

Proiect
Ingineria Sistemelor de Programare

Alexandru Cristian-Gabriel
Automatica si Informatica Aplicata
Anul IV

Cuprins

- Introducere
- Tehnologii si procese
- Cod sursa
- Demonstratie functionalitati

Introducere

Pentru acest proiect am ales ca tema creearea unei aplicatii WEB ce ajuta la managementul activitatii intr-un service auto. Pagina principala contine o lista cu activitatatile inregistrate si butoane ce permit modificarea unei intrari in tabel, stergerea ei si cel mai important un buton pentru adaugarea activitatilor noi.

Functionalitatiile aplicatiei:

- Adaugarea activitatii in tabel
- Modificarea detaliilor unei activitati
- Stergerea unei activitati

Tehnologii

Pentru creearea acestei aplicatii s-au folosit urmatoarele tehnologii:

- Front End
 - a. Bootstrap - este un framework CSS gratuit si open-source, directionat catre dezvoltarea web front-end receptiva si mobila. Contine sabloane de design bazate pe CSS si JavaScript pentru tipografie, formulare, butoane, navigare si alte componente de interfata.
 - b. Font Awesome Icons – este un repositoriu ce contine fonturi si iconite construit peste CSS si Less.
 - c. Handlebars – este un motor de sabloane ce permite transformarea transformarea sabloanelor in pagini HTML.
- Database
 - a. MongoDB - este o baza de date NoSQL open-source orientata pe documente. Aceasta baza de date beneficiaza de suport din partea companiei 10gen. MongoDB face parte din familia de sistemelor de baze de date NoSQL.

- b. Mongoose – este un framework ce permite simplificarea interactiunii cu baza de date MongoDB si a interogarilor facute.
- Backend
 - a. NodeJS – Node.js este un mediu de executie JavaScript de tip open-source, cross-platform, back-end care executa cod JavaScript in afara unui browser web.
 - b. Express JS - este un cadru de aplicatii web back-end pentru Node.js, lansat ca software gratuit si open-source sub licenta MIT. Este conceput pentru a construi aplicatii web si API-uri. A fost numit de facto standard de server pentru Node.js.

Cod sursa

Initierea proiectului prin creearea unui fisier “package.json” ce declara metadata despre aplicatie si modulele folosite. Acest fisier descarca si valideaza modulele, oferind control asupra dependentelor si versiunile disponibile.



```
1  {
2    "name": "proiect-isps",
3    "version": "1.0.0",
4    "description": "Proiect pentru cursul ISP - Automatica Anul 4, Semestrul 1",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"
8    },
9    "author": "Alexandru Gabriel",
10   "license": "ISC",
11   "dependencies": {
12     "body-parser": "^1.19.0",
13     "express": "^4.17.1",
14     "express-handlebars": "^5.2.0",
15     "mongoose": "^5.11.11",
16     "nodemon": "^2.0.7"
17   }
18 }
```

Prin fisierul “repairmodel.js” se creeaza un sablon ce declara proprietatiile obiectelor adaugate in baza de date si schema folosita.



```
1 //Importare Mongoose
2 const mongoose = require("mongoose");
3
4 //Declarare proprietati obiect MongoDB
5 var repairSchema = new mongoose.Schema({
6   clientName: {
7     type: String,
8   },
9   carModel: {
10     type: String,
11   },
12   plateNumber: {
13     type: String,
14   },
15   mechanicName: {
16     type: String,
17   },
18   workedHours: {
19     type: String,
20   },
21 });
22
23 //Initiere DB
24 mongoose.model("Repair", repairSchema);
```

Exemplu obiect din baza de date:

```
_id: ObjectId("5ffeb2e44b87ec2a381346e4")
clientName: "Gabriel Alexandru"
carModel: "Renault Clio 4"
plateNumber: "B105FNJ"
mechanicName: "Vali Nas Mare"
workedHours: "2"
__v: 0
```

