



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Ψηφιακών Συστημάτων

Μάθημα: «Δικτυοκεντρικά Πληροφοριακά Συστήματα»

Ακαδημαϊκό έτος 2023-2024

Απαλλακτική Εργασία – Ιανουάριος 2024

Φοιτητές:

Νικόλαος Κόλλιας – E19067

Νικόλαος Ζαγουρής - E20045

Αντώνιος Παπακωνσταντίνου - E20124

Γεώργιος Μαυρωνάς - E20098

Επιβλέπων Καθηγήτρια:

Κα. Κούφη Βασιλική

Περιεχόμενα

Κεφάλαιο 1: Εισαγωγή	3
Κεφάλαιο 2: Περιγραφή του συστήματος που υλοποιήθηκε	4
Κεφάλαιο 3: Υλοποίηση συστήματος	6
Κεφάλαιο 4: Εγχειρίδιο διαχειριστή	45
Κεφάλαιο 5: Εγχειρίδιο χρήστη	47

Κεφάλαιο 1: Εισαγωγή

Η παρούσα εργασία επικεντρώνεται στην ανάπτυξη ενός δικτυοκεντρικού πληροφοριακού συστήματος για τη διαχείριση χρηστών και κρατήσεων ενός γυμναστηρίου με την ονομασία "all 4 all".

Το σύστημα αποτελείται από δύο κύρια υποσυστήματα, το διαχειριστικό και το σύστημα των χρηστών, προσφέροντας λειτουργικότητες τόσο σε χρήστες όσο και σε διαχειριστές.

Για τους χρήστες, το σύστημα παρέχει δυνατότητες όπως περιήγηση στις υπηρεσίες του γυμναστηρίου, Κράτηση προϊόντος/ υπηρεσίας, Ιστορικό κρατήσεων, καθώς επίσης και Νέα/ανακοινώσεις που θα αναρτάει το γυμναστήριο.

Από την άλλη πλευρά, το διαχειριστικό υποσύστημα επιτρέπει στους διαχειριστές να διαχειρίζονται τις εγγραφές των χρηστών, να παρακολουθούν τις κρατήσεις και να διαχειρίζονται το πρόγραμμα του γυμναστηρίου. Τέλος, μέσω του διαχειριστικού συστήματος δίνεται η δυνατότητα για την διαχείριση των ανακοινώσεων που θα εκδίδει το γυμναστήριο.

Με αυτόν τον τρόπο, το σύστημα εξασφαλίζει αποτελεσματική και ολοκληρωμένη διαχείριση των δραστηριοτήτων του γυμναστηρίου "all 4 all", προσφέροντας ταυτόχρονα βελτιωμένη εμπειρία χρήσης για τους εγγεγραμμένους χρήστες και ευκολία στη διαχείριση για τους διαχειριστές.

Κεφάλαιο 2: Περιγραφή του συστήματος που υλοποιήθηκε

Όπως είδαμε στην εισαγωγή, το δικτυοκεντρικό πληροφοριακό σύστημα αναπτύχθηκε για τη διαχείριση χρηστών και κρατήσεων του γυμναστηρίου "all 4 all".

Το σύστημα αποτελείται από δύο κύρια υποσυστήματα:

1. Το διαχειριστικό σύστημα
2. Το σύστημα των χρηστών.

Κάθε υποσύστημα υποστηρίζει συγκεκριμένες λειτουργίες που προορίζονται είτε για τους διαχειριστές, είτε τους απλούς χρήστες του γυμναστηρίου.

Ας δούμε αναλυτικά τις λειτουργίες για το διαχειριστικό σύστημα:

1. Διαχείριση Αιτημάτων Εγγραφής

Ο διαχειριστής έχει τη δυνατότητα να διαχειρίζεται τα αιτήματα εγγραφής στο σύστημα. Αποφασίζει εάν θα εγκρίνει ή απορρίψει αιτήματα από νέους χρήστες.

2. Διαχείριση Χρηστών Συστήματος

Ο διαχειριστής έχει πρόσβαση στα στοιχεία των χρηστών και ευθύνεται για τη διαχείριση των πληροφοριών τους. Εκτελεί βασικές λειτουργίες CRUD (Create, Read, Update, Delete) για τα προσωπικά δεδομένα και μπορεί να αναθέσει ή να τροποποιήσει τους ρόλους των χρηστών. Αλληλοεπιδράει δηλαδή με την βάση δεδομένων. Θα δούμε αναλυτικές πληροφορίες για την βάση μας παρακάτω.

3. Διαχείριση Δομικών Στοιχείων

Ο διαχειριστής διαχειρίζεται τα δομικά στοιχεία του γυμναστηρίου, περιλαμβανομένων των γυμναστών, των διαθέσιμων προγραμμάτων, και του προγράμματος ομαδικών προγραμμάτων. Εκτελεί λειτουργίες CRUD για τη διατήρηση και την επεξεργασία των πληροφοριών.

4. Διαχείριση Ανακοινώσεων/Προσφορών

Ο διαχειριστής μπορεί να αναρτά ανακοινώσεις και προσφορές που θα είναι ορατές από τους χρήστες. Επιπλέον, έχει τη δυνατότητα να τροποποιεί και να διαγράφει ανακοινώσεις.

Ας δούμε, τώρα, αναλυτικά και τις λειτουργίες για το σύστημα των χρηστών:

1. Περιήγηση στις Υπηρεσίες του Γυμναστηρίου

Ο απλός χρήστης μπορεί να προβάλει τις υπηρεσίες του γυμναστηρίου, όχι όμως το πρόγραμμα των προγραμμάτων, χωρίς να είναι εγγεγραμμένος ή χωρίς να έχει συνδεθεί.

Σημαντικό, επίσης, είναι το γεγονός πως ακόμα και αν έχει κάνει την διαδικασία του /sign_up (θα το δούμε παρακάτω) για να κάνει σύνδεση, θα πρέπει πρώτα ο admin να έχει εγκρίνει την εγγραφή του.

2. Κράτηση Προϊόντος/Υπηρεσίας

Ο χρήστης μπορεί να κλείσει ραντεβού για τα προγράμματα του γυμναστηρίου, αναζητώντας τη διαθεσιμότητα και κάνοντας κράτηση. Υπάρχουν περιορισμοί στις ακυρώσεις.

3. Ιστορικό Κρατήσεων

Ο χρήστης μπορεί να δει το ιστορικό όλων των κρατήσεων που έχει πραγματοποιήσει στο σύστημα.

4. Νέα/Ανακοινώσεις

Ο χρήστης μπορεί να βλέπει νέα και ανακοινώσεις που έχουν αναρτηθεί από το γυμναστήριο.

Με αυτόν τον τρόπο, υλοποιήθηκε το δικτυοκεντρικό πληροφοριακό σύστημα και καλύπτει ένα ευρύ φάσμα λειτουργιών για την

ολοκληρωμένη διαχείριση του γυμναστηρίου "all 4 all", προσφέροντας ταυτόχρονα μία όμορφη εμπειρία πλοήγησης στον χρήστη.

Κεφάλαιο 3: Υλοποίηση συστήματος

Τόσο το σύστημα του χρήστη όσο και το σύστημα του διαχειριστή χρησιμοποιεί τις εξής τεχνολογίες:

1. Express.js:

Στον server-side κώδικα, χρησιμοποιούμε το Express.js για τον ορισμό των διαδρομών.

Το EJS, ή ενσωματωμένη JavaScript, είναι μια δημοφιλής μηχανή προτύπων για το Node.js και την ανάπτυξη ιστού. Μας επιτρέπει να δημιουργούμε δυναμικό περιεχόμενο HTML ενσωματώνοντας κώδικα JavaScript στα HTML αρχεία μας.

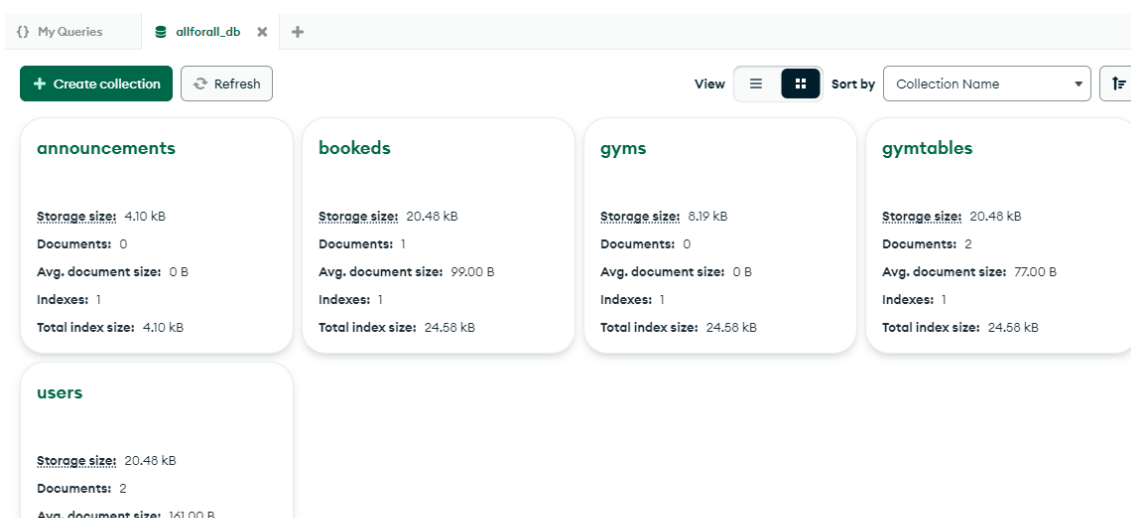
2. MongoDB & Mongoose:

Η βάση δεδομένων είναι υλοποιημένη με τη χρήση της MongoDB.

Το όνομα της βάσης μας είναι **allforall_db**, και αποτελείται από **5 collections**. Αυτά είναι τα:

1. **users**
2. **gyms**
3. **bookeds**
4. **announcements**
5. **gyms**

Ακολουθεί η αντίστοιχη φωτογραφία:



Το Mongoose χρησιμοποιείται για τη σύνδεση και τον ορισμό του μοντέλου στον server-side κώδικα.

3. HTML Templating με EJS:

Το EJS (Embedded JavaScript) χρησιμοποιείται για τη δημιουργία δυναμικών HTML σελίδων.

Το EJS (Embedded JavaScript) είναι ένα πλαίσιο εργασίας για το templating των HTML σελίδων στο πλαίσιο μιας web εφαρμογής.

Η κύρια λειτουργία του EJS είναι να επιτρέπει την ενσωμάτωση του JavaScript κώδικα μέσα στα HTML templates, καθιστώντας τα πιο δυναμικά και ευέλικτα.

4. Front-end:

Στα αρχεία .ejs χρησιμοποιούμε Bootstrap για τη βελτιστοποίηση του σχεδιασμού.

Η σελίδα παρουσιάζει πληροφορίες ανάλογες κάθε φορά με την σελίδα. Υπάρχουν διάφορα αρχεία .ejs που θα τα δούμε αναλυτικότερα παρακάτω.

5. Επικοινωνία με τη Βάση Δεδομένων:

Η επικοινωνία με τη βάση δεδομένων MongoDB γίνεται στον περισσότερο server-side κώδικα μέσω του Mongoose, που είναι ένα ODM (Object Data Modeling) βιβλιοθήκη για τη MongoDB.

Ο Mongoose προσφέρει ένα πιο αφαιρετικό και ευέλικτο τρόπο για την αλληλεπίδραση με τη βάση δεδομένων MongoDB στο πλαίσιο της εφαρμογής μας σε Node.js.

Τα βασικά στάδια της επικοινωνίας με τη βάση δεδομένων MongoDB με χρήση του Mongoose περιλαμβάνουν τα εξής:

1. Σύνδεση στη Βάση Δεδομένων:

Η πρώτη ενέργεια είναι η σύνδεση της εφαρμογής Node.js στη βάση δεδομένων MongoDB.

