

SERN

Web xoay quanh CRUD: Create, read, update, delete

|  |  |  |  |
| --- | --- | --- | --- |
| Back end | Sql | ORM(object relational mapping) – Sequelize | Bulid from Scratch |
| Express | Framework | APIs |
| Node.js | Environment |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Front end | React | Class:v | Base project |
| Redux | Mange state |  |
| Multiple languages |  |  |



Upload file đã thay đổi

Git add.

Git commit -r ‘......’

Tổng quan nodejs,express, và orm bất đồng bộ và promise

Promise là gì : **Promise** trong JavaScript là một đối tượng đại diện cho một quá trình xử lý bất đồng bộ, có thể hoàn thành hoặc thất bại trong tương lai. Nó giúp xử lý các tác vụ bất đồng bộ, như đọc dữ liệu từ file, gửi yêu cầu API, hoặc xử lý timeouts mà không cần sử dụng callback (tránh được vấn đề **callback hell**).

Trong một **Promise**, resolve và reject là hai hàm được sử dụng để báo hiệu kết quả của một tác vụ bất đồng bộ. Trong **return new Promise((resolve, reject) =>{ })**

### **Chi tiết về resolve và reject:**

1. **resolve(value)**:
   * Được gọi khi tác vụ bất đồng bộ hoàn thành thành công.
   * value là giá trị mà bạn muốn trả về từ Promise.
   * Khi resolve được gọi, Promise chuyển từ trạng thái **pending** sang trạng thái **fulfilled**.

**2.reject(reason)**:

* Được gọi khi có lỗi xảy ra trong quá trình thực hiện tác vụ.
* reason là lý do hoặc thông tin về lỗi.
* Khi reject được gọi, Promise chuyển từ trạng thái **pending** sang trạng thái **rejected**.

resolve được sử dụng để xác nhận rằng tác vụ đã hoàn thành thành công, và reject được sử dụng để thông báo rằng tác vụ đã thất bại.

Git push origin master

* Nodejs là gì(là 1 class form và môi trường cài đặt các thư viện của java scrip)

Sử dụng frame word là express cung cấp các công cụ để lập trình 1 website sử dụng nodejs

* ORM(Object-Relational Mapping) sự kết nối cảu các Object tham chiếu tới Database của chúng ta

Table user(id,fullname,address)

Bình thường : select \* from user where id =”...”

Còn dùng ORM table là object

User(id,fullname,address)

Db->User.find(where: id =7)

* Việc sử lý bất đồng bộ và promise giúp java scrip chạy nhanh hơn(tiến trình nhanh chạy trc chậm chạy sau)

Java

Php

Python

ruby

npm outdated (giúp xem hiện tại các thư viện có thư viện mới nào)

nếu muốn update thư viện sử dụng lệnh

npm install –save-exact @babel/core@7.19.3 @babel/node@7.19.1

body-parser: giup cta lay Api, lay cac tham so tu clinet goi len server cuar cta

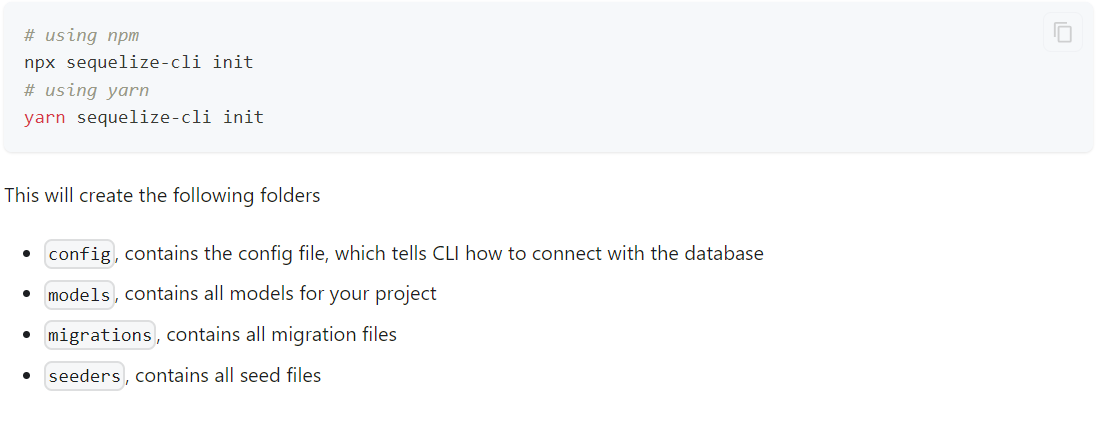
* Gitgnore : quy định cái nào đẩy lên server
* var(khai báo 1 cách normal) và let (khai báo phạm vi trong 1 file)
* cú pháp arrow function(tự tìm hiểu)

