นายสิทธิพงศ์ จำรัสฤทธิรงค์ 6410301019 log book

2024-12-20 YYYY-MM-DD Backend ทำเสร็จและทดสอบจนครบ

A screenshot of a computer

Description automatically generated

Backend ประกาศ API endpoint

A screenshot of a computer

Description automatically generated

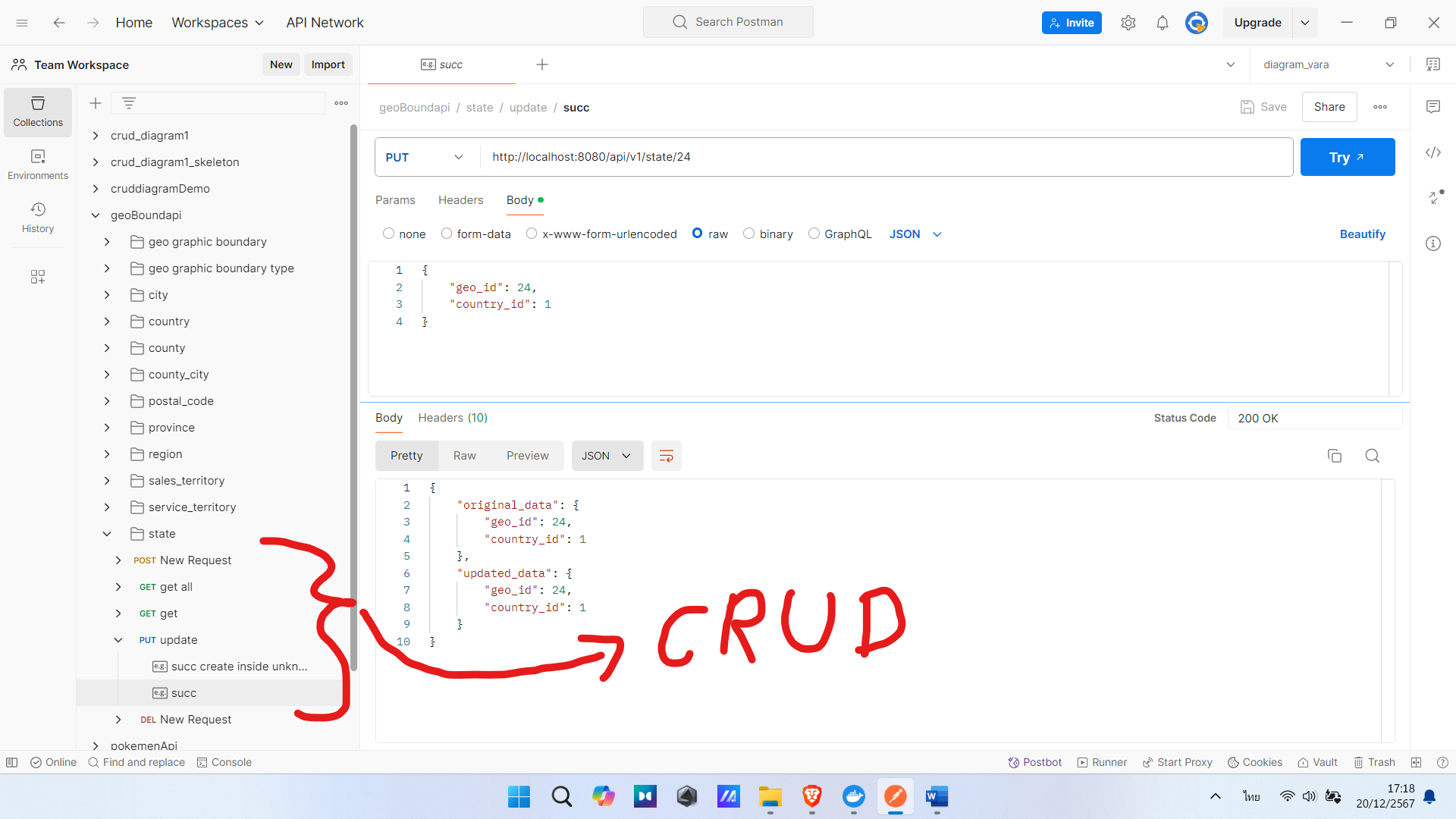
A screen shot of a computer

Description automatically generated

API endpoint คือการ map URL เข้ากับ service ที่ถูก implement ในรูปแบบ class ใน Laravel

