

CHƯƠNG 8: ELOQUENT ORM

1. Lý thuyết Cốt lõi (Khái niệm)

- ORM là gì? (Object-Relational Mapping).

Đây là một kỹ thuật "ánh xạ" (mapping) một bảng (table) trong CSDL (ví dụ: bảng users) thành một "Đối tượng" (Object) trong code (ví dụ: class User của PHP).

+ Eloquent (8.1): Là ORM của Laravel.

o Model: Mỗi bảng trong CSDL của bạn sẽ có một file Model tương ứng trong thư mục app/Models/. Ví dụ: Bảng sinhvien \rightarrow Model app/Models/SinhVien.php.

o Quy ước (Convention): Eloquent rất thông minh. Nếu bạn đặt tên Model là SinhVien (số ít, viết hoa chữ cái đầu), nó sẽ tự động hiểu rằng Model này làm việc với bảng sinhvien (số nhiều, viết thường).

+ Migrations (8.2): Là "hệ thống quản lý phiên bản" cho CSDL của bạn. Thay vì tạo bảng bằng tay trong phpMyAdmin, bạn sẽ viết code PHP (trong thư mục database/migrations/) để định nghĩa cấu trúc bảng.

o Lệnh tạo: `php artisan make:model TenModel -m` (cờ -m sẽ tạo luôn file migration).

o Lệnh chạy: `php artisan migrate` (để tạo bảng trong CSDL).

+ Truy vấn (8.3, 8.4): Đây là điều tuyệt vời nhất.

o Thay vì PDO (Chương 4): `$sql = "SELECT * FROM sinhvien";`

o Dùng Eloquent (Chương 8): `$all_students = SinhVien::all();`

o Thay vì INSERT (Chương 4): `$sql = "INSERT INTO sinhvien (ten, email) VALUES (?, ?)";`

o Dùng Eloquent (Chương 8): `SinhVien::create(['ten_sinh_vien' => $ten, 'email' => $email]);`

2. Nhiệm vụ Thực hành (BẮT BUỘC)

Kịch bản: Chúng ta sẽ "nâng cấp" PHT Quản lý sinh viên (từ Chương 4 & 5) lên chuẩn Laravel 100%, sử dụng Eloquent Model và Migrations.

A. Thiết lập Database & Migration (8.2)

3. Yêu cầu Bằng chứng (Proof of Work)

Bạn phải nộp lại 3 bằng chứng sau:

A. Code đã hoàn thiện:

1. Dán (paste) code hàm up() trong file Migration (..._create_sinh_vien_table.php).

```

public function up(): void
{
    Schema::create('sinh_viens', function (Blueprint $table) {
        $table->id();
        $table->string('ten_sinh_vien', 255);
        $table->string('email', 255)->unique();
        $table->timestamps();
    });
}

```

2. Dán (paste) toàn bộ code file app/Models/SinhVien.php.

```

<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class SinhVien extends Model
{
    use HasFactory;
    // TODO 8: Thêm mảng $fillable
    protected $fillable = [
        'ten_sinh_vien',
        'email',
    ];
}

```

3. Dán (paste) toàn bộ code file app/Http/Controllers/SinhVienController.php.

```

<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\SinhVien;

class SinhVienController extends Controller
{
    public function index()
    {
        return view('homepage', [
            'danhSachSV' => []
        ]);
    }
}

```

```
public function store(Request $request)
{
    $data = $request->all();
    SinhVien::create($data);
    return redirect()->route('sinhvien.index');
}
```

