# **TRIỂN KHAI PHÁT TRIỂN SẢN PHẨM**

# **TIẾN TRÌNH 1: KHỞI TẠO VÀ XÂY DỰNG BAN ĐẦU**

**Bước 1: Tạo và làm việc trên nhánh riêng (nếu chưa có)**

Đặt tên theo quy tắc: feature/ten-thanh-vien-ten-tinh-nang.

Vídụ:

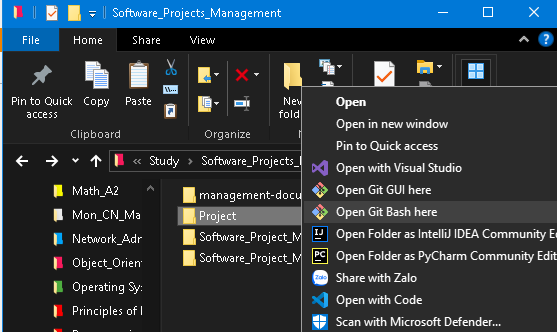
**feature/pchai-orderstatus…**

**feature/nphuy-brand..,**

**feature/nvkha-categories…**

**feature/bnmhuy-posttype…**

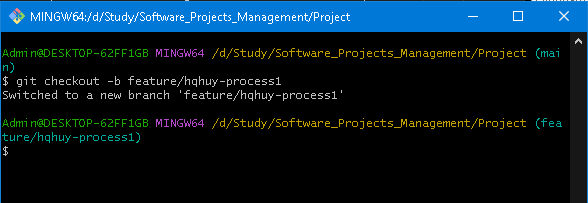
* + Trong thư mục chứa thư mục dự án >chuột trái> Open Git Bash here



* + Chạy lệnh sau:

# Tạo một nhánh mới và chuyển sang nhánh đó

***git checkout -b feature/tinh-nang-cua-ban***



**Bước 2: Kéo cập nhật mới nhất từ các thành viên khác từ nhánh main**

**# Lưu trữ các thay đổi hiện tại (nếu có)**

**git stash**

**# Chuyển về nhánh chính**

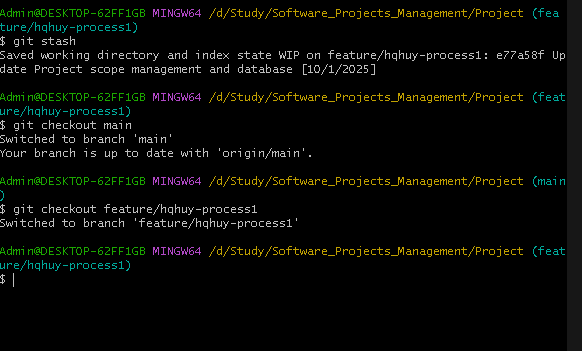
**git checkout main**

**# Kéo mã nguồn mới nhất từ remote**

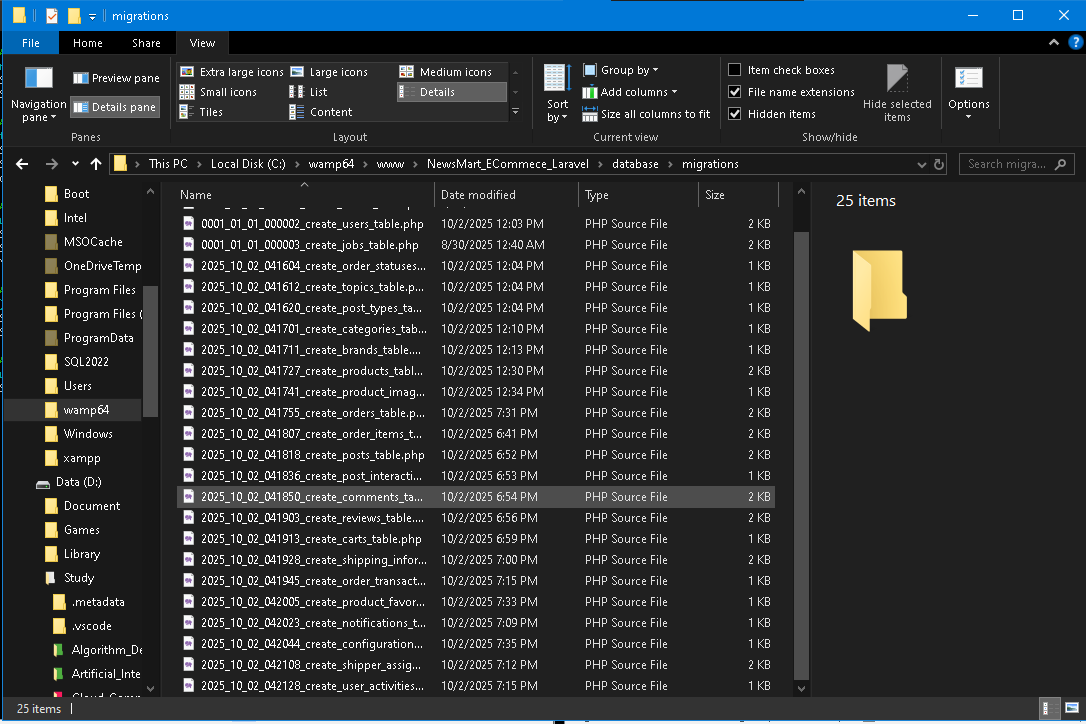
**git pull origin main**

**# Quay lại nhánh của bạn**

**git checkout feature/tinh-nang-cua-ban**

****

* Dự án đã được cập nhật các file trong database/migration chứa các tập tin cấu hình cơ sở dữ liệu.

****

**Bước 3: Tiến hành thực hiễn các tác vụ của từng thành viên:**

**…**

***Mọi người thực hiện tác vụ của mình trong giai đoạn này trước khi áp dụng các thay đổi.***

**…**

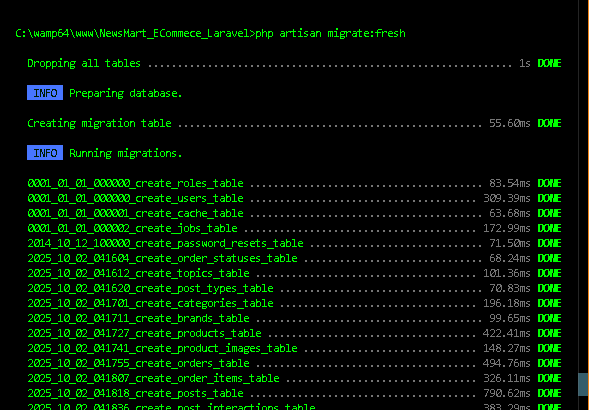
* Database trước đó được các thành viên tạo dùng để xây dựng sơ đồ, riêng database này sẽ là database chính để thực hiện dự án.
* Cập nhật file môi trường **.env:**

|  |
| --- |
| APP\_NAME=NewsMart  APP\_ENV=local  APP\_KEY=base64:UJHk/QKJvUMjR5/kpqjMlwv1pk1j0G0/myYn2npm7Iw=  APP\_DEBUG=true  APP\_URL=http://127.0.0.1:80/NewsMart\_ECommece\_Laravel  APP\_LOCALE=en  APP\_FALLBACK\_LOCALE=en  APP\_FAKER\_LOCALE=en\_US  APP\_MAINTENANCE\_DRIVER=file  # APP\_MAINTENANCE\_STORE=database  PHP\_CLI\_SERVER\_WORKERS=4  BCRYPT\_ROUNDS=12  LOG\_CHANNEL=stack  LOG\_STACK=single  LOG\_DEPRECATIONS\_CHANNEL=null  LOG\_LEVEL=debug  DB\_CONNECTION=mysql  DB\_HOST=127.0.0.1  DB\_PORT=3306  DB\_DATABASE=newsmart  DB\_USERNAME=root  DB\_PASSWORD= |

