

Express chat service demo

Nulladik lépés

1. Mongo DB telepítése
 - a. Indítás: `mongod --dbpath c:\Tools\MongoDB\Data`
2. Webstorm indítás
3. CMD indítás
4. Postman letöltés és betöltése a projektnek

Projekt inicializálás

1. `npm init`
2. Dependency-k hozzáadása
 - a. `npm install express --save`
 - b. `npm install body-parser --save`
 - c. `npm install mongoose --save`
3. `config/index.js` létrehozása

```
var config = {};
```

```
config.port = process.env.PORT || 8080;
```

```
config.mongoUri = process.env.MONGODB_URI || 'mongodb://localhost/chatdemo';
```

```
module.exports = config;
```

4. `server.js` létrehozása
 - a. Server snippet bemásolása
5. `npm start` vagy `node .`
6. Run config létrehozás & indítás
7. Indítás debug módban
 - a. `localhost:8080/api/`

User API

User API endpointok hozzáadása

1. User API létrehozás
 - a. `app/users/index.js`

```
var express = require('express');
```

```
var app = module.exports = express();
```

```
app.get('/', function(req, res) {  
  res.json({  
    message: 'List users'  
  });  
});
```

```
app.post('/', function(req, res) {  
  res.json({  
    message: 'Create new user: ' + req.params.userId  
  });  
});
```

```
app.get('/:userId', function(req, res) {  
  res.json({  
    message: 'Get User by Id: ' + req.params.userId  
  });  
});
```

```

app.put('/:userId', function(req, res) {
  res.json({
    message: 'Update User by id: ' + req.params.userId
  });
});

app.delete('/:userId', function(req, res) {
  res.json({
    message: 'Delete User by id: ' + req.params.userId
  });
});

```

2. API root definiálása
 - a. app/index.js

```

var express = require('express');
var app = module.exports = express();

app.get('/', function(req, res) {
  res.json({
    message: 'API v1 home'
  });
});

app.use('/users', require('./users'));

```

3. API root betöltése server.js-ben

```

var api = require('./app');
app.use('/api/v1/', api);

```

4. Próba! Postmanben és cmd-ben console logot megnézni

User model definiálása

1. Server.js-ben adatbázis kapcsolat beállítása

```

var mongoose = require('mongoose');

mongoose.connect(config.mongoUri);

```

2. User model létrehozás
 - a. app/users/model.js

```

var mongoose = require('mongoose');
var Schema = mongoose.Schema;

var userSchema = new Schema({
  username: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  },
  firstName: String,
  lastName: String,
  imageUrl: String
});

```

```

var Model = mongoose.model('User', userSchema);

module.exports = Model;

```

User API implementálás

```

var express = require('express');
var app = module.exports = express();
var User = require('./model');

app.get('/', function (req, res, next) {
  User.find().exec(function (err, users) {
    if (err) {
      return next(err);
    }
    res.json(users);
  });
});

app.post('/', function (req, res, next) {
  var user = new User({
    username: req.body.username,
    password: req.body.password,
    firstName: req.body.firstName,
    lastName: req.body.lastName,
    imageUrl: req.body.imageUrl
  });
  user.save(function (err, saveEntity) {
    if (err) {
      return next(err);
    }
    res.json(saveEntity);
  });
});

app.get('/:userId', function (req, res, next) {
  User.findById(req.params.userId, function (err, user) {
    if (err) {
      return next(err);
    }
    res.json(user);
  });
});

app.put('/:userId', function (req, res, next) {
  User.findByIdAndUpdate(req.params.userId, {
    $set: {
      username: req.body.username,
      password: req.body.password,
      firstName: req.body.firstName,
      lastName: req.body.lastName,
      imageUrl: req.body.imageUrl
    }
  }, {new: true}, function (err, user) {
    if (err) {
      return next(err);
    }
    res.json(user);
  });
});

```

```

app.delete('/:userId', function (req, res, next) {
  User.findByIdAndRemove(req.params.userId, function (err, result) {
    if (err) {
      return next(err);
    }
    res.json(result);
  }
});
});

```

Teszt és hiba bemutatás!

Mongoose hibakezelés

```

//Mongoose validation error handler middleware
app.use(function(err, req, res, next) {
  //if mongo error
  if(err.code === 11000 || err.name === 'ValidationError') {
    console.error(err.stack);
    return res.status(400).json({
      error: err.errors ? _.map(err.errors, 'message') : 'Validation error',
      message: err.message
    });
  }
  next(err);
});

```

Authentikáció

Passport telepítés és konfiguráció

1. Telepítés
 - a. npm install passport --save
 - b. npm install passport-http --save
2. Server.js konfiguráció
3. `var passport = require('passport');`
`app.use(passport.initialize());`

app/auth/index.js létrehozás

1. Snippet bemásolása
2. verifyPassword implementálása a User modellben

```

userSchema.methods.verifyPassword = function(password, cb) {
  cb(null, password === this.password);
};

```

app/account/index.js létrehozása

1. Snippet bemásolása
2. /me implementálása:

```

User.findById(req.user._id).exec(function (err, user) {
  if (err) {
    return next(err);
  }
  res.json(user);
});

```

3. /register implementálása:

```

var user = new User({
  username: req.body.username,
  password: req.body.password,
  firstName: req.body.firstName,
  lastName: req.body.lastName,
  imageUrl: req.body.imageUrl
});
user.save(function (err, entity) {
  if (err) {
    return next(err);
  }
  res.json(entity);
});

```

app/account/index.js

1. Account regisztrálása, app/index.js-ben

```
app.use('/account', require('./account'));
```

Próba

1. Regisztrálás postbanból
2. Auth adatok megadása postman-ben
3. /me lekérése

Room implementálása, önálló feladat!

