# Exercise (Laravel basics)

For those who cannot setup Docker on your machine, refer to the following steps to setup

+ Install Composer on your machine, refer to installation guideline here, click Next, Next, Next and Next, …

<https://getcomposer.org/download/>

+ After installing composer, open Cmd/Powershell on Windows and test whether Composer has been setup

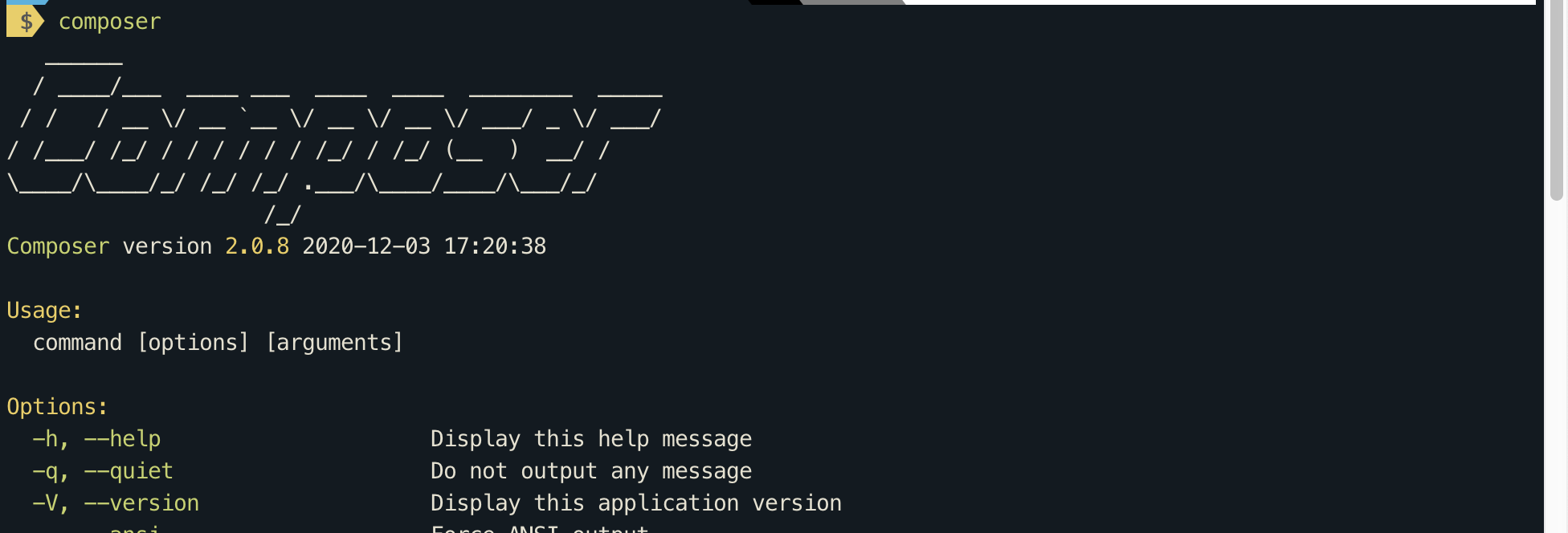
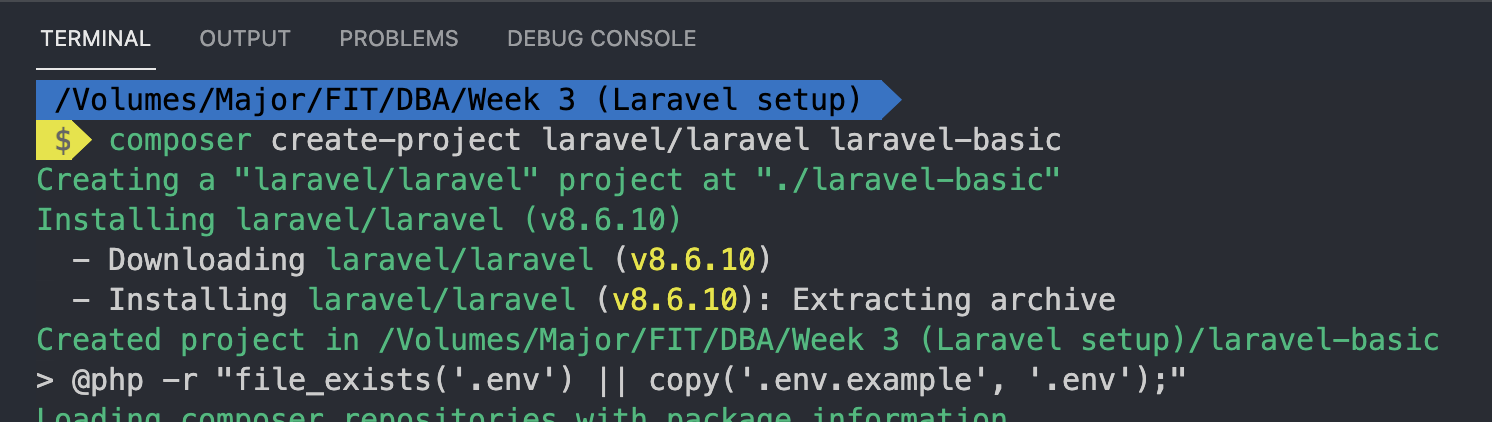


Figure 1: Composer setup successfully.

+ Next, install Laravel at C:\xampp\htdocs folder, following this command



Rename **Laravel-basic** to the name that you choose for this tutorial.

+ Install NodeJS and choose your download here <https://nodejs.org/en/download/>

After completing Laravel installation, if you can see Laravel is presented on screen by pointing to XAMPP and your Laravel folder, run php artisan serve, then move on.

