

**BỘ GIÁO DỤC VÀ ĐÀO 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 Uyên
MSSV: B2207576
Khóa 48

Cần Thơ, năm 2025

MỤC LỤC

MỤC LỤC	<u>_____</u>	i
DANH MỤC HÌNH	<u>_____</u>	ii
Ứng dụng Contactbook - Backend - Phần 2	<u>_____</u>	1
Bước 0: Cài đặt MongoDB	<u>_____</u>	1
Bước 1: Cài đặt thư viện MongoDB, định nghĩa hàm trợ giúp kết nối và lớp dịch vụ truy xuất cơ sở dữ liệu (CSDL)	<u>_____</u>	1
Bước 2: Cài đặt các handler	<u>_____</u>	4
ĐƯỜNG LINK GITHUB:	<u>_____</u>	16

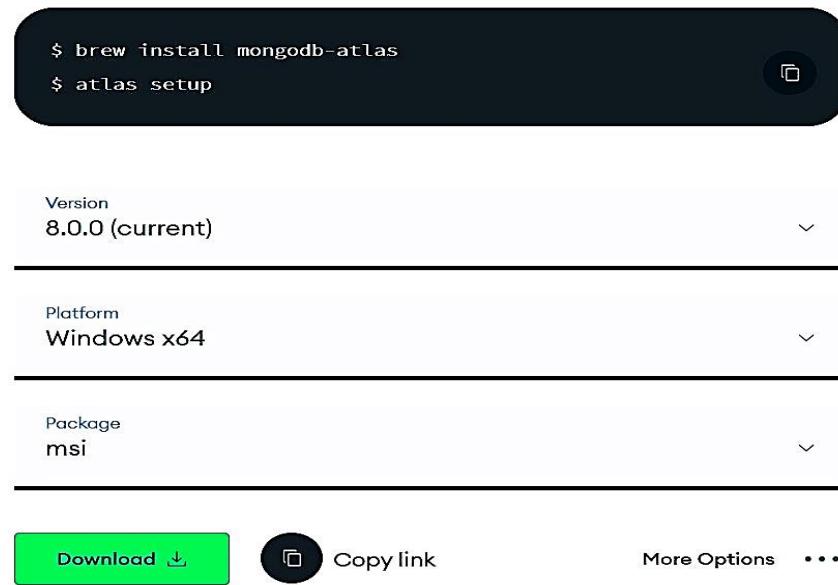
DANH MỤC HÌNH

Hình 1: Cài đặt MongoDB	1
Hình 2: Cài đặt thư viện mongodb	1
Hình 3: Hiệu chỉnh thêm đoạn code trong tập tin index.js	2
Hình 4: Tạo tập tin mongodb.utils.js	2
Hình 5: Thay đổi nội dung tập tin server.js	3
Hình 6: Định nghĩa các lớp dịch vụ ContactServices	3
Hình 7: Cài đặt handler create	4
Hình 8: Tạo phương thức create()	4
Hình 9: Hiệu chỉnh Header "Content-Type: application/json"	5
Hình 10: Đặt dữ liệu JSON trong phần Body	5
Hình 11: Kết quả MongoDB được tạo thành công	6
Hình 12: Cài đặt handler findAll	6
Hình 13: Tạo 2 phương thức find(), findByName()	6
Hình 14: Kết quả chạy thành công	7
Hình 15: Cài đặt handler findOne	7
Hình 16: Tạo phương thức findById()	8
Hình 17: Kết quả chạy thành công	8
Hình 18: Cài đặt handler update	8
Hình 19: Tạo phương thức update()	9
Hình 20: Kết quả chạy thành công	9
Hình 21: Trước khi update bên MongoDB	10
Hình 22: Sau khi update bên MongoDB	10
Hình 23: Cài đặt handler delete	10
Hình 24: Tạo phương thức create()	11
Hình 25: Kết quả chạy thành công	11
Hình 26: Kết quả được xóa thành công	11
Hình 27: Cài đặt handler findAllFavorite	12
Hình 28: Tạo phương thức findFavorite()	12
Hình 29: Kết quả chạy thành công cho ra rỗng	12
Hình 30: Kết quả hiện thị thành công các thông tin	13
Hình 31: Cài đặt handler deleteAll	13
Hình 32: Tạo phương thức deleteAll()	14

