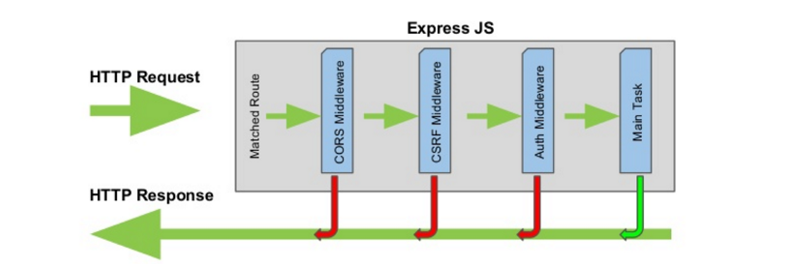
**Bài 1 : Tổng quan Express**

*Phân biệt Library và Framework ?*

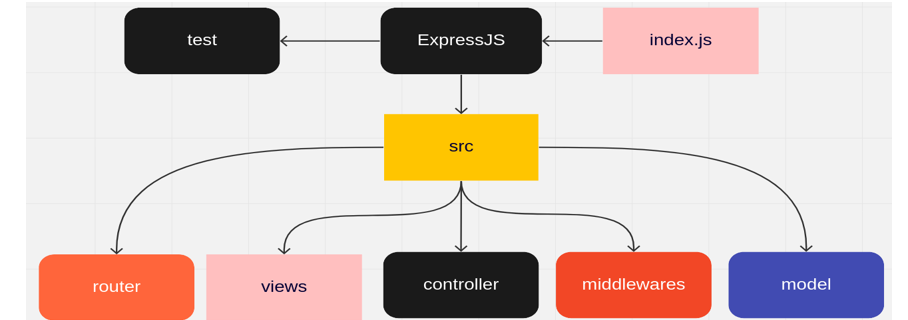
* Library : là 1 tập hợp các chức năng (functions) hoặc các lớp (class) được viết sẵn => Mỗi function hoặc class để phục vụ cho 1 công việc nào đó.
* Thư viện không đi kèm cấu trúc mà chỉ là một bộ chức năng phục vụ cho lập trình
* Vd : fs, file-upload, qs, …
* Framework : Là 1 tập hợp các library đã được đóng gói để hỗ trợ phát triển ứng dụng dựa trên framework đó và nó cung cấp nguyên tắc và cấu trúc ứng dụng mà ta phải tuân thủ theo nó (đảo ngược quyền điều khiển).
* Vd : Trong express tích hợp sẵn http , router , file , session , cookie, ….

*Express sử dụng làm gì ?*

* Express đón các request browser gửi đến => xử lý logic => và phản hồi response cho người dùng.



*Các thành phần chính của dự án express ?*

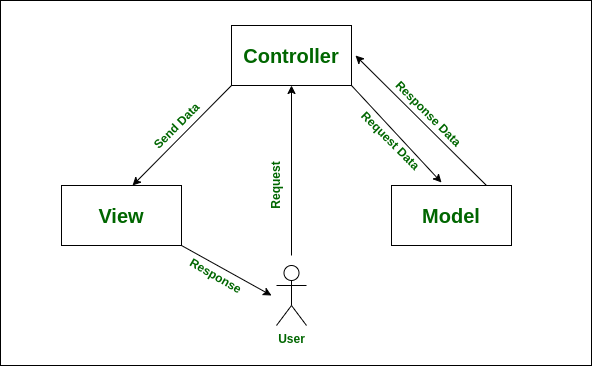


File index : Chứa các câu lệnh cần thiết để thực thi dự án

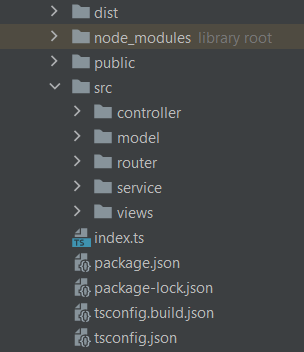
File test : Chứa các đoạn code phục vụ cho testing

Thư mục src : chứa toàn bộ source code của dự án và được chia theo mô hình khá phổ biến hiện nay là MVC.

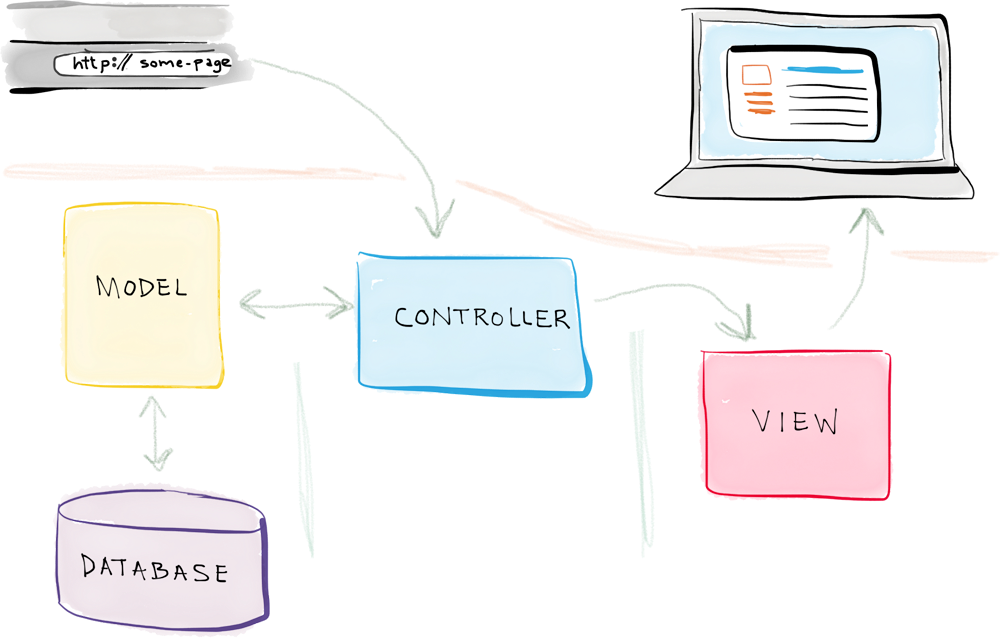
* M (Model) : Đại diện cho dữ liệu và xử lý dữ liệu
* V (View) : đại diện cho giao diện người dùng, hiển thị dữ liệu cho người dùng và cập nhật trạng thái của giao diện người dùng
* C (Controller): đại diện cho tầng trung gian giữa Model và View. Nó nhận lệnh từ view tương tác với model để lấy hoặc thực hiện thao tác với dữ liệu sau đó cập nhật lên view.



