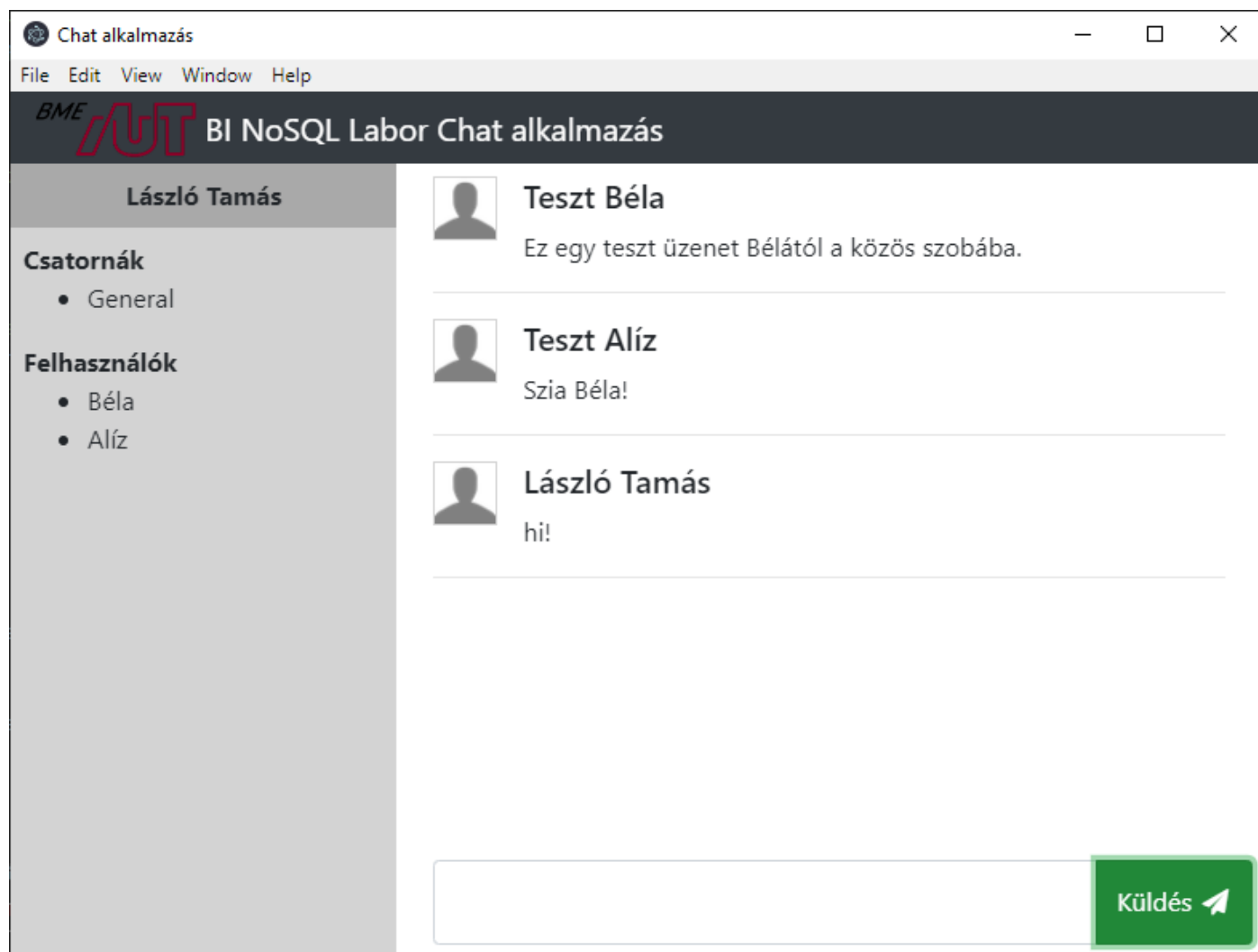
Üdv a BI Labor keretében elkészítendő chat alkalmazásban, ami a különböző NoSQL adatbázisok működését demonstrálja, egy kis ízelítővel Electron-ból.

