

# App Management

- Authentication part
- Electric Vehicle Management part

## Programs:

- *Postman*
- *Atlas - MongoDB Cloud*
- *JSON Web Token - <https://jwt.io/>*

# package.json

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure with files like package.json, node\_modules, server, controller, database, model, routes, services, test, views, a\_db.js, a\_index.js, config.env, info.log, joi.http, logger.js, package-lock.json, and good\_luck.txt.
- Editor View:** The package.json file is open in the main editor area. The code content is as follows:

```
1  {
2    "name": "exam-andreea-popa",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1",
8      "start": "nodemon server.js"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "axios": "^0.27.2",
15     "bcrypt": "^5.0.1",
16     "body-parser": "^1.20.0",
17     "dotenv": "^16.0.1",
18     "ejs": "3.1.8",
19     "express": "4.18.1",
20     "express-validator": "^6.14.2",
21     "joi": "17.6.0",
22     "jsonwebtoken": "^8.5.1",
23     "mongodb": "2.0.0",
24     "mongoose": "4.8.1",
25     "morgan": "1.10.0",
26     "nodemon": "2.0.19",
27     "pm2": "5.2.0",
28     "winston": "3.8.1",
29     "winston-mongodb": "5.0.7",
30     "winston-transport": "4.5.0"
31   },
32   "devDependencies": {
33     "chai": "4.3.6",
34     "jest": "28.1.3",
35     "mocha": "10.0.0",
36     "supertest": "6.2.4"
37   }
38 }
39 }
```

The status bar at the bottom indicates: Ln 27, Col 26 Spaces: 2 UTF-8 LF () JSON ⚡ Go Live ⚡ Spell ⚡

# Authentication - User Validation - joi

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows a project structure under "NODECOURSE" containing files like add\_piece.ejs, add\_piece.html, index.ejs, index.html, update\_piece.ejs, a\_db.js, a\_index.js, config.env, info.log, joi.http, logger.js, package-lock.json, package.json, server.js, and technical\_details.txt.
- Editor:** The main editor pane displays a file named "joi.http" which contains Node.js code. It defines a schema for "password" and "email" fields, and handles a POST request to "/api/signup".

```
// Define Schema
const schema = Joi.object({
  password: Joi.string()
    .min(4)
    .max(32)
    .required(),
  email: Joi.string()
    .email({
      minDomainSegments: 5,
      tlds: {
        allow: [ 'com', 'net', 'mx', 'ro' ]
      }
    })
});

router.post('/api/signup', (req, res) => {
  const { body } = req;
  try {
    const { error } = schema.validate(body);
    if (error) {
      res.status(400).send({
        message: error
      });
    } else {
      res.status(201).send({ message: 'OK' });
    }
  } catch (error) {
    res.status(500).send({ message: error.message });
  }
});
```
- Terminal:** The terminal shows output from npm audit and node a\_index.js, indicating no vulnerabilities found.

```
added 6 packages, and audited 700 packages in 3s
58 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
exam-andreea-popa git:(exam-andreea-popa) ✘ node a_index.js
{"level":"info","message":"Server running on http://localhost:5000","timestamp":"2022-08-05T09:14:23.776Z"}
```
- Status Bar:** Shows the current file is "joi.http", the file path is "nodeRepo/nodeRepo/exams/exam-andreea-popa/a\_routes/joi.http", the file type is "HTTP", and the status bar includes "Ln 1, Col 31", "Spaces: 4", "UTF-8", "LF", "HTTP", "Go Live", "No Environment", "Spell", and "Connect".

# Authentication - User Validation

The image shows a developer's environment with two main windows:

- VS Code Editor:** The left window displays the file `a_auth.js` (marked with a red 'U' icon). The code implements user validation logic for a sign-up endpoint. It includes validation for email and password, checks if the user already exists, and handles errors by returning a 400 status with an array of validation errors.
- API Testing Tool:** The right window is a browser-based tool for testing APIs. It shows a POST request to `http://localhost:5000/auth/signup`. The "Body" tab is selected, showing JSON input:

```
1 {  
2   ... "email": "happyhamster@gmail.com",  
3   ... "password": "carrotsAreMyLife"  
4 }
```

The response status is 200, and the message "Validation passed" is displayed.

# Authentication - password hashing

The image shows a developer's workspace with two main components:

- VS Code Editor:** On the left, the code for `a_index.js` is displayed. The code handles user validation and password hashing using bcrypt. It includes a POST endpoint for sign-up and a GET endpoint for retrieving all users.
- Postman API Client:** On the right, a browser tab titled "exam-andreea-popas\*" shows an API request to "Auth / Get all users". The request method is GET, and the URL is `http://localhost:5000/auth/all`. The "Headers" tab shows "Content-Type: application/json". The "Body" tab is set to "Pretty" and displays the JSON response from the API, which contains two user objects with email and password fields.

```
VS Code Editor Content (a_index.js excerpt):  
33     if(user) {  
34         res.status(400).json({  
35             "errors": [  
36                 {"msg": "User already existent"}  
37             ]  
38         })  
39     }  
40  
41     //console.log(email, password);  
42  
43     let hashedPassword = await bcrypt.hash(password, 10); // the bigger the  
44     // the better the security  
45  
46     users.push({ // in a_db  
47         email,  
48         password: hashedPassword  
49     })  
50  
51     console.log(hashedPassword);  
52  
53     res.send("Validation passed");  
54 }  
55  
56 router.get("/all", (req, res) => {  
57     res.json(users);  
58 } )  
59  
60 module.exports = router;
```

```
Postman Response Body:  
1 [ {  
2     "email": "andreea@yahoo.com",  
3     "password": "jaasss"  
4 }, {  
5     "email": "happyhamster@gmail.com",  
6     "password": "$2b$10$92ZvMMNiMWVCAckytvbtx.zw0IXmYJfNeaNc2Q7XDhv4tFhp2cIw."  
7 } ]  
8  
9  
10 ]
```

