

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**🙞🕮🙜**



**BÁO CÁO FINAL PROJECT LẦN 1   
MÔN: CÔNG NGHỆ PHẦN MỀM MỚI**

**ĐỀ TÀI:**

**WEBSITE BÁN SÁCH TRỰC TUYẾN**

**GVHD : Nguyễn Thanh Phước**

**SVTH :**

**Nguyễn Thành Như 17110202**

**Võ Ngọc Thuận 17110234**

**TP. Hồ Chí Minh, tháng 11 năm 2020**

# GitHub Project Link

https://github.com/VoNgocThuan/VoNgocThuan-NguyenThanhNhu-CacCNPMMoiFinalProject.git

# CSDL (tạm thời)

Nhóm em sử dụng MySQL

Source code:

-- phpMyAdmin SQL Dump

-- version 5.0.1

-- https://www.phpmyadmin.net/

--

-- Máy chủ: 127.0.0.1

-- Thời gian đã tạo: Th10 10, 2020 lúc 03:35 AM

-- Phiên bản máy phục vụ: 10.4.11-MariaDB

-- Phiên bản PHP: 7.4.3

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Cơ sở dữ liệu: `book\_management`

--

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `books`

--

CREATE TABLE `books` (

  `bookId` bigint(20) UNSIGNED NOT NULL,

  `name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `description` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `originalPrice` double NOT NULL,

  `price` double NOT NULL,

  `image` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `quantity` int(11) NOT NULL,

  `ratings` int(11) NOT NULL,

  `status` tinyint(1) NOT NULL DEFAULT 0 COMMENT '0=>hết hàng,  1=>hiện có',

  `category\_id` bigint(20) UNSIGNED NOT NULL,

  `created\_at` timestamp NULL DEFAULT NULL,

  `updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Đang đổ dữ liệu cho bảng `books`

--

INSERT INTO `books` (`bookId`, `name`, `description`, `originalPrice`, `price`, `image`, `quantity`, `ratings`, `status`, `category\_id`, `created\_at`, `updated\_at`) VALUES

(3, 'BĂNG ĐẢNG NỬA ĐÊM', 'Trong một trận thi đấu ở trường, Tom bị quả bóng cricket va vào đầu mạnh đến nỗi bất tỉnh. Khi tỉnh lại, cậu thấy mình đang ở trong bệnh viện Công tước Funt, và trước mặt cậu là một ông đẩy xe có diện mạo kỳ quái đáng sợ. Mọi chuyện dần trở nên tồi tệ hơn khi Tom phải chịu sự quản lý của y tá trưởng phụ trách phòng bệnh nhi - một người cực kì thích hành hạ trẻ con. Nhưng tất cả những điều đó không ngăn được cậu cùng với những người bạn cùng phòng bệnh của mình tham gia vào những chuyến phiêu lưu kỳ thú nhất đời người. Cùng nhau họ tạo nên Băng đảng Nửa đêm, với sứ mệnh mỗi đêm lại biến một ước mơ tưởng chừng hoang đường của một cô bé hay cậu bé trong phòng bệnh thành hiện thực.', 135000, 108000, 'http://static.nhanam.com.vn/thumb/0x320/crop/Books/Images/2020/7/30/4BK8T37Z.jpg', 10, 5, 1, 1, '2020-11-07 04:59:38', '2020-11-07 04:59:38');

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `categories`

--

CREATE TABLE `categories` (

  `id` bigint(20) UNSIGNED NOT NULL,

  `name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `description` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `status` tinyint(1) NOT NULL DEFAULT 0 COMMENT '0=>chưa có,  1=>hiện có',

  `user\_id` bigint(20) UNSIGNED NOT NULL,

  `created\_at` timestamp NULL DEFAULT NULL,

  `updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Đang đổ dữ liệu cho bảng `categories`

--

INSERT INTO `categories` (`id`, `name`, `description`, `status`, `user\_id`, `created\_at`, `updated\_at`) VALUES

(1, 'Thiếu nhi', 'Không chỉ chú trọng về nội dung truyền tải, từng cuốn sách dành cho các em nhỏ còn được quan tâm về hình thức bên ngoài bắt mắt, sinh động, phù hợp với từng lứa tuổi để tăng phần hứng thú đọc sách của các em nhỏ.', 1, 1, NULL, '2020-10-28 22:32:39'),

(3, 'Tâm lý, tâm linh, tôn giáo', 'Tâm linh là một khái niệm rộng, thay đổi theo hoàn cảnh, với nhiều sắc thái có thể cùng tồn tại. \nTâm lý học là ngành khoa học nghiên cứu về tâm trí và hành vi, tìm hiểu về các hiện tượng ý thức và vô thức, cũng như cảm xúc và tư duy. Đây là một bộ môn học thuật với quy mô nghiên cứu rất rộng.\nTôn giáo hay giáo phái có thể được định nghĩa là một hệ thống các văn hoá, tính ngưỡng, đức tin bao gồm các hành vi và hành động được chỉ định cụ thể, các quan niệm về thế giới, thể hiện thông qua các kinh sách, khải thị, các địa điểm linh thiêng, lời tiên tri, quan niệm đạo đức, hoặc tổ chức, liên quan đến nhân loại với các yếu tố siêu nhiên, siêu việt hoặc tâm linh.', 1, 1, '2020-10-13 22:17:38', '2020-10-28 22:32:26'),