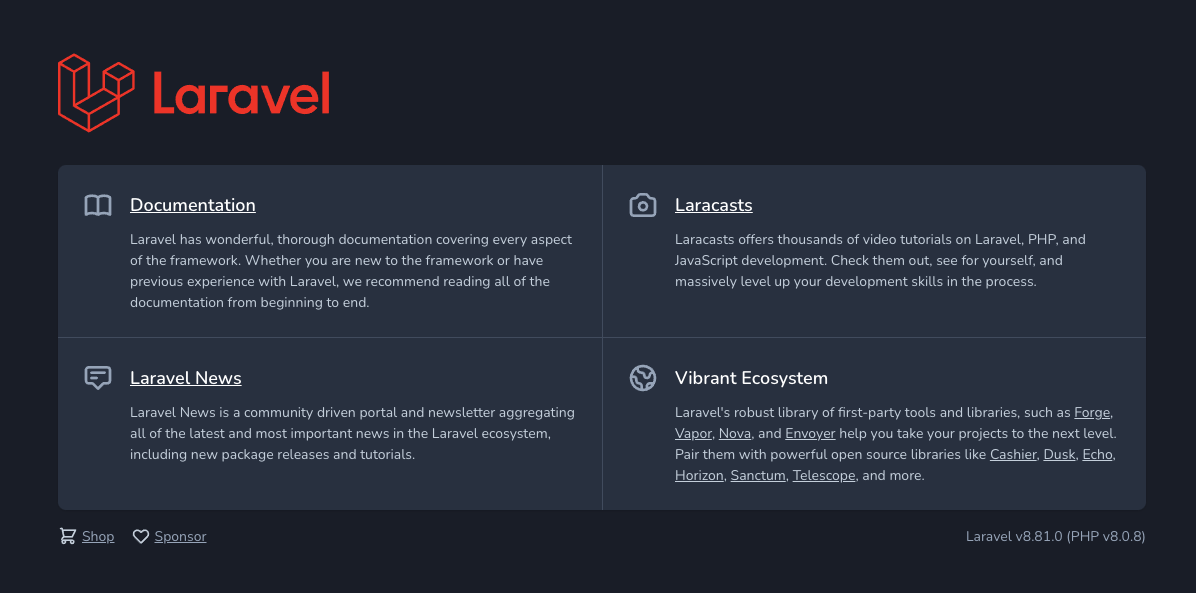


Figure : Laravel is booted up successfully

Change .env file to this

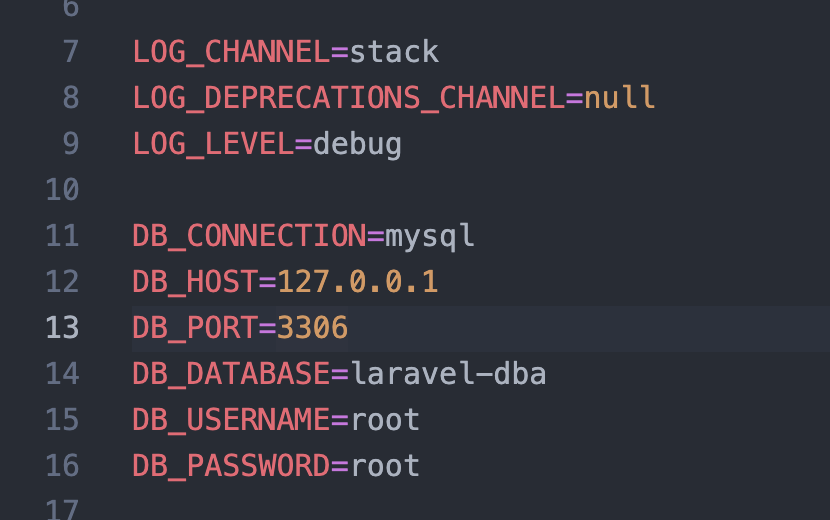


Figure :.env Laravel

Includes DB\_HOST = 127.0.0.1

If using XAmpp, change DB\_PASSWORD to (empty)

## Working with migrations

In Laravel, everything is done via migrations. So for simplicity, we will use existing scaffolding from Laravel to build UI.

