Model–View–Controller (MVC) Laravel API

Web Development and Security (ZEIT3119)

Week 8

Dr. Reza Rafeh

Revision

slido

Join at

slido.com

#3986 139

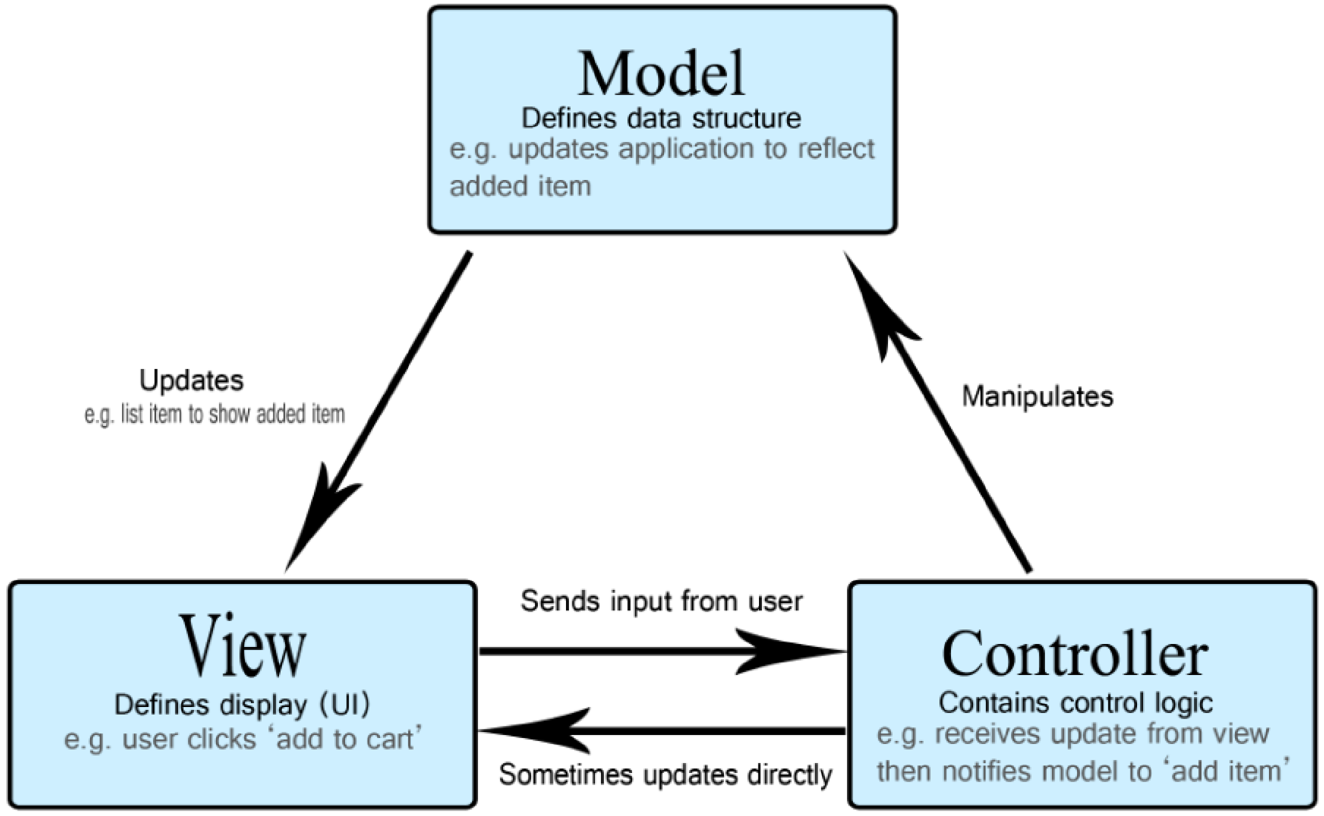
Outline

- Model–View–Controller (MVC)
- Laravel API
- Model
- Migration
- Controller
- CRUD Action Methods
- Eloquent
- Testing APIs with Postman
- Views
- Seeders

Model-View-Controller (MVC)

- MVC is a pattern for implementing user interfaces, data, and controlling logic
- MVC emphasizes a separation between the software's business logic and display.
- It helps in improving maintenance and a clearer separation of task among team members
- Other design patterns based on MVC:
  - MVVM (Model-View-View-Model)
  - MVP (Model-View-Presenter)
  - MVW (Model-View-Whatever)
- Three parts of MVC:
  - Model: Manages data and business logic
  - View: Handles layout and display
  - Controller: Routes commands to the model and view parts

MVC Example for a Shopping List App


Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Source: https://developer.mozilla.org/en-US/docs/Glossary/MVC

MVC on the Web

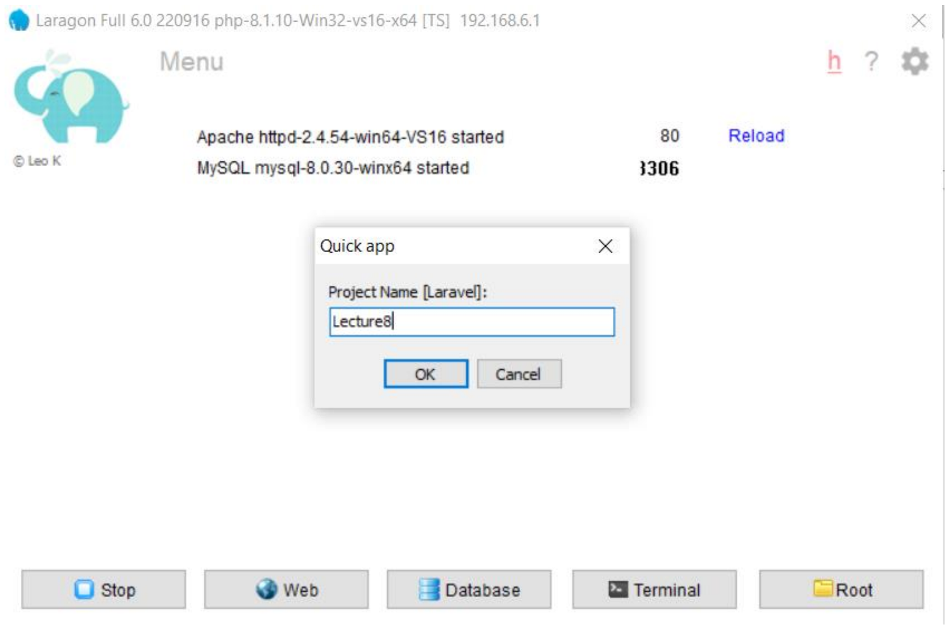
- Model: MySQL
- Control: JavaScript, PHP
- View: HTML, CSS
- Early days of Web:
  - MVC architecture was mostly implemented on the server-side
  - The client requested updates via forms or links
  - Updated views received and displayed on the browser
- These days:
  - More logic is pushed to the client side using client-side data stores, XMLHttpRequest, and allowing partial page updates as required

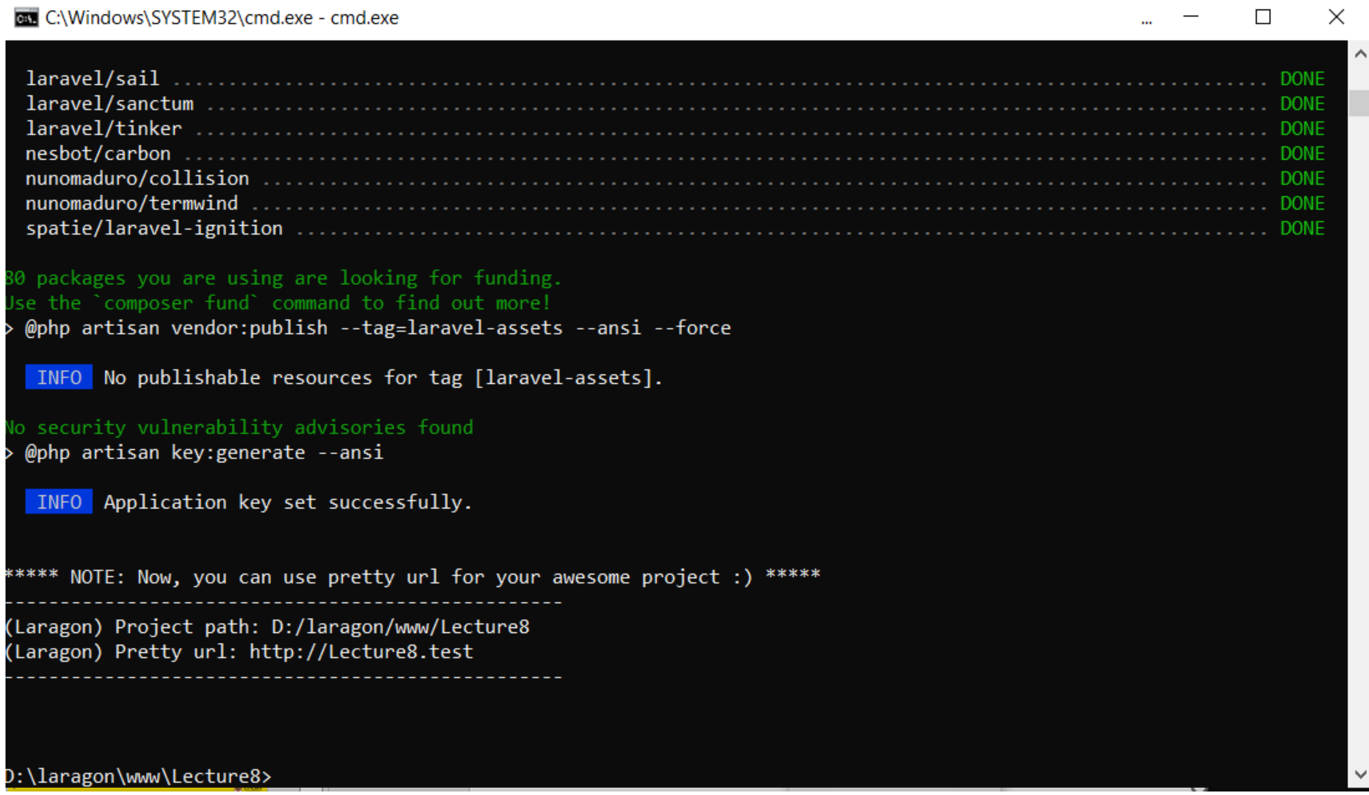
Laravel

- **Laravel** is an **open-source PHP** web framework designed for creating web applications that follow the **Model–View–Controller (MVC)** architectural pattern. We will primarily use the **Model** and **Controller** in this module.
- Extra Reading: https://developer.mozilla.org/en-US/docs/Glossary/MVC

Create Laravel API

Right-click on the window > Quick App > Laravel. You will be presented with another window prompting you to name the application. Once named, click the **OK** button.


Laragon Full 6.0 220916 php-8.1.10-Win32-vs16-x64 [TS] 192.168.6.1
Menu
Apache httpd-2.4.54-win64-VS16 started 80 Reload
MySQL mysql-8.0.30-winx64 started 3306
© Leo K
Quick app
Project Name [Laravel]:
Lecture8
OK
Cancel
Stop
Web
Database
Terminal
Root


C:\Windows\SYSTEM32\cmd.exe - cmd.exe

```
  laravel/sail ........................................................................................ DONE
  laravel/sanctum ..................................................................................... DONE
  laravel/tinker ...................................................................................... DONE
  nesbot/carbon ....................................................................................... DONE
  nunomaduro/collision ................................................................................ DONE
  nunomaduro/termwind ................................................................................. DONE
  spatie/laravel-ignition ............................................................................. DONE

80 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

  INFO  No publishable resources for tag [laravel-assets].

No security vulnerability advisories found
> @php artisan key:generate --ansi

  INFO  Application key set successfully.


***** NOTE: Now, you can use pretty url for your awesome project :) *****
-------------------------------------------------------
(Laragon) Project path: D:/laragon/www/Lecture8
(Laragon) Pretty url: http://Lecture8.test
-------------------------------------------------------


D:\laragon\www\Lecture8>
```

Laragon Directory Structure

- **app:** contains the core code such as controllers, models, providers.
- **config:** contains all of your project's configuration files such as auth, caching, database.
- **routes:** contains all of your project's route definitions. We are more concerned with api.php.

Laravel Modules

- **Composer** is the dependency manager for **Laravel** much like **npm** for **Node**. This comes by default. You should not need to install additional packages.
- **Artisan** is the command line interface included with **Laravel**. It provides useful commands that can help you while you are building your project.