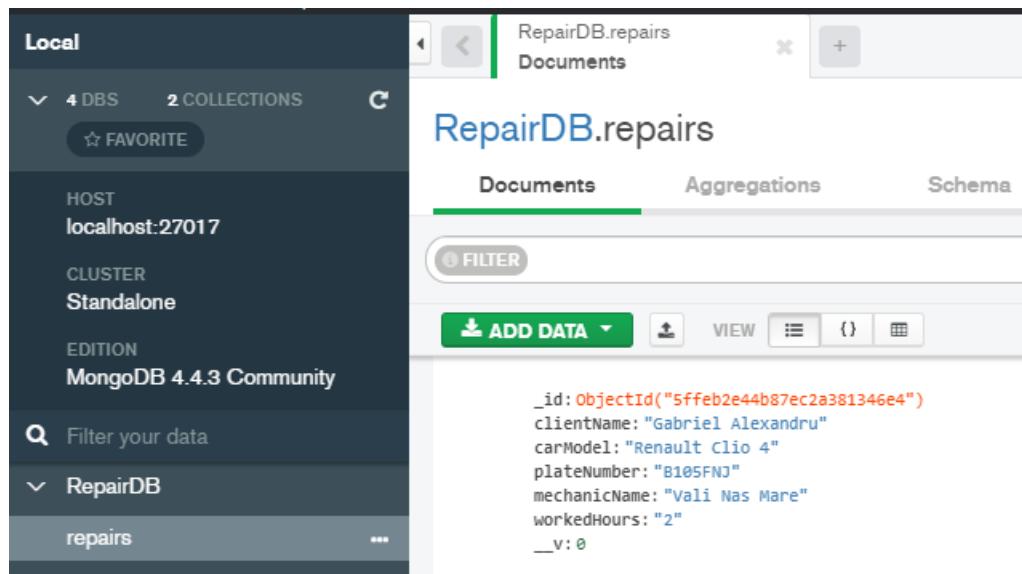
Prin fisierul "db.js" se creeaza conexiunea cu baza de date si se declara sаблонul folosit pentru atribuirea proprietatilor pentru obiectele ce urmeaza sa fie adaugate in baza de date.



```

1 //Importare Mongoose
2 const mongoose = require("mongoose");
3
4 //Conectare la DB
5 mongoose.connect(
6   "mongodb://localhost:27017/RepairDB",
7   { useNewUrlParser: true },
8   (err) => {
9     if (!err) {
10       console.log("Connection with the database successful.");
11     } else {
12       console.log("Error in connection: " + err);
13     }
14   }
15 );
16
17 //Creare DB prin template-ul din fisierul "repairmodel.js"
18 require("./repairmodel");

```



The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays the local host (localhost:27017) with 4 databases and 2 collections. One collection is named "repairs". The main window shows the "RepairDB" database with its "repairs" collection. The "Documents" tab is selected, showing a single document with the following fields:

- `_id: ObjectId("5ffeb2e44b87ec2a381346e4")`
- `clientName: "Gabriel Alexandru"`
- `carModel: "Renault Clio 4"`
- `plateNumber: "B105FNJ"`
- `mechanicName: "Vali Nas Mare"`
- `workedHours: "2"`
- `__v: 0`

Fisierul “server.js” este fisierul ce declara toate proprietatile pentru aplicatia Express precum importarea modulelor necesare si declararea motorului de sabloane.

```
1 //Acces la fisierul "db.js" ce initieaza conexiunea cu baza de date
2 require("./models/db");
3
4 //Importare module
5 const express = require("express");
6 const path = require("path");
7 const exphbs = require("express-handlebars");
8 const bodyParser = require("body-parser");
9
10 //Importare functii de routing Express
11 const repairController = require("./controllers/repairController");
12 //Importare functie ce permite encodarea URL-ului atunci cand se face request-uri HTTP
13 const { urlencoded } = require("body-parser");
14
15 //Initierea aplicatiei Express
16 var app = express();
17
18 //Encodarea URL-ului atunci cand se face un request "POST"
19 app.use(
20   bodyParser.urlencoded({
21     extended: true,
22   })
23 );
24
25 //
26 app.use(bodyParser.json());
27
28 //Declararea locatiei unde se afla fisierele template pentru Handlebars
29 app.set("views", path.join(__dirname, "/views/"));
30 //Initializarea template engine-ului Handlebars
31 app.engine(
32   "hbs",
33   exphbs({
34     extname: "hbs",
35     defaultLayout: "mainLayout",
36     layoutsDir: __dirname + "/views/layouts/",
37   })
38 );
39 //Setarea engine-ului ca Hbs
40 app.set("view engine", "hbs");
41
42 //Functie ce asculta portul 3000 pentru a verifica daca aplicatia Express functioneaza corect
43 app.listen(3000, () => {
44   console.log("Express app started on port: 3000");
45 });
46
47 //Importarea functiilor de routing pentru actiunile CRUD ale aplicatiei
48 app.use("/repair", repairController);
```

