BỘ GIÁO DỤC VÀ DÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO THỰC HÀNH BACKEND_2
PHÁT TRIỂN ỨNG DỤNG WEB
CT449 - 01

Giảng viên hướng dẫn:
ThS. Nguyễn Minh Trung

Sinh viên thực hiện:
Tên: Trần Trương Ngọc Uyển
MSSV: B2207576
Khóa 48

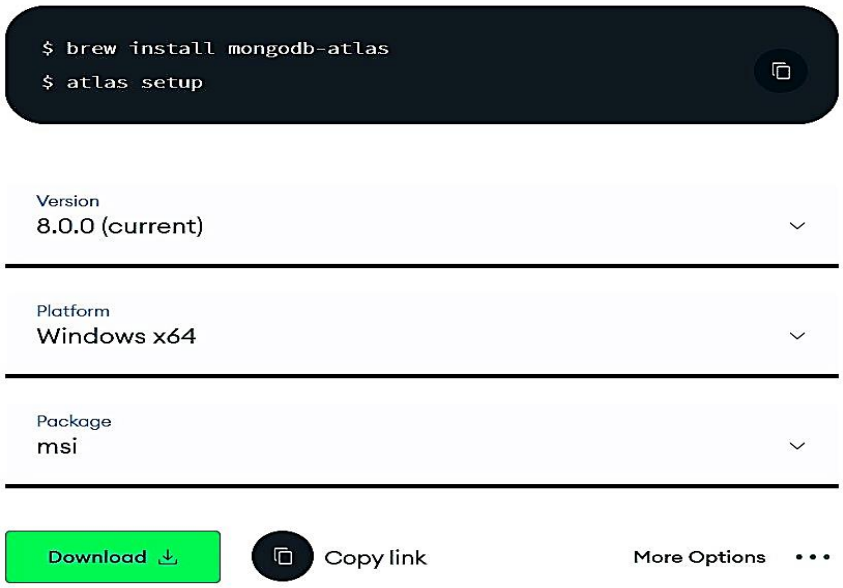
Cần Thơ, năm 2025

MỤC LỤC

DANH MỤC HÌNH

Ứng dụng Contactbook - Backend - Phần 2

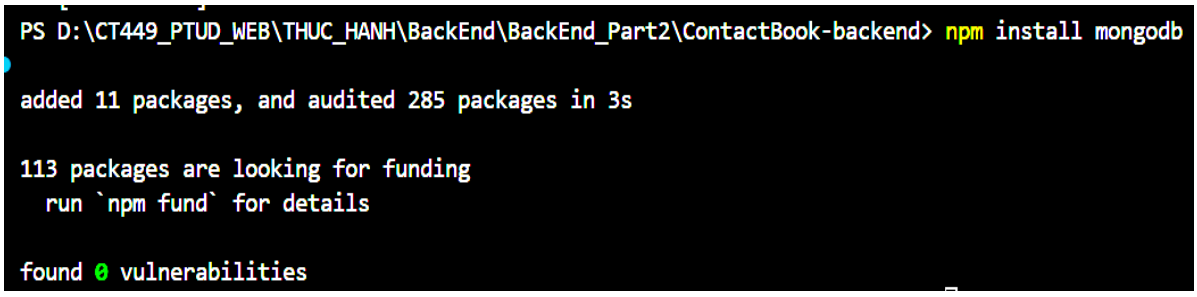
Bước 0: Cài đặt MongoDB



Hình 1: Cài đặt MongoDB

Bước 1: Cài đặt thư viên MongoDB, định nghĩa hàm trợ giúp kết nối và lớp dịch vụ truy xuất cơ sở dữ liệu (CSDL)

– Cài đặt thư viện mongodb vào dự án: **npm install mogodb**



Hình 2: Cài đặt thư viện mongodb

– Trong thư mục app/config, hiệu chỉnh thêm đoạn code trong tập tin index,js

app.js | index.js 3, M

BackEnd > BackEnd_Part1 > ContactBook-backend > app > config > index.js > config > db > uri

```
1  const config = {
2      app: {
3          port: process.env.PORT || 3000,
4      },
5      db: {
6          uri : process.env.MONGOD_URI || "mongodb://127.0.0.1:27017/contactbook"
7      }
8  };
9
10 module.exports = config;
```