Hình 33: Kết quả được xóa thành công	14
Hình 34: Lưu thay đổi trên vào git	14
Hình 35: Các tệp đã chạy thành công	15
Hình 36: Được tạo thành công	15
Hình 37: Tệp đã tải lên GitHub thành công	15
Hình 38: Cây thư mục hiện tại	16

Ứng dụng Contactbook - Backend - Phần 2

Bước 0: Cài đặt MongoDB



Hình 1: Cài đặt MongoDB

Bước 1: Cài đặt thư viện 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 mongodb`

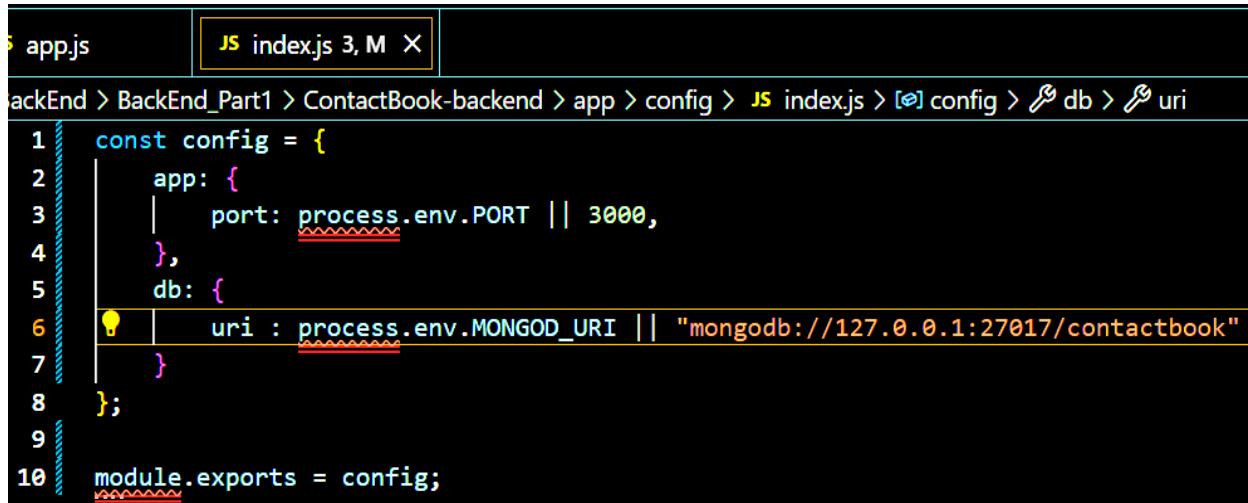
```
PS D:\CT449_PTUD_WEB\THUC_HANH\BackEnd\BackEnd_Part2>ContactBook-backend> npm install mongodb
added 11 packages, and audited 285 packages in 3s

113 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Hình 2: Cài đặt thư viện mongodb

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



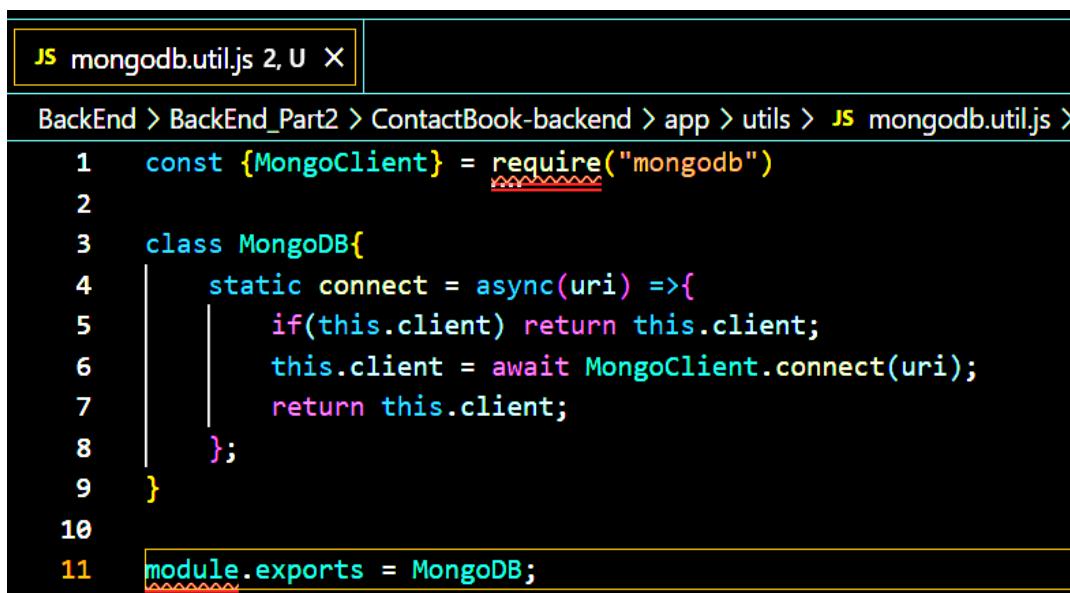
```