Connecting to MySQL

- In the .env file, modify your database credentials so that your project is connected to MySQL.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=lecture8
DB_USERNAME=root
DB_PASSWORD=
```

Model

- A model class represents the logical structure and relationship of a database table. In Laravel, each table corresponds to a model. A model allows you to retrieve, create, update and delete data.
- To create a new model and migration, run the following command:

  php artisan make:model Institution --migration

- Note: this should create a directory called **Models**. If it does not, create the directory and copy Institution.php and User.php into it. Additionally, you will need to change the namespace from App to App\Models.
- In app\Models\Institution.php, specify the columns you wish to interact with. For example:

```
class Institution extends Model
{
    use HasFactory;
    protected $fillable = ['id','name', 'region', 'country'];
}
```

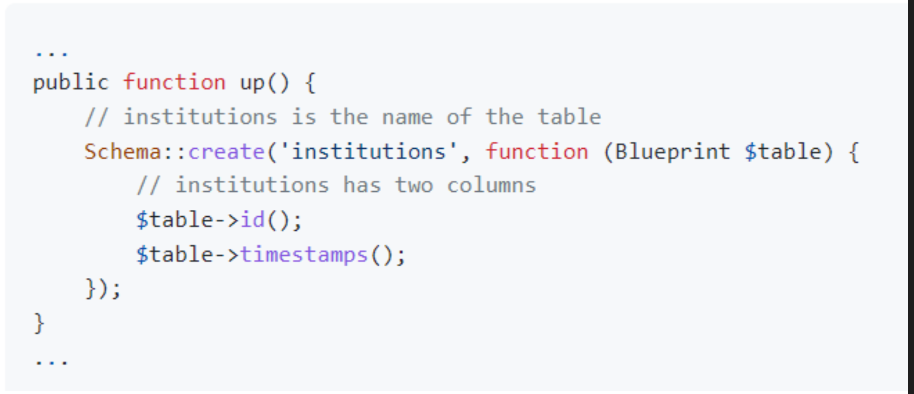
Migration

To make id unique:

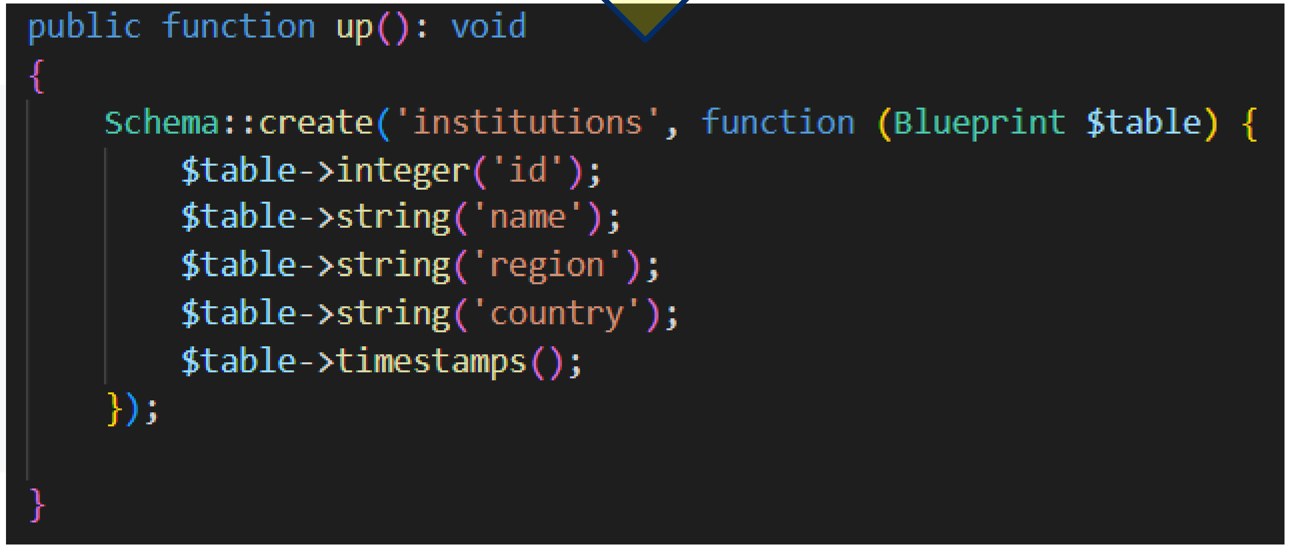
;

```
$table->integer('id')->unique
```

- You can think of migrations like version control for your database. They allow you to define and share the application's schema definitions.
- In the database\migrations directory, you will see a migration file for the Institution model class.



```
...
public function up() {
    // institutions is the name of the table
    Schema::create('institutions', function (Blueprint $table) {
        // institutions has two columns
        $table->id();
        $table->timestamps();
    });
}
...
```



```
public function up(): void
{
    Schema::create('institutions', function (Blueprint $table) {
        $table->integer('id');
        $table->string('name');
        $table->string('region');
        $table->string('country');
        $table->timestamps();
    });

}
```

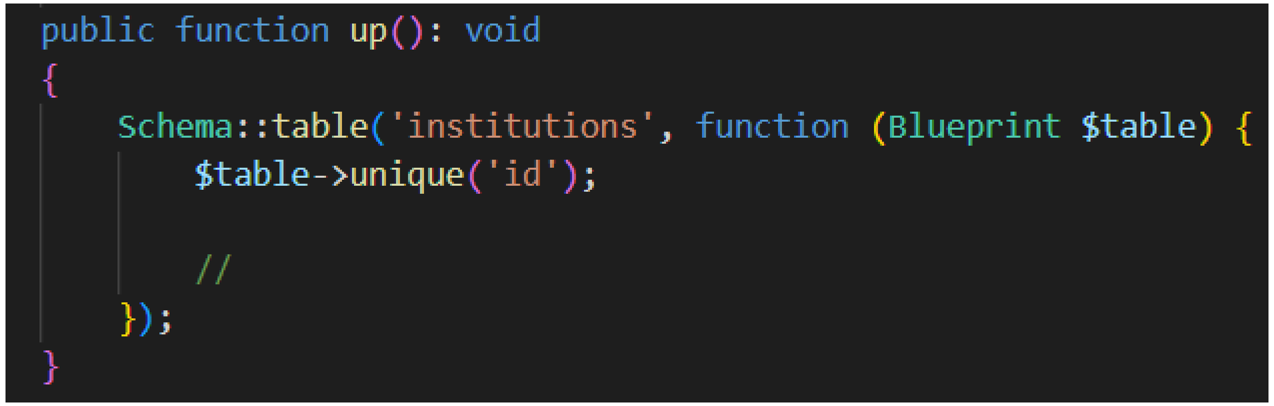
- Modify this migration file by adding a column for id, name, region and country. Id is integer and all other three columns are of type string.

Migration (Cont.)

- **up:** The up method is used to add new tables, columns, or indexes to your database,
- **down:** The down method should reverse the operations performed by the up method.
- To update a table, create another migration file:

  php artisan make:migration update_institutions_table --table=institutions

- Then, you can update columns or indexes, e.g. to make the id column unique:



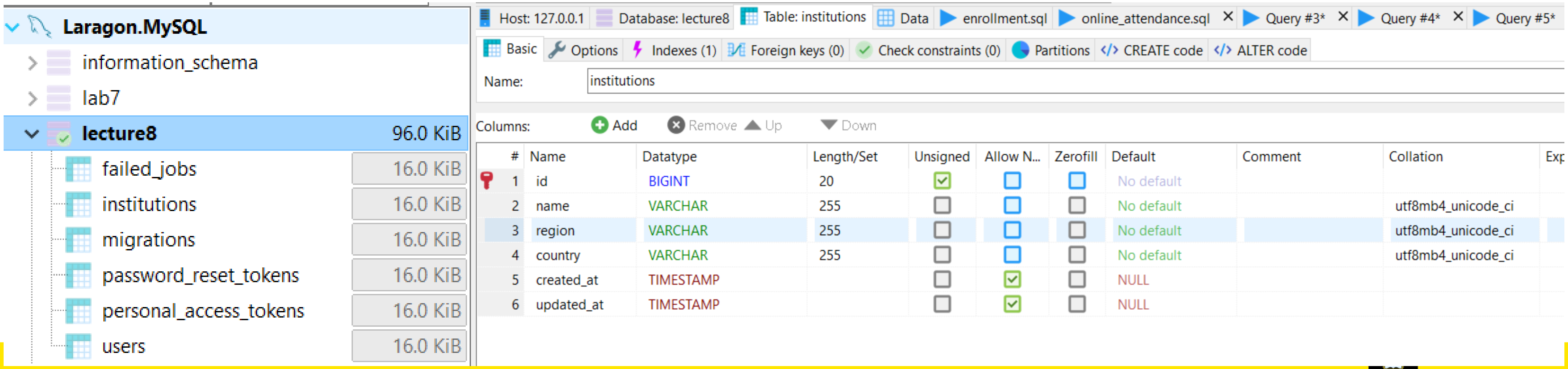
```
public function up(): void
{
    Schema::table('institutions', function (Blueprint $table) {
        $table->unique('id');

        //
    });
}
```

Migration – Cont.

➢ If you change a migration file, you will have an outstanding migration. This means that your database schema will not reflect the columns specified in your migration file. To run all outstanding migrations, run the following command:

php artisan migrate

➢ Go to your MySQL database and refresh the window. You should see six tables:



Laragon.MySQL
- information_schema
- lab7
- lecture8 — 96.0 KiB
  - failed_jobs — 16.0 KiB
  - institutions — 16.0 KiB
  - migrations — 16.0 KiB
  - password_reset_tokens — 16.0 KiB
  - personal_access_tokens — 16.0 KiB
  - users — 16.0 KiB

Host: 127.0.0.1 | Database: lecture8 | Table: institutions | Data | enrollment.sql | online_attendance.sql | Query #3* | Query #4* | Query #5*

Basic | Options | Indexes (1) | Foreign keys (0) | Check constraints (0) | Partitions | CREATE code | ALTER code

Name: institutions

Columns: Add | Remove | Up | Down

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Default	Comment	Collation
1	id	BIGINT	20	✓			No default		
2	name	VARCHAR	255				No default		utf8mb4_unicode_ci
3	region	VARCHAR	255				No default		utf8mb4_unicode_ci
4	country	VARCHAR	255				No default		utf8mb4_unicode_ci
5	created_at	TIMESTAMP			✓		NULL		
6	updated_at	TIMESTAMP			✓		NULL		

Controller

- A controller class contains public action methods used to handle various HTTP methods, i.e., GET, POST, PUT and DELETE. These action methods handle incoming requests, retrieve the necessary model data and return the appropriate responses.
- Create a new controller by running the following command:

  php artisan make:controller InstitutionController --api

- In the app\Http\Controllers directory, you will find all your controllers including InstitutionController.php.

CRUD Action Methods

➢ In InstitutionController.php, you will see the following CRUD action methods:

```
...
class InstitutionController extends Controller {
    public function index() {
        // Some code
    }

    public function store(Request $request) {
        // Some code
    }

    public function show($id) {
        // Some code
    }

    public function update(Request $request, $id) {
        // Some code
    }

    public function destroy($id) {
        // Some code
    }
}
```

Import the Institution model to use the institutions table

```
...
use App\Models\Institution;

class InstitutionController extends Controller {
    ...
}
```

Eloquent

- Eloquent is an Object-Relational Mapping (ORM) that allows you to query & manipulate data using an Object-Oriented programming language
- Each web framework has one or more ORMs which encapsulate the code needed to query & manipulate data
- So, you do not need to use SQL. You interact directly with an object in the same programming language you are using, i.e., PHP.

Eloquent (Cont.)

Read All

```
public function index() {
    return Institution::all();

    // SQL equivalent: SELECT * FROM institutions;
}
```

Read One

```
public function show($id) {
    return Institution::find($id);

    // SQL equivalent: SELECT * FROM institutions WHERE id = $id;
}
```

Create

```
public function store(Request $request) {
    return Institution::create($request->all());

    // SQL equivalent:
    // INSERT INTO institutions
    // VALUES ($request->name, $request->region, $request->country);
}
```

Eloquent (Cont.)

Update

```
public function update(Request $request, $id) {
    $institution = Institution::find($id);
    $institution->update($request->all());
    return $institution;

    // SQL equivalent:
    // UPDATE institutions
    // SET name = $request->name, region = $request->region, country = $request->country
    // WHERE id = $id;
}
```

Delete

```
public function destroy($id) {
    return Institution::destroy($id);

    // SQL equivalent:
    // DELETE FROM institutions
    // WHERE id = $id;
}
```

Route

- In the routes directory, open the api.php file & create the following API endpoints:

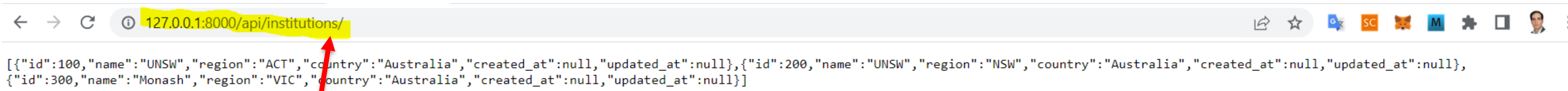
```
...
Route::group(['prefix' => 'institutions'], function () {
    Route::get('/', [InstitutionController::class, 'index']);
    Route::get('/{id}', [InstitutionController::class, 'show']);
    Route::post('/', [InstitutionController::class, 'store']);
    Route::put('/{id}', [InstitutionController::class, 'update']);
    Route::delete('/{id}', [InstitutionController::class, 'destroy']);
});
```

- Make sure you import **InstitutionController** by adding the following statement:

```
...

use App\Http\Controllers\InstitutionController; // If you do not add this statement, you will
                                                // not have access to the action methods in this controller
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;