Hình 3: Hiệu chỉnh thêm đoạn code trong tập tin index,js

– Định nghĩ lớp trợ giúp kết nối đến mongoDB: app/utils/mongodb.util.js

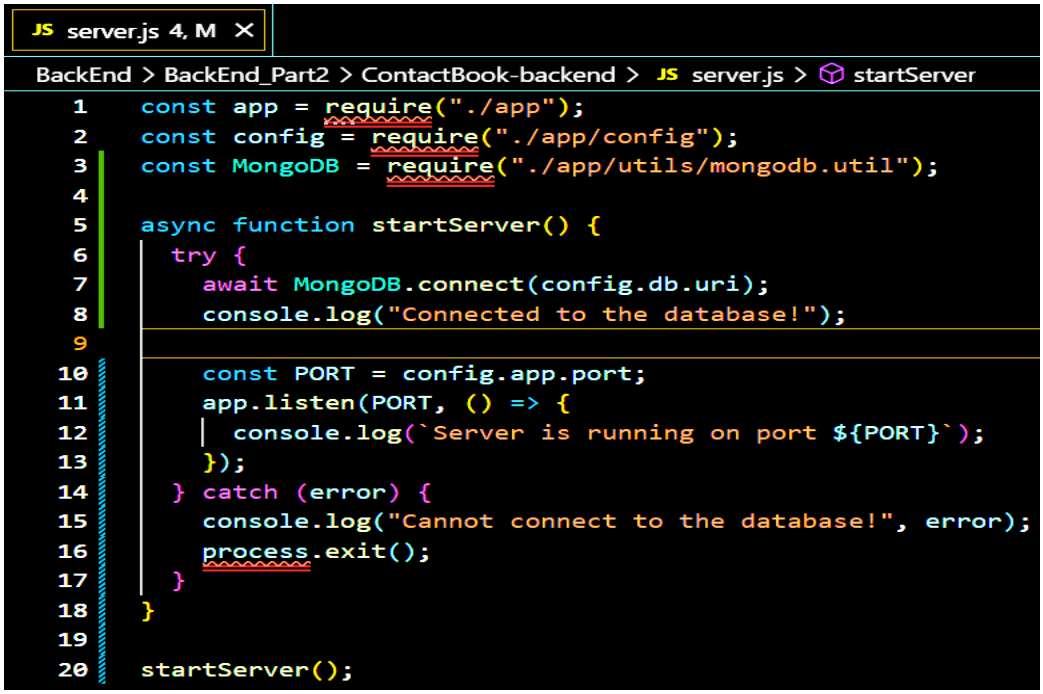
mongodb.util.js 2, U

BackEnd > BackEnd_Part2 > ContactBook-backend > app > utils > 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;
```

Hình 4: Tạo tập tin mongodb.utils.js

– Thay toàn bộ tập tin server.js



Hình 5: Thay đổi nôi dung tập tin server.js

– Định nghĩa các lớp dịch vụ ContactServices (trong tập tin app/services/service.mjs) chứa các API của thư viện MongoDB để thực hiện thao tác với CSDL MongoDB

Hình 6: Định nghĩa các lớp dịch vụ ContactServices

Bước 2: Cài đặt các handler

1. Cài đặt handler create

– Hiệu chỉnh tập tin app/controllers/contact.controller.js

contact.controller.js 9+, M

BackEnd > BackEnd_Part2 > ContactBook-backend > app > controllers > contact.controller.js > ...

```
104  exports.create = async(req,res,next) =>{
105      if(!req.body?.name){
106          return next(new ApiError(400, "name can not be emty"));
107      }
108      try{
109          const contactService = new ContactService(MongoDB.client);
110          const document = await contactService.create(req.body);
111          return res.send(document);
112      } catch(error){
113          return next(
114              new ApiError(500, "An error occurred while creting the contact")
115          );
116      }
117  };
```

Hình 7: Cài đặt handler create

– Tạo phương thức create() trong lớp ContactService

server.js 4, M | contact.service.js 2, U

BackEnd > BackEnd_Part2 > ContactBook-backend > app > services > contact.service.js > ...

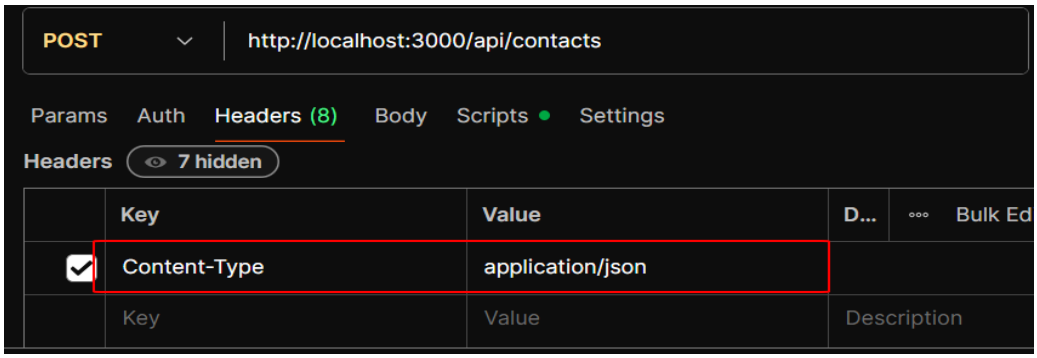
```
1   const { ObjectId } = require("mongodb");
2
3   class ContactService {
4       constructor(client) {
5       this.Contact = client.db().collection("contacts");
6     }
7     // Định nghĩa các phương thức truy xuất CSDL sử dụng mongodb API
8     extractContactData(payload) {
9       const contact = {
10        name: payload.name,
11        email: payload.email,
12        address: payload.address,
13        phone: payload.phone,
14        favorite: payload.favorite,
15      };
16
17      // Remove undefined fields
18      Object.keys(contact).forEach(
19          (key) => contact[key] === undefined && delete contact[key]);
20
21      return contact;
22    }
```