$ app.js          JS index.js 3, M X
BackEnd > BackEnd_Part1 > ContactBook-backend > app > config > JS index.js > [o] 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 nghĩa lớp trợ giúp kết nối đến MongoDB: app/utils/mongodb.util.js



```

JS mongodb.util.js 2, U X
BackEnd > BackEnd_Part2 > ContactBook-backend > app > 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;

```

Hình 4: Tạo tập tin mongodb.util.js

- Thay toàn bộ tập tin server.js

```

JS server.js 4, M X
BackEnd > BackEnd_Part2 > ContactBook-backend > JS server.js > startServer
1  const app = require("./app");
2  const config = require("./app/config");
3  const MongoDB = require("./app/utils/mongodb.util");
4
5  async function startServer() {
6    try {
7      await MongoDB.connect(config.db.uri);
8      console.log("Connected to the database!");
9
10     const PORT = config.app.port;
11     app.listen(PORT, () => {
12       |   console.log(`Server is running on port ${PORT}`);
13     });
14   } catch (error) {
15     console.log("Cannot connect to the database!", error);
16     process.exit();
17   }
18 }
19
20 startServer();

```

Hình 5: Thay đổi nội 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

```

1  const { ObjectId } = require("mongodb");
2
3  class ContactService {
4    constructor(client) {
5      this.Contact = client.db().collection("contacts");
6    }
7
8    // Định nghĩa các phương thức truy xuất CSDL sử dụng mongodb API
9  }
10
11 module.exports = ContactService;
12

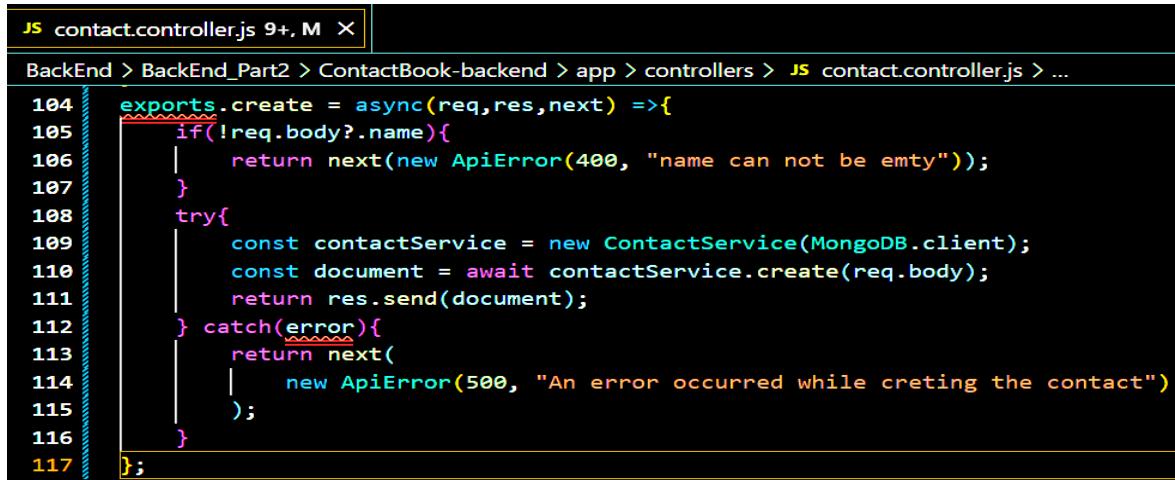
```