...
```

Note: All routes in api.php are prefixed with /api.

Test API in the Browser

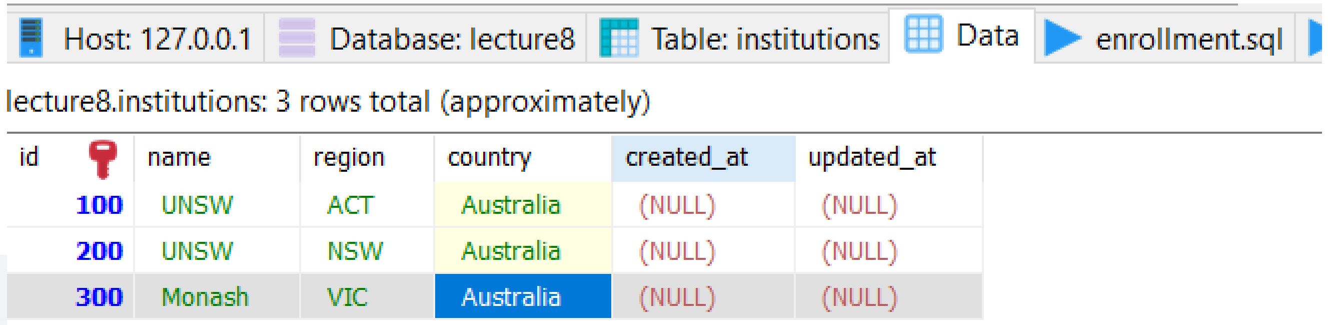


127.0.0.1:8000/api/institutions/

```
[{"id":100,"name":"UNSW","region":"ACT","country":"Australia","created_at":null,"updated_at":null},{"id":200,"name":"UNSW","region":"NSW","country":"Australia","created_at":null,"updated_at":null},
{"id":300,"name":"Monash","region":"VIC","country":"Australia","created_at":null,"updated_at":null}]
```

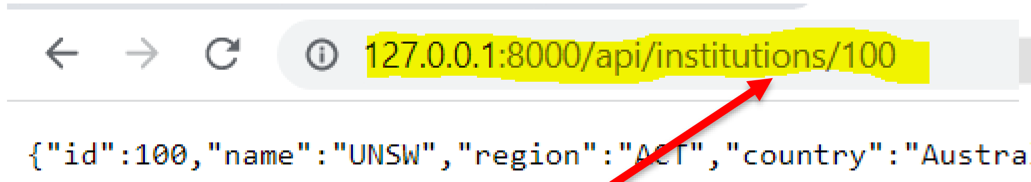
```php
public function index()
{
    return Institution::all();
}
```

```php
Route::get('/', [InstitutionController::class, 'index']);
```



lecture8.institutions: 3 rows total (approximately)

id	name	region	country	created_at	updated_at
100	UNSW	ACT	Australia	(NULL)	(NULL)
200	UNSW	NSW	Australia	(NULL)	(NULL)
300	Monash	VIC	Australia	(NULL)	(NULL)



127.0.0.1:8000/api/institutions/100

```
{"id":100,"name":"UNSW","region":"ACT","country":"Australia","created_at":null,"updated_at":null}
```

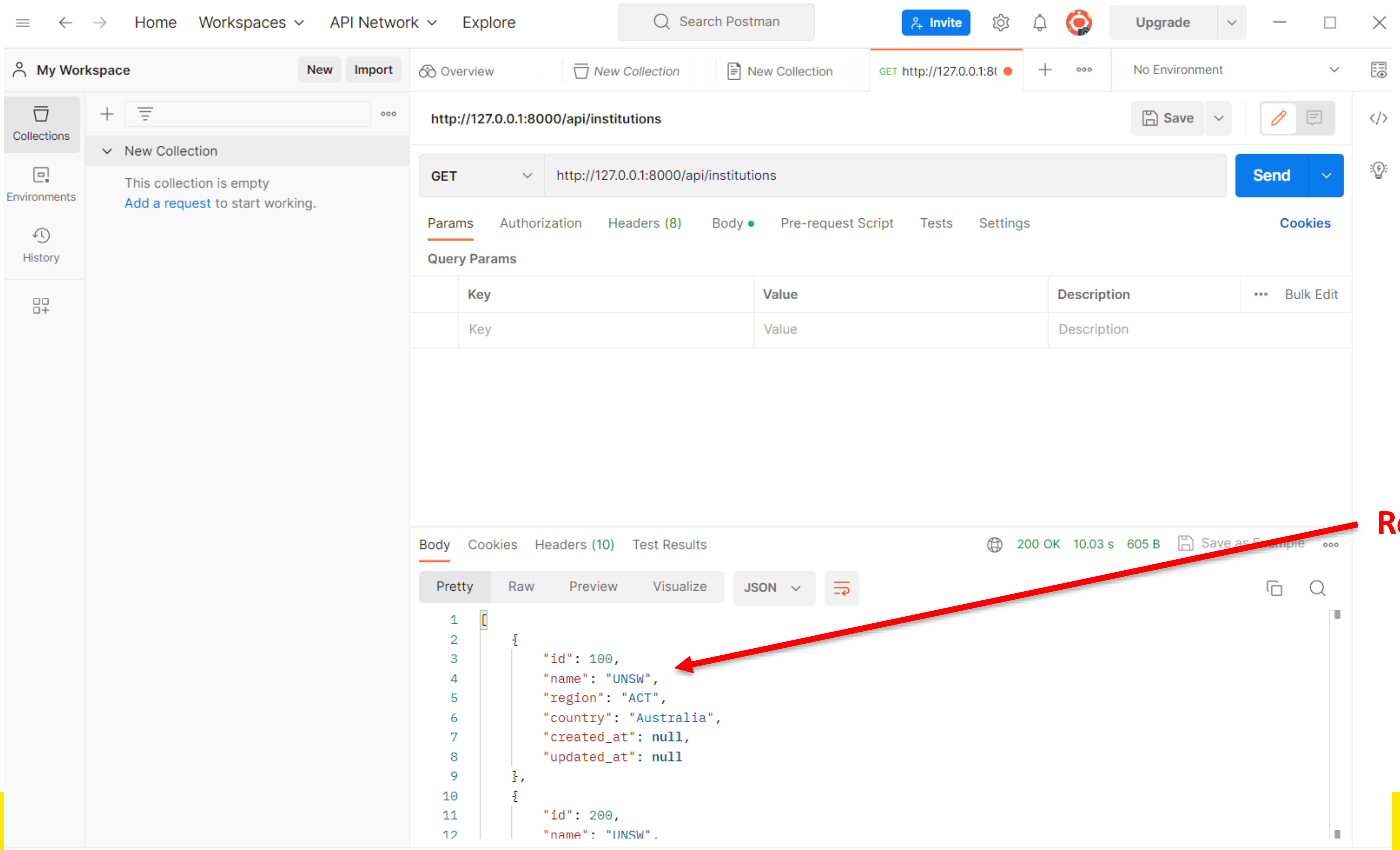
```php
public function show($id)
{
    return Institution::find($id);
}
```

```php
Route::get('/{id}', [InstitutionController::class, 'show']);
```

Postman

- Postman is an API Platform for developers to design, build, test and iterate their APIs.
- As of February 2023, more than 25 million registered users and 75,000 open APIs use Postman.
- It also has abilities to document APIs.
- It comes as Browser and Desktop versions.

Test API in Postman - get



http://127.0.0.1:8000/api/institutions

GET http://127.0.0.1:8000/api/institutions

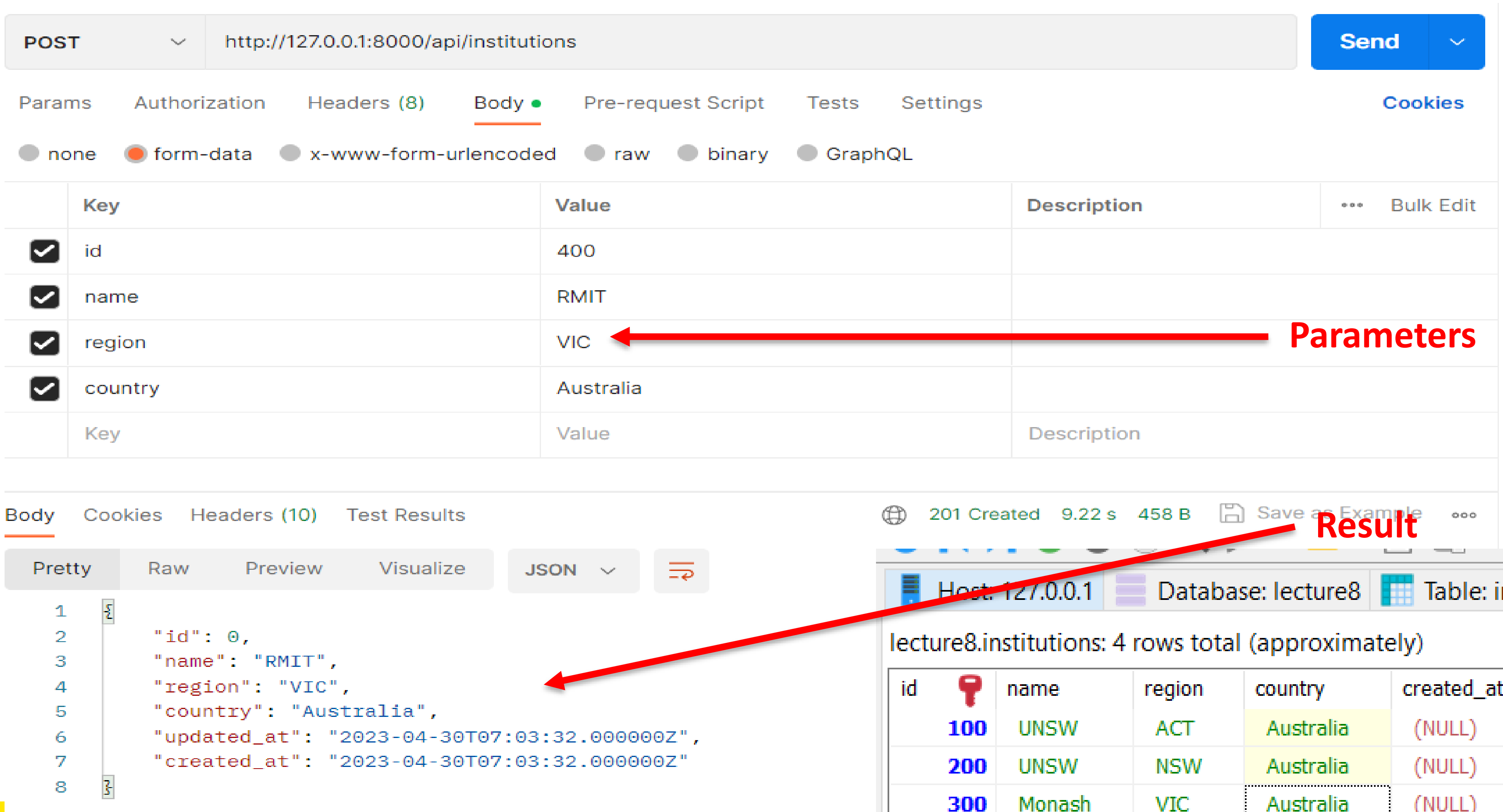
200 OK 10.03 s 605 B