URL ล่างสุดคือ localhost:8080/api/v1/test จะ response เป็น hello world เพื่อทดสอบ unit test แค่ส่วน API เท่านั้น

URL บนสุดคือ auth:sanctum มีมากับ Laravel มีไว้จัดการเรื่อง authentication และ privilege ในการ access ระบบ

โดย service ที่ทดสอบมีดังนี้

1. geo graphic boundary
2. geo graphic boundary type
3. city
4. country
5. county
6. county\_city
7. postal\_code
8. province
9. region
10. sales\_territory
11. service\_territory
12. state

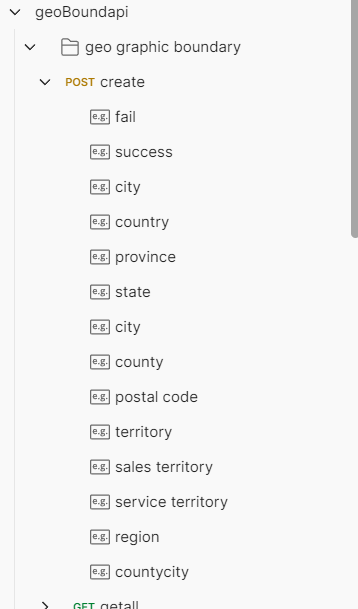
แต่ละ service จะมี 5 action

1. Create คือเพิ่มข้อมูล
2. Read all คืออ่านทั้งหมด
3. Read one คืออ่านตัวเดียวตาม id ที่กำหนด
4. Update แก้ data ตาม id ที่กำหนด
5. Delete ลบ data ตาม id ที่กำหนด

A screenshot of a computer

Description automatically generated

ที่ title ของ service collection แต่ตัวจะมีชื่อและคำอธิบาย



ข้างใน create จะมีการบันทึกประวัติการ query ที่สำเร็จเอาไว้เป็นตัวอย่างในการเรียกใช้ API

A screenshot of a computer

Description automatically generated

Folder ใน project มี folder เก็บคำอธิบายเกี่ยวกับ backend แบบครบถ้วนทั้งการสร้าง การใช้งาน

ผลงาน Frontend

A screenshot of a computer

Description automatically generated

จากรูปด้านบนคือหน้า menu หลัก

A screenshot of a computer

Description automatically generated

จากรูปด้านบนคือหน้า CRUD app เกี่ยวกับ country(ประเทศ)

A screenshot of a computer

Description automatically generated

จากรูปด้านบนคือการ create country (เพิ่มประเทศใหม่)

A blue and white screen

Description automatically generated

รูปด้านบนคือระบบกำลังเพิ่มประเทศใหม่ มีหมุนๆ loading เพื่อให้ user รู้ว่า app กำลังทำงานอยู่

A screenshot of a computer

Description automatically generated

รูปด้านบนคือประเทศใหม่ที่ถูกเพิ่ม

A screenshot of a computer

Description automatically generated

รูปด้านบนคือข้อมูลก่อนจะ update ที่ถูกดึงจากตารางมาให้แก้ใข เพื่อความสะดวก ไม่ต้องกรอกใหม่เองหมด

