

Bước 1:

```
const config = {
  app: {
    port: process.env.PORT || 3000,
  },
  db: {
    uri: process.env.MONGODB_URI || "mongodb://127.0.0.1:27017/contactbook"
  }
};

module.exports = config;
```

Định nghĩa lớp trợ giúp kết nối đến Mongo DB:

```
> utils > js mongodb.util.js > ...
1  const { MongoClient } = require("mongodb");
2
3  class MongoDB {
4    static connect = async (uri) => {
5      if (this.client) return this.client;
6      this.client = await MongoClient.connect(uri);
7      return this.client;
8    }
9  }
10
11 module.exports = MongoDB;
```

server.js

```
const app = require("./app");
const config = require("./app/config");
const MongoDB = require("./app/utils/mongodb.util");

async function startServer() {
  try {
    await MongoDB.connect(config.db.uri);
    console.log("Connected to the databases!");

    const PORT = config.app.port;
    app.listen(PORT, () => {
      console.log(`Server is running on port ${PORT}`);
    });
  } catch (error) {
    console.log("Cannot connect to the databases!", error);
    process.exit();
  }
}

startServer();
```

contact.service.js

```
const { ObjectId, ReturnDocument, ObjectId } = require("mongodb");

class ContactService {
  constructor(client) {
    this.Contact = client.db().collection("contacts");
  }
  // Định nghĩa các phương thức truy xuất CSDL sử dụng mongodb API
  extractConacData(payload) {
    const contact = {
      name: payload.name,
      email: payload.email,
      address: payload.address,
      phone: payload.phone,
      favorite: payload.favorite,
    };
    // Remove undefined fields
    Object.keys(contact).forEach(
      (key) => contact[key] === undefined && delete contact[key]
    );
    return contact;
  }
  async create(payload) {
    const contact = this.extractConacData(payload);
    const result = await this.Contact.findOneAndUpdate(
      contact,
      { $set: { favorite: contact.favorite === true } },
      { ReturnDocument: "after", upsert: true }
    );
    return result;
  }

  async find(filter) {
    const cursor = await this.Contact.find(filter);
    return await cursor.toArray();
  }

  async findByName(name) {
    return await this.find({
      name: { $regex: new RegExp(new RegExp(name)), $options: "i" },
    });
  }

  async findById(id) {
    return await this.Contact.findOne({
      _id: ObjectId.isValid(id) ? new ObjectId(id) : null,
    });
  }
}
```

```

    async update(id, payload) {
      const filter = {
        _id: ObjectId.isValid(id) ? new ObjectId(id) : null,
      };
      const update = this.extractConacData(payload);
      const result = await this.Contact.findOneAndUpdate(
        filter,
        { $set: update },
        { returnOriginal: "after" } // Use this for MongoDB versions < v4.2
      );
      return result;
    }

    async delete(id) {
      const result = await this.Contact.findOneAndDelete({
        _id: ObjectId.isValid(id) ? new ObjectId(id) : null,
      });
      return result;
    }

    async findFavorite() {
      return await this.find({ favorite: true });
    }

    async deleteAll() {
      const result = await this.Contact.deleteMany({});
      return result.deleteCount;
    }
  }

  module.exports = ContactService;

```

Bước 2: Cài đặt các handler
Cài đặt handler create

```
exports.create = async (req, res) => {
  if (!req.body?.name) {
    return next(new ApiError(400, "Name can not be empty"));
  }

  try {
    const contactService = new ContactService(MongoDB.client);
    const document = await contactService.create(req.body);
    return res.send(document);
  } catch (error) {
    return next(
      new ApiError(500, "An error occurred while creating the contact")
    );
  }
};
```

```
extractConacData(payload) {
  const contact = {
    name: payload.name,
    email: payload.email,
    address: payload.address,
    phone: payload.phone,
    favorite: payload.favorite,
  };
  //Remove underfined fields
  Object.keys(contact).forEach(
    (key) => contact[key] === undefined && delete contact[key]
  );
  return contact;
}

async create(payload) {
  const contact = this.extractConacData(payload);
  const result = await this.Contact.findOneAndUpdate(
    contact,
    { $set: { favorite: contact.favorite === true } },
    { ReturnDocument: "after", upsert: true }
  );
  return result;
}
```


Dùng postman để kiểm tra:

The screenshot displays the Postman interface for a POST request. The top section shows the method 'POST' and the URL 'http://localhost:3000/api/contacts'. The 'Body' tab is selected, showing a JSON payload. The bottom section shows the 'Body' tab of the response, displaying a JSON object with an additional '_id' field.

