1. Làm báo cáo phân tích source mẫu. Về cấu trúc, chức năng, các mục, mô hình code xử lý trong source.
   1. App: Chứa tất cả các project được tạo, hầu hết các class trong project được tạo đều ở đây.
      * **App/Console**: Chứa các tệp tin định nghĩa bao gồm các câu lệnh trên artisan.
        + Helpers: Chức năng thiết lập và định nghĩa: json, slug, random chuỗi, đường dẫn, ký tự, ảnh. *(note: pusher)*
        + Kernel: Tệp định nghĩa câu lệnh artisan.
      * **App/Exceptions**: Chứa tất cả các tệp tin quản lý, điều hướng lỗi.
        + Handler: Tệp quản lý, điều hướng lỗi.
      * **App/Http/Controller**: Chứa các controller của project.
        + Admin: Thư mục chứa các controller liên quan đến người quản trị.
          - DashboardController: Chức năng: điều hướng trang mặc định của admin.
          - ErrorController: Chức năng: điều hướng trang lỗi khi xảy ra lỗi(404,505).
          - LibraryController: Chức năng: điều hướng về trang thư viện, tìm kiếm file, xử lý file và ảnh, xóa và tải file*(note: leftjoin, offset, intval, firstOrFail).*
          - PagesController: Chức năng CRUD pages và tìm kiếm trang.
          - ProfileController: Chức năng: cập nhật lại thông tin cá nhân, kiểm tra sự tồn tại của email và số điện thoại.
          - ProjectController: Chức năng: điều hướng trang dựa án, CRUD dựa án, xử lý ảnh và đường dẫn (*note: when*).
          - RolesController: Chức năng: Điều hướng trang cài đặt hệ thống phân quyền, CRUD hệ thống phân quyền.
          - ServicesController: Chức năng: Điều hướng trang dịch vụ, CRUD dịch vụ.
          - SettingController: Chức năng: Điều hướng về trang cài đặt hệ thống, xử lý file, cập nhật: logo, icon, tên trang web.
          - UsersController: Chức năng: Điều hướng về trang thành viên, CRUD thành viên, xóa hàng loạt.
        + Auth: Thư mục chứa các controller liên quan đến hành động xác minh danh tính của người dùng hay gọi là xác thực.
          - ForgotPasswordController: Chức năng: quên mật khẩu(mặc định của laravel).
          - LoginController: Chức năng: Đăng nhập, đăng xuất.
          - RegisterController: Chức năng: Đăng ký tài khoản
          - ResetPasswordController: Chức năng: Đặt lại mật khẩu.
          - VerificationController: Chức năng: Xác minh tài khoản.
        + Guest: Thư mục chứa các controller liên quan đến khách hàng vãng lai.
          - HomeController: Chức năng: Điều hướng trang “Tra cứu thông tin dịch vụ”, tìm kiếm thông tin trang “Tra cứu thông tin dịch vụ”, lấy thông tin trang “Tra cứu thông tin dịch vụ”, tải file.
          - LoginController: Chức năng: Đăng nhập, kiểm tra tài khoản, đăng xuất.
          - PagesController: Chức năng: Lấy thông tin trang.
        + Controller: Chức năng: Sử dụng những thư viện: authorizesRequests, DispatchesJob, ValidatesRequests.
      * **App/Http/Middleware**: Chức năng: Chứa các tệp tin lọc và ngăn chặn các request.
        + Authenticate: Chuyển hướng trang người dùng, khi họ không được xác thực.
        + CheckAuth: Chức năng: Xử lý yêu cầu đăng nhập (*note:Closure*)
        + CheckForMaintenanceMode: Chức năng: Những url sẽ còn hoạt động khi hệ thống vào trạng thái bảo trì.
        + CheckPermission: Chức năng: Yêu cầu cần quyền để sử dụng chức năng này (*note: pluck, collect, explode, contains*).
        + CustomCKFinderAuth: Chức năng: thiết lập khóa giấy phép, và tên giấy phép (*note: Config, ckfinder*).
        + EncryptCookies: Chức năng: Không mã hóa tên của Cookies.
        + RedirectIfAuthenticated: Chức năng: Điều hướng trang khi xác thực đăng nhập thành công.
        + TrimStrings: Chức năng: Tên của những thuộc tính không nên cắt.
        + TrustProxies: Chức năng: Các tiêu đề nên được sử dụng để phát hiện proxy.
        + VerifyCrsfToken: Chức năng: Các url nên được loại trừ khỏi xác minh CSRF.
      * **App\Models**: Chức năng: Chứa các mô hình dữ liệu.
