



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

**ΠΡΟΗΓΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΠΛΗΡΟΦΟΡΙΚΗΣ - ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ
ΚΑΙ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ**

ΠΡΟΗΓΜΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

FULL STACK JAVASCRIPT FRAMEWORKS

ΑΝΑΦΟΡΑ ΤΕΛΙΚΗΣ ΕΡΓΑΣΙΑΣ

ΜΕΤΑΠΤΥΧΙΑΚΟΣ ΦΟΙΤΗΤΗΣ:

- **ΝΙΚΟΛΑΣ ΠΑΤΕΡΑΣ – ΜΠΣΠ21043**

Πειραιάς, Φεβρουάριος 2022

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ.....	2
1. ΕΙΣΑΓΩΓΗ	3
2. ΛΕΙΤΟΥΡΓΙΕΣ ΚΑΙ REQUESTS	4
2.1. CREATE	4
2.2. READ.....	7
2.3. UPDATE	11
2.4. DELETE.....	13
3. ΒΙΒΛΙΟΓΡΑΦΙΑ ΚΑΙ ΠΗΓΕΣ	14
4. ΒΙΒΛΙΟΘΗΚΕΣ & ΕΡΓΑΛΕΙΑ	15

1. ΕΙΣΑΓΩΓΗ

Στην παρούσα εργασία υλοποιήθηκε ένα **CRUD** μοντέλο σε **NodeJS** συμπεριλαμβάνοντας βιβλιοθήκες σαν τη **mongoose** και **express** μαζί με την βοήθεια της **MongoDB** για τα δεδομένα.

Το Node.js είναι ένα backend περιβάλλον εκτέλεσης **JavaScript** ανοιχτού κώδικα, πολλαπλών πλατφορμών, που εκτελείται σε «V8 Engine» και εκτελεί κώδικα JavaScript εκτός προγράμματος περιήγησης Ιστού. Το Node.js επιτρέπει στους προγραμματιστές να χρησιμοποιούν JavaScript για τη σύνταξη εργαλείων γραμμής εντολών και για ενέργειες από την πλευρά του διακομιστή που εκτελούνται από την πλευρά του διακομιστή για την παραγωγή δυναμικού περιεχομένου ιστοσελίδας προτού σταλεί η σελίδα στο πρόγραμμα περιήγησης ιστού του χρήστη.

Γιατί **MongoDB** και όχι **MySQL**; Αρχικά, τα αντικείμενα στη Mongo είναι **JSON**, οπότε είναι τόσο απλό για να μετακινηθούν από JavaScript σε JSON και πίσω. Επίσης, η εγκατάσταση της είναι παρά πολύ απλή και μας δίνει άπειρες δυνατότητες από το UI της μαζί με το MongoDB Atlas που είναι μια υπηρεσία βάσης δεδομένων πολλαπλού-νέφους και απλοποιεί την ανάπτυξη και τη διαχείριση των βάσεων δεδομένων, ενώ προσφέρει την ευελιξία που χρειάζεται για τη δημιουργία ανθεκτικών και αποδοτικών παγκόσμιων εφαρμογών στους παρόχους cloud.