A screenshot of a computer

Description automatically generated

จากรูปด้านบนคือประเทศที่ถูกแก้ใขข้อมูล



จากรูปคือ การหมุน loading เพื่อให้ user รู้ว่าระบบไม่ได้ค้าง มันแค่ช้า

A screenshot of a computer

Description automatically generated

รูปด้านบนคือข้อมูลที่ถูกเพิ่มลงในระบบแล้ว

A screenshot of a computer

Description automatically generated

จากรูปด้านบนคือคือประเทศประเทศได้ถูกลบไปแล้ว

A screenshot of a computer

Description automatically generated

สามารถเลือกได้ว่าจะ show ทีละกี่ data ต่อหน้า และบอกด้วยว่าหน้าล่าสุดแสดงข้อมูลที่เท่าไรถึงเท่าไร

หลักการทำงาน Frontend

A screenshot of a computer

Description automatically generated



A diagram of a diagram

Description automatically generated

รูปด้านบนคือ overview หลักการทำงานของ react คร่าวๆ

Library ตัวที่สำคัญหลักๆ มีดังนี้

1. React

React เป็นไลบรารี JavaScript ที่ใช้สำหรับสร้างส่วนติดต่อผู้ใช้ (User Interface) แบบ Component

1. React Router DOM

React Router DOM คือไลบรารีที่ช่วยจัดการ การนำทาง (Routing) โดยสร้าง URL ที่กำหนดเองและเชื่อมโยงกับ Component ต่าง ๆ เพื่อแสดงหน้าที่แตกต่างกัน

1. MUI

MUI (Material-UI) เป็นไลบรารี UI Component ที่สร้างจาก Material Design ของ Google ช่วยให้สร้าง UI ที่ดูเป็นมาตรฐานและมืออาชีพได้อย่างรวดเร็ว

1. MUI icon

MUI Icon เป็นไลบรารีที่รวบรวม ไอคอนแบบเวกเตอร์ (SVG) ที่ใช้ใน UI มีไอคอนมากกว่า 1,000 ตัว เช่น ไอคอนเมนู, การนำทาง, หรือสถานะต่าง ๆ

1. SWC & vite ทำ hotmodule replacement

SWC และ Vite เป็นเครื่องมือที่ช่วยในการพัฒนา React อย่างรวดเร็ว ตัวแปลงโค้ด (Transpiler) SWC ทำหน้าที่เหมือน Babel แต่ทำงานได้เร็วกว่า ทำให้เร็วกว่า JavaScript Transpilers ทั่วไป โดยความหมายของ transpiler คือ แปลงจาก Tsx Jsx เป็น html + javascript ที่ browser เข้าใจได้

1. Axios

Axios คือไลบรารี JavaScript ที่ใช้สำหรับสร้างการร้องขอ HTTP (Hypertext Transfer Protocol) เพื่อแลกเปลี่ยนข้อมูลกับเซิร์ฟเวอร์หรือ API (Application Programming Interface) มีคุณสมบัติหลักๆ ดังนี้:

* สร้างการร้องขอ HTTP (GET, POST, PUT, DELETE, PATCH)
* รองรับการใช้งานกับ Promise และ Async/Await
* สามารถกำหนดค่า Header, Parameter และ Body ของการร้องขอได้
* มีการจัดการข้อผิดพลาดและสถานะการร้องขอ
* รองรับการใช้งานกับ JSON และข้อมูลอื่นๆ

สาเหตุที่ผมใช้ axios แทนการเขียน native request เพราะความง่าย

นี่คือตัวอย่างการใช้งาน axios เทียบกับ native

|  |  |
| --- | --- |
| axios | native |
| import axios from 'axios';  const updatePerson = async () => {  const person = {  id: 1,  name: 'John Doe',  age: 30  };  try {  const response = await axios.put('https://api.example.com/persons/1', person);  console.log(response.data);  } catch (error) {  console.error(error.message);  }  };  updatePerson(); | const updatePerson = async () => {  const person = {  id: 1,  name: 'John Doe',  age: 30  };  try {  const response = await fetch('https://api.example.com/persons/1', {  method: 'PUT',  headers: {  'Content-Type': 'application/json'  },  body: JSON.stringify(person)  });  const data = await response.json();  console.log(data);  } catch (error) {  console.error(error.message);  }  };  updatePerson(); |

