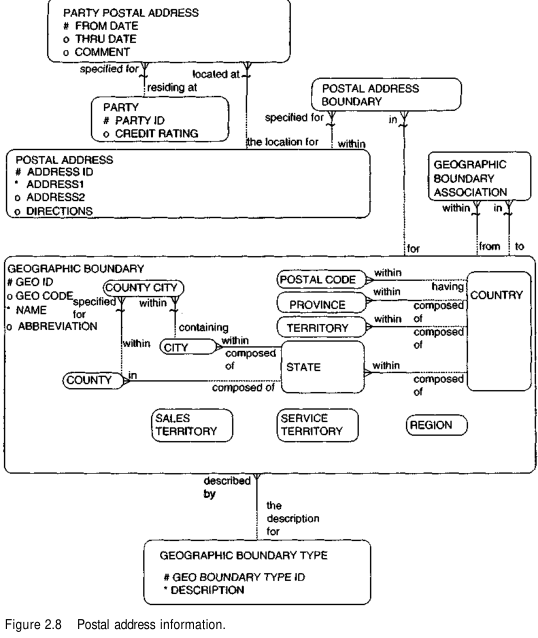
2024-12-08 yyyy-mm-dd

วันที่ 2024-12-06 วันศุกร์ที่ผ่านมา อาจารย์มอบหมายงานทดสอบฝีมือดังนี้

ทำ datamodel เกี่ยวกับ geographic boundaries ตามแบบในหนังสือ Data model resource book vol 1