        + Pages
        + Projectfile
        + Projects
        + Roles
        + Services
        + Settings
        + Users
      * **App\Providers:** Chức năng: Chứa các file thực việc khai báo service và bind vào trong Service Container.
      * **Bootstrap:** Chức năng: Chứa những file khởi động cảu frame work và những file cấu hình auto loading, route, và file cache.
      * **Config:** Chức năng: Chứa tất cả các file cấu hình.
        + App (mặc định)
        + Auth (mặc định)
        + Broadcasting (mặc định)
        + Cache (mặc định)
        + Ckfinder (import)
        + Database (mặc định)
        + Filesystems (mặc định)
        + Hashing (mặc định)
        + Logging (import)
        + Mail (mặc định)
        + Queue (mặc định)
        + Seotools (import)
        + Services (mặc định)
        + Session (mặc định)
        + View (mặc định)
      * **Database:** Chứa 2 thư mục migration (tạo và thao tác database) và seeds (tạo dữ liệu mẫu), tiện lợi để lưu trữ dữ liệu sau này.
      * **Database/factories:** Chứa các file định nghĩa các cột bảng dữ liệu để tạo ra các dữ liệu mẫu.
      * **Database/migrations**: Chứa các file tạo và chỉnh sửa dữ liệu.
        + Create\_roles\_table: Tạo ra bảng dữ liệu có các thuộc tính: id, role\_type, role\_name, role\_note, permission.
        + Create\_users\_table: Tạo ra bảng dữ liệu có các thuộc tính: id (khóa chính), type, rolesID (khóa ngoại), username, full\_name, email, phone\_number, password.
        + Create\_password\_reset\_table: Tạo ra bảng dữ liệu có các thuộc tính: email, token.
        + Create\_services\_table: Tạo ra bảng dữ liệu có các thuộc tính: id (khóa chính), service\_name, service\_description, service\_slug, service\_land.
        + Create\_projects\_table: Tạo ra bảng dữ liệu có các thuộc tính: id (khóa chính), companyID (khóa ngoại), serviceID (khóa ngoại), projects\_name, time\_begin, time\_end, paths, projects\_note.
        + Create\_settings\_table: Tạo ra bảng dữ liệu có các thuộc tính: id (khóa chính), website\_admin, website\_logo, website\_icon, website\_logo\_text, website\_title, website\_name, website\_address, website\_phone\_number, website\_table\_number\_page.
        + Create\_projects\_files\_table: Tạo ra bảng dữ liệu có các thuộc tính: id (khóa chính), projectID (khóa ngoại), key, file\_url, file\_name.
        + Create\_pages\_table: Tạo ra bảng dữ liệu có các thuộc tính: id (khóa chính), key, link, type\_open, status, slug, content.
      * **Database/seeds**: Chứa các file tạo dữ liệu thêm vào CSDL
        + DatabaseSeeder: Chức năng: gọi các lớp đã được tạo dữ liệu thêm vào CSDL.
        + PagesTableSeeder: Tạo ra file dữ liệu thêm vào bảng pages trong CSDL.
        + ProjectsTableSeeder: Tạo ra file dữ liệu thêm vào bảng projects trong CSDL.
        + RolesTableSeeder: Tạo ra file dữ liệu thêm vào bảng roles trong CSDL.
        + ServicesTableSeeder: Tạo ra file dữ liệu thêm vào bảng services trong CSDL.
        + SettingsTableSeeder: Tạo ra file dữ liệu thêm vào bảng settings trong CSDL.
        + UsersTableSeeder:Tạo ra file dữ liệu thêm vào bảng users trong CSDL.
      * **Public:** Chứa file javaScript và CSS.
        + App: Chứa các đoạn js xử lý formData và Databases.
          - Admin/Auth/auth.min.js: Sau khi submit dữ liệu từ form-login. Ajax sẽ điều hướng về Controller (LoginController). Sau đó sẽ gọi hà post login để kiểm tra dữ liệu của 2 thuộc tính username và password. Nếu thành công sẽ được điều hướng về trang dasboard, còn nếu thất bại sẽ thông báo lỗi cho người dùng biết.
          - Admin/Dashboard/dashboard.min.js: Ở đây dữ liệu của dashboard được khởi tạo.
          - Admin/File/file.js: Tạo một DataTable trên table-file với các thuộc tính như **serverSides, processing, paging, lengthChange, ordering, info, responsive** và loại bỏ một số thuộc tính **searching, autowidth.**

