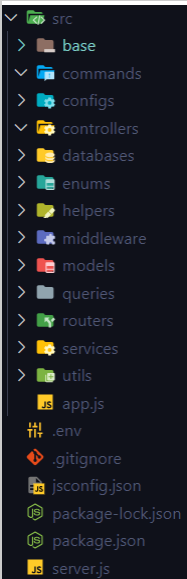


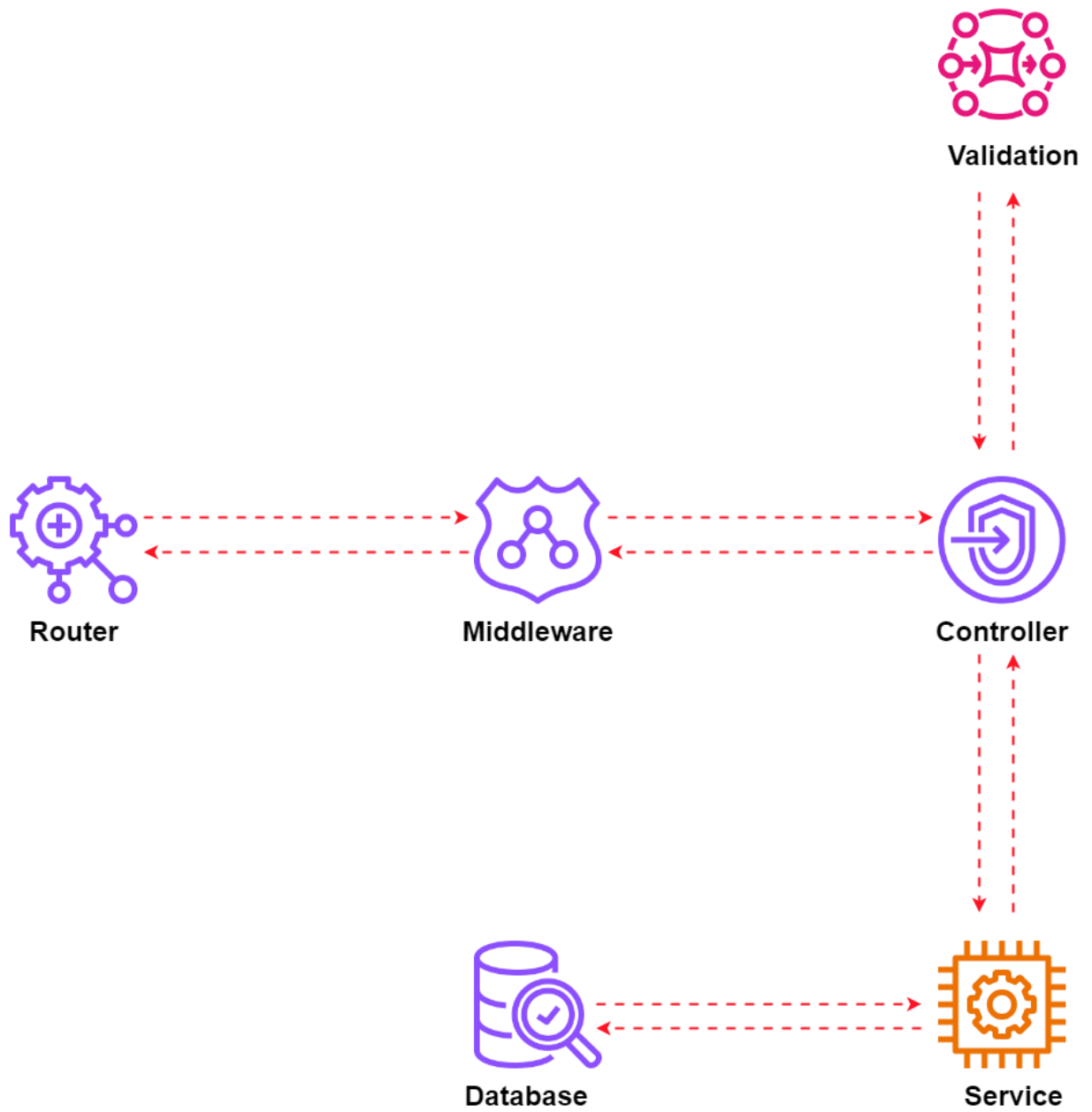
## 1. Convention

Convention	Description
class	Pascal style Example: "AuthenticationService.js"
package	Lowercase style
variable	Camel case
method/function	Camel case
const	Snake uppercase Example: "HTTP_STATUS"