ให้ทำ แค่ crud ที่อยู่ โดยจะต้องทำ

Database ด้วย Postgresql

Backend ด้วย laravel PHP

Frontend react

Architecture ที่ใช้

A diagram of a software development process

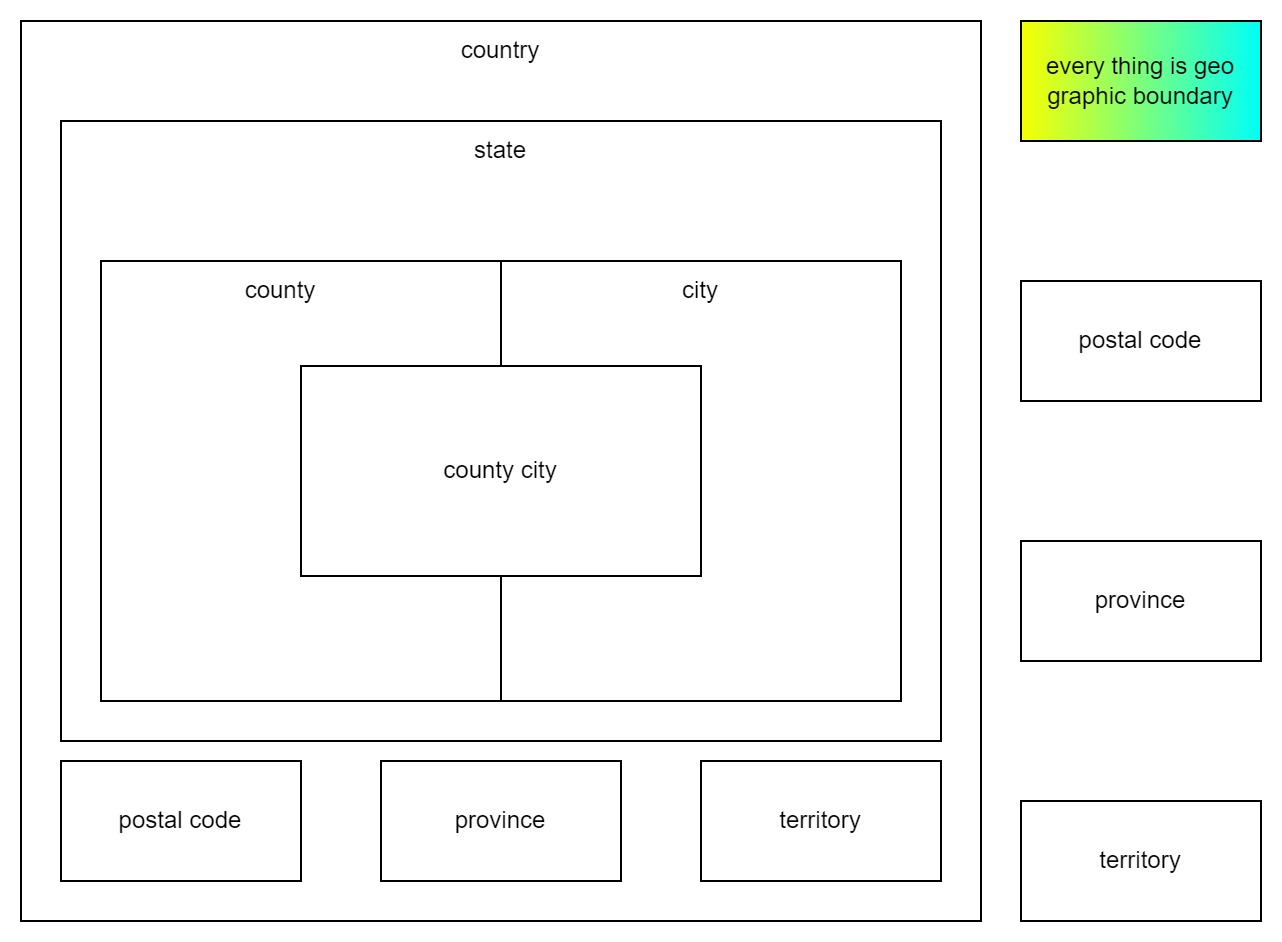
Description automatically generated

หลักการทำงานคร่าวๆ ของ Laravel

A diagram of a computer system

Description automatically generated

มีการทำ dropdown list ให้ user เลือกข้อมูลที่จะกรอกใน form ได้แค่เท่าที่มี ป้องกัน user error



จากรูปคือ diagram ที่แสดงความใหญ่ของแต่ table เทียบกัน

A white board with black text

Description automatically generated

จากรูปคือการออกแบบ ux ui คร่าวๆ สำหรับหน้ารายละเอียดเกี่ยวกับ geo boundary แต่ละตัว

ตัวอย่างที่ยกมาคือ detail ของ state สาเหตุที่ยกตัวนี้มาเพราะว่ามันอธิบาย concept ต่างๆ ได้ครบดี

1. Inside country คือระบุว่าอยู่ใน country ใหน form ในส่วนนี้จะรับข้อมูลผ่าน dropdown list ไม่ให้ user กรอกเองแบบอิสระ เพื่อป้องกันความผิดพลาดของ user ยกตัวอย่างเช่น user จะกรอกว่ากรุงเทพ และอยู่ในประเทศไทย แต่ user กรอกผิดเป็น ไทยย มีการกด ย เพิ่มเกินมา 1 ตัว ซึ่งนี่ถือว่าเป็นเรื่องใหญ่มากในระบบฐานข้อมูล ถ้าเรากดค้นหาทุกจังหวัดในประเทศ ไทย ตัวที่ user กรอกผิดก็จะไม่ออกมาด้วย
2. โยนภาระมาที่ frontend react ในการ create ต้อง request service 2 ครั้ง
3. Create suppertype โดยการ request หา geo graphic boundary (เป็น supper type) ให้สร้าง data ตัวใหม่ และส่ง payload ไปให้ service จะตอบ id ที่พึ่งสร้างพร้อม data ที่สร้างแล้วกลับมา
4. รอ step 1 ทำเสร็จ ต้อง await
5. Check ว่าเป็น geo boundary ประเภทใหน
6. ยกตัวอย่างเช่นเป็น state/จังหวัด ต้อง request ไปที่ state service (เป็น subtype) ที่จัดการ state table ให้สร้าง state ตัวใหม่เพิ่ม และส่ง payload ไปให้ ใน payload จะต้องแนบ id ตัวที่พึ่งสร้างสำเร็จไปด้วย

คำสั่งไว้สั่ง artisan ใน Laravel backend ของ project ที่อาจารย์ได้มอบหมาย

Microsoft Windows [Version 10.0.22631.4460]

(c) Microsoft Corporation. All rights reserved.