```
[
    {
        "id": 100,
        "name": "UNSW",
        "region": "ACT",
        "country": "Australia",
        "created_at": null,
        "updated_at": null
    },
    {
        "id": 200,
        "name": "UNSW",
```

To test APIs on the local host, only Postman desktop can be used

Result

Test API in Postman - post



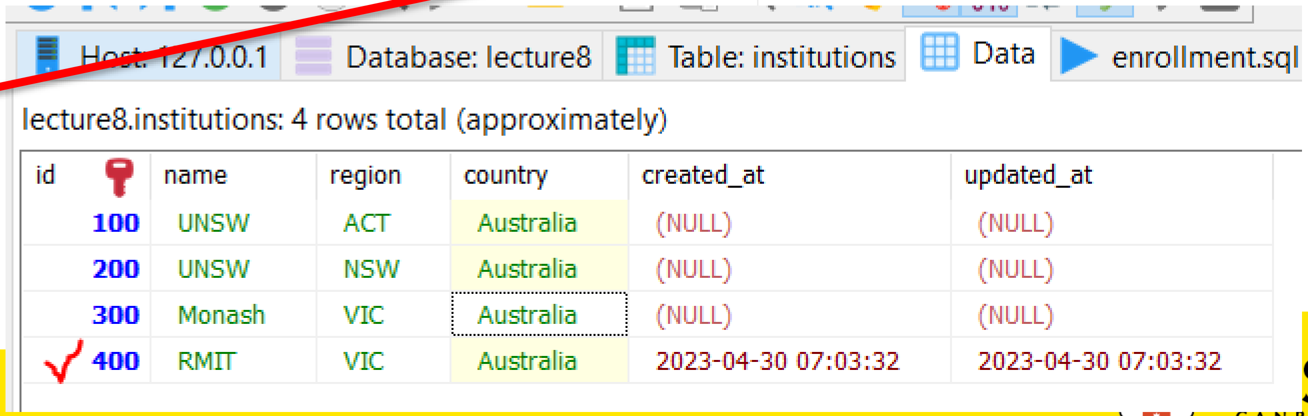
POST http://127.0.0.1:8000/api/institutions

none [✓ form-data] x-www-form-urlencoded raw binary GraphQL

	Key	Value	Description
✓	id	400	
✓	name	RMIT	
✓	region	VIC	
✓	country	Australia	
	Key	Value	Description

201 Created 9.22 s 458 B

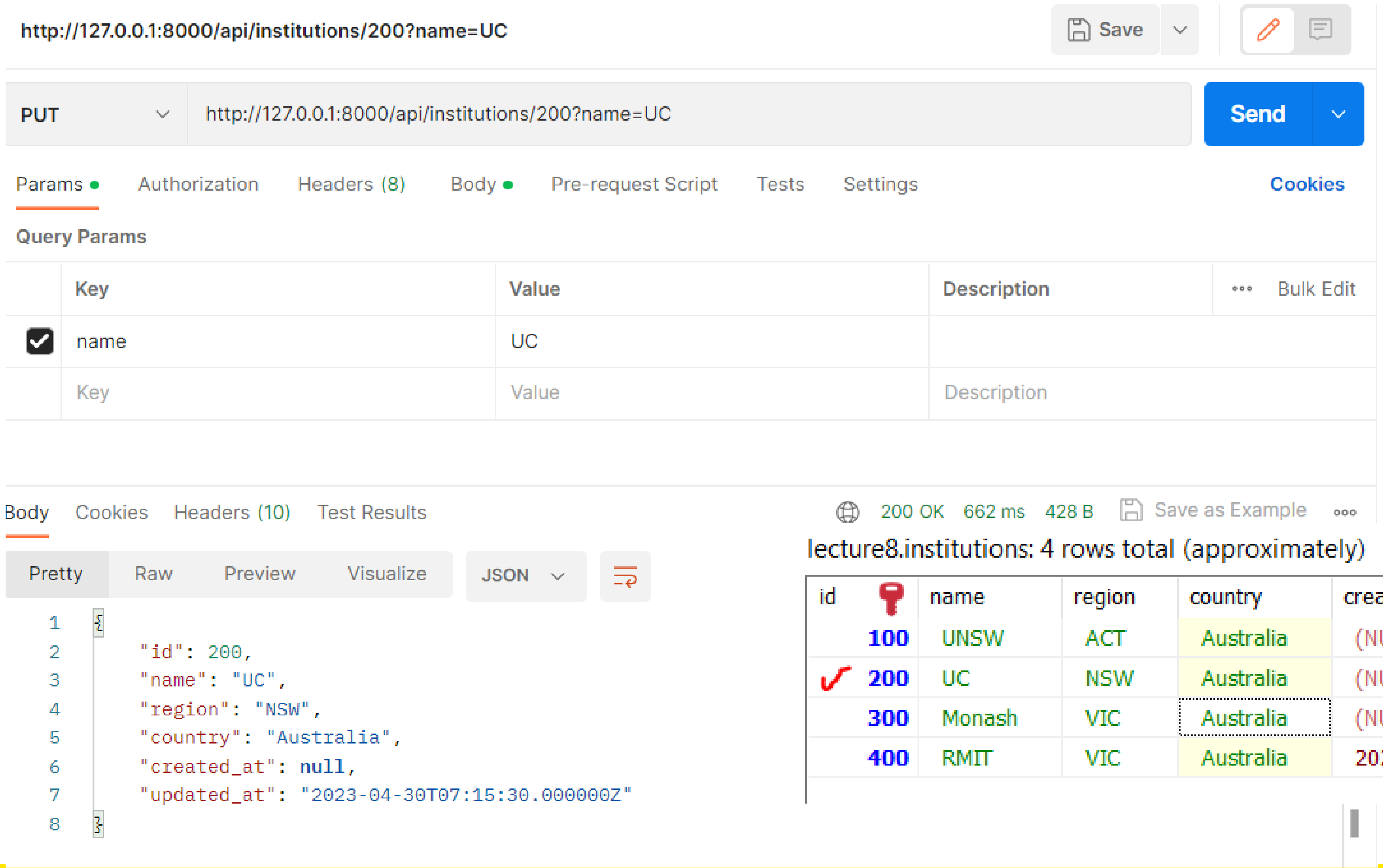
```
{
    "id": 0,
    "name": "RMIT",
    "region": "VIC",
    "country": "Australia",
    "updated_at": "2023-04-30T07:03:32.000000Z",
    "created_at": "2023-04-30T07:03:32.000000Z"
}
```



lecture8.institutions: 4 rows total (approximately)

id	name	region	country	created_at	updated_at
100	UNSW	ACT	Australia	(NULL)	(NULL)
200	UNSW	NSW	Australia	(NULL)	(NULL)
300	Monash	VIC	Australia	(NULL)	(NULL)
✓ 400	RMIT	VIC	Australia	2023-04-30 07:03:32	2023-04-30 07:03:32

Test API in Postman - put



http://127.0.0.1:8000/api/institutions/200?name=UC

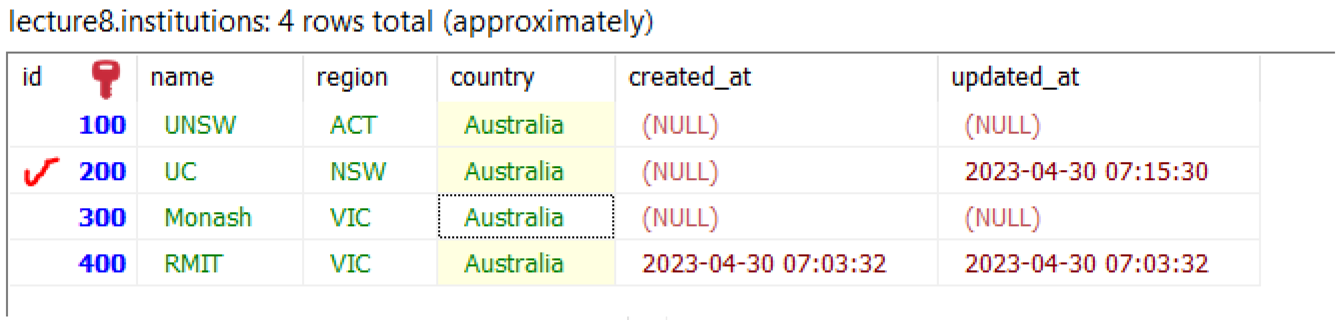
PUT http://127.0.0.1:8000/api/institutions/200?name=UC

Query Params

	Key	Value	Description
✓	name	UC	

200 OK 662 ms 428 B

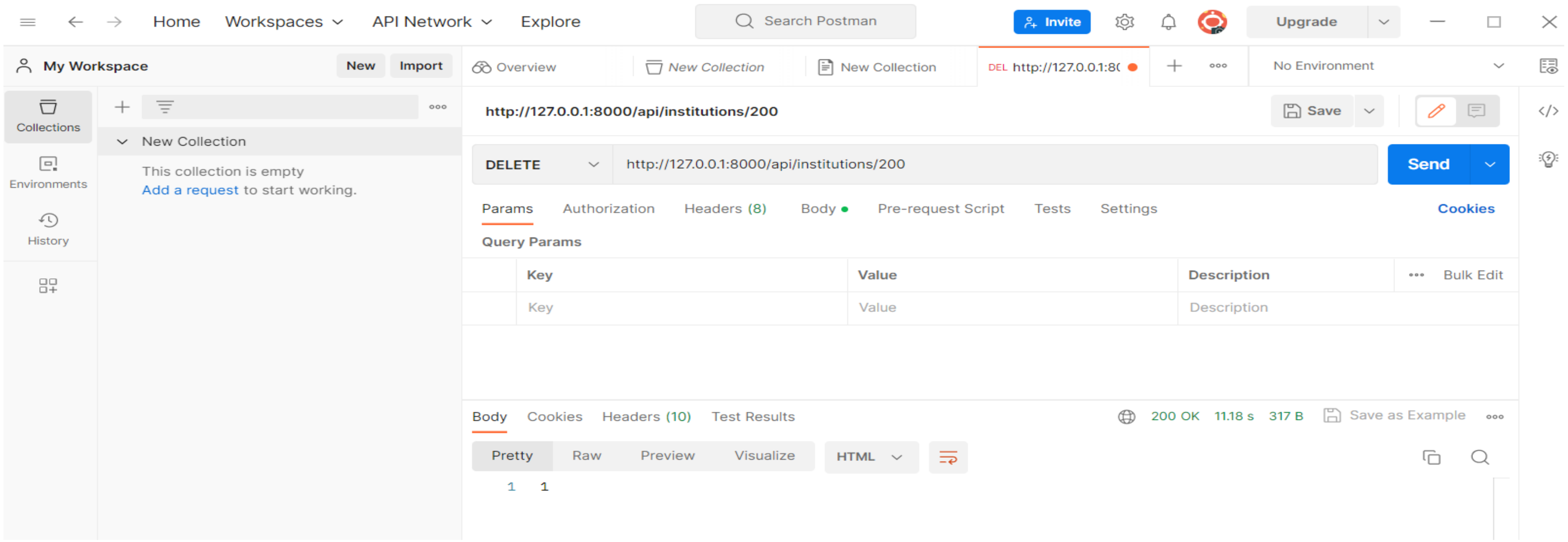
```
{
    "id": 200,
    "name": "UC",
    "region": "NSW",
    "country": "Australia",
    "created_at": null,
    "updated_at": "2023-04-30T07:15:30.000000Z"
}
```



lecture8.institutions: 4 rows total (approximately)

	id	name	region	country	created_at	updated_at
	100	UNSW	ACT	Australia	(NULL)	(NULL)
✓	200	UC	NSW	Australia	(NULL)	2023-04-30 07:15:30
	300	Monash	VIC	Australia	(NULL)	(NULL)
	400	RMIT	VIC	Australia	2023-04-30 07:03:32	2023-04-30 07:03:32

Test API in Postman - delete



New Collection

This collection is empty

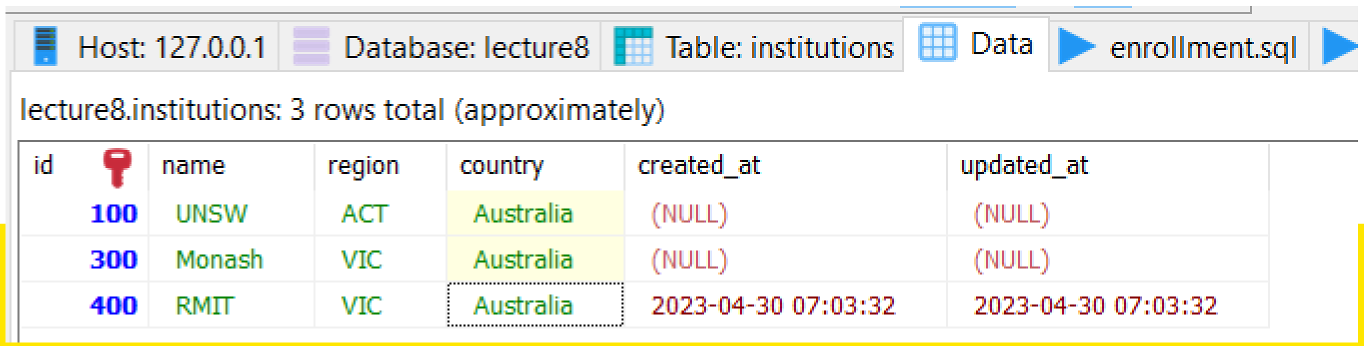
Add a request to start working.

http://127.0.0.1:8000/api/institutions/200

DELETE http://127.0.0.1:8000/api/institutions/200

200 OK 11.18 s 317 B