Hình 6: Định nghĩa các lớp dịch vụ ContactServices

Bước 2: Cài đặt các handler

1. Cài đặt handler create

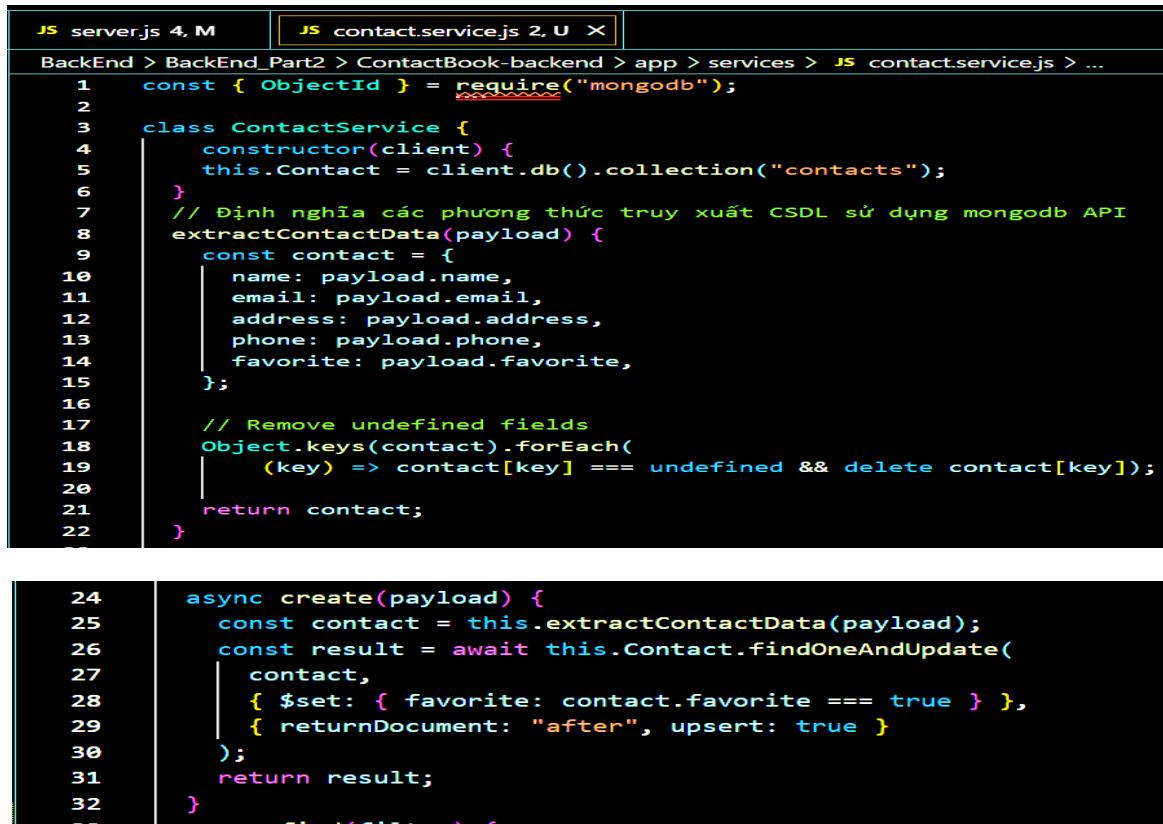
- Hiệu chỉnh tập tin app/controllers/contact.controller.js



```
JS contact.controller.js 9+, M X
BackEnd > BackEnd_Part2 > ContactBook-backend > app > controllers > JS 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 empty"));
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 creating the contact")
115     );
116   }
117};
```

Hình 7: Cài đặt handler create

- Tạo phương thức create() trong lớp ContactService



```
JS server.js 4, M JS contact.service.js 2, U X
BackEnd > BackEnd_Part2 > ContactBook-backend > app > services > JS 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
22      return contact;
23    }
24
25    async create(payload) {
26      const contact = this.extractContactData(payload);
27      const result = await this.Contact.findOneAndUpdate(
28        contact,
29        { $set: { favorite: contact.favorite === true } },
30        { returnDocument: "after", upsert: true }
31      );
32      return result;
33    }
34  }
```

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

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/contacts`. The Headers tab is selected, showing a table with one row. The row has 'Content-Type' as the key and 'application/json' as the value, both highlighted with a red border.

Key	Value	D...	...	Bulk Ed
Content-Type	application/json			
Key	Value	Description		

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

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/contacts`. The Headers tab is selected, showing a table with one row. The row has 'Content-Type' as the key and 'application/json' as the value, both highlighted with a red border.