Αυτό γίνεται με τη χρήση του `mongoose.connect()`.

```
1  const express = require("express");
2  const app = express();
3  const mongoose = require("mongoose");
4  const bodyParser = require("body-parser");
5  const ejs = require('ejs');
6  const session = require('express-session');
7  const { kStringMaxLength } = require('buffer');
8
9  app.use(session({
10     secret: 'mysecret',
11     resave: false,
12     saveUninitialized: false
13 }));
14
15 let login = null;
16 let gymtablelist;
17 let gymlist;
18 let userslist;
19 let historyslist;
20 let bookslist;
21
22 app.use(bodyParser.urlencoded({extended: true}));
23
24 app.set('view engine', 'ejs');
25
26 mongoose.connect("mongodb://localhost:27017/allforall_db")
27
```

2. Ορισμός Μοντέλων:

Τα μοντέλα Mongoose καθορίζουν τη δομή των δεδομένων στη βάση δεδομένων.

Καθορίζουν τα collections, τα πεδία των εγγραφών και τις επιλογές για τον τρόπο αποθήκευσης.

Για παράδειγμα:

```
28 // create data schema
29 const announcementSchema = {
30   title: String,
31   content: String
32 }
33
34 const UserSchema = {
35   name: String,
36   password: String,
37   email: String,
38   phone: String,
39   address: String,
40   country: String,
41   username: String,
42   lastname: String,
43   cancellations: Number,
44   status: String,
45   role: String
46 }
47
48 const gymtableSchema = {
49   ex: String,
50   time: String,
51
```


Στην συνέχεια μπορούμε να κάνουμε όλες τις ενέργειες CRUD στη βάση δεδομένων μας.

Θα δούμε πρώτα την υλοποίηση του συστήματος διαχειριστή.

Για εξοικονόμηση χώρου , η ακόλουθη φωτογραφία είναι 35 γραμμές σταθερού κώδικα οι οποίες χρησιμοποιούνται σε κάθε .ejs αρχείο που αφορά τον admin. Οι λειτουργίες της κάθε σελίδας θα παραθέτονται σε ξεχωριστή φωτογραφία.

```
1 <!DOCTYPE html>
2 <% if (role == "admin") { %>
3 <html lang="en">
4
5 <head>
6   <meta charset="UTF-8">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>Document</title>
9   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
10     integrity="sha384-BVYiiSIFeK1dGmJRAKycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
11 </head>
12
13 <body>
14
15   <nav class="navbar navbar-default navbar-static-top">
16     <div class="container-fluid">
17       <div class="navbar-header">
18         <a class="navbar-brand" href="mainpage_style.html">
19           <!-- 
20         </a>
21       </div>
22       <div class="collapse navbar-collapse">
23         <ul class="nav navbar-nav navbar-right">
24           <li><a href="/admin_announ">Publish Announcements</a></li>
25           <li><a href="/admin_req">Register Requests</a></li>
26           <li><a href="/admin_users">Users Managment</a></li>
27           <li><a href="/admin_trainning">Admin Training</a></li>
28           <li><a href="/admin_program">Managment Program</a></li>
29           <li><a href="/logout">Sign-Out</a></li>
30         </ul>
31       </div>
32     </div>
33   </nav>
34
35   <h2 style="text-align: center;">Register Requests</h2>
```

1. Διαχείριση αιτημάτων εγγραφής στο σύστημα:

Στο κομμάτι του Server ο κώδικας υλοποιείται ως εξής:

```
294 //ADMIN REQUESTS
295 app.get('/admin_req', (req, res) => {
296
297     User.find().then((users)=> {
298         res.render('admin_requests', {
299             usersList: users,
300             email: req.session.email,
301             role: req.session.role
302         })
303     })
304 })
305
306
307 app.post('/checking_req', (req, res) => {
308
309     const email = req.body.email;
310     const status = req.body.status;
311     const check = req.body.c;
312
313     if (check == "approve") {
314         User.updateOne({ email: email }, {
315             $set: {
316                 status: "accepted"
317             }
318         }).then(() => {
319             res.redirect("/admin_req");
320         });
321     } else if (check == "decline") {
322         User.updateOne({ email: email }, {
323             $set: {
324                 status: "rejected"
325             }
326         }).then(() => {
327             res.redirect("/admin_req");
328         });
329     }
330
331 });
```

Στο κομμάτι που βλέπει ο χρήστης ο κώδικας υλοποιείται ως εξής:

Το αρχείο ονομάζεται **admin_requests**:

```
37 <table class="table">
38   <thead>
39     <tr>
40       <th scope="col">Email</th>
41       <th scope="col">Name</th>
42       <th scope="col">Phone</th>
43       <th scope="col">Status</th>
44     </tr>
45   </thead>
46   <tbody>
47     <%usersList.forEach(user=> {%>
48       <% if (user.role != "admin") {%>
49         <tr>
50           <th scope="row">
51             <%= user.email %>
52           </th>
53           <td>
54             <%= user.name %>
55           </td>
56           <td>
57             <%= user.phone %>
58           </td>
59           <td>
60             <%= user.status %>
61           </td>
62           <td>
63             <form action="/checking_req" method="post">
64               <input type="hidden" name="email" value="<%= user.email %>">
65               <button type="submit" value="decline" name="c" >Decline</button>
66               <button type="submit" value="approve" name="c" >Accept</button>
67             </form>
68           </td>
69         </tr>
70       <%}%>
71     <%}%>
72   </tbody>
73 </table>
74 </body>
75 </html>
76 </html>
77 <%}%>
```

Αυτό που γίνεται στην ουσία είναι το εξής. Στο κομμάτι του server ξεκινάμε και μέσω της μεθόδου get καλούμε το endpoint /admin_req. Αυτό στο σώμα του , κάνει μία αναζήτηση στο collection με το όνομα users και στην συνέχεια εφόσον υπάρχουν χρήστες , θα γίνει render η σελίδα admin_requests την οποία θα βλέπει ο admin.

Σε αυτή τη σελίδα , όπως βλέπουμε , θα υπάρχει ένας πίνακας με τα στοιχεία του χρήστη ο οποίος έχει κάνει εγγραφή (email, name, phone,

status). Από εκεί και πέρα είναι στο χέρι του admin εάν θα κάνει accept ή reject το αίτημα του.

Τέλος , μέσω της μεθόδου post καλούμε το /checking_req , που στην ουσία , είναι το action της φόρμας που βλέπει ο admin.

Έτσι, αναλόγως πιο κουμπί θα πατήσει ο admin (accept ή decline) θα αλλάζει η στήλη status , του αντίστοιχου user. Και ταυτόχρονα , θα ενημερώνονται τα στοιχεία στη βάση δεδομένων μας.

2. Διαχείριση χρηστών συστήματος

Στην πλευρά του server ο κώδικας είναι ο εξής:

```
333 //ADMIN USERS MANAGMENT
334 app.get('/admin_users', (req, res) => {
335
336     User.find().then((users)=> {
337         res.render('admin_users', {
338             usersList: users,
339             email: req.session.email,
340             role: req.session.role
341         })
342     })
343 })
344
```

```
347 //ADMIN USERS UPDATE
348 app.post('/update_users', (req, res) => {
349
350     const email = req.body.email;
351     User.findOne({ email: email }).then((user) => {
352         console.log("ok!");
353         if (user) {
354             console.log("ok2!");
355             User.updateOne({ email: email }, {
356                 $set: {
357                     username: req.body.username,
358                     email: req.body.email,
359                     phone: req.body.phone,
360                     lastname: req.body.lastname,
361                     name: req.body.name,
362                     country: req.body.country,
363                     address: req.body.address,
364                     cancellations: req.body.cancellations,
365                     role: req.body.role
366                 }
367             }).then(() => {
368                 res.redirect("/admin_users");
369             });
370
371         }
372     }
373     else {
374         res.send(`<script>alert('Email does not exist. Try again.');
```

```

381 //delete user
382
383 app.post('/delete_user', (req, res) => {
384
385     const email = req.body.email;
386     User.findOne({ email: email }).then((user) => {
387         if (user) {
388             User.deleteOne({ email: email }).then(() => {
389                 res.redirect("/admin_users");
390             });
391         }
392         else {
393             res.send(`<script>alert('Email does not exist. Try again.');

```

Στο κομμάτι του χρήστη ο κώδικας είναι ο εξής και βρίσκεται στο αρχείο **admin_users.ejs**

```

35     <h2 style="text-align: center;">Users Managment</h2>
36
37     <table class="table">
38         <thead>
39             <tr>
40                 <th scope="col">Email</th>
41                 <th scope="col">Name</th>
42                 <th scope="col">Phone</th>
43                 <th scope="col">Address</th>
44                 <th scope="col">Country</th>
45                 <th scope="col">Username</th>
46                 <th scope="col">Lastname</th>
47                 <th scope="col">Cancellations</th>
48                 <th scope="col">Role</th>
49                 <th scope="col">Status</th>
50             </tr>
51         </thead>

```

```

52 <tbody>
53 <%usersList.forEach(user=> {%
54 <tr>
55 <th scope="row">
56 <%= user.email %>
57 </th>
58 <td>
59 <%= user.name %>
60 </td>
61 <td>
62 <%= user.phone %>
63 </td>
64 <td>
65 <%= user.address %>
66 </td>
67 <td>
68 <%= user.country %>
69 </td>
70 <td>
71 <%= user.username %>
72 </td>
73 <td>
74 <%= user.lastname %>
75 </td>
76 <td>
77 <%= user.cancellations %>
78 </td>
79 <td>
80 <%= user.role %>
81 </td>
82 <td>
83 <%= user.status %>
84 </td>
85 <td>
86 <form action="/delete_user" method="post">
87 <input type="hidden" name="email" value="<%= user.email %>">
88 <button type="submit" value="delete" name="delete" >Delete</button>
89 </form>
90 </td>
91 </tr>
92 <%}%>
93 </tbody>
94 </table>
95

```

```

99 <div class="container">
100 <div class="row">
101 <div class="col-md-6 col-md-offset-3">
102 <div class="panel panel-default">
103 <div class="panel-heading">
104 <h3 class="panel-title">Update User</h3>
105 </div>
106 <div class="panel-body">
107 <form action="/update_users" method="POST">
108 <div class="form-group">
109 <label for="username">Username</label>
110 <input type="text" class="form-control" id="username" name="username" required>
111 </div>
112 <div class="form-group">
113 <label for="email">Email address</label>
114 <input type="email" class="form-control" id="email" name="email" required>
115 </div>
116 <div class="form-group">
117 <label for="lastname">Lastname</label>
118 <input type="text" class="form-control" id="lastname" name="lastname" required>
119 </div>
120
121 <div class="form-group">
122 <label for="name">Name</label>
123 <input type="text" class="form-control" id="name" name="name" required>
124 </div>
125
126 <div class="form-group">
127 <label for="phone">Phone-number</label>
128 <input type="text" class="form-control" id="phone" name="phone" required>
129 </div>
130
131 <div class="form-group">
132 <label for="country">Country</label>
133 <input type="text" class="form-control" id="country" name="country" required>
134 </div>
135
136 <div class="form-group">
137 <label for="address">Address</label>
138 <input type="text" class="form-control" id="address" name="address" required>
139 </div>
140
141 <div class="form-group">
142 <label for="cancellations">Cancellations</label>
143 <input type="number" class="form-control" id="cancellations" name="cancellations" required>
144 </div>
145
146 <div class="form-group">
147 <label for="role">Role</label>
148 <input type="text" class="form-control" id="role" name="role" required>
149 </div>
150

```

Ο παραπάνω κώδικας του server, αναφέρεται σε λειτουργίες που αφορούν τη διαχείριση χρηστών από τον διαχειριστή του συστήματος.

Συγκεκριμένα:

1. Εμφάνιση Χρηστών στον Πίνακα Διαχείρισης (με την μέθοδο GET /admin_users):

Δημιουργεί μια διαδρομή στο /admin_users που αντιστοιχεί σε HTTP GET αίτηση.

Καλεί τον server να αναζητήσει όλους τους χρήστες στη βάση δεδομένων με χρήση της μεθόδου User.find().