```
24    async create(payload) {
25      const contact = this.extractContactData(payload);
26      const result = await this.Contact.findOneAndUpdate(
27        contact,
28        { $set: { favorite: contact.favorite === true } },
29        { returnDocument: "after", upsert: true }
30      );
31      return result;
32    }
```

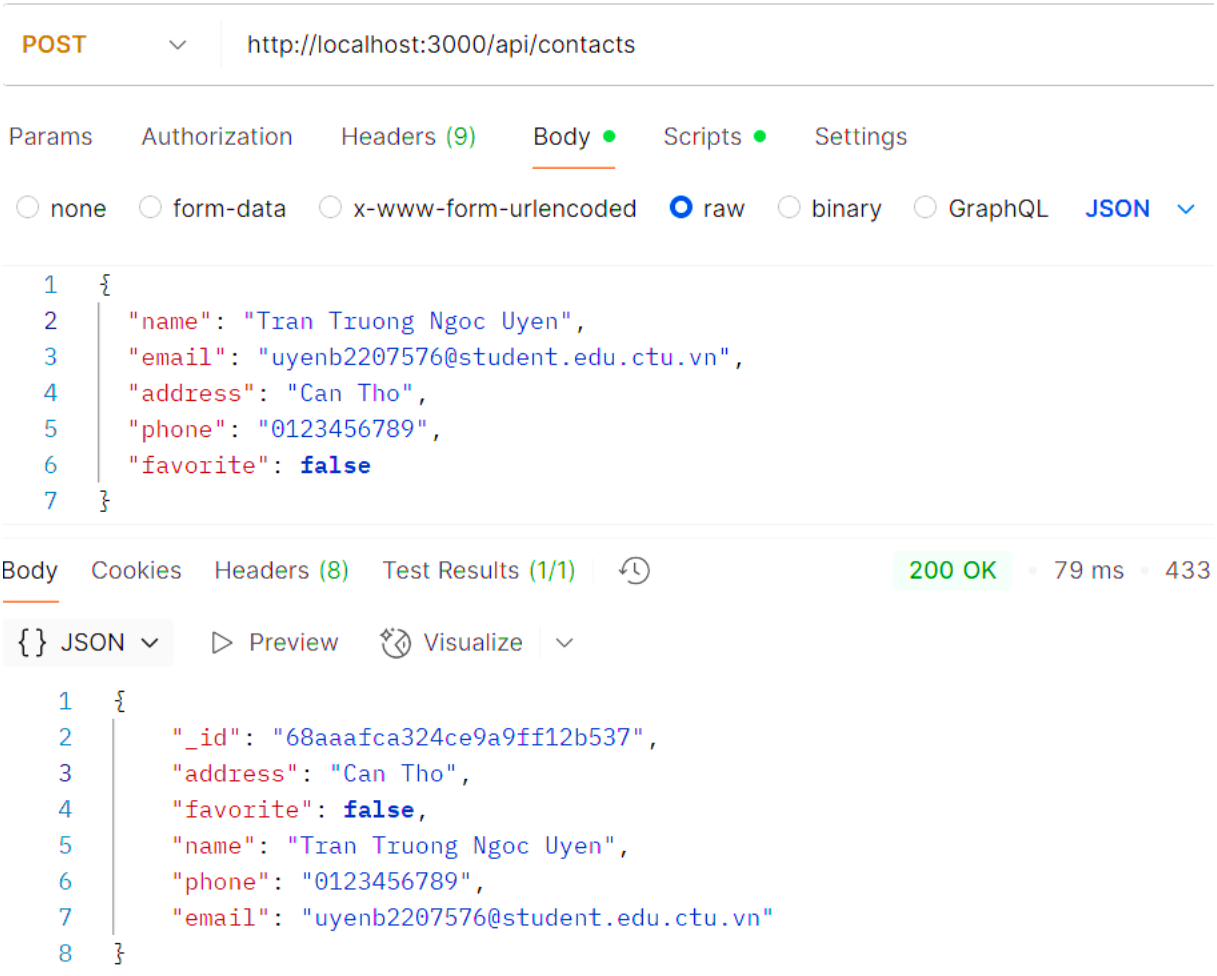
Hình 8: Tạo phương thức create()

– Kiểm tra với **ứng dụng Postman cho create**

- Để gửi dữ liệu JSON về sever với Postman, cần đặt Header "Content-Type: application/json" và đặt dữ liệu JSON trong phần Body của yêu cầu

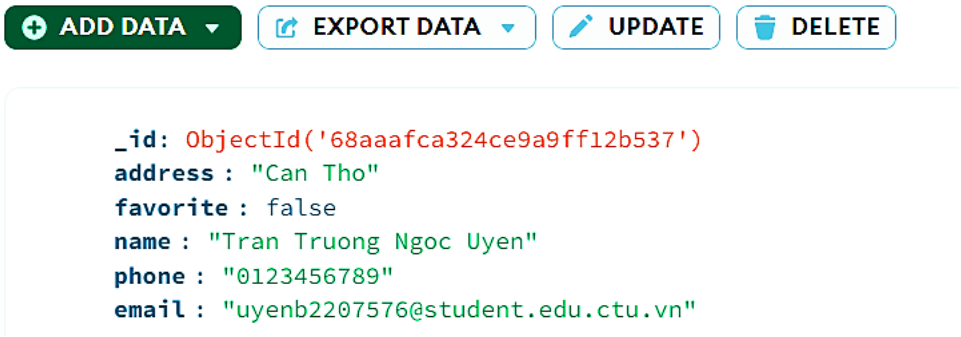


Hình 9: Hiệu chỉnh Header "Content-Type: application/json"



Hình 10: Đặt dữ liệu JSON trong phần Body

– Kết quả bên MongoDB:



Hình 11: Kết quả MongoDB được tạo thành công

2. Cài đặt handler findAll

– Hiệu chỉnh tập tin app/controllers/contact.controller.js

```
exports.findAll = async (req, res, next)=>{
    let document = [];
    try{
        const contactService = new ContactService(MongoDB.client);
        const {name} = req.query;
        if(name){
            document = await contactService.findByName(name);