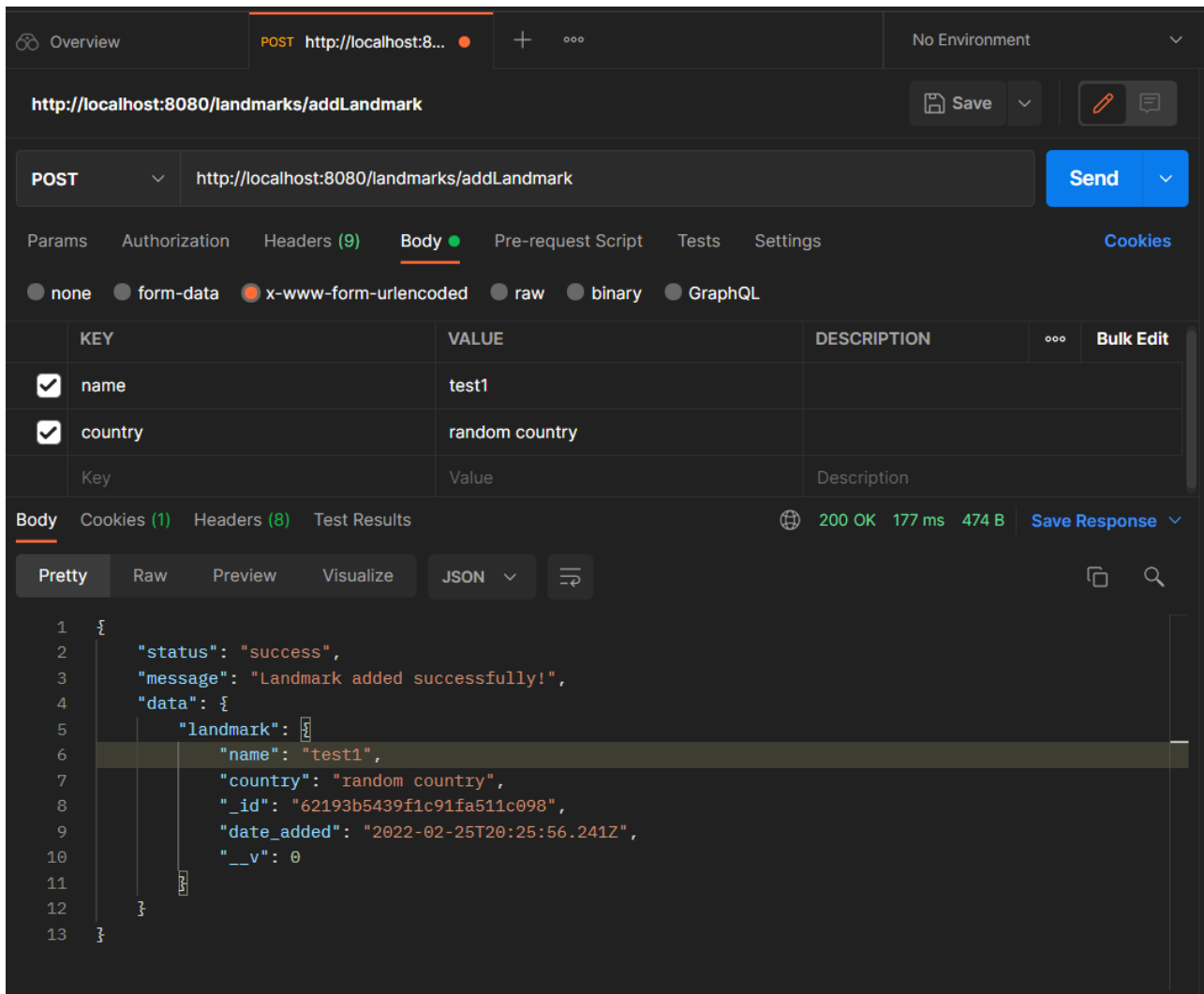
Τέλος, χρησιμοποιήθηκε version control για εύκολη διαχείριση του project,

<https://github.com/IHateSyntaxErrors/Points-of-Interest-API>

2. ΛΕΙΤΟΥΡΓΙΕΣ ΚΑΙ REQUESTS

2.1. CREATE

A. Αποθήκευση ενός νέου εγγράφου στη βάση και η επιστροφή του σωστού status στον χρήστη για τα σημεία ενδιαφέροντος:



The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/landmarks/addLandmark
- Body Type:** x-www-form-urlencoded
- Body Data:**

KEY	VALUE	DESCRIPTION
name	test1	
country	random country	
- Response:** 200 OK, 177 ms, 474 B
- Response Body (JSON):**

```
{  "status": "success",  "message": "Landmark added successfully!",  "data": {    "landmark": {      "name": "test1",      "country": "random country",      "_id": "62193b5439f1c91fa511c098",      "date_added": "2022-02-25T20:25:56.241Z",      "__v": 0    }  }}
```

B. Αποθήκευση ενός νέου εγγράφου στη βάση και η επιστροφή του σωστού status στον χρήστη για τους χρήστες (Εάν το email είναι κρατημένο):

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/users/addUser
- Body Type:** x-www-form-urlencoded
- Body Data:**