Felhasználónév

Szerver IP

Szerver Jelszó

Bejelentkezés

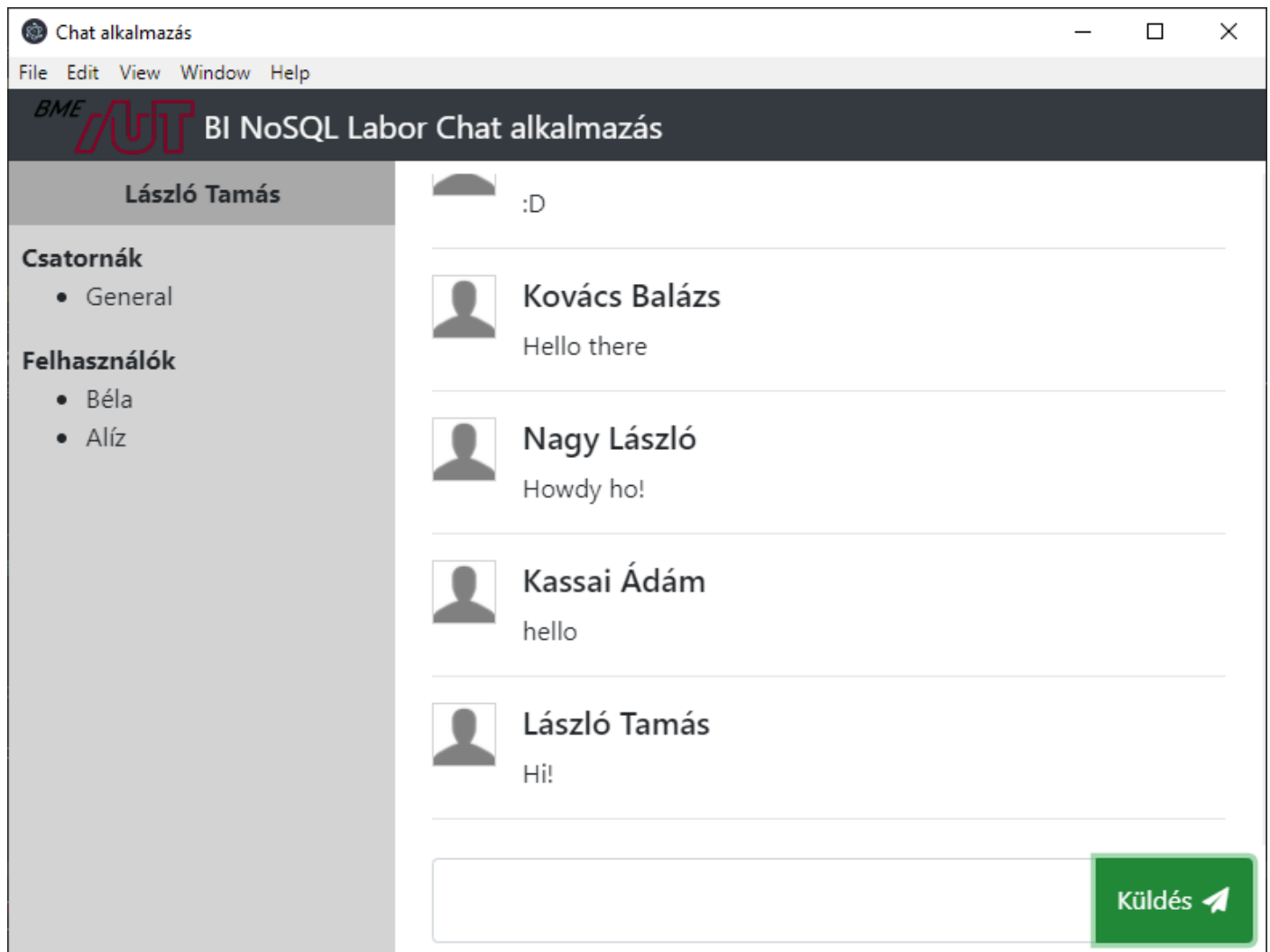


### 3) ADATBÁZIS

#### FELADAT LEÍRÁSA

Első feladatként perzisztáljuk az üzeneteket MongoDB-ben.