Request:

```
POST http://localhost:3000/api/contacts
```

Request Body (JSON):

```
{  "name": "Long Tran",  "email": "longtran@example.com",  "address": "Ving Long",  "phone": "0914029213104"}
```

Response Body (JSON):

```
{  "_id": "66e82827d6072f9b8fd3621d",  "address": "Ving Long",  "email": "longtran@example.com",  "name": "Long Tran",  "phone": "0914029213104",  "favorite": false}
```

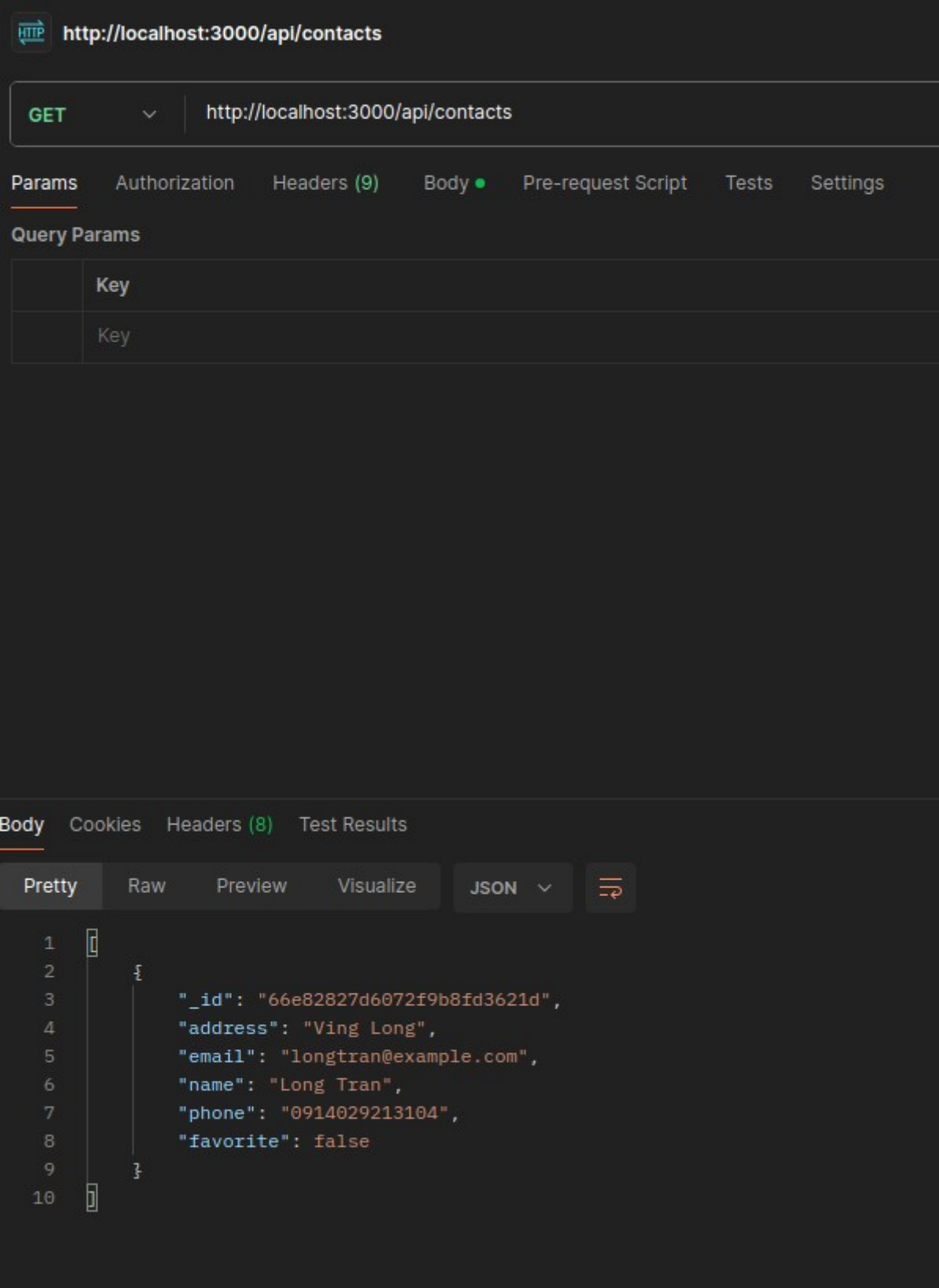
Cài đặt handlerfindAll:

```
1 exports.findAll = async (req, res, next) => {
2   let documents = [];
3
4   try {
5     const contactService = new ContactService(MongoDB.client);
6     const { name } = req.query;
7     if (name) {
8       documents = await contactService.findByName(name);
9     } else {
10      documents = await contactService.find({});
11    }
12  } catch (error) {
13    console.error("Error retrieving contacts:", error.message);
14    console.error("Stack trace:", error.stack);
15    return next(
16      new ApiError(500, "An error occurred while retrieving contacts")
17    );
18  }
19  return res.send(documents);
20 };
21
```

```
async find(filter) {
  const cursor = await this.Contact.find(filter);
  return await cursor.toArray();
}

async findByName(name) {
  return await this.find({
    name: { $regex: new RegExp(new RegExp(name)), $options: "i" },
  });
}
```