Αν υπάρχουν χρήστες, επιστρέφει τη σελίδα admin_users.ejs με τα δεδομένα των χρηστών, συμπεριλαμβανομένων των email, ονόματος, τηλεφώνου και ρόλου.

Τα δεδομένα του συνδεδεμένου διαχειριστή (email και ρόλος) περνιούνται επίσης στη σελίδα.

2. Ενημέρωση Πληροφοριών Χρήστη (POST /update_users):

Δημιουργεί μια διαδρομή στο /update_users που αντιστοιχεί σε HTTP POST αίτηση. Η φόρμα αυτή είναι στο admin_users.ejs

Παίρνει τα δεδομένα από τη φόρμα ενημέρωσης χρήστη (όπως username, email, phone, κλπ.) με χρήση της req.body.

Αναζητά τον χρήστη στη βάση δεδομένων με βάση το email και, αν υπάρχει, ενημερώνει τα στοιχεία του με τα νέα δεδομένα.

Επιστρέφει στη σελίδα /admin_users μετά την ενημέρωση.

3. Διαγραφή Χρήστη (POST /delete_user):

Δημιουργεί μια διαδρομή στο /delete_user που αντιστοιχεί σε HTTP POST αίτηση. Η φόρμα αυτή είναι στο admin_users.ejs

Παίρνει το email από τη φόρμα διαγραφής χρήστη με χρήση της req.body.

Αναζητά τον χρήστη στη βάση δεδομένων με βάση το email και, αν υπάρχει, τον διαγράφει.

Επιστρέφει στη σελίδα /admin_users μετά τη διαγραφή.

Αυτές οι λειτουργίες επιτρέπουν στον διαχειριστή να διαχειρίζεται τους χρήστες του συστήματος, εμφανίζοντας, ενημερώνοντας και διαγράφοντας τα δεδομένα των χρηστών.

3. Διαχείριση δομικών στοιχείων του συστήματος:

Στην πλευρά του server , ο κώδικας είναι ο εξής:

```
401 //admin training
402 app.get('/admin_training', (req, res) => {
403
404     Gym.find().then((gyms)=> {
405         res.render('admin_training', {
406             gymList: gyms,
407             email: req.session.email,
408             role: req.session.role
409         })
410     })
411 })
412
413 //update training
414 app.post('/update_training', (req, res) => {
415
416     const exersice = req.body.exersice;
417     const trainer = req.body.trainer;
418     const category = req.body.category;
419
420     Gym.findOne({ ex: exersice }).then((gym) => {
421         if (gym) {
422             Gym.updateOne({ ex: exersice }, {
423                 $set: {
424                     trainer: trainer,
425                     category: category
426                 }
427             }).then(() => {
428                 res.redirect("/admin_training");
429             });
430         } else {
431             res.send('<script>alert('Exersice is not available. Try again.');
```



```

458 //create a new exersice
459 app.post('/add_trainning', (req, res) => {
460   const exersice = req.body.exersice;
461   const trainer = req.body.trainer;
462   const category = req.body.category;
463
464   Gym.findOne({ exersice: exersice }).then((gym) => {
465     if (gym) {
466       res.send('<script>alert('Exersice is already taken');window.location='/admin_trainning';</script>');
467     }
468     else {
469       let newGym = new Gym({
470         ex: exersice,
471         trainer: trainer,
472         category: category
473       });
474       newGym.save();
475       res.redirect("/admin_trainning");
476     }
477   })
478 })
479
480
481
482 app.get('/admin_program', async (req, res) => {
483
484   const gyms = await Gym.find();
485   gymlist = gyms;
486
487   const timeSlots = [
488     { time: '9:00', booked: false },
489     { time: '9:30', booked: true },
490     { time: '10:00', booked: false },
491     { time: '10:30', booked: false },
492     { time: '11:00', booked: false },
493   ];
494
495   Gymtable.find().then((gymTable)=> {
496     res.render('admin_program', {
497       gymtablelist: gymTable,
498       email: req.session.email,
499       role: req.session.role,
500       timeSlots: timeSlots,
501       gymlist: gymlist
502     })
503   })
504 })
505

```

```

507 app.post('/create_table', (req, res) => {
508
509   const time = req.body.time;
510   const day = req.body.day;
511   const ex = req.body.exersice;
512   const email = req.session.email;
513   const user = "";
514
515   Gymtable.findOne({ time: time, day: day }).then((gymtable) => {
516
517     if (gymtable) {
518       res.send('<script>alert('Table is already taken');window.location='/admin_program';</script>');
519     }
520     else {
521       let newGymtable = new Gymtable({
522         ex: ex,
523         time: time,
524         day: day,
525         user: user
526       });
527       newGymtable.save();
528       res.redirect("/admin_program");
529     }
530   })
531 })
532
533 app.post('/delete_table', (req, res) => {
534
535   const id = req.body.id;
536
537   Gymtable.findOne({ _id: id }).then((gymtable) => {
538     if (gymtable) {
539       Gymtable.deleteOne({ _id: id }).then(() => {
540         res.redirect("/admin_program");
541       });
542     }
543     else {
544       res.send('<script>alert('Table is not available. Try again.');

```

Ο admin από την πλευρά του θα βλέπει τα εξής μέσω των 2 .ejs αρχείων. Αυτά τα αρχεία είναι τα:

1. admin_training

```
37 <table class="table">
38   <thead>
39     <tr>
40       <th scope="col">ID</th>
41       <th scope="col">Exercise</th>
42       <th scope="col">Trainer</th>
43       <th scope="col">Category</th>
44     </tr>
45   </thead>
46   <tbody>
47     <%gymList.forEach(gym=> {%)>
48       <tr>
49         <th scope="row">
50           <%= gym._id %>
51         </th>
52         <td>
53           <%= gym.ex %>
54         </td>
55         <td>
56           <%= gym.trainer %>
57         </td>
58         <td>
59           <%= gym.category %>
60         </td>
61         <td>
62           <form action="/delete_training" method="post">
63             <input type="hidden" name="exercise" value="<%= gym.ex %>">
64             <button type="submit" value="delete" name="delete">Delete</button>
65           </form>
66         </td>
67       </tr>
68     <%}%>
69   </tbody>
70 </table>
71 <br>
72 <br>
73 <br>
```

```
74 <div class="container">
75   <div class="row">
76     <div class="col-md-6 col-md-offset-3">
77       <div class="panel panel-default">
78         <div class="panel-heading">
79           <h3 class="panel-title">New Training</h3>
80         </div>
81         <div class="panel-body">
82           <form action="/add_training" method="POST">
83             <div class="form-group">
84               <label for="exercise">Exercise</label>
85               <input type="text" class="form-control" id="exercise" name="exercise" required>
86             </div>
87             <div class="form-group">
88               <label for="trainer">Trainer name</label>
89               <input type="text" class="form-control" id="trainer" name="trainer" required>
90             </div>
91             <div class="form-group">
92               <label for="category">Category</label>
93               <select id="category" name="category">
94                 <option value="cardio">Cardio</option>
95                 <option value="weight">Weight</option>
96               </select>
97             </div>
98             <button type="submit" class="btn btn-primary btn-block">Create</button>
99           </form>
100         </div>
101       </div>
102     </div>
103   </div>
104 </div>
105
106 <div class="col-md-6 col-md-offset-3">
107   <div class="panel panel-default">
108     <div class="panel-heading">
109       <h3 class="panel-title">Update Training</h3>
110     </div>
111     <div class="panel-body">
112       <form action="/update_training" method="POST">
113         <div class="form-group">
114           <label for="exercise">Exercise</label>
115           <input type="text" class="form-control" id="exercise" name="exercise" required>
116         </div>
117         <div class="form-group">
118           <label for="trainer">Trainer name</label>
119           <input type="text" class="form-control" id="trainer" name="trainer" required>
120         </div>
121         <div class="form-group">
122           <label for="category">Category</label>
```

2. admin_program

```
36 <h2 style="text-align: center;">Managment Program</h2>
37
38 <table class="table">
39   <thead>
40     <tr>
41       <th scope="col">ID</th>
42       <th scope="col">Exersice</th>
43       <th scope="col">Time</th>
44       <th scope="col">Day</th>
45       <th scope="col">User</th>
46     </tr>
47   </thead>
48   <tbody>
49     <%gymtablelist.forEach(gymTable=> {%)>
50       <tr>
51         <th scope="row">
52           <%= gymTable.id %>
53         </th>
54         <td>
55           <%= gymTable.ex %>
56         </td>
57         <td>
58           <%= gymTable.time %>
59         </td>
60         <td>
61           <%= gymTable.day %>
62         </td>
63         <td>
64           <%= gymTable.user %>
65         </td>
66         <td>
67           <form action="/delete_table" method="post">
68             <input type="hidden" name="id" value="<%= gymTable.id %>">
69             <button type="submit" value="delete" name="delete" >Delete</button>
70           </form>
71         </td>
72       </tr>
73     <%}%>
74   </tbody>
75 </table>
76 <br>
77 <br>
78 <br>
79
```

```

83
84 <div class="row">
85   <div class="col-md-12">
86     <div class="panel panel-default">
87       <div class="panel-heading">
88         <h3 class="panel-title">Time Table</h3>
89       </div>
90
91       <table class="table table-bordered">
92         <thead>
93           <tr>
94             <th>Time</th>
95             <th>Monday</th>
96             <th>Tuesday</th>
97             <th>Wednesday</th>
98             <th>Thursday</th>
99             <th>Friday</th>
100           </tr>
101         </thead>
102         <tbody>
103           <% timeSlots.forEach(slot-> { %>
104             <tr>
105               <td>
106                 <% slot.time%>
107               </td>
108               <td>
109                 <% gymtablelist.forEach(exercise-> { %>
110                   <% if (exercise.day=="Monday" && exercise.time==slot.time) { %>
111                     <% if (exercise.user != "") { %>
112                       <span style="color: red"><% exercise.ex %></span>
113                     <% } else { %>
114                       <% exercise.ex %>
115                     <% } %>
116                   <% } %>
117                 <% }) %>
118               </td>
119               <td>
120                 <% gymtablelist.forEach(exercise-> { %>
121                   <% if (exercise.day=="Tuesday" && exercise.time==slot.time) { %>
122                     <% if (exercise.user != "") { %>
123                       <span style="color: red"><% exercise.ex %></span>
124                     <% } else { %>
125                       <% exercise.ex %>
126                     <% } %>
127                   <% } %>
128                 <% }) %>
129               </td>
130               <td>
131                 <% gymtablelist.forEach(exercise-> { %>
132                   <% if (exercise.day=="Wednesday" && exercise.time==slot.time) { %>
133                     <% if (exercise.user != "") { %>
134                       <span style="color: red"><% exercise.ex %></span>
135                     <% } else { %>
136                       <% exercise.ex %>
137                     <% } %>
138                   <% } %>
139                 <% }) %>
140               </td>
141               <td>
142                 <% gymtablelist.forEach(exercise-> { %>
143                   <% if (exercise.day=="Thursday" && exercise.time==slot.time) { %>
144                     <% if (exercise.user != "") { %>
145                       <span style="color: red"><% exercise.ex %></span>
146                     <% } else { %>
147                       <% exercise.ex %>
148                     <% } %>
149                   <% } %>
150                 <% }) %>
151               </td>
152               <td>
153                 <% gymtablelist.forEach(exercise-> { %>
154                   <% if (exercise.day=="Friday" && exercise.time==slot.time) { %>
155                     <% if (exercise.user != "") { %>
156                       <span style="color: red"><% exercise.ex %></span>
157                     <% } else { %>
158                       <% exercise.ex %>
159                     <% } %>
160                   <% } %>
161                 <% }) %>
162               </td>
163             </tr>
164             <% }) %>
165           </tbody>
166         </table>
167       </div>
168     </div>
169   </div>

```

```

172 <div class="col-md-6 col-md-offset-3">
173   <div class="panel panel-default">
174     <div class="panel-heading">
175       <h3 class="panel-title">Update Training</h3>
176     </div>
177     <div class="panel-body">
178       <form action="/create_table" method="POST">
179         <label for="exersice">Service:</label>
180         <select id="exersice" name="exersice">
181           <% gymlist.forEach(function(gym) { %>
182             <option value="<%= gym.ex %">
183               <%= gym.ex %>
184             </option>
185             <% }); %>
186           </select>
187
188           <label for="day">Day:</label>
189           <select id="day" name="day">
190             <option value="Monday">Monday</option>
191             <option value="Tuesday">Tuesday</option>
192             <option value="Wednesday">Wednesday</option>
193             <option value="Thursday">Thursday</option>
194             <option value="Friday">Friday</option>
195           </select>
196
197           <label for="time">Time:</label>
198           <select id="time" name="time">
199             <% timeSlots.forEach(slot=> { %>
200               <option value="<%= slot.time %">
201                 <%= slot.time %>
202               </option>
203               <% }) %>
204             </select>
205
206           <button type="submit" class="btn btn-primary btn-block">Create</button>
207         </form>
208       </div>
209     </div>
210   </div>
211 </div>
212
213 </body>
214 </html>
215 </html>
216 </html>
217 </html>
218 </html>
219 <%}%>

```

Ο παραπάνω κώδικας στο κομμάτι του server αφορά τον διαχειριστή του συστήματος και περιλαμβάνει λειτουργίες που σχετίζονται με τη διαχείριση προπονήσεων (trainings) και του προγράμματος προπόνησης (program).