#### MEGOLDÁS, MAGYARÁZAT, ÉRTÉKEK



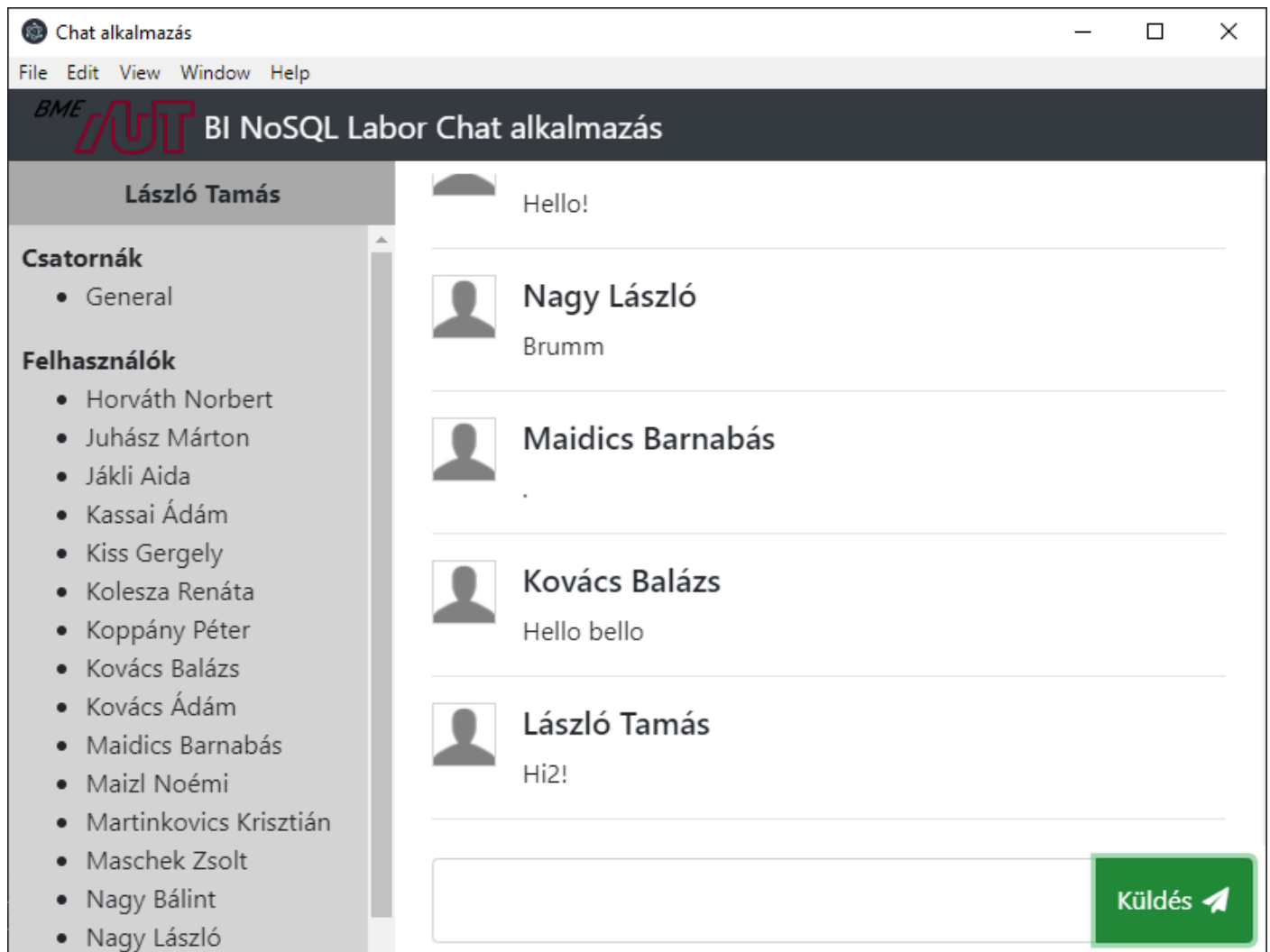
#### 4) VALÓS IDEJŰ KOMMUNIKÁCIÓ

##### FELADAT LEÍRÁSA

Az éppen aktuálisan online felhasználók listáját nem célszerű általában adatbázisban tárolni, hisz gyakran változhat, folyamatosan frissíteni kellhet, ráadásul nem is üzleti adat. A mi alkalmazásunkban éppen ezért ezt egy REDIS kulcs-érték tárral fogjuk megoldani. Ennek két feladata lesz:

- Tárolni az éppen aktuális felhasználók listáját.
- Értesíteni a feliratkozókat ha ez változik.