Kiểm tra bằng postman:



Cài đặt findOne

```
exports.findOne = async (req, res, next) => {
  try {
    const contactService = new ContactService(MongoDB.client);
    const document = await contactService.findById(req.params.id);
    if (!document) {
      return next(new ApiError(404, "Contact not found"));
    }
    return res.send(document);
  } catch (error) {
    return next(
      new ApiError(500, `Error retrieving contact with id=${req.params.id}`)
    );
  }
};
```

```
async findByName(name) {
  return await this.find({
    name: { $regex: new RegExp(new RegExp(name)), $options: "i" },
  });
}

async findById(id) {
  return await this.Contact.findOne({
    _id: ObjectId.isValid(id) ? new ObjectId(id) : null,
  });
}
```

Kiểm tra bằng Postman:

HTTP

http://localhost:3000/api/contacts/66e82827d6072f9b8fd3621d

GET

http://localhost:3000/api/contacts/66e82827d6072f9b8fd3621d

Params

Authorization

Headers (9)

Body ●

Pre-request Script

Tests

Settings

Query Params

	Key
	Key

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

{

"_id": "66e82827d6072f9b8fd3621d",

"address": "Ving Long",

"email": "longtran@example.com",

"name": "Long Tran",

"phone": "0914029213104",

"favorite": false

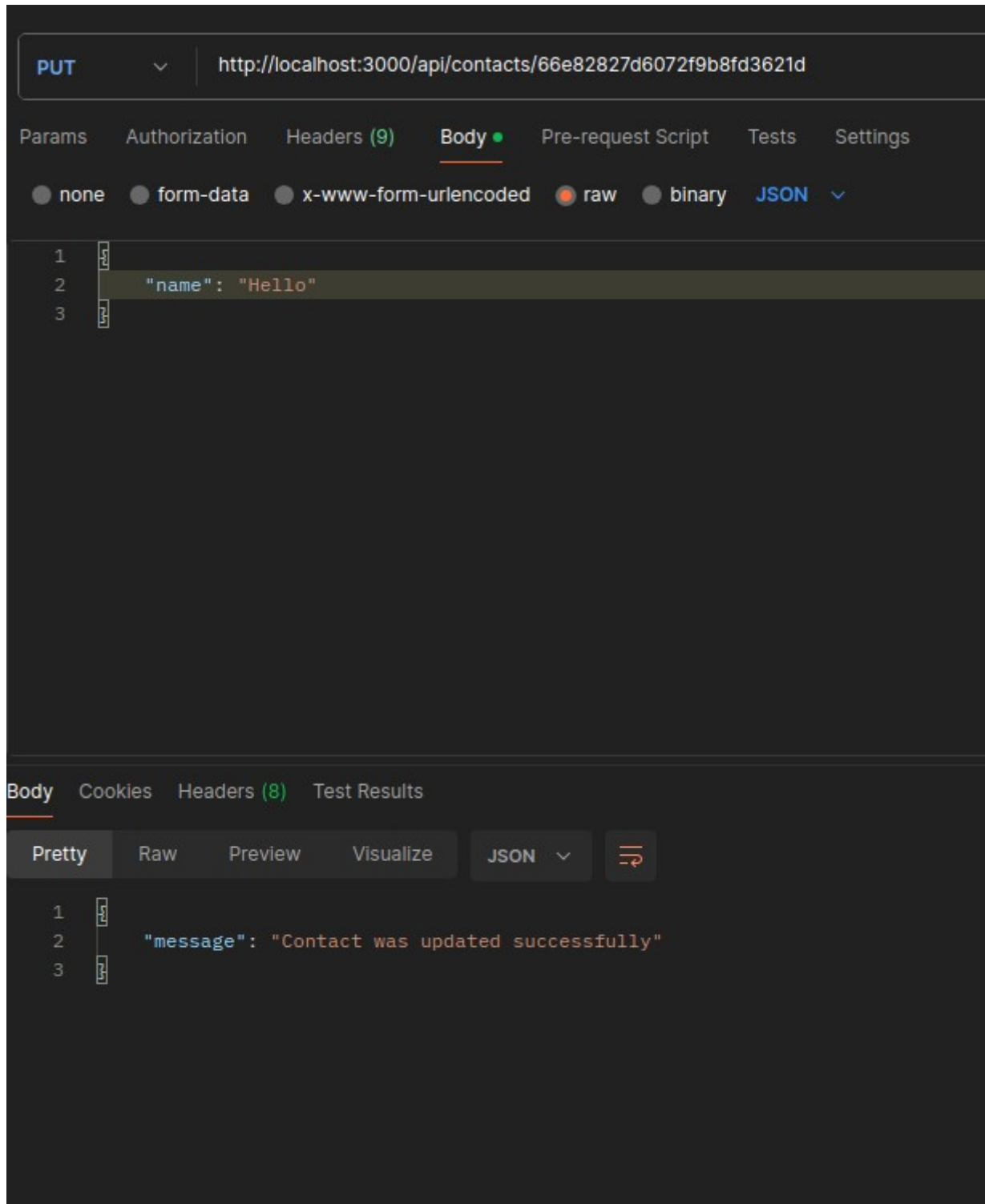
}

Cài đặt handler update

```
async update(id, payload) {  
  const filter = {  
    _id: ObjectId.isValid(id) ? new ObjectId(id) : null,  
  };  
  const update = this.extractConacData(payload);  
  const result = await this.Contact.findOneAndUpdate(  
    filter,  
    { $set: update },  
    { returnOriginal: "after" } // Use this for MongoDB versions < v4.2  
  );  
  return result;  
}
```

```
exports.update = async (req, res, next) => {  
  if (Object.keys(req.body).length === 0) {  
    return next(new ApiError(400, "Data to update can not be empty"));  
  }  
  
  try {  
    const contactService = new ContactService(MongoDB.client);  
    const document = await contactService.update(req.params.id, req.body);  
  
    if (!document) {  
      console.error(`Contact with ID ${req.params.id} not found`);  
      return next(new ApiError(404, "Contact not found"));  
    }  
  
    return res.send({ message: "Contact was updated successfully" });  
  } catch (error) {  
    console.error(  
      `Error while updating contact with ID ${req.params.id}:`,  
      error.message  
    );  
    console.error(error.stack);  
  
    return next(  
      new ApiError(500, `Error updating contact with id=${req.params.id}`)  
    );  
  }  
};
```

