



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ  
UNIVERSITY OF PIRAEUS

# Απαλλακτική Εργασία

## Full Stack JavaScript Frameworks

### Smart-Tourism-App

Παναγιώτης Παπακώστας



# Περιγραμμα παρουσίασης

1. Εισαγωγή
2. Σχεδίαση και Υλοποίηση
3. Λειτουργικότητα
4. Παρατηρήσεις



# Εισαγωγή

- Η εφαρμογή **Smart-Tourism-App** αποτελεί μια εφαρμογή η οποία επιτρέπει στους χρήστες (τουρίστες ή κατοίκους) να συμβάλλουν στον έξυπνο τουρισμό μιας πόλης, αποθηκεύοντας σε μια βάση δεδομένων τα αγαπημένα τους μέρη και παρέχοντας πληροφορίες για τοπικά αξιοθέατα.
- Υλοποιήθηκε ένα API, το οποίο υποστηρίζει τις βασικές λειτουργίες διαχείρισης δεδομένων (Create, Read, Update, Delete - CRUD).
- Το API αυτό διασφαλίζει την ομαλή λειτουργία του εκάστοτε συστήματος, επιτρέποντας τη διαχείριση των χρηστών και των σημείων ενδιαφέροντος (POI).
- Παρακάτω παρουσιάζεται η σχεδίαση της εφαρμογής καθώς και ενδεικτικά screenshots που αφορούν την υλοποίηση και τη λειτουργικότητά της.

# Σχεδίαση και Υλοποίηση

- Το project αναπτύχθηκε σε **Express.js**.
- Είναι οργανωμένο σε **MVC αρχιτεκτονική**.
- Για την αποθήκευση των δεδομένων, χρησιμοποιείται NoSQL Βάση Δεδομένων **MongoDB** σε συνδυασμό με τη βιβλιοθήκη **Mongoose**.

```
✓ SMART-TOURISM-APP
  ✓ controllers
    JS poiController.js
    JS userController.js
  ✓ middleware
    JS img-upload.js
  ✓ models
    JS poiModel.js
    JS userModel.js
  > node_modules
  > public\img\pois
  ✓ routes
    JS poiRoutes.js
    JS userRoutes.js
  JS app.js
  ⚙ config.ENV
  {} package-lock.json
  {} package.json
  JS server.js
```



# Σχεδίαση και Υλοποίηση – Λειτουργίες του API

Υποστηρίζονται οι παρακάτω λειτουργίες:

- Εγγραφή Χρήστη
- Προσθήκη Σημείου Ενδιαφέροντος (POI)
- Ανάγνωση Πληροφοριών Χρήστη
- Ανακάλυψη Σημείων Ενδιαφέροντος (POI)
- Ενημέρωση Πληροφοριών Χρήστη/ Σημείων ενδιαφέροντος
- Διαγραφή Χρήστη ή POI

# Σχεδίαση και Υλοποίηση – Models

► User

```
// Schema
const userSchema = new mongoose.Schema({
  name: {
    type:String,
    required: [true, "The name is a mandatory field!"],
  },
  age: {
    type:String,
    required: [true, "The age is a mandatory field!"],
  },
  email: {
    type:String,
    lowercase: true,
    required: [true, "The e-mail is a mandatory field!"],
    unique: true
  }
});
```

- Τα πεδία name, age, email είναι υποχρεωτικά
- Το email αποθηκεύεται με μικρά γράμματα στη βάση, ανεξάρτητα από την είσοδο του χρήστη
- Το email είναι μοναδικό (ελέγχεται και παρακάτω με Middleware)

► POI

```
// Schema
const poiSchema = new mongoose.Schema({
  name: String,
  category: String,
  address: String,
  geoCoordinates: {
    type: {
      type: String,
      enum: ['Point'],
    },
    coordinates: {
      type: [Number],
    }
  },
  photo: {
    type: String,
    default: 'default.png'
  }
});
```

# Σχεδίαση και Υλοποίηση – Λειτουργίες API – User Routes

## ► User Routes

```
1 // Import express
2 const express = require('express');
3 // Import userController module
4 const userController = require(`${__dirname}/../controllers/userController`);
5
6 const routes = express.Router();
7
8 // User Routes
9 routes.route('/')
10 .get(userController.getAllUsers)
11 .post(userController.checkUserAge, userController.checkUserEmail, userController.createUser)
12
13 routes.route('/:email')
14 .get(userController.getUserByEmail)
15 .patch(userController.checkUserAge, userController.updateUserByEmail)
16 .delete(userController.deleteUserByEmail)
17
18 module.exports = routes;
```

# Σχεδίαση και Υλοποίηση – Λειτουργίες API – POI Routes

## ► POI Routes

```
1  // Import express
2  const express = require('express');
3  // Import multer configured middleware
4  const fileUpload = require('../middleware/img-upload');
5  // Import poiController module
6  const poiController = require(`${__dirname}/../controllers/poiController`);
7
8  const routes = express.Router();
9
10 // POI Routes
11 routes.route('/')
12   .get(poiController.getAllPois)
13   .post(fileUpload, poiController.createPoi)
14
15 routes.route('/getPoi')
16   .get(poiController.getPoi)
17
18 routes.route('/:id')
19   .patch(fileUpload, poiController.updatePoiById)
20   .delete(poiController.deletePoiById)
21
22
23 routes.route('/image/:id')
24   .patch(fileUpload, poiController.updatePoiById)
25
26 module.exports = routes;
```



# Σχεδίαση και Υλοποίηση – Λειτουργίες API – User Controller

## ► Create User

```
// Create a new user
exports.createUser = async (req, res) => {
  try {
    const newUser = await User.create(req.body);

    res.status(201).json({
      status: 'success',
      data: {
        tour: newUser
      }
    });
  } catch (err) {
    res.status(400).json({
      status: 'fail',
      message: err
    });
  }
};
```

# Σχεδίαση και Υλοποίηση – Λειτουργίες API - User Controller

## ► Get All Users

```
// Get all users
exports.getAllUsers = async (req, res) => {
  try {
    const users = await User.find();

    res.status(200).json({
      status: 'success',
      results: users.length,
      data: {
        users
      }
    });
  } catch (err) {
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

## ► Get User By Email (εμφάνιση μόνο του ονόματος)

```
// Get a user by email and show only their name
exports.getUserByEmail = async (req, res) => {
  try {
    const user = await User.findOne({email: req.params.email}, { name: 1 });

    if (!user) {
      return res.status(404).json({
        status: 'fail',
        message: 'User not found'
      });
    }

    res.status(200).json({
      status: 'success',
      data: {
        user
      }
    });
  } catch (err) {
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