##### MEGOLDÁS, MAGYARÁZAT, ÉRTÉKEK

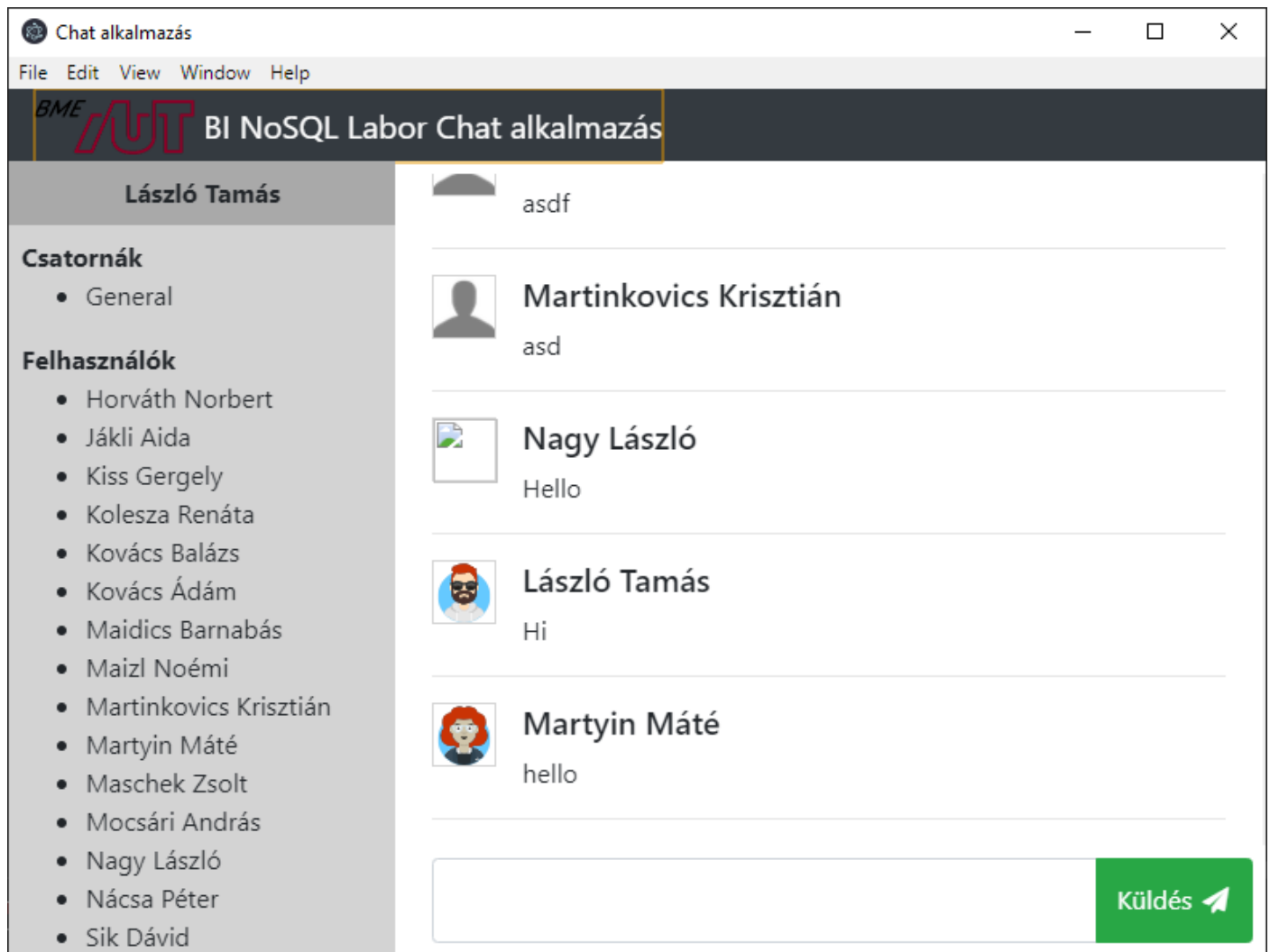


## 5) AVATAROK

### FELADAT LEÍRÁSA

Legyen lehetősége a felhasználónak saját avatar URL megadására, ezt tárolja az adatbázis a Messages collectionben, az egyes dokumentumokban az avatarUrl mező alatt. Figyelj a kulcsok pontos betartására, ha mindenki így implementálja a kliensét, akkor egymás avatar-jai látszódnak másoknál is.

### MEGOLDÁS, MAGYARÁZAT, ÉRTÉKELES



## CHAT-SERVICE.JS

A fájl elején felvettem egy új változót:

```
let myAvatar;
```

És a connect függvényben új paraméterként felvettem és elmentem a fenti változóba.

```
chatService.connect = function (username, serverAddress, password, avatar, successCb, failCb, messageCallback, userCallback) {  
  myUsername = username;  
  myAvatar = avatar;  
}
```

Message modelbe felvettem új propetyként:

```
const Message = mongoose.model('Message', new mongoose.Schema({  
  user: String,  
  date: Date,  
  content: String,  
  room: String,  
  avatarUrl: String  
}));
```

Módosítottam a sendMessage fv-t, hogy fűzze hozzá minden új üzenetemhez az avatarom url-jét.

```
// Üzenetet küld
chatService.sendMessage = function (roomId, message) {
  let msg = new Message({
    user: myUsername,
    date: message.date,
    content: message.content,
    room: roomId,
    avatarUrl: myAvatar
  });
  msg.save().then(function () {
    // Szólunk hogy frissítettük a szobában az üzeneteket
    redisClient.publish(roomsChannel, roomId)
  })
};
```

## CHAT.HTML

Új beviteli mezőt vettem fel

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

## CHAT-CONTROLLER.JS:

Kiolvasom az új beviteli mezőt és átadom a connect fv-nek a saját urlemet.

```
let avatarInput = document.getElementById('avatarInput');

if (_.isEmpty(usernameInput.value) || _.isEmpty(serverInput.value)) {
  alert('Kérlek add meg az összes adatot!');
} else {
  myUsername = _.escape(usernameInput.value);
  chatService.connect(usernameInput.value, serverInput.value, passwordInput.value, avatarInput.value, function () {
```

Továbbá messageRendernél a message objektumokat vizsgálom, hogy létezik-e az avatar url. Ha igen akkor azt az url-t szűröm be a kép helyére, egyébként pedig a defaultot.

```
// Megjelenít egy új üzenetet az üzenő területen
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');
```