(4, 'Truyện, tiểu thuyết', 'Tiểu thuyết là một thể loại văn xuôi có hư cấu, thông qua nhân vật, hoàn cảnh, ... Đại thuyết là kinh sách của các thánh nhân viết như Kinh Thư, Kinh Thi của ... ở Việt Nam, khi nói đến tiểu thuyết, độc giả thường hiểu đó là tác phẩm truyện dài.', 1, 1, '2020-10-25 04:46:16', '2020-10-28 22:32:18'),

(5, 'Giáo trình', 'Giáo trình là hệ thống chương trình giảng dạy của một môn học. Nó là tài liệu học tập hoặc giảng dạy được thiết kế và biên soạn dựa trên cơ sở chương trình môn học. Mục đích để làm tài liệu giảng dạy chính thức cho giáo viên, hoặc / và làm tài liệu học tập chính thức cho học sinh, sinh viên.', 1, 1, '2020-10-25 08:00:17', '2020-10-28 22:32:09'),

(6, 'Văn hóa xã hội – Lịch sử', 'Từ ngàn năm nay, văn hóa và xã hội được coi là hai thành phần quan trọng góp phần làm nên sự giàu mạnh của đất nước. Có thể thấy văn hóa – xã hội giúp đất nước đa dạng, giàu đẹp về bản sắc, chính trị  – quân sự giúp xã hội cân bằng, ổn định, còn kinh tế đưa đất nước phát triển tiến bước với toàn cầu.', 1, 1, '2020-10-25 20:11:08', '2020-10-28 22:31:58'),

(7, 'Văn học nghệ thuật', 'Văn học nghệ thuật không bao giờ là một tổng cộng các văn bản vô định hình-đồng chất: nó không chỉ là sự tổ chức, mà còn là một cơ chế tự tổ chức. Trên trình độ cao hơn về tổ chức văn học, nó phân xuất ra một nhóm văn bản trừu tượng hơn so với cái khối văn bản còn lại, cái cấp độ, tức là các siêu văn bản.', 1, 1, '2020-10-25 20:18:23', '2020-10-28 22:31:52'),

(9, 'Khoa học công nghệ – Kinh tế', 'Trong thời kì kinh tế hội nhập ngày nay, con người được hưởng một cuộc sống đầy đủ và chất lượng. Để có được những thành tựu to lớn này là công sức và thành tựu của ngành khoa học – công nghệ.', 1, 1, '2020-10-25 20:28:06', '2020-10-28 22:31:35'),

(11, 'Chính trị – pháp luật', 'Những cuốn sách hay về chính trị, pháp luật và những thăng trầm trong lịch sử phát triển của các quốc gia trên thế giới.', 1, 1, '2020-10-25 20:28:38', '2020-10-28 22:10:56');

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `failed\_jobs`

--

CREATE TABLE `failed\_jobs` (

  `id` bigint(20) UNSIGNED NOT NULL,

  `uuid` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `connection` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `queue` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `payload` longtext COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `exception` longtext COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `failed\_at` timestamp NOT NULL DEFAULT current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `migrations`

--

CREATE TABLE `migrations` (

  `id` int(10) UNSIGNED NOT NULL,

  `migration` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `batch` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Đang đổ dữ liệu cho bảng `migrations`

--

INSERT INTO `migrations` (`id`, `migration`, `batch`) VALUES

(1, '2014\_10\_12\_000000\_create\_users\_table', 1),

(2, '2014\_10\_12\_100000\_create\_password\_resets\_table', 1),

(3, '2019\_08\_19\_000000\_create\_failed\_jobs\_table', 1),

(4, '2020\_10\_13\_093051\_create\_categories\_table', 1),

(5, '2020\_10\_13\_093143\_create\_books\_table', 1),

(6, '2016\_06\_01\_000001\_create\_oauth\_auth\_codes\_table', 2),

(7, '2016\_06\_01\_000002\_create\_oauth\_access\_tokens\_table', 2),

(8, '2016\_06\_01\_000003\_create\_oauth\_refresh\_tokens\_table', 2),

(9, '2016\_06\_01\_000004\_create\_oauth\_clients\_table', 2),

(10, '2016\_06\_01\_000005\_create\_oauth\_personal\_access\_clients\_table', 2),

(11, '2020\_11\_06\_064851\_create\_books', 3),

(12, '2020\_11\_06\_065052\_create\_books\_table', 4),

(13, '2020\_11\_06\_071432\_create\_books\_table', 5),

(14, '2020\_11\_06\_092035\_create\_shopping\_cart\_table', 6),

(15, '2020\_11\_07\_111307\_create\_books\_table', 7),

(16, '2020\_11\_07\_115019\_create\_shopping\_cart\_table', 8);

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `oauth\_access\_tokens`

--

CREATE TABLE `oauth\_access\_tokens` (

  `id` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `user\_id` bigint(20) UNSIGNED DEFAULT NULL,