**Mô hình MVC**

* **Tìm vào router đầu tiên**

Cli command line interface(giao diện gtiep dòng lệnh)

*# using npm*  
npx sequelize-cli init



* npm install --save-dev sequelize
* $ node\_modules/.bin/sequelize init
* npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,email:string
* npm install --save mysql2

//THÊM BẢNG

* npx sequelize-cli db:migrate
* npx sequelize-cli seed:generate --name demo-user

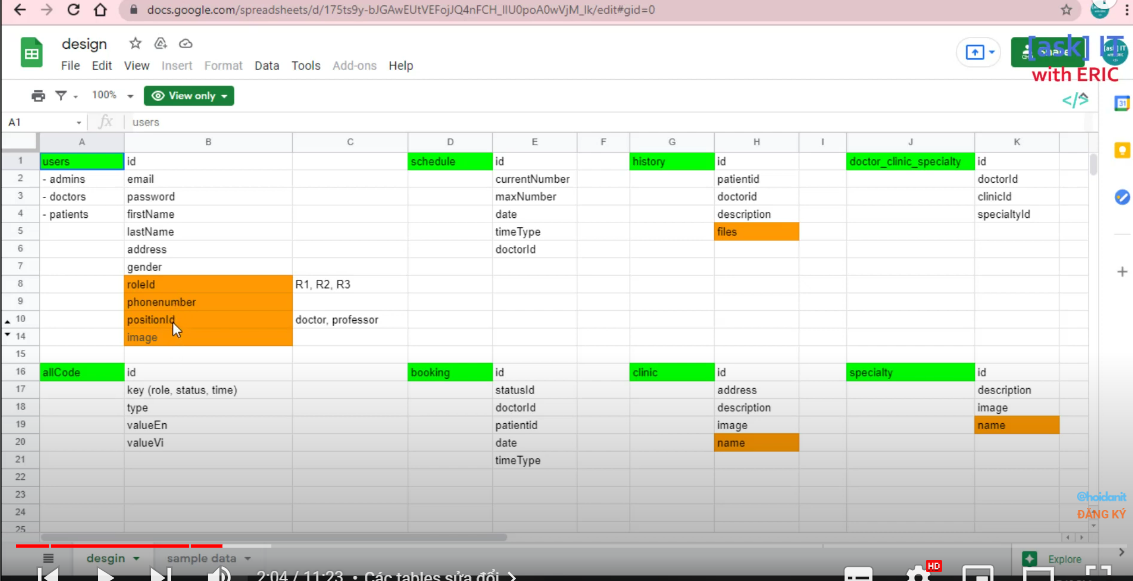
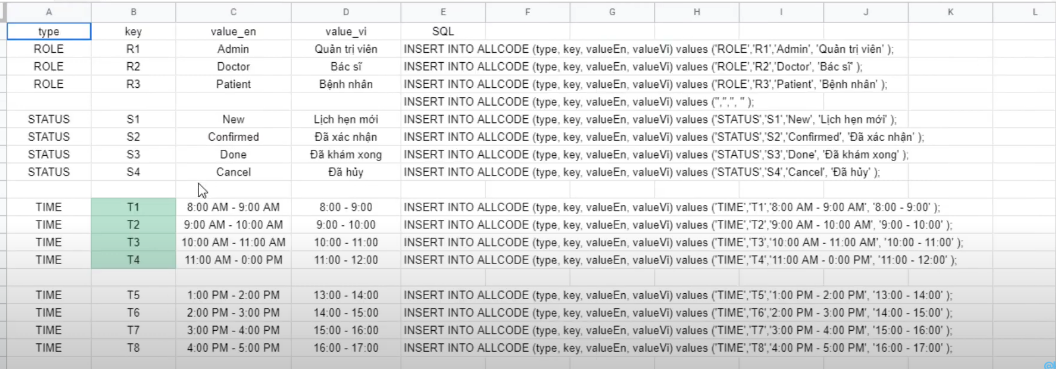
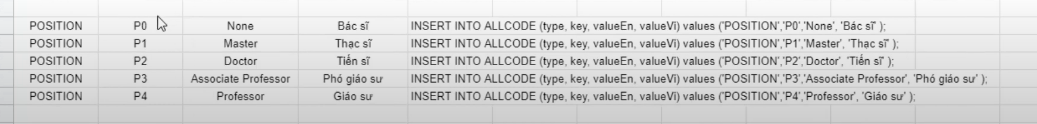
//THÊM DỮ LIỆU