1. app/rooms/index.js és app/rooms/model.js
2. Model:

```

var roomSchema = new Schema({
  name: {
    type: String,
    required: true
  },
  creatorId: {
    type: Schema.Types.ObjectId,
    ref: 'User'
  }
});

```

3. API:
 - a. GET /rooms
 - b. POST /rooms
 - c. GET /rooms/:roomId
 - d. PUT /rooms/:roomId, auth required!
 - e. DELETE /rooms/:roomId, auth required!
4. Register room
5. Próba
 - a. 2-3 szoba létrehozása, törlés, módosítás, listázás

Messages

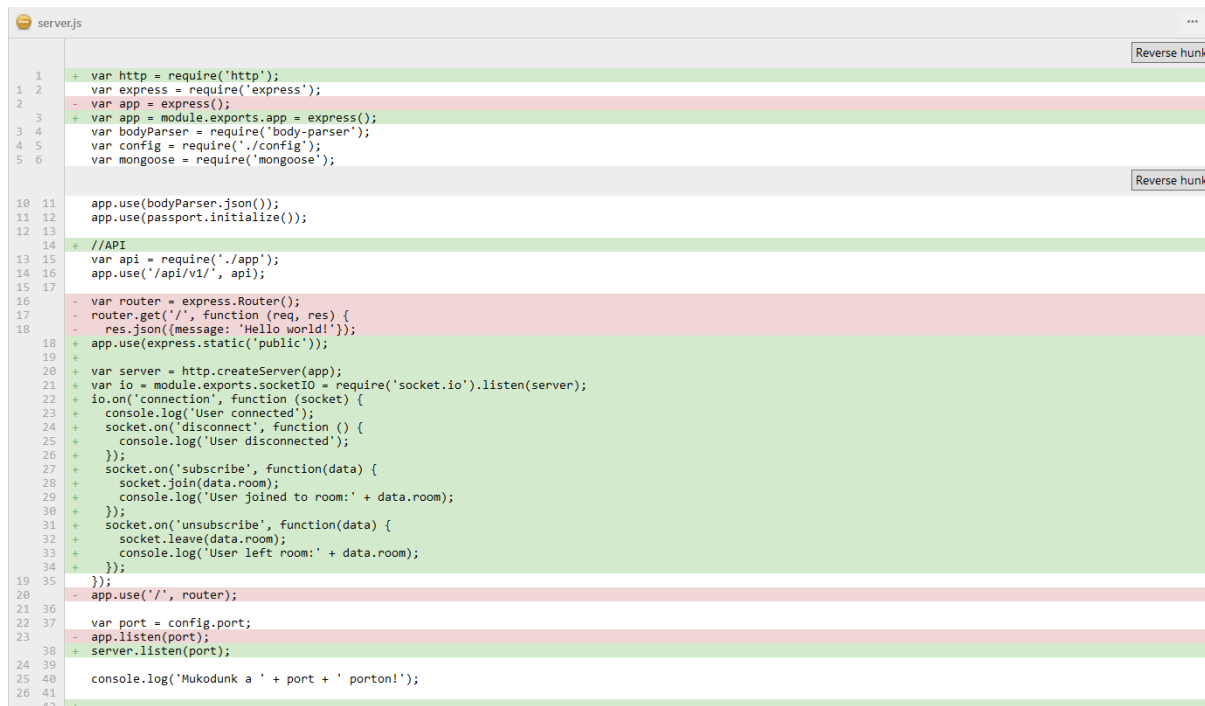
1. app/rooms/messages/model.js
 - a. Snippet beszúrása
2. app/rooms/messages/index.js
 - a. Snippet beszúrása
3. app/rooms/index.js-ben messages regisztrálása

```
app.use('/:roomId/messages', require('./messages'));
```

4. Próba
 - a. Egy szobába üzenet írás, módosítás, törlés, listázás

Socket

1. npm install socket.io --save
2. server.js módosítás
 - a.



```
1  + var http = require('http');
2  var express = require('express');
3  - var app = express();
4  + var app = module.exports.app = express();
5  var bodyParser = require('body-parser');
6  var config = require('./config');
7  var mongoose = require('mongoose');
8
9
10 app.use(bodyParser.json());
11 app.use(passport.initialize());
12
13 + //API
14 var api = require('./app');
15 app.use('/api/v1/', api);
16
17 - var router = express.Router();
18 - router.get('/', function (req, res) {
19   res.json({message: 'Hello world!'});
20 - app.use(express.static('public'));
21 +
22 + var server = http.createServer(app);
23 + var io = module.exports.socketIO = require('socket.io').listen(server);
24 + io.on('connection', function (socket) {
25 +   console.log('User connected');
26 +   socket.on('disconnect', function () {
27 +     console.log('User disconnected');
28 +   });
29 +   socket.on('subscribe', function(data) {
30 +     socket.join(data.room);
31 +     console.log('User joined to room:' + data.room);
32 +   });
33 +   socket.on('unsubscribe', function(data) {
34 +     socket.leave(data.room);
35 +     console.log('User left room:' + data.room);
36 +   });
37 + });
38 - app.use('/', router);
39
40 var port = config.port;
41 - app.listen(port);
42 + server.listen(port);
43
44 console.log('Mukodunk a ' + port + ' porton!');
```

- b. io.on snippet bemásolása
3. Kliens letöltése, és bemásolása a public mappába
 4. App futtatás és teszt

Roomhoz userek hozzáadása

1. app/rooms/users/index.js
2. Snippet bemásolása
3. Próba