D:\scwork\year4\_final\proj\crud\_from\_datamodel\geo\_boundary\_fullstack\_docker>docker compose exec backend sh

time="2024-12-08T11:44:29+07:00" level=warning msg="D:\\scwork\\year4\_final\\proj\\crud\_from\_datamodel\\geo\_boundary\_fullstack\_docker\\docker-compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"

/var/www/html # php artisan make:controller Api/v1/GeographicBoundaryController --api

INFO Controller [app/Http/Controllers/Api/v1/GeographicBoundaryController.php] created successfully.

/var/www/html # php artisan make:model v1/GeographicBoundaryModel

INFO Model [app/Models/v1/GeographicBoundaryModel.php] created successfully.

/var/www/html # php artisan make:model v1/GeographicBoundaryTypeModel

/var/www/html # php artisan make:controller Api/v1/GeographicBoundaryTypeController --api

INFO Controller [app/Http/Controllers/Api/v1/GeographicBoundaryTypeController.php] created successfully.

/var/www/html # php artisan make:model v1/CountryModel

INFO Model [app/Models/v1CountryModel.php] created successfully.

/var/www/html # php artisan make:model v1/ProvinceModel

INFO Model [app/Models/v1/ProvinceModel.php] created successfully.

/var/www/html # php artisan make:model v1/CountryModel

INFO Model [app/Models/v1/CountryModel.php] created successfully.

/var/www/html # php artisan make:model v1/CityModel

INFO Model [app/Models/v1/CityModel.php] created successfully.

/var/www/html # php artisan make:model v1/CountyModel

INFO Model [app/Models/v1/CountyModel.php] created successfully.

/var/www/html # php artisan make:model v1/CountyCityModel

INFO Model [app/Models/v1/CountyCityModel.php] created successfully.

/var/www/html # php artisan make:model v1/PostalCodeModel

INFO Model [app/Models/v1/PostalCodeModel.php] created successfully.

/var/www/html # php artisan make:model v1/ProvinceModel

ERROR Model already exists.

/var/www/html # php artisan make:model v1/RegionModel

INFO Model [app/Models/v1/RegionModel.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/CityController --api

INFO Controller [app/Http/Controllers/Api/v1/CityController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/CountryController --

api

INFO Controller [app/Http/Controllers/Api/v1/CountryController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/CountyController --a

pi

INFO Controller [app/Http/Controllers/Api/v1/CountyController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/CountyCityController

--api

INFO Controller [app/Http/Controllers/Api/v1/CountyCityController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/PostalCodeController

--api

INFO Controller [app/Http/Controllers/Api/v1/PostalCodeController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/ProvinceController -

-api

INFO Controller [app/Http/Controllers/Api/v1/ProvinceController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/RegionController --a

pi

INFO Controller [app/Http/Controllers/Api/v1/RegionController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/SalesTerritoryContro

ller --api

INFO Controller [app/Http/Controllers/Api/v1/SalesTerritoryController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/ServiceTerritoryCont

roller --api

INFO Controller [app/Http/Controllers/Api/v1/ServiceTerritoryController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/StateController --ap

i

INFO Controller [app/Http/Controllers/Api/v1/StateController.php] created successfully.

/var/www/html # php artisan make:controller Api/v1/TerritoryController

--api

INFO Controller [app/Http/Controllers/Api/v1/TerritoryController.php] created successfully.

/var/www/html # php artisan make:model v1/SalesTerritoryModel

INFO Model [app/Models/v1/SalesTerritoryModel.php] created successfully.

/var/www/html # php artisan make:model v1/ServiceTerritoryModel

INFO Model [app/Models/v1/ServiceTerritoryModel.php] created successfully.

/var/www/html # php artisan make:model v1/StateModel

INFO Model [app/Models/v1/StateModel.php] created successfully.

/var/www/html # php artisan make:model v1/TerritoryModel

INFO Model [app/Models/v1/TerritoryModel.php] created successfully.