KEY	VALUE
first_name	Nick
last_name	Pat
email	test8@hotmail.com
password	123456
- Status:** 400 Bad Request
- Response Time:** 131 ms
- Response Size:** 576 B
- Response Body (JSON):**

```
{  "status": "fail",  "message": "Email is already in use!",  "data": {    "user": {      "_id": "621a3f458c6a32474d5b746a",      "first_name": "Nick",      "last_name": "Pat",      "email": "test8@hotmail.com",      "password": "$2a$10$EeuX3yVSBcnhMwKbHxNd807iHyA7RNH3mSbfuj66mxKpAacwd7jW6",      "date_registered": "2022-02-26T14:55:01.708Z",      "__v": 0    }  }}
```

C. Αποθήκευση ενός νέου εγγράφου στη βάση και η επιστροφή του σωστού status στον χρήστη για τους χρήστες:

The screenshot shows a REST client interface with the following details:

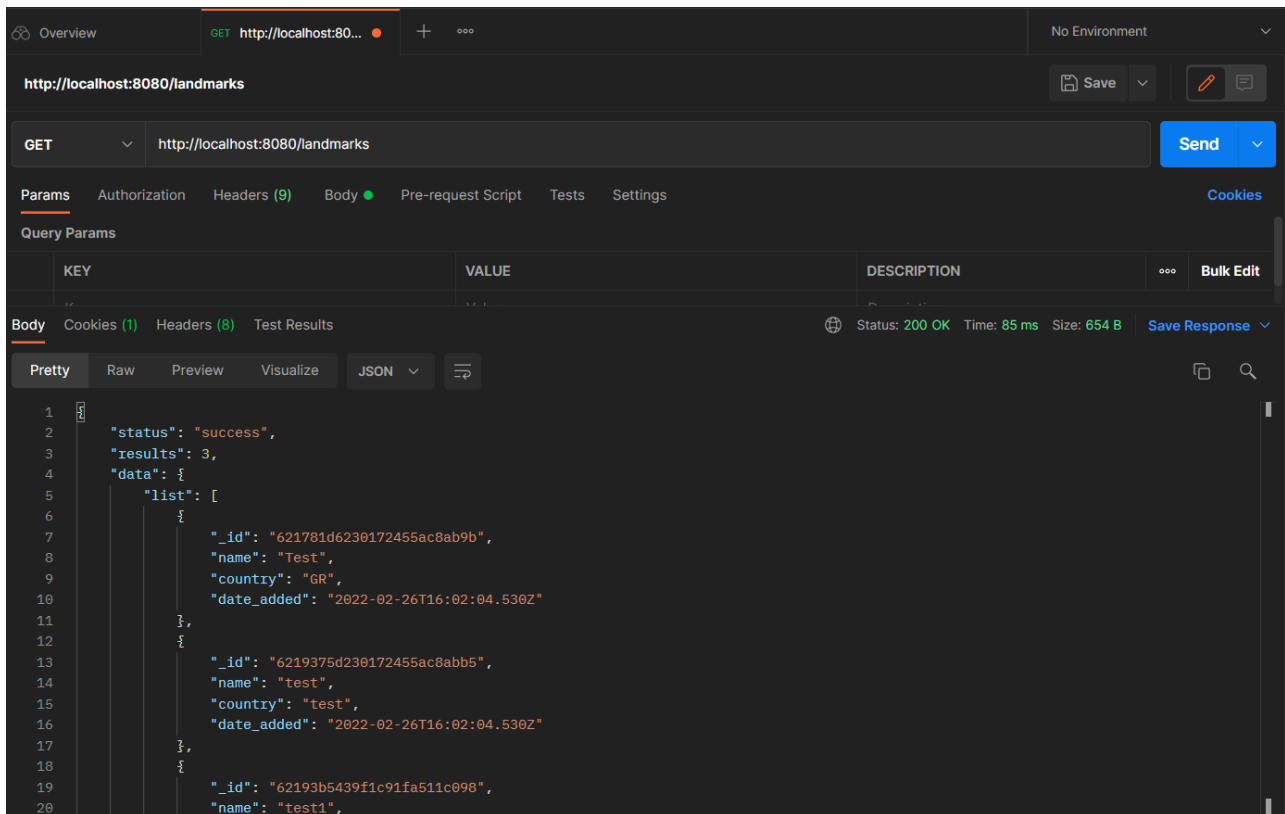
- Method:** POST
- URL:** http://localhost:8080/users/addUser
- Body Type:** x-www-form-urlencoded
- Form Data:**