  `client\_id` bigint(20) UNSIGNED NOT NULL,

  `name` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

  `scopes` text COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

  `revoked` tinyint(1) NOT NULL,

  `created\_at` timestamp NULL DEFAULT NULL,

  `updated\_at` timestamp NULL DEFAULT NULL,

  `expires\_at` datetime DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Đang đổ dữ liệu cho bảng `oauth\_access\_tokens`

--

INSERT INTO `oauth\_access\_tokens` (`id`, `user\_id`, `client\_id`, `name`, `scopes`, `revoked`, `created\_at`, `updated\_at`, `expires\_at`) VALUES

('0e51d28924e5bcd955e1db312fa0aed11a4d5065351a31eb62a2b0bd1695c9c7ea409325f9c2e989', 6, 3, 'authToken', '[]', 0, '2020-11-02 22:03:47', '2020-11-02 22:03:47', '2021-05-03 05:03:46'),

('20ddaac7024983e24bafb14c4e852cb2216ae15e1ebc42f6feba5453d7d96d07be4f5b9cb4223aa0', 1, 3, 'Token Name', '[]', 0, '2020-10-31 03:05:27', '2020-10-31 03:05:27', '2021-05-01 10:05:24'),

('2d62763a9a19bcb1cdbe4ec2be9744d602a4ed31fa0faf3e762264ea5bdf363093e03782fec88710', 4, 3, 'authToken', '[]', 0, '2020-11-01 01:33:00', '2020-11-01 01:33:00', '2021-05-01 08:32:58'),

('3a3b1029dda46deb7a4bec58345c50a458fe3332f8ea8860d41e57b917faa3de02744e8f7490028a', 1, 3, 'authToken', '[]', 0, '2020-11-05 23:56:47', '2020-11-05 23:56:47', '2021-05-06 06:56:47'),

('4a0007c7326aff0111107bf126f5d2ae100a4276896209b9894cbe25e8130d32c2ef8e0eac303b4b', 2, 3, 'authToken', '[]', 0, '2020-10-31 10:17:41', '2020-10-31 10:17:41', '2021-05-01 17:17:41'),

('4bb552f9fe5c8ba46384ab8ac85734ca1dffc558401326bbd0b5048d2bb9b3f075737fb54819d6f5', 8, 3, 'authToken', '[]', 0, '2020-11-03 00:31:08', '2020-11-03 00:31:08', '2021-05-03 07:31:07'),

('4cf5df12730c4c71c060fd17870ac94ea7d8823a7a12d1797913e42aeb789a2f687be3c392c0e912', 1, 3, 'authToken', '[]', 0, '2020-11-03 00:21:07', '2020-11-03 00:21:07', '2021-05-03 07:21:07'),

('523ed10d2e1af4274283898f5a175d82eb7860a4600c7ecf4795487f0421e6148a37cc7754a455f2', NULL, 3, 'authToken', '[]', 0, '2020-11-07 04:05:52', '2020-11-07 04:05:52', '2021-05-07 11:05:51'),

('53f94fceedd928a045df057d26ee84cc73b14f9617cb08d1827181b767156703b89c232504cd8579', NULL, 3, 'authToken', '[]', 0, '2020-11-07 04:06:31', '2020-11-07 04:06:31', '2021-05-07 11:06:31'),

('597c183ce3a416169a3ee14b7e8c28a4edb68ad3dc8e243801bd215637830ce44dc25b976ce068ca', 1, 3, 'authToken', '[]', 0, '2020-11-02 23:03:23', '2020-11-02 23:03:23', '2021-05-03 06:03:23'),

('7ccd5b969e268df379e8db414143ee5a8ab3e7dfc03c01a7d5d3274e867398b8b4e0224f80b4d82e', 1, 3, 'authToken', '[]', 0, '2020-11-04 23:26:06', '2020-11-04 23:26:06', '2021-05-05 06:26:04'),

('86ee8730eb043810152f2f6cfff5c5fb2ed84857a2265888c9caf02a87ec9070814a3552f75442ea', NULL, 3, 'authToken', '[]', 0, '2020-11-07 04:20:47', '2020-11-07 04:20:47', '2021-05-07 11:20:47'),

('8fda2c227ffdb55c3899d36db3b9acfb02378068b73ebf8bc6affd32382ca5dc46491dd68d47181a', 1, 3, 'authToken', '[]', 0, '2020-11-02 23:05:01', '2020-11-02 23:05:01', '2021-05-03 06:05:01'),

('93efb170f25dfaebae520f18fb2462de9fa3f9c4e202539e4fa41db1571a139226bc676f52cdc9f6', 5, 3, 'authToken', '[]', 0, '2020-11-01 01:41:59', '2020-11-01 01:41:59', '2021-05-01 08:41:59'),

('9cc53729c8351e615be53bfae36cad578319210cf564c0c7b77920c35e5d83989734e3e73ba471a3', 3, 3, 'authToken', '[]', 0, '2020-11-01 00:09:55', '2020-11-01 00:09:55', '2021-05-01 07:09:54'),

('a6fd21f484ac0f5c072ba28a8693f75cc861b485e83389a3d60daf0f1f233ac7db0017963b9ca3c1', 1, 3, 'Token Name', '[]', 0, '2020-10-30 02:02:35', '2020-10-30 02:02:35', '2021-10-30 09:02:35'),