# JWT

The screenshot shows a developer's environment with multiple windows open:

- VS Code Editor:** The left pane displays the file `a_index.js` (marked with a green 'U' icon). The code handles user validation and database insertion. It includes a check for existing users, hashing of passwords, and sending validation messages.
- API Network Tab:** The right pane shows a browser-based API tool with the following details:
  - Overview:** GET `http://localhost:3000/e`, POST `SignUp`, GET `Get all users`.
  - Auth / Get all users:** A GET request to `http://localhost:5000/auth/all`.
  - Headers (6):** Shows the number of headers.
  - Body:** Shows the response body in JSON format.
  - Query Params:** A table with one row: `Key` (Value) and `Value` (Des).
  - Body:** Shows the raw JSON response.
  - Cookies:** Shows the raw JSON response.
  - Headers (7):** Shows the raw JSON response.
  - Test Results:** Shows the raw JSON response.
- Terminal:** The bottom pane shows the output of the command `[nodemon] starting 'node a_index.js'`. It indicates the server is running on port 5000 and provides a long, encoded password value.

# JWT



JWT

Debugger Libraries Introduction Ask

Crafted by  auth0 ?

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJl  
bWFpbCI6ImhhcHB5Y2F0QGdtYWlsLmNvbSIsI  
mlhdCI6MTY1OTYwNDA4NywiZXhwIjoxNjYzMjA0  
MDg3fQ.KqrK-  
zW_RxSmDYYATvg3nBNddw0wfaEhViAlMP4B0kY
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "email": "happyCat@gmail.com",  
  "iat": 1659604087,  
  "exp": 1663204087  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  randomtextjaja  
)  secret base64 encoded
```

Weak secret!

 Signature Verified

SHARE JWT

# JWT

The screenshot shows the Postman application interface. At the top, there are navigation tabs: Home, Workspaces, API Network, and Explore. A search bar says "Search Postman" and there are buttons for "Invite", "Settings", "Notifications", and "Upgrade". Below the header, there's a sidebar with sections like Overview, Auth / SignUp, and a detailed view of a POST request to `http://localhost:5000/auth/signup`. The request has 9 Headers. One header, `x-auth-token`, is checked and has a value of `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImhhcHB5Y2F0Q...KqzK-zW-RxSmDYYATvg3nBNddw0wfahVia1MP4B0kY`. The "Body" tab is selected, showing a JSON response:

```
1 {  
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImhhcHB5Y2F0QGdtYWlsLmNvbSIsImlhCI6MTY1OTYwNDA4NywiZXhwIjoxNjYzMjA0MDg3fQ.  
3 }
```

At the bottom, there are status indicators: Status: 200 OK, Time: 129 ms, Size: 415 B, and a "Save Response" button. The footer includes links for Online, Find and Replace, Console, Cookies, Capture requests, Bootcamp, Runner, Trash, and Help.

# SignUp, GET all users

The screenshot shows two separate Postman sessions side-by-side.

**Left Session (Auth / SignUp):**

- Method: POST
- URL: `http://localhost:5000/auth/signup`
- Body tab selected, showing JSON input:

```
1 "email": "happydog@gmail.com",
2 ...
3 "password": "myDearHumanIsMyLife123"
```
- Body tab expanded to show response:

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
3   eyJlbWFpbCI6ImhhcHB5ZG9nQGdtYWlsLmNvbSIiMlhdCI6MTY1OTYwNTc3NywiZXhwIjoxNjYzMjA1Nzc3fQ.
4   FVNQFynJSFVmJzoCMUmu2xT1WOMIaCJHDdgbZXQFkQ"
5 }
```

**Right Session (Auth / Get all users):**

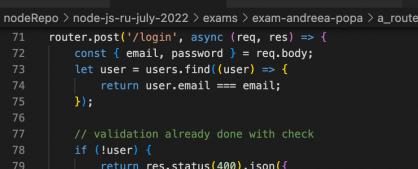
- Method: GET
- URL: `http://localhost:5000/auth/all`
- Query Params tab expanded, showing table:

KEY	VALUE	DESCRIPTION
Key	Value	Description
- Body tab expanded to show response:

```
1 {
2   "email": "andrea@yahoo.com",
3   "password": "jaasss"
4 },
5 {
6   "email": "happydog@gmail.com",
7   "password": "$2b$10$meV662SFVbrRdY4q1poan.YmACTjHe/Rlo96ofz/AnH527DAjm6Wy"
8 }
```

Both sessions have tabs for Body, Cookies, Headers, and Test Results, and status bars indicating Status: 200 OK and Time: 5 ms.

# Login



```
JS a_index.js U JS a_auth.js U ● ⚡ technical_details.txt U
nodeRepo > nodejs-ru-july-2022 > exams > exam-andreea-popa > a_routes > JS a_index.js
1 router.post('/login', async (req, res) => {
2     const { email, password } = req.body;
3     let user = users.find((user) => {
4         return user.email === email;
5     });
6
7     // validation already done with check
8     if (!user) {
9         return res.status(400).json({
10             "errors": [
11                 {
12                     "msg": "Invalid Credentials"
13                 }
14             ]
15         });
16     }
17
18     let isMatch = await bcrypt.compare(password, user.password);
19
20     if (!isMatch) {
21         return res.status(400).json({
22             "errors": [
23                 {
24                     "msg": "Invalid Credentials"
25                 }
26             ]
27         });
28     }
29
30     res.json(user);
31 }
```

The screenshot shows a terminal window with several tabs open, indicating a Node.js development environment. The tabs include:

- `JS a_index.js U`
- `JS a_auth.js U`
- `technical_details.txt U`
- `a_routes > JS a`

The main content area displays the code for `a_auth.js`:

```
nodeRepo > node-js-ru-july-2022 > exams > exam-andreea-popa > a_routes > JS a

71 router.post('/Login', async (req, res) => {
72   const { email, password } = req.body;
73   let user = users.find({user});
74   return user.email === email;
75 });
76
77 // validation already done with check
78 if (!user) {
79   return res.status(400).json({
80     "errors": [
81       {
82         "msg": "Invalid Credentials"
83       }
84     ]
85   });
86
87   let isMatch = await bcrypt.compare(password, user.password);
88
89   if (!isMatch) {
90     return res.status(400).json({
91       "errors": [
92         {
93           "msg": "Invalid credentials"
94         }
95       ]
96     });
97
98   const token = await JWT.sign({ // to create token
99     email // the payload, not sensitive info
100   }
```

At the bottom of the terminal, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The DEBUG CONSOLE tab is currently selected.