Cấu trúc dự án :

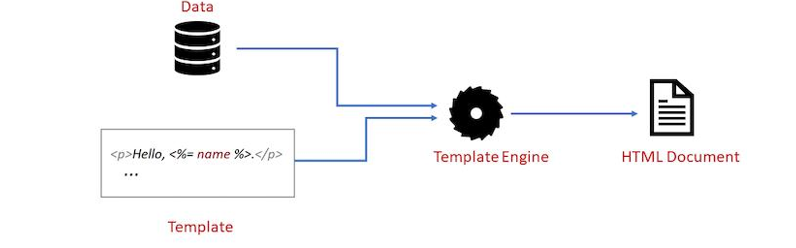


*Mô hình ứng dụng web ?*



*Template engine Ejs ?*

* EJS (Embedded Javascript) : sử dụng để nhúng Js để nhúng vào code HTML để tạo ra mã đánh dấu (markup) - từ đó tạo ra mã đánh dấu khi chạy.



Cài đặt :

* npm i ejs
* Cài đặt thuộc tính cấu hình cho ứng dụng express :

app.set('views', './src/views');

app.set('view engine', 'ejs');

* Khi sử dụng trong file ejs :

<% %> : cho phép viết Javascript trong file .ejs

<%= %> : để hiển thị giá trị của 1 biến hoặc biểu thức

<%- %> : cho phép hiển thị nội dung HTML mà không bị HTML mã hóa

<%# %> : để comment code

Ví dụ :

<div>

<div>

<p>

<% var myName="John Does" %>

Hello <%-myName%>!

</p>

</div>

</div>

*Routing ?*

* Nói đến việc xác định cách mà ứng dụng phản hồi yêu cầu của người dùng đến một điểm cuối cụ thể.

**app.get("/", (req, res) => {**

**res.send("hello world")**

**})**

**=> Kết quả trả về sẽ khi truy cập vào ‘/’ sẽ là page có đoạn text “hello world”**

*Cách xử lý request ?*

* Request là đối tượng đại diện cho HTTP Request, chứa toàn bộ thông tin mà client gửi lên server.
* Có các thuộc tính req.params, req.body,..
* Để có thể đọc được req.body ta sẽ cài thêm thư viện body-parser và sử dụng trong index.ts như sau :

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

* Để bắt được parameter sử dụng như sau :

**app.get('/user/:id', function (req, res) {**

**res.send('user ' + req.params.id)**

**})**

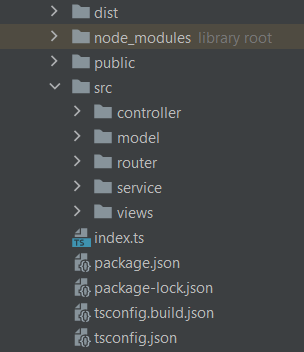
*Xử lý static file ?*

* Là các file HTML, CSS, ảnh, Javascript và chúng có nội dung không thay đổi khi có request tới nó.
* Cú pháp :

**app.use(express.static('path'));**

* (**\***path : là vị trí folder chứa các file tĩnh cần đọc và theo cấu trúc thư mục của chúng ta là ‘./public)

Cấu trúc dự án :



Copy Config : <https://james.codegym.vn/mod/assign/view.php?id=16218>

* Vì khi viết code trong express chúng ta sẽ viết bằng typescript và viết theo mô hình MVC nên ta sẽ cần thêm các file cấu hình
* Các bước tạo :
* Bước 1 : Tải các module cần thiết về dự án

npm init

npm i -D typescript tsc tsc-watch rimraf @types/express

npm i express body-parser multer ejs

npm i typeorm mysql reflect-metadata --save

* Bước 2 : Tạo file tsconfig.json để cấu hình typescript có nội dung như sau

{

"compilerOptions": {

"module": "commonjs",

"declaration": true,

"esModuleInterop": true,

"removeComments": true,

"emitDecoratorMetadata": true,

"experimentalDecorators": true,

"target": "es2017",

"sourceMap": true,

"outDir": "./dist",

"baseUrl": "./",

"incremental": true,

"skipLibCheck": true,

"noUnusedParameters": false,

"noUnusedLocals": false,

},

"exclude": ["node\_modules", "dist"]