เพื่อให้ง่ายในการอ่าน code นี่คือ theme ที่ใช้ใน program vs code

A screenshot of a computer

Description automatically generated

สาเหตุที่ตั้งเป็น theme ขาวเพราะว่ามันดูธรรมดาเหมือนเครื่องคอมของคนทั่วๆ ไป และการแสดงผล GUI ของแต่ละ program สีขาวก็มักจะเป็นสี default อยู่แล้ว ทำให้ bug ยากกว่า theme dark

controller

<?php

namespace App\Http\Controllers\Api\v1;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\v1\GeographicBoundaryModel as Model;

use Illuminate\Validation\ValidationException;

class GeographicBoundaryController extends Controller

{

    /\*\*

*\* Display a listing of the resource.*

\*/

    public function index()

    {

        // *ดึงข้อมูลพร้อมความสัมพันธ์ 'type'*

        $geo = Model::with(['geoType', 'specific'])->get();

        // *ส่งข้อมูลในรูปแบบ JSON Response*

        return response()->json($geo, 200);

    }

    /\*\*

*\* Store a newly created resource in storage.*

\*/

    public function store(Request $request)

    {

        //

        try {

            $request->validate([

                'geo\_code' => 'nullable|string|max:50|unique:geographic\_boundary,geo\_code',

                'name' => 'required|string|max:255|unique:geographic\_boundary,name',

                'abbreviation' => 'nullable|string|max:50',

                'type\_id' => 'required|integer|exists:geographic\_boundary\_type,type\_id',

            ]);

            $geo = Model::create($request->all());

            return response()->json($geo, 201);

        } catch (ValidationException $e) {

            return response()->json([

                'error' => 'Validation failed',

                'messages' => $e->errors(),

            ], 422);

        }

    }

    /\*\*

*\* Display the specified resource.*

\*/

    public function show(string $id)

    {

        //

        $item = Model::with('geoType')->findOrFail($id);

        return response()->json($item);

    }

    /\*\*

*\* Update the specified resource in storage.*

\*/

    public function update(Request $request, string $id)

    {

        //

        $item = Model::find($id);

        if (!$item) {

            return response()->json(['error' => 'Product not found'], 404);

        }

        $originalData = $item->toArray();

        try {

            $request->validate([

                'geo\_code' => 'nullable|string|max:50|unique:geographic\_boundary,geo\_code',

                'name' => 'required|string|max:255|unique:geographic\_boundary,name',

                'abbreviation' => 'nullable|string|max:50',

                'type\_id' => 'required|integer|exists:geographic\_boundary\_type,type\_id',

            ]);

            $item->update($request->all());

            return response()->json([

                'original\_data' => $originalData,

                'updated\_data' => $item

            ], 200);

        } catch (ValidationException $e) {

            return response()->json([

                'error' => 'Validation failed',

                'messages' => $e->errors(),

            ], 422);

        }

    }

    /\*\*

*\* Remove the specified resource from storage.*

\*/

    public function destroy(string $id)

    {

        // *ตรวจสอบว่ามี item อยู่ในฐานข้อมูลหรือไม่*

        $item = Model::find($id);

        if (!$item) {

            return response()->json(['error' => 'Product not found'], 404);

        }

        // *เก็บข้อมูลก่อนลบ*

        $deletedData = $item->toArray();

        // *ลบข้อมูล*

        $item->delete();

        // *ส่งข้อมูลที่ถูกลบกลับไป*

        return response()->json([

            'deleted\_data' => $deletedData

        ], 200);

    }

}

นี่คือ model

<?php

namespace App\Models\v1;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use App\Models\v1\GeographicBoundaryTypeModel;

class GeographicBoundaryModel extends Model

{

    use HasFactory;

    protected $table = 'geographic\_boundary';

    protected $primaryKey = 'geo\_id';

    protected $fillable = [

        'geo\_code',

        'name',

        'type\_id'

    ];

    public const CREATED\_AT = null;

    public const UPDATED\_AT = null;

    public function geoType()