Output from the terminal shows the server running on port 5000:

```
Server running on http://localhost:5000
[nodemon] restarting due to changes...
[nodemon] starting `node a_index.js`
Server running on http://localhost:5000
[nodemon] restarting due to changes...
[nodemon] starting `node a_index.js`
Server running on http://localhost:5000
```

The screenshot shows the Postman interface with the following details:

- Header Bar:** Home, Workspaces, API Network, Explore.
- Left Sidebar:** Overview, Auth / Login.
- Request Section:** Method: POST, URL: http://localhost:5000/auth/login.
- Params Tab:** none selected.
- Authorization Tab:** None.
- Headers Tab:** (8) items listed.
- Body Tab (selected):** Type: JSON, Value:

```
1 ... "email": "happydog@gmail.com",
2 ... "password": "myDearHumanIsMyLife123"
```
- Pre-request Script Tab:** None.
- Tests Tab:** None.
- Settings Tab:** None.

The screenshot shows the Postman application interface. At the top, there are three tabs: 'Overview' (selected), 'Workspaces', and 'API Network'. Below the tabs, there are four items: 'GET http://localhost:3C' (red dot), 'POST SignUp' (orange dot), 'GET Get all users' (red dot), and 'POST Login' (orange dot). The main area shows a 'Auth / Login' collection. A 'POST' request is selected with the URL 'http://localhost:5000/auth/login'. The 'Body' tab is active, showing a JSON payload:

```
1 ...
2   "email": "happydog@gmail.com",
3   "password": "myMehHumanIsMyLife123"
4
```

The 'JSON' dropdown is open. Below the body, there are tabs for 'Body', 'Cookies', 'Headers (7)', and 'Test Results'. The 'Body' tab is active, showing the JSON response:

```
1 ...
2   "errors": [
3     {
4       "msg": "Invalid credentials"
5     }
6   ]
7
```

The 'JSON' dropdown is open. At the bottom, there are status indicators: 'Online' (green), 'Find and Replace' (blue), 'Console' (grey), and 'Cook' (blue).

# Authorization

The screenshot shows the Visual Studio Code interface. On the left, there are several icons for file operations like copy, paste, and search. The main area contains three code editors:

- JS a\_db.js U**: A script defining two pieces of hardware: a DC Motor and an AC Motor.
- JS a\_index.js U**: An Express router configuration.
- JS a\_auth.js U**: A script defining two pieces of hardware: a Charger and a Reducer.

The terminal at the bottom shows the server is running on port 5000.

```
const publicPieces = [
  {
    piecetype: "DC Motor",
    piecenumber: "543",
    descrip: "DC Motor for EV"
  },
  {
    piecetype: "AC Motor",
    piecenumber: "943",
    descrip: "AC Motor for EV"
  }
]

const router = require('express').Router();
const {publicPieces, privatePieces} = require('../a_db');

router.get('/public', (req,res)=>{
  res.json(publicPieces);
})

router.get('/private', (req,res)=>{
  res.json(privatePieces);
})

module.exports = router;
```

PROBLEMS 23 OUTPUT DEBUG CONSOLE TERMINAL GITLENS: VISUAL FILE HISTORY

```
Server running on http://localhost:5000
[nodemon] restarting due to changes...
[nodemon] starting 'node a_index.js'
Server running on http://localhost:5000
[nodemon] restarting due to changes...
[nodemon] starting 'node a_index.js'
Server running on http://localhost:5000
```

The screenshot shows the Postman application interface. At the top, it displays the URL `http://localhost:5000/pieces/public`. The request method is set to `GET`. The response body is shown in JSON format:

```
{
  "piecetype": "DC Motor",
  "piecenumber": "543",
  "descrip": "DC Motor for EV"
},
{
  "piecetype": "AC Motor",
  "piecenumber": "943",
  "descrip": "AC Motor for EV"
}
```

The interface also includes sections for Headers, Cookies, and Test Results. On the right side, there is a detailed view of the API network tab, showing the same JSON response.

# Authorization

- Token could not be accessed

The screenshot shows a developer's workspace with several open files in VS Code:

- `a_index.js`: A simple Express Router setup.
- `a_auth.js`: Handles user authentication logic.
- `a_piece.js`: The main file being edited, which contains the logic for handling public and private pieces.
- `a_db.js`: Database access logic.
- `technical_...`: Another file in the technical directory.

The `a_piece.js` file contains the following code:

```
const router = require("express").Router();
const { publicPieces, privatePieces } = require("../a_db");

router.get('/public', (req, res) => {
  res.json(publicPieces);
})

// next -> accessed if everything is fine with the req
router.get('/private', (req, res, next) => {
  let userValid = false;

  if (userValid) {
    next();
  } else {

    return res.status(400).json({
      "errors": [
        {
          "msg": "Access denied"
        }
      ]
    })
  }
  res.json(privatePieces);
})

module.exports = router;
```

Postman API request details:

- Method: GET
- URL: `http://localhost:5000/pieces/private`
- Headers: (6)
- Query Params:

KEY	VALUE
Key	Value

- Body:

Pretty	Raw	Preview	Visualize	JSON
1 2 3 4 5 6 7	"errors": [ { "msg": "Access denied" } ]			

# Check middleware checkAuth

The image shows two side-by-side Postman interface screenshots demonstrating the use of a middleware named `checkAuth`.