The Body tab is selected, showing a raw JSON payload:

```

1 {
2   "name": "Tran Truong Ngoc Uyen",
3   "email": "uyenb2207576@student.edu.ctu.vn",
4   "address": "Can Tho",
5   "phone": "0123456789",
6   "favorite": false
7 }
  
```

The response shows a 200 OK status with 79 ms latency and a content length of 433 bytes. The JSON response body is identical to the one sent in the request:

```

1 {
2   "_id": "68aaafca324ce9a9ff12b537",
3   "address": "Can Tho",
4   "favorite": false,
5   "name": "Tran Truong Ngoc Uyen",
6   "phone": "0123456789",
7   "email": "uyenb2207576@student.edu.ctu.vn"
8 }
  
```

Hình 10: Đặt dữ liệu JSON trong phần Body

- Kết quả bên MongoDB:

```
_id: ObjectId('68aaafca324ce9a9ff12b537')
address : "Can Tho"
favorite : false
name : "Tran Truong Ngoc Uyen"
phone : "0123456789"
email : "uyenb2207576@student.edu.ctu.vn"
```

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

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:3000/api/contacts
- Body (JSON):**

```

1  {
2   "name": "Tran Truong Ngoc Uyen",
3   "email": "uyenb2207576@student.edu.ctu.vn",
4   "address": "Can Tho",
5   "phone": "0123456789",
6   "favorite": false
7 }
```

- Response Headers:** 200 OK, 79 ms, 433
- Response Body (JSON):**

```

1  {
2   "_id": "68aaafca324ce9a9ff12b537",
3   "address": "Can Tho",
4   "favorite": false,
5   "name": "Tran Truong Ngoc Uyen",
6   "phone": "0123456789",
7   "email": "uyenb2207576@student.edu.ctu.vn"
8 }
```

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}`
      );
  };
};
```

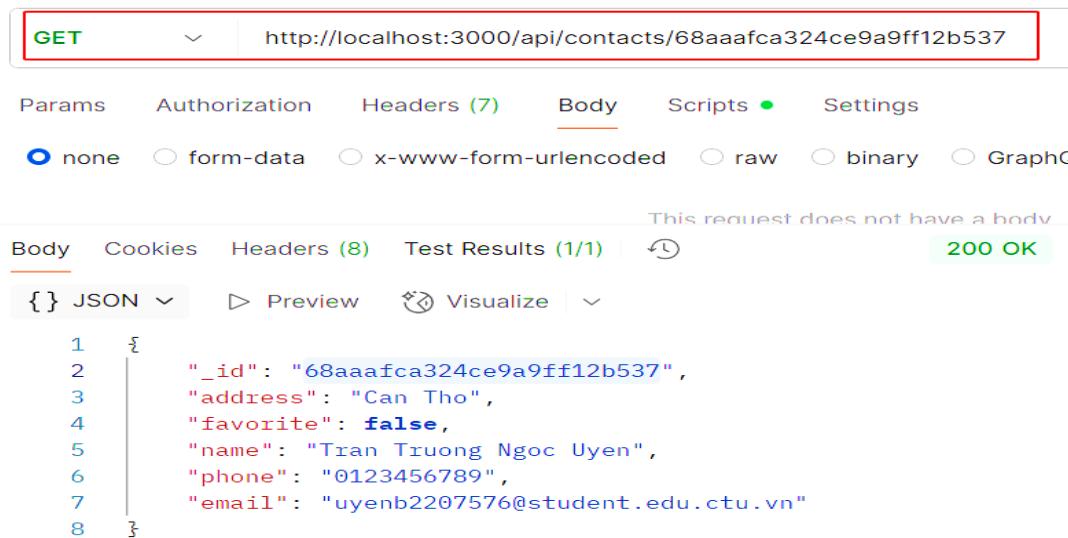
Hình 15: Cài đặt handler findOne

- 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 empty"));
  }
  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 successfully"});
  } 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

The screenshot shows a Postman request configuration for a PUT method. The URL is `http://localhost:3000/api/contacts/68aaafca324ce9a9ff12b537`. The Body tab is selected, showing a JSON payload:

```

1  {
2    "name": "Uyen Tran B2207576 Update",
3    "email": "uyenb2207576@student.edu.ctu.vn",
4    "address": "An Giang",
5    "phone": "0123456789",
6    "favorite": false
7  }

```

The response section shows a 200 OK status with a message: "Contact was updated successfully".

Hình 20: Kết quả chạy thành công

- Trước khi update bên MongoDB