        }
        else{
            document = await contactService.find({});
        }
    } catch(error){
        return next(
            new ApiError(500, "An error occurred while retrieving contacts")
        );
    }
    return res.send(document)
}
```

Hình 12: Cài đặt handler findAll

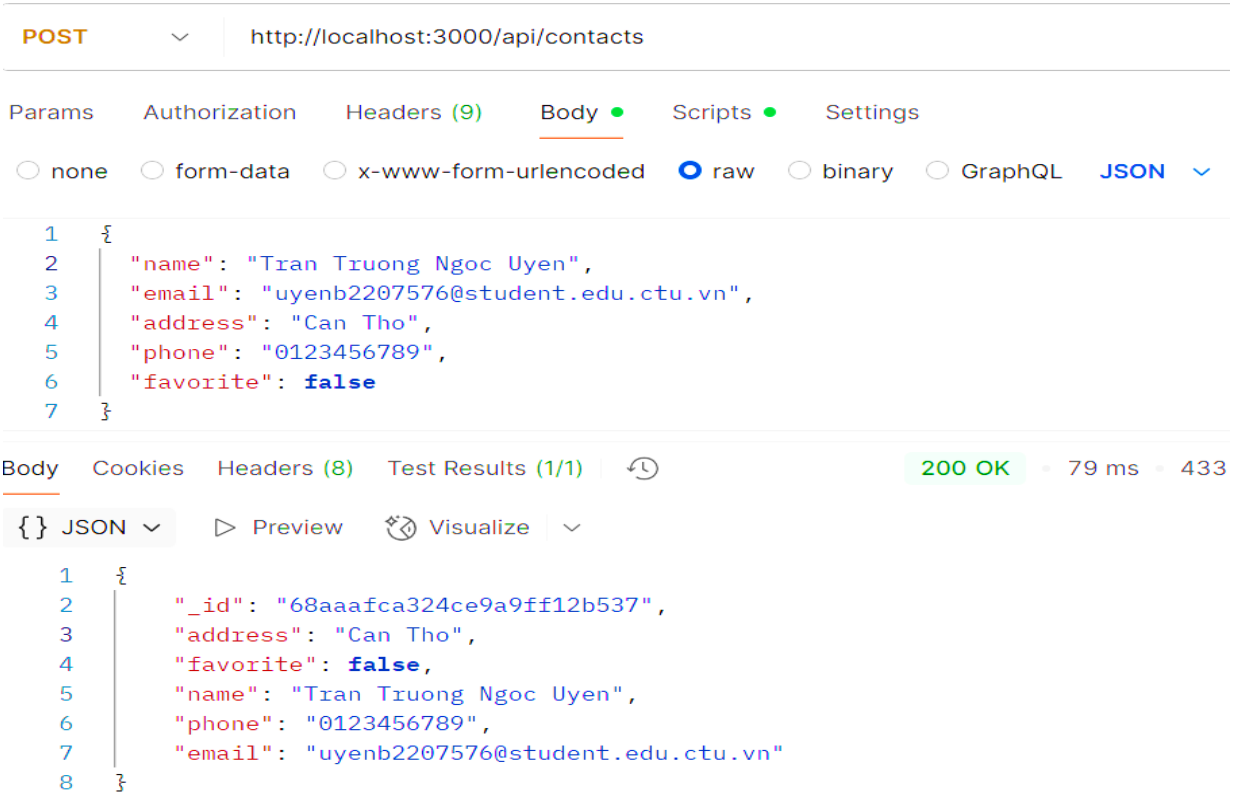
– Tạo 2 phương thức find(), findByName() trong lớp ContactService

```
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(name), $options: "i" },
    });
}
```

Hình 13: Tạo 2 phương thức find(), findByName()

– Kiểm tra với **ứng dụng Postman cho findAll**



Hình 14: Kết quả chạy thành công

3. Cài đặt handler findOne

– Hiệu chỉnh tập tin app/controllers/contact.controller.js

```
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}`
            )
        );
    }
};
```

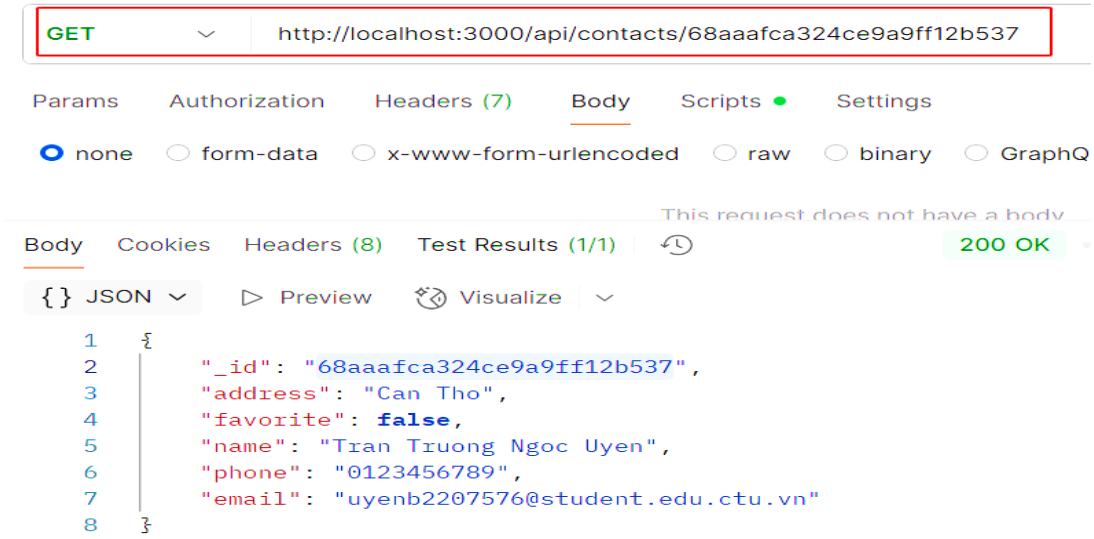
Hình 15: Cài đặt handler findOne

– Tạo phương thức findById() trong lớp ContactService

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

Hình 16: Tạo phương thức findById()

– Kiểm tra với **ứng dụng Postman cho findOne**



Hình 17: Kết quả chạy thành công

4. Cài đặt handler update

– Hiệu chỉnh tập tin app/controllers/contact.controller.js

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

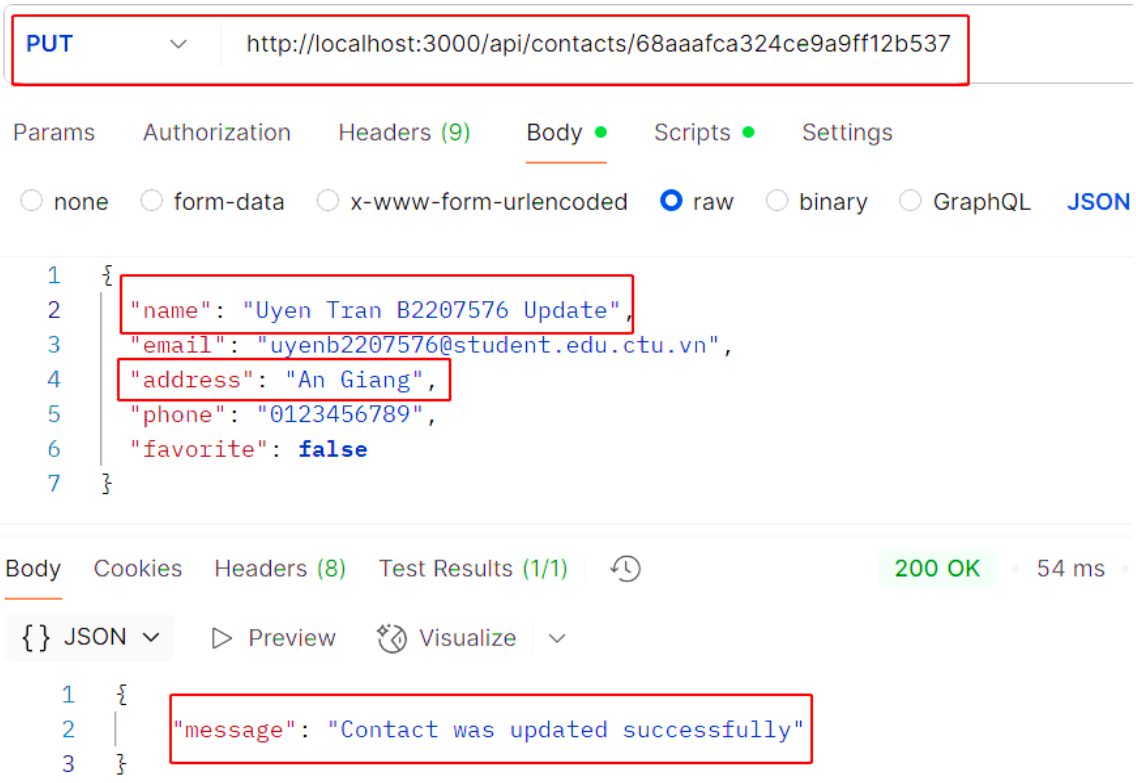
Hình 18: Cài đặt handler update

– Tạo phương thức update() trong lớp ContactService