```
1
```

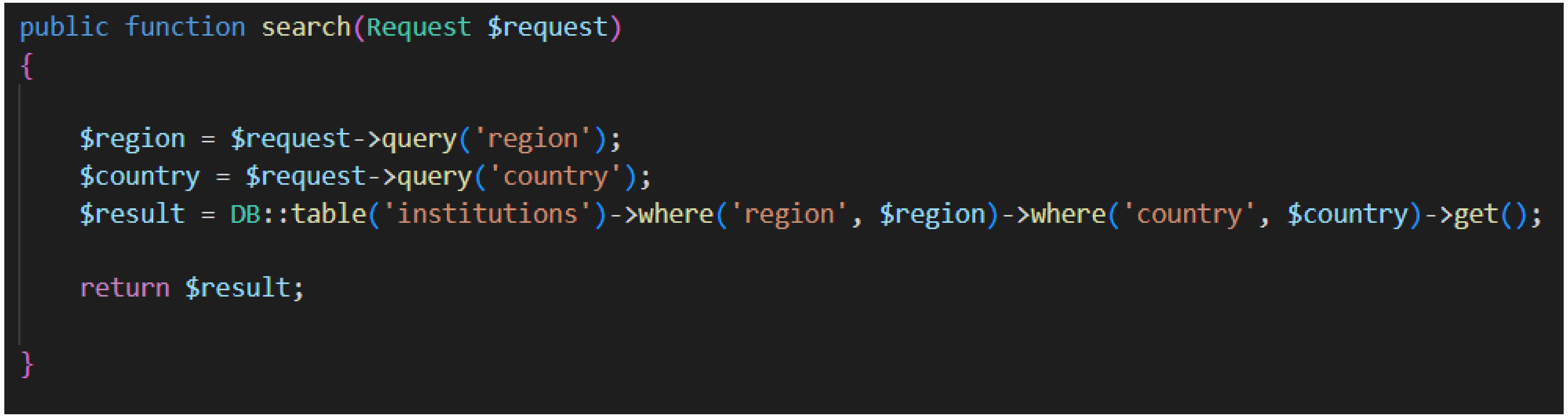


lecture8.institutions: 3 rows total (approximately)

id	name	region	country	created_at	updated_at
100	UNSW	ACT	Australia	(NULL)	(NULL)
300	Monash	VIC	Australia	(NULL)	(NULL)
400	RMIT	VIC	Australia	2023-04-30 07:03:32	2023-04-30 07:03:32

Using Queries

- To pass several parameters to an API function, an object of type Request can be defined. For example, we want to have a search function that searches institutions based on region and country:

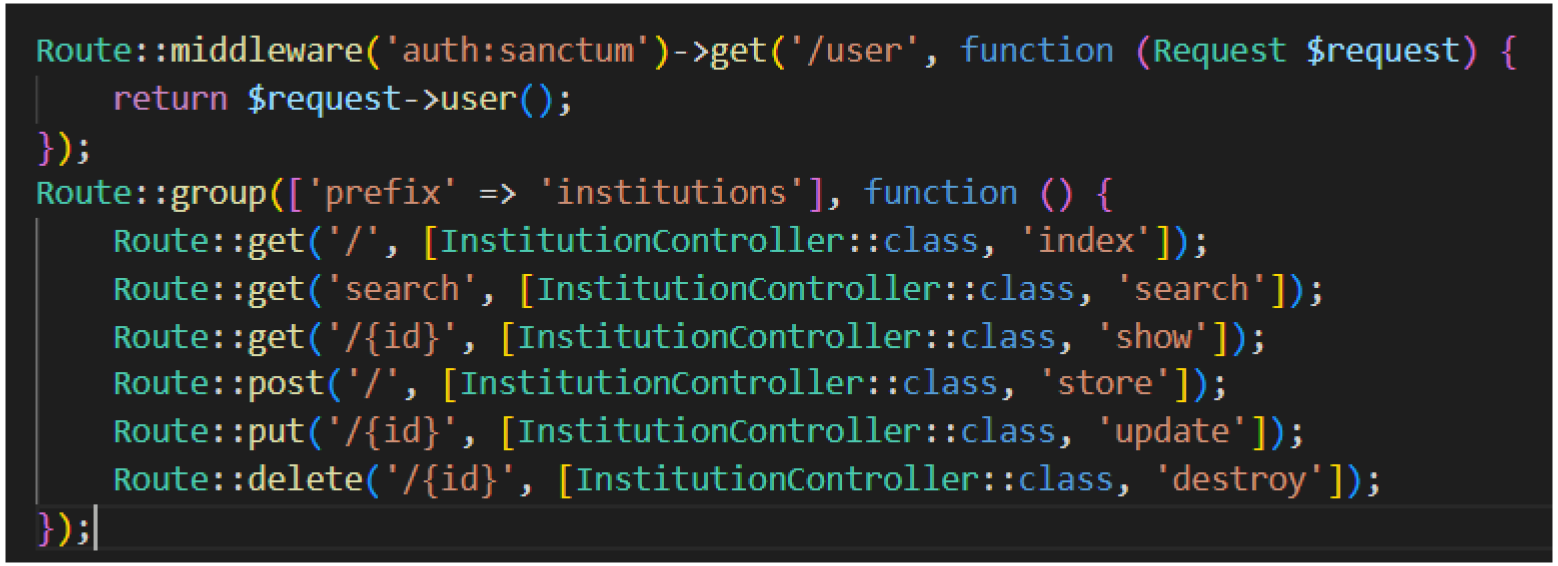


```
public function search(Request $request)
{

    $region = $request->query('region');
    $country = $request->query('country');
    $result = DB::table('institutions')->where('region', $region)->where('country', $country)->get();

    return $result;

}
```

- https://laravel.com/docs/10.x/queries

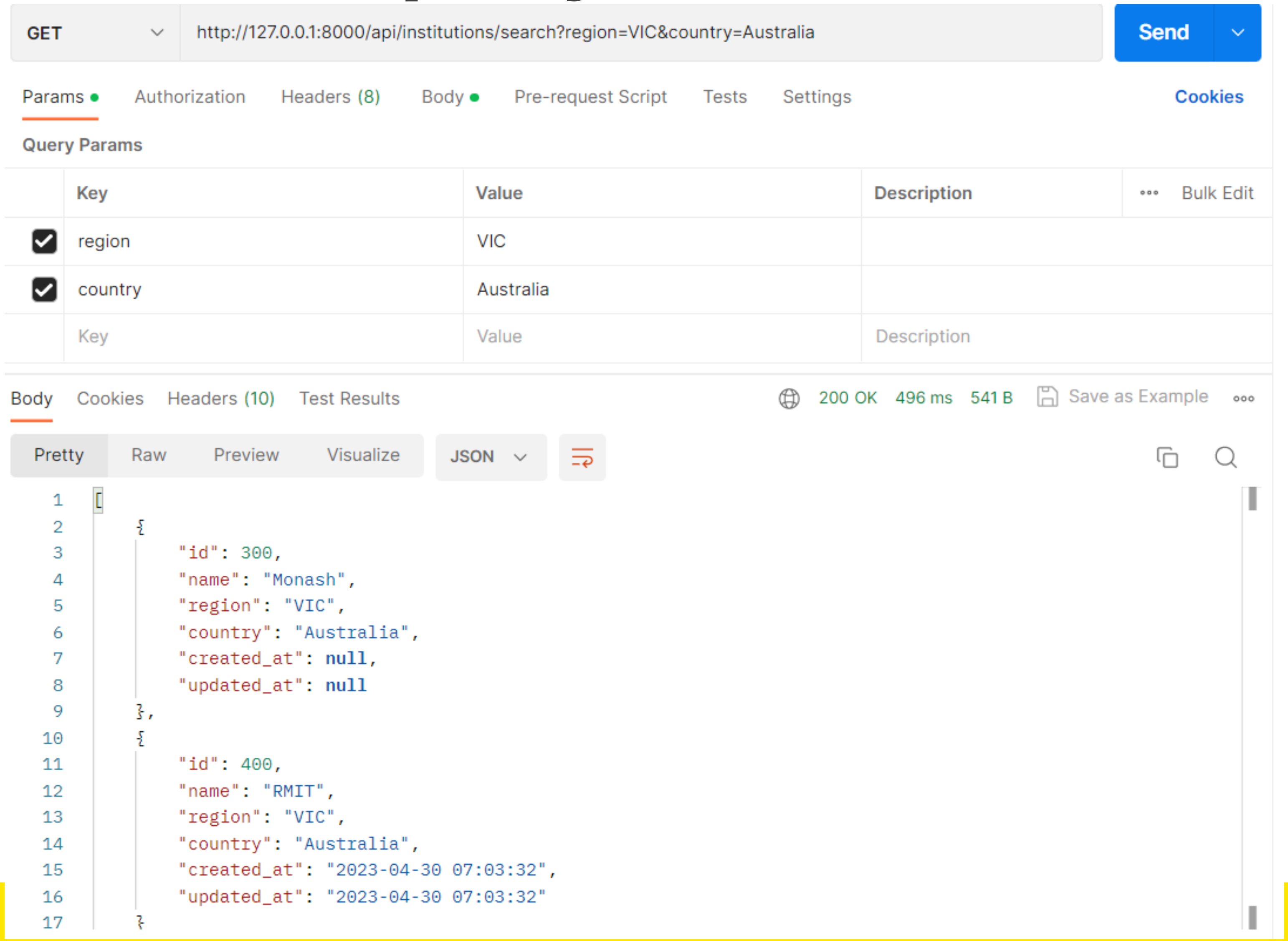
Using Queries (Cont.)

- Add the new search function to the route (api.php)
- Search must come before /{id}. Why?



```php
Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});
Route::group(['prefix' => 'institutions'], function () {
    Route::get('/', [InstitutionController::class, 'index']);
    Route::get('search', [InstitutionController::class, 'search']);
    Route::get('/{id}', [InstitutionController::class, 'show']);
    Route::post('/', [InstitutionController::class, 'store']);
    Route::put('/{id}', [InstitutionController::class, 'update']);
    Route::delete('/{id}', [InstitutionController::class, 'destroy']);
});
```

Test the query in Postman with Parameters



http://127.0.0.1:8000/api/institutions/search?region=VIC&country=Australia

Query Params

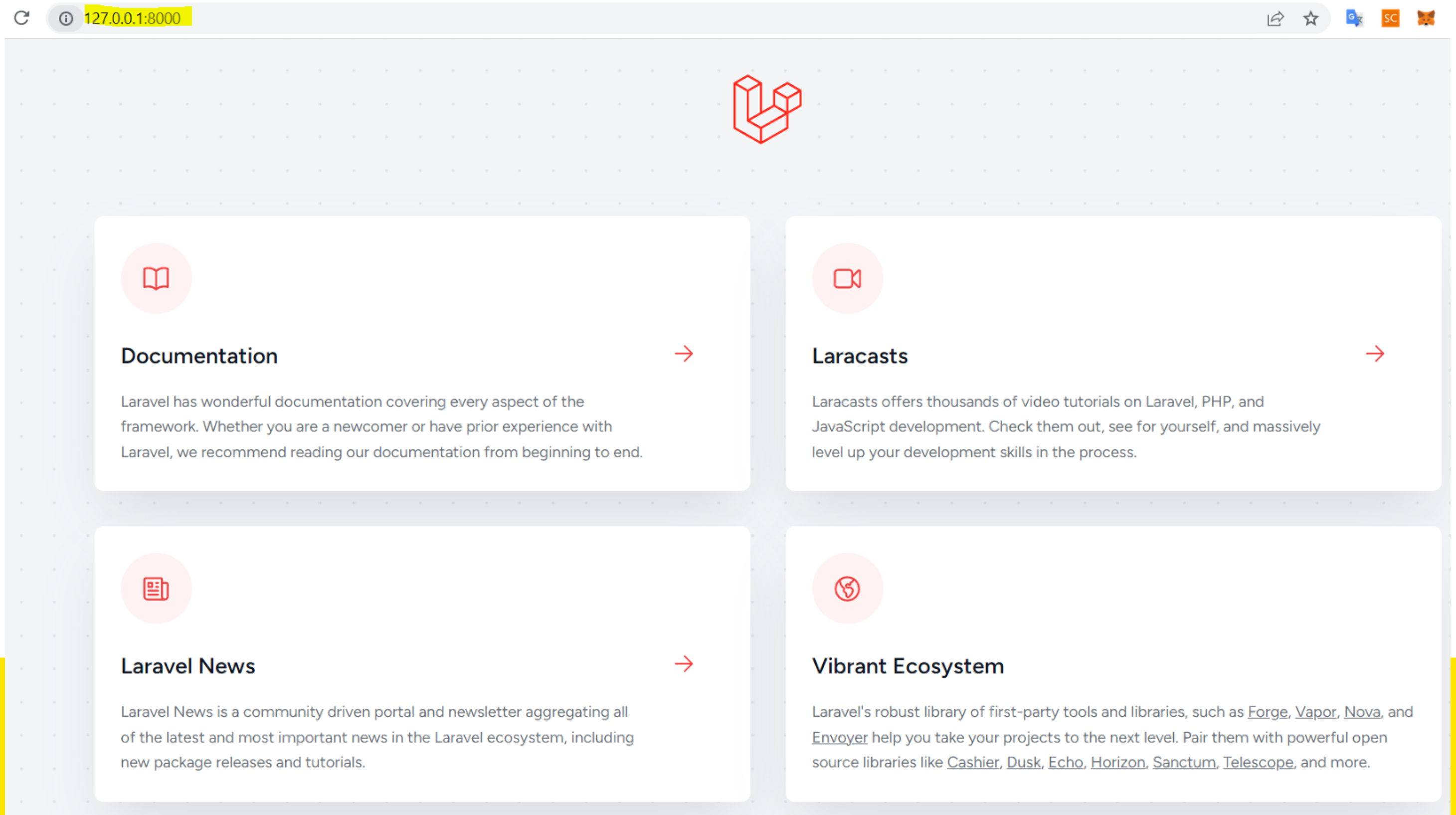
Key	Value	Description
☑ region	VIC	
☑ country	Australia	
Key	Value	Description

200 OK 496 ms 541 B