Install Laravel Breeze in Terminal

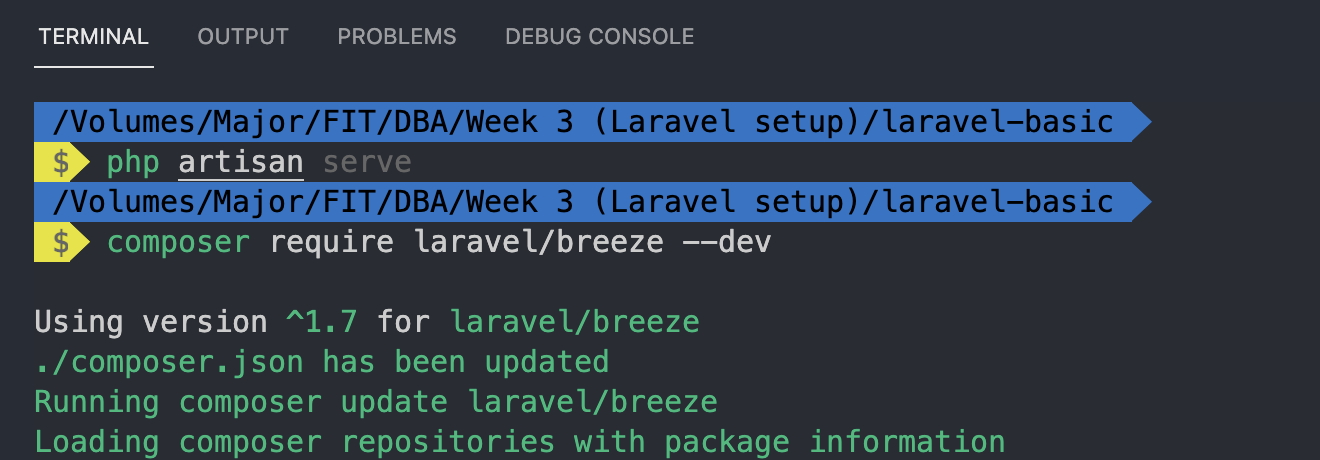
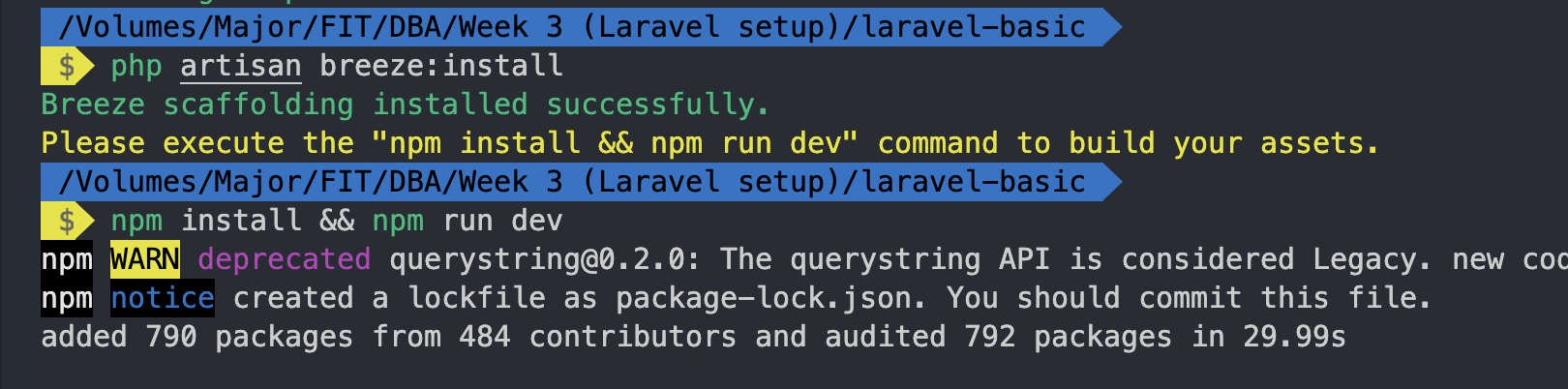


Figure : Laravel Breeze installation

Install Laravel breeze and run npm



Run php artisan serve again

If you see the following image, then you are done

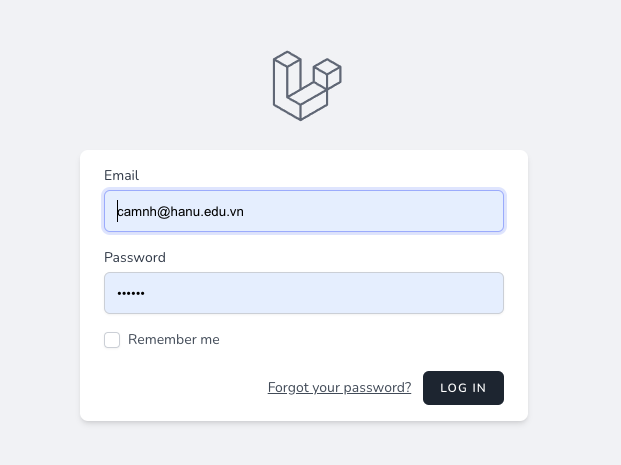


Figure : Laravel complete setup

If PHPMyadmin is available on your machine, use that.

Connection settings may be varied within machines.

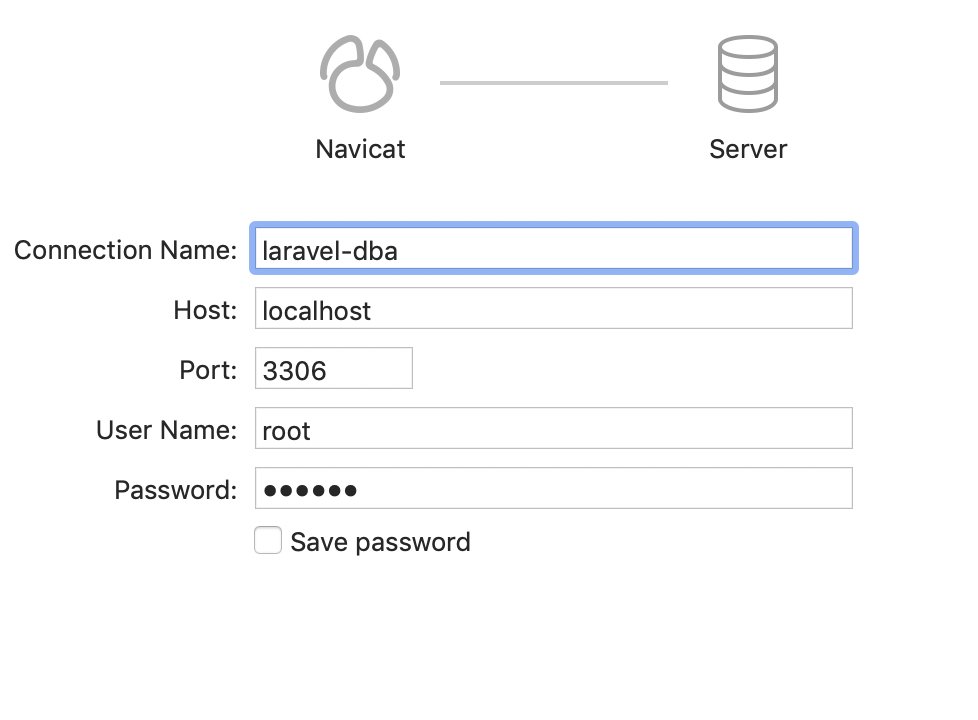


Figure : Laravel-dba connect parameters, using Navicat Premium

Go to php container and run the following command to work with database in Laravel