4. Dán (paste) code 2 route trong routes/web.php.

```
<?php

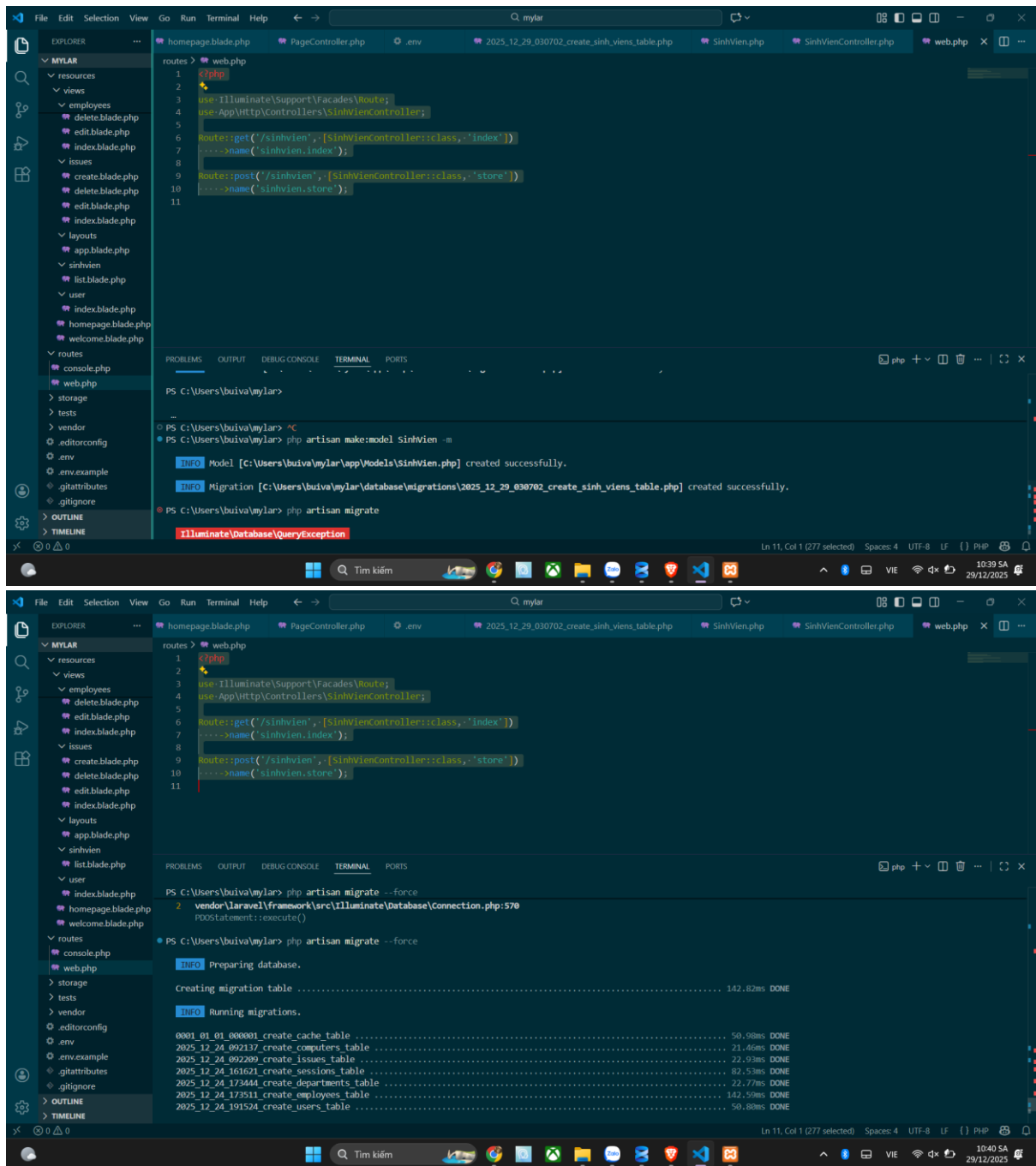
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\SinhVienController;

Route::get('/sinhvien', [SinhVienController::class, 'index'])
    ->name('sinhvien.index');

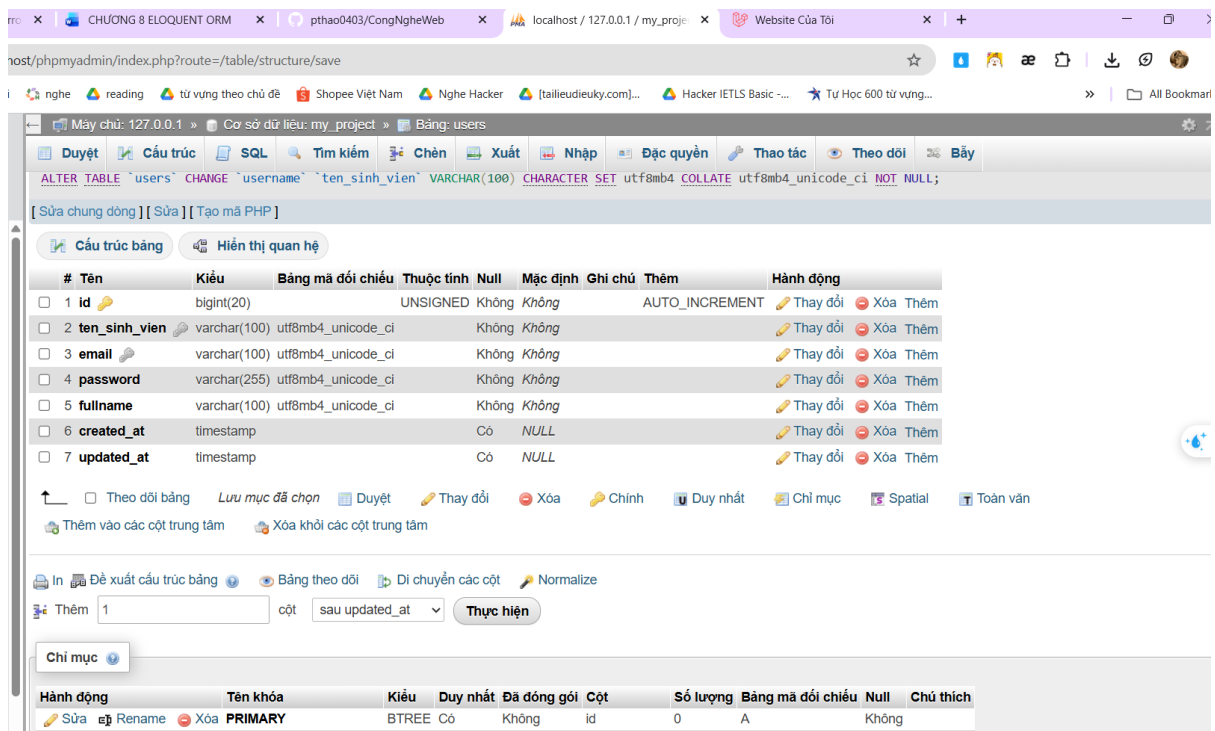
Route::post('/sinhvien', [SinhVienController::class, 'store'])
    ->name('sinhvien.store');
```