```

_id: ObjectId('68aaafca324ce9a9ff12b537')
address : "Can Tho"
favorite : false
name : "Tran Truong Ngoc Uyen"
phone : "0123456789"
email : "uyenb2207576@student.edu.ctu.vn"
  
```

Hình 21: Trước khi update bên MongoDB

- Sau khi update bên MongoDB

```

▶ _id: ObjectId('68aaafca324ce9a9ff12b537')
address : "An Giang"
favorite : false
name : "Uyen Tran B2207576 Update"
phone : "0123456789"
email : "uyenb2207576@student.edu.ctu.vn"
  
```

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 successfully"});
  } catch(error){
    return next(
      new ApiError(
        500,
        `Could not delete contact with id =${req.params.id}`
      )
    );
  }
};
  
```

Hình 23: Cài đặt handler delete

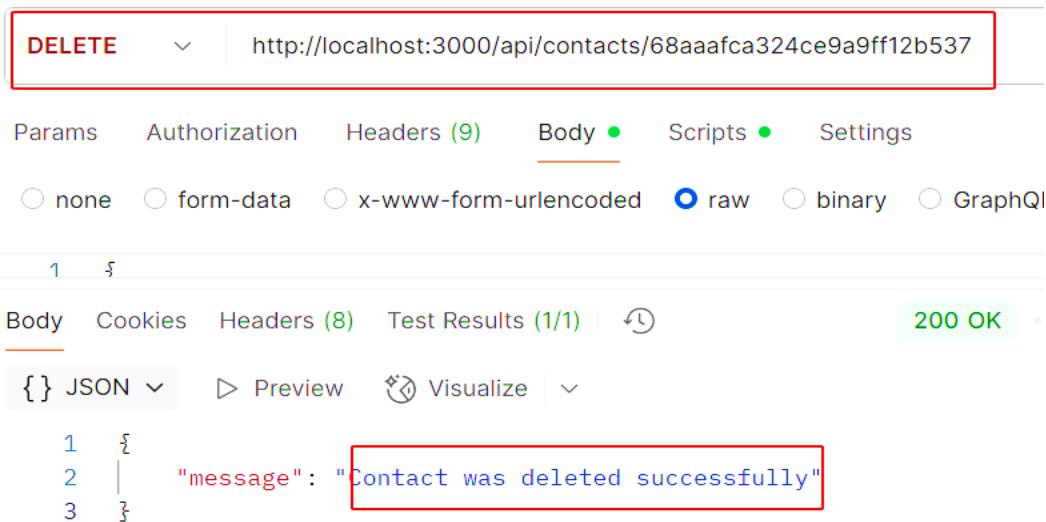
- Tạo phương thức delete() trong lớp ContactService

```

    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



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

The screenshot shows a Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:3000/api/contacts/favorite
- Params:** none
- Headers:** (7)
- Body:** (Raw JSON response shown below)
- Test Results:** (1/1) 200 OK
- Visualize:** Preview of the JSON response.

The JSON response body is:

```

1   [
2     {
3       "_id": "68aab988324ce9a9ff12b53a",
4       "favorite": true,
5       "email": "uyenb2207576@student.edu.ctu.vn",
6       "address": "Can Tho",
7       "name": "Uyen Tran B2207576",
8       "phone": "0123456789"
9     }
10  ]

```

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({
      message: `${deleteCount} contacts were deleted successfully`,
    });
  }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