# Σχεδίαση και Υλοποίηση – Λειτουργίες

## API - User Controller

### ► Update User by Email

```
// Update a user by email
exports.updateUserByEmail = async (req, res) => {
  try {
    const user = await User.findOneAndUpdate({email: req.params.email}, req.body, {
      new: true,
    });

    if (!user) {
      return res.status(404).json({
        status: 'fail',
        message: 'User not found'
      });
    }

    res.status(200).json({
      status: 'success',
      data: {
        user
      }
    });
  } catch (err) {
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

### ► Delete User By Email

```
// Delete a user by email
exports.deleteUserByEmail = async (req, res) => {
  try {
    const user = await User.findOneAndDelete({email: req.params.email});

    if (!user) {
      return res.status(404).json({
        status: 'fail',
        message: 'User not found'
      });
    }

    res.status(200).json({
      status: 'success',
      data: null
    });
  } catch (err) {
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

# Σχεδίαση και Υλοποίηση – Λειτουργίες API - Χρήση Middleware

- Έλεγχος ηλικίας χρήστη

```
// Middleware to check if the user's age is 18 or older
exports.checkUserAge = (req, res, next) => {
  var age = req.body.age * 1;

  if (age < 18) {
    return res.status(404).json({
      status: 'fail',
      message: "User must be 18 years old or older!"
    });
  }
  next();
};
```

- Έλεγχος για υπάρχον e-mail

```
// Middleware to check if the user's email already exists
exports.checkUserEmail = async (req, res, next) => {
  const email = req.body.email;
  const user = await User.findOne({ email: email });

  if (user) {
    return res.status(404).json({
      status: 'fail',
      message: "User's email already exists!"
    });
  }
  next();
};
```

# Σχεδίαση και Υλοποίηση – Λειτουργίες API – POI Controller

## ► Create POI

```
// Create a new Point of Interest (POI)
exports.createPoi = async (req, res) => {
  try {

    var myBody = req.body;
    const data = { ...myBody };

    // If there's a file (photo) attached, update the 'photo' field
    if (req.file) {
      const photo = req.file.filename;
      data.photo = photo;
    }

    // Parse geoCoordinates from string to object if applicable
    if (data.geoCoordinates && typeof data.geoCoordinates === 'string') {
      data.geoCoordinates = JSON.parse(data.geoCoordinates);
    }

    // Create a new POI in the database
    const newPoi = await Poi.create(data);

    // Respond with success status and the created POI
    res.status(201).json({
      status: 'success',
      data: {
        tour: newPoi
      }
    });
  } catch (err) {
    // Respond with failure status and the error message
    res.status(400).json({
      status: 'fail',
      message: err
    });
  }
};
```

# Σχεδίαση και Υλοποίηση – Λειτουργίες API - POI Controller

## ► Get All POIS

```
// Get all Points of Interest (POIs)
exports.getAllPois = async (req, res) => {
  try {
    // Retrieve all POIs from the database
    const pois = await Poi.find();

    // Respond with success status and the list of POIs
    res.status(200).json({
      status: 'success',
      results: pois.length,
      data: {
        pois
      }
    });
  } catch (err) {
    // Respond with failure status and the error message
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

## ► Get POI By Any Parameter

```
// Get a specific Point of Interest (POI) by a given parameter
exports.getPoi = async (req, res) => {
  try {
    // Extract the parameter name and value from the request query
    const paramName = Object.keys(req.query)[0];
    const paramValue = req.query[paramName];

    // Check if required parameters are present
    if (!paramName || !paramValue) {
      return res.status(400).json({
        status: 'fail',
        message: 'Invalid request. Missing paramName or paramValue.'
      });
    }

    // Construct a query object based on the provided parameter
    const query = { [paramName]: paramValue };
    // Find the POI in the database
    const poi = await Poi.findOne(query);

    // If the POI is not found, respond with a 404 status and message
    if (!poi) {
      return res.status(404).json({
        status: 'fail',
        message: 'POI not found'
      });
    }

    // Respond with success status and the retrieved POI
    res.status(200).json({
      status: 'success',
      data: {
        poi
      }
    });
  } catch (err) {
    // Respond with failure status and the error message
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

# Σχεδίαση και Υλοποίηση – Λειτουργίες

## API - POI Controller

### ► Update POI by ID

```
// Update a specific Point of Interest (POI) by ID
exports.updatePoiById = async (req, res) => {
  try {
    // Extract the request body and create an updates object
    var myBody = req.body;
    const updates = { ...myBody };
    // If there's a file (photo) attached, update the 'photo' field
    if (req.file) {
      const photo = req.file.filename;
      updates.photo = photo;
    }

    // Parse geoCoordinates from string to object if applicable
    if (updates.geoCoordinates && typeof updates.geoCoordinates === 'string') {
      updates.geoCoordinates = JSON.parse(updates.geoCoordinates);
    }

    // Find and update the specified POI in the database
    const poi = await Poi.findOneAndUpdate({ _id: req.params.id }, updates, {
      new: true,
    });
    // If the POI is not found, respond with a 404 status and message
    if (!poi) {
      return res.status(404).json({
        status: 'fail',
        message: 'POI not found'
      });
    }
    // Respond with success status and the updated POI
    res.status(200).json({
      status: 'success',
      data: {
        poi
      }
    });
  } catch (err) {
    // Respond with failure status and the error message
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

### ► Delete POI By ID

```
// Delete a specific Point of Interest (POI) by ID
exports.deletePoiById = async (req, res) => {
  try {
    // Find and delete the specified POI in the database
    const poi = await Poi.findOneAndDelete({ _id: req.params.id });
    // If the POI is not found, respond with a 404 status and message
    if (!poi) {
      return res.status(404).json({
        status: 'fail',
        message: 'POI not found'
      });
    }
    // Respond with success status and null data (since the POI is deleted)
    res.status(200).json({
      status: 'success',
      data: null
    });
  } catch (err) {
    // Respond with failure status and the error message
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

# Σχεδίαση και Υλοποίηση – Λειτουργίες API – Upload Image related to POI

- Middleware to upload image

```
JS img-upload.js X
middleware > JS img-upload.js > ...
1  //import multer
2  const multer = require('multer');
3
4  const MIME_TYPE_MAP = {
5    'image/png': 'png',
6    'image/jpeg': 'jpeg',
7    'image/jpg': 'jpg',
8  };
9
10 const fileUpload = multer({
11   limits: 5000000,
12   storage: multer.diskStorage({
13     destination: (req, file, cb) => {
14       cb(null, `${__dirname}/../public/img/pois`);
15     },
16     filename: (req, file, cb) => {
17       cb(null, file.originalname);
18     }
19   }),
20   fileFilter: (req, file, cb) => {
21     const isValid = file.mimetype in MIME_TYPE_MAP;
22     let error = isValid ? null : new Error('Invalid mime type!');
23     cb(error, isValid);
24   }
25 });
26
27
28 module.exports = fileUpload.single('photo');
```