Fisierul “repaircontroller.js” contine toate functiile aplicatiei ce permite routing request-urilor HTTP cu ajutorul framework-ului Express.



```
1 //Importare Express
2 const express = require("express");
3
4 //Atribuirea variabilei router functiile de routing din Express
5 var router = express.Router();
6
7 //Importare Mongoose si declararea variabilei ce v-a completa proprietatiile din DB schema Repair
8 const mongoose = require("mongoose");
9 const Repair = mongoose.model("Repair");
10
11 //Exceptie pentru a repara alertele cu functii deprecated
12 mongoose.set("useFindAndModify", false);
13
14 //Initializarea template-ului "addOrEdit" ce contine form
15 router.get("/", (req, res) => {
16   res.render("repair/addOrEdit", {
17     viewTitle: "Insert Repair",
18   });
19 });
20
21 //Functia Create - POST HTTP Request
22 router.post("/", (req, res) => {
23   if (req.body._id == "") insertRecord(req, res);
24   else updateRecord(req, res);
25 });


```

(Prima parte a fisierului)

```
1 //Functie ce creeaza o conexiuni intre field-urile din form-ul HTML si field-urile din Schema Repair
2 function insertRecord(req, res) {
3     var repair = new Repair();
4     repair.clientName = req.body.clientName;
5     repair.carModel = req.body.carModel;
6     repair.plateNumber = req.body.plateNumber;
7     repair.mechanicName = req.body.mechanicName;
8     repair.workedHours = req.body.workedHours;
9     repair.save((err, doc) => {
10         if (!err) res.redirect("repair/list");
11         else {
12             console.log("Error during record insertion:" + err);
13         }
14     });
15 }
16
17 //Functia Update
18 function updateRecord(req, res) {
19     Repair.findOneAndUpdate(
20         { _id: req.body._id },
21         req.body,
22         { new: true },
23         (err, doc) => {
24             if (!err) {
25                 res.redirect("repair/list");
26             } else console.log("Error: " + err);
27         }
28     );
29 }
30
31 //Functia Read - GET HHTP Request
32 router.get("/list", (req, res) => {
33     Repair.find((err, docs) => {
34         if (!err) {
35             res.render("repair/list", {
36                 list: docs,
37             });
38         } else {
39             console.log("Error in retrieving repair list: " + err);
40         }
41     }).lean();
42 });

```

(A doua parte a fisierului)



```

1 //Functia Update - POST HTTP Request
2 router.get("/:id", (req, res) => {
3   Repair.findById(req.params.id, (err, doc) => {
4     if (!err) {
5       res.render("repair/addOrEdit", {
6         viewType: "Update Repair",
7         repair: doc,
8       });
9     }
10   }).lean();
11 });
12
13 //Functia Delete - DELETE HTTP Request
14 router.get("/delete/:id", (req, res) => {
15   Repair.findByIdAndRemove(req.params.id, (err, doc) => {
16     if (!err) {
17       res.redirect("/repair/list");
18     } else {
19       console.log("Error during delete: " + err);
20     }
21   }).lean();
22 });
23
24 //Expunerea variabilei router ca modul
25 module.exports = router;

```

(A treia parte a fisierului)

Fisierul “mainLayout.hbs” este un wrapper HTML ce este reutilizat pentru a afisa diferite “views” ale aplicatiei.