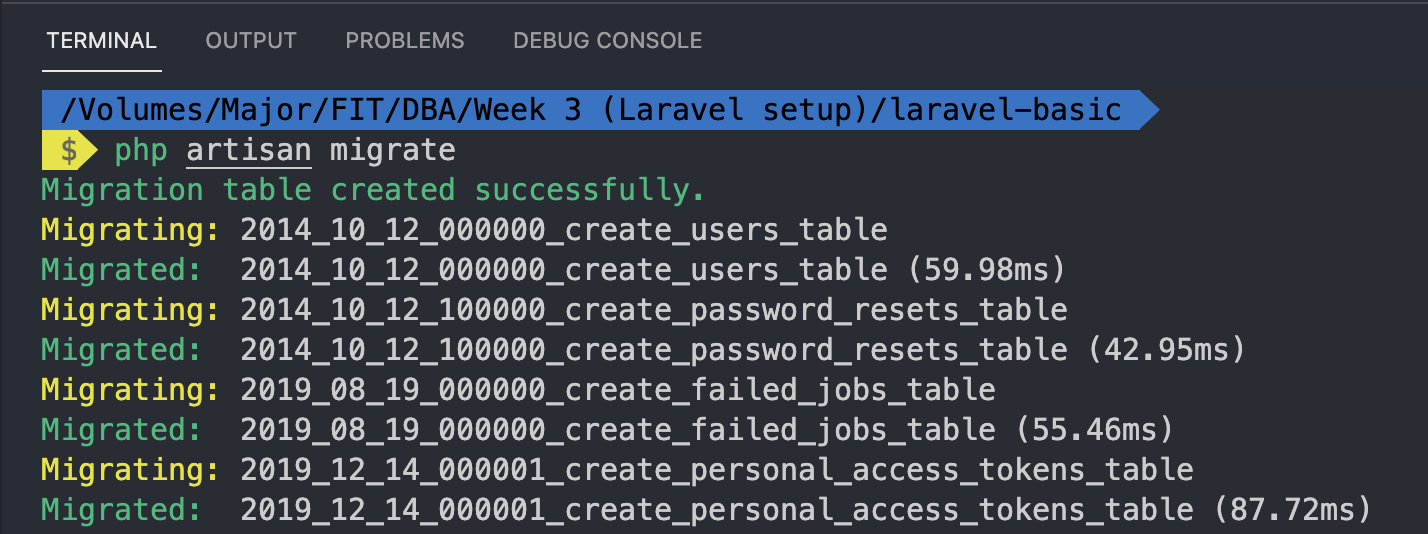


Figure : Running migration to connect to database

If you cannot make migrations, go to database and DELETE ALL TABLES exists in database. That’s all.

## Working with CRUD

At this step, you are required to use Laravel to perform basic CRUD operations. We will use problem from DBS called “Student management system”

Inside “Student management system”, we have table students, courses and enrolments

Student table, for simplicity has the following attributes

* id (PK)
* name (varchar)
* dob (date)
* phone\_number (varchar)

First, we will create migrations to add a new student: called create\_students\_table.