```
async update(id, payload) {
  const filter = {
    _id: ObjectId.isValid(id) ? new ObjectId(id) : null,
  };
  const update = this.extractContactData(payload);
  const result = await this.Contact.findOneAndUpdate(
    filter,
    { $set: update },
    { returnDocument: "after" }
  );
  return result.value; //return the updated document
}
```

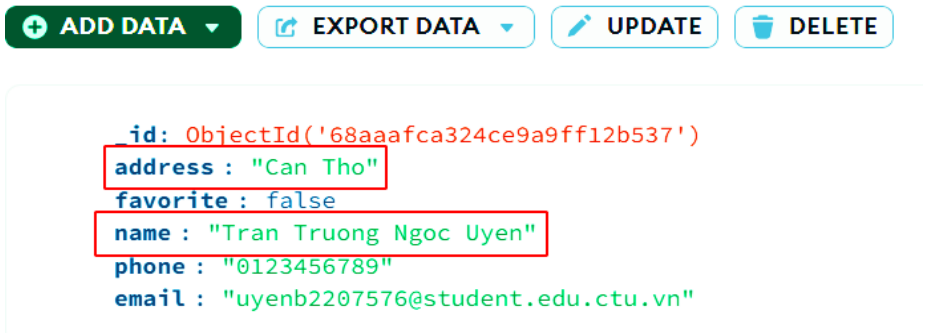
Hình 19: Tạo phương thức update()

– Kiểm tra với **ứng dụng Postman cho update**



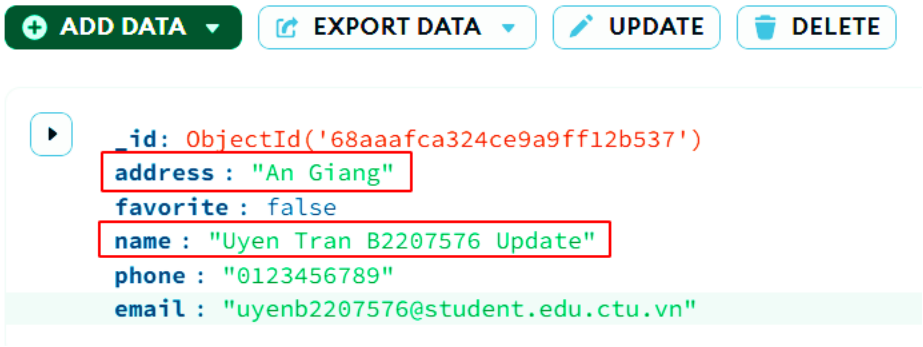
Hình 20: Kết quả chạy thành công

– Trước khi update bên MongoDB



Hình 21: Trước khi update bên MongoDB

– Sau khi update bên MongoDB



Hình 22: Sau khi update bên MongoDB

⇨ Update thành công

5. Cài đặt handler delete

– Hiệu chỉnh tập tin app/controllers/contact.controller.js