Kiểm tra bằng postman:





http://localhost:3000/api/contacts/66e82827d6072f9b8fd3621d

GET



http://localhost:3000/api/contacts/66e82827d6072f9b8fd3621d

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings



none



form-data



x-www-form-urlencoded



raw



binary

JSON



1



2

"name": "Hello"

3



Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON



1



2

"_id": "66e82827d6072f9b8fd3621d",

3

"address": "Ving Long",

4

"email": "longtran@example.com",

5

"name": "Hello",

6

"phone": "0914029213104",

7

"favorite": false

8

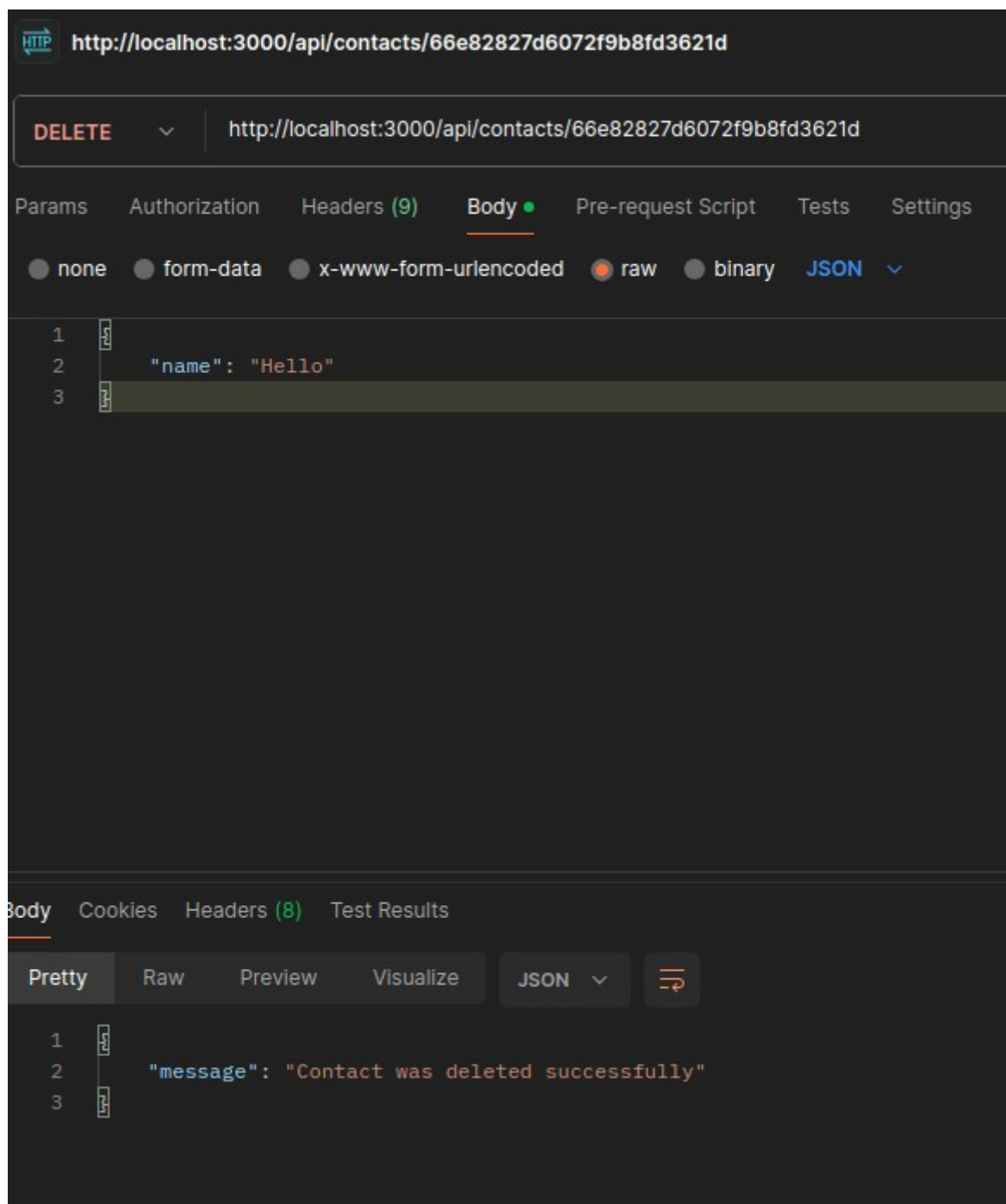


Cài đặt handler delete

```
exports.delete = async (req, res, next) => {
  try {
    const contactService = new ContactService(MongoDB.client);
    const document = await contactService.delete(req.params.id);
    if (!document) {
      return next(new ApiError(404, "Contact not found"));
    }
    return res.send({ message: "Contact was deleted successfully" });
  } catch (error) {
    return next(
      new ApiError(500, `Could not delete contact with id=${req.params.id}`)
    );
  }
};
```

```
async delete(id) {
  const result = await this.Contact.findOneAndDelete({
    _id: ObjectId.isValid(id) ? new ObjectId(id) : null,
  });
  return result;
}
```


Kiểm tra bằng postman



Cài đặt handler `findAllFavorie`

```
exports.findAllFavorite = async (req, res, next) => {
  try {
    const contactService = new ContactService(MongoDB.client);
    const documents = await contactService.findFavorite();
    return res.send(documents);
  } catch (error) {
    return next(
      new ApiError(500, "An error occurred while retrieving favorite contacts")
    );
  }
};
```

```
async findFavorite() {
  return await this.find({ favorite: true });
}
```

Kiểm tra với postman

The image shows the Postman interface for a GET request to `http://localhost:3000/api/contacts/favorite`. The request is configured with the following details:

- Method:** GET
- URL:** `http://localhost:3000/api/contacts/favorite`
- Params:** None
- Authorization:** None
- Body Type:** JSON
- Body Content:**

```
1 {  
2   "name": "Long Tran123231",  
3   "email": "longtran@example.com",  
4   "address": "Ving Long",  
5   "phone": "0914029213104",  
6   "favorite": true  
7 }
```

The response is displayed in the "Body" tab, showing a JSON object with the following structure:

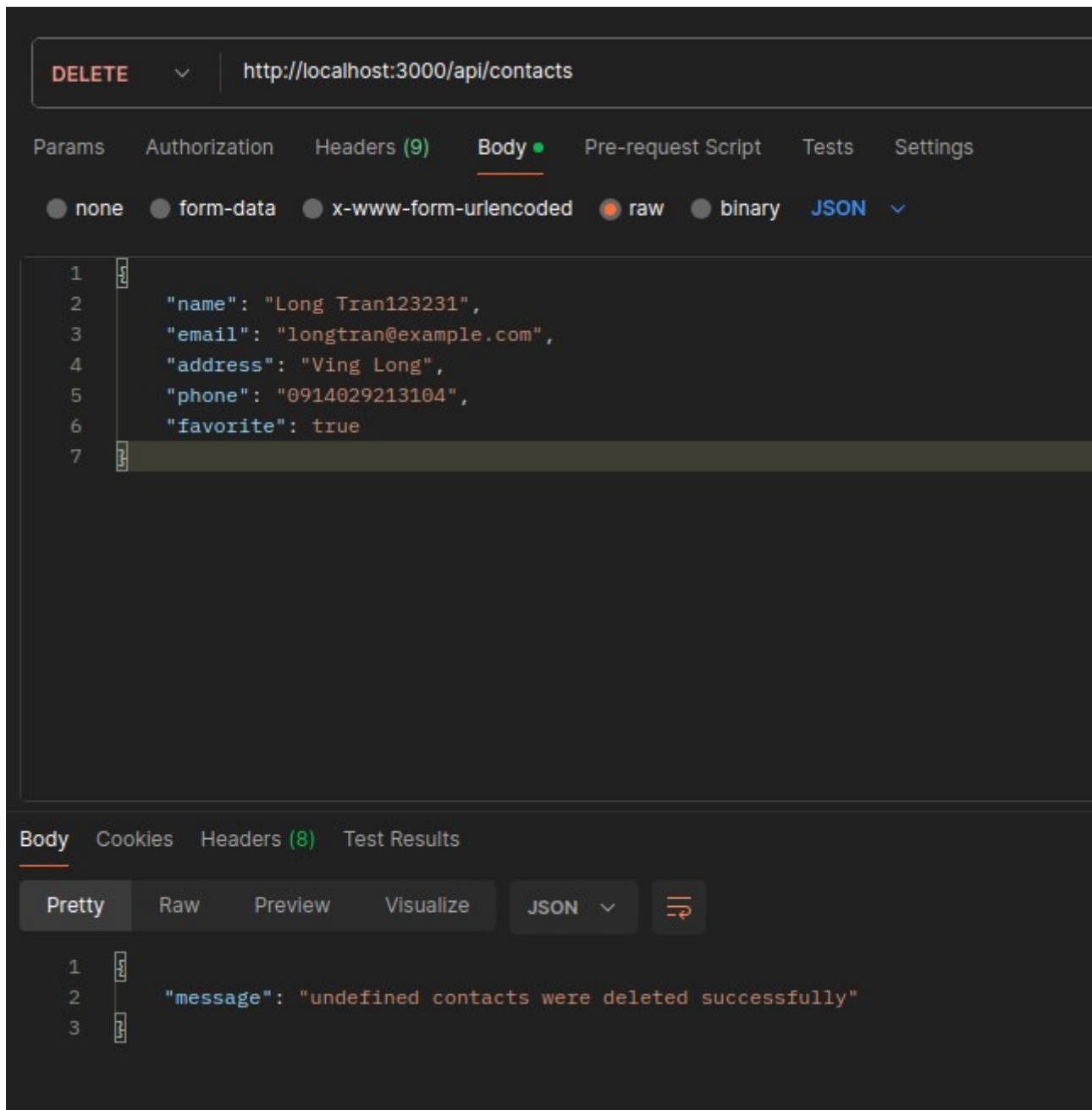
```
1 {  
2   "  
3     "_id": "66e83504d6072f9b8fd3652c",  
4     "email": "longtran@example.com",  
5     "address": "Ving Long",  
6     "phone": "0914029213104",  
7     "favorite": true,  
8     "name": "Long Tran123231"  
9   }  
10 }
```

Cài đặt handler deleteAll:

```
exports.deleteAll = async (req, res, next) => {
  try {
    const contactService = new ContactService(MongoDB.client);
    const deleteCount = await contactService.deleteAll();
    return res.send({
      message: `${deleteCount} contacts were deleted successfully`,
    });
  } catch (error) {
    return next(
      new ApiError(500, "An error occurred while removing all contacts")
    );
  }
};
```

```
2
3   async deleteAll() {
4     const result = await this.Contact.deleteMany({});
5     return result.deleteCount;
6   }
7 }
8
```

Kiểm tra bằng postman:



- ▼ BACKEND_1-CD459F8CB718078910C6EBBCD3E...
 - ▼ app
 - ▼ config
 - JS index.js
 - ▼ controllers
 - JS contact.controller.js
 - ▼ routes
 - JS contact.route.js
 - ▼ services
 - JS contact.service.js
 - ▼ utils
 - JS mongodb.util.js
 - JS api-error.js
 - > node_modules
 - ⚙ .eslintrc.js
 - 📁 .gitignore
 - JS app.js
 - 👤 B2111791_Pham_Tan_Dat_BACKEND_1...
 - { } package-lock.json
 - { } package.json
 - JS server.js