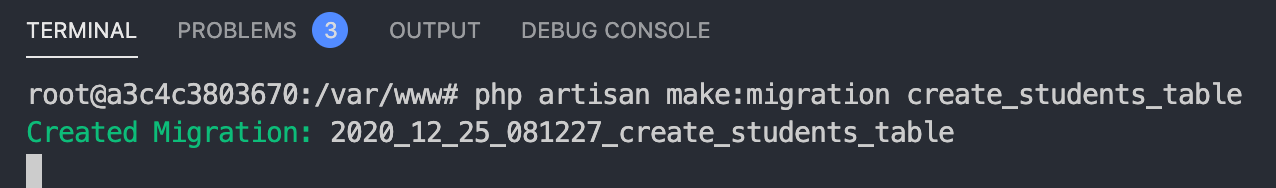


Figure : Create student migration table

Inside database/migrations folder, at last file, add the following detail

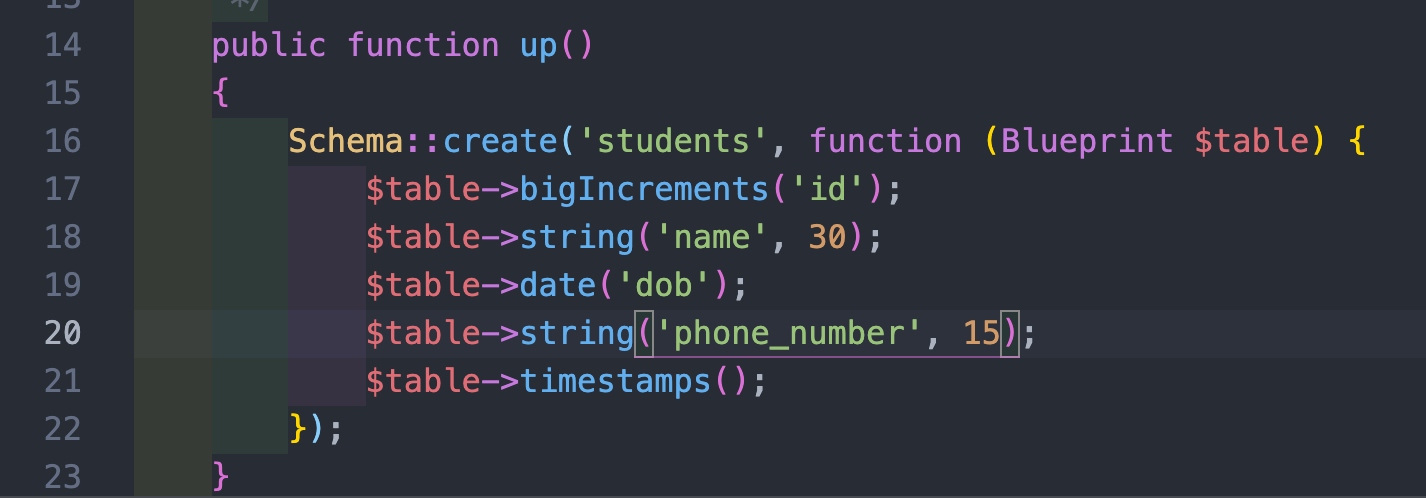


Figure :Add students table using migration

After writing migrations, run the following to migrate to database php artisan migrate inside docker containers

## Create UI for “Retrieve | Read” command.

You need to authenticate yourself on this application by going to /register like this

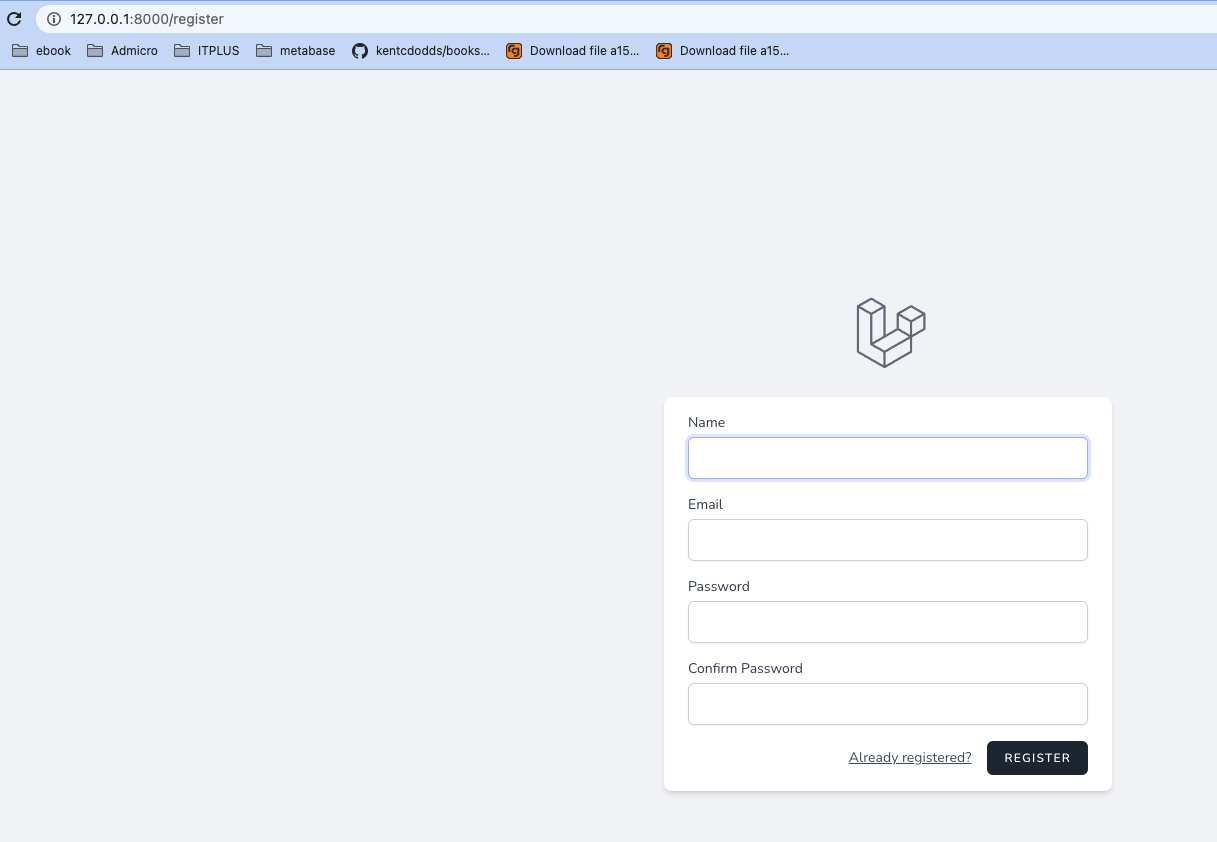


Figure : Register account on Laravel

First of all, we write rules in routes/web.php file. Remember, everything in Laravel must go from this file first.

Running the following command to create controller for Student. Resource controller means it will create CRUD function without body automatically.

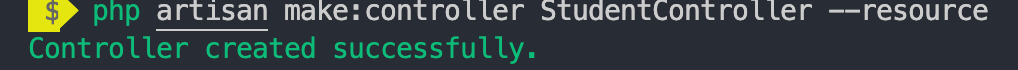


Figure 11: Create StudentController as Resource controller

Create student resource route



Figure : Adding route for Laravel – line 24.