**Left Panel (Request 1):**

- Method:** POST
- URL:** `http://localhost:5000/auth/signup`
- Headers:** (9 items)
- Body (JSON):**

```
1 {
2   "email": "happydog@gmail.com",
3   "password": "myDearHumanIsMyLife123"
4 }
```

**Right Panel (Request 2):**

- Method:** GET
- URL:** `http://localhost:5000/pieces/private`
- Headers:** (7 items)
- Header x-auth-token:** eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI...
- Body (JSON):**

```
1 {
2   "piecetype": "Charger",
3   "piecenumber": "2345",
4   "descrip": "Charger for EV"
5 }
```

# Supertest

The screenshot shows a Visual Studio Code interface with the following details:

- EXPLORER**: Shows a tree view of files and folders. Opened files include `a_index.js`, `a_index.test.js`, and `a_piece.test.js`. Other files like `a_db.js` and `a_index.html` are also listed.
- TABS**: Three tabs are open:
  - `a_index.js` (marked with a green checkmark)
  - `a_index.test.js` (marked with a green checkmark)
  - `a_piece.js` (marked with a red X)
- TERMINAL**: The terminal shows command-line output related to module loading and promise execution.
- PROBLEMS**: A sidebar showing potential issues or warnings.
- OUTPUT**: A sidebar showing build or test results.
- DEBUG CONSOLE**: A sidebar showing debugger-related information.
- GITLENS: VISUAL FILE HISTORY**: A sidebar showing file history for the current file.
- SQL CONSOLE: MESSAGES**: A sidebar showing messages from the SQL console.
- STATUS BAR**: Shows the current workspace path as `node - exam-andreea-popa` and various icons for file operations.

# Logger

The screenshot shows a Node.js application structure in VS Code:

- EXPLORER:** Shows files and folders:
  - OPEN EDITORS:** GROUP 1: JS a\_index.js (active), JS logger.js, JS package.json, JS a\_auth.js, JS server.js. GROUP 2: JS technical\_details.txt, JS router.js, services render.js, JS a\_db.js, JS a\_index.js (highlighted), config.env, info.log, JS logger.js, JS package-lo..., JS package.json, JS server.js, JS technical\_d..., JS good\_luck.txt.
  - NODECOURSE:** JS router.js, services render.js, JS a\_db.js, JS a\_index.js (highlighted), config.env, info.log, JS logger.js, JS package-lo..., JS package.json, JS server.js, JS technical\_d..., JS good\_luck.txt.
- CODE EDITOR:** JS a\_index.js (NodeCourse)

```
//const PORT = 5000;
ap.listen(5000, () => {
  logger.log('info',`Server running on http://localhost:5000`)
})
```
- TERMINAL:**

```
exam-andreea-popa git:(exam-andreea-popa) ✘ npm start
> exam-andreea-popa@1.0.0 start
> nodemon a_index.js

[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting node a_index.js
{"level":"info","message":"Server running on http://localhost:5000","timestamp":"2022-08-04T14:10:11.069Z"}
[nodemon] restarting due to changes...
[nodemon] starting 'node a_index.js'
{"level":"info","message":"Server running on http://localhost:5000","timestamp":"2022-08-04T14:11:34.374Z"}
```

## Car Management System

[New Piece](#) 

ID	PieceType	Number of Pieces	Description	Action
1	AC Motor	1789	AC Motor for EV	
2	PCU	379	Electric Power Control Unit for EV	
3	Battery	3456	Super-Battery for EV	
4	DC Motor	847	DC Motor for EV	
5	Reducer	724	Reducer for EV	
6	Charger	6284	Charger for EV	

Data | Cloud: MongoDB Cloud

cloud.mongodb.com/v2/62ea4aa32459d97e2a6a152a#metrics/replicaSet/62ea4af2ab939d02a518c835/explorer/test/piecedbs/find

All Clusters Get Help Andreea-Cristiana

PIECES Atlas App Services Charts

DEPLOYMENT Database Data Lake PREVIEW

DATA SERVICES Triggers Data API Data Federation

SECURITY Database Access Network Access Advanced

ANDREEA-CRISTIANA'S ORG - 2022-08-03 > PIECES > DATABASES

# Cluster0

VERSION 5.0.10 REGION AWS Frankfurt (eu-central-1)

Overview Real Time Metrics Collections Search Profiler Performance Advisor Online Archive Cmd Line Tools

DATABASES: 1 COLLECTIONS: 1

+ Create Database

Search Namespaces

test piecedbs

STORAGE SIZE: 36KB TOTAL DOCUMENTS: 6 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes 6

INSERT DOCUMENT

FILTER { field: 'value' } OPTIONS Apply Reset

QUERY RESULTS: 1-6 OF 6

\_id: ObjectId("62ea5db654de6c9b8ba85959")  
pieceType: "AC Motor"  
pieceNumber: "1789"  
descript: "AC Motor for EV"  
\_\_v: 0

\_id: ObjectId("62ea6e343b13ca992ac6d3a3")  
pieceType: "PCU"  
pieceNumber: "379"  
descript: "Electric Power Control Unit for EV"

Overview    GET Show Pieces    PUT Update Piece    DEL Delete Piece    POST New Request    +    ...    No Environment    Save    ...   