('a8451ef4550be2ee146fa2281f851d517683354060071fd09f242e7fa7e1c5d9c30f8df292fb9a57', 7, 3, 'authToken', '[]', 0, '2020-11-02 22:19:07', '2020-11-02 22:19:07', '2021-05-03 05:19:07'),

('a9bc381d94d8209702cbdadbff56f0de3e3929e10d94de2696b672533d272577523c743810ffe109', 8, 3, 'authToken', '[]', 0, '2020-11-03 00:31:28', '2020-11-03 00:31:28', '2021-05-03 07:31:27'),

('add576092b81a75447930c6e8aed7ad4e6425cb773e703cc5ac855028f79ed297a516836b8887025', 8, 3, 'authToken', '[]', 0, '2020-11-03 00:31:46', '2020-11-03 00:31:46', '2021-05-03 07:31:46'),

('c8a3abd4e8fa49ea44431e0f0e950926bea1dcba8b0b20e8451c1b9d356e6f13f7ad53b7d4407620', 1, 3, 'Token Name', '[]', 0, '2020-10-31 10:14:44', '2020-10-31 10:14:44', '2021-05-01 17:14:44'),

('c9d0b60f5a311c99d9120ee48a75b6dada193a19a299a896720d43ac7b5282d5e0e308ba1ae2a1c9', 1, 3, 'authToken', '[]', 0, '2020-11-05 21:25:55', '2020-11-05 21:25:55', '2021-05-06 04:25:50'),

('e35d06d952e08d67ee4f0a5288ab48aac0720a4ecef2bee7b74cd0f0b2ca8c8f7a176364bde8db03', 1, 3, 'authToken', '[]', 0, '2020-11-02 23:32:05', '2020-11-02 23:32:05', '2021-05-03 06:32:05'),

('ef11b0b84fd4c81d60f40f6771ca0eb8e5633d995a2dc1a4a1524b46d6f11af4a9cb0daa532b0057', 1, 3, 'authToken', '[]', 0, '2020-10-31 09:54:47', '2020-10-31 09:54:47', '2021-05-01 16:54:46'),

('f6531c2fb6da5c2f9a9397f4d62b84052afaa6eb563f492cf45cf4d9e3eb81cac580c6caa95f1248', 1, 3, 'authToken', '[]', 0, '2020-11-02 23:42:03', '2020-11-02 23:42:03', '2021-05-03 06:42:03');

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `oauth\_auth\_codes`

--

CREATE TABLE `oauth\_auth\_codes` (

  `id` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `user\_id` bigint(20) UNSIGNED NOT NULL,

  `client\_id` bigint(20) UNSIGNED NOT NULL,

  `scopes` text COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

  `revoked` tinyint(1) NOT NULL,

  `expires\_at` datetime DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `oauth\_clients`

--

CREATE TABLE `oauth\_clients` (

  `id` bigint(20) UNSIGNED NOT NULL,

  `user\_id` bigint(20) UNSIGNED DEFAULT NULL,

  `name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `secret` varchar(100) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

  `provider` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

  `redirect` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `personal\_access\_client` tinyint(1) NOT NULL,

  `password\_client` tinyint(1) NOT NULL,

  `revoked` tinyint(1) NOT NULL,

  `created\_at` timestamp NULL DEFAULT NULL,

  `updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Đang đổ dữ liệu cho bảng `oauth\_clients`

--

INSERT INTO `oauth\_clients` (`id`, `user\_id`, `name`, `secret`, `provider`, `redirect`, `personal\_access\_client`, `password\_client`, `revoked`, `created\_at`, `updated\_at`) VALUES

(1, NULL, 'Laravel Personal Access Client', 'c9ZM0KeEGEjSh8k7Lig0R5bwWhKNeccQWPSBghox', NULL, 'http://localhost', 1, 0, 0, '2020-10-30 01:17:56', '2020-10-30 01:17:56'),

(2, NULL, 'Laravel Password Grant Client', 'GnNH716QY7256RSTB2vpyfqBMsEvc9cBDzbJuqTZ', 'users', 'http://localhost', 0, 1, 0, '2020-10-30 01:17:56', '2020-10-30 01:17:56'),

(3, NULL, 'authToken', 'k3VxraqgVM2wgjMMQkXx9nh2ZKxMSbl6pj050Hqa', NULL, 'http://localhost', 1, 0, 0, '2020-10-30 01:32:13', '2020-10-30 01:32:13');

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `oauth\_personal\_access\_clients`

--

CREATE TABLE `oauth\_personal\_access\_clients` (

  `id` bigint(20) UNSIGNED NOT NULL,

  `client\_id` bigint(20) UNSIGNED NOT NULL,

  `created\_at` timestamp NULL DEFAULT NULL,

  `updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Đang đổ dữ liệu cho bảng `oauth\_personal\_access\_clients`

--

INSERT INTO `oauth\_personal\_access\_clients` (`id`, `client\_id`, `created\_at`, `updated\_at`) VALUES

(1, 1, '2020-10-30 01:17:56', '2020-10-30 01:17:56'),

(2, 3, '2020-10-30 01:32:13', '2020-10-30 01:32:13');