# Σχεδίαση και Υλοποίηση – Χρήση Μεταβλητών Περιβάλλοντος

- Χρησιμοποιούνται μεταβλητές περιβάλλοντος τόσο για τη σύνδεση στη ΒΔ όσο και για τον ορισμό της πόρτας που 'ακούει' ο server.
- Αρχείο config.ENV

```
config.ENV
1  PORT=8080
2  DATABASE=mongodb+srv://<USERNAME>:<PASSWORD>@cluster0.zzuwf6d.mongodb.net/Smart-Tourism?retryWrites=true&w=majority&appName=Cluster0
3  DATABASE_USERNAME=
4  DATABASE_PASSWORD=
5
```

- Αρχείο server.js

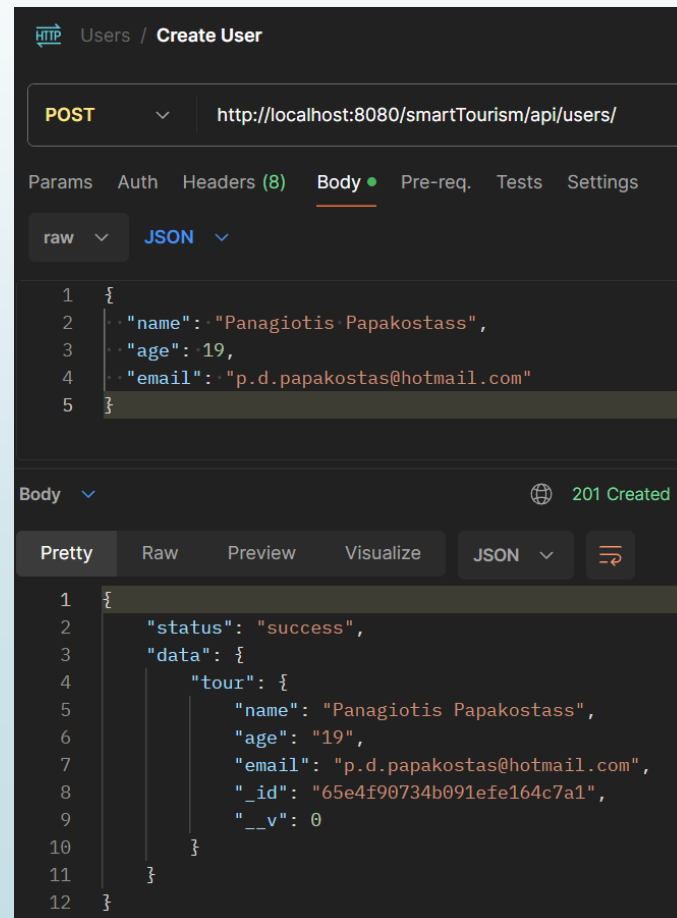
```
const mydb = process.env.DATABASE.replace('<USERNAME>', process.env.DATABASE_USERNAME).replace('<PASSWORD>', process.env.DATABASE_PASSWORD);
```

```
// Port
const port = process.env.PORT || 8443;

// Start server
app.listen(port, () => {
  console.log(`Listening on port ${port}...`)
});
```

# Λειτουργικότητα – Λειτουργίες User

## ➡ Create User



The screenshot displays a REST client interface for a POST request to `http://localhost:8080/smartTourism/api/users/`. The request body is a JSON object with user details. The response, shown in the 'Body' tab, is a 201 status code with a JSON body indicating success and providing the created user's details, including a unique ID and version number.

```
HTTP Users / Create User

POST http://localhost:8080/smartTourism/api/users/

Params Auth Headers (8) Body Pre-req. Tests Settings
raw JSON

1 {
2   "name": "Panagiotis Papakostass",
3   "age": 19,
4   "email": "p.d.papakostas@hotmail.com"
5 }

Body
201 Created

Pretty Raw Preview Visualize JSON
1 {
2   "status": "success",
3   "data": {
4     "tour": {
5       "name": "Panagiotis Papakostass",
6       "age": "19",
7       "email": "p.d.papakostas@hotmail.com",
8       "_id": "65e4f90734b091efe164c7a1",
9       "__v": 0
10    }
11  }
12 }
```

# Λειτουργικότητα – Λειτουργίες User

## ► Create User (Age < 18)

The screenshot shows a REST client interface with a POST request to `http://localhost:8080/smartTourism/api/users/`. The request body is a JSON object: `{ "name": "Panagiotis Papakostass", "age": 17, "email": "p.d.papakostas@hotmail.com" }`. The response status is `404 Not Found`. The response body is a JSON object: `{ "status": "fail", "message": "User must be 18 years old or older!" }`.

```
POST http://localhost:8080/smartTourism/api/users/

Params Auth Headers (8) Body Pre-req. Tests Settings
raw JSON

1 {
2   "name": "Panagiotis Papakostass",
3   "age": 17,
4   "email": "p.d.papakostas@hotmail.com"
5 }

Body 404 Not Found

Pretty Raw Preview Visualize JSON
1 {
2   "status": "fail",
3   "message": "User must be 18 years old or older!"
4 }
```

## ► Create User (Existing email)

The screenshot shows a REST client interface with a POST request to `http://localhost:8080/smartTourism/api/users/`. The request body is a JSON object: `{ "name": "Panagiotis Papakostass", "age": 19, "email": "p.d.papakostas@gmail.com" }`. The response status is `404 Not Found`. The response body is a JSON object: `{ "status": "fail", "message": "User's email already exists!" }`.

```
POST http://localhost:8080/smartTourism/api/users/

Params Auth Headers (8) Body Pre-req. Tests Settings
raw JSON

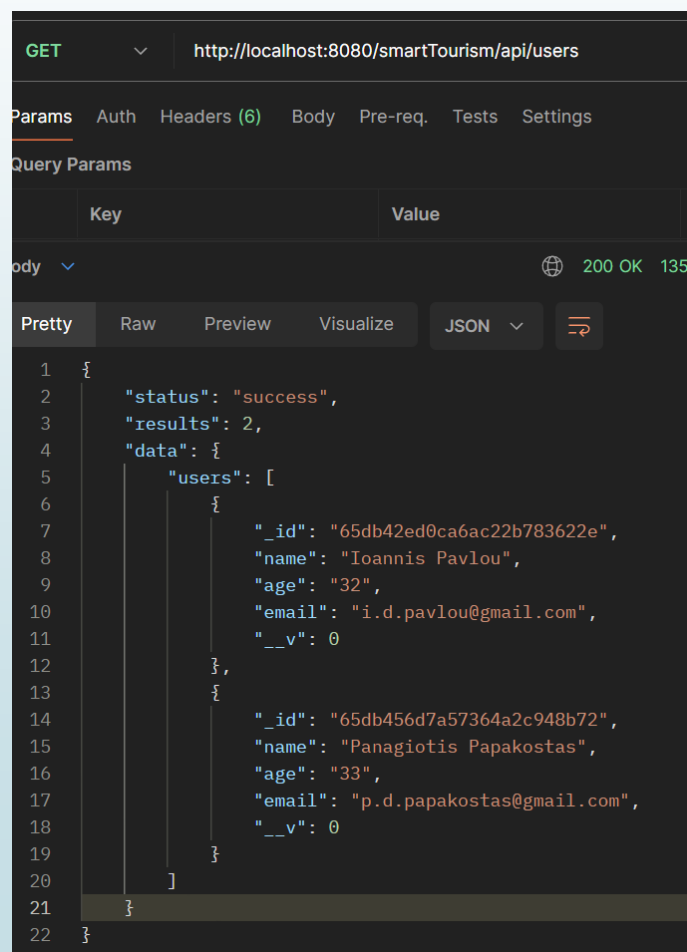
1 {
2   "name": "Panagiotis Papakostass",
3   "age": 19,
4   "email": "p.d.papakostas@gmail.com"
5 }

Body 404 Not Found

Pretty Raw Preview Visualize JSON
1 {
2   "status": "fail",
3   "message": "User's email already exists!"
4 }
```