KEY	VALUE
first_name	Nick
last_name	Pat
email	test8@hotmail.com
password	123456
- Response:** 200 OK, 520 ms, 570 B
- Response Body (JSON):**

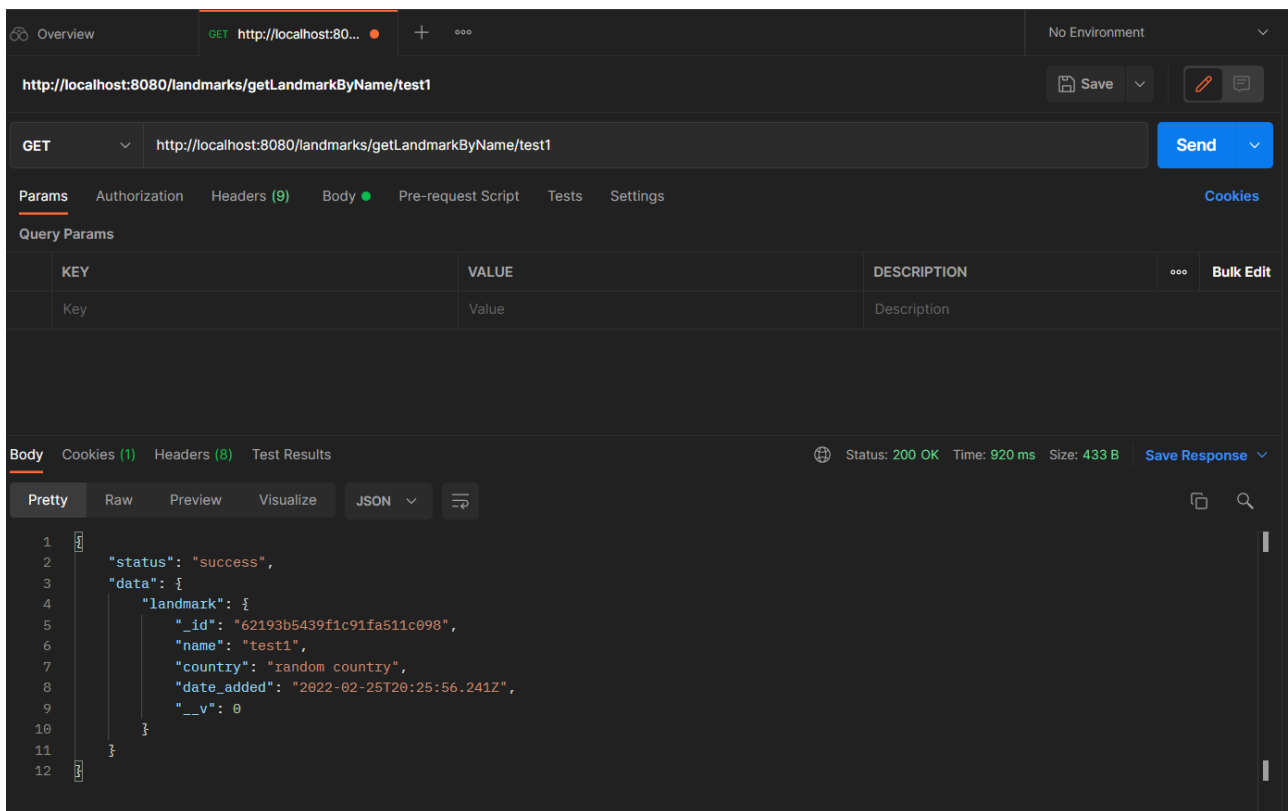
```
{  "status": "success",  "message": "User added successfully!",  "data": {    "user": {      "first_name": "Nick",      "last_name": "Pat",      "email": "test8@hotmail.com",      "password": "$2a$10$EeuX3yVSBcnhMwKbHxNd807iHyA7RNH3mSbfuj66mxKpAacwd7jW6",      "_id": "621a3f458c6a32474d5b746a",      "date_registered": "2022-02-26T14:55:01.708Z",      "__v": 0    }  }  }
```