}

* Bước 3 : Tạo file tsconfig.build.json có nội dung như sau :

{

"extends": "./tsconfig.json",

"exclude": ["node\_modules", "test", "dist", "\*\*/\*spec.ts"]

}

* Bước 4 : Thêm script để biên dịch chương trình => Trong file package.json => Mục script thêm đoạn code sau :

"build": "rimraf dist && tsc -p tsconfig.build.json",

"start:dev": "rimraf dist && tsc-watch -p tsconfig.build.json --onSuccess \"node dist/index.js\""

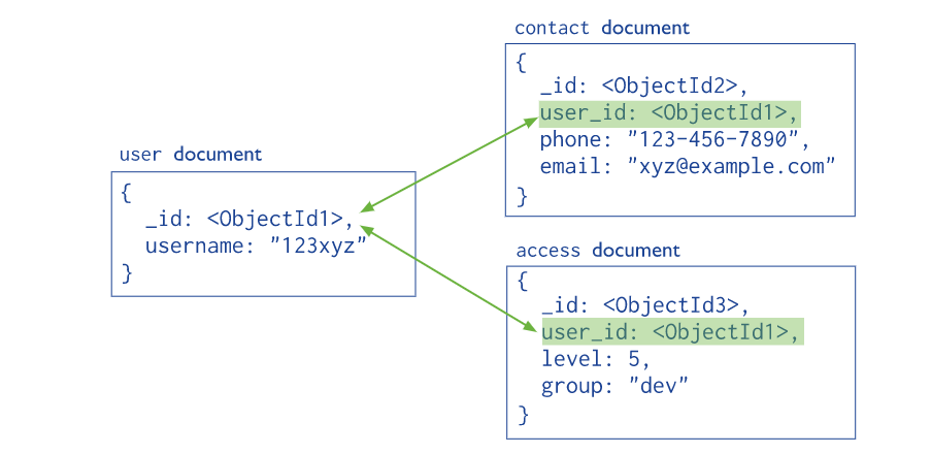
=> câu lệnh để chạy dự án sẽ là npm run start:dev

* Bước 5 : Tạo các folder như src để chứa code dự án, public để chứa các file tĩnh, index.ts sẽ là file gốc để chạy dự án.

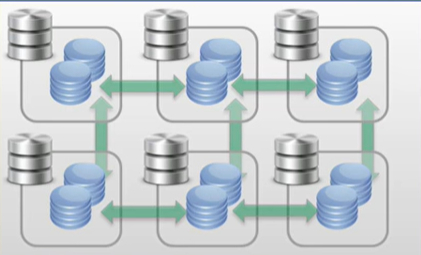
**NoSql và Mongoose DB**

*Documentation :* [MongoDB CRUD Operations — MongoDB Manual](https://www.mongodb.com/docs/manual/crud/)

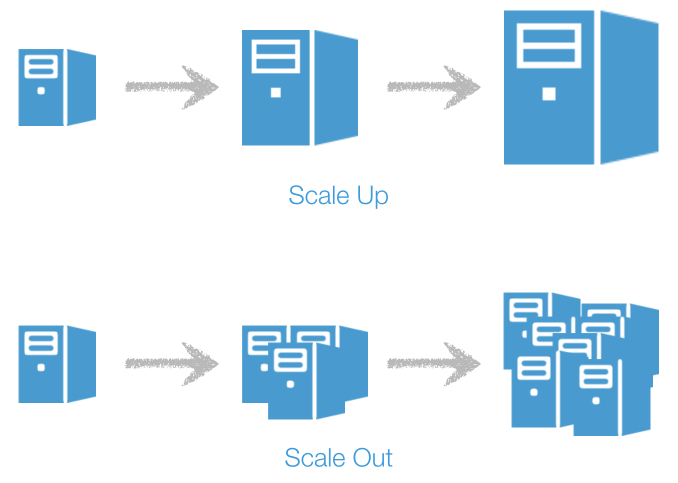
* Khái niệm :
* Non - Relational hoặc Not Only SQL



* Không sử dụng mô hình dữ liệu quan hệ (RDBMs)
* Mô hình lưu trữ { “key” : “value” }
* Hệ thống lưu trữ phân tán

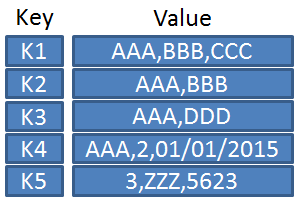


=> Là hệ cơ sở không ràng buộc, phân tán, mã nguồn mở, khả năng mở rộng theo chiều ngang, có thể chứa hàng petabytes, độ chịu tải và chịu lỗi cao, yêu cầu về tài nguyên phần cứng thấp.