    {

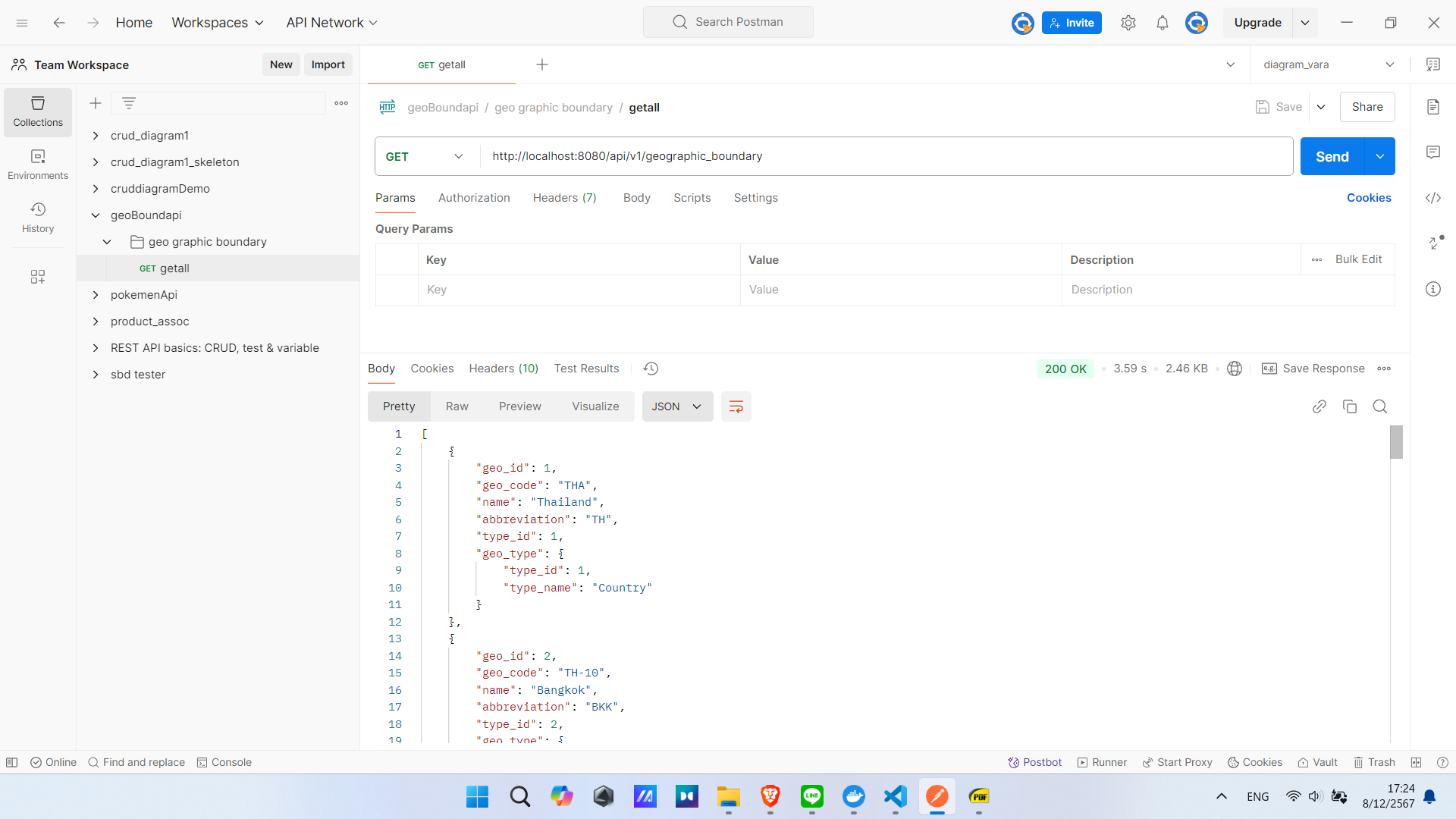
        return $this->belongsTo(GeographicBoundaryTypeModel::class, 'type\_id');