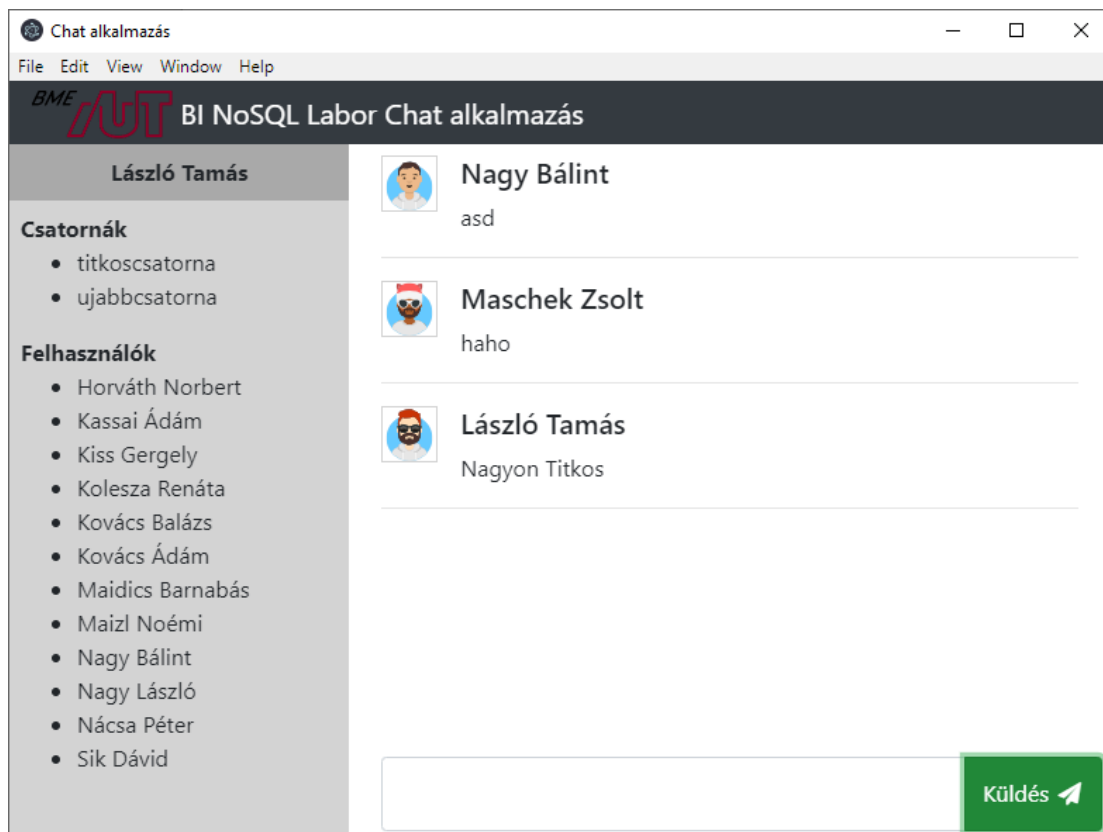
```
let avatarUrl = message.avatarUrl !== undefined? message.avatarUrl : "assets/user.png"
;
// Kitöltünk és hozzáadunk egy új üzenetet a HTML sablon alapján
messageArea.insertAdjacentHTML('beforeEnd',
  '<div class="media messages">' +
  '' +
  '<div class="media-body">' +
  '<h5 class="mt-0">' + _.escape(message.user) + '</h5>' + _.escape(message.content) +
  '</div>' +
  '</div>' +
  '<hr>'
);
```

## 6) CSATORNÁK

### FELADAT LEÍRÁSA

A chat program tetszőleges számú csatornát tud kezelni, ám jelenleg csak a default létezik, valamint az egyes privát üzeneteknek jön létre a háttérben szoba. Valósítsd meg, hogy az alkalmazás a csatornák listáját a MongoDB channels collectionjéből vegye induláskor.

### MEGOLDÁS, MAGYARÁZAT, ÉRTÉKELES



## CHAT-SERVICE.JS

Létrehoztam a channel modelt.

```
const Channel = mongoose.model('Channel', new mongoose.Schema({
  name: String
}));
```

Továbbá létrehoztam a csatornákat lekérő függvény a service-en belül.

```
// Visszaadja a csatornák listáját
chatService.getChannels = function (cb) {
  Channel.find({}, function (err, ch) {
    debugger;
    cb(ch)
  });
};
```

## Chat-controller.js

Létrehoztam egy, a csatornák listáját a felületen frissítő függvényt.

```
// Frissítjük a csatorna lista tartalmát
chatController.refreshChannels = function () {
  document.getElementById('channel-list').innerHTML = '';
  let channelList = document.getElementById('channel-list');
  // Betöltjük a felhasználókat (magunkat nem írjuk ki)
  chatService.getChannels(function (channels) {
    _.forEach(channels, function (channel) {
      channelList.insertAdjacentHTML('beforeEnd', '<li class="selector-panel-item" onclick="chatController.changeRoom(\''+channel.name+'\')">'+channel.name+'</li>');
    });
  });
};
```

Amit mind sikeres csatlakozásnál, mind a felhasználók számának változásakor meghívok.

```
chatService.connect(usernameInput.value, serverInput.value, passwordInput.value, avatarInput.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.refreshRoom();
  chatController.refreshChannels();
});
```