Create views for all CRUD commands for Student: Creating a new folder in resources/views

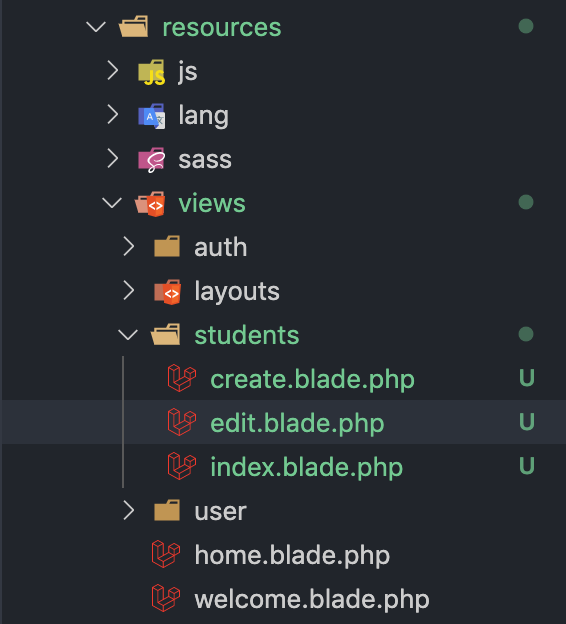


Figure : Create folder students inside views folder

Lastly, we create model Student to interact with database: Running php artisan make:model Student. Model file is inserted in App\Models folder

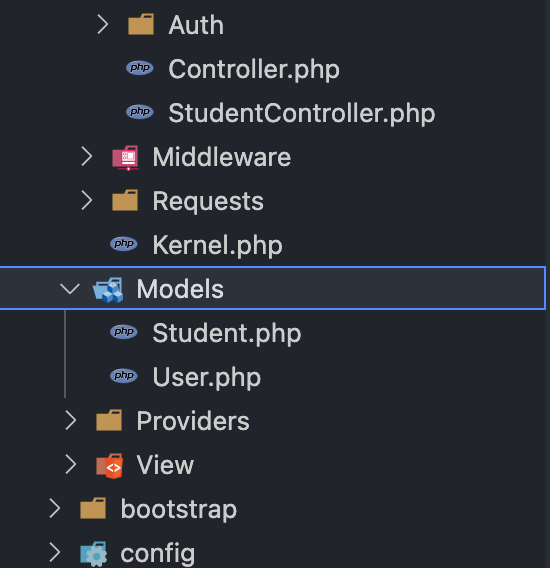


Figure : Make model Student

Push data to view by calling model Student, retrieve all Student and push those data to views.

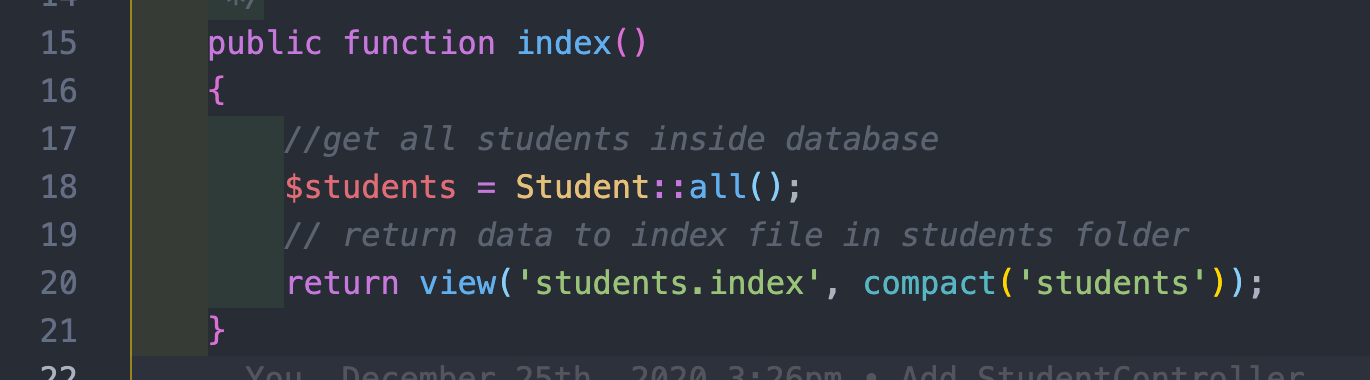


Figure : Get all students inside database and return those data to view

For detail of the file view students.index, navigate to <https://gitlab.com/laravel100/tutorial-3/-/blob/master/resources/views/students/index.blade.php>

Layout file is changed, so you need to copy my edited layout file at resources/views/layouts/app.blade.php to work.

Note that @extends is used to specify which layout will be used. Layout is common skeleton (header, sidebar, footer) is used throughout view.



Figure 16: Fragment of view file

## Create UI and logic for “Create”

UI for create is https://gitlab.com/laravel100/tutorial-3/-/blob/master/resources/views/students/create.blade.php

Add $guarded property to model to allow every attribute can be pushed into databases

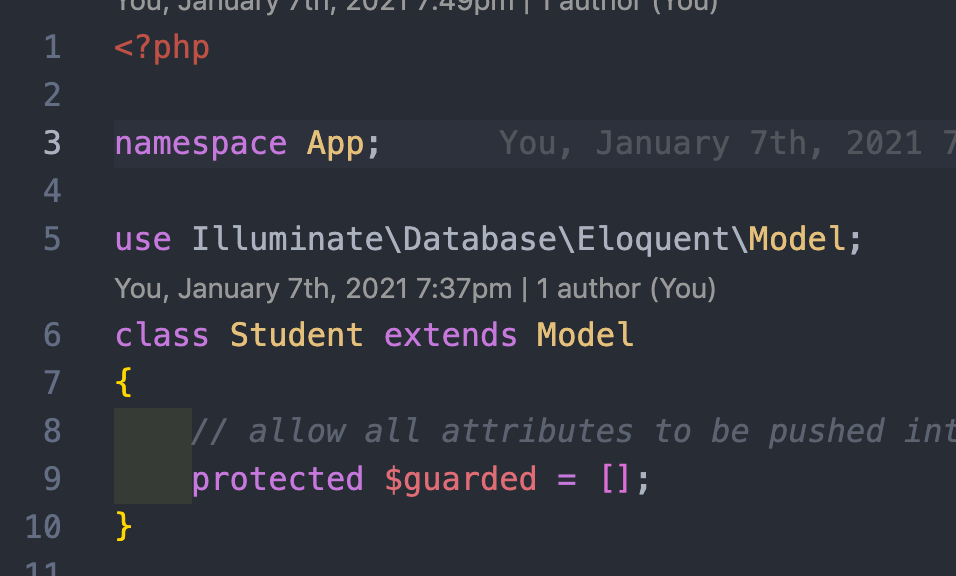


Figure 17: Add $guarded property to model

Next is the logic to create new student.