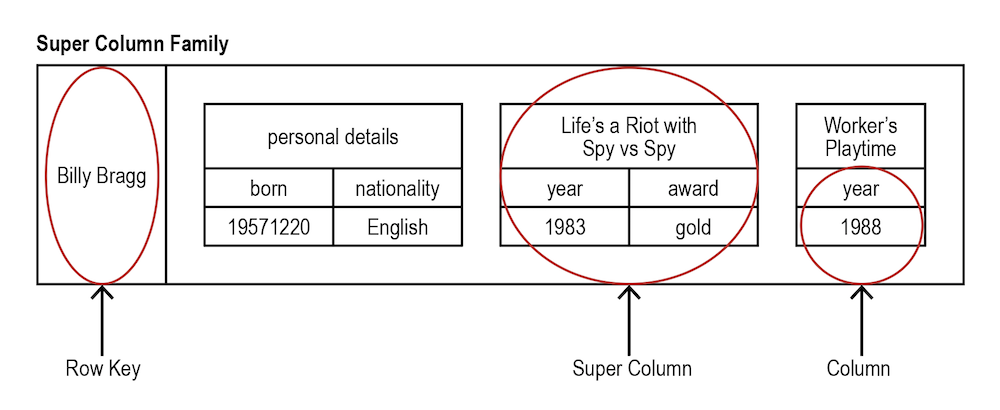


* **Phân loại**

1. Key-values Stores



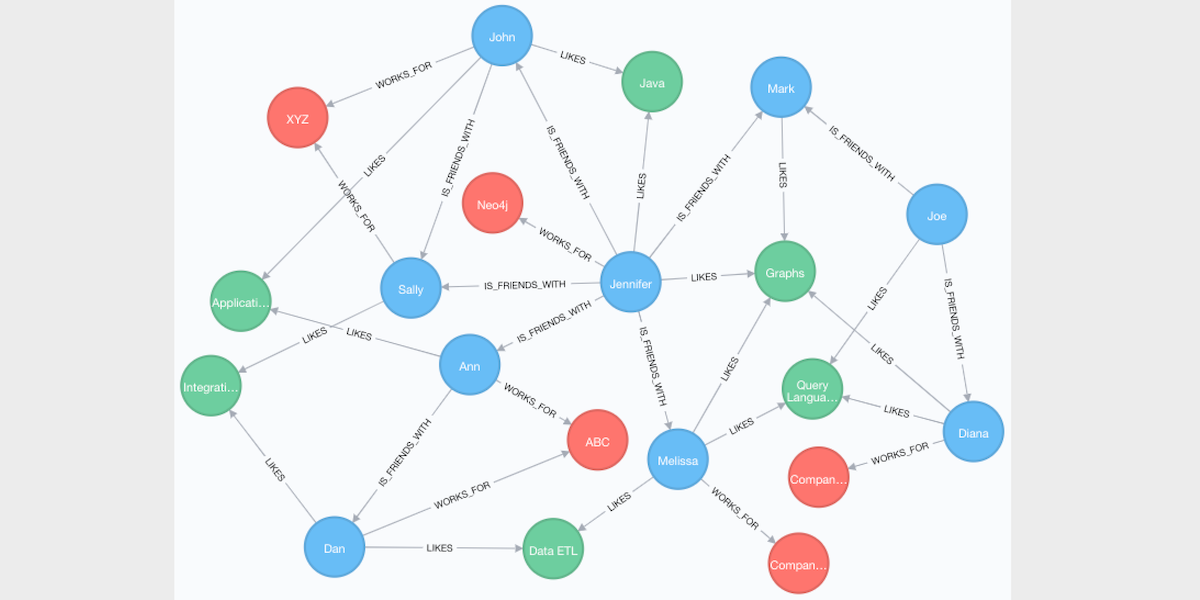
2. Column Family Stores



3. Document Databases



4. Graph Databases



***So sánh NoSQL và SQL***

| Tính năng | SQL | No SQL |
| --- | --- | --- |
| Cấu trúc dữ liệu | Sử dụng mô hình quan hệ với các bảng hàng cột để lưu trữ và truy vấn dữ liệu | NoSql có thể sử dụng nhiều cấu trúc dạng khác nhau như document, graph hoặc column family để lưu trữ và truy vấn dữ liệu |
| Ngôn ngữ truy vấn | SQL sử dụng ngôn ngữ có cấu trúc (structured query language) để truy vấn dữ liệu | Sử dụng các loại ngôn ngữ truy vấn khác nhau tùy thuộc vào cấu trúc dữ liệu. |
| Khả năng mở rộng | SQL có khả năng mở rộng thấp hơn vì nó được thiết kế để hoạt động trên 1 máy chủ duy nhất và có sự ràng buộc giữa các bảng | NoSql có khả năng mở rộng tốt hơn SQL vì nó có thể phân tán trên nhiều máy chủ và được thiết kế để xử lý lượng dữ liệu lớn |
| Tính nhất quán dữ liệu | Tuân thủ các nguyên tắc ACID để đảm bảo tính nhất quán dữ liệu trong các giao dịch | Sử dụng nguyên tắc BASE để đảm bảo tính khả dụng và nhất quán dữ liệu trong môi trường phân tán |
| Ứng dụng | Sử dụng với các ứng dụng có cấu trúc rõ ràng và chặt chẽ với tần suất truy vấn cao như : ứng dụng tài chính, ngân hàng, quản lý khách hàng, … | Thường sử dụng với các ứng dụng không cố định, không có cấu trúc rõ ràng, tính mở rộng cao và tần suất truy vấn thấp : mạng xã hội, diễn đàn,... |

Câu hỏi :

* NoSQL có thể thay thế SQL ?

https://www.youtube.com/watch?v=RP4enc7z1Lw

So sánh :

* Cơ sở dữ liệu là cách tổ chức và lưu trữ và truy xuất dữ liệu hiệu quả => Cơ sở dữ liệu quan hệ là tổ chức dữ liệu theo các bảng có quan hệ với nhau
* Sql Db : Table , Row , Column , Joins , Primary Key
* NoSql Db : Collection , Document ,Field , Embedded Document , Linking , Primary Key