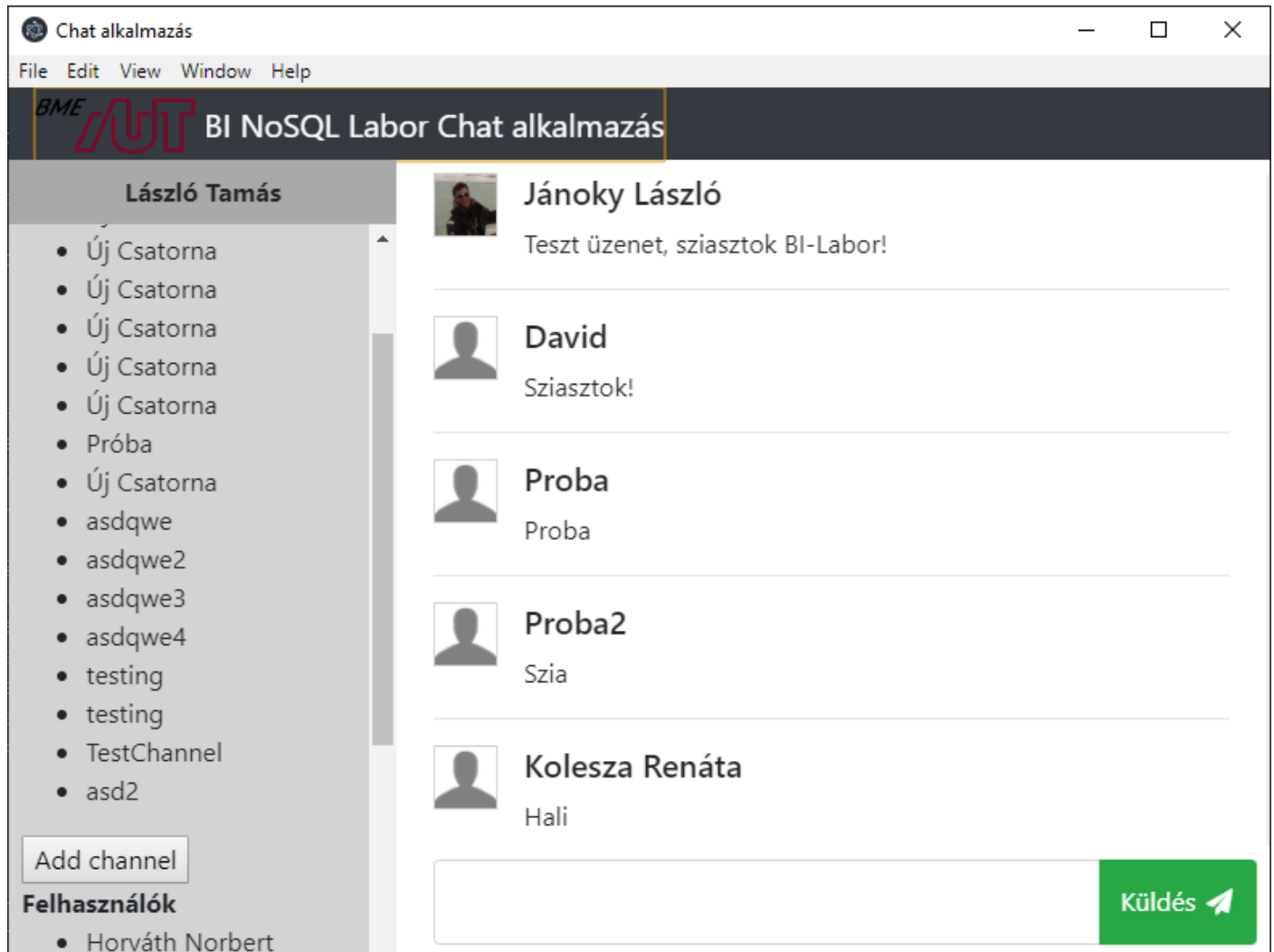
```
    },  
    function (err) {  
        alert("Nem sikerült csatlakozni az adatbázishoz: " + err)  
    },  
    // Új üzenet érkezett valahova (esemény a room_channel-ben)  
    function (roomName) {  
        if (roomName === selectedRoom) {  
            chatController.refreshRoom();  
        }  
    },  
    // Változott a felhasználók száma  
    function () {  
        chatController.refreshUsers();  
        chatController.refreshChannels();  
    });
```

## 7) CSATORNA VÁLTOZÁS FIGYELÉS

### FELADAT LEÍRÁSA

Egészítsd ki az előző funkciót azal, hogy az alkalmazás figyelje a REDIS-es channels\_channel-t, az ide érkező események hatására frissítve a szoba listát.