**Bước 4:**

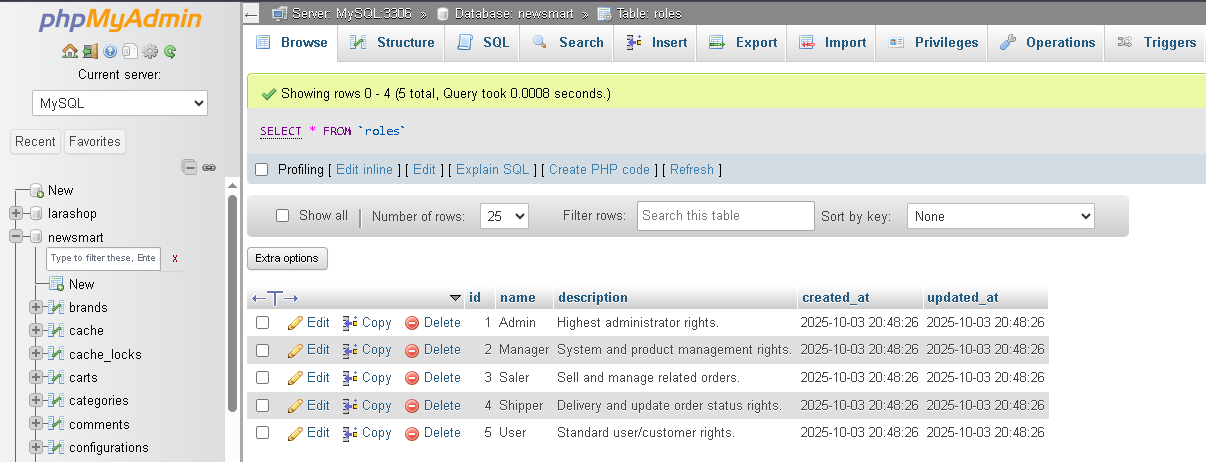
* Khi đã kéo mã nguồn mới nhất từ repository về qua lệnh **git pull origin main** thì tiến hành mở command line chạy lệnh tạo database:

**php artisan migrate:fresh**

****

* Import vào database newsmart:

|  |
| --- |
| INSERT INTO `roles` (`name`, `description`, `created\_at`, `updated\_at`) VALUES  ('Admin', 'Highest administrator rights.', NOW(), NOW()),  ('Manager', 'System and product management rights.', NOW(), NOW()),  ('Saler', 'Sell and manage related orders.', NOW(), NOW()),  ('Shipper', 'Delivery and update order status rights.', NOW(), NOW()),  ('User', 'Standard user/customer rights.', NOW(), NOW()); |



**Bước 5: Triển khai phát triển trang web**

* Cập nhật các file **app/Http/Controllers/[file đang phát triển]**
  + Ví dụ **app/Http/Controllers/BrandControllers.php**
  + Thống nhất đặt tên các hàm là getList, getadd, postAdd, getUpdate, postUpdate, getDelete…