**ข้อดีของ Axios**

1. ใช้งานง่ายและเข้าใจได้ง่าย
2. ไม่ต้องกำหนด Header และ Method อย่างชัดเจน
3. การจัดการข้อผิดพลาดที่ดีกว่า

หลักการทำงานของ code

ส่วนต่อไปของเอกสารนี้ จะเป็นการอธิบาย code Frontend คร่าวๆ

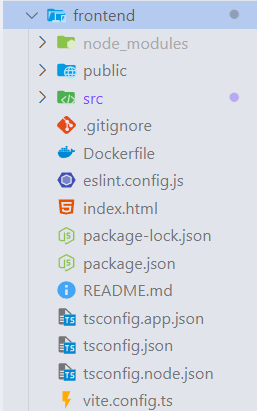
A screenshot of a computer

Description automatically generated

จากภาพประกอบข้างต้น Docker Compose เป็นเทคโนโลยีที่ช่วยให้การรวม Container หลายตัวเข้าด้วยกันเป็นไปอย่างสะดวก โดย Docker เปรียบเทียบแนวคิดนี้กับถังที่สามารถบรรจุ Container ได้หลายตัว ในการใช้งาน Docker Compose นักพัฒนาสามารถเขียนไฟล์ข้อความประเภท YAML เพื่อกำหนดการตั้งค่าต่างๆ โดยไฟล์ YAML มีรูปแบบที่คล้ายคลึงกับ JSON แต่เน้นการใช้งาน Indentation (การเว้นวรรคด้วย Tab) เพื่อสร้าง Key:Value และ Dash (-) เพื่อสร้าง Array ทำให้ไฟล์ YAML มีลักษณะที่อ่านและเข้าใจได้ง่ายสำหรับมนุษย์

เริ่มต้นด้วยหัวข้อแรกคือ การใช้เทคโนโลยี Containerization ซึ่งเป็นการนำแอปพลิเคชันในส่วน Frontend มารันในสภาพแวดล้อมที่คล้ายกับ Virtual Machine (VM) แต่มีความแตกต่างจาก VM เนื่องจาก Container ไม่มี Kernel เป็นของตนเอง โดย Container จะใช้ Kernel เดียวกันกับระบบปฏิบัติการ Windows และมีการทำ Virtualization ในระดับที่เพียงพอสำหรับการทำงานของแอปพลิเคชัน นอกจากนี้ Container ยังสามารถทำงานได้บนทุกเครื่องคอมพิวเตอร์ที่รองรับและติดตั้ง Docker ได้

สำหรับ Container ของส่วน Frontend ที่ผู้เขียนพัฒนานั้นถูกสร้างขึ้นโดยใช้ Dockerfile ที่อยู่ในโฟลเดอร์ ./frontend ในการตั้งค่าเพิ่มเติม มีการใช้งาน Volume ซึ่งเป็นการนำโฟลเดอร์จากโลกเสมือน (Docker) มาแสดงในโลกจริง (Windows) เพื่ออำนวยความสะดวกในการพัฒนาแอปพลิเคชัน โดยไฟล์โปรแกรมจะยังคงอยู่แม้ว่า Instance ของ Container จะถูกลบออกไป นอกจากนี้ยังมีการตั้งค่า Depending on Backend ซึ่งหมายความว่า Container ของ Frontend จะเริ่มทำงานหลังจาก Backend ทำงานสำเร็จ เนื่องจาก Frontend จำเป็นต้องเชื่อมต่อกับ Backend เพื่อทำงานร่วมกัน หาก Backend มีปัญหาหรือหยุดทำงาน Frontend จะไม่สามารถใช้งานได้และแสดงข้อความ Error นอกจากนี้ Container ยังถูกตั้งค่าให้เชื่อมต่อกับ Network “app-network” ซึ่งในทางปฏิบัติเปรียบเสมือนการเชื่อมต่อคอมพิวเตอร์เข้ากับ Switch พร้อมทั้งกำหนด Domain Name “frontend” เพื่อให้อุปกรณ์หรือแอปพลิเคชันอื่นสามารถส่งข้อมูลเข้ามาได้โดยใช้ชื่อ Domain Name ดังกล่าวโดยไม่จำเป็นต้องทราบ IP Address