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `oauth\_refresh\_tokens`

--

CREATE TABLE `oauth\_refresh\_tokens` (

  `id` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `access\_token\_id` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `revoked` tinyint(1) NOT NULL,

  `expires\_at` datetime DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `password\_resets`

--

CREATE TABLE `password\_resets` (

  `email` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `token` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `created\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Cấu trúc bảng cho bảng `users`

--

CREATE TABLE `users` (

  `id` bigint(20) UNSIGNED NOT NULL,

  `name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `email` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `email\_verified\_at` timestamp NULL DEFAULT NULL,

  `password` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

  `remember\_token` varchar(100) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

  `created\_at` timestamp NULL DEFAULT NULL,

  `updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Đang đổ dữ liệu cho bảng `users`

--

INSERT INTO `users` (`id`, `name`, `email`, `email\_verified\_at`, `password`, `remember\_token`, `created\_at`, `updated\_at`) VALUES

(1, 'Thuan', 'vongocthuan1808@gmail.com', NULL, '$2y$10$jv4jJ6M8crOuC7Nq2mK40./7Dxh8iGKhgqSHaBlVkp4bkRiKZp2Ui', NULL, '2020-10-13 03:31:02', '2020-10-13 03:31:02'),

(8, 'Marco Reus', 'reus@gmail.com', NULL, '$2y$10$ZOuMvjsEYiu/coNcqb1AIuMe6h02Xh.ZQOQUZk3S8aU0K26RgzJvq', NULL, '2020-11-03 00:31:08', '2020-11-03 00:31:08');

--

-- Chỉ mục cho các bảng đã đổ

--

--

-- Chỉ mục cho bảng `books`

--

ALTER TABLE `books`

  ADD PRIMARY KEY (`bookId`),

  ADD KEY `books\_category\_id\_foreign` (`category\_id`);

--

-- Chỉ mục cho bảng `categories`

--

ALTER TABLE `categories`

  ADD PRIMARY KEY (`id`),

  ADD KEY `categories\_user\_id\_foreign` (`user\_id`);

--

-- Chỉ mục cho bảng `failed\_jobs`

--

ALTER TABLE `failed\_jobs`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `failed\_jobs\_uuid\_unique` (`uuid`);

--

-- Chỉ mục cho bảng `migrations`

--

ALTER TABLE `migrations`

  ADD PRIMARY KEY (`id`);

--

-- Chỉ mục cho bảng `oauth\_access\_tokens`

--

ALTER TABLE `oauth\_access\_tokens`

  ADD PRIMARY KEY (`id`),

  ADD KEY `oauth\_access\_tokens\_user\_id\_index` (`user\_id`);

--

-- Chỉ mục cho bảng `oauth\_auth\_codes`

--

ALTER TABLE `oauth\_auth\_codes`

  ADD PRIMARY KEY (`id`),

  ADD KEY `oauth\_auth\_codes\_user\_id\_index` (`user\_id`);

--

-- Chỉ mục cho bảng `oauth\_clients`

--

ALTER TABLE `oauth\_clients`

  ADD PRIMARY KEY (`id`),

  ADD KEY `oauth\_clients\_user\_id\_index` (`user\_id`);

--

-- Chỉ mục cho bảng `oauth\_personal\_access\_clients`

--

ALTER TABLE `oauth\_personal\_access\_clients`

  ADD PRIMARY KEY (`id`);

--

-- Chỉ mục cho bảng `oauth\_refresh\_tokens`

--

ALTER TABLE `oauth\_refresh\_tokens`

  ADD PRIMARY KEY (`id`),

  ADD KEY `oauth\_refresh\_tokens\_access\_token\_id\_index` (`access\_token\_id`);

--

-- Chỉ mục cho bảng `password\_resets`

--

ALTER TABLE `password\_resets`

  ADD KEY `password\_resets\_email\_index` (`email`);

--

-- Chỉ mục cho bảng `users`

--

ALTER TABLE `users`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `users\_email\_unique` (`email`);

--

-- AUTO\_INCREMENT cho các bảng đã đổ

--

--

-- AUTO\_INCREMENT cho bảng `books`

--

ALTER TABLE `books`

  MODIFY `bookId` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT cho bảng `categories`

--

ALTER TABLE `categories`

  MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=13;

--

-- AUTO\_INCREMENT cho bảng `failed\_jobs`

--

ALTER TABLE `failed\_jobs`

  MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT;

--

-- AUTO\_INCREMENT cho bảng `migrations`

--

ALTER TABLE `migrations`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=17;

--

-- AUTO\_INCREMENT cho bảng `oauth\_clients`

--

ALTER TABLE `oauth\_clients`

  MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT cho bảng `oauth\_personal\_access\_clients`

--

ALTER TABLE `oauth\_personal\_access\_clients`

  MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT cho bảng `users`

--

ALTER TABLE `users`

  MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

--

-- Các ràng buộc cho các bảng đã đổ

--

--

-- Các ràng buộc cho bảng `books`

--

ALTER TABLE `books`

  ADD CONSTRAINT `books\_category\_id\_foreign` FOREIGN KEY (`category\_id`) REFERENCES `categories` (`id`);