Luồng sự kiện 1: Khi ajax được gọi lần thứ nhất, dữ liệu sẽ được gom lại thành một đối tượng bằng $.extend, rùi sẽ được điều hướng về router của libraryController sau đó sẽ được gọi function table, lúc này ở controller bắt đầu kết bảng dữ liệu với nhau. Sau đó dữ liệu đó được đổ lên DataTable tương ứng theo các thuộc tính.

Luồng sự kiện 2: Khi bấm vào nút btn-delete, thông báo sẽ hiện ra và khi xác nhận đồng ý, sẽ được điều hướng về LibraryController gọi phương thức delete để xử lý. Sau đó load lại trang.

Luồng sự kiện 3: Khi bấm vào btn-dowload sẽ được điều hướng về một cửa sổ mới và gọi LibraryController cùng với phương thức dowload. Sau đó file sẽ được tải về máy.

* + - * + Admin/pages/action.min.js:

Luồng sự kiện 1: Khi bấm vào nút .btn-update dữ liệu sẽ truyền id qua PagesController và gọi phương thức postUpdate, lúc này dữ liệu được cập nhật và trả về 1 mảng json.

Luồng sự kiện 2: Khi bấm vào .pages-status sẽ được điều hướng về PagesController rùi sẽ được điều hướng về phương thức postStatus, lúc này ở controller sẽ thực hiện chức năng cập nhật lại trạng thái. Nếu thành công sẽ hiện thông báo thành công bằng toast cho người dùng biết, nếu thất bại sẽ hiện thông báo bằng toast .

Luồng sự kiện 3: Tạo ra #formAction với các validate và add các thêm các element. Sau khi gọi ajax với phương thức post, dữ liệu là formdata qua PagesController với phương thức PostInsert dữ liệu được thêm vào database, sau đó trả về một mảng json. Khi đoạn ajax được thực hiện thành công sẽ hiên thông báo thành công và hiện 2 nút Save và Apply. Nếu thất bại sẽ hiện thông báo thất bại và hiện 2 nút Save và Apply.

Luồng sự kiện 4: Khi click vào nút Save hoặc Apply dữ liệu trong formData sẽ submit.

* + - * + Admin/pages/pages.min.js: Khởi tạo đối tượng pages. Tạo một DataTable trên table-pages với các thuộc tính như **serverSides, processing, paging, lengthChange, ordering, info, responsive** và loại bỏ một số thuộc tính **searching, autowidth.**

Luồng sự kiện 1: Khi ajax đầu tiên được gọi với phương thức get và dữ liệu (search) được gộp lại bởi phương thức $.extend, và sau đó được điều hướng PagesController với phương thức getDatatale, ở đây dữ liệu sẽ được tìm theo (search) và trả về một mảng dữ liệu. Sau đó được lên Datatable theo các thuộc tính (id, name, link, status, key).

Luồng sự kiện 2: Khi bấm vào nút insert, sẽ được điều hướng sang PagesController cùng với phương thức postInsert. Ở đây dữ liệu sẽ được xử lý thêm vào database, sau đó sẽ trả về một mảng json.