## 2. Structure source code

	server.js	File khởi đầu của ứng dụng, dùng để khởi tạo và chạy server Nodejs.  The starting point of the application, used to initialize and run the Nodejs server.
	package.json	Là file quan trọng nhất, chứa toàn bộ thông tin về dự án, tên dự án, mô tả, tác giả, phiên bản, các package dependencies, scripts... Các thư viện sử dụng trong dự án sẽ được khai báo và quản lý thông qua file này.  The most important file, contains all information about the project, project name, description, author, version, package dependencies, scripts, ... Libraries used in the project will be declared and managed through this file.
	src/app.js	This is the central configuration file of the application, declares routers, connects to the database, registers and configures middlewares, services, declares constants, environments, ... is the place to initialize and configure the entire application.
	src/services	Folder chứa các services dùng để định nghĩa các business logic, xử lý nghiệp vụ chính của hệ thống. Các services sẽ được sử dụng bởi các routers để xử lý dữ liệu và nghiệp vụ.  Folder containing services used to define business logic, process the main business of the system. Services will be used by routers to process data and business.
	src/routers	Đây là nơi định nghĩa tất cả các routers của ứng dụng, các routers sẽ nhận và xử lý các request từ client gửi lên. Mỗi router sẽ đăng ký và xử lý các APIs riêng biệt.  This is where all the routers of the application are defined, the routers will receive and process requests sent from the client. Each router will register and process separate APIs.
	src/configs	Chứa các file cấu hình các service, database, credentials, environments, constants,... của ứng dụng. Các file cấu hình có thể được tách riêng cho từng môi trường khác nhau.  Contains configuration files for services, databases, credentials, environments, constants, ... of the application. Configuration files can be separated for different environments.
	src/database	Chứa các file để khởi tạo và quản lý kết nối tới các database, thao tác với các bảng trong database. Kết nối và thao tác với database sẽ được tách riêng thành các data access layer.  Contains files to initialize and manage connections to databases, manipulate tables in databases. Database connection and manipulation will be separated into data access layers.
	src/enums	Định nghĩa các constants và enum dùng chung trong toàn bộ ứng dụng, như các mã lỗi, mã trạng thái, các trạng thái app,... giúp tránh magic numbers và dễ quản lý hơn.  Defines constants and enums used throughout the application, such as error codes, status codes, app states, ... helps to avoid magic numbers and easier to manage.
	src/helpers	Là nơi để định nghĩa các hàm tiện ích, các tool function dùng chung để xử lý logic, validate, formatter,... có thể sử dụng lại ở nhiều nơi trong code.  Is where to define utility functions, tool functions used to process logic, validate, formatter, ... can be reused in many places in the code.
	src/models	Định nghĩa các model dùng để ánh xạ các bảng trong database thành các object trong code. Các model sẽ dùng trong các services, routers để thao tác với dữ liệu.  Defines models to map database tables to objects in code. Models will be used in services, routers to manipulate data.

src/middleware	<p>Chứa các middleware dùng để xử lý các request từ client. Các middleware có thể xử lý authentication, authorization, validation, logging, handle errors,... rất quan trọng trong việc xử lý request.</p> <p>Contains middleware used to process requests from clients. Middleware can handle authentication, authorization, validation, logging, handle errors, ... very important in processing requests.</p>
src/utils	<p>Chứa các hàm tiện ích khác, có thể sử dụng lại cho nhiều mục đích khác nhau trong ứng dụng. Các utils function có thể là parse, convert, validate,...</p> <p>Contains other utility functions, which can be reused for various purposes in the application. Utils functions can be parse, convert, validate, ...</p>
src/base	<p>Định nghĩa các lớp base chung cho các đối tượng khác kế thừa và sử dụng. Ví dụ các base controller, base service, base model,...</p> <p>Defines common base classes for other objects to inherit and use. For example, base controllers, base services, base models, ...</p>



## Conntroller code style

```
const ExampleService = require("@services/ExampleService");
const express = require("express")
const Joi = require('joi');

class ExampleController {
  /**
   *
   * @param {express.Request} request
   * @param {express.Response} response
   */
  static get(request, response) {
    const { query } = request
    const schema = Joi.object(
      {
        id: Joi.string().required()
      }
    )

    //Validation request
    schema.validate(query)

    ExampleService.get().send(response)
  }
}

module.exports = ExampleController
```