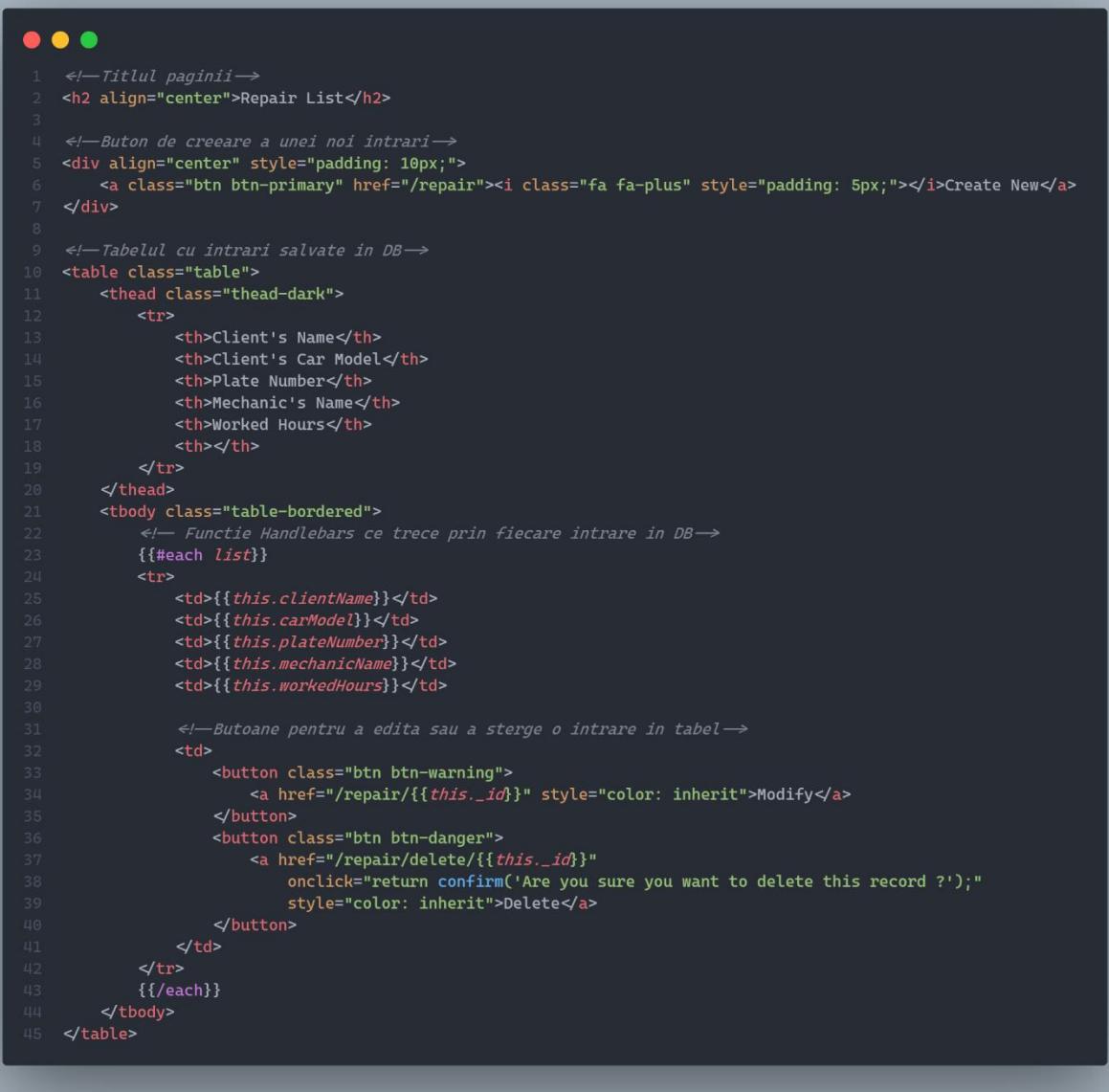


```

1 <!DOCTYPE html>
2
3 <html>
4 <head>
5   <title>Repair Shop Calendar App</title>
6
7   <!-- Import Bootstrap si Font Awesome Icons -->
8   <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
9   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
10 </head>
11
12 <body class="bg-light">
13   <div class="container-fluid">
14     <div class="row">
15       <div class="col-md offset-md text-center" style="background-color: white; margin-top: 25px; padding: 20px;">
16         {{body}}
17       </div>
18     </div>
19   </div>
20 </body>
21 </html>

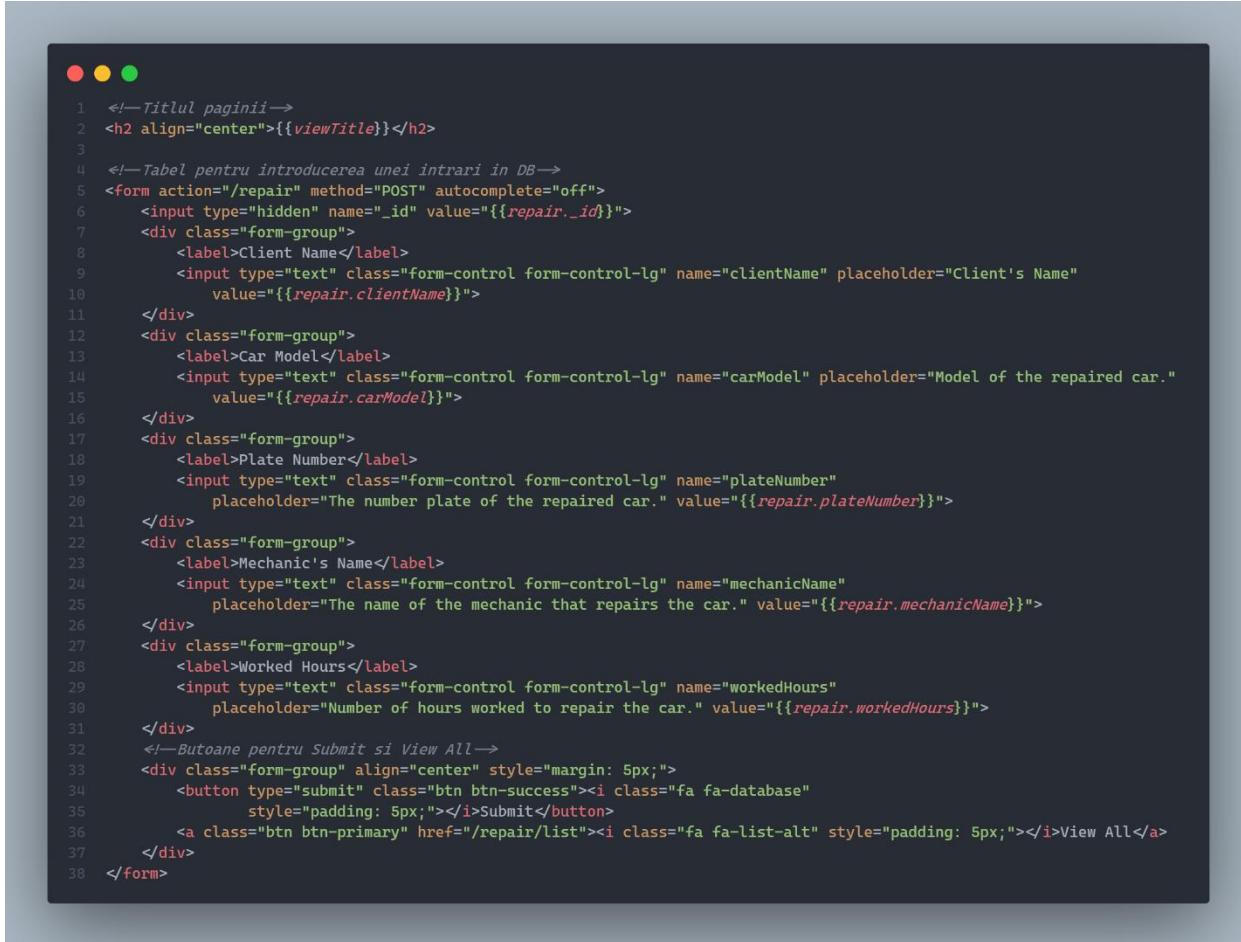
```