* npx sequelize-cli db:seed:all

cấu linh cấu hình connecdb sequelize và cli

**https://sequelize.org/docs/v6/getting-started/**

[**https://sequelize.org/docs/v7/cli/**](https://sequelize.org/docs/v7/cli/)

**** **** 

Crud(create,read,update,delete)

**Req** và **Res** thường được sử dụng làm viết tắt trong lập trình web, đặc biệt trong các framework như Express.js, để chỉ **Request** (Yêu cầu) và **Response** (Phản hồi).

* **Req** là viết tắt của **Request**, đại diện cho yêu cầu mà máy khách (client) gửi đến máy chủ (server). Yêu cầu này chứa thông tin như phương thức HTTP (GET, POST, PUT, DELETE), URL được yêu cầu, headers, và dữ liệu yêu cầu (body).
* **Res** là viết tắt của **Response**, đại diện cho phản hồi mà máy chủ gửi lại cho máy khách sau khi xử lý yêu cầu. Phản hồi có thể bao gồm trạng thái HTTP (như 200, 404), nội dung trả về (HTML, JSON, v.v.), và headers.

Bookstrap hỗ trở viết sẳn các class

<https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp>

Mã hóa mật khẩu một cách an toàn

npm install --save bcrypt@

bcryptjs

<https://www.npmjs.com/package/bcryptjs>

npm bcryptjs

**hash password(so sanhs password trên db vs clinet) :**

bcrypt.compareSync("not\_bacon", hash); *// false*

react: user password : 123

dtb: user password: adhasodhoqwhjdkahdlsa

Crud

**sequelize find dùng** findOne trả ra 1 object

[**https://sequelize.org/docs/v6/core-concepts/model-querying-finders/**](https://sequelize.org/docs/v6/core-concepts/model-querying-finders/)

[**https://sequelize.org/docs/v6/core-concepts/model-querying-basics/**](https://sequelize.org/docs/v6/core-concepts/model-querying-basics/)

**API là cụm viết tắt của Giao diện chương trình ứng dụng**

**Backend:node.js(database)**

**Frontend: react.js(giao diện người dùng)**

**Apis?**

**Json?**

**Object js?>**

[**https://localhost:8080/get-crud**](https://localhost:8080/get-crud) **-> api(1 đường link trên con server gọi api nó sẽ xuất ra data)**

**database =>data**

**api render ra object ->json**

Cài Đặt Postman Để Test APIs

**Life cycle**

Cách Cấu Hình và Sử Dụng Nhiều Phiên Bản Của Node.JS Song Song Trên Cùng Một Máy Tính(cài nhiều version node.js)

[**https://www.youtube.com/watch?v=ccjKHLyo4lM**](https://www.youtube.com/watch?v=ccjKHLyo4lM)

[**https://github.com/haryphamdev/Frontend-React.JS-QuickStart**](https://github.com/haryphamdev/Frontend-React.JS-QuickStart)

**Form Login mãu dựa vào để code:**

https://codepen.io/rares-lungescu/pen/KLbMvo

Horizontal line

**State(kiểm soát giá trị trong các biến trong các component đang sử dụng) and lifcycle ?**

Trong React, **state** là một đối tượng mà mỗi component có thể sử dụng để lưu trữ và quản lý dữ liệu động của mình. Nó khác với **props** ở chỗ **state** có thể thay đổi được, trong khi **props** là dữ liệu mà component nhận từ cha và không thể tự ý thay đổi.