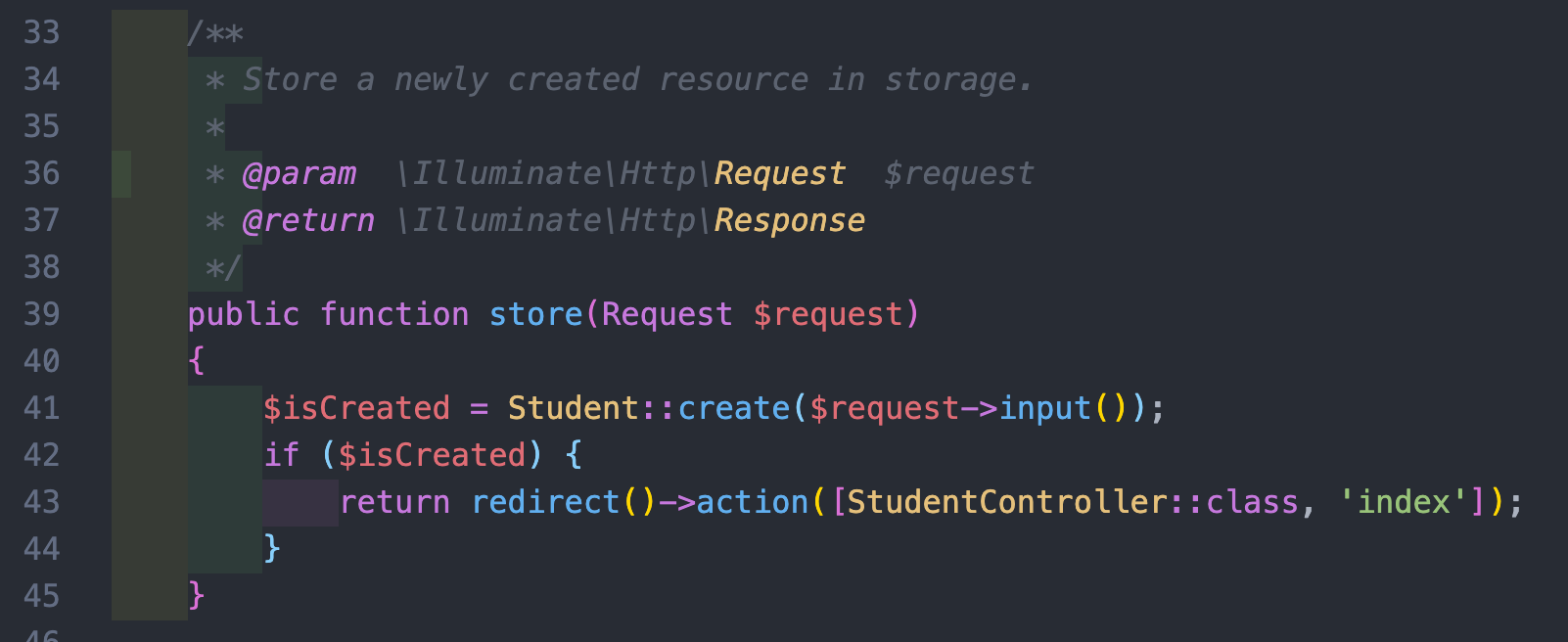


Figure 18: Create new Student logic

## Create UI and logic for “Update”

UI of “Update” is specified here <https://gitlab.com/laravel100/tutorial-3/-/blob/master/resources/views/students/edit.blade.php>