Pieces / Show Pieces

GET http://localhost:3000/api/pieces

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 63 ms Size: 945 B Save Response

Pretty Raw Preview Visualize JSON

```
13 "descript": "Electric Power Control Unit for EV",
14 "__v": 0
15 },
16 {
17   "_id": "62ea833fcfcf0eb9e3bb0e9d3",
18   "pieceType": "Battery",
19   "pieceNumber": "3456",
20   "descript": "Super-Battery for EV",
21   "__v": 0
22 },
23 {
24   "_id": "62eaa1faa85e725cf85aecd3",
25   "pieceType": "DC Motor",
26   "pieceNumber": "847",
27   "descript": "DC Motor for EV",
28   "__v": 0
29 },
30 {
31   "_id": "62eaa265a85e725cf85aecd7",
32   "pieceType": "Reducer",
33   "pieceNumber": "724",
34   "descript": "Reducer for EV",
35   "__v": 0
36 },
37 {
38   "_id": "62ec32d0dc2fa0c60bb7bf6a",
39   "pieceType": "Charger",
40   ...
41 }
```

Get

# Update

Home Workspaces API Network Explore Search Postman Invite Settings Upgrade No Environment

Overview GET Show Pieces PUT Update Piece DEL Delete Piece POST New Request + ... Save ... Send Cookies

Pieces / Show Pieces

GET http://localhost:3000/api/pieces

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
18:     "pieceType": "Battery",
19:     "pieceNumber": "3456",
20:     "descript": "Super-Battery for EV",
21:     "__v": 0
22:   },
23:   {
24:     "_id": "62eaa1faa85e725cf85aecd3",
25:     "pieceType": "DC Motor",
26:     "pieceNumber": "847",
27:     "descript": "DC Motor for EV",
28:     "__v": 0
29:   },
30:   {
31:     "_id": "62eaa265a85e725cf85aecd7",
32:     "pieceType": "Reducer",
33:     "pieceNumber": "4832",
34:     "descript": "Reducer for EV",
35:     "__v": 0
36:   },
37:   {
38:     "_id": "62ec32d0dc2fa0c60bb7bf6a",
39:     "pieceType": "Charger",
40:     "pieceNumber": "6284",
41:     "descript": "Charger for EV",
42:     "__v": 0
43:   }
44: ]
```

Status: 200 OK Time: 55 ms Size: 946 B Save Response

PUT http://localhost:3000/api/pieces/62eaa265a85e725cf85aecd7

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> pieceNumber	4832

Key Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1:   {
2:     "_id": "62eaa265a85e725cf85aecd7",
3:     "pieceType": "Reducer",
4:     "pieceNumber": "24",
5:     "descript": "Reducer for EV",
6:     "__v": 0
7:   }
```

Online Find and Replace Console Cookies Capture requests Bootcamp Runner Trash

# Insert

Electric Car Pieces × +

localhost:3000/add-piece

## Car Management System

«All Pieces

New Piece

Insert the piece data

Piece Type  
Super-Charger

Number of Pieces  
746

Description  
Super-Charger for Electric Vehicles

Save

Electric Car Pieces × +

localhost:3000

## Car Management System

New Piece 🚗

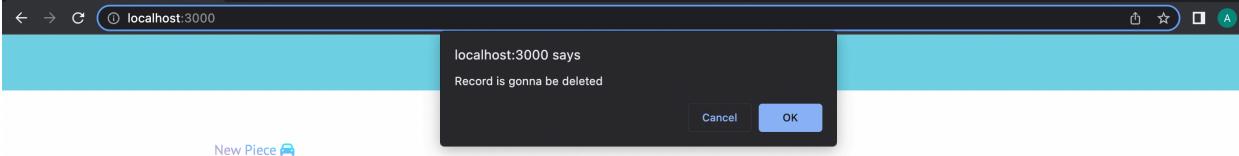
ID	PieceType	Number of Pieces	Description	Action
1	AC Motor	1789	AC Motor for EV	
2	PCU	379	Electric Power Control Unit for EV	
3	Battery	3456	Super-Battery for EV	
4	DC Motor	847	DC Motor for EV	
5	Reducer	4832	Reducer for EV	
6	Charger	6284	Charger for EV	
7	Super-Charger	746	Super-Charger for Electric Vehicles	

# Delete

Electric Car Pieces

New Piece 🚗

Car Manager



ID	PieceType	Number of Pieces	Description	Action
1	AC Motor	1789	AC Motor for EV	
2	PCU	379	Electric Power Control Unit for EV	
3	Battery	3456	Super-Battery for EV	
4	DC Motor	847	DC Motor for EV	
5	Reducer	4832	Reducer for EV	
6	Charger	6284	Charger for EV	
7	Super-Charger	746	Super-Charger for Electric Vehicles	

New Piece 🚗

ID	PieceType	Number of Pieces	Description	Action
1	AC Motor	1789	AC Motor for EV	
2	PCU	379	Electric Power Control Unit for EV	
3	Battery	3456	Super-Battery for EV	
4	DC Motor	847	DC Motor for EV	
5	Reducer	4832	Reducer for EV	
6	Charger	6284	Charger for EV	

# New Delete

Electric Car Pieces

localhost:3000

New Piece 🚗

localhost:3000 says  
Record is gonna be deleted

Cancel OK

ID	PieceType	Number of Pieces	Description	Action
1	AC Motor	1789	AC Motor for EV	
2	PCU	379	Electric Power Control Unit for EV	
3	Battery	3456	Super-Battery for EV	
4	DC Motor	847	DC Motor for EV	
5	Reducer	4832	Reducer for EV	
6	Charger	6284	Charger for EV	

Data | Cloud: MongoDB Cloud

cloud.mongodb.com/v2/62ea4aa32459d97e2a6a152a#metrics/replicaSet/62ea4af2ab939d02a518c835/explorer/test/piecedbs/find

All Clusters Get Help Andreea-Cristiana

PIECES Atlas App Services Charts

+ Create Database

Search Namespaces

DEPLOYMENT Database Data Lake PREVIEW

DATA SERVICES Triggers Data API Data Federation

SECURITY Database Access

test piecedbs

test.piecedbs

STORAGE SIZE: 36KB TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 34KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER { field: 'value' } OPTIONS Apply Reset