|  |
| --- |
| <?php  namespace App\Http\Controllers;  use App\Models\Brand;  use Illuminate\Http\Request;  use Illuminate\Support\Str;  class BrandController extends Controller  {      public function getList()      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to access brand management.');          }          //$brands = Brand::orderBy('id', 'asc')->get();          $brands = Brand::all();          return view('brands.index', compact('brands'));      }      public function getAdd()      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to add brands.');          }          return view('brands.add');      }      public function postAdd(Request $request)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to add brands.');          }          // Validation          $request->validate([              'name' => ['required', 'string', 'max:255', 'unique:brands'],              'slug' => ['nullable', 'string', 'max:255', 'unique:brands'],              'address' => ['nullable', 'string', 'max:500'],              'email' => ['nullable', 'string', 'email', 'max:255'],              'contact' => ['nullable', 'string', 'max:20'],              'description' => ['nullable', 'string', 'max:1000'],              'logo' => ['nullable', 'image', 'mimes:jpeg,png,jpg,gif', 'max:2048'],          ]);          $brand = new Brand();          $brand->name = $request->name;          $brand->slug = $request->slug ?: Str::slug($request->name, '-');          $brand->address = $request->address;          $brand->email = $request->email;          $brand->contact = $request->contact;          $brand->description = $request->description;          // Handle logo upload if provided          if ($request->hasFile('logo')) {              $logoPath = $request->file('logo')->store('brands/logos', 'public');              $brand->logo = $logoPath;          }          $brand->save();          return redirect()->route('brands')->with('success', 'Brand created successfully!');      }      public function getUpdate($id)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to edit brands.');          }          $brand = Brand::findOrFail($id);          return view('brands.update', compact('brand'));      }      public function postUpdate(Request $request, $id)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to edit brands.');          }          $brand = Brand::findOrFail($id);          $request->validate([              'name' => ['required', 'string', 'max:255', 'unique:brands,name,' . $id],              'slug' => ['nullable', 'string', 'max:255', 'unique:brands,slug,' . $id],              'address' => ['nullable', 'string', 'max:500'],              'email' => ['nullable', 'string', 'email', 'max:255'],              'contact' => ['nullable', 'string', 'max:20'],              'description' => ['nullable', 'string', 'max:1000'],              'logo' => ['nullable', 'image', 'mimes:jpeg,png,jpg,gif', 'max:2048'],          ]);          $brand->name = $request->name;          $brand->slug = $request->slug ?: Str::slug($request->name, '-');          $brand->address = $request->address;          $brand->email = $request->email;          $brand->contact = $request->contact;          $brand->description = $request->description;          if ($request->hasFile('logo')) {              if ($brand->logo && \Storage::disk('public')->exists($brand->logo)) {                  \Storage::disk('public')->delete($brand->logo);              }                $logoPath = $request->file('logo')->store('brands/logos', 'public');              $brand->logo = $logoPath;          }          $brand->save();          return redirect()->route('brands')->with('success', 'Brand updated successfully!');      }      public function getDelete($id)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to delete brands.');          }          $brand = Brand::findOrFail($id);          if ($brand->logo && \Storage::disk('public')->exists($brand->logo)) {              \Storage::disk('public')->delete($brand->logo);          }          $brandName = $brand->name;          $brand->delete();          return redirect()->route('brands')->with('success', "Brand '{$brandName}' deleted successfully!");      }      private function canManageProducts()      {          if (!auth()->check()) {              return false;          }            try {              $userRole = auth()->user()->role->name ?? 'User';                return in\_array(strtolower($userRole), ['admin', 'manager', 'saler']);          } catch (\Exception $e) {              return false;          }      }      public function getBrandsData()      {          if (!$this->canManageProducts()) {              return response()->json(['error' => 'Unauthorized'], 403);          }          $brands = Brand::select(['id', 'name', 'slug', 'email', 'address', 'contact', 'description', 'logo', 'created\_at'])                        ->orderBy('name', 'asc')                        ->get();          return response()->json($brands);      }      public function searchBrands(Request $request)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to search brands.');          }          $query = $request->get('q', '');            $brands = Brand::where(function($q) use ($query) {              $q->where('name', 'LIKE', "%{$query}%")                ->orWhere('description', 'LIKE', "%{$query}%")                ->orWhere('email', 'LIKE', "%{$query}%")                ->orWhere('address', 'LIKE', "%{$query}%");          })          ->orderBy('name', 'asc')          ->get();          return view('brands.index', compact('brands'));      }  } |

* Tạo thư mục **resources/views/[tên bảng viết thường]**
  + Ví dụ **resources/views/brands**
  + Thống nhất các tên file **index.blade.php, add.blade.php, update.blade.php, delete.blade.php**
* Phát triển code trang **index.blade.php**