```
[
    {
        "id": 300,
        "name": "Monash",
        "region": "VIC",
        "country": "Australia",
        "created_at": null,
        "updated_at": null
    },
    {
        "id": 400,
        "name": "RMIT",
        "region": "VIC",
        "country": "Australia",
        "created_at": "2023-04-30 07:03:32",
        "updated_at": "2023-04-30 07:03:32"
    }
```

Laravel Views

- Views separate your controller / application logic from your presentation logic
- Views are stored in the **resources/views** directory
- There is one view in resources: welcome.blade.php


127.0.0.1:8000

Documentation

Laravel has wonderful documentation covering every aspect of the framework. Whether you are a newcomer or have prior experience with Laravel, we recommend reading our documentation from beginning to end.

Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.

Laravel News

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials.

Vibrant Ecosystem

Laravel's robust library of first-party tools and libraries, such as Forge, Vapor, Nova, and Envoyer help you take your projects to the next level. Pair them with powerful open source libraries like Cashier, Dusk, Echo, Horizon, Sanctum, Telescope, and more.

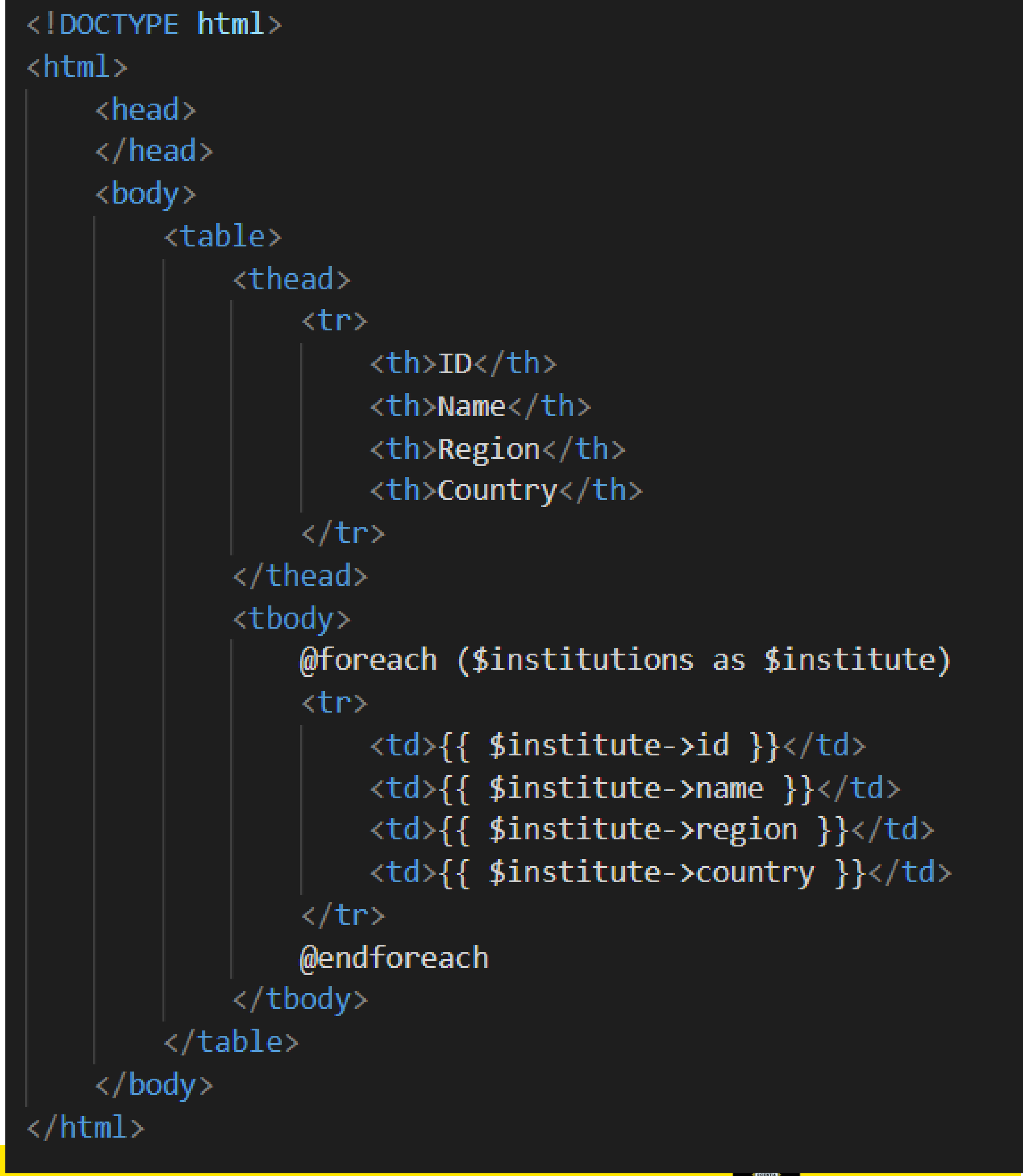
Adding a New View

- ➢ Create a file named **institutions.blade.php**

  in **views** folder
- ➢ Create a table to show all institutions data
- ➢ Create a controller:

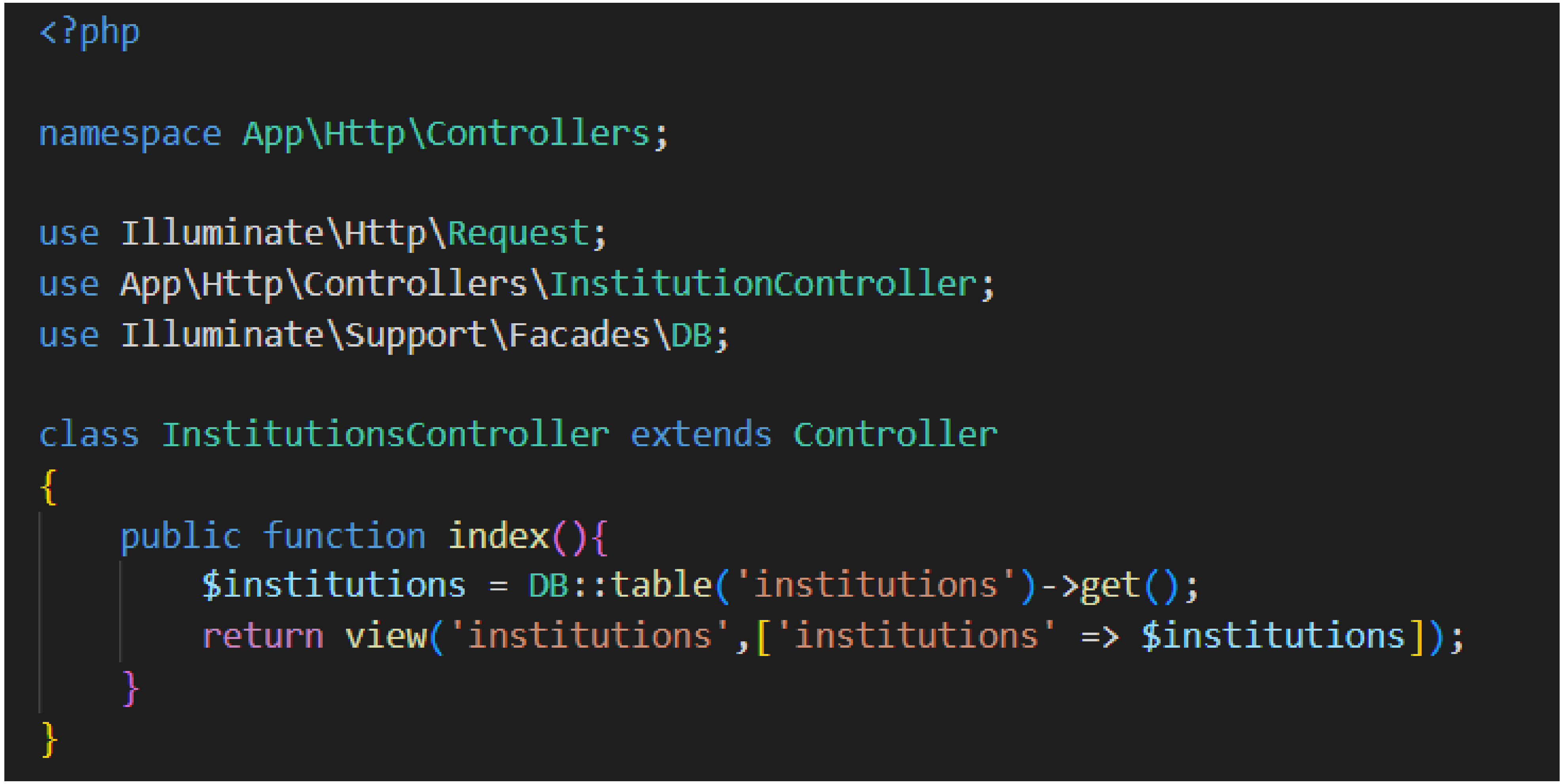
php artisan make:controller InstitutionsController

https://laravel.com/docs/10.x/views



```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <table>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>Region</th>
                    <th>Country</th>
                </tr>
            </thead>
            <tbody>
                @foreach ($institutions as $institute)
                <tr>
                    <td>{{ $institute->id }}</td>
                    <td>{{ $institute->name }}</td>
                    <td>{{ $institute->region }}</td>
                    <td>{{ $institute->country }}</td>
                </tr>
                @endforeach
            </tbody>
        </table>
    </body>
</html>
```

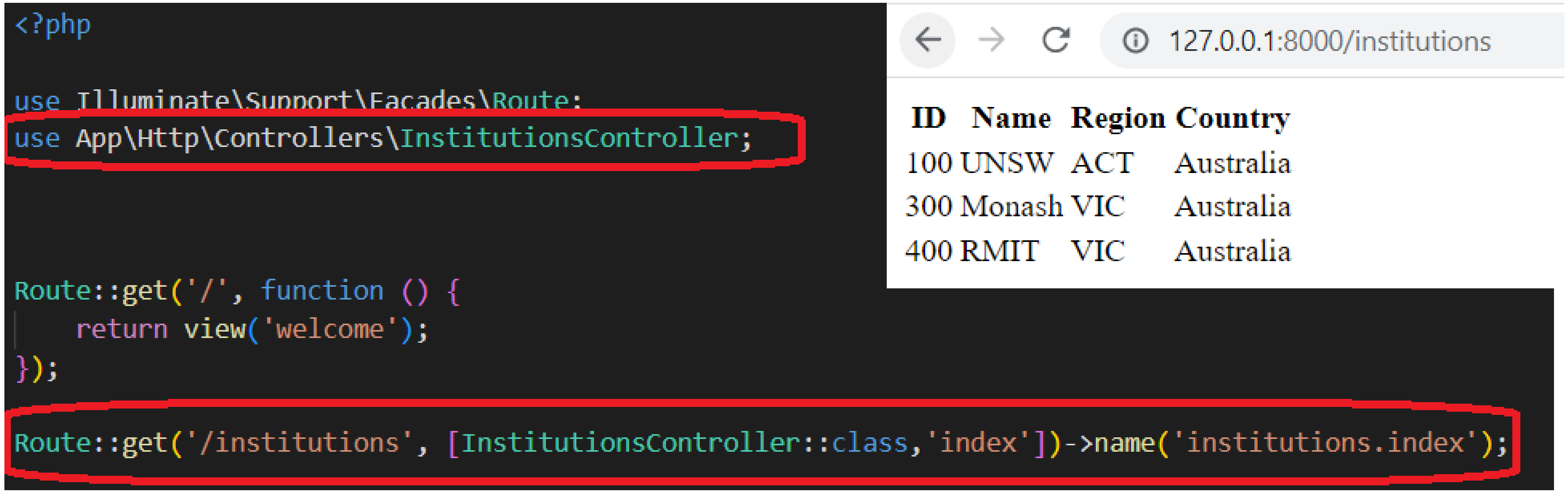
Adding a New View - Controller



```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Controllers\InstitutionController;
use Illuminate\Support\Facades\DB;

class InstitutionsController extends Controller
{
    public function index(){
        $institutions = DB::table('institutions')->get();
        return view('institutions',['institutions' => $institutions]);
    }
}
```

Adding a New View - Route

- Edit file **web.php** in **routes** folder:

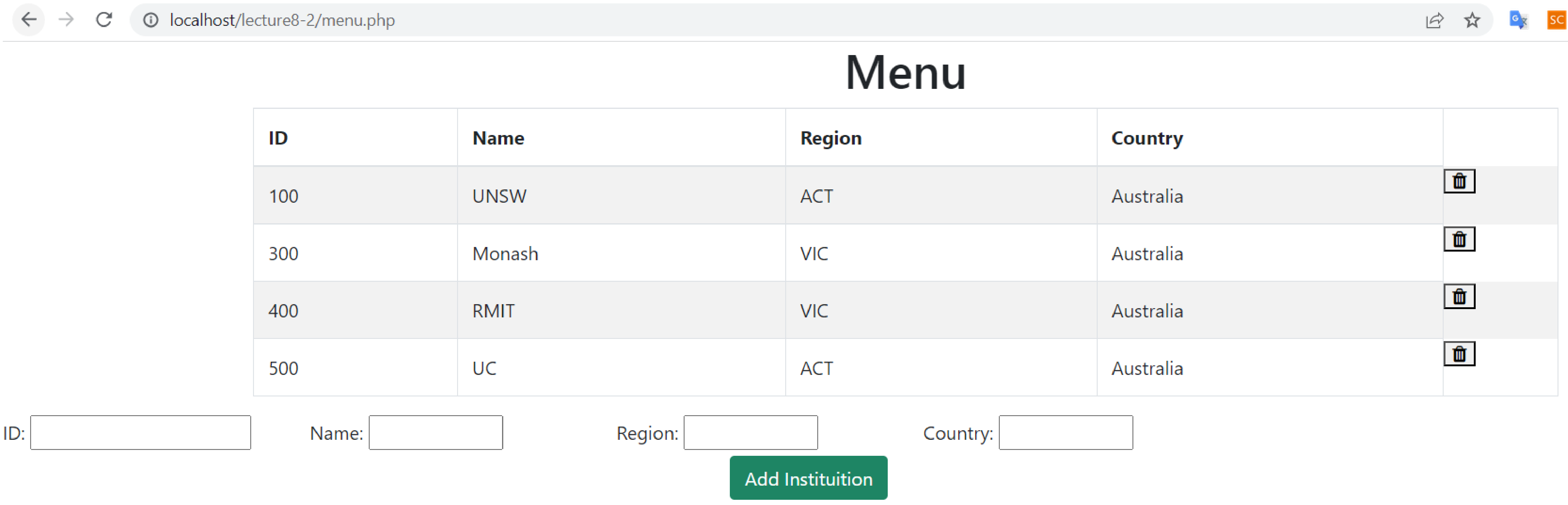


```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\InstitutionsController;


Route::get('/', function () {
    return view('welcome');
});