Luồng sự kiện 3: Khi bấm vào nút update, sẽ được điều hướng sang PagesController cùng với phương thức postUpdate. Ở đây dữ liệu sẽ được xử lý cập nhật và database, sau đó sẽ trả về một mảng json.

Luồng sự kiện 4: Khi bấm vào .pages-status, ajax sẽ được gọi đồng thời sẽ được điều hướng về PagesController cùng với phương thức postStatus, Ở đây thuộc tính status sẽ được hoán đổi (0 và 1). Sau đó sẽ trả về một mảng json. Nếu thành công sẽ hiện thông báo thành công cho người dùng. Nếu thất bại sẽ hiện thông báo thất bại cho người dùng.

Luồng sự kiện 5: Tạo một formAction cùng với các validate và các element. Sau đó ajax được gọi với phương thức post và dữ liệu là formdata sẽ được điều hướng sang pagesController cùng với phương thức postUpdate, ờ đây dữ liệu sẽ được cập nhật và trả về một mảng dữ liệu json. Khi đoạn ajax được thực hiện thành công sẽ hiên thông báo thành công và hiện 2 nút Save và Apply. Nếu thất bại sẽ hiện thông báo thất bại và hiện 2 nút Save và Apply.

* + - * + Admin/profile/profile.js: Khởi tạo dữ liệu profile

Luồng sự kiện 1: Tạo một formProfile cùng với validate và các element, ajax sẽ được gọi và được điều hướng ProfileController cùng với phương thức postUpdate, ở đây dữ liệu sẽ được kiểm tra tồn tại chưa nếu thành công sẽ trả về mảng dữ liệu, còn nếu thất bại sẽ trả về một mảng json. Khi đoạn ajax được thực hiện thành công sẽ hiện thông báo thành công người dùng và hiện thông tin người dùng mới cập nhật, nếu thất bại sẽ hiện thông báo thất bại cho người dùng biết.

Luồng sự kiện 2: Khi submit dữ liệu trong formChangePassword, đoạn ajax được thực hiện cùng với phương thức post và dữ liệu là từ formdata. Sau đó sẽ được điều hướng qua ProfileController cùng với phương thức postChangePassword, ở đây dữ liệu sẽ cập nhật lại mật khẩu, nếu thành công sẽ trả mảng json cập nhật mật khẩu thành công, nếu thất bại sẽ trả về mảng json thông báo lỗi thất bại. Khi đoạn ajax được thực hiện thành công sẽ hiện thông báo thành công cho người dùng biết, và reset lại form mật khẩu, nếu thất bại sẽ hiện thông báo thất bại cho người dùng biết.

* + - * + Admin/projects/action.min.js: Khởi tạo dữ liệu cho project, khởi tạo thời gian bắt đầu, và khởi tạo thời gian kết thúc, thiết lập nút chọn ảnh và bức ảnh.

Luồng sự kiện 1: Khi chọn click vào .btn-delete-images, sẽ thêm id vào mảng deleteId sau đó remove (“.item-images”).

Luồng sự kiện 2: Tạo formAction cùng với các validate và các element, khởi tạo formData với các thuộc tính dữ liệu (action, id, projects\_note, deleteID và paths). ajax sẽ được gọi cùng với phương thức post và dữ liệu là formdata sau khi và được điều hướng sang ProjectsController với phương thức postUpdate hoặc postInsert. Dữ liệu sẽ được cập nhật lại trong database, sau đó trả về một mảng json. Khi đoạn ajax dược thực hiện thành công sẽ hiện thông báo thành công và xuất hiện nút save và apply, nếu chọn Save sẽ được điều hướng về ProjectController và gọi phương thức getExpired, rùi sẽ được điều hướng về trang Dựa án còn hạn, còn chọn apply sẽ điều hướng về ProjetcController gọi phương thức getUpdate ở đây sẽ tìm dữ liệu cần update theo id sau đó điều hướng về trang cập nhật bài viết. Còn nếu thất bại sẽ hiện thị thông báo thất bại cho người dùng.