    }

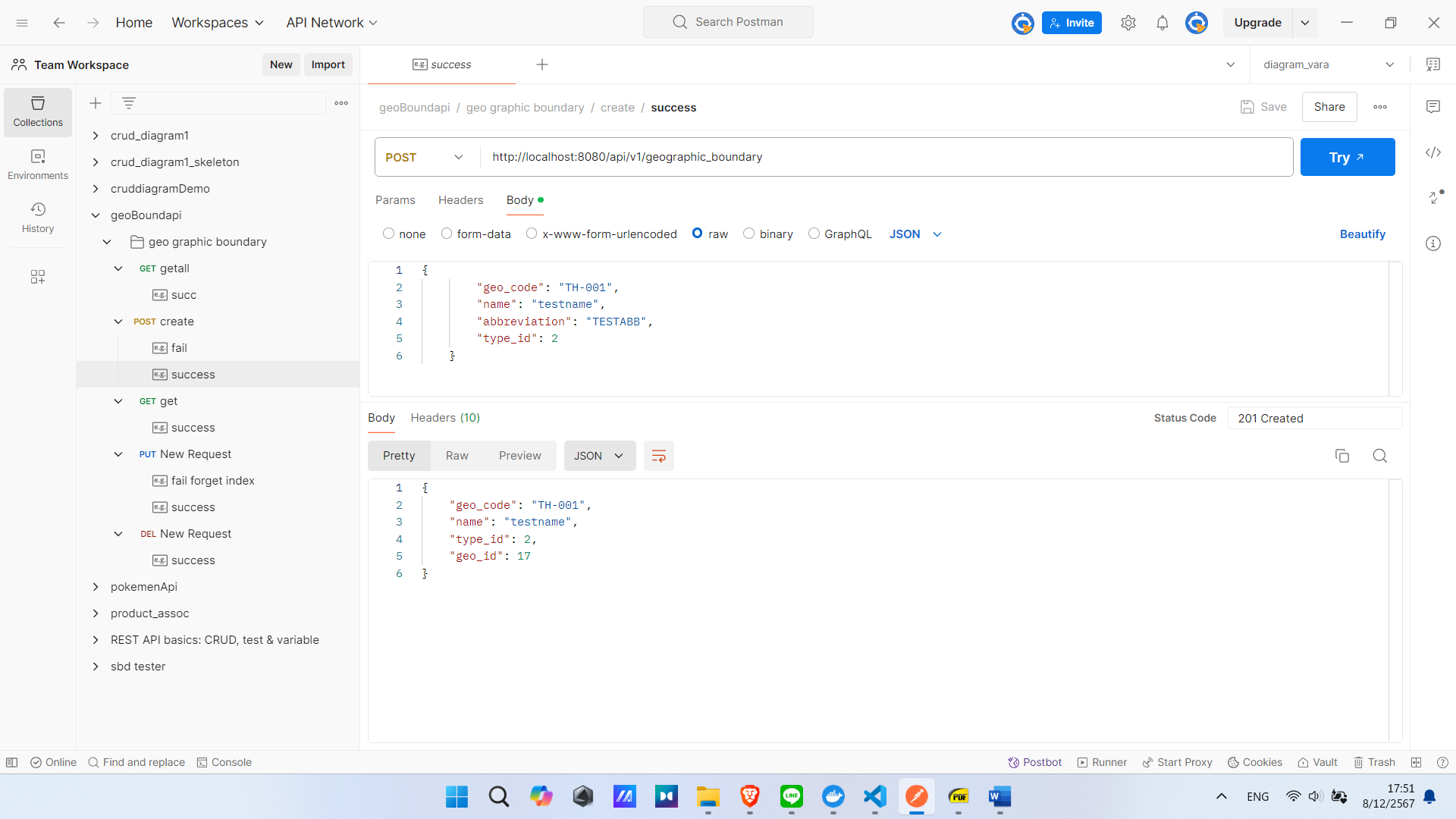
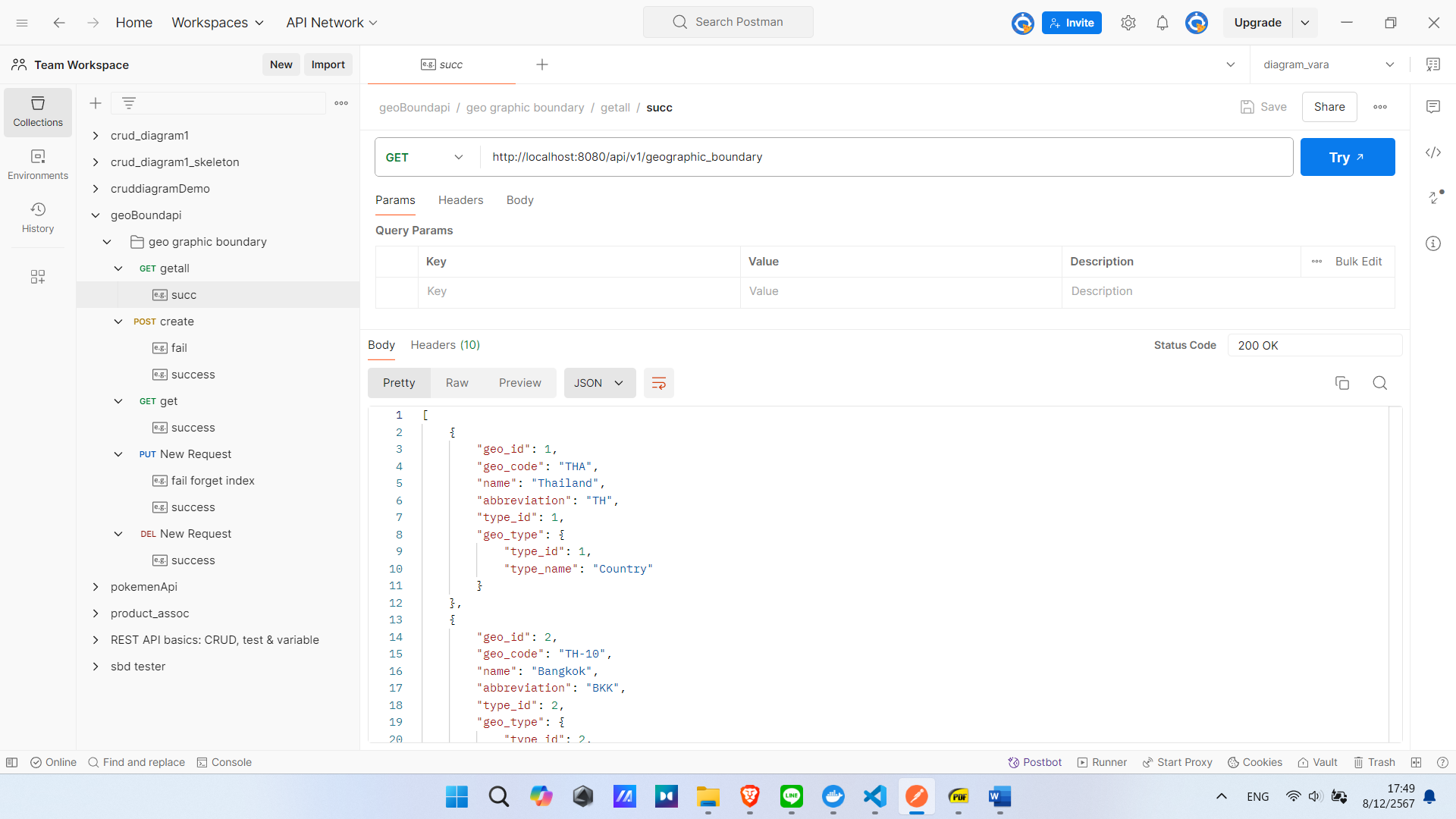
    // *ใช้ Inheritance กับ Subtype Models*

    // *Morph to Specific Subtypes*

}

ทดสอบการทำงานของ api ด้วย program postman

Api ส่วนที่ทดสอบคือ api suppertype ที่ชื่อว่า geo graphic boundary เท่านั้น ขั้นต่อไปจะต้องทำ service subtype ด้วย



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

การทดสอบตามรูปดังกล่าวเป็นเทคนิคการทำ unit test ตามที่ อาจารย์ เพชรารัตน์ สอนในวิชา software quality assurance