Route::get('/institutions', [InstitutionsController::class,'index'])->name('institutions.index');
```

127.0.0.1:8000/institutions

ID	Name	Region	Country
100	UNSW	ACT	Australia
300	Monash	VIC	Australia
400	RMIT	VIC	Australia

Using Laravel API in Other Applications

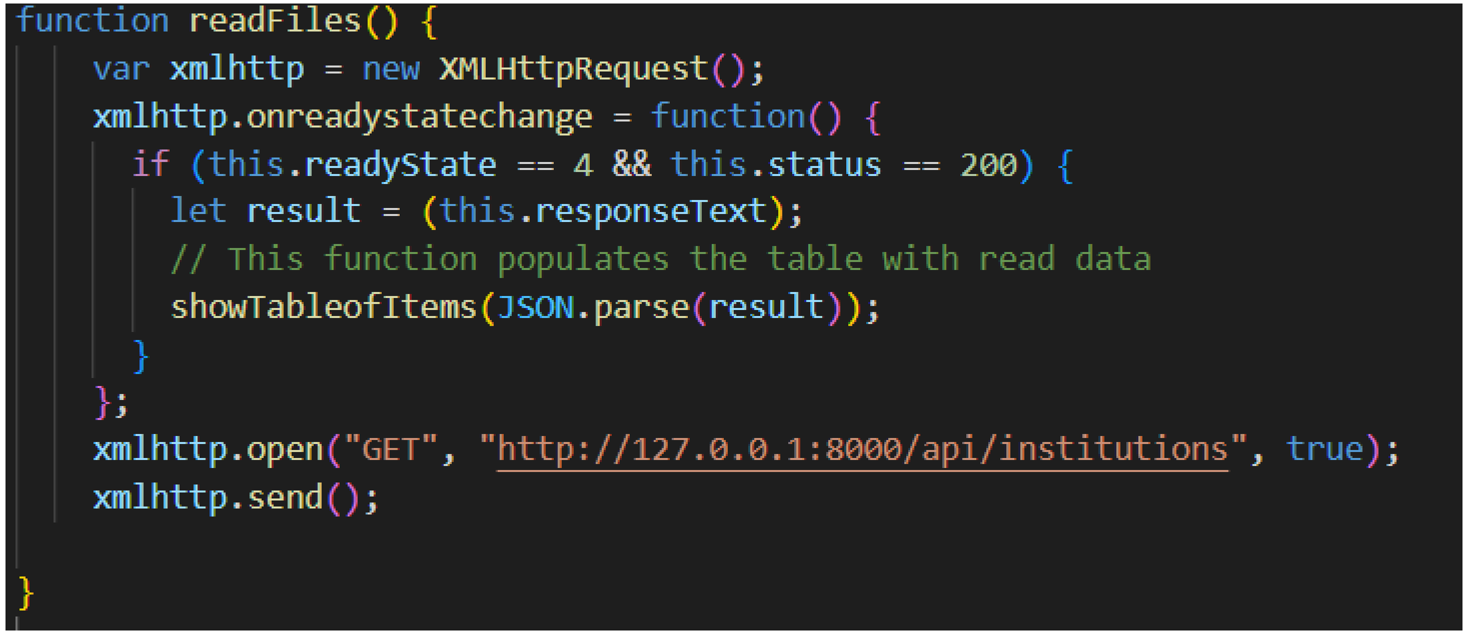

localhost/lecture8-2/menu.php

Menu

ID	Name	Region	Country	
100	UNSW	ACT	Australia	🗑
300	Monash	VIC	Australia	🗑
400	RMIT	VIC	Australia	🗑
500	UC	ACT	Australia	🗑

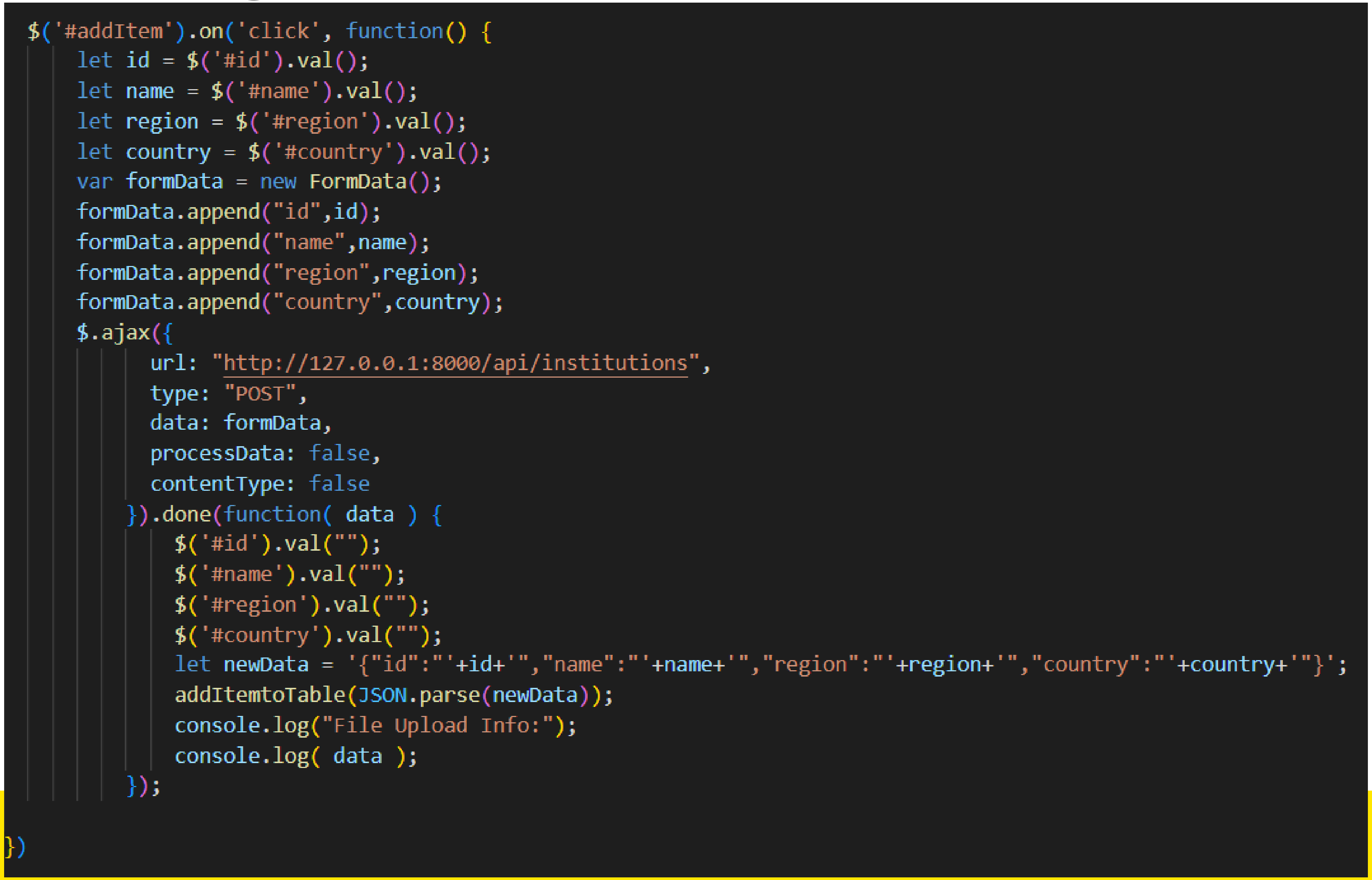
ID: ____ Name: ____ Region: ____ Country: ____

Reading Data



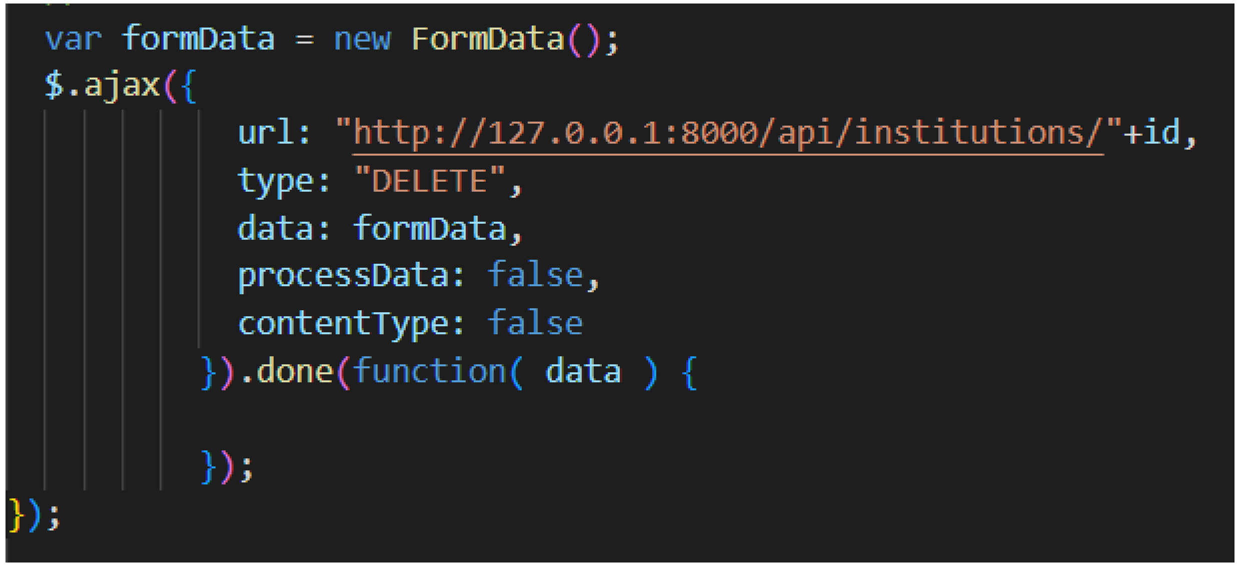
```javascript
function readFiles() {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        let result = (this.responseText);
        // This function populates the table with read data
        showTableofItems(JSON.parse(result));
      }
    };
    xmlhttp.open("GET", "http://127.0.0.1:8000/api/institutions", true);
    xmlhttp.send();

}
```

Adding Data



```
$('#addItem').on('click', function() {
    let id = $('#id').val();
    let name = $('#name').val();
    let region = $('#region').val();
    let country = $('#country').val();
    var formData = new FormData();
    formData.append("id",id);
    formData.append("name",name);
    formData.append("region",region);
    formData.append("country",country);
    $.ajax({
          url: "http://127.0.0.1:8000/api/institutions",
          type: "POST",
          data: formData,
          processData: false,
          contentType: false
        }).done(function( data ) {
            $('#id').val("");
            $('#name').val("");
            $('#region').val("");
            $('#country').val("");
            let newData = '{"id":"'+id+'","name":"'+name+'","region":"'+region+'","country":"'+country+'"}';
            addItemtoTable(JSON.parse(newData));
            console.log("File Upload Info:");
            console.log( data );
        });

})
```

Delete Data



```
    var formData = new FormData();
    $.ajax({
                url: "http://127.0.0.1:8000/api/institutions/"+id,
                type: "DELETE",
                data: formData,
                processData: false,
                contentType: false
            }).done(function( data ) {