# Λειτουργικότητα – Λειτουργίες User

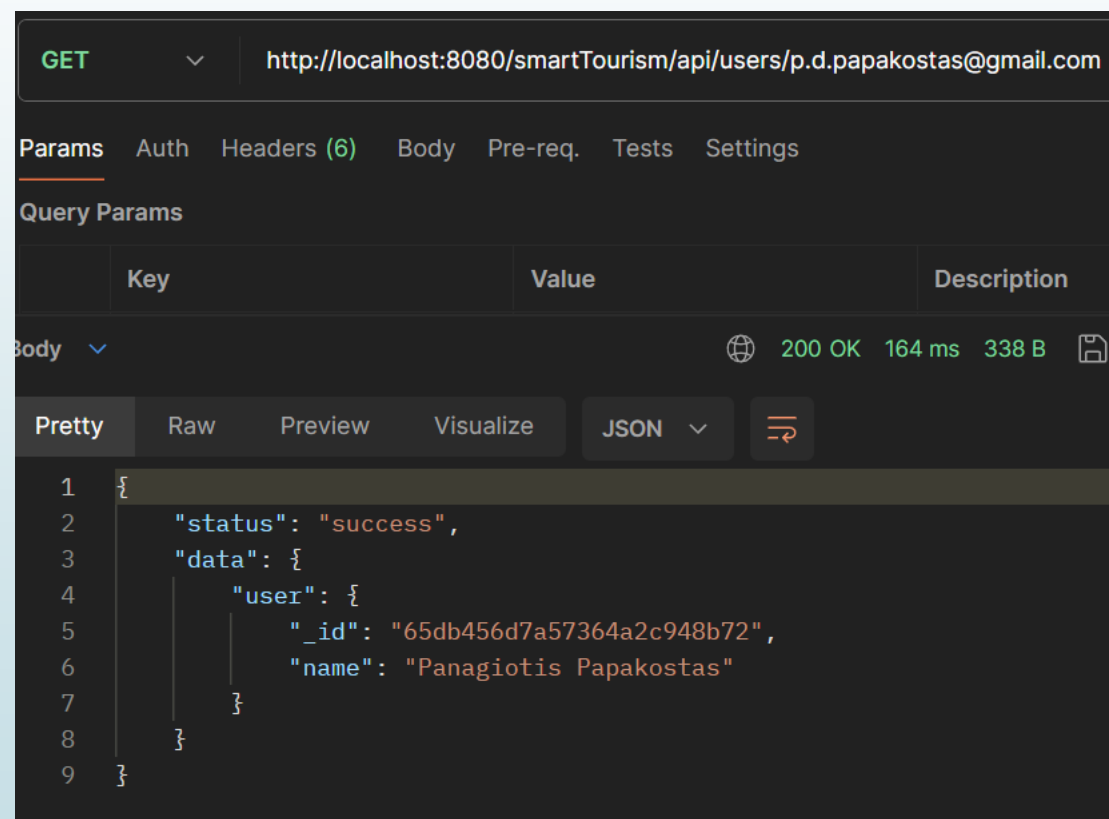
➡ Get All Users



```
GET http://localhost:8080/smartTourism/api/users

Params Auth Headers (6) Body Pre-req. Tests Settings
Query Params
Key Value
Body 200 OK 135
Pretty Raw Preview Visualize JSON ↗
1 {
2   "status": "success",
3   "results": 2,
4   "data": {
5     "users": [
6       {
7         "_id": "65db42ed0ca6ac22b783622e",
8         "name": "Ioannis Pavlou",
9         "age": "32",
10        "email": "i.d.pavlou@gmail.com",
11        "__v": 0
12      },
13      {
14        "_id": "65db456d7a57364a2c948b72",
15        "name": "Panagiotis Papakostas",
16        "age": "33",
17        "email": "p.d.papakostas@gmail.com",
18        "__v": 0
19      }
20    ]
21  }
22 }
```