QUERY RESULTS: 1-5 OF 5

```
_id: ObjectId("62ea5db654de6c9b8ba85959")
pieceType: "AC Motor"
pieceNumber: "1789"
descript: "AC Motor for EV"
__v: 0
```

```
_id: ObjectId("62ea6e343b13ca992ac6d3a3")
pieceType: "PCU"
pieceNumber: "379"
descript: "Electric Power Control Unit for EV"
__v: 0
```

```
_id: ObjectId("62ea833fcfc0eb9e3bb0e9d3")
```

Atlas Blog Contact Sales

# Configurable Environment

The screenshot shows a code editor interface with the following details:

- EXPLORER View:** Shows a file tree structure for a project named "NODECOURSE". The tree includes "assets", "css", "img", "js" (containing "index.js"), "node\_modules", "server" (containing "controller", "model", "routes", "services", "render.js", and "views"), and "database" (containing "connection.js").
- connection.js File:** A JavaScript file located at `nodeRepo > node-js-ru-july-2022 > exams > exam-andreea-popa > server > database > connection.js`. The code connects to MongoDB using mongoose and logs the connection status.
- config.env File:** An environment variable file located at `nodeRepo > node-js-ru-july-2022 > exams > exam-andreea-popa > config.env`. It defines the port and MongoDB URI.
- TERMINAL View:** Shows the output of a command: `GET /api/pieces 200 1010 - 44.274 ms`.
- STATUS BAR:** Displays the current file as "exam-andreea-popa\*", line count (0), character count (8), and various status indicators like "Connect".

# ODM-typed Model

The screenshot shows a VS Code interface with the following components:

- Explorer** (left sidebar): Shows the project structure for "nodeCourse". The "index.js" file is open in the editor.
- Editor**: The main area displays the "index.js" file content. It includes imports from "mongoose" and "express", definitions for "Piece" schema and "Piecedb" model, and a function that handles an "update" event by sending an AJAX request to update a piece in the database.
- Terminal**: Shows the command "node - exam-andreea-popa" followed by a list of network requests and their times.

```
const mongoose = require('mongoose');
const pieceType = {
  type: String,
  required: true
};
const pieceNumber = {
  type: String,
  required: true
};
const descriptor = {
  type: String,
  required: true
};

const Piecedb = mongoose.model('piecedb', schema);

module.exports = Piecedb;
```

```
GET /js/index.js 304 - - 0.816 ms
GET /api/pieces 200 1010 - 57.236 ms
GET / 200 7147 - 79.841 ms
GET /css/style.css 304 - - 2.061 ms
GET /js/index.js 304 - - 1.637 ms
GET /api/pieces 200 1010 - 43.080 ms
DELETE /api/pieces/delete/62ec2833dc2f1a0c60bb7bf56 404 184 - 2.893 ms
GET /api/pieces 200 1010 - 44.274 ms
```

# Model-View-Controller

The screenshot shows a code editor with three tabs open, illustrating the Model-View-Controller (MVC) architecture:

- model.js**: Handles the data model. It defines a schema for a piece using the mongoose library, specifying fields for pieceType, pieceNumber, and description.
- index.ejs**: Handles the view. It contains an HTML template for displaying a list of pieces and a form for adding new pieces. The template uses Bootstrap classes for styling.
- controller.js**: Handles the controller logic. It exports two functions: `create` and `find`. The `create` function validates input, saves a new piece to the database, and returns a response. The `find` function retrieves pieces from the database and returns them.

The code editor interface includes a sidebar with file navigation, a bottom navigation bar with tabs like PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, GITLENS: VISUAL FILE HISTORY, and SQL CONSOLE: MESSAGES, and a status bar at the bottom.

```
model.js
const mongoose = require('mongoose');
var schema = new mongoose.Schema({
  pieceType: {
    type: String,
    required:true
  },
  pieceNumber: {
    type: String,
    required:true
  },
  descript: {
    type: String,
    required: true
  }
}); //define a shape of the doc
const Piecedb = mongoose.model('piecedb', s
module.exports = Piecedb;
```

```
index.ejs
de header -->
e('include/_header')>
ude header -->
ain Site -->
id="site-main">
iv class="container">
<div class="box-nav d-flex justify-between">
<a href="/add-piece" class="border-shadow">
New Piece
</a>
</div>
<!-- form handling -->
<form action="/" method="POST">
<table class="table">
  <thead class="thead-dark">
    <tr>
      <th>ID</th>
      <th>PieceType</th>
      <th>Number of Pieces</th>
      <th>Description</th>
    <!--
      <th>Motor</th>
      <th>Reducer</th>
      <th>Battery</th>
      <th>On-board charger</th>
      <th>Electric Power Control</th>
    -->
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
```

```
controller.js
//const { rawListeners } = require('../model/model');
var Piecedb = require('../model/model');
// creating and saving new piece -> API
exports.create = (req, res) => {
  //validate req, if the user makes a post request
  if (!req.body) {
    res.status(400).send({ message: "please provide some data" });
    return;
  }
  //new piece
  const piece = new Piecedb({
    pieceType: req.body.pieceType,
    pieceNumber: req.body.pieceNumber,
    descript: req.body.descript
  });
  // save piece in db
  piece
    .save(piece)
    .then(data => {
      //res.send(data);
      res.redirect('/add-piece'); // redirecting
    })
    .catch(error => {
      res.status(500).send({
        message: "Piece could not be saved"
      });
    });
}
// retrieve and return all/one pieces
exports.find = (req, res) => {
```

GET /api/pieces 200 1010 - 44.274 ms

# Monitoring tool - pm2

The image shows a terminal window and a Postman interface side-by-side.