* + - * + Admin/projects/projects.min.js: Kiểm tra ngày tháng của dự án, định dạng lại ngày tháng của dự án, khởi tạo project. Tạo một DataTable trên table-pages với các thuộc tính như **serverSides, processing, paging, lengthChange, ordering, info, responsive** và loại bỏ một số thuộc tính **searching, autowidth.**

Luồng sự kiện 1: Khi ajax đầu tiên được gọi với phương thức get và dữ liệu (search) được gộp lại bởi phương thức $.extend, và sau đó được điều hướng về ProjectController với thuộc tính getDatatableAll ở đây dữ liệu sẽ được tìm theo (search) và trả về một mảng dữ liệu. Sau đó được lên Datatable theo các thuộc tính (id, company\_name, projects\_name, service\_name, time\_begin, time\_end, update\_at).

Luồng sự kiện 2: Khi nhập thông tin vào thanh tìm kiếm, datatables sẽ reload lại.

Luồng sự kiện 3: Khi chọn .btn-delete, modal-text-delete sẽ hiện ra.

Luồng sự kiện 4: Khi chọn .btn-update sẽ được điều hướng về ProjectsController với phương thức getUpdate ở đây dữ liệu sẽ được tìm theo id sau đó sẽ được điều hướng về trang cập nhật bài viết.

Luồng sự kiện 5: Khi chọn .btn-info, ajax sẽ được gọi và điều hướng về ProjectController với phương thức getInformation ở dây sẽ tìm thông tin theo dữ liệu được truyền qua, sau đó được điều hướng về trang modalContent.

Luồng sự kiện 6: Khi chọn #btn-insert, sẽ được điều hướng về trang ProjectsController với phương thức getInsert ở đây sẽ được điều hướng về trang tạo bài viết.

Luồng sự kiện 7: Khi chọn .post-status ajax sẽ được gọi với phương thức post và dữ liệu gồm (id và status). Nếu ajax được thực hiện thành công sẽ hiện thông báo thành công, còn nếu thất bại sẽ hiện thông báo thất bại.

* + - * + Admin/roles/roles.min.js: Khởi tạo dữ liệu cho roles và tạo một DataTable trên table-pages với các thuộc tính như **serverSides, processing, paging, lengthChange, ordering, info, responsive** và loại bỏ một số thuộc tính **searching, autowidth.**

Luồng sự kiện 1: Khi ajax đầu tiên được gọi với phương thức get và dữ liệu (#search và #id\_branch\_search) được gộp lại bởi phương thức $.extend, và sau đó được điều hướng về ProjectController với thuộc tính getDatatableAll ở đây dữ liệu sẽ được tìm theo (search) và trả về một mảng dữ liệu. Sau đó được lên Datatable theo các thuộc tính (id, role\_name, role\_note, id).

Luồng sự kiện 2: Khi nhập dữ liệu vào (#search,#id\_branch\_search,#formSearch), table sẽ được reload.

Luồng sự kiện 3: Khi chọn .btn-delete modal-text-delete sẽ hiẹn ra.

Luồng sự kiện 3: Khi chọn .btn-update ajax sẽ được gọi với phương thức get và được điều hướng về RolesController với phương thức getUpdate, ở đây dữ liệu sẽ được tìm kiếm theo admin và id sau đó được trả về mảng json. Sau khi đoạn ajax được thực hiện thành công thì tên và chú thích cũng được cập nhật lại. Còn nếu thất bại sẽ hiện thông báo thất bại cho người dùng biết.

Luồng sự kiện 4: Khi chọn #btn-ínsert, khởi tạo các thuộc tính về mặc định (rỗng). Khởi tạo (#formAction) với các validate và element. Khởi tạo FormData với thuộc tính (id). Ajax sẽ được gọi với phương thức post và dữ liệu là formData và được điều hướng RolesController với phương thức postInsert ở đây dữ liệu đươc thêm vào database và trả về một mảng json. Nếu như ajax được thực hiện thành công thì sẽ hiện thông báo thành công và table sẽ tự động load lại. Còn nếu thất bại sẽ hiện thông báo thất bại cho người dùng biết.