➡ Get User By Email

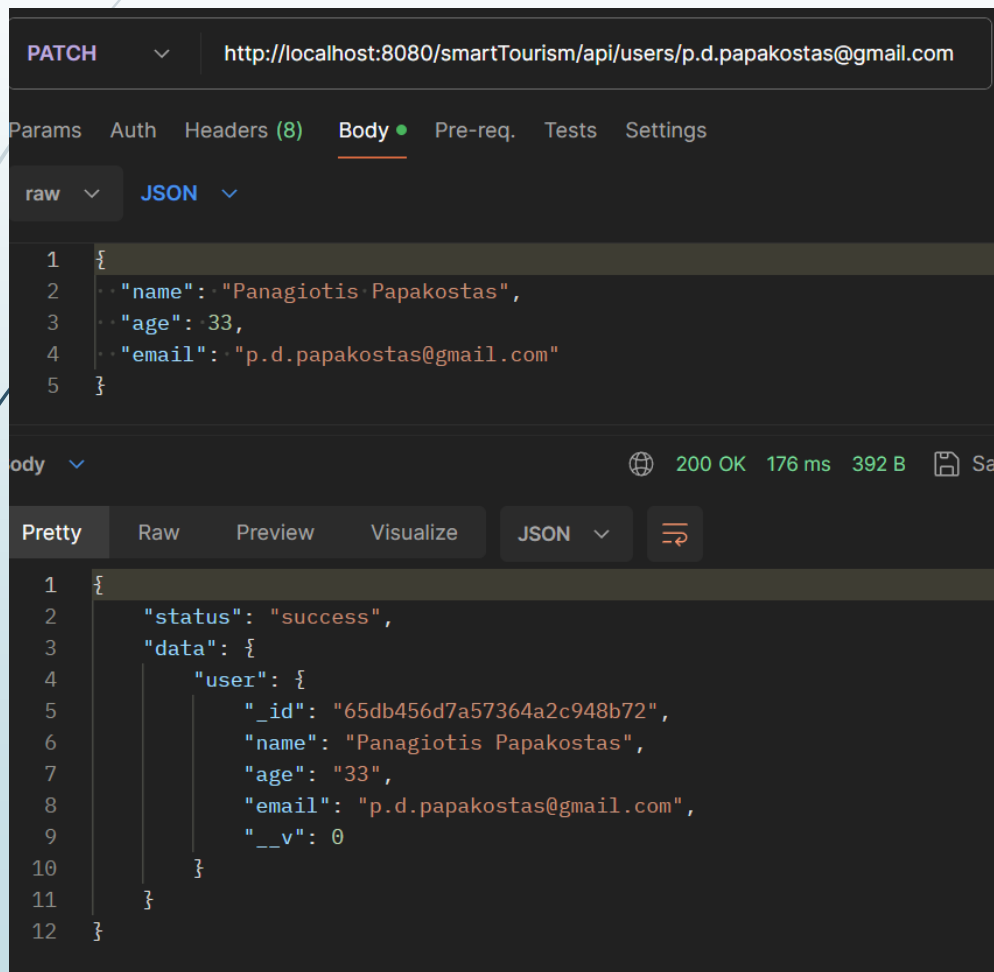


```
GET http://localhost:8080/smartTourism/api/users/p.d.papakostas@gmail.com

Params Auth Headers (6) Body Pre-req. Tests Settings
Query Params
Key Value Description
Body 200 OK 164 ms 338 B
Pretty Raw Preview Visualize JSON ↗
1 {
2   "status": "success",
3   "data": {
4     "user": {
5       "_id": "65db456d7a57364a2c948b72",
6       "name": "Panagiotis Papakostas"
7     }
8   }
9 }
```

# Λειτουργικότητα – Λειτουργίες User

## ➤ Update User By Email



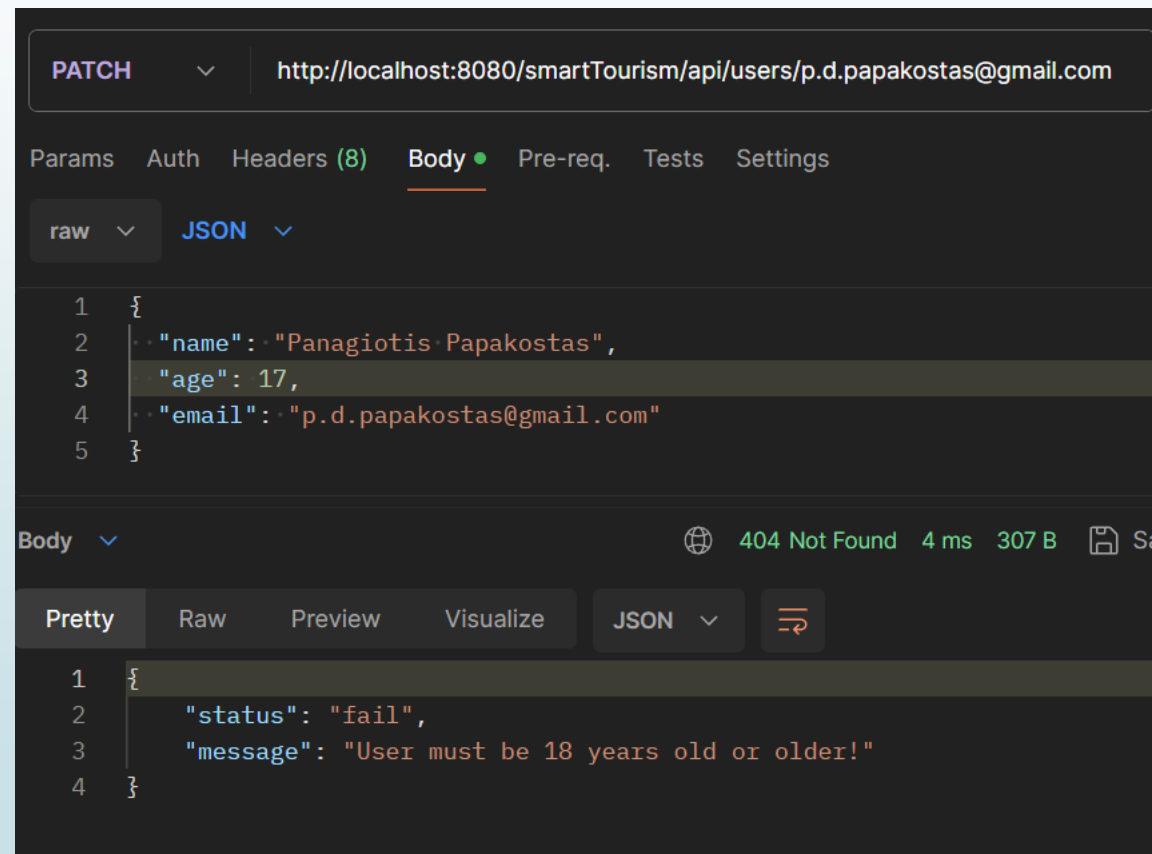
The screenshot shows a REST client interface with a PATCH request to `http://localhost:8080/smartTourism/api/users/p.d.papakostas@gmail.com`. The request body is a JSON object with `"name": "Panagiotis Papakostas", "age": 33, "email": "p.d.papakostas@gmail.com"`. The response status is `200 OK` with a response time of `176 ms` and a size of `392 B`. The response body is a JSON object with `"status": "success", "data": { "user": { "_id": "65db456d7a57364a2c948b72", "name": "Panagiotis Papakostas", "age": "33", "email": "p.d.papakostas@gmail.com", "__v": 0 } }`.

```
PATCH http://localhost:8080/smartTourism/api/users/p.d.papakostas@gmail.com

Params Auth Headers (8) Body Pre-req. Tests Settings
raw JSON
1 {
2   "name": "Panagiotis Papakostas",
3   "age": 33,
4   "email": "p.d.papakostas@gmail.com"
5 }

body 200 OK 176 ms 392 B
Pretty Raw Preview Visualize JSON
1 {
2   "status": "success",
3   "data": {
4     "user": {
5       "_id": "65db456d7a57364a2c948b72",
6       "name": "Panagiotis Papakostas",
7       "age": "33",
8       "email": "p.d.papakostas@gmail.com",
9       "__v": 0
10    }
11  }
12 }
```