## MEGOLDÁS, MAGYARÁZAT, ÉRTÉKELES



## CHAT-SERVICE.JS

A connect függvényt kibővítettem egy új paraméterrel (channelCallback). Feliratkoztam a channels\_channel-re és ha arra jön üzenetet, akkor meghívom a callback függvényt.

```
const channelsChannel = 'channels_channel';
```

```
chatService.connect = function (username, serverAddress, password, avatar, successCb, failCb, messageCallback, userCallback, channelCallback) {
  myUsername = username;
  myAvatar = avatar;
  let dbReady = false;
  let mqReady = false;

  let db = mongoose.connect('mongodb://bilabor:' + password + '@' + serverAddress + ':27017/bilabor?authSource=admin', {useNewUrlParser: true, useUnifiedTopology: true});
  redisClient = redis.createClient({
    host: serverAddress, password: password, 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 figyelniünk kell
  // A subscribehhoz külön kliens kell, ezért lemásoljuk az eredetit
  redisSubscriberClient = redisClient.duplicate();
  redisSubscriberClient.subscribe(roomsChannel);
  redisSubscriberClient.subscribe(usersChannel);
  redisSubscriberClient.subscribe(channelsChannel);
  redisSubscriberClient.on('message', function (channel, message) {
    if (channel === roomsChannel) {
      // Ha a szoba channel-
      be érkezik üzenet azt jelenti valamelyik szobába frissíteni kell az üzeneteket
      messageCallback(message);
    } else if (channel === usersChannel) {
      // Ha a user channelbe érkezik üzenet azt jelenti változott a user lista
      userCallback();
    } else if (channel === channelsChannel) {
      // Ha a user channelbe érkezik üzenet azt jelenti változott a user lista
      channelCallback();
    }
  });
});

successCb();
}
```

Emellett létrehoztam egy addChannel függvényt, ami egy új csatornát ad hozzá teszteléshez.

```
chatService.addChannel = function () {
  if (!_.isUndefined(redisClient)) {
    let ch = new Channel({
      name: "testing"
    });
    ch.save().then(function () {
      redisClient.publish(channelsChannel, ch.name);
    })
  }
};
```

## CHAT-CONTROLLER.JS

A refreshChannel függvényt most már az új callback függvényben hívom meg.

```
chatService.connect(usernameInput.value, serverInput.value, passwordInput.value, avatarInput.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.refreshRoom();
    chatController.refreshChannels();
},
function (err) {
    alert("Nem sikerült csatlakozni az adatbázishoz: " + err)
},
// Új üzenet érkezett valahova (esemény a room_channel-ben)
function (roomName) {
    if (roomName === selectedRoom) {
        chatController.refreshRoom();
    }
},
// Változott a felhasználók száma
function () {
    chatController.refreshUsers();
},
// Változott a csatornák száma
function () {
    chatController.refreshChannels();
});
```

Továbbá létrehoztam egy új függvényt a channel beszúrásához.

```
chatController.addChannel = function () {
    chatService.addChannel();
};
```

## CHAT.HTML

Chat beszúráshoz felvettem egy gombot.

```
<div class="selector-panel-body">
    <b>Csatornák</b>
    <ul id="channel-list">
```

```
        <li class="selector-panel-  
item" onclick="chatController.changeRoom('default')">General</li>  
    </ul>  
    <button onclick="chatController.addChannel()">Add channel</button>  
    <br>  
    <b>Felhasználók</b>  
    <ul id="user-list">  
        <!-- Ide jönnek a felhasználók -->  
    </ul>  
</div>
```