Luồng sự kiện 5: Khi chọn #onDelete sẽ được điều hướng về RolesController với phương thức postDelete ở đây sẽ tìm dữ liệu theo Admin và id và xóa. Nếu thành công sẽ trả về thông báo json thành công. Còn nếu thất bại sẽ trả về thông báo json thất bại. Nếu chạy thành công table sẽ load lại và ẩn modal đi.

* + - * + Admin/service/service.min.js: Khởi tạo dữ liệu cho service và tạo một DataTable trên table-service với các thuộc tính như **serverSides, processing, paging, lengthChange, ordering, info, responsive** và loại bỏ một số thuộc tính **searching, autowidth.**

Luồng sự kiện 1: Khi ajax đầu tiên được gọi với phương thức get và dữ liệu (#search) được gộp lại bởi phương thức $.extend, và sau đó được điều hướng về ServicesController với thuộc tính getDataTable ở đây dữ liệu sẽ được tìm theo (#search) và trả về một mảng dữ liệu. Sau đó được lên Datatable theo các thuộc tính (id, service\_name, service\_description, create\_at, update\_at).

Luồng sự kiện 2: Khi nhập dữ liệu vào (#search, #id\_branch\_search, #formSearch) table sẽ tự động reload dữ liệu.

Luồng sự kiện 3: Khi chọn .btn-delete modal-text-delete sẽ hiẹn ra.

Luồng sự kiện 4: Khi chọn .btn-update ajax sẽ được điều hướng về ServiceController với phương thức getUpdate ở đây dữ liệu sẽ được tìm kiếm theo service.id và trả về một mảng dữ liệu. Khi ajax thực hiện thành công button Save sẽ cập nhập thành button update, service\_name và service\_description cũng được cập nhật lại, Nếu thất bại chương trình sẽ thông báo thất bại cho người dùng.

Luồng sự kiện 5: Khi chọn vào (#btn-insert) modal sẽ cập nhật tên thành thêm mới và service\_name và service\_description sẽ trở thành rỗng. Button sẽ có đường dẫn đến insert. Tạo #formAction với các validate và các element, sau đó tạo formData cùng với các thuộc tính (id, type). Ajax được gọi với phương thức Post và dữ liệu là formdata và được điều hướng về ServicesController với phương thức postInsert ở đây dữ liệu sẽ được thêm vào database và trả lại thông báo json. Sau khi ajax thực hiện thành công table sẽ tự động reload lại và modal sẽ ẩn đi, và xuất hiện thông báo cho người dùng biết đã thành công. Còn nếu lỗi sẽ xuất hiện thông báo thất bại cho người dùng.

Luồng sự kiện 6: Khi click vào #onDelete sẽ được điều hướng về ServicesController với phương thức postDelete ở đây dữ liệu sẽ được tìm kiếm theo id và bị xóa, sau đó sẽ trả lại thông báo json. Sau khi thành công sẽ ẩn modal và table sẽ reload lại. Còn nếu thất bại sẽ ẩn modal.

* + - * + Admin/setting/setting.min.js: Khởi tạo dữ liệu setting. Thiết lập các hàm chọn và thay dổi icon, tên trang web, logo

Luồng sự kiện 1: Tạo #formSettingAction với các validate (required và maxlenght) và các element, tạo dữ liệu formdata, ajax được gọi với phương thức post và dữ liệu là formdata, sau đó được điều hướng về SettingController với phương thức postUpdate ở đây dữ liệu sẽ được tìm và được cập nhật mới, sau đó sẽ trả về một thông báo json. Nếu thành công dữ liệu sẽ trả về một thông báo thành công. Nều thất bại sẽ thông báo cho người dùng biết.