```

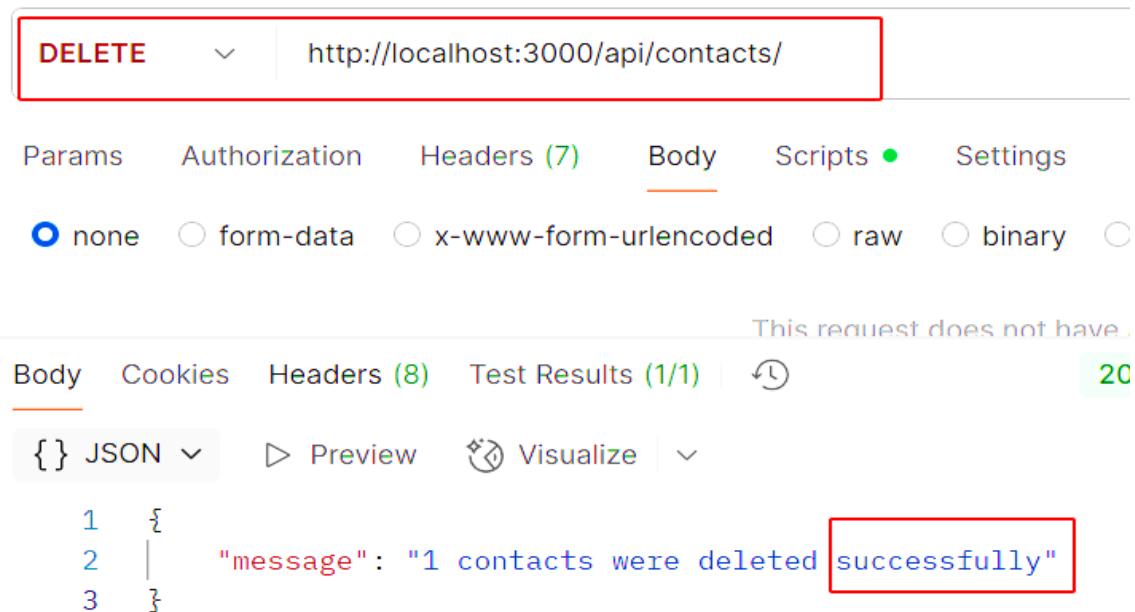
    }
}

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
```

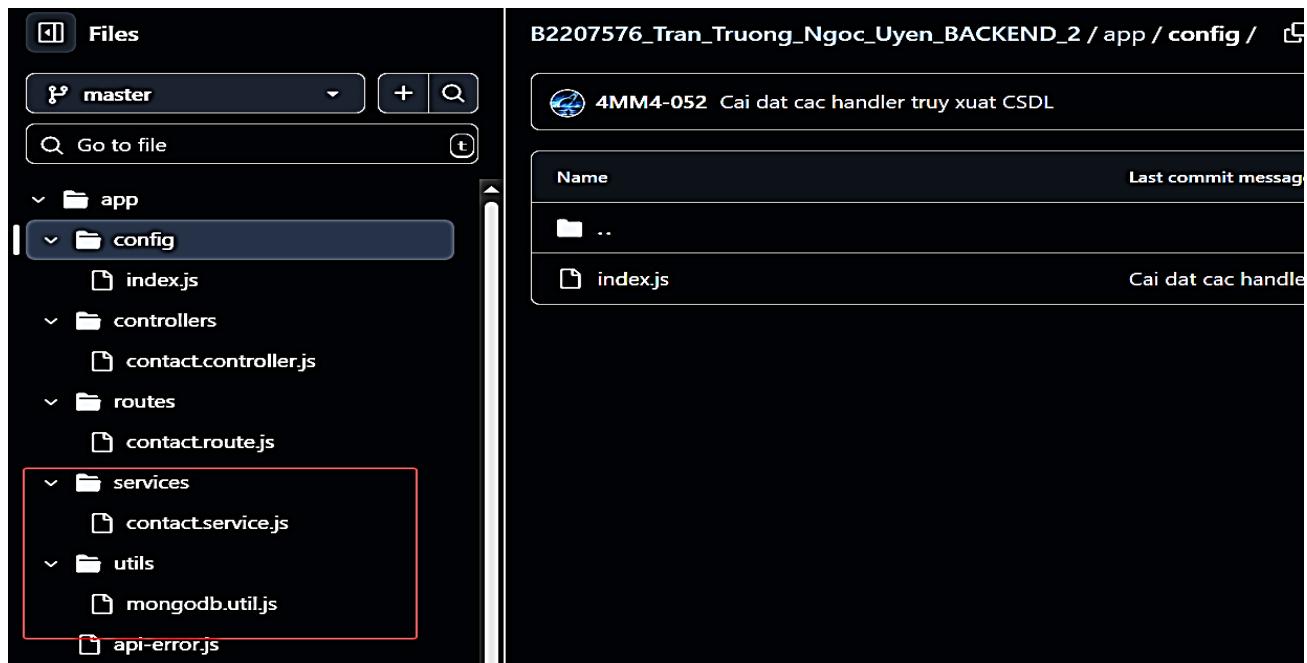
Hình 35: Các tệp đã chạy thành công

- 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:

```
BackEnd_Part2
  ContactBook-backend
    app
      config
        index.js
      controllers
        contact.controller.js
      routes
        contact.route.js
      services
        contact.service.js
      utils
        mongodb.util.js
        api-error.js
    node_modules
    .gitignore
    app.js
    eslint.config.mjs
    package-lock.json
    package.json
    server.js
```

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