            });
});
```

Seeding

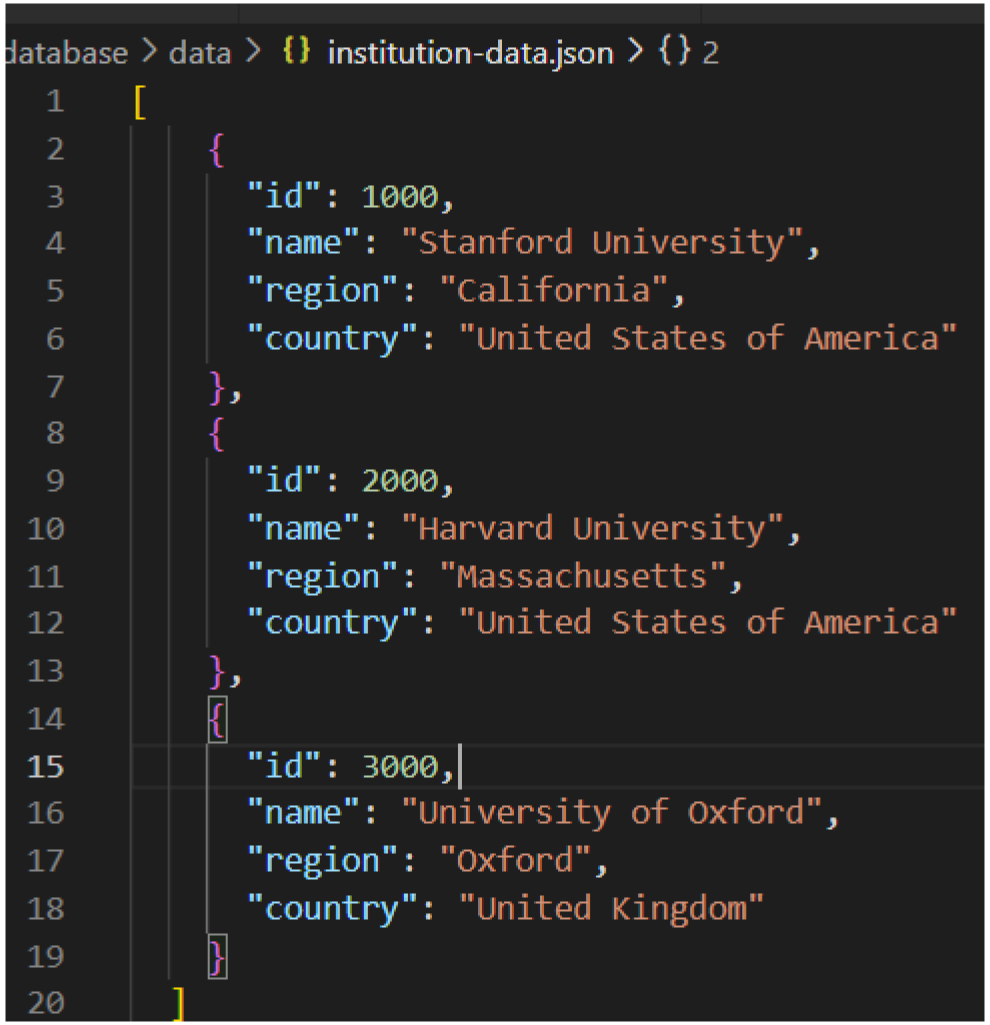
- Laravel includes the ability to seed your database with data using seed classes.
- All seed classes are stored in the **database/seeders** directory.
- By default, a DatabaseSeeder class is defined for you. From this class, you may use the call method to run other seed classes, allowing you to control the seeding order.

https://laravel.com/docs/10.x/seeding

Seeding Institution

- In the database directory, create a new directory called **data**.
- Create a JSON file - **institution-data.json** into the **database\data** directory.
- Create a Seeder class which will seed the institutions tables appropriate JSON file. To do this, run the following commands:

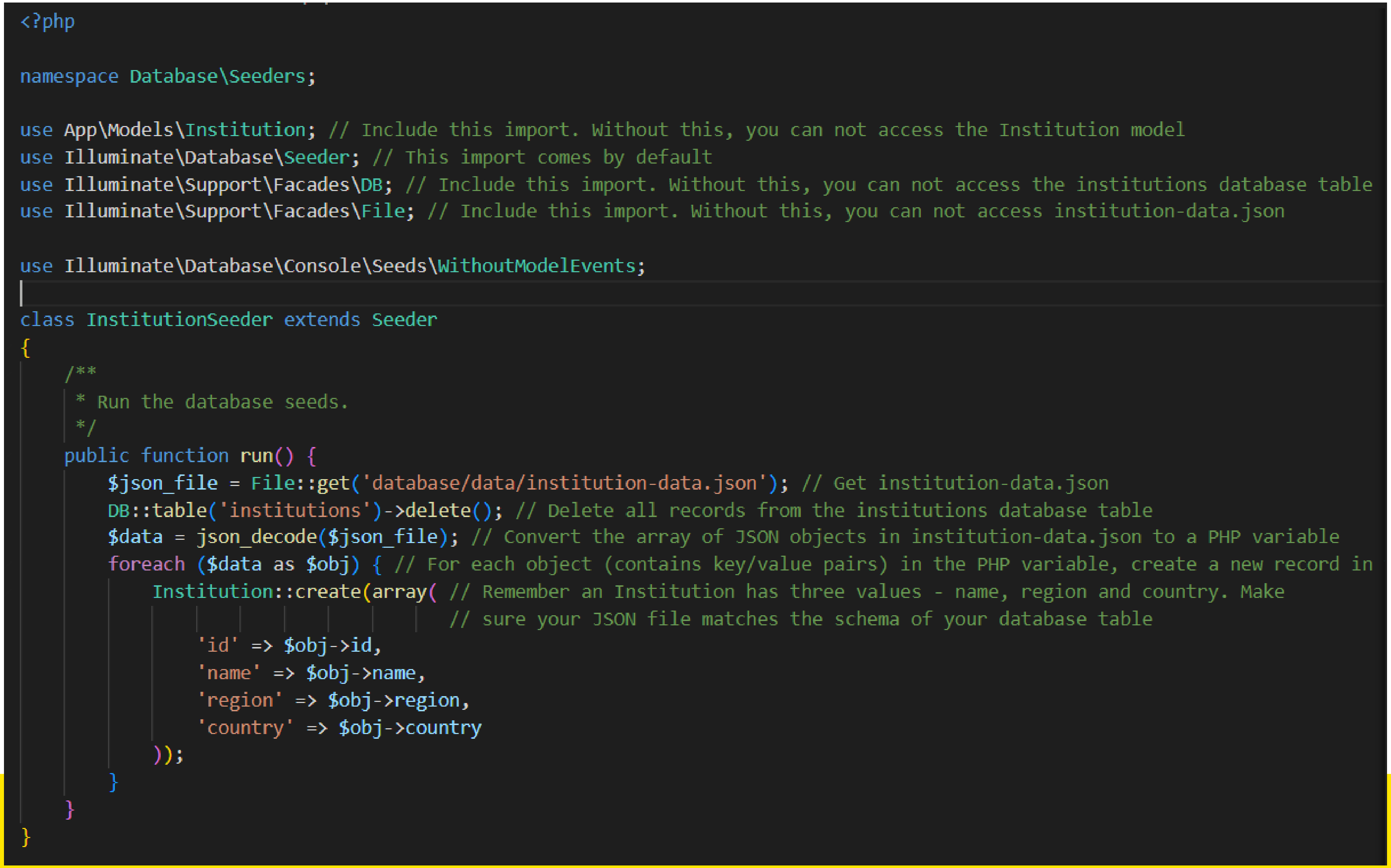
  php artisan make:seeder InstitutionSeeder



database > data > {} institution-data.json > {} 2

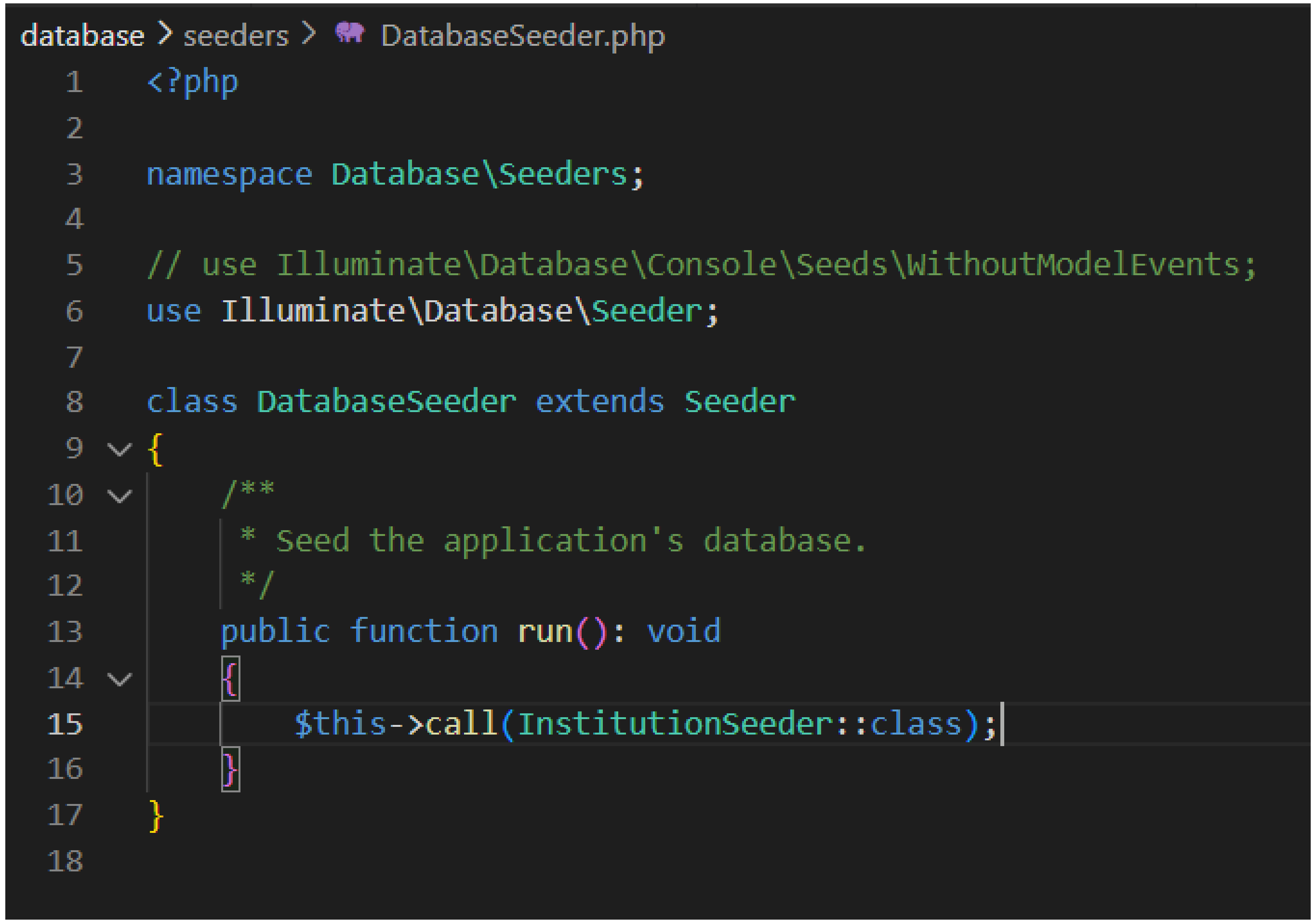
```json
[
    {
        "id": 1000,
        "name": "Stanford University",
        "region": "California",
        "country": "United States of America"
    },
    {
        "id": 2000,
        "name": "Harvard University",
        "region": "Massachusetts",
        "country": "United States of America"
    },
    {
        "id": 3000,
        "name": "University of Oxford",
        "region": "Oxford",
        "country": "United Kingdom"
    }
]
```

InstitutionSeeder Class



```php
<?php

namespace Database\Seeders;

use App\Models\Institution; // Include this import. Without this, you can not access the Institution model
use Illuminate\Database\Seeder; // This import comes by default
use Illuminate\Support\Facades\DB; // Include this import. Without this, you can not access the institutions database table
use Illuminate\Support\Facades\File; // Include this import. Without this, you can not access institution-data.json

use Illuminate\Database\Console\Seeds\WithoutModelEvents;

class InstitutionSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run() {
        $json_file = File::get('database/data/institution-data.json'); // Get institution-data.json
        DB::table('institutions')->delete(); // Delete all records from the institutions database table
        $data = json_decode($json_file); // Convert the array of JSON objects in institution-data.json to a PHP variable
        foreach ($data as $obj) { // For each object (contains key/value pairs) in the PHP variable, create a new record in
            Institution::create(array( // Remember an Institution has three values - name, region and country. Make
                                       // sure your JSON file matches the schema of your database table
                'id' => $obj->id,
                'name' => $obj->name,
                'region' => $obj->region,
                'country' => $obj->country
            ));
        }
    }
}
```

Database Seeder

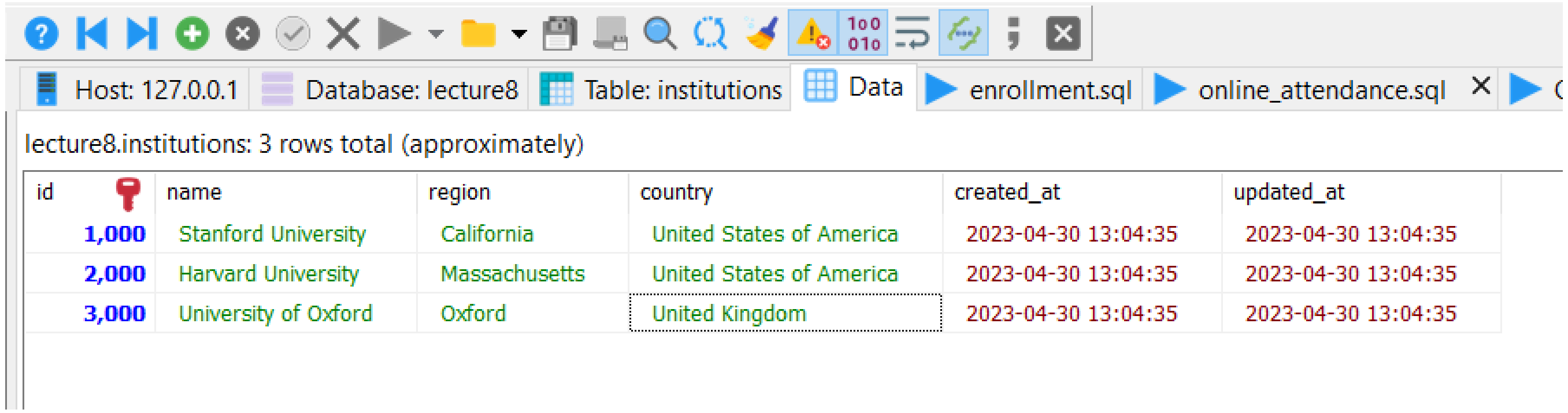


database > seeders > DatabaseSeeder.php

```php
<?php

namespace Database\Seeders;

// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        $this->call(InstitutionSeeder::class);
    }
}

```

Run the Seeder

php artisan db:seed



lecture8.institutions: 3 rows total (approximately)

id	name	region	country	created_at	updated_at
1,000	Stanford University	California	United States of America	2023-04-30 13:04:35	2023-04-30 13:04:35
2,000	Harvard University	Massachusetts	United States of America	2023-04-30 13:04:35	2023-04-30 13:04:35
3,000	University of Oxford	Oxford	United Kingdom	2023-04-30 13:04:35	2023-04-30 13:04:35

Final Note

- **Please do not forget Quiz 2 is this week:**
    - Quiz will be during your scheduled lab time.
    - Quiz is closed book.
    - Quiz duration is 40 minutes.
    - Quiz will cover material discussed in weeks 4, 5, and 6.
    - Monday lab: 16:20
    - Wednesday lab: 15:20