นี่คือ structure frontend โดย folder ที่ใช้ทำงานเป็นหลักคือ SRC ย่อมาจากคำว่า source

my-react-app/

│

├── public/ # ไฟล์สาธารณะ เช่น รูปภาพ ฟอนต์

│ ├── favicon.ico

│ └── images/

│

├── src/ # โฟลเดอร์หลักสำหรับโค้ดของแอปพลิเคชัน

│ ├── assets/ # ไฟล์สื่อ เช่น รูปภาพ, ไอคอน

│ ├── components/ # คอมโพเนนต์ที่ใช้ซ้ำ

│ ├── pages/ # หน้าแต่ละหน้าในแอปพลิเคชัน

│ ├── hooks/ # Custom hooks

│ ├── context/ # Context API สำหรับ state management

│ ├── services/ # ฟังก์ชันช่วยเหลือหรือ API calls

│ ├── styles/ # สไตล์ CSS หรือ SCSS

│ ├── utils/ # ฟังก์ชันช่วยเหลือทั่วไป

│ └── App.jsx # คอมโพเนนต์หลักของแอปพลิเคชัน

│

├── index.html # ไฟล์ HTML หลัก

├── vite.config.js # การตั้งค่าของ Vite

└── package.json # ข้อมูลโปรเจกต์และ dependencies

Library ที่ผมใช้ มีดังนี้

├── @emotion/react@11.14.0

├── @emotion/styled@11.14.0

├── @eslint/js@9.16.0

├── @mui/icons-material@6.2.0

├── @mui/material@6.2.0

├── @mui/x-data-grid@7.23.2

├── @types/react-dom@18.3.5

├── @types/react@18.3.16

├── @vitejs/plugin-react-swc@3.7.2

├── axios@1.7.9

├── eslint-plugin-react-hooks@5.1.0

├── eslint-plugin-react-refresh@0.4.16

├── eslint@9.16.0

├── globals@15.13.0

├── react-dom@18.3.1

├── react-router-dom@7.0.2

├── react@18.3.1

├── typescript-eslint@8.17.0

├── typescript@5.6.3

└── vite@6.0.3

ผมจะอธิบายโดยอิงจากการสิ่งที่ user เห็นไล่ไปหาสิ่งที่อยู่เบื่องหลัง

A screenshot of a computer

Description automatically generated

สิ่งที่ user ใช้งานทั้ง app จะถูกทำงานอยู่ที่หน้า index.html ทั้งหมด โดยสิ่งที่เป็นหัวใจสำคัญให้ app นี้ทำงานได้คือ

<div id="root"></div>

Tag ตัวนี้คือตัวรากสุดของ app เป็น div (กล่องปล่าว) ที่ถูกกำหนด id ว่า root เพื่อให้ JavaScript สามารถระบุ element ที่จะต้องกระทำอะไรบางอย่างใส่ได้

<script type="module" src="/src/main.tsx"></script>

Tag ตัวนี้คือตัวที่ระบุว่าจะเอา script มาจาก /src/main.tsx โดยอิงจุดเริ่มต้นคือ file index.html

A screen shot of a computer

Description automatically generated

จากรูปดังกล่าวคือ code main.tsx คือ code หลักที่จะถูก execute เป็นที่แรก เปรียบเสมือนกับ int main() ในภาษา C