```
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({massege: "Contact was deleted successfuly"});
    } catch(error){
        return next(
            new ApiError(
                500,
                `Could not delete contact with id =${req.params.id}`
            )
        );
    }
};
```

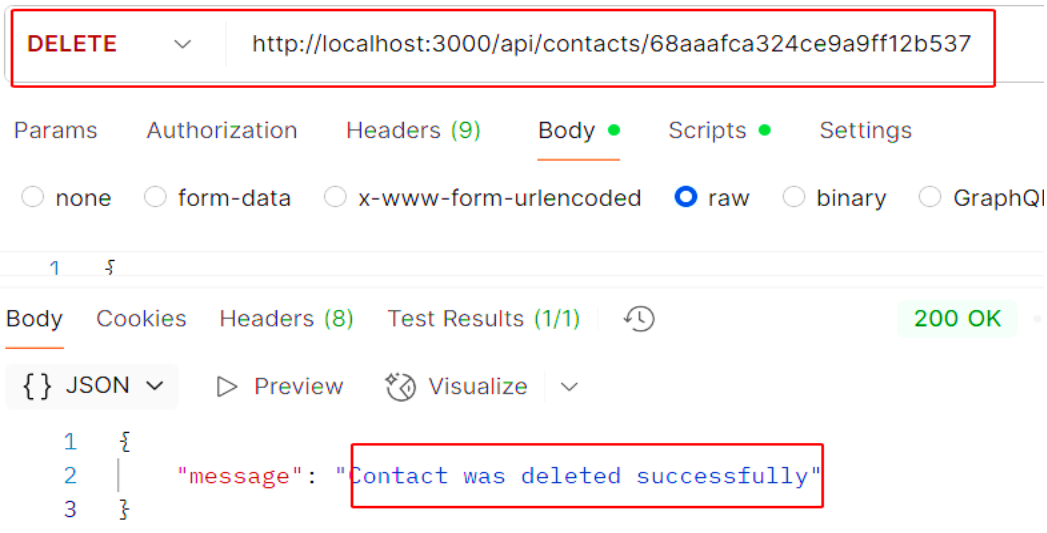
Hình 23: Cài đặt handler delete

– Tạo phương thức delete() trong lớp ContactService

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

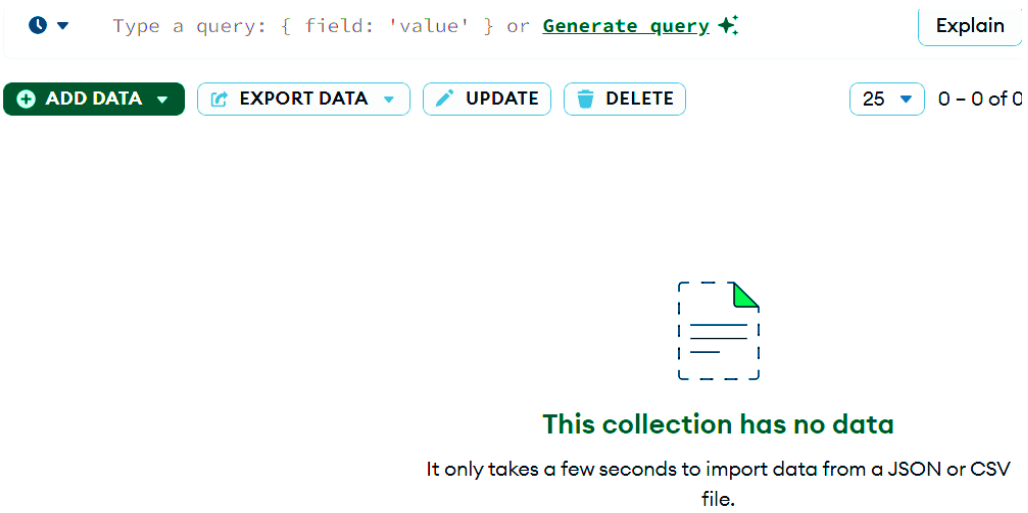
Hình 24: Tạo phương thức create()

– Kiểm tra với **ứng dụng Postman cho delete**



Hình 25: Kết quả chạy thành công

– **Kiểm tra bên MongoDB** được xóa thành công



Hình 26: Kết quả được xóa thành công

6. Cài đặt handler findAllFavorite

– Hiệu chỉnh tập tin app/controllers/contact.controller.js

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

Hình 27: Cài đặt handler findAllFavorite

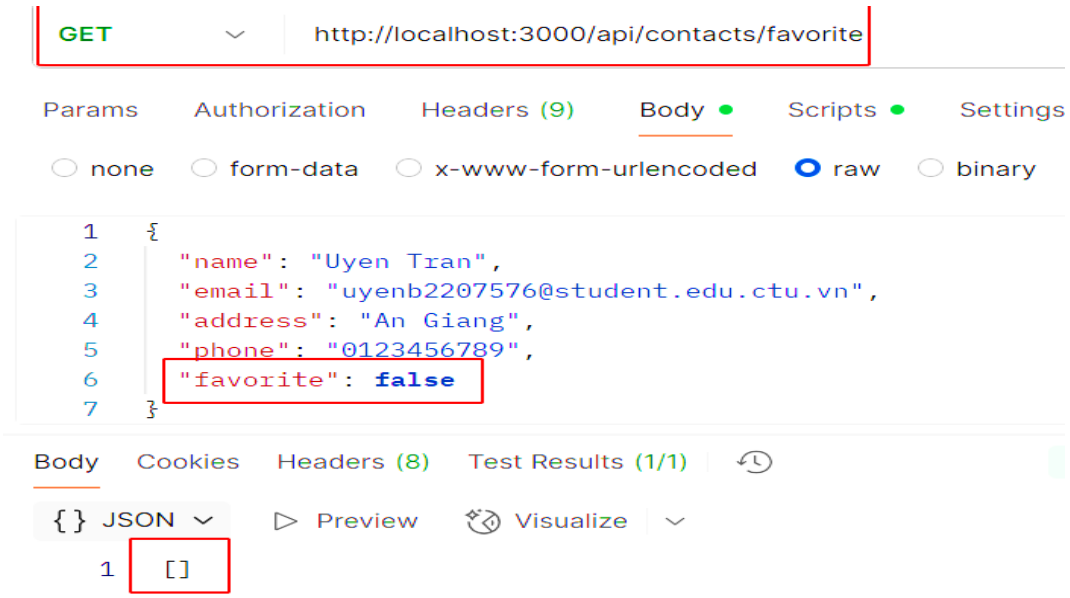
– Tạo phương thức findFavorite() trong lớp ContactService

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

Hình 28: Tạo phương thức findFavorite()

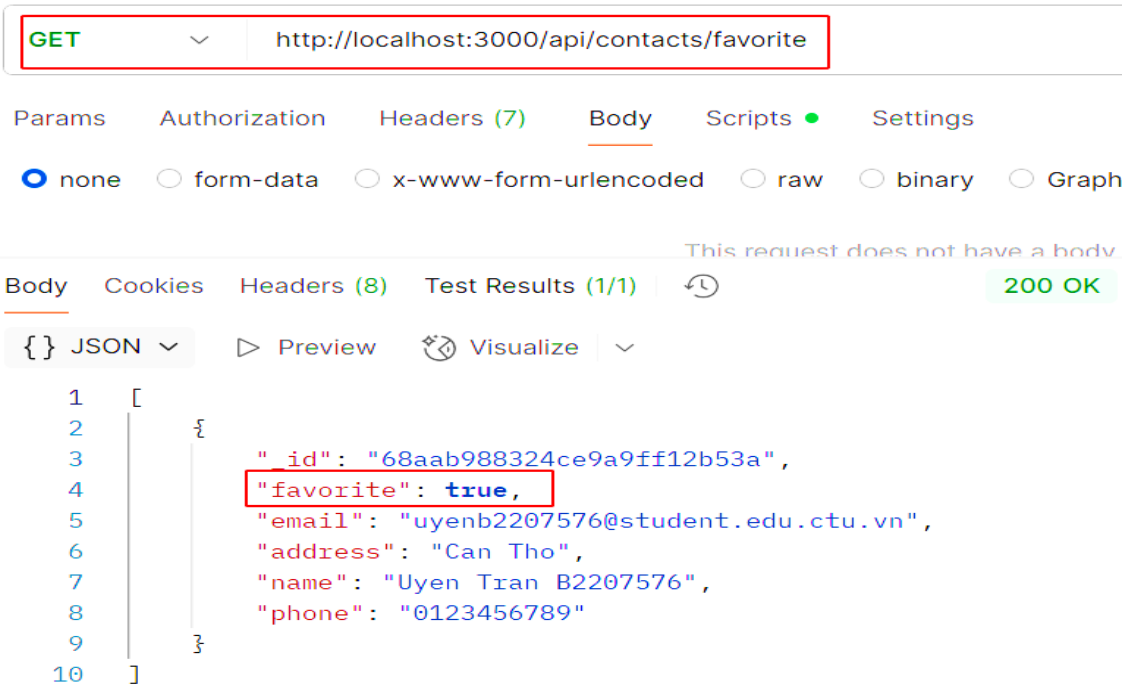
– Kiểm tra với **ứng dụng Postman cho findAllFavorite**

- Khi **favorite = false** thì kết quả sẽ là **rỗng**



Hình 29: Kết quả chạy thành công cho ra rỗng

- Khi **favorite = true** thì kết quả sẽ hiện thị toàn bộ thông tin của người đó



Hình 30: Kết quả hiện thị thành công các thông tin

7. Cài đặt handler deleteAll

– Hiệu chỉnh tập tin app/controllers/contact.controller.js

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

};
```