1. Εμφάνιση Προπονήσεων (GET /admin_training):

Αντιστοιχεί σε μια HTTP GET αίτηση στο /admin_training.

Επιστρέφει τη σελίδα admin_training.ejs, εμφανίζοντας τη λίστα των προπονήσεων (gyms).

Περνάει τα δεδομένα του συνδεδεμένου διαχειριστή (email και ρόλος) στη σελίδα.

2. Ενημέρωση Προπονήσεων (POST /update_training):

Χειρίζεται μια HTTP POST αίτηση στο /update_training. Αυτό είναι το action της φόρμας που βρίσκεται μέσα στο admin_training.ejs

Παίρνει τα δεδομένα της φόρμας ενημέρωσης προπόνησης (exercise, trainer, category) με χρήση req.body.

Ελέγχει αν η άσκηση υπάρχει και, αν υπάρχει, ενημερώνει τα στοιχεία της στη βάση δεδομένων.

Αν η άσκηση δεν υπάρχει, επιστρέφει ένα μήνυμα λάθους.

3. Διαγραφή Προπονήσεων (POST /delete_training):

Χειρίζεται μια HTTP POST αίτηση στο /delete_training. Αυτό είναι το action της φόρμας που βρίσκεται μέσα στο **admin_training.ejs**

Παίρνει τα δεδομένα της φόρμας διαγραφής προπόνησης (exercise) με χρήση req.body.

Ελέγχει αν η άσκηση υπάρχει και, αν υπάρχει, τη διαγράφει από τη βάση δεδομένων. Επιπλέον, διαγράφει αντίστοιχες εγγραφές από έναν πίνακα (Gymtable).

Αν η άσκηση δεν υπάρχει, επιστρέφει ένα μήνυμα λάθους.

4. Προσθήκη Νέας Προπόνησης (POST /add_training):

Χειρίζεται μια HTTP POST αίτηση στο /add_training. Αυτό είναι το action της φόρμας που βρίσκεται μέσα στο **admin_training.ejs**

Παίρνει τα δεδομένα της φόρμας προσθήκης προπόνησης (exercise, trainer, category) με χρήση req.body.

Ελέγχει αν η άσκηση υπάρχει και, αν δεν υπάρχει, δημιουργεί μια νέα εγγραφή στη βάση δεδομένων. Αν η άσκηση υπάρχει, επιστρέφει ένα μήνυμα λάθους.

5. Εμφάνιση Προγράμματος (GET /admin_program):

Αντιστοιχεί σε μια HTTP GET αίτηση στο /admin_program.

Επιστρέφει τη σελίδα admin_program.ejs, εμφανίζοντας το πρόγραμμα προπόνησης.

Περνάει τα δεδομένα του συνδεδεμένου διαχειριστή, τα time slots και τη λίστα των προπονήσεων στη σελίδα.

6. Δημιουργία Πίνακα Προγράμματος (POST /create_table):

Χειρίζεται μια HTTP POST αίτηση στο /create_table.

Παίρνει τα δεδομένα της φόρμας δημιουργίας πίνακα προγράμματος (time, day, exercise) με χρήση req.body.

Ελέγχει αν ο πίνακας είναι ήδη καταληφθεί και, αν όχι, δημιουργεί μια νέα εγγραφή στον πίνακα (Gymtable). Αν ο πίνακας είναι καταληφθεί, επιστρέφει ένα μήνυμα λάθους.

7. Διαγραφή Πίνακα Προγράμματος (POST /delete_table):

Χειρίζεται μια HTTP POST αίτηση στο /delete_table.

Παίρνει το ID του πίνακα από τη φόρμα διαγραφής πίνακα προγράμματος με χρήση req.body.

Ελέγχει αν ο πίνακας υπάρχει και, αν όχι, τον διαγράφει από τη βάση δεδομένων. Αν ο πίνακας δεν υπάρχει, επιστρέφει ένα μήνυμα λάθους.

Όλος αυτός ο κώδικας αντιπροσωπεύει τον τρόπο με τον οποίο ο διαχειριστής μπορεί να διαχειριστεί τις προπονήσεις και το πρόγραμμα προπόνησης στο σύστημα.

4. Διαχείριση ανακοινώσεων/ προσφορών

Ο κώδικας στον server μας είναι ο εξής:

```
// ANNOUNCEMENTS
app.get('/admin_announ', (req, res) => {
  console.log(req.session.email);
  Announcement.find().then((announcements)=> {
    res.render('admin_announcements', {
      announcementsList: announcements,
      email: req.session.email,
      role: req.session.role
    })
  })
});

app.post('/publish', (req, res) => {
  const title = req.body.title;
  const announcements = req.body.announcements;
  let newannouncement = new Announcement({
    title: title,
    content: announcements
  });

  newannouncement.save();
  res.redirect("/admin_announ");
});
```


Και ο admin θα βλέπει την σελίδα admin_announcements.ejs η οποία είναι κάπως έτσι:

```
36 <h2 style="text-align: center;">Announcements</h2>
37
38 <table class="table">
39   <thead>
40     <tr>
41       <th scope="col">ID</th>
42       <th scope="col">Title</th>
43       <th scope="col">content</th>
44     </tr>
45   </thead>
46   <tbody>
47     <%announcementsList.forEach(announcement=> {%)>
48       <tr>
49         <th scope="row">
50           <%= announcement._id %>
51         </th>
52         <td>
53           <%= announcement.title %>
54         </td>
55         <td>
56           <%= announcement.content %>
57         </td>
58       </tr>
59     <%})%>
60   </tbody>
61 </table>
62 <h1>Add announcement</h1>
63
64 <form action="/publish" method="post">
65
66   <label for="title">Title</label>
67   <input type="text" class="form-control" id="title" name="title" required>
68
69   <label for="announcements">Announcement:</label>
70   <textarea id="announcements" name="announcements" rows="4" required></textarea>
71
72   <br>
73
74   <button type="submit">submit</button>
75 </form>
76
77 </body>
78
79 </html>
80
81 </html>
82 <%}%>
```

Ο παραπάνω κώδικας , στο κομμάτι του server , αναφέρεται σε λειτουργίες που σχετίζονται με τη διαχείριση ανακοινώσεων από τον διαχειριστή του συστήματος.

Συγκεκριμένα:

1. Εμφάνιση Ανακοινώσεων (GET /admin_announ):

Αντιστοιχεί σε μια HTTP GET αίτηση στο /admin_announ, το οποίο στην ουσία είναι το endpoint , όταν πατάμε το κουμπί δηλαδή της διαχείρισης ανακοινώσεων.

Εκτελεί μια αναζήτηση στη βάση δεδομένων για να ανακτήσει όλες τις ανακοινώσεις.

Επιστρέφει τη σελίδα admin_announcements.ejs παρέχοντας τη λίστα των ανακοινώσεων (announcementsList) προς εμφάνιση στον διαχειριστή.

Περνάει τα δεδομένα του συνδεδεμένου διαχειριστή (email και ρόλος) στη σελίδα.

2. Δημοσίευση Νέας Ανακοίνωσης (POST /publish):

Χειρίζεται μια HTTP POST αίτηση στο /publish. Είναι το action της φόρμας στο **admin_announcements.ejs**

Παίρνει τα δεδομένα της φόρμας δημοσίευσης ανακοίνωσης (τίτλος και περιεχόμενο) με χρήση req.body.

Δημιουργεί μια νέα ανακοίνωση χρησιμοποιώντας το Mongoose Model Announcement.

Αποθηκεύει τη νέα ανακοίνωση στη βάση δεδομένων.

Ανακατευθύνει τον διαχειριστή πίσω στη σελίδα εμφάνισης ανακοινώσεων (/admin_announ) μετά τη δημοσίευση.

Ο κώδικας αυτός επιτρέπει στον διαχειριστή να δει όλες τις υπάρχουσες ανακοινώσεις και να δημοσιεύσει νέες προσθέτοντας τίτλο και περιεχόμενο.

Θα δούμε τώρα την υλοποίηση του συστήματος χρήστη.

Για εξοικονόμηση χώρου , η ακόλουθη φωτογραφία είναι 36 γραμμές σταθερού κώδικα οι οποίες χρησιμοποιούνται σε κάθε .ejs αρχείο που αφορά τον user. Οι λειτουργίες της κάθε σελίδας θα παραθέτονται σε ξεχωριστή φωτογραφία.

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
9     integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
10 </head>
11
12 <body>
13
14   <nav class="navbar navbar-default navbar-static-top">
15     <div class="container-fluid">
16       <div class="navbar-header">
17         <a class="navbar-brand" href="mainpage_style.html">
18           <!--  -->
19         </a>
20       </div>
21       <div class="collapse navbar-collapse">
22         <ul class="nav navbar-nav navbar-right">
23           <li><a href="/">Home</a></li>
24           <li><a href="/announcements">Announcements</a></li>
25           <% if (login != true) { %>
26             <li><a href="/signin">Sign-In</a></li>
27             <li><a href="/signup">Sign-Up</a></li>
28           <% } %>
29           <% if (login == true) { %>
30             <li><a href="/history">History</a></li>
31             <li><a href="/logout">Sign-Out</a></li>
32           <% } %>
33         </ul>
34       </div>
35     </div>
36   </nav>
```

1. Διαδικασία signup του χρήστη

Στο κομμάτι του server ο κώδικας είναι ο εξής:

```
236 app.get('/signup', (req, res) => {
237   res.render('sign-up')
238 })
239
240 app.post('/signup', (req, res) => {
241   let newUser = new User({
242     username: req.body.username,
243     password: req.body.password,
244     email: req.body.email,
245     phone: req.body.phone,
246     lastname: req.body.lastname,
247     name: req.body.name,
248     country: req.body.country,
249     address: req.body.address,
250     cancellations: 0,
251     status: "pending",
252     role: "user"
253   });
254   const email = req.body.email;
255   User.findOne({ email: email }).then((user) => {
256     if (user) {
257       res.send(`<script>alert('Email is already taken');window.location='/signup';</script>`);
258     }
259     else {
260       newUser.save();
261       res.redirect("/signin");
262     }
263   })
264 }
265 })
```

Η σελίδα η οποία βλέπει ο χρήστης είναι το αρχείο sign-up και είναι έτσι:

```

32 <div class="container">
33   <div class="row">
34     <div class="col-md-6 col-md-offset-3">
35       <div class="panel panel-default">
36         <div class="panel-heading">
37           <h3 class="panel-title">Sign Up</h3>
38         </div>
39         <div class="panel-body">
40           <form action="/signup" method="POST">
41             <div class="form-group">
42               <label for="username">Username</label>
43               <input type="text" class="form-control" id="username" name="username" required>
44             </div>
45             <div class="form-group">
46               <label for="email">Email address</label>
47               <input type="email" class="form-control" id="email" name="email" required>
48             </div>
49             <div class="form-group">
50               <label for="lastname">Lastname</label>
51               <input type="text" class="form-control" id="lastname" name="lastname" required>
52             </div>
53
54             <div class="form-group">
55               <label for="name">Name</label>
56               <input type="text" class="form-control" id="name" name="name" required>
57             </div>
58
59             <div class="form-group">
60               <label for="phone">Phone-number</label>
61               <input type="text" class="form-control" id="phone" name="phone" required>
62             </div>
63
64             <div class="form-group">
65               <label for="country">Country</label>
66               <input type="text" class="form-control" id="country" name="country" required>
67             </div>
68
69             <div class="form-group">
70               <label for="address">Address</label>
71               <input type="text" class="form-control" id="address" name="address" required>
72             </div>
73
74             <div class="form-group">
75               <label for="password">Password</label>
76               <input type="password" class="form-control" id="password" name="password" required>
77             </div>
78             <button type="submit" class="btn btn-primary btn-block">Submit</button>
79           </form>
80         </div>
81       </div>
82     </div>
83   </div>
84 </div>
85
86 </body>
87
88 </html>

```

Στην ουσία είναι απλά μία φόρμα εγγραφής με το κουμπί.

Ο παραπάνω κώδικας του server αναφέρεται στη διαχείριση εγγραφής νέων χρηστών στο σύστημα.

Συγκεκριμένα:

1. Δημιουργία νέου χρήστη (POST /signup):

Χειρίζεται μια HTTP POST αίτηση στο /signup, που είναι το action στην φόρμα που υπάρχει στο αρχείο **sign-up.ejs**

Δημιουργεί ένα νέο αντικείμενο χρήστη (newUser) χρησιμοποιώντας τα δεδομένα από τη φόρμα εγγραφής (username, password, email, phone, lastname, name, country, address, cancellations, status, role).

Πραγματοποιεί έλεγχο για το αν η διεύθυνση email που επιλέγει ο νέος χρήστης είναι ήδη καταχωρημένη στο σύστημα.

Αν το email υπάρχει ήδη, επιστρέφει ένα μήνυμα λάθους.

Διαφορετικά, αποθηκεύει το νέο αντικείμενο χρήστη στη βάση δεδομένων.

Ανακατευθύνει τον χρήστη στη σελίδα σύνδεσης (/signin) μετά την επιτυχημένη εγγραφή.

Ο κώδικας αυτός είναι υπεύθυνος για τη δημιουργία και αποθήκευση νέων χρηστών στη βάση δεδομένων, ελέγχοντας παράλληλα την μοναδικότητα της διεύθυνσης email.

2. Διαδικασία sign-in του χρήστη στο σύστημα

Στο κομμάτι του server ο κώδικας είναι ο εξής:

```
268 app.get('/signin', (req, res) => {
269   res.render('sign-in')
270 })
271
272 app.post('/signin', (req, res) => {
273   const email = req.body.email;
274   const password = req.body.password;
275   User.findOne({ email: email, password: password }).then((user) => {
276
277     if (user && user.status == "accepted" && user.role == "user") {
278       req.session.login = true;
279       req.session.role = user.role;
280       req.session.email = email;
281       res.redirect("/");
282     } else if (user && user.role == "admin") {
283       req.session.email = email;
284       req.session.role = user.role;
285       res.redirect("/admin_announ");
286     }
287     else {
288       res.send(`<script>alert('Failed Sign-In');window.location='/signin';</script>`);
289     }
290   })
291 })
292
293
```

Στο κομμάτι του χρήστη , η σελίδα που βλέπει ο ίδιος είναι η sign-in.ejs και είναι έτσι:

```
56 <div class="container">
57   <div class="row">
58     <div class="col-md-6 col-md-offset-3">
59       <div class="panel panel-default">
60         <div class="panel-heading">
61           <h3 class="panel-title">Sign In</h3>
62         </div>
63         <div class="panel-body">
64           <form action="/signin" method="POST">
65             <div class="form-group">
66               <label for="email">Email address</label>
67               <input type="email" class="form-control" id="email" name="email">
68             </div>
69             <div class="form-group">
70               <label for="password">Password</label>
71               <input type="password" class="form-control" id="password" name="password">
72             </div>
73             <button type="submit" class="btn btn-primary btn-block">Submit</button>
74           </form>
75         </div>
76       </div>
77     </div>
78   </div>
79 </div>
80
81 </body>
82
83 </html>
```

Στην ουσία είναι απλά μία φόρμα σύνδεσης με το κουμπί.

Ο παραπάνω κώδικας , του server , αντιστοιχεί σε λειτουργίες που αφορούν τη σελίδα σύνδεσης (sign-in) ενός χρήστη στο σύστημα. Ας αναλύσουμε τον κώδικα:

1. Εμφάνιση Σελίδας Σύνδεσης (GET /signin):

Αντιστοιχεί σε μια HTTP GET αίτηση στο /signin.

Επιστρέφει τη σελίδα sign-in.ejs για να εμφανίσει τη φόρμα σύνδεσης.

Επεξεργασία Σύνδεσης (POST /signin):

Χειρίζεται μια HTTP POST αίτηση στο /signin, το οποίο είναι το action της φόρμας που βρίσκεται στο αρχείο **sign-in.ejs**.

Παίρνει τα δεδομένα της φόρμας σύνδεσης (email και password) με χρήση req.body.

Κάνει αναζήτηση στη βάση δεδομένων για έναν χρήστη με τα συγκεκριμένα email και password.

Ελέγχει τον τύπο του χρήστη (κανονικός χρήστης ή διαχειριστής) και την κατάσταση του λογαριασμού (αποδεκτός ή όχι).

Αν η σύνδεση είναι επιτυχής, αποθηκεύει τις πληροφορίες του χρήστη στο session και κατευθύνει τον χρήστη ανάλογα με τον ρόλο του (σελίδα διαχειριστή για διαχειριστές, αλλιώς στην αρχική σελίδα).

Αν η σύνδεση αποτύχει, επιστρέφει ένα μήνυμα λάθους και καθοδηγεί τον χρήστη πίσω στη σελίδα σύνδεσης.

ΠΡΟΣΟΧΗ! Μόνο εάν ο διαχειριστής έχει κάνει accept το αίτημα του χρήστη για εγγραφή , θα μπορέσει ο χρήστης να κάνει σύνδεση.

3. Περιήγηση στις υπηρεσίες του γυμναστηρίου

Ο κώδικας στον server είναι ο εξής:

```
98 app.get('/', async (req, res) => {
99   const timeSlots = [
100     { time: '9:00', booked: false },
101     { time: '9:30', booked: true },
102     { time: '10:00', booked: false },
103     { time: '10:30', booked: false },
104     { time: '11:00', booked: false },
105   ];
106
107   const gymtable = await Gymtable.find();
108   gymtablelist = gymtable;
109
110   const gyms = await Gym.find();
111   gymlist = gyms;
112
113   // console.log(gymlist);
114
115   // console.log(gymtablelist);
116
117   // console.log(userslist);
118
119   // console.log(currentDayOfWeek);
120
121   res.render('mainpage', { login: req.session.login, timeSlots: timeSlots, gymtablelist: gymtablelist, gymlist: gymlist });
122 })
123 }
```

Ο χρήστης θα βλέπει την αρχική σελίδα του γυμναστηρίου η οποία είναι το αρχείο mainpage.ejs

```
40 <div class="container">
41   <div class="row">
42     <div class="col-md-6">
43       <div class="panel panel-default">
44         <div class="panel-heading">
45           <h3 class="panel-title">Weight Training</h3>
46         </div>
47         <ul class="list-group">
48           <% gymlist.forEach(function(gym) { %>
49             <% if (gym.category === "weight") { %>
50               <li class="list-group-item"><%= gym.ex %> - <%= gym.trainer %></li>
51             <%>
52           }); %>
53         </ul>
54       </div>
55     </div>
56     <div class="col-md-6">
57       <div class="panel panel-default">
58         <div class="panel-heading">
59           <h3 class="panel-title">Cardio</h3>
60         </div>
61         <ul class="list-group">
62           <% gymlist.forEach(function(gym) { %>
63             <% if (gym.category === "cardio") { %>
64               <li class="list-group-item"><%= gym.ex %> - <%= gym.trainer %></li>
65             <%>
66           }); %>
67         </ul>
68       </div>
69     </div>
70   </div>
71 </div>
```

```

72 <% if (login==true) { %>
73 <div class="row">
74 <div class="col-md-12">
75 <div class="panel panel-default">
76 <div class="panel-heading">
77 <h3 class="panel-title">Time Table</h3>
78 </div>
79 <table class="table table-bordered">
80 <thead>
81 <tr>
82 <th>Time</th>
83 <th>Monday</th>
84 <th>Tuesday</th>
85 <th>Wednesday</th>
86 <th>Thursday</th>
87 <th>Friday</th>
88 </tr>
89 </thead>
90 <tbody>
91 <% timeSlots.forEach(slot-> { %>
92 <tr>
93 <td>
94 <% slot.time%>
95 </td>
96 <td>
97 <% gymtablelist.forEach(exercise-> { %>
98 <% if (exercise.day=="Monday" && exercise.time==slot.time) { %>
99 <% if (exercise.user != "") { %>
100 <span style="color: red"><% exercise.ex %></span>
101 <% }else{ %>
102 <% exercise.ex %>
103 <% } %>
104 <% } %>
105 <% }) %>
106 </td>
107 <td>
108 <% gymtablelist.forEach(exercise-> { %>
109 <% if (exercise.day=="Tuesday" && exercise.time==slot.time) { %>
110 <% if (exercise.user != "") { %>
111 <span style="color: red"><% exercise.ex %></span>
112 <% }else{ %>
113 <% exercise.ex %>
114 <% } %>
115 <% } %>
116 <% }) %>
117 </td>
118 <td>
119 <% gymtablelist.forEach(exercise-> { %>
120 <% if (exercise.day=="Wednesday" && exercise.time==slot.time) { %>
121 <% if (exercise.user != "") { %>
122 <span style="color: red"><% exercise.ex %></span>
123 <% }else{ %>
124 <% exercise.ex %>
125 <% } %>
126 <% } %>
127 <% }) %>
128 </td>
129 <td>
130 <% gymtablelist.forEach(exercise-> { %>
131 <% if (exercise.day=="Thursday" && exercise.time==slot.time) { %>
132 <% if (exercise.user != "") { %>
133 <span style="color: red"><% exercise.ex %></span>
134 <% }else{ %>
135 <% exercise.ex %>
136 <% } %>
137 <% } %>
138 <% }) %>
139 </td>
140 <td>
141 <% gymtablelist.forEach(exercise-> { %>
142 <% if (exercise.day=="Friday" && exercise.time==slot.time) { %>
143 <% if (exercise.user != "") { %>
144 <span style="color: red"><% exercise.ex %></span>
145 <% }else{ %>
146 <% exercise.ex %>
147 <% } %>
148 <% } %>
149 <% }) %>
150 </td>
151 </tr>
152 <% } %>
153 </tbody>
154 </table>
155 </div>
156 </div>
157 </div>
158 <% } %>

```