--

-- Các ràng buộc cho bảng `categories`

--

ALTER TABLE `categories`

  ADD CONSTRAINT `categories\_user\_id\_foreign` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`);

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

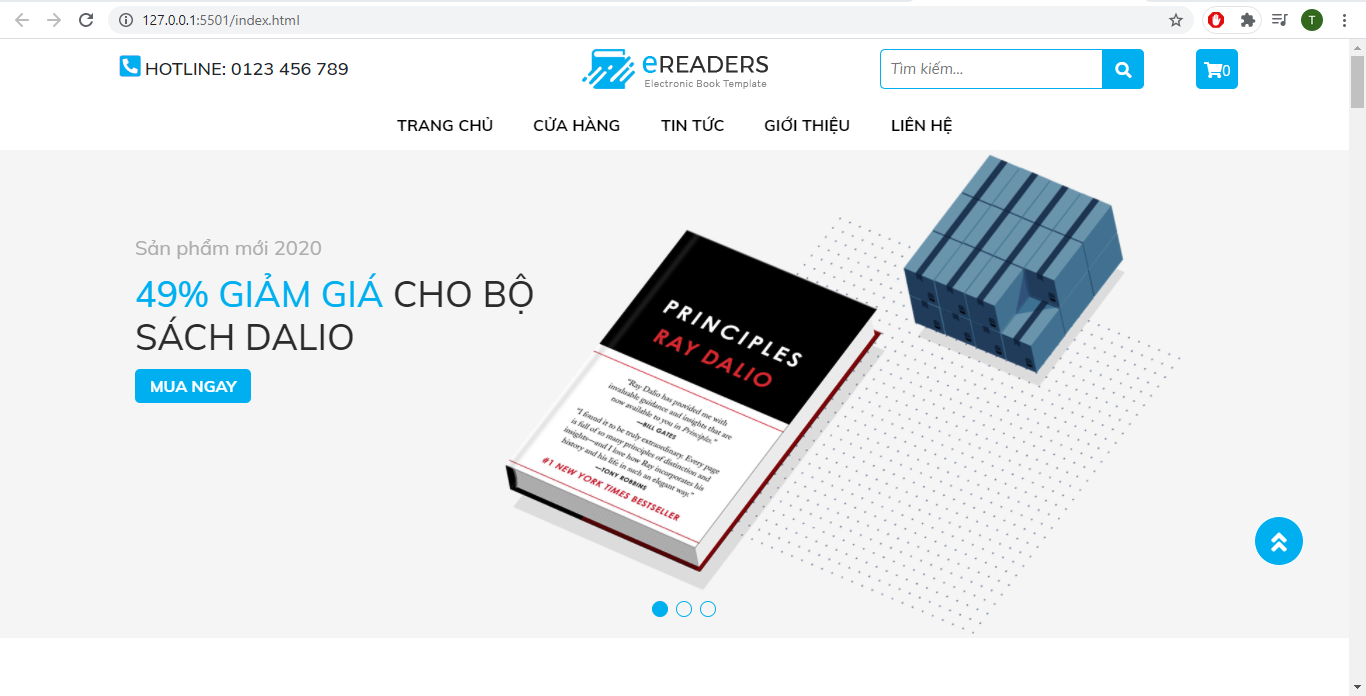
/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

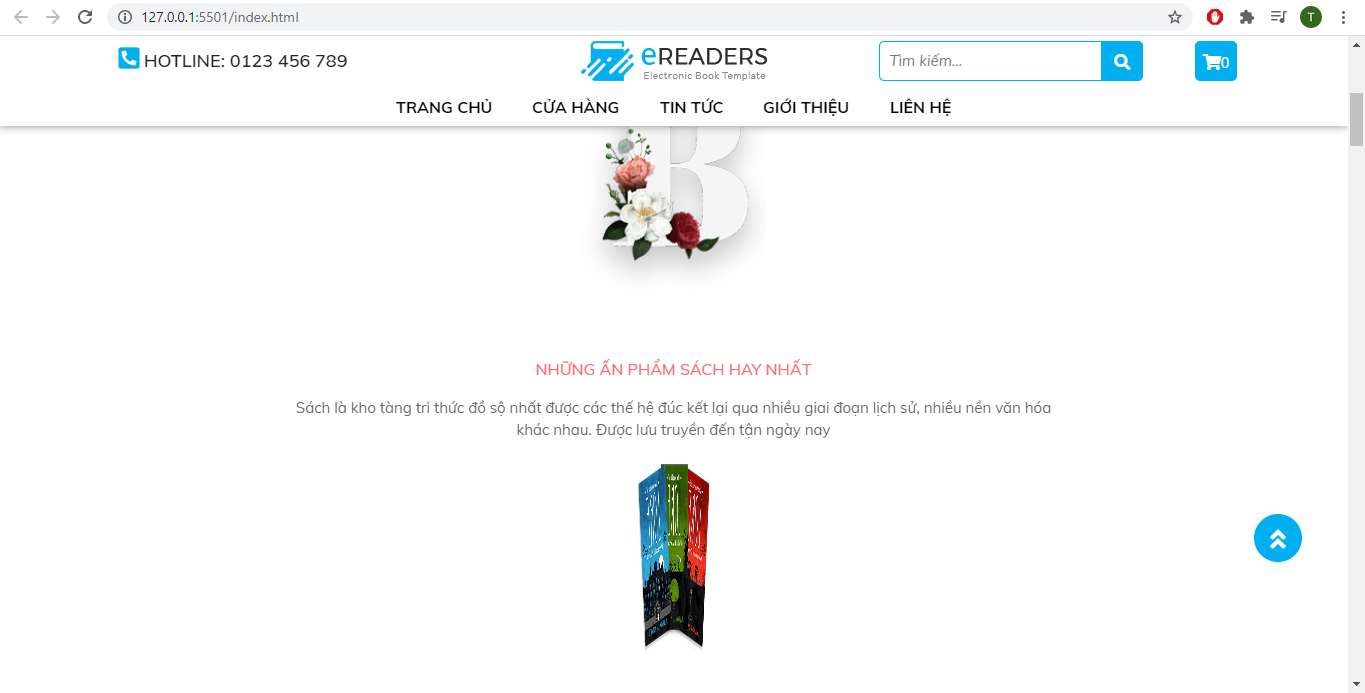
/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