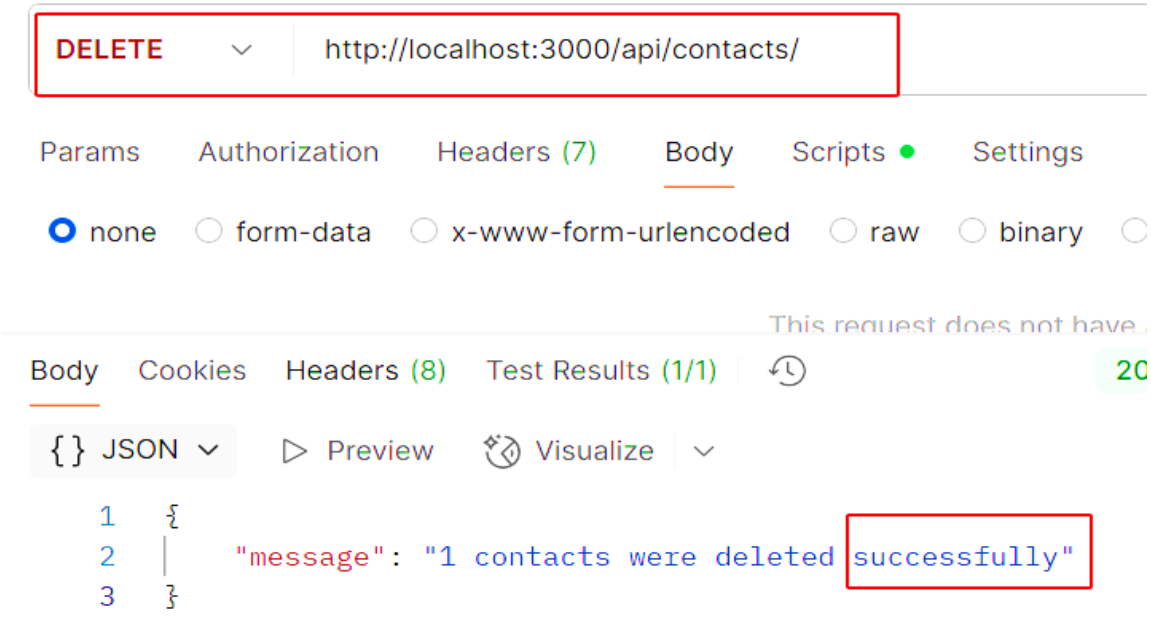
Hình 31: Cài đặt handler deleteAll

– Tạo phương thức deleteAll() trong lớp ContactService

```
  async deleteALL() {
    const result = await this.Contact.deleteMany({});
    return result.deletedCount;
  }
}
module.exports = ContactService;
```

Hình 32: Tạo phương thức deleteAll()

– Kiểm tra với **ứng dụng Postman cho deleteAll**



Hình 33: Kết quả được xóa thành công

– Lưu thay đổi trên vào git

git add -u
git add app/utils app/services

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2\ContactBook-backend> git add -u
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'eslint.config.mjs', LF will be replaced by CRLF the next time Git touches it

PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2\ContactBook-backend> git add app/utils app/services
```

Hình 34: Lưu thay đổi trên vào git

– Kiểm tra lại bằng lệnh **git status**

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2\ContactBook-backend> git status
On branch master
Your branch is up to date with 'backup/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   app.js
        modified:   app/api-error.js
        modified:   app/config/index.js
        modified:   app/controllers/contact.controller.js
        modified:   app/routes/contact.route.js
        new file:   app/services/contact.service.js
        new file:   app/utils/mongodb.util.js
```