Fisierul “list.hbs” este template-ul folosit de Handlebars pentru a afisa pagina principala ce contine tabelul cu activitati extrase din baza de date si butoanele pentru actiunile declarate in sectiunea “Introducere”.



```
1  <!-- Titlul paginii-->
2  <h2 align="center">Repair List</h2>
3
4  <!--Buton de creeare a unei noi intrari-->
5  <div align="center" style="padding: 10px;">
6      <a class="btn btn-primary" href="/repair"><i class="fa fa-plus" style="padding: 5px;"></i>Create New</a>
7  </div>
8
9  <!--Tabelul cu intrari salvate in DB-->
10 <table class="table">
11     <thead class="thead-dark">
12         <tr>
13             <th>Client's Name</th>
14             <th>Client's Car Model</th>
15             <th>Plate Number</th>
16             <th>Mechanic's Name</th>
17             <th>Worked Hours</th>
18             <th></th>
19         </tr>
20     </thead>
21     <tbody class="table-bordered">
22         <!-- Functie Handlebars ce trece prin fiecare intrare in DB-->
23         {{#each list}}
24             <tr>
25                 <td>{{this.clientName}}</td>
26                 <td>{{this.carModel}}</td>
27                 <td>{{this.plateNumber}}</td>
28                 <td>{{this.mechanicName}}</td>
29                 <td>{{this.workedHours}}</td>
30
31                 <!-- Butoane pentru a edita sau a sterge o intrare in tabel-->
32                 <td>
33                     <button class="btn btn-warning">
34                         <a href="/repair/{{this._id}}" style="color: inherit">Modify</a>
35                     </button>
36                     <button class="btn btn-danger">
37                         <a href="/repair/delete/{{this._id}}"
38                             onclick="return confirm('Are you sure you want to delete this record ?');"
39                             style="color: inherit">Delete</a>
40                     </button>
41                 </td>
42             </tr>
43         {{/each}}
44     </tbody>
45 </table>
```