Service case throw exception

```
const { HTTP_CODE, HTTP_REASON } =
require("@src/base/enums/HttpStatus");
const ServiceException =
require("@src/base/services/ServiceException");

class ExampleService {
  /**
   *
   * @returns Simple information
   */
  static get() {

    //Logic query database

    throw ServiceException.builder(
      HTTP_CODE.NOT_ACCEPTABLE,
      HTTP_CODE.NOT_ACCEPTABLE,
      HTTP_REASON.NOT_ACCEPTABLE,
      null,
      null
    )

  }
}
```

## Service return result

```
const { HTTP_CODE } = require("@src/base/enums/HttpStatus");
const ServiceJsonResult =
  require("@src/base/services/ServiceJsonResult");

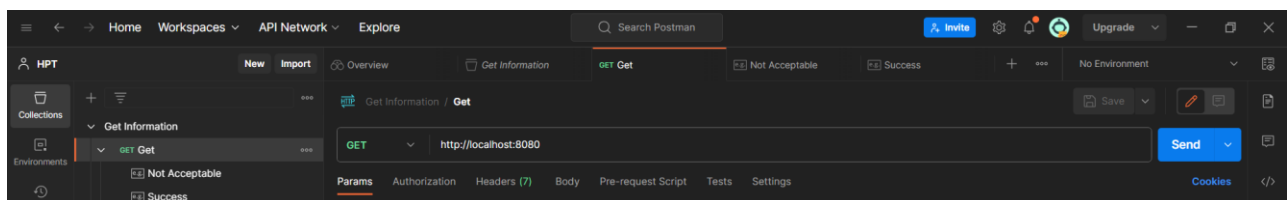
class ExampleService {
  /**
   *
   * @returns Simple information
   */
  static get() {

    //Logic query database

    return ServiceJsonResult.builder(
      HTTP_CODE.OK,
      HTTP_CODE.OK,
      {
        name: "R&D",
        location: "HCM"
      },
      "Get info success",
      {
        address: "HPT HCM"
      }
    )
  }
}

module.exports = ExampleService
```

## Post man document



HomeWorkspacesAPI NetworkExplore

Search Postman

Invite

Upgrade

HPT

NewImport

OverviewGet InformationGET GetNot AcceptableSuccess

No Environment

Get Information / Get / Not Acceptable

Save

GET

http://localhost:8080

Try

Params

Headers

Body

Query Params

Key	Value	Description
Key	Value	Description

Body

Headers (20)

Status Code 406 Not Acceptable

Pretty

Raw

Preview

JSON

```
1 {
2   "error": true,
3   "success": false,
4   "code": 406,
5   "httpStatus": 406,
6   "message": "Not Acceptable."
7 }
```

HomeWorkspacesAPI NetworkExplore

Search Postman

Invite

Upgrade

HPT

NewImport

OverviewGet InformationGET GetNot AcceptableSuccess

No Environment

Get Information / Get / Success

Save

GET

http://localhost:8080

Try

Params

Headers

Body

Query Params

Key	Value	Description
Key	Value	Description

Body

Headers (20)

Status Code 200 OK

Pretty

Raw

Preview

JSON

```
1 {
2   "error": false,
3   "success": true,
4   "code": 200,
5   "httpStatus": 200,
6   "message": "Get info success",
7   "payload": {
8     "name": "trungna",
9     "position": "Backend"
10  },
11   "meta": null
12 }
```

Documentation

## Get Information

Demo document generate via postman

### GET Get Simple Information

[Open request →](#)

`http://localhost:8080`

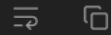
Simple docs via postman

## Example

Not Acceptable ▾

### Request

cURL



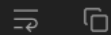
```
curl --location 'http://localhost:8080'
```

### Response

Body Headers (20)

406 NOT ACCEPTABLE


json



```
{
  "error": true,
  "success": false,
  "code": 406,
  "httpStatus": 406,
  "message": "Not Acceptable."
}
```



## Example

Success 

### Request

cURL



```
curl --location 'http://localhost:8080'
```

### Response

Body

Headers (20)

200 OK

json



```
{
  "error": false,
  "success": true,
  "code": 200,
  "httpStatus": 200,
  "message": "Get info success",
  "payload": {
    "name": "trungnm",
    "position": "Backend"
  }
}
```

View More