* + - * + Admin/user/user.min.js: Khởi tạo dữ liệu user và tạo một DataTable trên table-service với các thuộc tính như **serverSides, processing, paging, lengthChange, ordering, info, responsive** và loại bỏ một số thuộc tính **searching, autowidth.**

Luồng sự kiện 1: Ajax sẽ được gọi với phương thức get và dữ liệu được gộp bởi $.extend (#type và #search). Sau đó được điều hướng về UsersController với phương thức getDatatale, ở đây dữ liệu được tìm kiếm theo type và serach (có thể bị rỗng). Sau đó trả về một mảng json dữ liệu. Sau đó sẽ load lên datatables với các thuộc tính tương ứng (id, full\_name, host, phone\_number, email, address, note, total\_projects).

Luồng sự kiện 2: Khi nhập dữ liệu vào (#search, #id\_branch\_search, #fỏmSearch), Datatables sẽ tự động reload lại.

Luồng sự kiện 3: Khi chọn .btn-update, #formAction sẽ tự reset, thay đổi nội dung modal, loại bỏ class error. Khi ajax được gọi với phương thức get và dữ liệu là id, sẽ được điều hướng về UserController với phương thức getUpdate ở đây dữ liệu sẽ được tìm theo id và cập nhật dữ liệu, sau đó trả về một mảng json. Nếu ajax thực hiện thành công thay đổi button save thành update và hiện thông tin trên modal. Nếu thất bại sẽ thông báo cho người dùng biết.

Luồng sự kiện 4: Thay đổi trạng thái và phê duyệt, Ajax được gọi với phương thức post và dữ liệu là một mảng id, sẽ diều hướng controller tùy vào tùy chọn người dùng. Sau khi ajax có thể xóa hoặc chấp nhận, trạng thái, bỏ trạng thái hàng loạt, nếu thành công sẽ hiện thông báo thành công. Nếu thất bại sẽ hiện thông báo thất bại.

Luồng sự kiện 5: Khi chọn #onDelete sẽ được điều hướng về UsersController với phương thức postDelete ở đây sẽ tìm kiếm theo id và xóa, sau đó trả về một mảng json. Sau khi thành công table sẽ tự động reload lại và ẩn modal, còn nếu thất bại ẩn modal.

* + - * + Admin/main.min.js: Xử lý mã hóa base64, thiết lập định dạng ảnh, thiết lập logo, thiết lập icon, thiết lập file admin, đọc ảnh, trạng thái màu chữ, alterjs(thông báo js),định dạng thời gian, từ ngữ, thiết lập chống corss.
        + Guest/auth/auth.min.js: Sau khi submit dữ liệu vào #form-login, ajax sẽ được gọi với phương thức post và dữ liệu lấy từ formdata và được chuyển hướng về LoginController với phương thức getLogin sẽ kiểm tra thông tin người dùng (username và password). Sau khi ajax thực hiện thành công sẽ điều hướng về trang home, còn nếu thất bại sẽ hiện thông báo thất bại cho người dùng biết.
        + Guest/home/home.min.js: Khởi tạo giá trị home. Tạo #search-form với các validate và element, tạo một DataTable trên #table-search với các thuộc tính như **serverSides, processing, paging, lengthChange, ordering, info, responsive** và loại bỏ một số thuộc tính **searching, autowidth.**

Luồng sự kiện 1: Ajax sẽ được gọi với phương thức get và dữ liệu được gộp bởi $.extend (search:searchkey). Sau đó được điều hướng về HomeController với phương thức searchHome, ở đây dữ liệu được tìm kiếm theo search và được trả về một mảng dữ liệu, sau đó sẽ được đổ lên datatables theo các thuộc tính sau (id, full\_name, service\_name, projects\_name, time\_end, projectsID ).

Luồng sự kiện 2: Khi chọn .btn-view-info ajax sẽ được gọi với phương thức get, và dữ liệu (id và prọectID) và sẽ được điều hướng về HomeController với phương thức getInformation ở đây dữ liệu sẽ được tìm kiếm theo id và projectId và trả về một dữ liệu json. Khi ajax thực hiện thành công sẽ hiện modal và đổ dữ liệu lên modal. Còn nếu thất bại sẽ hiện thị lỗi cho người dùng biết.