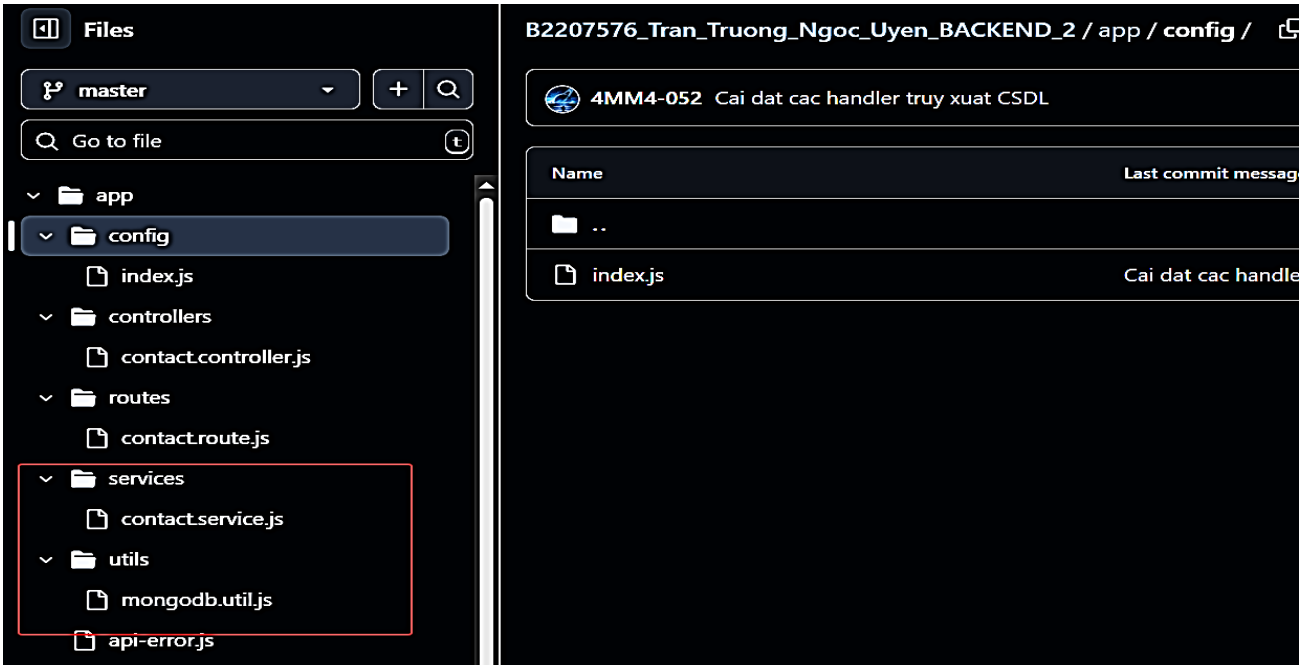
Hình 35: Các tệp đã chạy thành công

– Thực hiện lệnh **git commit -m** để lưu các thay đổi lên git

```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2\ContactBook-backend> git commit -m "Cai dat cac handler
truy xuat CSDL"
[master cb889e4] Cai dat cac handler truy xuat CSDL
 10 files changed, 420 insertions(+), 46 deletions(-)
 create mode 100644 app/services/contact.service.js
 create mode 100644 app/utils/mongodb.util.js
```

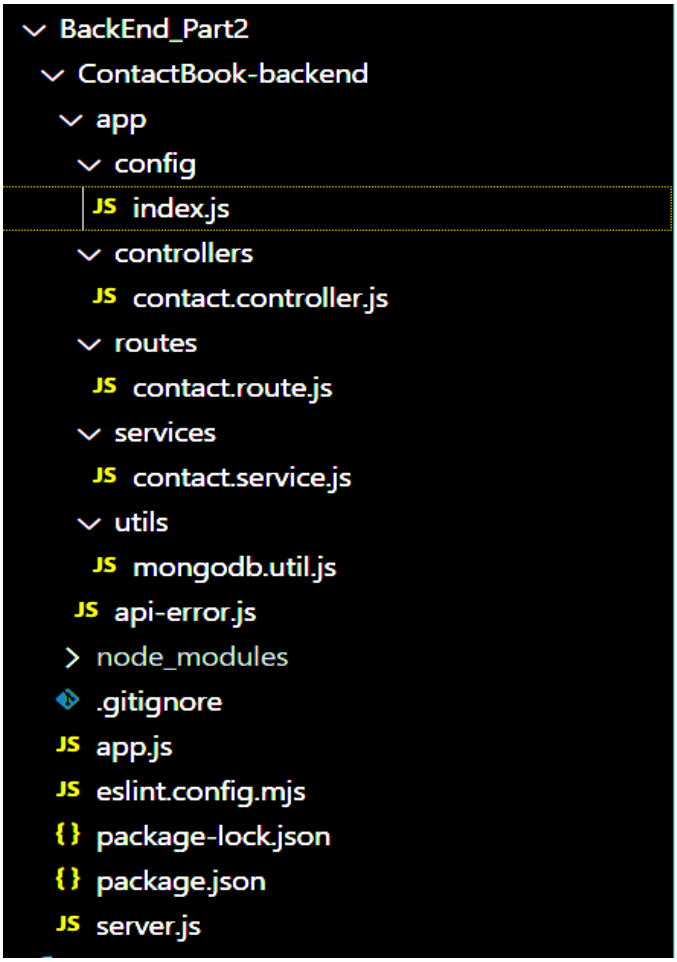
Hình 36: Được tạo thành công

– Kiểm tra lại các tập tin đã được tải lên GitHub sau khi thực hiện lệnh **git push**



Hình 37: Tệp đã tải lên GitHub thành công

– Cấu trúc thư mục dự án đến hiện tại như sau:



Hình 38: Cây thư mục hiện tại

ĐƯỜNG LINK GITHUB:

Link mã nguồn:

https://github.com/4MM4-052/B2207576_Tran_Truong_Ngoc_Uyen_BACKEND_2