2.2. READ

A. Να καθίσταται δυνατή η ανάγνωση και η επιστροφή (με το σωστό request) Όλων των σημείων ενδιαφέροντος από τη βάση:



Β. Ένα σημείο ενδιαφέροντος με βάση το όνομα που περνάει ως παράμετρος στο request:



The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/landmarks/getLandmarkByName/test1`
- Method:** GET
- Status:** 200 OK
- Time:** 920 ms
- Size:** 433 B

The response body is displayed in JSON format:

```
1 {
2   "status": "success",
3   "data": {
4     "landmark": {
5       "_id": "62193b5439f1c91fa511c098",
6       "name": "test1",
7       "country": "random country",
8       "date_added": "2022-02-25T20:25:56.241Z",
9       "__v": 0
10    }
11  }
12 }
```


C. Όλοι οι χρήστες από τη βάση:

```
1  {
2    "status": "success",
3    "results": 2,
4    "data": {
5      "list": [
6        {
7          "_id": "6218a3c7238172455ac8ab9e",
8          "first_name": "Nick",
9          "last_name": "Pat",
10         "email": "test@hotmail.com",
11         "date_registered": "2022-02-26T16:08:12.881Z"
12       },
13       {
14         "_id": "621a3d6687b21314e2e7be13",
15         "first_name": "Nick",
16         "last_name": "Pat",
17         "email": "test2@hotmail.com",
18         "password": "123456",
19         "date_registered": "2022-02-26T14:47:02.199Z",
20         "__v": 0
21       }
22     ]
23   }
24 }
```

```
11  "date_registered": "2022-02-26T16:08:12.881Z",
12  },
13  {
14    "_id": "621a3ddff7bcb46a3dc635d7e",
15    "first_name": "Nick",
16    "last_name": "Pat",
17    "email": "test3@hotmail.com",
18    "password": "123456",
19    "date_registered": "2022-02-26T14:49:03.034Z",
20    "__v": 0
21  },
22  {
23    "_id": "621a3ddff7bcb46a3dc635d7e",
24    "first_name": "Nick",
25    "last_name": "Pat",
26    "email": "test3@hotmail.com",
27    "password": "123456",
28    "date_registered": "2022-02-26T14:49:03.034Z",
29    "__v": 0
30  },
31  ]
32 }
```

D. Ένας χρήστης με βάση το email (που θα δίνεται ως παράμετρος στο request), θα εμφανίζεται μόνο το όνομα του χρήστη στην οθόνη:

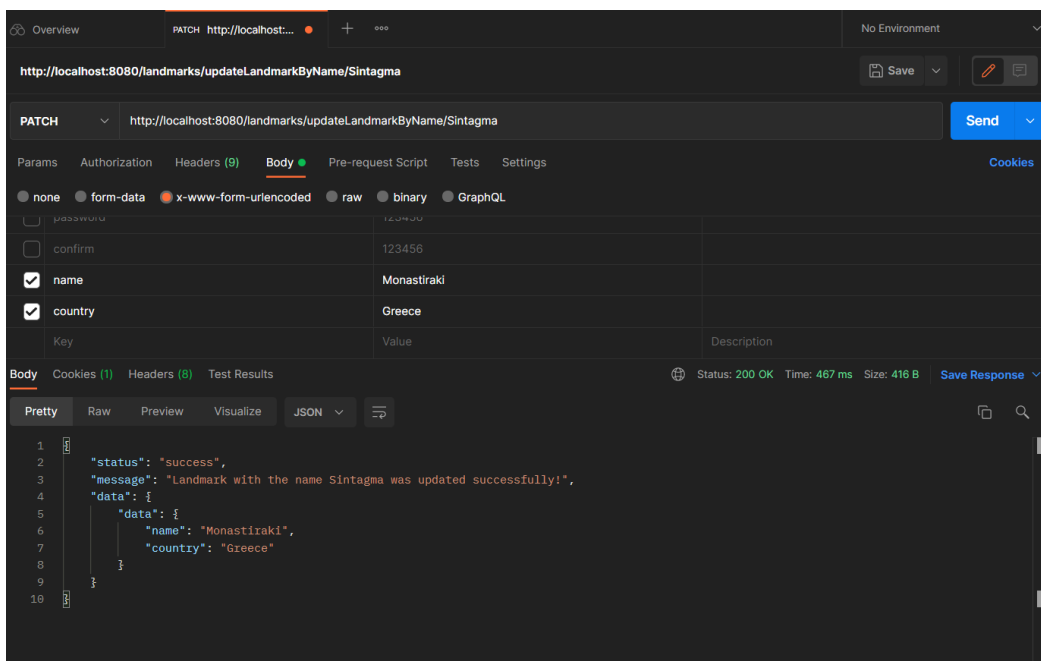
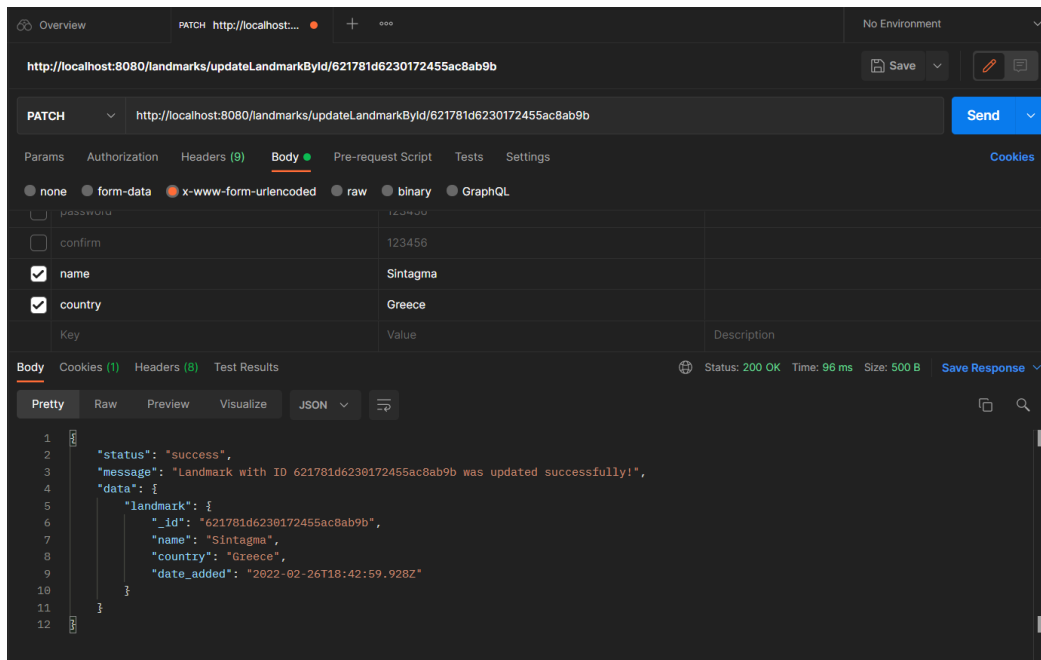
The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/users/getUserNameByEmail/test@hotmail.com`
- Method:** GET
- Params:** Query Params table with one entry: Key: `test`, Value: `test@hotmail.com`.
- Body:** Pretty view of the response JSON:

```
1 {
2   "status": "success",
3   "data": {
4     "first_name": "Nick",
5     "last_name": "Pat"
6   }
7 }
```
- Status:** 200 OK, Time: 106 ms, Size: 334 B

2.3. UPDATE

Α. Θα μπορεί να γίνει update ενός σημείου ενδιαφέροντος με βάση το όνομα, ή δίνοντας το id:



B. Θα μπορεί να γίνει update ενός χρήστη με βάση το email του:

The screenshot shows a REST client interface with a PATCH request to `http://localhost:8080/users/updateUserByEmail/test@hotmail.com`. The request body is in x-www-form-urlencoded format with the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> first_name	Nikolas	
<input checked="" type="checkbox"/> last_name	Pateras	
<input checked="" type="checkbox"/> email	test_2@hotmail.com	
<input checked="" type="checkbox"/> password	123456	

The response status is 200 OK, with a time of 104 ms and a size of 475 B. The response body is in JSON format:

```
1 {
2   "status": "success",
3   "message": "User with the email test@hotmail.com was updated successfully!",
4   "data": {
5     "data": {
6       "first_name": "Nikolas",
7       "last_name": "Pateras",
8       "email": "test_2@hotmail.com",
9       "password": "123456"
10    }
11  }
12 }
```