**ORM**

* Object Relational Mapping => Kỹ thuật ánh xạ giúp các đối tượng có thể tương tác với các bảng trong cơ sở dữ liệu
* Vd : Trong ts có đối tượng là

Id:

Name

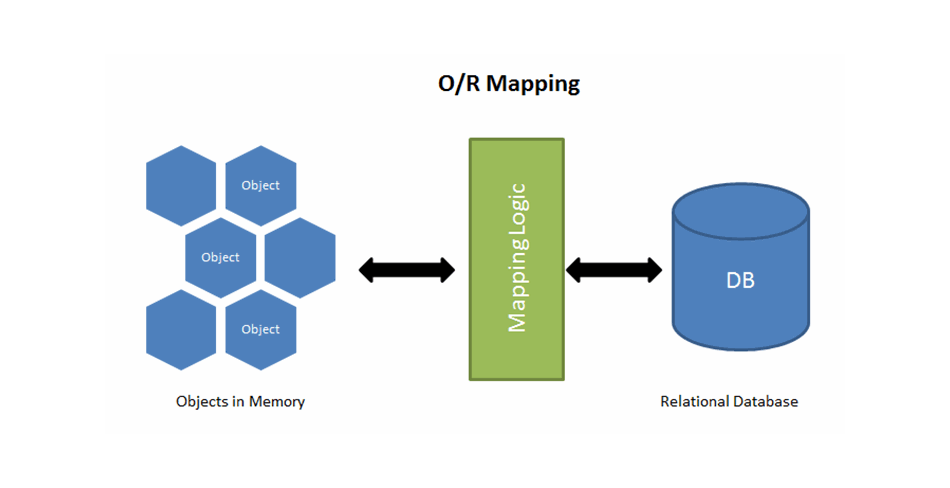
price

Description

=> Ánh xạ với đối tượng product trong cơ sở dữ liệu => để tương tác với cơ sở dữ liệu cũng có các cột id, name, price, description

**Ưu điểm**: Nhanh , dễ dàng , dễ tương tác với cơ sở dữ liệu

**Nhược điểm** : đối với các câu lệnh truy vấn phức tạp thì không nên dùng orm nên tự viết native query( các câu query thuần)



Active record : là phương pháp ánh xạ 1-1 => là các đối tượng có thể ảnh hưởng và lưu trữ trực tiếp dữ liệu vào database.

Product.name = ‘Linh’

Product.price = 1;

Product.save()

=> có k có id thì sẽ tạo mới , nếu có id thì sẽ là cập nhật

DataMapper => chỉ là đối tượng bình thường khi thay đổi dữ liệu thì không làm ảnh hưởng `

đến database sử dụng 1 đối tượng tên là repository

=> cung cấp các phương thức tương tác database: find(), save(), remove()....

* Cài đặt :
* Bước 1 : Trong folder src thêm file data-source :

import "reflect-metadata"

import { DataSource } from "typeorm"

export const AppDataSource = new DataSource({

type: "mysql",

host: "localhost",

port: 3306,

username: "root",

password: "123456",

database: "dbTest",

synchronize: true,

entities: ["dist/src/entity/\*.js"],

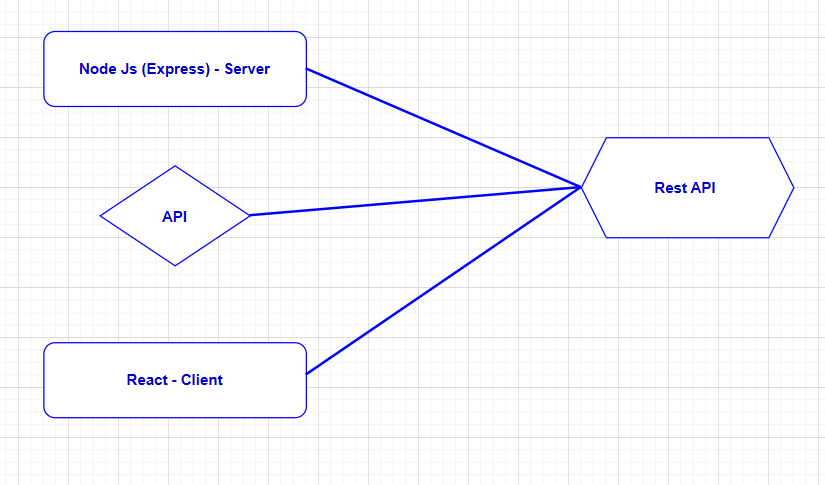
})

* Bước 2 : npm i typeorm reflect-metadata --save
* Bước 3 : Tạo database trong mysql để thực hiện tương tác.
* Bước 4 : Trong file index.ts thêm đoạn code thực hiện kết nối đến database :