A screenshot of a computer program

Description automatically generated

จาก code ส่วนนี้คือ การ load component หน้า page แบบ lazy loading โดยที่ lazy loading คือ load ค่อย load หน้าเว็บจาก frontend server ตอนที่จะต้องใช้

A screen shot of a computer

Description automatically generated

จากรูปดังกล่าว หมายเลข 1 คือ component หน้า homepage ที่ถูก load มาก่อน หลังจากนั้น user ค่อย click ปุ่ม country สิ่งที่เกิดขึ้นคือ browser จะ request ไปเรียก component country จาก frontend server มาเพิ่ม สาเหตุที่ต้องทำแบบนี้เพราะ

1. ประหยัด ram ไม่ต้อง load component ที่ยังไม่ได้แสดงออกมา
2. ประหยัด internet component ที่เป็น service หลายๆ ตัวจะถูก set ให้ request ข้อมูลจาก backend แบบ auto ตั้งแต่เริ่มทำงาน สิ่งที่เกิดขึ้นคือตอนเปิด website ขึ้นมา เครื่องคอมจะกระตุกมากๆ และ load website นานมากๆ

A screenshot of a computer program

Description automatically generated

ส่วนต่อมาคือ create root คือ function ที่จะชี้เป้าหมายไปที่ id root จาก index.html รูปที่ผ่านๆ มา จุดสำคัญคือ suspense fallback ที่ครอบ tag router โดย tag ดังกล่าวมีความหมายว่าระหว่างที่ load component นั้นๆ มาจาก frontend server ให้แสดง หมุนๆ loading ไปพลางๆ ก่อน เพื่อให้ user รู้ว่า app กำลังทำงานอยู่ แต่แค่ช้าเฉยๆ

A screenshot of a computer

Description automatically generated

จากรูปดังกล่าวคือ component ที่เป็นตัวแทนของ หน้าบริการเกี่ยวกับ country

A screenshot of a computer

Description automatically generated

จากรูปดังกล่าวคือ สิ่งที่ถูก import เข้ามาใน country.tsx โดยจะมีสิ่งต่างๆ ดังนี้

lib state, use effect ของ react โดย usestate มีไว้สำหรับสร้างตัวแปร ที่จะเอาผลลัพธ์ไปแสดงแบบ Realtime use-effect เป็นการสร้าง function ที่จะ trigger ตัวเองโดยอิงกับอะไรซักอย่าง ในที่นี้ผมเอามาอิงกับการที่ component country load สำเร็จ ให้ทำการดังข้อมูลประเทศทั้งหมดมาแสดงในทันที

app bar custom คือ navigation bar ส่วนที่อยู่ด้านบนสุดของ website ส่วนมากจะมีไว้ใช้สำหรับการนำทางไปหน้าต่างๆ ของ web

grid col def มีไว้สำหรับ check type ตัวแปรตัวหนึ่งที่จะไว้ใช้ในการ config ตารางข้อมูล mui สาเหตุที่ต้องมีการ check type เพราะว่า project ที่ทำเป็น MUI ทุกตัวแปรต้อง check type เสมอ

Datatable คือตารางข้อมูล โดย base มาจาก mui และผมเอามา custom เองให้เหมาะสมกับ project ของผม

Button คือปุ่มของ mui และ loading ก็คือ loading หมุนๆ ของ mui (ในอนาคตอาจจะทำ loading ของตัวเอง)

Create country update country และอื่นๆ คือ service ที่ช่วยติดต่อกับ backend ผมเขียนเป็น file แยกเพื่อให้ code lean มากขึ้น

A screenshot of a computer program

Description automatically generated

จาก code ส่วนนี้ คือ code ที่ไว้ระบุรายละเอียดต่างๆ ใน column โดย 2 ตัวสุดท้ายจะแปลกๆ หน่อยเพราะมันคือการ render ปุ่ม update delete ที่เมื่อเวลา action จะส่งค่าไปยัง function พิเศษที่ไว้จัดการ