* + - * + Guest/main.min.js: Thiết lập chống corss
      * Ckeditor: Thư viện CK editor
      * Data: Chứa các file âm thanh, logo, icon, gif.
      * CKfinder: Thư viện CK finder
      * Themes: Chứa các đoạn css xử lý giao diện.
        + Admin
        + Guest
      * Uploads: Chứa các tệp file và ảnh.
        + Files
        + Images
    - **Resources**: Chứa những file view và raw, các file biên soạn như LESS, SASS, hoặc JavaScript. Ngoài ra còn chứa tất cả các file lang trong project.
      * Resources/views: Chứa các file view xuất giao diện người dùng.
    - **Routes**: Chứa tất cả các điều khiển route (đường dẫn) trong project. Chứa các file route sẵn có
      * Admin.php
      * Api.php
      * Channels.php
      * Ckfinder.php
      * Console.php
      * Web.php
    - **Storage:** Chứa các file biên soạn blade templates của bạn, file based sessions, file caches, và những file sinh ra từ project
      * App: dùng để chứa những file sinh ra từ project.
      * Framework: Chứa những file sinh ra từ framework và caches.
      * Logs: Chứa những file logs.
      * /Strorage/app/public: Lưu những file người dùng tạo ra như hình ảnh.
    - **Tests**: Chứa những file tests, như PHPUnit test.
    - **Vendor:** Chứa các thư viện của Composer.
    - **.env:** Chứa các config chính của laravel.
    - **Artisan:** file thực hiện lệnh của laravel
    - **.gitattributes && .gitignore:** file dành cho xử lý git.
    - **Composer.json && composer.lock && composer-setup.php:** File của Composer.
    - **Package.js:** file package.js, chứa các package cần dùng cho projects.
    - **Phpunit.xml:** file phpunit.xml, xml của phpunit dùng để testing project.
    - **Webpack.min.js:** file dùng để build các webpack.
  1. **Mô hình xử lý code :** 
     + Mô hình xử lý code trong source là mô hình MVC. Viết tắt của cụm từ “Model-View-Controller”. MVC là một mẫu kiến trúc phần mền để tạo lập giao diện người dùng trên máy tính. MVC chia thành ba phần được kết nối với nhau. Bao gồm Model (dữ liệu), View (giao diện) và Controller (bộ điều khiển).
     + Mô hình gồm 3 loại chính:
       - Model(M)
         * Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng. Bộ phận này là một cầu nối giữa 2 thành phần bên dưới là View và Controller. Model thể hiện dưới hình thức là một cơ sở dữ liệu hoặc có khi chỉ đơn giản là một file XML bình thường. Model thể hiện rõ các thao tác với cơ sở dữ liệu như cho phép xem, truy xuất, xử lý dữ liệu,…
       - View(V)
         * Đây là phần giao diện (theme) dành cho người sử dụng. Nơi mà người dùng có thể lấy được thông tin dữ liệu của MVC thông qua các thao tác truy vấn như tìm kiếm hoặc sử dụng thông qua các website.
         * Thông thường, các ứng dụng web sử dụng MVC View như một phần của hệ thống, nơi các thành phần HTML được tạo ra. Bên cạnh đó, View cũng có chức năng ghi nhận hoạt động của người dùng để tương tác với Controller. Tuy nhiên, View không có mối quan hệ trực tiếp với Controller, cũng không được lấy dữ liệu từ Controller mà chỉ hiện thị yêu cầu chuyển cho Controller.
       - Controller(C)
         * Bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua view. Từ đó, Controller đưa ra dữ liệu phù hợp với người dùng. Bên cạnh đó, Controller còn có chức năng kết nối với model.
         * Luồng đi trong mô hình MVC, khi một yêu cầu từ máy client gửi tới server, Controller sẽ thực hiện nhiệm vụ của mình đó là tiếp nhận và xử lý yêu cầu. Trong trường hợp cần thiết, nó có thể liên hệ Model - bộ phận làm việc với database để hỗ trợ.
         * Khi xử lý xong yêu cầu, kết quả sẽ được trả về View. Tại View sẽ tạo mã HTML thành giao diện và trả về hiện thị trên trình duyệt.