```

162     <% if (login==true) { %>
163
164         <form style="text-align: center;" action="/book" method="post">
165             <label for="service">Service:</label>
166             <select id="service" name="service">
167                 <% gymlist.forEach(function(gym) { %>
168                     <option value="<%= gym.ex %>">
169                         <%= gym.ex %>
170                     </option>
171                 <% }); %>
172             </select>
173
174             <label for="day">Day:</label>
175             <select id="day" name="day">
176                 <option value="Monday">Monday</option>
177                 <option value="Tuesday">Tuesday</option>
178                 <option value="Wednesday">Wednesday</option>
179                 <option value="Thursday">Thursday</option>
180                 <option value="Friday">Friday</option>
181             </select>
182
183
184             <label for="time">Time:</label>
185             <select id="time" name="time">
186                 <% timeSlots.forEach(slot=> { %>
187                     <option value="<%= slot.time %>">
188                         <%= slot.time %>
189                     </option>
190                 <% }) %>
191             </select>
192
193             <button type="submit">Book</button>
194         </form>
195
196     <% } %>
197
198
199 </html>
200
201 </html>
202
203 </html>
204

```

Ο παραπάνω κώδικας, του server, αναφέρεται σε μια λειτουργία που σχετίζεται με την αρχική σελίδα της εφαρμογής.

Συγκεκριμένα:

1. Εμφάνιση Αρχικής Σελίδας (GET /):

Αντιστοιχεί σε μια HTTP GET αίτηση στον ριζικό κατάλογο (/).

Προσδιορίζει μια σταθερή λίστα timeSlots που περιέχει χρονικά διαστήματα και την κατάσταση τους (κρατημένα ή όχι).

Κάνει μια ασύγχρονη αναζήτηση στη βάση δεδομένων για τη λίστα των πινάκων γυμναστηρίου (gymtable).

Κάνει μια ασύγχρονη αναζήτηση στη βάση δεδομένων για τη λίστα των γυμναστηρίων (gyms).

Εκτελεί το render της σελίδας mainpage.ejs, περνώντας τα δεδομένα των χρονικών διαστημάτων (timeSlots), των πινάκων γυμναστηρίου (gymtablelist) και των γυμναστηρίων (gymlist).

Περνάει επίσης το διακριτικό του συνδεδεμένου χρήστη (req.session.login) για να αλλάξει δυναμικά την προεπιλεγμένη κατάσταση σύνδεσης στην αρχική σελίδα.

ΠΡΟΣΟΧΗ! Μόνο εάν ο χρήστης κάνει επιτυχή σύνδεση θα του εμφανίζεται η επιλογή για να κάνει book

4. Κράτηση προϊόντος/ υπηρεσίας

Στον server ο κώδικας είναι ο ακόλουθος:

```
app.post('/book', (req, res) => {  
  
  const time = req.body.time;  
  const day = req.body.day;  
  const ex = req.body.service;  
  const email = req.session.email;  
  const user = "";  
  
  const currentnumday = dayToNumber(currentDayOfWeek);  
  
  Gymtable.findOne({ ex: ex, time: time, day: day }).then((gymtable) => {  
    if (gymtable) {  
      User.findOne({ email: email }).then((user) => {  
  
        inttime = timeToInt(time);  
  
        // && inttime >= getCurrentHour() && currentnumday <= dayToNumber(gymtable.day) && inttime >= getCurrentHour()  
  
        if (gymtable.user === "" && user.cancellations < 2) {  
          // console.log("ok1");  
  
          Gymtable.updateOne({ _id: gymtable._id }, {  
            $set: {  
              user: email  
            }  
          }).then(() => {  
            let newBook = new Book({  
              ex: ex,  
              day: day,  
              time: time,  
              user: email,  
              status: "ongoing"  
            });  
            newBook.save();  
            console.log("Gymtable updated");  
            res.redirect('/');  
          });  
        } else {  
          // console.log("ok2");  
          res.redirect('/');  
          // res.send(<script>alert('That time slot is already booked');window.location='/signup';</script>);  
        }  
      });  
    } else {  
      // console.log("ok3");  
      res.redirect('/');  
      // res.send(<script>alert('That time slot is already booked');window.location='/signup';</script>);  
    }  
  });  
});  
});
```

```

201 app.post('/cancelbooking', (req, res) => {
202
203     const time = req.body.time;
204     const day = req.body.day;
205     const ex = req.body.service;
206     const email = req.session.email;
207
208     Book.findOne({ ex: ex, time: time, day: day }).then((booked) => {
209
210         if(booked){
211             inttimegym = timeToInt(booked.time);
212             inttime = timeToInt(time);
213
214             if(inttimegym <= inttime+2 && booked.status == "ongoing"){
215                 Book.deleteOne({ ex: ex, time: time, day: day }).then(function(){
216                     console.log("Data deleted"); // Success
217                 }).catch(function(error){
218                     console.log(error); // Failure
219                 });
220                 Gymtable.findOne({ ex: ex, time: time, day: day }).then((gymtable) => {
221                     Gymtable.updateOne({ _id: gymtable._id }, {
222                         $set: {
223                             user: ""
224                         }
225                     }).then(() => {
226                         res.redirect('/');
227                     });
228                 });
229             }else{
230                 res.redirect('/');
231             }
232         }else{
233             res.redirect('/');
234         }
235     });
236
237     // res.redirect('/');
238
239 }
240 });
241

```

Ο χρήστης στην ουσία βλέπει το mainpage.ejs που έχει επισυναπτεί παραπάνω. Στην ουσία, εφόσον ο χρήστης είναι συνδεδεμένος θα μπορεί να κάνει κράτηση / ακύρωση.

Ο παραπάνω κώδικας του server , αντιπροσωπεύει λειτουργίες που σχετίζονται με τον χειρισμό κρατήσεων (booking) και ακυρώσεων (cancellation) χρηστών για προπονήσεις σε γυμναστήρια.

1. Κράτηση Προπόνησης (POST /book):

Χειρίζεται μια HTTP POST αίτηση στο /book.

Παίρνει τα δεδομένα της φόρμας κράτησης προπόνησης (time, day, service) με χρήση req.body.

Ελέγχει τη διαθεσιμότητα του χρονοδιαγράμματος προπονήσεων (Gymtable) και τη διαθεσιμότητα του χρήστη.

Αν ο πίνακας προγράμματος είναι διαθέσιμος και ο χρήστης δεν έχει ξεπεράσει τον αριθμό των επιτρεπόμενων ακυρώσεων, πραγματοποιεί την κράτηση ενημερώνοντας τους πίνακες Gymtable και Book.

Αν υπάρχει σύγκρουση (π.χ., ο χρήστης ήδη έχει κράτηση σε αυτό το χρονικό διάστημα), απορρίπτει την κράτηση.

2. Ακύρωση Κράτησης (POST /cancelbooking):

Χειρίζεται μια HTTP POST αίτηση στο /cancelbooking.

Παίρνει τα δεδομένα της φόρμας ακύρωσης κράτησης (time, day, service) με χρήση req.body.

Ελέγχει αν η κράτηση υπάρχει και αν πληροί τις συνθήκες για ακύρωση (π.χ., έχει περάσει λιγότερο από ένα διάστημα από την κράτηση και η κατάσταση είναι "ongoing").

Αν όλα είναι εντάξει, διαγράφει την κράτηση από τον πίνακα Book και ενημερώνει τον πίνακα Gymtable.

Αν υπάρχουν προβλήματα ή οι συνθήκες δεν πληρούνται, απλά ανακατευθύνει τον χρήστη στην αρχική σελίδα.

Συνολικά, ο κώδικας αυτός ελέγχει και διαχειρίζεται τις κρατήσεις και ακυρώσεις προπονήσεων ανάλογα με τις συνθήκες και τη διαθεσιμότητα του προγράμματος προπονήσεων.

5. Ιστορικό κρατήσεων

Ο κώδικας στον server είναι ο εξής:

```
182 app.get('/history', async (req, res) => {  
183  
184   const timeSlots = [  
185     { time: '9:00', booked: false },  
186     { time: '9:30', booked: true },  
187     { time: '10:00', booked: false },  
188     { time: '10:30', booked: false },  
189     { time: '11:00', booked: false },  
190   ];  
191  
192   const gyms = await Gym.find();  
193   gymlist = gyms;  
194  
195   const books = await Book.find();  
196   bookslist = books;  
197  
198   res.render('history', { bookslist: bookslist, login: req.session.login, email: req.session.email, timeSlots: timeSlots, gymlist: gymlist });  
199 });
```

Ο χρήστης , βλέπει τη σελίδα **history.ejs** η οποία είναι η εξής:


```

40 <table class="table">
41   <thead>
42     <tr>
43       <th scope="col">Exercise</th>
44       <th scope="col">Day</th>
45       <th scope="col">Time</th>
46       <th scope="col">Status</th>
47     </tr>
48   </thead>
49   <tbody>
50     <%bookslist.forEach(Book=> {%>
51       <%if (Book.user === email){%>
52         <tr>
53           <th scope="row">
54             <%= Book.ex %>
55           </th>
56           <td>
57             <%= Book.day %>
58           </td>
59           <td>
60             <%= Book.time %>
61           </td>
62           <td>
63             <%= Book.status %>
64           </td>
65         </tr>
66       <%}%>
67     <%}%>
68   </tbody>
69 </table>
70 </body>
71
72 <form style="text-align: center;" action="/cancelbooking" method="post">
73   <label for="service">Service:</label>
74   <select id="service" name="service">
75     <% gymlist.forEach(function(gym) { %>
76       <option value="<%= gym.ex %>">
77         <%= gym.ex %>
78     </option>
79     <% }); %>
80   </select>
81
82   <label for="day">Day:</label>
83   <select id="day" name="day">
84     <option value="Monday">Monday</option>
85     <option value="Tuesday">Tuesday</option>
86     <option value="Wednesday">Wednesday</option>
87     <option value="Thursday">Thursday</option>
88     <option value="Friday">Friday</option>
89   </select>
90
91   <label for="time">Time:</label>
92   <select id="time" name="time">
93     <% timeSlots.forEach(slot=> { %>
94       <option value="<%= slot.time %>">
95         <%= slot.time %>
96       </option>
97     <% }) %>
98   </select>
99
100   <button type="submit">Cancel</button>
101 </form>
102
103 </html>
104
105 </html>
106

```

Ο παραπάνω κώδικας του server, αντιστοιχεί σε έναν Express route handler που αντιστοιχεί στο endpoint /history για μια HTTP GET αίτηση. Ας κάνουμε μία συνοπτική ανάλυση:

Επισκόπηση Route:

Τύπος Αίτησης: HTTP GET στο /history.

Λειτουργία: Ανάκτηση ιστορικού (history) σχετικά με κρατήσεις.

Στατική Λίστα Ώρων:

Ορίζεται μια στατική λίστα ωρών (timeSlots) με στοιχεία που αναφέρουν τον χρόνο και το αν έχει γίνει κράτηση σε αυτόν τον χρόνο.

Αναζήτηση Αιθουσών (Gyms):

Χρησιμοποιεί το Mongoose Model Gym για να ανακτήσει όλες τις αίθουσες (gyms) από τη βάση δεδομένων.

Οι αίθουσες αποθηκεύονται στη μεταβλητή gymlist.

Αναζήτηση Κρατήσεων (Books):

Χρησιμοποιεί το Mongoose Model Book για να ανακτήσει όλες τις κρατήσεις (books) από τη βάση δεδομένων.

Οι κρατήσεις αποθηκεύονται στη μεταβλητή booklist.

Απόκριση με Σελίδα Ιστορικού:

Επιστρέφει τη σελίδα history.ejs, περνώντας τη λίστα των κρατήσεων (bookslis), τον χρήστη που είναι συνδεδεμένος (req.session.login), το email του χρήστη (req.session.email), τη στατική λίστα ωρών (timeSlots) και τη λίστα των αιθουσών (gymlist).

6. Νέα/ανακοινώσεις

Ο κώδικας του server είναι ο εξής:

```
86 app.get('/announcements', (req, res) => {
87   Announcement.find().then((announcements)=> {
88     res.render('announcements', {
89       announcementsList: announcements,
90       login: req.session.login
91     })
92   })
93 })
94
```

Ο χρήστης βλέπει την σελίδα announcements.ejs :