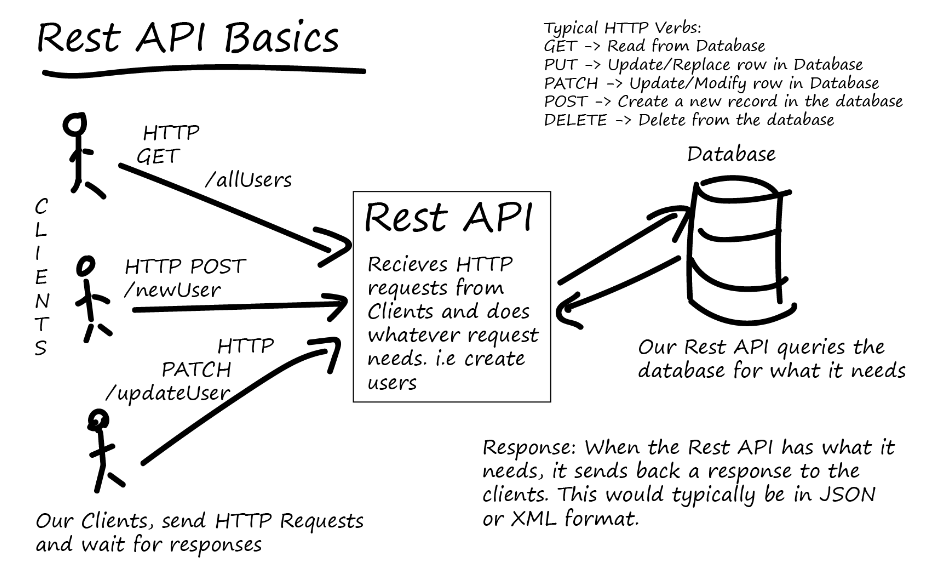
AppDataSource.initialize().then(() => { console.log('Connect Database Success') })

* Bước 5: Vào entity thêm các file.

**RestAPI**



API là gì ? (Application Programing Interface) là các phương thức giao thức hay đơn giản là cầu nối trung gian cung cấp các tài nguyên khác nhau cho ứng dụng



REST API sử dụng các thành phần của HTTP như URL, phương thức và các định dạng dữ liệu để truyền thông tin giữa các ứng dụng.

REST API (Representational State Transfer API) là một kiểu kiến trúc API cho phép các ứng dụng tương tác với nhau thông qua giao thức HTTP. REST API được thiết kế để sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để truy xuất và thay đổi dữ liệu trên máy chủ.

Website : web truyền thống là những web có đầy đủ giao diện và logic.

Web Service : là 1 dịch vụ web , tài nguyên có sẵn trên internet cung cấp các tài nguyên cho các ứng dụng khác và đặc biệt không có giao diện người dùng

Backend

Frontend

Restful

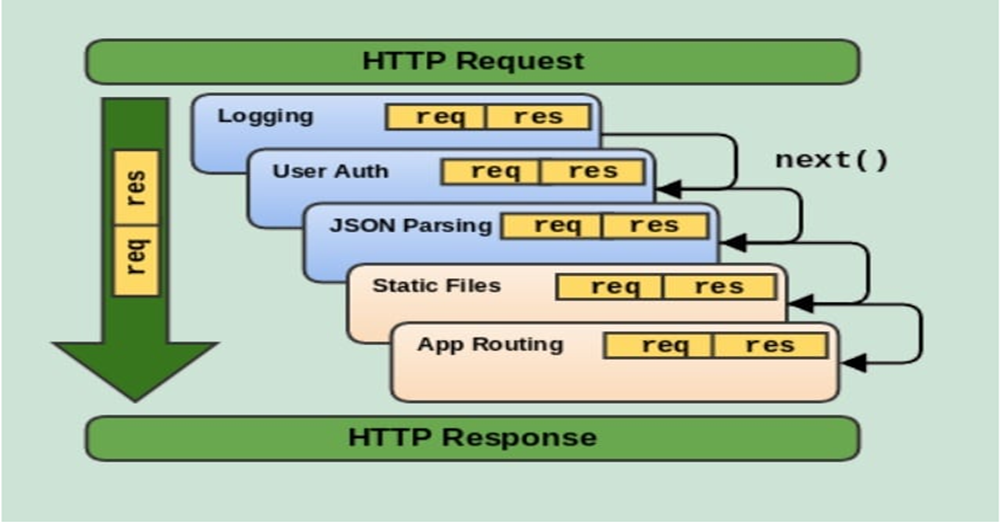
Json

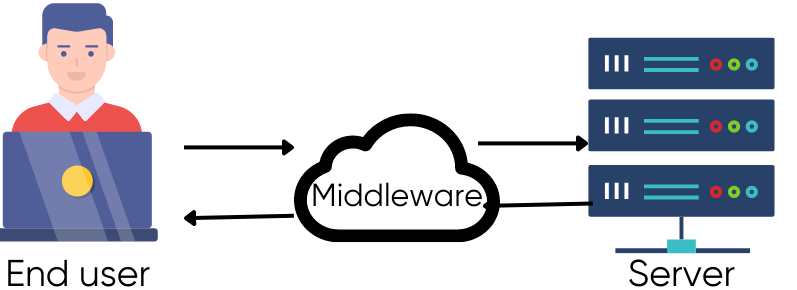
**Middleware**

* *Tìm hiểu về middleware*
* *Ý nghĩa*
* *Vai trò*
* *Ứng dụng*
* *Các loại middleware trong express*

**Khái niệm:**

* Middleware là các functions có quyền truy cập vào đối tượng request , response và 1 middleware function trong 1 chu kì req-res của ứng dụng.
* Ý nghĩa : là phần trung gian đứng giữa các thành phần trong mô hình phần mềm