A screenshot of a computer code

Description automatically generated

ส่วนแรกที่ app จะทำคือทำ function นี้ก่อน

A screenshot of a computer code

Description automatically generated

Function นี้จะ check ก่อนว่ามี argument id ส่งเข้ามามั้ย ถ้ามีแสดงว่าเป็นการ read one ถ้าไม่มี แสดงว่าเป็นการ read all ใต้ if แต่ละตัวมีการทำ try catch ไว้จับ error เพื่อให้ react ไม่ดับในกรณีที่ request fail

A screenshot of a computer code

Description automatically generatedA screenshot of a computer code

Description automatically generated

ฟังก์ชันทั้งสองนี้ทำงานร่วมกัน โดยฟังก์ชันด้านล่างมีหน้าที่ Map ค่าจาก Array ทีละตัว และส่งค่าไปยังฟังก์ชันด้านบน เพื่อทำการแปลงวัตถุ (Object) ซึ่งในภาษา Python เรียกว่า Dictionary (dict) การทำงานของฟังก์ชันด้านบนคือการแยก Key ภายใน Boundary ออกมาอย่างละเอียด

สาเหตุที่ต้องดำเนินการในลักษณะนี้ เนื่องจากข้อมูลที่ถูกสร้างขึ้นจาก Laravel ไม่สามารถใช้งานร่วมกับ Data Table ของ Material-UI (MUI) ได้โดยตรง จึงมีความจำเป็นต้องทำการแปลงรูปแบบข้อมูลให้เหมาะสมก่อนที่จะนำไปใช้งาน

A screenshot of a computer program

Description automatically generated

Code ส่วนนี้คือส่วนที่ country page จะถูก render ให้ client เห็น โดยอธิบายย่อๆ ได้ดังนี้ render app bar ไว้ด้านบน ต่อมาก็ check ว่า loading อยู่มั้ยด้วยการใช้ if แบบย่อๆ ใน js ถ้ามัน loading เสร็จแล้วก็ให้ render data table และรับ props หลายตัว ไม่ขอลงรายละเอียด

ปุ่มคือปุ่มเพิ่ม country ใหม่ เวลากดจะไปเรียก component ด้านล่างให้แสดง ซึ่งในศัพท์ frontend ก็คือ modal ภาษาบ้านๆ คือ popup ให้กรอกข้อมูล ถ้าในกรณีที่ create new country สร้าง popup หน้ากรอกข้อมูลปล่าวๆ ออกมา แต่ถ้าแสดง popup จากการกดหปุ่ม update จะมีการเอา value ใน row นั้นๆ ไป map ลงใน form กรอกข้อมูล เพื่อที่ user เวลาแก้ใขข้อมูลเพียงบางตัวจะได้แก้ได้สดวกๆ จากข้อมูลเก่าที่มีอยู่ เวลาที่ save หรือกดนอก model เพื่อปิด form จะต้องมีการ clear value ใน form ทุกครั้งรายละเอียดอยู่ใน function closemodal()

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

นี่คือ code ของ component nav bar โดยสังเขป

โดยส่วนที่ต้องแก้ใข เพื่อสร้าง route มีดังนี้

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

ถ้ากดปุ่มที่ชื่อว่า go to home จะ redirect กลับ home

A screenshot of a computer

Description automatically generatedA screenshot of a computer code

Description automatically generated A screenshot of a computer

Description automatically generated

Title text คือ เอา text จาก props มาแสดง

A screen shot of a computer

Description automatically generated

นี่คือ code ในส่วน modal

A screen shot of a computer program

Description automatically generated

ตัวที่คอยจัดการเกี่ยวกับข้อมูลภายใน form จะอยู่ใน state ตัวเดียว ชื่อว่า formData