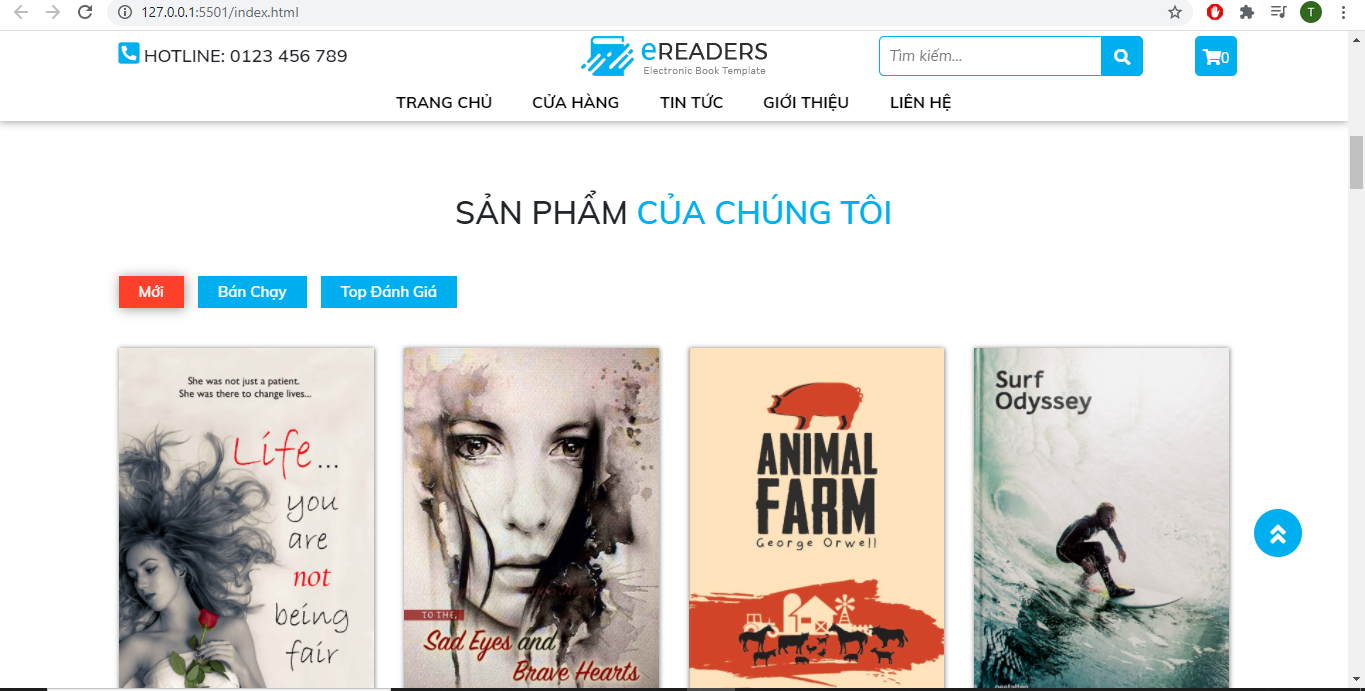
# Giao diện các màn hình chức năng chính