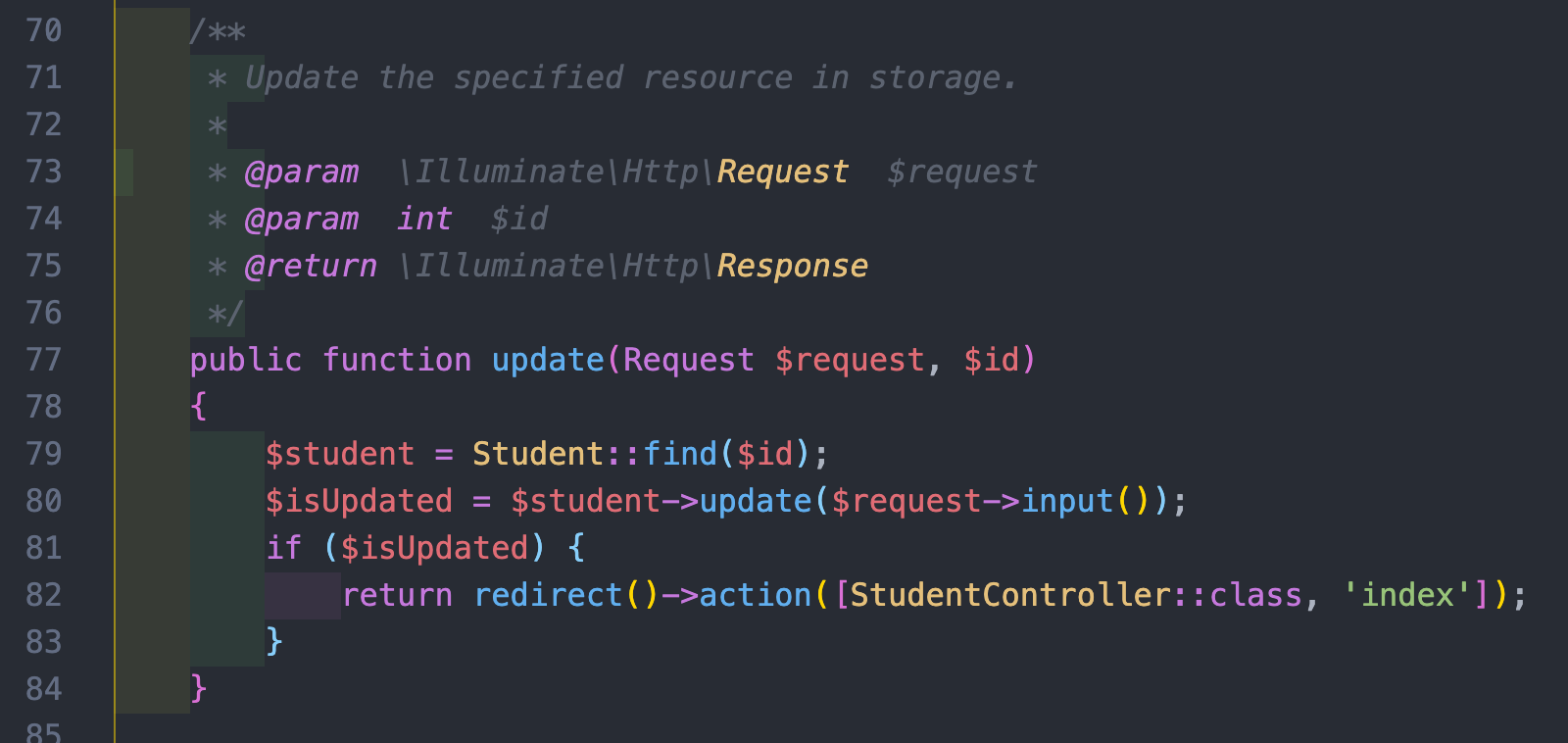


Figure 19: Update in controller

* 1. Logic for Delete

First of all, we find student using $id parameter, then redirect to index() function

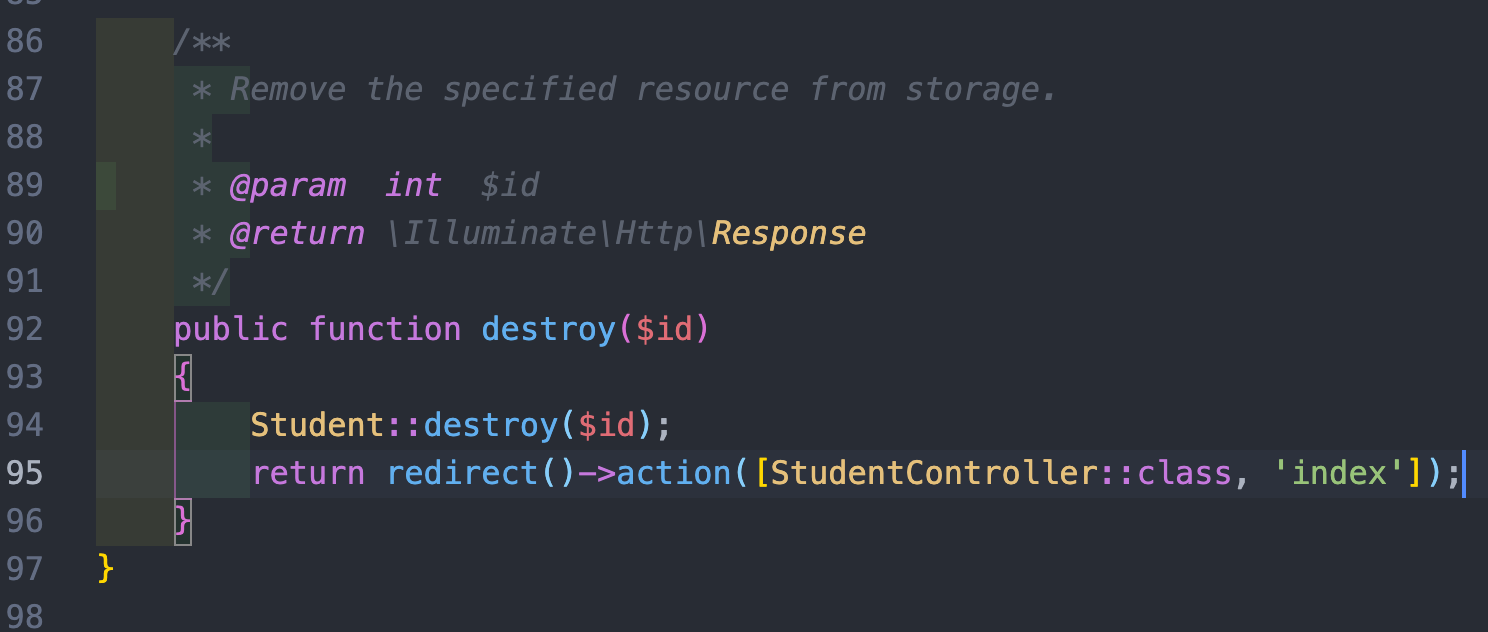


Figure 20: Delete in controller

Full source code for tutorial is here: <https://gitlab.com/laravel100/tutorial-3>

## Exercises

Based on source code before, implement a full CRUD of Course (includes creating migrations, models, views, controllers in MVC). Course attributes include

* Course code
* Course name
* Start date (Assume that course name is unique)
* End date