**Terminal (Left):**

```
technical_details.txt U package.json U JS router.js U
nodeRepo > node-js-ru-july-2022 > exams > exam-andreea-popa > server > routes > JS router.js ...
24  /**
25  * @description for add pieces
26
27  Monitor in production:
28  $ pm2 monitor
29
30  Make pm2 auto-boot at server restart:
31  $ pm2 startup
32
33  To go further checkout:
34  http://pm2.io/
35
36 + exam-andreea-popa git:(exam-andreea-popa) ✘ pm2 logs
37 [TAILING] Tailing last 15 lines for [all] processes (change the value with --lines option)
38 /Users/andreea.popa/.pm2/pm2.log last 15 lines:
39
PM2 : 2022-08-04T17:53:39: PM2 log: PM2 version          : 5.2.0
PM2 : 2022-08-04T17:53:39: PM2 log: Node.js version      : 16.16.0
PM2 : 2022-08-04T17:53:39: PM2 log: Current arch        : arm64
PM2 : 2022-08-04T17:53:39: PM2 log: PM2 home           : /Users/andreea.popa/.pm2
PM2 : 2022-08-04T17:53:39: PM2 log: PM2 PID file        : /Users/andreea.popa/.pm2/pid
PM2 : 2022-08-04T17:53:39: PM2 log: RPC socket file     : /Users/andreea.popa/.pm2/rpc.sock
PM2 : 2022-08-04T17:53:39: PM2 log: BUC socket file    : /Users/andreea.popa/.pm2/pub.sock
PM2 : 2022-08-04T17:53:39: PM2 log: Application log path: /Users/andreea.popa/.pm2/logs
PM2 : 2022-08-04T17:53:39: PM2 log: Worker Interval     : 30000
PM2 : 2022-08-04T17:53:39: PM2 log: Process dump file   : /Users/andreea.popa/.pm2/dump.pm2
PM2 : 2022-08-04T17:53:39: PM2 log: Concurrent actions  : 2
PM2 : 2022-08-04T17:53:39: PM2 log: SIGTERM timeout     : 1600
PM2 : 2022-08-04T17:53:39: PM2 log: =====
PM2 : 2022-08-04T17:53:39: PM2 log: App [server:0] starting in -fork mode-
PM2 : 2022-08-04T17:53:39: PM2 log: App [server:0] online
40
41 /Users/andreea.popa/.pm2/logs/server-error.log last 15 lines:
42 /Users/andreea.popa/.pm2/logs/server-out.log last 15 lines:
43 0|server | {"level": "info", "message": "Server running on http://localhost:3000", "timestamp": "2022-08-04T14:53:40.194Z"}
44 0|server | MongoDB connected: ac-dsf2rpe-shard-00.0.abbzmf6.mongodb.net
45 0|server | GET /api/pieces 200 810 - 54.608 ms
46 0|server | GET / 200 6189 - 128.646 ms
47 0|server | GET /api/pieces 200 810 - 41.115 ms
48 0|server | GET / 200 6189 - 57.100 ms
49 0|server | GET /api/pieces 200 810 - 36.500 ms
50 0|server | GET /api/pieces 200 810 - 45.803 ms
51 0|server | GET /api/pieces 200 810 - 96.634 ms
52 0|server | GET /api/pieces 200 810 - 40.594 ms
```

**Postman (Right):**

Pieces / Show Pieces

GET http://localhost:3000/api/pieces

Params Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
_id: "62ea5db654d6e6c9b8ba8595",
pieceType: "AC Motor",
pieceNumber: "1789",
descript: "AC Motor for EV",
__v: 0
},
{
_id: "62ea6e343b13ca992ac6d3a3",
pieceType: "PCU",
pieceNumber: "145".
```

Online Find and Replace Console

# Development - pm2

technical\_details.txt — NodeCourse

technical\_details.txt U package.json U JS router.js U

nodeRepo > node-js-ru-july-2022 > exams > exam-andreea-popa > technical\_details.txt