```
39
40 <table class="table">
41   <thead>
42     <tr>
43       <th scope="col">ID</th>
44       <th scope="col">Title</th>
45       <th scope="col">content</th>
46     </tr>
47   </thead>
48   <tbody>
49     <%announcementsList.forEach(announcement=> {%>
50       <tr>
51         <th scope="row">
52           <%= announcement._id %>
53         </th>
54         <td>
55           <%= announcement.title %>
56         </td>
57         <td>
58           <%= announcement.content %>
59         </td>
60       </tr>
61     <%%>%>
62   </tbody>
63 </table>
64 </body>
65
66 </html>
67
68 </html>
```

Ο παραπάνω κώδικας αντιστοιχεί σε έναν Express route handler που αντιμετωπίζει HTTP GET αιτήσεις προς το /announcements. Η λειτουργία του κώδικα είναι η εξής:

Εμφάνιση Ανακοινώσεων (GET /announcements):

Αντιστοιχεί σε μια HTTP GET αίτηση στο /announcements.

Εκτελεί μια αναζήτηση στη βάση δεδομένων χρησιμοποιώντας το Mongoose Model Announcement για να ανακτήσει όλες τις ανακοινώσεις.

Αφού λάβει τα δεδομένα, αποστέλλει μια απάντηση στον πελάτη, χρησιμοποιώντας το res.render, για να εμφανίσει τη σελίδα announcements.ejs.

Περνάει τη λίστα των ανακοινώσεων (announcementsList) και την κατάσταση σύνδεσης (login) στο template engine που χρησιμοποιείται για τη σελίδα.

Συνοπτικά, αυτός ο κώδικας εξυπηρετεί τον σκοπό να εμφανίσει στους χρήστες τις υπάρχουσες ανακοινώσεις του συστήματος όταν προσπαθούν να αποκτήσουν πρόσβαση στη σελίδα /announcements.

Κεφάλαιο 4: Εγχειρίδιο διαχειριστή

Βήμα 1

Κατεβάζουμε το nodeJS στον Η/Υ μας (<https://nodejs.org/en/download>)

Βήμα 2

Σιγουρευόμαστε ότι το PATH για το nodeJS έχει μπει στις μεταβλητές συστήματος

Βήμα 3

Δημιουργούμε τον φάκελο που θα βάλουμε μέσα τα αρχεία μας και ανοίγουμε το CMD .

Στο CMD μεταφερόμαστε στον φάκελο που έχουμε τα αρχεία της εργασίας μας. Δηλαδή κάνουμε:

```
cd ptath_του_φακελου_που_δημιουργησαμε
```

και έπειτα γράφουμε

```
npm init
```

για να αρχικοποιήσουμε το nodeJS project.

Βήμα 4

Βάζουμε στο φάκελο που φτιάξαμε τον φάκελο του project και θα εγκαταστήσουμε τα απαραίτητα packages για να λειτουργήσει ο κώδικας.