### Ý nghĩa của state trong đoạn code:

* **this.state** trong React component là nơi bạn lưu trữ các thông tin mà component cần để duy trì hoặc hiển thị trong quá trình render.
* Khi bạn thay đổi state, component sẽ tự động render lại giao diện để phản ánh những thay đổi đó.

### Các khái niệm chính về API(Application Programming Interface): là một tập hợp các quy tắc và định dạng mà các ứng dụng phần mềm có thể sử dụng để giao tiếp và trao đổi dữ liệu với nhau. API đóng vai trò là cầu nối giữa các hệ thống hoặc ứng dụng khác nhau, cho phép chúng chia sẻ thông tin hoặc thực hiện các hành động mà không cần biết chi tiết nội bộ của nhau.

1. **Request và Response**: Một ứng dụng (client) gửi yêu cầu (request) tới một API, và API sẽ phản hồi (response) với kết quả yêu cầu.
   * **Request** thường chứa thông tin về dữ liệu cần truy vấn hoặc hành động cần thực hiện.
   * **Response** chứa dữ liệu hoặc kết quả của hành động đó, dưới dạng JSON, XML hoặc các định dạng khác.
2. **Endpoints**: Mỗi API có các endpoint, là các URL cụ thể để truy cập các chức năng hoặc tài nguyên của API.
3. **Phương thức HTTP**: API thường sử dụng các phương thức HTTP để xác định loại yêu cầu:
   * **GET**: Lấy dữ liệu.
   * **POST**: Tạo dữ liệu mới.
   * **PUT/PATCH**: Cập nhật dữ liệu hiện có.
   * **DELETE**: Xóa dữ liệu.

### 1. req **(request)**

* **req** đại diện cho **yêu cầu** từ phía client gửi tới server. Nó chứa thông tin về yêu cầu HTTP, chẳng hạn như:
  + **URL** được yêu cầu.
  + **Phương thức HTTP** (GET, POST, PUT, DELETE, v.v.).
  + **Headers** và **body** của yêu cầu.
  + **Params**, **query**, và **cookies** của yêu cầu.

### 2. res **(response)**

* **res** đại diện cho **phản hồi** mà server sẽ gửi về phía client. Đối tượng này có nhiều phương thức để gửi dữ liệu về cho client, chẳng hạn như:
  + **res.send()**: Gửi nội dung phản hồi (chuỗi, đối tượng, JSON, v.v.).
  + **res.json()**: Gửi phản hồi dưới dạng JSON.
  + **res.status()**: Đặt mã trạng thái HTTP (200, 404, 500, v.v.).
  + **res.render()**: Render một trang HTML.

Rest API(Representational State Transfer Application Programming Interface): là một kiểu kiến trúc cho phép giao tiếp giữa các hệ thống thông qua giao thức HTTP. Đây là một trong những kiểu API phổ biến nhất, được sử dụng rộng rãi trong việc phát triển các dịch vụ web.

Status(200), status(500)

JWT(Json web token) là một tiêu chuẩn mở (RFC 7519) để truyền tải dữ liệu an toàn giữa hai bên dưới dạng đối tượng JSON. Dữ liệu trong JWT được mã hóa và có thể được xác minh tính toàn vẹn vì được ký số bằng khóa bí mật (hoặc khóa công khai nếu sử dụng phương pháp RSA hoặc ECDSA

**Cors :**

https://stackoverflow.com/questions/46337471/how-to-allow-cors-in-react-js