## ➤ Update User By Email (Age < 18)



The screenshot shows a REST client interface with a PATCH request to `http://localhost:8080/smartTourism/api/users/p.d.papakostas@gmail.com`. The request body is a JSON object with `"name": "Panagiotis Papakostas", "age": 17, "email": "p.d.papakostas@gmail.com"`. The response status is `404 Not Found` with a response time of `4 ms` and a size of `307 B`. The response body is a JSON object with `"status": "fail", "message": "User must be 18 years old or older!"`.

```
PATCH http://localhost:8080/smartTourism/api/users/p.d.papakostas@gmail.com

Params Auth Headers (8) Body Pre-req. Tests Settings
raw JSON
1 {
2   "name": "Panagiotis Papakostas",
3   "age": 17,
4   "email": "p.d.papakostas@gmail.com"
5 }

Body 404 Not Found 4 ms 307 B
Pretty Raw Preview Visualize JSON
1 {
2   "status": "fail",
3   "message": "User must be 18 years old or older!"
4 }
```

# Λειτουργικότητα – Λειτουργίες User

## ➤ Update User By Email (Existing email)

```
PATCH http://localhost:8080/smartTourism/api/users/p.d.papakostas@gmail.com

Params Auth Headers (8) Body Pre-req. Tests Settings
raw JSON
1 {
2   "name": "Panagiotis Papakostas",
3   "age": 19,
4   "email": "i.d.pavlou@gmail.com"
5 }

Body 404 Not Found 162 ms 615 B
Pretty Raw Preview Visualize JSON
1 {
2   "status": "fail",
3   "message": {
4     "ok": 0,
5     "code": 11000,
6     "codeName": "DuplicateKey",
7     "keyPattern": {
8       "email": 1
9     },
10    "keyValue": {
11      "email": "i.d.pavlou@gmail.com"
12    },
13    "$clusterTime": {
14      "clusterTime": {
15        "$timestamp": "7342269257054421033"
16      },
17      "signature": {
18        "hash": "5MQ+fqBGznS7oeBQ6M1INEmEZZI=",
19        "keyId": {
20          "low": 43,
21          "high": 1699557791,
22          "unsigned": false
23        }
24      }
25    },
26    "operationTime": {
27      "$timestamp": "7342269257054421031"
28    }
29  }
30 }
```

## ➤ Update User By Email (User not exists)

```
PATCH http://localhost:8080/smartTourism/api/users/p.d.papakostas@yahoo.com

Params Auth Headers (8) Body Pre-req. Tests Settings
raw JSON
1 {
2   "name": "Panagiotis Papakostas",
3   "age": 33,
4   "email": "p.d.papakostas@gmail.com"
5 }

Body 404 Not Found 211 ms 286 B
Pretty Raw Preview Visualize JSON
1 {
2   "status": "fail",
3   "message": "User not found"
4 }
```

# Λειτουργικότητα – Λειτουργίες User

➡ Delete User By Email

The screenshot shows a Postman interface for a DELETE request. The URL is `http://localhost:8080/smartTourism/api/users/p.d.papakostas@hotmail.com`. The response status is `200 OK` with a response time of `140 ms` and a body size of `267 B`. The response body is a JSON object: `{ "status": "success", "data": null }`.

Key	Value	Description
Key	Value	Description

```
1 {
2   "status": "success",
3   "data": null
4 }
```

➡ Delete User By Email (User not exists)

The screenshot shows a Postman interface for a DELETE request. The URL is `http://localhost:8080/smartTourism/api/users/p.d.papakostas@yahoo.com`. The response status is `404 Not Found` with a response time of `135 ms` and a body size of `286 B`. The response body is a JSON object: `{ "status": "fail", "message": "User not found" }`.

Key	Value	Description
Key	Value	Description

```
1 {
2   "status": "fail",
3   "message": "User not found"
4 }
```

# Λειτουργικότητα – Λειτουργίες POI

## ➡ Create POI

```
POST http://localhost:8080/smartTourism/api/pois/

{
  "name": "Brunello",
  "category": "Restaurants",
  "address": "Loukianou 21b, Athina 106 75",
  "geoCoordinates": {
    "type": "Point",
    "coordinates": [57.948627, 23.856803]
  }
}
```

201 Created 155 m

```
{
  "status": "success",
  "data": {
    "tour": {
      "name": "Brunello",
      "category": "Restaurants",
      "address": "Loukianou 21b, Athina 106 75",
      "geoCoordinates": {
        "type": "Point",
        "coordinates": [
          57.948627,
          23.856803
        ]
      },
      "photo": "default.png",
      "_id": "65e5e3dd34b091efe164c7b0",
      "__v": 0
    }
  }
}
```

## ➡ Create POI with Photo

```
POST http://localhost:8080/smartTourism/api/pois/

form-data
Key      Value
name     Text  Brunello
category Text  Restaurants
address  Text  Loukianou 21b, Athina 106 75
geoCoordinates Text  {"type": "Point", "coordinates": [...]}
photo    File  brunello.jpg
```

201 Created 164 m

```
{
  "status": "success",
  "data": {
    "tour": {
      "name": "Brunello",
      "category": "Restaurants",
      "address": "Loukianou 21b, Athina 106 75",
      "geoCoordinates": {
        "type": "Point",
        "coordinates": [
          57.948627,
          23.856803
        ]
      },
      "photo": "brunello.jpg",
      "_id": "65e5e48334b091efe164c7b4",
      "__v": 0
    }
  }
}
```



# Λειτουργικότητα – Λειτουργίες POI

## ➡ Get All POIs

```
GET http://localhost:8080/smartTourism/api/pois

Params Auth Headers (6) Body Pre-req. Tests Settings
Query Params
Key Value Description
Body 200 OK 132 ms 973 B
Pretty Raw Preview Visualize JSON

1 {
2   "status": "success",
3   "results": 3,
4   "data": {
5     "pois": [
6       {
7         "geoCoordinates": {
8           "type": "Point",
9           "coordinates": [
10            38.029599,
11            23.766345
12          ]
13        },
14        "_id": "65db713af40173336007ca14",
15        "name": "Alsos of Nea Smyrnis",
16        "category": "Parks",
17        "address": "Leof. El. Venizelou 12, Nea Smyrni",
18        "photo": "alsos-neas-smyrnis.jpg",
19        "__v": 0
20      },
```