```
npm -i express mongoose ejs pody-parser
```

Βήμα 5

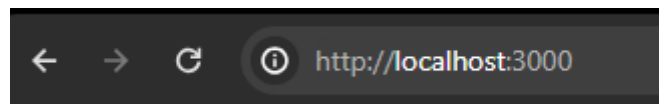
Τέλος κατεβάζουμε το MongoDB και το MongoDBCompass και σιγουρευόμαστε ότι το connection είναι mongodb://localhost:27017

Αυτή ήταν η αρχική εγκατάσταση.

Έπειτα , κάθε φορά για να τρέχει ο server , θα χρειάζεται μέσω CMD να κάνουμε cd στον φάκελο με τα αρχεία μας.

Αφού μεταβούμε εκεί , θα γράφουμε npm start και ο server θα τρέχει.

Στην συνέχεια , ανοίγουμε έναν browser και μεταβαίνουμε στην διεύθυνση:



Εκεί θα τρέχει κανονικά η σελίδα μας.

Κεφάλαιο 5: Εγχειρίδιο χρήστη

Ακολουθεί λεπτομερές εγχειρίδιο χρήσης με περιγραφή όλων των λειτουργιών που αφορούν το διαχειριστή και τον απλό χρήστη του συστήματος.

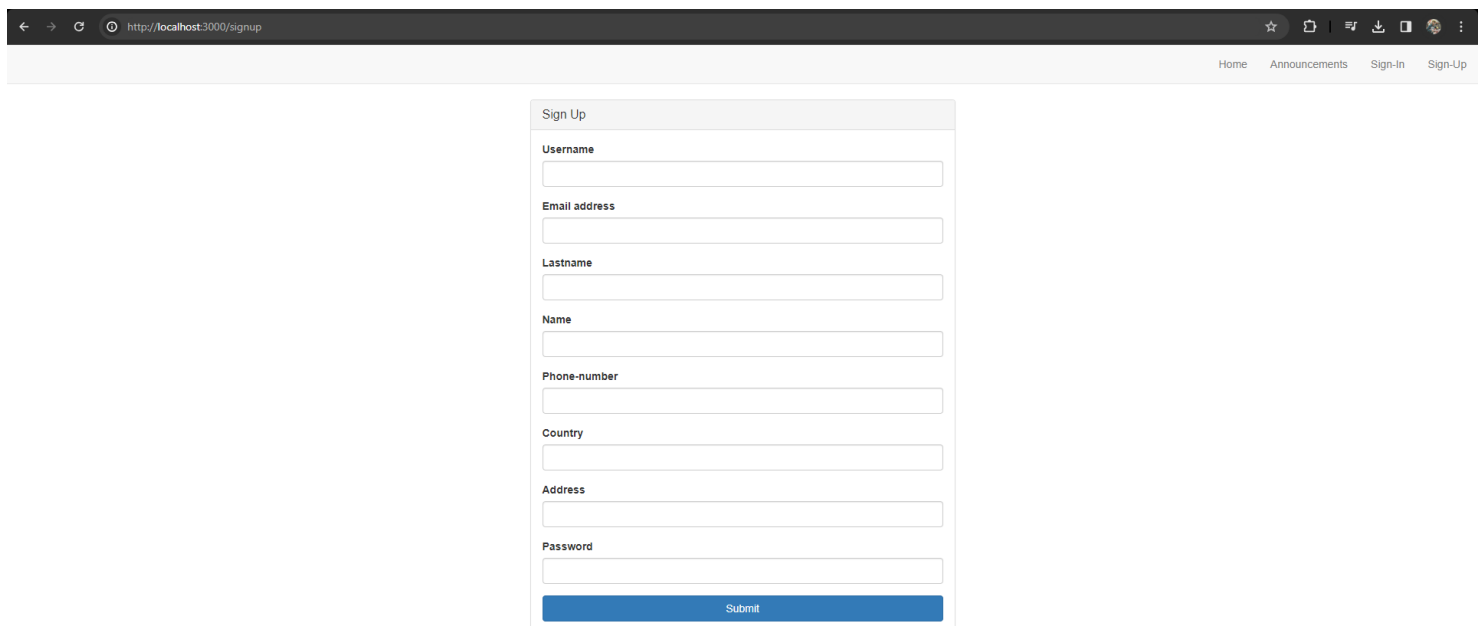
Ας δούμε αρχικά τις λειτουργίες ενός απλού User.

Ο χρήστης πάνω δεξιά έχει το μενού πλοήγησης. Αυτό αποτελείται από τις επιλογές Home , Announcements , Sign-in , Sign-up.

Εάν ο χρήστης κάνει Sign-in χωρίς να έχει λογαριασμό , θα εμφανιστεί μήνυμα σφάλματος.

Επομένως , πάμε να δούμε την λειτουργία του sign-up.

1. Λειτουργία Sign Up



The screenshot shows a web browser window with the address bar displaying 'http://localhost:3000/signup'. The browser's navigation bar includes links for 'Home', 'Announcements', 'Sign-In', and 'Sign-Up'. The main content area features a 'Sign Up' form with the following fields: Username, Email address, Lastname, Name, Phone-number, Country, Address, and Password. A blue 'Submit' button is located at the bottom of the form.

Ο χρήστης βλέπει την παραπάνω φόρμα. Η φόρμα αυτή είναι η φόρμα εγγραφής.

Ας την συμπληρώσουμε:

Sign Up

Username
antonhspap

Email address
antonhspap@icloud.com

Lastname
Papakonstantinou

Name
Antonis Papakonstantinou

Phone-number
6986680496

Country
Ελλάδα

Address
ΧΡΥΣΟΣΤΟΜΟΥ ΣΜΥΡΝΗΣ 17Α

Password
...

Submit

Επομένως έχουμε Password: 123 και email: antonhspap@icloud.com

Όπως βλέπουμε στην ακόλουθη εικόνα , παρόλο που ο χρήστης έκανε επιτυχής εγγραφή , όταν συνδέεται με τα σωστά στοιχεία , του εμφανίζει μήνυμα λάθους. **Αυτό συμβαίνει καθώς ο διαχειριστής δεν έχει κάνει δεκτό το αίτημα εγγραφής του.**

Ο ιστότοπος localhost:3000 λέει

Failed Sign-In



Για την διευκόλυνση της παρουσίασης , θα κάνω accept το αίτημα του user και θα συνεχίσουμε. Την διαδικασία accept/decline από την μεριά του διαχειριστή θα την παρουσιάσουμε αργότερα.

Συνεχίζουμε , αφού έχει γίνει accept το αίτημα του χρήστη από τον διαχειριστή , στην διαδικασία του sign-in.

2. Λειτουργία Sign-In

Sign In

Email address
antonhsap@icloud.com

Password

Submit

Αφού πατήσουμε το κουμπί και τα στοιχεία είναι τα σωστά , μας πετάει στο main-page της ιστοσελίδας μας.

Weight Training

Cardio

treximo - nikos

Time Table

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00					treximo
9:30					
10:00					
10:30					
11:00					

Service: Day: Time:

3. Booking κάποιας άσκησης

Στο main page ο χρήστης βλέπει το πρόγραμμα του γυμναστηρίου και μπορεί να κλείσει εάν επιθυμεί κάποιο πρόγραμμα.

Για παράδειγμα , αυτή τη στιγμή υπάρχει η προπόνηση treximo την Παρασκευή στις 09.00 το πρωί.

Θα πάμε κάτω στην μπάρα και αφού βάλουμε την σωστή ημέρα και ώρα θα πατήσουμε book.

Service: Day: Time:

Φυσικά , εάν μπούνε λάθος στοιχεία στο booking bar θα εμφανιστεί μήνυμα λάθους.

Αμέσως μόλις το πατήσουμε το χρώμα της άσκησης θα γίνει κόκκινο , επειδή το έχουμε κλείσει:

Weight Training

Cardio

treximo - nikos

Time Table

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00					treximo
9:30					
10:00					
10:30					
11:00					

Service: Day: Time:

4. Εμφάνιση ανακοινώσεων

Announcements		
ID	Title	content
65b19f8dfd92c683cf6cb388	Ανακοίνωση σχετικά με την διεξαγωγή προπονήσεων	Η προπόνηση Δευτέρας Τρίτης , Τετάρτης και Πέμπτης , ακυρώνεται λόγω ίωσης των γυμναστών. Ξανά μαζί σας από Παρασκευή!

Μόλις ο χρήστης πατήσει στο κουμπί Announcements θα δει τις ανακοινώσεις που υπάρχουν , εάν υπάρχουν. Στην περίπτωση μας όπως θα δείτε παραπάνω , υπάρχει 1 ανακοίνωση.

5. Εμφάνιση ιστορικού κρατήσεων

Πατώντας ο χρήστης στο κουμπί History έχει την δυνατότητα να δει το ιστορικό των κρατήσεων του όπως φαίνεται παρακάτω. Βλέπουμε κανονικά την προπόνηση της παρασκευής που κλείσαμε.

Booking History			
Exercise	Day	Time	Status
treximo	Friday	9:00	ongoing

Service: Day: Time:

6. Ακύρωση προπόνησης

Στο κουμπί history από κάτω υπάρχει το search bar για την ακύρωση προπόνησης.

Service: Day: Time:

Βάζοντας την άσκηση που έχουμε κάνει book , την ημέρα και την ώρα , πατώντας το κουμπί cancel ακυρώνουμε την προπόνηση. Ισχύει η προϋπόθεση με την έως πριν 2 ώρες που αναφέρεται ρητά στην εκφώνηση.

[Home](#) [Announcements](#) [History](#) [Sign-Out](#)

Weight Training

Cardio
treximo - nikos

Time Table	Monday	Tuesday	Wednesday	Thursday	Friday
9:00					treximo
9:30					
10:00					
10:30					
11:00					

Service: Day: Time:

[Home](#) [Announcements](#) [History](#) [Sign-Out](#)

Booking History

Exercise	Day	Time	Status
Service: <input type="text" value="treximo"/> Day: <input type="text" value="Monday"/> Time: <input type="text" value="9:00"/> <input type="button" value="Cancel"/>			

Τέλος , όπως βλέπουμε , εφόσον μία προπόνηση ακυρωθεί , στο πρόγραμμα επιστρέφει σε γκρι γραμματοσειρά και βγαίνει από το ιστορικό κρατήσεων του χρήστη.

7. Sign – out

Ο χρήστης , τέλος , έχει την δυνατότητα να κάνει sign out από το σύστημα από το κουμπί πάνω τέρμα δεξιά .

Ας δούμε τώρα τις λειτουργίες ενός Admin.

Sign In

Email address

admin@1

Password

...

Submit

Στην σελίδα του sign in έχουμε περασμένο στη βάση by default έναν admin με **email: admin@1** και **password: 123**

1. Publish announcements / Δημοσίευση των ανακοινώσεων

Με το που γίνει η σύνδεση του admin , του εμφανίζει το home page , που το έχουμε κάνει να είναι η σελίδα Publish Announcements.

Publish Announcements Register Requests Users Managment Admin Training Managment Program Sign-Out		
Announcements		
ID	Title	content
65b19f8dfd92c683cf6cb388	Ανακοίνωση σχετικά με την διεξαγωγή προπονήσεων	Η πρόπονηση Δευτέρας Τρίτης , Τετάρτης και Πέμπτης , ακυρώνεται λόγω ίωσης των γυμνασίων. Ξανά μαζί σας από Παρασκευή!

Add announcement

Title

Announcement:

submit

Εδώ ο admin , εφόσον συμπληρώσει το θέμα της ανακοίνωσης και το περιεχόμενο της , το δημοσιεύει και φαίνεται κανονικά στις ανακοινώσεις του γυμναστηρίου όπως είδαμε παραπάνω.

Ο admin έχει επιλογές πάνω δεξιά στο μενού το οποίο είναι κάπως έτσι:

Publish Announcements	Register Requests	Users Managment	Admin Training	Managment Program	Sign-Out
---------------------------------------	-----------------------------------	---------------------------------	--------------------------------	-----------------------------------	--------------------------

2. Register Requests

Στην καρτέλα αυτή , ο admin , έχει την δυνατότητα να διαχειρίζεται τα αιτήματα των χρηστών. Δηλαδή , κάθε χρήστης που κάνει εγγραφή , πριν συνδεθεί θα πρέπει να αποδειχτεί το αίτημα εγγραφής του ο admin.

Register Requests					
Email	Name	Phone	Status		
user@1	antonis	69	rejected	Decline	Accept
antonhspap@icloud.com	Antonis Papakonstantinou	6986680496	accepted	Decline	Accept

Έτσι , για κάθε χρήστη ο admin έχει την επιλογή για να κάνει accept ή decline το αίτημα του.

ΠΡΟΣΟΧΗ! Μόνο οι χρήστες με status = accepted μπορούν να κάνουν επιτυχημένο login.

3. Users Management

Users Managment									
Email	Name	Phone	Address	Country	Username	Lastname	Cancellations	Role	Status
admin@1							0	admin	Delete
user@1	antonis	69	d	gr	antonhap	pap	0	user	rejected Delete
antonhap@icloud.com	Antonis Papakostantinou	695660496	ΧΡΥΣΟΘΤΟΜΟΥ ΙΜΑΨΙΗΣ 17Α	Ελλάδα	antonhap	Papakostantinou	0	user	accepted Delete

Update User

Username

Email address

Lastname

Name

Phone-number

Country

Address

Cancellations

Role

Update

Σε αυτή την ενότητα , ο admin μπορεί να διαχειρίζεται και να αλλάζει τα στοιχεία των χρηστών που είναι εγγεγραμμένοι στο σύστημα μας.

Φυσικά μπορεί και να διαγράψει χρήστες.

Έστω ότι έχουμε τους ακόλουθους χρήστες:

Users Management									
Email	Name	Phone	Address	Country	Username	Lastname	Cancellations	Role	Status
admin@1							0	admin	<button>Delete</button>
user@1	antonis	69	d	gr	antonhsap	pap	0	user	rejected <button>Delete</button>
antonhsap@icloud.com	Antonis Papakonstantinou	6966660496	XPYZOETOMOY IMYPNHE 17A	Ελλάδα	antonhsap	Papakonstantinou	0	user	accepted <button>Delete</button>

Και θέλουμε να αλλάξουμε τον χρήστη με email antonhspap@icloud.com:

Users Managment

Email	Name	Phone	Address	Country	Username	Lastname	Cancellations	Role	Status	
admin@1							0	admin		Delete
user@1	antonis	69	d	gr	antonhspap	pap	0	user	rejected	Delete
antonhspap@icloud.com	Antonis Papakonstantinou	69	address2	CountryChange	antonhspapCHANGE	Papakonstantinou	0	user	accepted	Delete

Αφού συμπλήρωσα την φόρμα μου , και πάτησα το κουμπί UPDATE
όπως βλέπετε στην παραπάνω φωτογραφία η αλλαγή στον τελευταίο
χρήστη έχει πραγματοποιηθεί!

4. Admin Training

[Publish Announcements](#) [Register Requests](#) [Users Managment](#) [Admin Training](#) [Managment Program](#) [Sign-Out](#)

Admin Training

ID	Exercise	Trainer	Category	
65b19e4afd92c683cf6cb36d	treximo	nikos	cardio	Delete

New Training

Exercise

Trainer name

Category

Weight ▾

Create

Update Training

Exersice

Trainer name

Category

Cardio ▾

Update

Ενεργοποιήστε τα Windows
Μετάβαση στις ρυθμίσεις για ενεργοποίηση των Windows.

Στην σελίδα αυτή , ο admin , μπορεί να προσθέτει , να διαγράφει , να βλέπει και να κάνει update κάποια προπόνηση.

Για παράδειγμα , πάμε να βάλουμε μία ακόμη προπόνηση:

[Publish Announcements](#) [Register Requests](#) [Users Managment](#) [Admin Training](#) [Managment Program](#) [Sign-Out](#)

Admin Training

ID	Exercise	Trainer	Category	
65b19e4afd92c683cf6cb36d	treximo	nikos	cardio	Delete

New Training

Exercise

pagkos 200kg

Trainer name

antonis

Category

Weight ▾

Create

Μόλις πατήσουμε το κουμπί **create** παρατηρούμε πως η προπόνηση μας έχει προστεθεί στον πίνακα των προπονήσεων. Φυσικά , κάθε αλλαγή που κάνουμε εδώ είναι ορατή και στον user.

Publish Announcements Register Requests Users Managment Admin Training Managment Program Sign-Out				
Admin Training				
ID	Exercise	Trainer	Category	
65b19e4afd92c683cf6cb36d	treximo	nikos	cardio	Delete
65b22a7a97ce01f809917c94	pagkos 200kg	antonis	weight	Delete

ΠΡΟΣΟΧΗ! Ίσως χρειαστεί μία μικρή ανανέωση (refresh) για να φανεί και να ανανεωθεί ο πίνακας μετά την δημιουργία προπόνησης.

Προχωράμε στο update κάποια προπόνησης.

Publish Announcements Register Requests Users Managment Admin Training Managment Program Sign-Out				
Admin Training				
ID	Exercise	Trainer	Category	
65b19e4afd92c683cf6cb36d	treximo	nikos	cardio	Delete
65b22a7a97ce01f809917c94	pagkos 200kg	antonis	weight	Delete

Έστω πως στην άσκηση treximo θέλουμε ως trainer τον George:

Admin Training				
ID	Exercise	Trainer	Category	
65b19e4afd92c683cf6cb36d	treximo	nikos	cardio	Delete
65b22a7a97ce01f809917c94	pagkos 200kg	antonis	weight	Delete

New Training

Exercise

Trainer name

Category

Cardio

Create

Update Training

Exersice

treximo

Trainer name

george

Category

Cardio

Update

Ενεργοποιήστε τα Windows
Μετάβαση στις ρυθμίσεις για ενεργοποίηση των Windows.

Μόλις πατήσουμε το κουμπί update αφού έχουμε συμπληρώσει τα πεδία της φόρμας , θα δούμε πως η αλλαγή θα γίνει επιτυχώς.

Admin Training

ID	Exercise	Trainer	Category	
65b19e4afd92c683cf6cb36d	treximo	george	cardio	<button>Delete</button>
65b22a7a97ce01f809917c94	pagkos 200kg	antonis	weight	<button>Delete</button>

Φυσικά , δίπλα από κάθε προπόνηση, υπάρχει το κουμπί της διαγραφής. Πατώντας το μπορούμε να κάνουμε διαγραφή σε όποια άσκηση δεν επιθυμούμε.

5. Management program

[Publish Announcements](#) [Register Requests](#) [Users Management](#) [Admin Training](#) [Managment Program](#) [Sign-Out](#)

Managment Program

ID	Exersice	Time	Day	User	
65b19e6afd92c683cf6cb377	treximo	9:00	Friday		<button>Delete</button>

Time Table					
Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00					treximo
9:30					
10:00					
10:30					
11:00					

Update Training

Service: treximo

Day: Monday

Time: 9:00

Create

Στο ακόλουθο παράθυρο , ο admin , μπορεί να διαχειρίζεται το πρόγραμμα. Για την ακρίβεια μπορεί να προσθέτει στο πρόγραμμα ΜΟΝΟ ασκήσεις τις οποίες τις έχει δημιουργήσει προηγουμένως στο admin training.

Στην παραπάνω φωτογραφία , παρατηρούμε πως υπάρχει μία φόρμα με 3 dropdown box. Στο πρώτο έχουμε τις ασκήσεις και το περιεχόμενο του

είναι οι ασκήσεις που δημιούργησε προηγουμένως ο admin στο admin training.

Πάμε τώρα να προσθέσουμε την άσκηση στο πρόγραμμα:

[Publish Announcements](#) [Register Requests](#) [Users Management](#) [Admin Training](#) [Managment Program](#) [Sign-Out](#)

Managment Program

ID	Exersice	Time	Day	User
6b19e6afd92c683cf6cb377	treximo	9:00	Friday	<button>Delete</button>

Time Table

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00					treximo
9:30					
10:00					
10:30					
11:00					

Update Training

Service: pagkos 200kg Day: Wednesday Time: 9:00

Create

Μόλις επιλέξουμε την άσκηση που θέλουμε , και επιλέξουμε ώρα και ημέρα , η άσκηση θα προστεθεί ΜΟΝΟ εάν εκείνη την ημέρα και ώρα δεν υπάρχει άλλη άσκηση. Εάν υπάρχει θα εμφανίζεται μήνυμα λάθους.

[Publish Announcements](#) [Register Requests](#) [Users Management](#) [Admin Training](#) [Managment Program](#) [Sign-Out](#)

Managment Program

ID	Exersice	Time	Day	User
65b19e6afd92c683cf6cb377	treximo	9:00	Friday	<button>Delete</button>
65b23c5e97ce01f809817caa	pagkos 200kg	9:00	Wednesday	<button>Delete</button>

Time Table

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00			pagkos 200kg		treximo
9:30					
10:00					
10:30					
11:00					

Update Training

Service: treximo Day: Monday Time: 9:00

Create

Έτσι βάλαμε την άσκηση ragkos 200kg την Τετάρτη στις 09.00 το πρωί.
Φυσικά για κάθε άσκηση στο πρόγραμμα υπάρχει και η διαγραφή.

Αυτή ήταν η παρουσίαση της εργασίας μας και της προσπάθειας μας.

Ευχαριστούμε για τον χρόνο σας!