**Chức năng :**

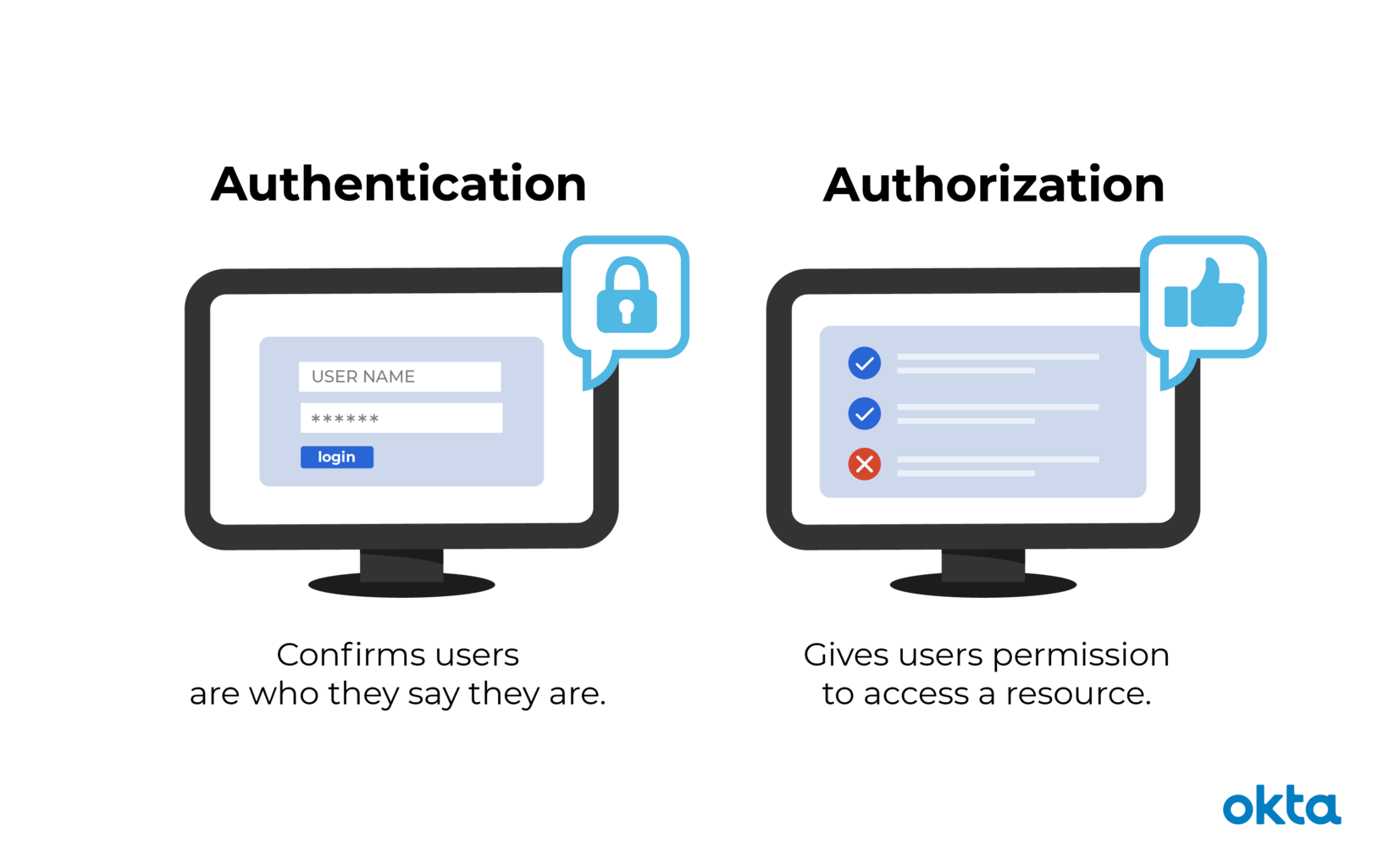
* Kiểm soát req => Cho phép vào hoặc không , cho phép thay đổi req và res
* Thực thi bất kì đoạn code nào.
* Thực hiện thay đổi với request và response.
* Kết thúc chu kì của request và response.
* Gọi middleware tiếp theo.

**Các loại middleware:**

* Application - level middleware : là middleware được sử dụng ở tầng ứng dụng hay được tích hợp trực tiếp vào đối tượng app
* Router - level middleware : là middleware được tích hợp vào đối tượng router của express
* Error - handling middleware : là các middleware functions dùng cho việc xử lý lỗi
* Built-in middleware : là những middleware được phát triển đóng gói module riêng và đc bảo trì bởi đội ngũ express
* Third-party middleware : là những middleware được lập trình và maintain bởi cộng đồng

**Authentication và Authorization**

* Authentication : Xác thực => Cơ chế bảo mật xem người dùng đã đăng nhập hệ thống hay chưa => Nếu có thì sử dụng các tính năng
* Authorization : Phân quyền => Sau khi người dùng xác thực (authentication) => Hệ thống sẽ kiểm tra người dùng được phép làm gì trong ứng dụng . VD : tài khoản user chỉ có quyền lấy danh sách học viên và lớp học



* JWT là cơ chế giúp ta bảo vệ dữ liệu (Sai)
* JWT là chìa khóa để truy cập tài nguyên và dữ liệu
* JWT giúp cho các request gửi từ phía client có thể xác thực yêu cầu
* Các thành phần JWT:
* Header: chứa kiểu dữ liệu và thuật toán để mã hóa
* Payload : chứa nội dung thông tin mà chúng ta muốn chứa trong token: username , id , role, …
* Chữ ký điện tử