## ➡ Get POI by Name

```
GET http://localhost:8080/smartTourism/api/pois/getPoi?name=Veikou Grove

Params Auth Headers (6) Body Pre-req. Tests Settings
Query Params
Key Value Description
_id 65db73a735f6ee5a5653ccde
name Veikou Grove
address Leof. El. Venizelou 12, Nea Smyrni
Key Value Description
Body 200 OK 137 ms 496 B
Pretty Raw Preview Visualize JSON

1 {
2   "status": "success",
3   "data": {
4     "poi": {
5       "geoCoordinates": {
6         "type": "Point",
7         "coordinates": [
8           38.029599,
9           23.766345
10        ]
11      },
12      "_id": "65db73a735f6ee5a5653ccde",
13      "name": "Veikou Grove",
14      "category": "Parks",
15      "address": "Anagenniseos & Galatsi 111 46",
16      "photo": "alsos-veikou.jpg",
17      "__v": 0
18    }
19  }
20 }
```

# Λειτουργικότητα – Λειτουργίες POI

## ➡ Get POI by Address

GET <http://localhost:8080/smartTourism/api/pois/getPoi?address=Leof. El. Ven...>

Params • Auth Headers (6) Body Pre-req. Tests Settings

Query Params

<input type="checkbox"/> Key	Value	Description
<input type="checkbox"/> _id	65db73a735f6ee5a5653ccde	
<input type="checkbox"/> name	Veikou Grove	
<input checked="" type="checkbox"/> address	Leof. El. Venizelou 12, Nea Smyrni	
<input type="checkbox"/> Key	Value	Description

body [200 OK](#) 134 ms 514 B [Save](#)

Pretty Raw Preview Visualize JSON [↔](#)

```
1 {
2   "status": "success",
3   "data": {
4     "poi": {
5       "geoCoordinates": {
6         "type": "Point",
7         "coordinates": [
8           38.029599,
9           23.766345
10        ]
11      },
12      "_id": "65db713af40173336007ca14",
13      "name": "Alsos of Nea Smyrnis",
14      "category": "Parks",
15      "address": "Leof. El. Venizelou 12, Nea Smyrni",
16      "photo": "alsos-neas-smyrnis.jpg",
17      "__v": 0
18    }
19  }
20 }
```

## ➡ Get POI by ID

GET [http://localhost:8080/smartTourism/api/pois/getPoi?\\_id=65db73a735f6ee5a5653ccde](http://localhost:8080/smartTourism/api/pois/getPoi?_id=65db73a735f6ee5a5653ccde)

Params • Auth Headers (6) Body Pre-req. Tests Settings

Query Params

<input type="checkbox"/> Key	Value	Description
<input checked="" type="checkbox"/> _id	65db73a735f6ee5a5653ccde	
<input type="checkbox"/> name	Veikou Grove	
<input type="checkbox"/> address	Leof. El. Venizelou 12, Nea Smyrni	
<input type="checkbox"/> Key	Value	Description

body [200 OK](#) 560 ms 496 B [Save](#)

Pretty Raw Preview Visualize JSON [↔](#)

```
1 {
2   "status": "success",
3   "data": {
4     "poi": {
5       "geoCoordinates": {
6         "type": "Point",
7         "coordinates": [
8           38.029599,
9           23.766345
10        ]
11      },
12      "_id": "65db73a735f6ee5a5653ccde",
13      "name": "Veikou Grove",
14      "category": "Parks",
15      "address": "Anagenniseos &, Galatsi 111 46",
16      "photo": "alsos-veikou.jpg",
17      "__v": 0
18    }
19  }
20 }
```

# Λειτουργικότητα – Λειτουργίες POI

## Update POI by ID

PATCH http://localhost:8080/smartTourism/api/pois/65db713af40173336007ca14

Params Auth Headers (8) Body Pre-req. Tests Settings

raw JSON

```
1 {
2   "name": "Alsos of Nea Smyrnis",
3   "category": "Parks",
4   "address": "Leof. El. Venizelou 12, Nea Smyrni",
5   "geoCoordinates": {
6     "type": "Point",
7     "coordinates": [37.948627, 23.716803]
8   }
9 }
```

body 200 OK 153 ms 514 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "success",
3   "data": {
4     "poi": {
5       "geoCoordinates": {
6         "type": "Point",
7         "coordinates": [
8           37.948627,
9           23.716803
10        ]
11      },
12      "_id": "65db713af40173336007ca14",
13      "name": "Alsos of Nea Smyrnis",
14      "category": "Parks",
15      "address": "Leof. El. Venizelou 12, Nea Smyrni",
16      "photo": "alsos-neas-smyrnis.jpg",
17      "__v": 0
18    }
19  }
20 }
```

## Update POI by ID with Photo

PATCH http://localhost:8080/smartTourism/api/pois/65db713af40173336007ca14

Params Auth Headers (8) Body Pre-req. Tests Settings

form-data

	Key	Value	Description
<input checked="" type="checkbox"/>	name	Text Alsos of Nea Smyrnis	
<input checked="" type="checkbox"/>	category	Text Parks	
<input checked="" type="checkbox"/>	address	Text Leof. El. Venizelou 12, Nea Smy...	
<input checked="" type="checkbox"/>	geoCoordinates	Text {"type": "Point", "coordinates": [...]	
<input checked="" type="checkbox"/>	photo	File alsos-neas-smyrn...	
	Key	Text Value	Description

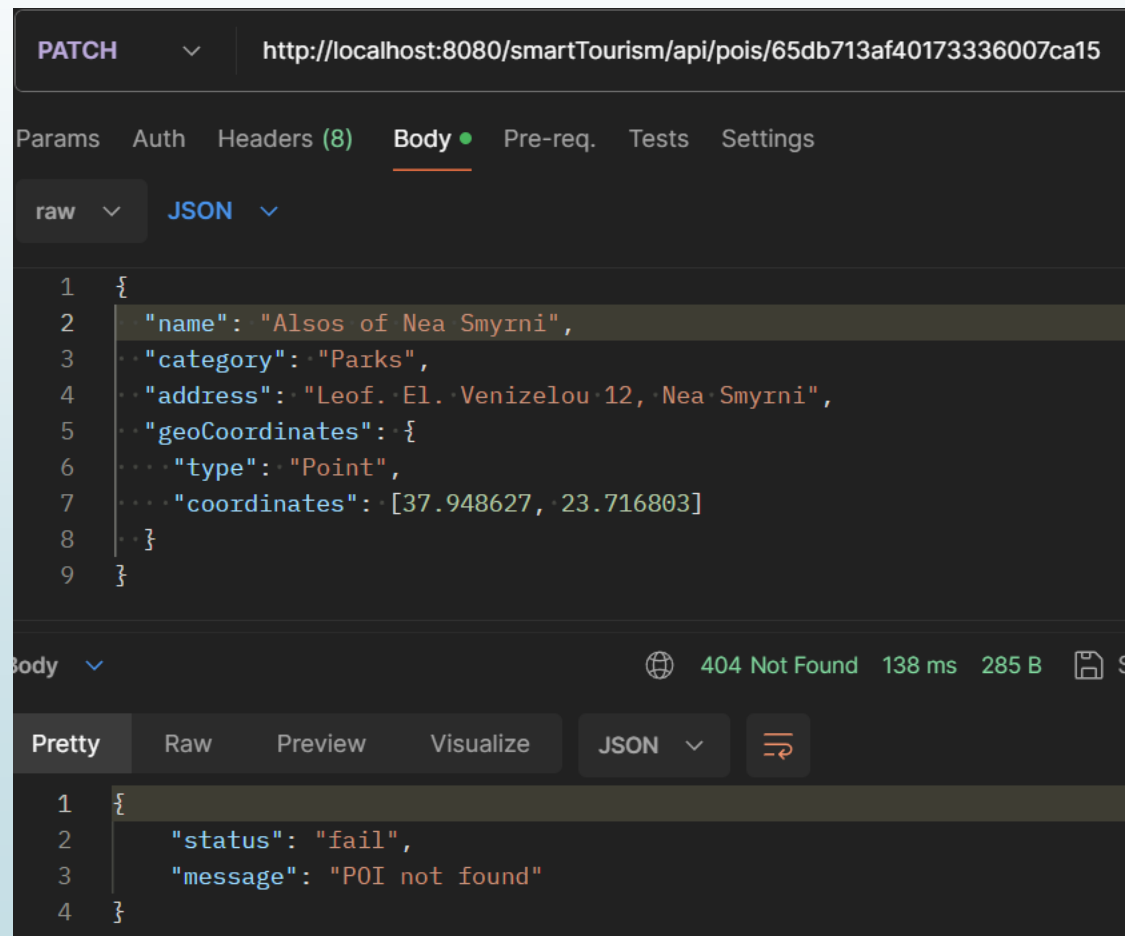
body 200 OK 141 ms 514 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "success",
3   "data": {
4     "poi": {
5       "geoCoordinates": {
6         "type": "Point",
7         "coordinates": [
8           38.029599,
9           23.766345
10        ]
11      },
12      "_id": "65db713af40173336007ca14",
13      "name": "Alsos of Nea Smyrnis",
14      "category": "Parks",
15      "address": "Leof. El. Venizelou 12, Nea Smyrni",
16      "photo": "alsos-neas-smyrnis.jpg",
17      "__v": 0
18    }
19  }
20 }
```