2.4. DELETE

Α. Θα μπορεί να γίνει διαγραφή ενός χρήστη / σημείου ενδιαφέροντος

Overview **DEL** http://localhost:80... + ... No Environment

http://localhost:8080/users/deleteUser/621a3f0100ca3408e28531d6 Save

DELETE http://localhost:8080/users/deleteUser/621a3f0100ca3408e28531d6 Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

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

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> first_name	Nikolas			
<input checked="" type="checkbox"/> last_name	Pateras			
<input checked="" type="checkbox"/> email	test_2@hotmail.com			
<input checked="" type="checkbox"/> password	123456			

Body Cookies (1) Headers (8) Test Results Status: 200 OK Time: 102 ms Size: 363 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "success",
3   "message": "User with ID 621a3f0100ca3408e28531d6 was deleted successfully!"
4 }
```

Overview **DEL** http://localhost:80... + ... No Environment

http://localhost:8080/landmarks/deleteLandmark/6219375d230172455ac8abb5 Save

DELETE http://localhost:8080/landmarks/deleteLandmark/6219375d230172455ac8abb5 Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

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

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> first_name	Nikolas			
<input checked="" type="checkbox"/> last_name	Pateras			
<input checked="" type="checkbox"/> email	test_2@hotmail.com			
<input checked="" type="checkbox"/> password	123456			

Body Cookies (1) Headers (8) Test Results Status: 200 OK Time: 75 ms Size: 368 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "success",
3   "message": "Landmark with ID 6219375d230172455ac8abb5 was deleted successfully!"
4 }
```

3. ΒΙΒΛΙΟΓΡΑΦΙΑ ΚΑΙ ΠΗΓΕΣ

1. Διαφάνειες μαθήματος (e-class).
2. <https://blog.logrocket.com/building-structuring-node-js-mvc-application/>
3. <https://www.youtube.com/watch?v=RU9Vzu9uDYU>
4. <https://www.mongodb.com/basics/clusters/mongodb-cluster-setup>
5. <https://www.udemy.com/course/nodejs-the-complete-guide/>
6. <https://www.mongodb.com/languages/express-mongodb-rest-api-tutorial>
7. <https://github.com/Musawirkhann/node-express-mongodb-mvc-ejs-crud>
8. <https://github.com/bigpreshy/mvc/>
9. https://github.com/bradtraversy/node_passport_login/
10. <https://github.com/bezkoder/node-js-jwt-auth-mongodb/>

4. ΒΙΒΛΙΟΘΗΚΕΣ & ΕΡΓΑΛΕΙΑ

Όνομα	Έκδοση	Τύπος
Visual Studio Code	1.64.2	Εργαλείο
MongoDB	5.0.6	Εργαλείο
MongoDB Compass	1.30.1	Εργαλείο
dotenv	16.0.0	Βιβλιοθήκη
jsonwebtoken	8.5.1	Βιβλιοθήκη
mongoose	6.2.2	Βιβλιοθήκη
express	4.17.3	Βιβλιοθήκη
express-async-errors	3.1.1	Βιβλιοθήκη
joi	17.6.0	Βιβλιοθήκη
Joi-objectid	4.0.2	Βιβλιοθήκη
bcryptjs	2.4.3	Βιβλιοθήκη
cors	2.8.5	Βιβλιοθήκη
body-parser	1.19.2	Βιβλιοθήκη
winston	3.6.0	Βιβλιοθήκη
nodemon	2.0.15	Βιβλιοθήκη

Συμπληρωματικά, το αρχείο `packages.json` που βρίσκεται στο directory μας, είναι η καρδιά οποιουδήποτε NodeJS project. Καταγράφει σημαντικά metadata σχετικά με ένα project που απαιτείται πριν από τη δημοσίευση στο NPM, και επίσης ορίζει λειτουργικά χαρακτηριστικά ενός project που χρησιμοποιεί το npm για την εγκατάσταση εξαρτήσεων, την εκτέλεση σεναρίων και την αναγνώριση του σημείου εισόδου στο πακέτο μας. Με άλλα λόγια, δημιουργείται αυτόματα από το NPM με όλες τις βιβλιοθήκες που χρησιμοποιούμε συμπεριλαμβάνοντας τις εκδόσεις αυτών.