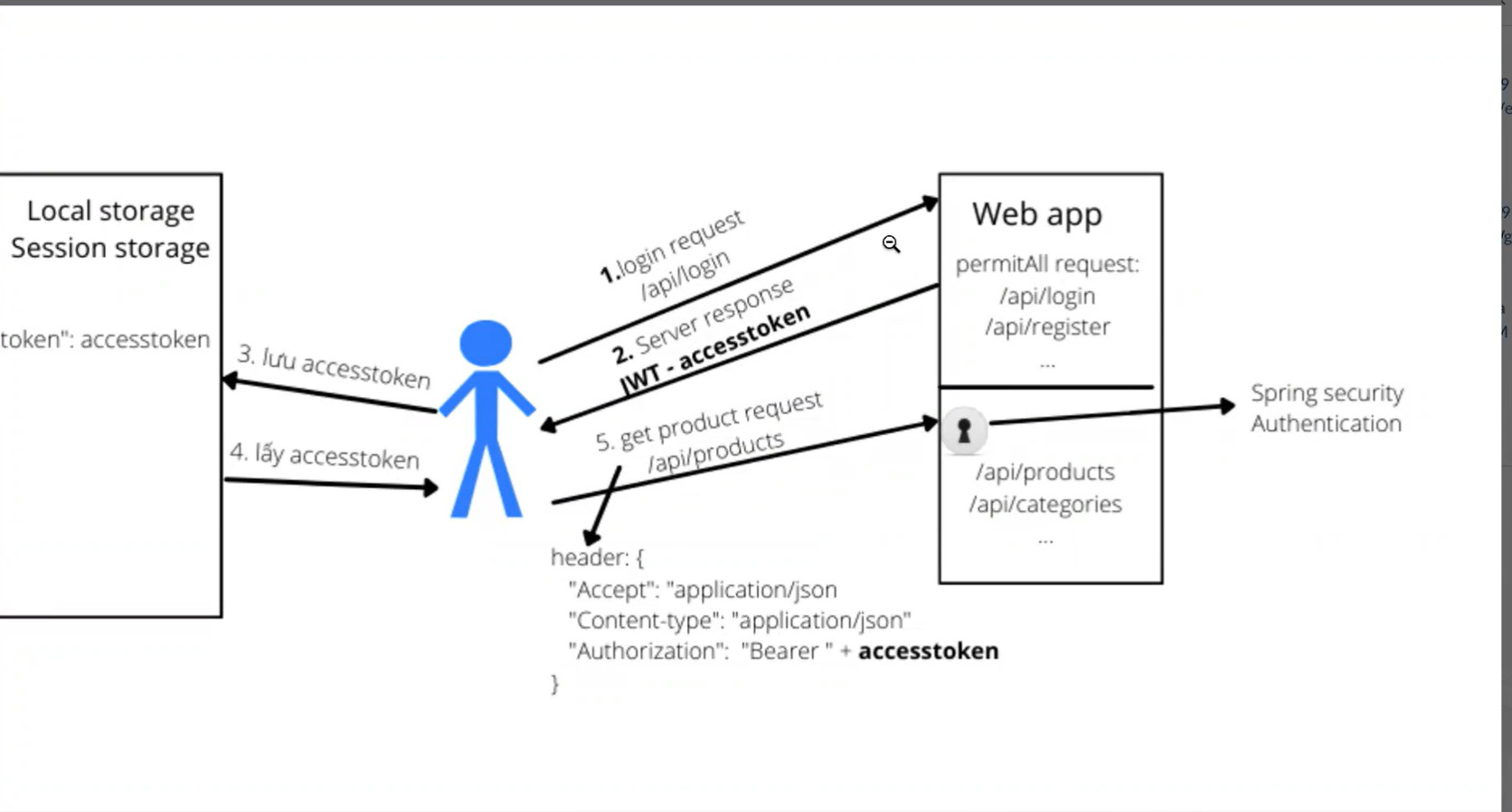
localStorage.setItem(‘ACCESS\_TOKEN’)

headers: {

'Content-Type': 'application/json',

'Authorization': 'Bearer ' + localStorage.getItem('ACCESS\_TOKEN')

}



Câu hỏi :

1. Bạn có thể giải thích sự khác biệt giữa Express và Node.js không?
2. Framework là gì ?
3. Phân biệt Framework và library ?
4. Express có hỗ trợ các phương thức HTTP nào? Và bạn có thể giải thích chúng được không?
5. Bạn có biết về RESTful API không? Express có thể được sử dụng để tạo ra một RESTful API không? Nếu có, hãy cho biết cách thực hiện điều này.
6. Bạn có thể giải thích khái niệm các phương thức HTTP được sử dụng trong REST API như GET, POST, PUT và DELETE được không?
7. API là gì ?
8. Bạn có thể giải thích khái niệm REST là gì không? REST API là gì
9. CSR là gì ? SSR là gì ? Nên sử dụng cái nào hơn ?
10. Bạn có thể giải thích khái niệm ORM là gì không?
11. ORM có những ưu điểm gì đáng chú ý?
12. Hãy cho biết một số ví dụ về cách sử dụng ORM trong dự án của bạn.
13. Type ORM là gì ?
14. Bạn có thể giải thích khái niệm NoSQL là gì không?
15. So sánh NoSQL và SQL ?
16. Ajax là gì ?
17. Bạn có thể giải thích cách AJAX hoạt động không?
18. JQuery là gì ? $.ajax({}) là gì ?
19. Ejs là gì ?
20. Parameter là gì ? Query là gì ? Khi nào dùng parameter, khi nào dùng query ?
21. Các tiêu chuẩn để thiết kế REST API ?
22. Web service là gì ?