Fisierul “addOrEdit.hbs” este template-ul ce permite adaugarea sau editarea unei activitati in tabel.



The screenshot shows a code editor window with a dark theme. The title bar has three colored circles (red, yellow, green). The main area contains the following code:

```
1  <!-- Titlul paginii-->
2  <h2 align="center">{{viewTitle}}</h2>
3
4  <!-- Tabel pentru introducerea unei intrari in DB-->
5  <form action="/repair" method="POST" autocomplete="off">
6      <input type="hidden" name="_id" value="{{repair._id}}>
7      <div class="form-group">
8          <label>Client Name</label>
9          <input type="text" class="form-control form-control-lg" name="clientName" placeholder="Client's Name"
10             value="{{repair.clientName}}>
11      </div>
12      <div class="form-group">
13          <label>Car Model</label>
14          <input type="text" class="form-control form-control-lg" name="carModel" placeholder="Model of the repaired car."
15             value="{{repair.carModel}}>
16      </div>
17      <div class="form-group">
18          <label>Plate Number</label>
19          <input type="text" class="form-control form-control-lg" name="plateNumber"
20             placeholder="The number plate of the repaired car." value="{{repair.plateNumber}}>
21      </div>
22      <div class="form-group">
23          <label>Mechanic's Name</label>
24          <input type="text" class="form-control form-control-lg" name="mechanicName"
25             placeholder="The name of the mechanic that repairs the car." value="{{repair.mechanicName}}>
26      </div>
27      <div class="form-group">
28          <label>Worked Hours</label>
29          <input type="text" class="form-control form-control-lg" name="workedHours"
30             placeholder="Number of hours worked to repair the car." value="{{repair.workedHours}}>
31      </div>
32  <!-- Butoane pentru Submit si View All-->
33  <div class="form-group" align="center" style="margin: 5px;">
34      <button type="submit" class="btn btn-success"><i class="fa fa-database"
35                     style="padding: 5px;"></i>Submit</button>
36      <a class="btn btn-primary" href="/repair/list"><i class="fa fa-list-alt" style="padding: 5px;"></i>View All</a>
37  </div>
38 </form>
```

Functionalitatiile aplicatiei

Pagina principala ce contine lista cu activitati.