```
75
76 pm2 start server.js -> start server, app is online
77 pm2 logs -> start server
78 pm2 stop server -> stop server, app is offline
79
```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL ... zsh - exam-andreea-popa +

PM2 | 2022-08-04T17:53:39: PM2 log: Process dump file : /Users/andreea.popa/.pm2/dump.pm2
PM2 | 2022-08-04T17:53:39: PM2 log: Concurrent actions : 2
PM2 | 2022-08-04T17:53:39: PM2 log: SIGTERM timeout : 1600
PM2 | 2022-08-04T17:53:39: PM2 log: App [server:0] starting in -fork mode-
PM2 | 2022-08-04T17:53:39: PM2 log: App [server:0] online
PM2 | 2022-08-04T18:01:44: PM2 log: Stopping app:server:0
PM2 | 2022-08-04T18:01:44: PM2 log: App [server:0] exited with code [0] via signal [SIGINT]
PM2 | 2022-08-04T18:01:45: PM2 log: pid=99177 msg=process Killed
PM2 | 2022-08-04T18:04:41: PM2 log: App [server:0] starting in -fork mode-
PM2 | 2022-08-04T18:04:41: PM2 log: App [server:0] online

/Users/andreea.popa/.pm2/logs/server-error.log last 15 lines:
/Users/andreea.popa/.pm2/logs/server-out.log last 15 lines:
0|server | {"level": "info", "message": "Server running on http://localhost:3000", "timestamp": "2022-08-04T14:53:40.194Z"}
0|server | MongoDB connected: ac-dsf2rpe-shard-00-00.abbzmf6.mongodb.net
0|server | GET /api/pieces 200 810 - 54.608 ms
0|server | GET / 200 6189 - 128.646 ms
0|server | GET /api/pieces 200 810 - 41.115 ms
0|server | GET / 200 6189 - 57.100 ms
0|server | GET /api/pieces 200 810 - 36.500 ms
0|server | GET / 200 6189 - 45.803 ms
0|server | GET /api/pieces 200 810 - 96.634 ms
0|server | GET /api/pieces 200 810 - 40.594 ms
0|server | {"level": "info", "message": "Server running on http://localhost:3000", "timestamp": "2022-08-04T15:04:41.812Z"}
0|server | MongoDB connected: ac-dsf2rpe-shard-00-00.abbzmf6.mongodb.net

^C → exam-andreea-popa git:(exam-andreea-popa) x pm2 stop server  
[PM2] Applying action stopProcessId on app [server](ids: [ 0 ])  
[PM2] [server](0) ✓

id	name	mode	o	status	cpu	memory
0	server	fork	0	stopped	0%	0b

+ exam-andreea-popa git:(exam-andreea-popa) x

EXPLORER ...

OPEN EDITORS

technical\_details.txt U package.json U JS router.js nodeRe... U

NODECOURSE

JS router.js U services U JS render.js U test JS router.test.js U views JS a\_db.js U

TERMINAL ... zsh - exam-andreea-popa +

db.net

0|server | GET /api/pieces 200 810 - 54.608 ms
0|server | GET / 200 6189 - 128.646 ms
0|server | GET /api/pieces 200 810 - 41.115 ms
0|server | GET / 200 6189 - 57.100 ms
0|server | GET /api/pieces 200 810 - 36.500 ms
0|server | GET / 200 6189 - 45.803 ms
0|server | GET /api/pieces 200 810 - 96.634 ms
0|server | GET /api/pieces 200 810 - 40.594 ms
0|server | {"level": "info", "message": "Server running on http://localhost:3000", "timestamp": "2022-08-04T15:04:41.812Z"}
0|server | MongoDB connected: ac-dsf2rpe-shard-00-00.abbzmf6.mongodb.net

^C → exam-andreea-popa git:(exam-andreea-popa) x pm2 stop server  
[PM2] Applying action stopProcessId on app [server](ids: [ 0 ])  
[PM2] [server](0) ✓

id	name	mode	o	status	cpu
0	server	fork	0	stopped	0%

→ exam-andreea-popa git:(exam-andreea-popa) x pm2 ls

id	name	mode	o	status	cpu
0	server	fork	0	stopped	0%

→ exam-andreea-popa git:(exam-andreea-popa) x

0 0 8 Connect I F Plain Text O Port: 5500 ▲ 6 Spell ↗

Could not send request

Error: connect ECONNREFUSED 127.0.0.1:3000 | View in Col

EXPLORER ...

OPEN EDITORS

technical\_details.txt U config.env U package.json U

nodeRepo > node-js-ru-july-2022 > exams > exam-andreea-popa > technical\_details.txt

```
1 for the client-side of the app
2
3 npm installed dependencies:
4 - express - to develop the node app
5 - morgan - log a message at every request
6 - nodemon - to restart automatically the server
7 - ejs - to create dynamic html
8 - body-parser - serialize the data using {body}
9 - dotenv - separate the secret data from the source-code - avoid sharing database login credential with other people, for example
10 - mongoose - connect mongodb
11 - axios - easy to make req in express app
12
13 - express-validator - lib that allows us to validate things in app
14 - bcrypt - library for hashing text (password at authentication part)
15
16 folders:
17
18 assets:
19 - css - css files
20 - img - images
21 - js - js files
22
23 views: html files, default folder for html files
24
25 server: here are created services as model, mongodb connection
26 - MVC - model view controller design pattern
27 - model - mongodb data: data validation and processing, creating mongo scheme
28 - controller - user req, creating functions
29
30
31
32 views -> include - folder for partial files; they are represented with _
33 i erased the html files after implanting the functionality to .ejs files
34
35
36
37
38 For the server-side of the app:
39 - creating routes, services
40 - mongodb -in cloud->
41 database access: user - admin, password - admin
```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL GITLENS: VISUAL FILE HISTORY SQL CONSOLE: MESSAGES

node - exam-andreea-popa + ×

GET /js/index.js 304 -- 0.635 ms

EXPLORER ... nodeRepo > node-js-ru-july-2022 > exams > exam-andreea-popa > technical\_details.txt

OPEN EDITORS

technical\_details.txt config.env package.json

nodeRepo > node\_js\_ru-july-2022 > exams > exam-andreea-popa > technical\_details.txt

```
37
38  For the server-side of the app:
39  - creating routes, services
40  - mongodb -in cloud->
41      database access: user - admin, password - admin
42      ip access: 5.2.198.48/32
43      config.env file: mongodb+srv://admin:<password>@cluster0.abbzmf6.mongodb.net/?retryWrites=true&w=majority
44          | mongodb+srv://admin:admin@cluster0.abbzmf6.mongodb.net/?retryWrites=true&w=majority
45
46
47  - in model folder -> model.js - db schema
48
49  API route in server -> routes -> router.js
50
51  in services folder -> render.js - connect the db in order to get the ejs files updated
52
53  in views-> include -> _show.ejs - all the data of pieces
54  _show.ejs is called in index.ejs, just like _header.ejs and _footer.ejs
55  in routes -> router.js we have add-piece and update-piece
56
57  we use jquery library (for DOM operations) and link it in _footer.ejs
58  -> by pressing submit in add_piece.ejs -> the data is stored in db, too
59  we have mongoose
60
61
62  For auth:
63
64
65  JWT - JSON Web Token contains info about who the client is
66  - goes from client - server
67  - not for sensitive info, as password, bcs it can be easily decoded using jwt.io
68  - to ensure the user is really authenticated
69
70  to restrictionate access between public and private-> middleware
71  JSON token - inside Header <-> authorized to access file
72
73
74  pm2 start server.js -> start server, app is online
75  pm2 logs -> start server
76  pm2 stop server -> stop server, app is offline
77  pm2 ls -> lists the processes
```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL GITLENS: VISUAL FILE HISTORY SQL CONSOLE: MESSAGES

node - exam-andreea-popa + ×

GET /js/index.js 304 -- 0.635 ms