|  |
| --- |
| <?php  namespace App\Http\Controllers;  use App\Models\Brand;  use Illuminate\Http\Request;  use Illuminate\Support\Str;  class BrandController extends Controller  {      public function getList()      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to access brand management.');          }          //$brands = Brand::orderBy('id', 'asc')->get();          $brands = Brand::all();          return view('brands.index', compact('brands'));      }      public function getAdd()      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to add brands.');          }          return view('brands.add');      }      public function postAdd(Request $request)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to add brands.');          }          // Validation          $request->validate([              'name' => ['required', 'string', 'max:255', 'unique:brands'],              'slug' => ['nullable', 'string', 'max:255', 'unique:brands'],              'address' => ['nullable', 'string', 'max:500'],              'email' => ['nullable', 'string', 'email', 'max:255'],              'contact' => ['nullable', 'string', 'max:20'],              'description' => ['nullable', 'string', 'max:1000'],              'logo' => ['nullable', 'image', 'mimes:jpeg,png,jpg,gif', 'max:2048'],          ]);          $brand = new Brand();          $brand->name = $request->name;          $brand->slug = $request->slug ?: Str::slug($request->name, '-');          $brand->address = $request->address;          $brand->email = $request->email;          $brand->contact = $request->contact;          $brand->description = $request->description;          // Handle logo upload if provided          if ($request->hasFile('logo')) {              $logoPath = $request->file('logo')->store('brands/logos', 'public');              $brand->logo = $logoPath;          }          $brand->save();          return redirect()->route('brands')->with('success', 'Brand created successfully!');      }      public function getUpdate($id)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to edit brands.');          }          $brand = Brand::findOrFail($id);          return view('brands.update', compact('brand'));      }      public function postUpdate(Request $request, $id)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to edit brands.');          }          $brand = Brand::findOrFail($id);          $request->validate([              'name' => ['required', 'string', 'max:255', 'unique:brands,name,' . $id],              'slug' => ['nullable', 'string', 'max:255', 'unique:brands,slug,' . $id],              'address' => ['nullable', 'string', 'max:500'],              'email' => ['nullable', 'string', 'email', 'max:255'],              'contact' => ['nullable', 'string', 'max:20'],              'description' => ['nullable', 'string', 'max:1000'],              'logo' => ['nullable', 'image', 'mimes:jpeg,png,jpg,gif', 'max:2048'],          ]);          $brand->name = $request->name;          $brand->slug = $request->slug ?: Str::slug($request->name, '-');          $brand->address = $request->address;          $brand->email = $request->email;          $brand->contact = $request->contact;          $brand->description = $request->description;          if ($request->hasFile('logo')) {              if ($brand->logo && \Storage::disk('public')->exists($brand->logo)) {                  \Storage::disk('public')->delete($brand->logo);              }                $logoPath = $request->file('logo')->store('brands/logos', 'public');              $brand->logo = $logoPath;          }          $brand->save();          return redirect()->route('brands')->with('success', 'Brand updated successfully!');      }      public function getDelete($id)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to delete brands.');          }          $brand = Brand::findOrFail($id);          if ($brand->logo && \Storage::disk('public')->exists($brand->logo)) {              \Storage::disk('public')->delete($brand->logo);          }          $brandName = $brand->name;          $brand->delete();          return redirect()->route('brands')->with('success', "Brand '{$brandName}' deleted successfully!");      }      private function canManageProducts()      {          if (!auth()->check()) {              return false;          }            try {              $userRole = auth()->user()->role->name ?? 'User';                return in\_array(strtolower($userRole), ['admin', 'manager', 'saler']);          } catch (\Exception $e) {              return false;          }      }      public function getBrandsData()      {          if (!$this->canManageProducts()) {              return response()->json(['error' => 'Unauthorized'], 403);          }          $brands = Brand::select(['id', 'name', 'slug', 'email', 'address', 'contact', 'description', 'logo', 'created\_at'])                        ->orderBy('name', 'asc')                        ->get();          return response()->json($brands);      }      public function searchBrands(Request $request)      {          if (!$this->canManageProducts()) {              abort(403, 'You do not have permission to search brands.');          }          $query = $request->get('q', '');            $brands = Brand::where(function($q) use ($query) {              $q->where('name', 'LIKE', "%{$query}%")                ->orWhere('description', 'LIKE', "%{$query}%")                ->orWhere('email', 'LIKE', "%{$query}%")                ->orWhere('address', 'LIKE', "%{$query}%");          })          ->orderBy('name', 'asc')          ->get();          return view('brands.index', compact('brands'));      }  } |

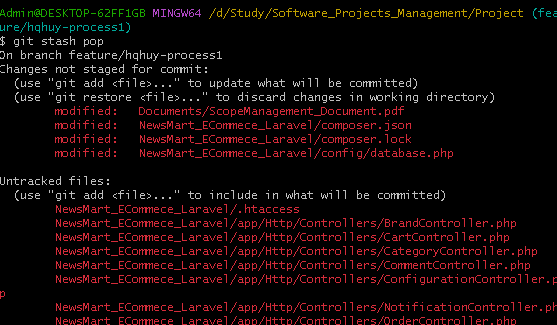
* **âf**

**Bước 6: Cập nhật thay đổi**

* Sau khi đã hoàn thành những đoạn code mới trên repository của bạn hãy thực hiện cập nhật dự án vào repo.