# Λειτουργικότητα – Λειτουργίες POI

- ➡ Update POI by ID (POI not exists)



The screenshot shows a REST client interface with a PATCH request to `http://localhost:8080/smartTourism/api/pois/65db713af40173336007ca15`. The request body is a JSON object with the following structure:

```
1 {  
2   "name": "Alsos of Nea Smyrni",  
3   "category": "Parks",  
4   "address": "Leof. El. Venizelou 12, Nea Smyrni",  
5   "geoCoordinates": {  
6     "type": "Point",  
7     "coordinates": [37.948627, 23.716803]  
8   }  
9 }
```

The response status is `404 Not Found` with a response time of `138 ms` and a body size of `285 B`. The response body is a JSON object:

```
1 {  
2   "status": "fail",  
3   "message": "POI not found"  
4 }
```

# Λειτουργικότητα – Λειτουργίες POI

➡ Delete POI by ID

The screenshot shows a REST client interface with a DELETE request to `http://localhost:8080/smartTourism/api/pois/65e5e3dd34b091efe164c7b0`. The response status is `200 OK` with a response time of `300 ms` and a body size of `267 B`. The response body is a JSON object:

```
1 {  
2   "status": "success",  
3   "data": null  
4 }
```

➡ Create POI by ID (POI not found)

The screenshot shows a REST client interface with a DELETE request to `http://localhost:8080/smartTourism/api/pois/65e5e48334b091efe164c7c4`. The response status is `404 Not Found` with a response time of `140 ms` and a body size of `285 B`. The response body is a JSON object:

```
1 {  
2   "status": "fail",  
3   "message": "POI not found"  
4 }
```

# Λειτουργικότητα – Αποθήκευση στη ΒΔ

## ➡ Users

**Smart-Tourism.Users**

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 208B TOTAL DOCUMENTS: 2

[Find](#) [Indexes](#) [Schema Anti-Patterns 0](#) [Agg](#)

[Filter](#) Type a query: { field: 'value' }

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('65db42ed0ca6ac22b783622e')
name: "Ioannis Pavlou"
age: 32
email: "i.d.pavlou@gmail.com"
__v: 0
```

```
_id: ObjectId('65db456d7a57364a2c948b72')
name: "Panagiotis Papakostas"
age: "33"
email: "p.d.papakostas@gmail.com"
__v: 0
```

## ➡ POIs

**Smart-Tourism.POIs**

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 683B TOTAL DOCUMENTS: 3

[Find](#) [Indexes](#) [Schema Anti-Patterns 0](#) [Agg](#)

[Filter](#) Type a query: { field: 'value' }

QUERY RESULTS: 1-3 OF 3

```
_id: ObjectId('65db713af40173336007ca14')
name: "Alsos of Nea Smyrnis"
category: "Parks"
address: "Leof. El. Venizelou 12, Nea Smyrni"
geoCoordinates: Object
photo: "alsos-neas-smyrnis.jpg"
__v: 0
```

```
_id: ObjectId('65db73a735f6ee5a5653ccde')
name: "Veikou Grove"
category: "Parks"
address: "Anagenniseos &, Galatsi 111 46"
geoCoordinates: Object
photo: "alsos-veikou.jpg"
__v: 0
```

```
_id: ObjectId('65e4feae34b091efe164c7ac')
name: "Brunello"
category: "Restaurants"
address: "Loukianou 21b, Athina 106 75"
geoCoordinates: Object
photo: "brunello.jpg"
__v: 0
```

# Παρατηρήσεις

- Για τη ΒΔ Έχει χρησιμοποιηθεί το ATLAS.
- Πριν την εκκίνηση της εφαρμογής πρέπει να εκτελεστεί η εντολή **npm install**, για να εγκατασταθούν τα απαιτούμενα dependencies που ορίζονται στο αρχείο **package.json** (ο φάκελος **node\_modules** δεν συμπεριλαμβάνεται στα αρχεία της εργασίας που έχουν παραδοθεί).
- Η εκκίνηση της εφαρμογής γίνεται εκτελώντας την εντολή **npm start** στο terminal.

```
"scripts": {  
  "start": "nodemon server.js",  
}
```

- Έχει επίσης αποσταλεί και το postman collection για την εκτέλεση των λειτουργιών που περιεγράφηκαν παραπάνω.