B. Ảnh chụp màn hình Kết quả (3 ẢNH):

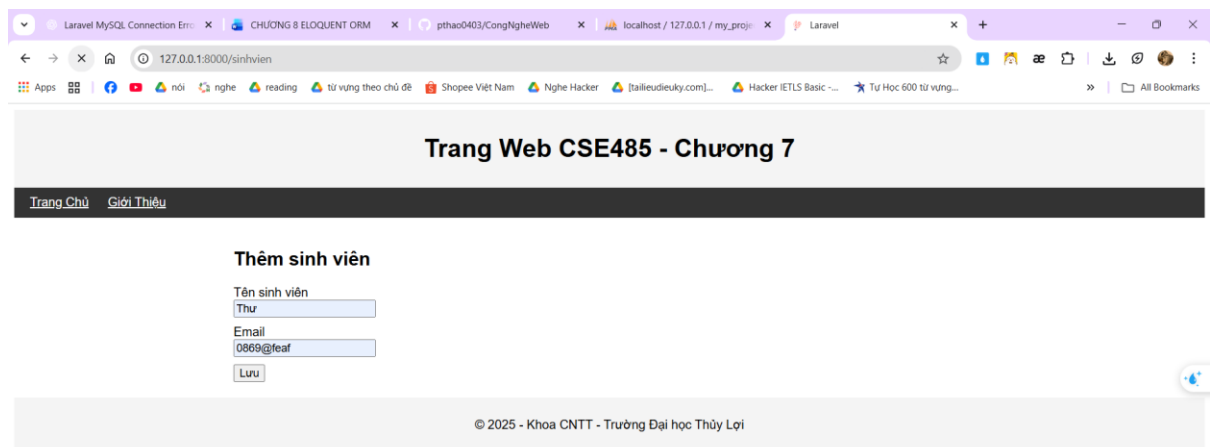
1. Ảnh 1 (Terminal): Chụp màn hình Terminal sau khi chạy php artisan make:model ... -m VÀ php artisan migrate thành công.



2. Ảnh 2 (phpMyAdmin): Chụp màn hình tab "Structure" (Cấu trúc) của bảng `sinh_viens` trong CSDL, cho thấy rõ các cột `id`, `ten_sinh_vien`, `email`, `created_at`.



3. Ảnh 3 (Trình duyệt Web): Chụp ảnh màn hình trang /sinhvien, sau khi đã dùng form thêm 1-2 sinh viên (chứng minh ::create() và ::all() đều hoạt động). \



(Dán Code A1, A2, A3, A4 và 3 Ảnh B1, B2, B3 của bạn vào đây)

4. Câu hỏi Phản biện (Bắt buộc)

Sau khi hoàn thành Phần 2 & 3, hãy đặt 01 câu hỏi tư duy. (Gợi ý: "Hãy so sánh PHT này (Eloquent) với PHT Chương 4 (PDO)."

Lợi ích lớn nhất của Eloquent là gì? Khi nào (trường hợp nào) thì dùng Eloquent sẽ tiện lợi hơn, và khi nào thì dùng PDO/SQL thuần có thể sẽ tốt hơn?).

Câu hỏi của tôi là: Hãy phân tích vai trò của **Route** trong Laravel đối với việc điều hướng request từ trình duyệt đến Controller.

Việc đặt nhiều logic xử lý trực tiếp trong Route Closure so với việc tách logic sang Controller sẽ ảnh hưởng như thế nào đến tính tổ chức, khả năng mở rộng và bảo trì của ứng dụng?

5. Kết nối Đánh giá (Rất quan trọng) Eloquent ORM (Chương 8) là "trái tim" xử lý dữ liệu của Laravel. Kỹ năng make:model, migrate, và sử dụng các phương thức ::all(), ::find(), ::create() là tối quan trọng. Đây là kiến thức trọng tâm của cả Bài tập trên lớp (Phần Laravel) (20%) 2 và Bài tập lớn theo nhóm (50%)³. Nếu không nắm vững Eloquent, bạn không thể xây dựng bất kỳ chức năng nào cho dự án của mình.