**# Áp dụng các thay đổi đã lưu**

**git stash pop**

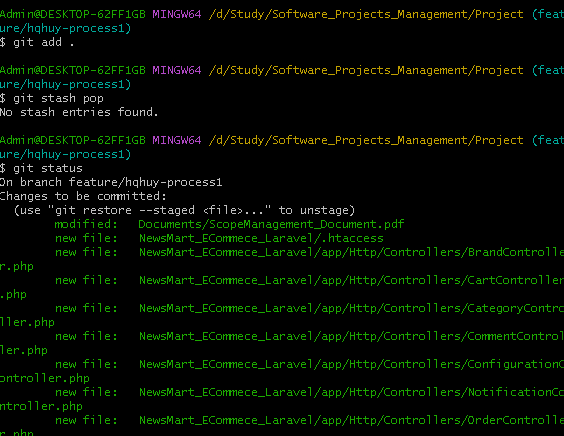
****

# **Thêm các đổi**

**git add .**

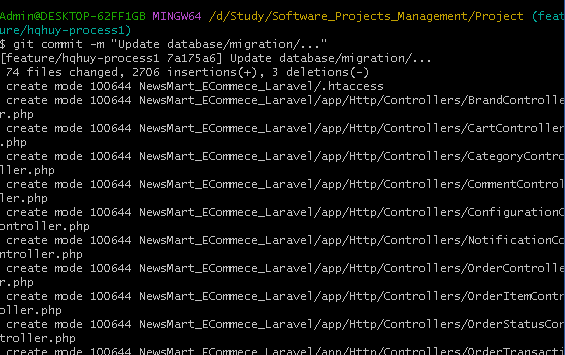
**#kiểm tra thay đổi**

**git status**

****

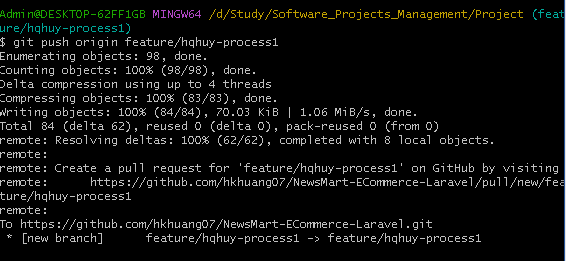
**# chấp nhận các thay đổi**

**git commit –m “[Message…]”**

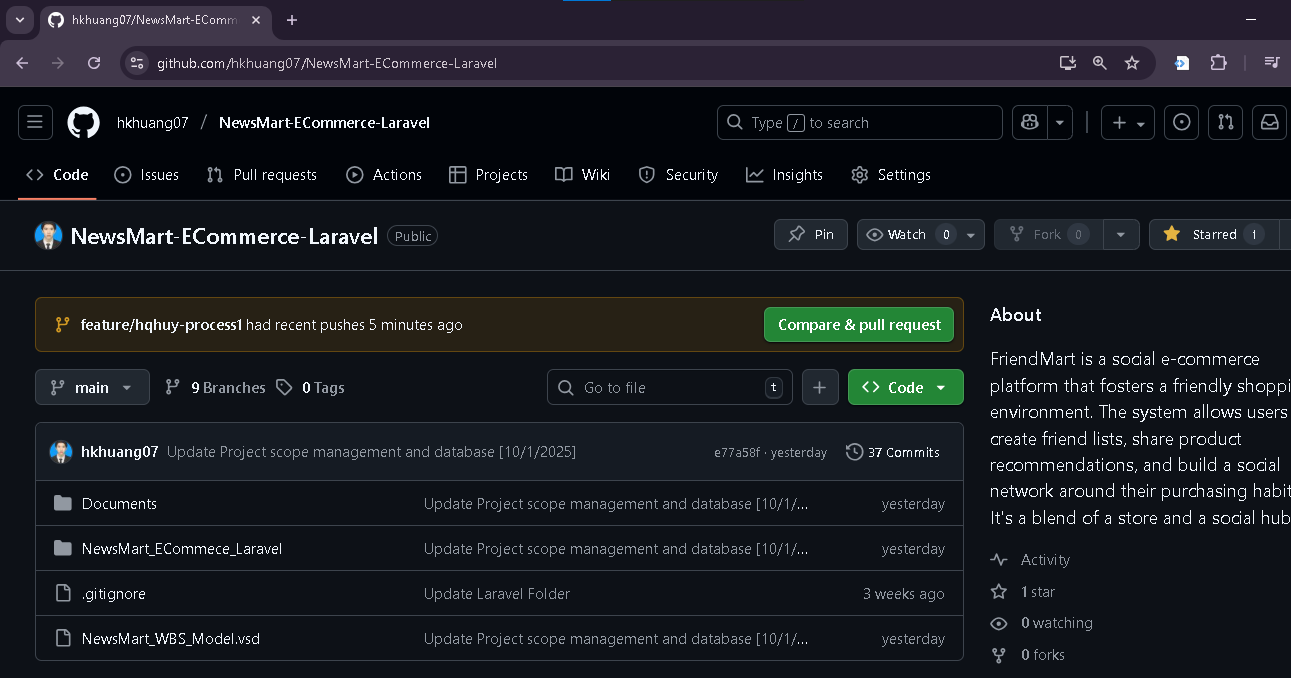


**# cập nhật lên repository**

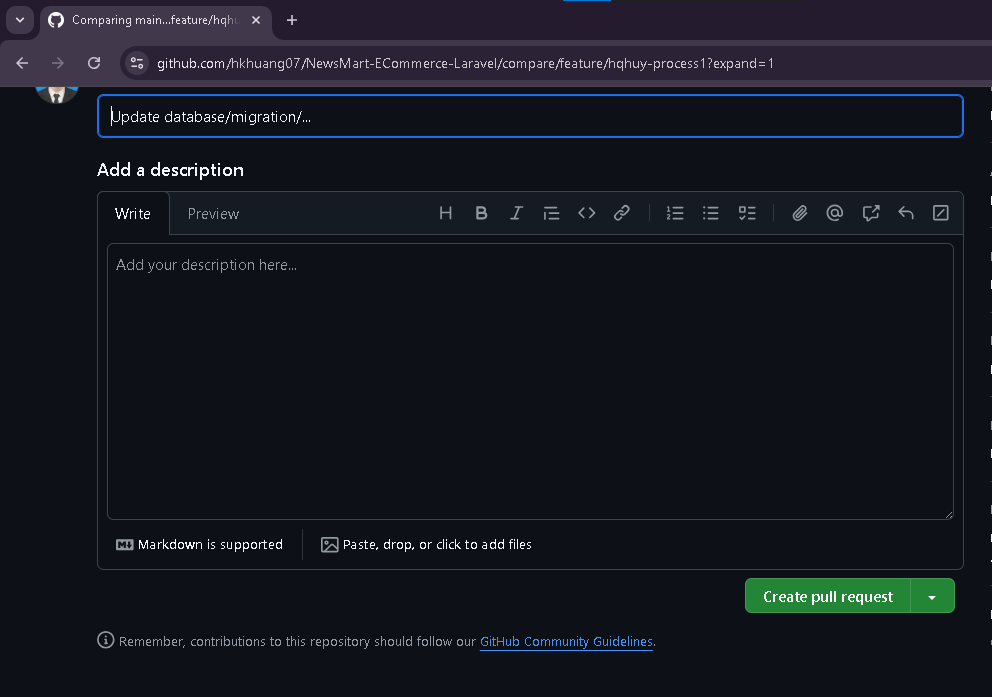
**git push origin feature/ten-nhanh-cua-ban**

****

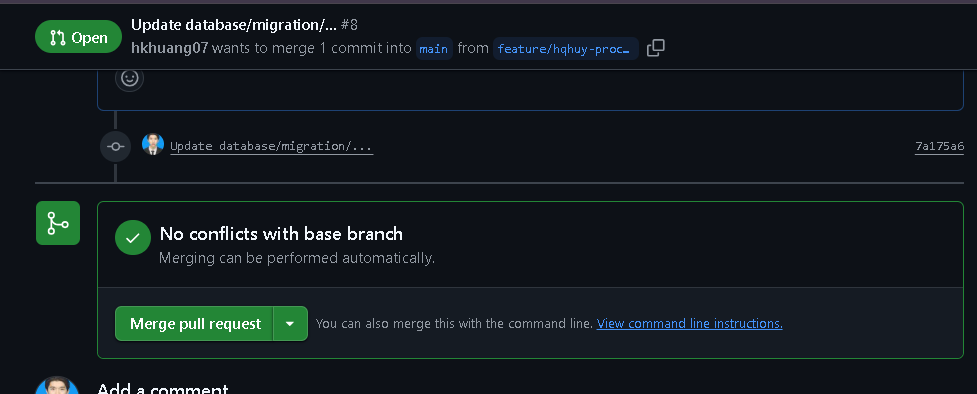
* + Yêu cầu được gửi lên nhánh main (Compare & Pull Request), sẽ được admin(leader) kiểm duyệt có ảnh hưởng đến code khác hay không để cập nhật tiến độ dự án
  + Nếu được **Compare & Pull request (**Dành cho tài khoản tạo repository**)**



* + Tạo **pull request**



* + Confirm Merge – Xác nhận trộn cập nhật mới vào dự án **(**Dành cho tài khoản tạo repository**)**



* + Trạng thái dự án đã được cập nhật từ 1 nhánh thành viên
    - Folder Demo đã được cập nhật vào nhánh main