Repair List					
Client's Name	Client's Car Model	Plate Number	Mechanic's Name	Worked Hours	
Gabriel Alexandru	Renault Clio 4	B105FNJ	Vali Nas Mare	2	<button>Modify</button> <button>Delete</button>
Ion Iliescu	Mercedes Maybach S600	B01ION	Gica din coltul strazii	45	<button>Modify</button> <button>Delete</button>
Razvan Iliescu	Peugeot 508	B508PGT	Vali Nas Mare	3	<button>Modify</button> <button>Delete</button>

Pagina ce permite adaugarea unei activitati.

Insert Repair

Client Name

Car Model

Plate Number

Mechanic's Name

Worked Hours

Submit
View All

```
_id: ObjectId("5ffec7a94b87ec2a381346e7")
clientName: "Corobaia Vlad"
carModel: "Kia ProCeed"
plateNumber: "B345KIA"
mechanicName: "Vali Nas Mare"
workedHours: "7"
__v: 0
```

Repair List					
+ Create New					
Client's Name	Client's Car Model	Plate Number	Mechanic's Name	Worked Hours	
Gabriel Alexandru	Renault Clio 4	B105FNJ	Vali Nas Mare	2	Modify Delete
Ion Iliescu	Mercedes Maybach S600	B01ION	Gica din coltul strazii	45	Modify Delete
Razvan Iliescu	Peugeot 508	B508PGT	Vali Nas Mare	3	Modify Delete
Corobaia Vlad	Kia ProCeed	B345KIA	Vali Nas Mare	7	Modify Delete

Butonul “Modify” permite modificarea detaliilor unei intrari in tabel.

Repair List					
+ Create New					
Client's Name	Client's Car Model	Plate Number	Mechanic's Name	Worked Hours	
Gabriel Alexandru	Renault Clio 4	B105FNJ	Vali Nas Mare	2	Modify Delete
Ion Iliescu	Mercedes Maybach S600	B01ION	Gica din coltul strazii	45	Modify Delete
Razvan Iliescu	Peugeot 508	B508PGT	Vali Nas Mare	3	Modify Delete
Corobaia Vlad	Kia ProCeed	B345KIA	Gica din coltul strazii	13	Modify Delete

```

_id: ObjectId("5ffec7a94b87ec2a381346e7")
clientName: "Corobaia Vlad"
carModel: "Kia ProCeed"
plateNumber: "B345KIA"
mechanicName: "Gica din coltul strazii"
workedHours: "13"
__v: 0

```

Butonul “Delete” permite stergerea unei intrari in tabel.

A screenshot of a web application interface. At the top, there is a confirmation dialog box with a red border. The message inside says "localhost:3000 afisează mesajul" and "Are you sure you want to delete this record ?" with two buttons: "OK" and "Anulează". A red arrow points from the text above to this dialog box. Below the dialog is a table with the following data:

Client's Name	Client's Car Model	Plate Number	Mechanic's Name	Worked Hours		
Gabriel Alexandru	Renault Clio 4	B105FNJ	Vali Nas Mare	2	<button>Modify</button>	<button>Delete</button>
Ion Iliescu	Mercedes Maybach S600	B01ION	Gica din coltul strazii	45	<button>Modify</button>	<button>Delete</button>
Razvan Iliescu	Peugeot 508	B508PGT	Vali Nas Mare	3	<button>Modify</button>	<button>Delete</button>
Corobaia Vlad	Kia ProCeed	B345KIA	Gica din coltul strazii	13	<button>Modify</button>	<button>Delete</button>

A screenshot of a "Repair List" page. At the top, there is a blue button labeled "+ Create New". Below it is a table with the same data as the previous screenshot. The last row, which corresponds to the entry for "Corobaia Vlad", is missing. Instead, there is a red-bordered box at the bottom containing the message "Intrarea din tabel cu "Corobaia Vlad" a fost eliminata." (The entry in the table with "Corobaia Vlad" was deleted.)