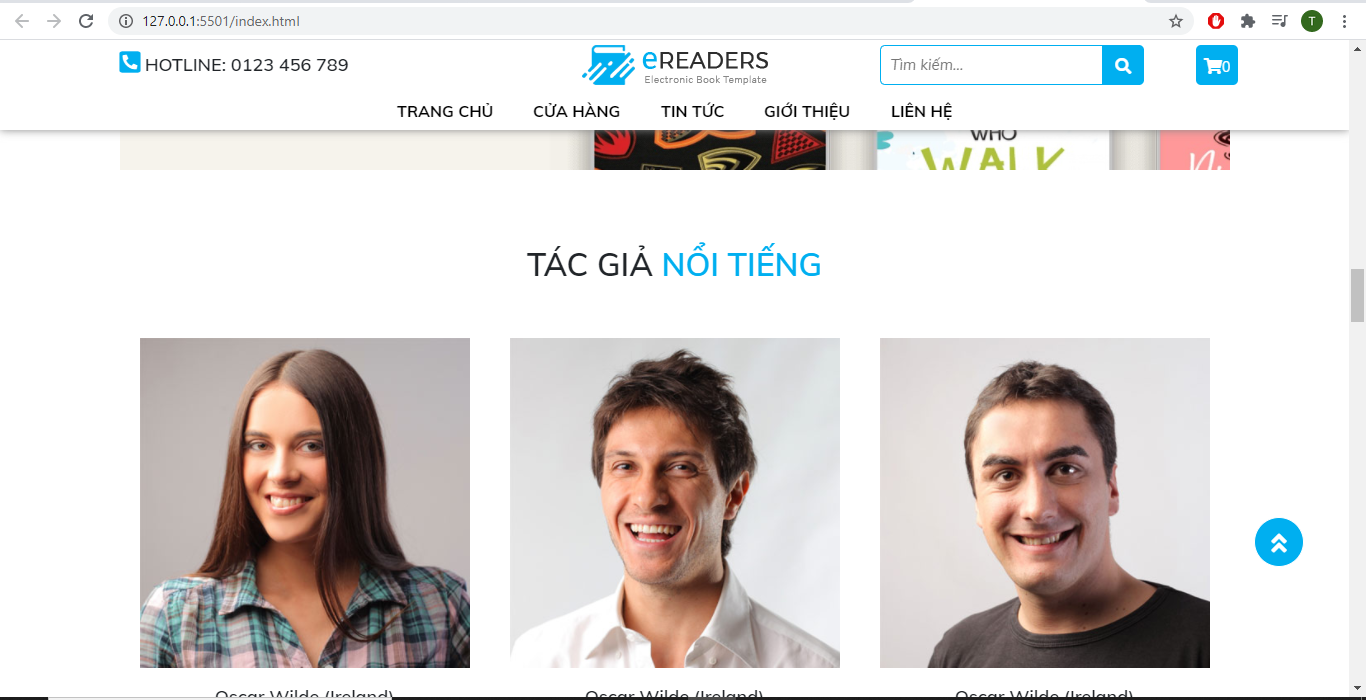
## Khách hàng

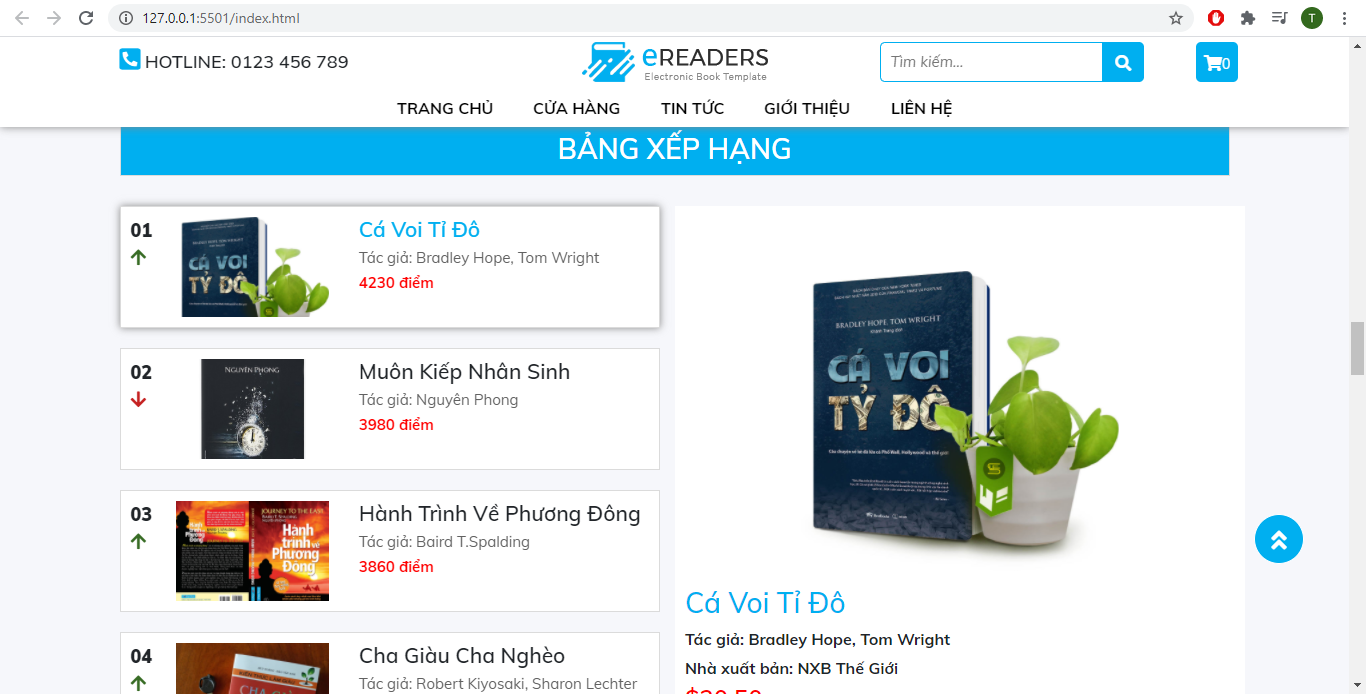
### Trang chủ

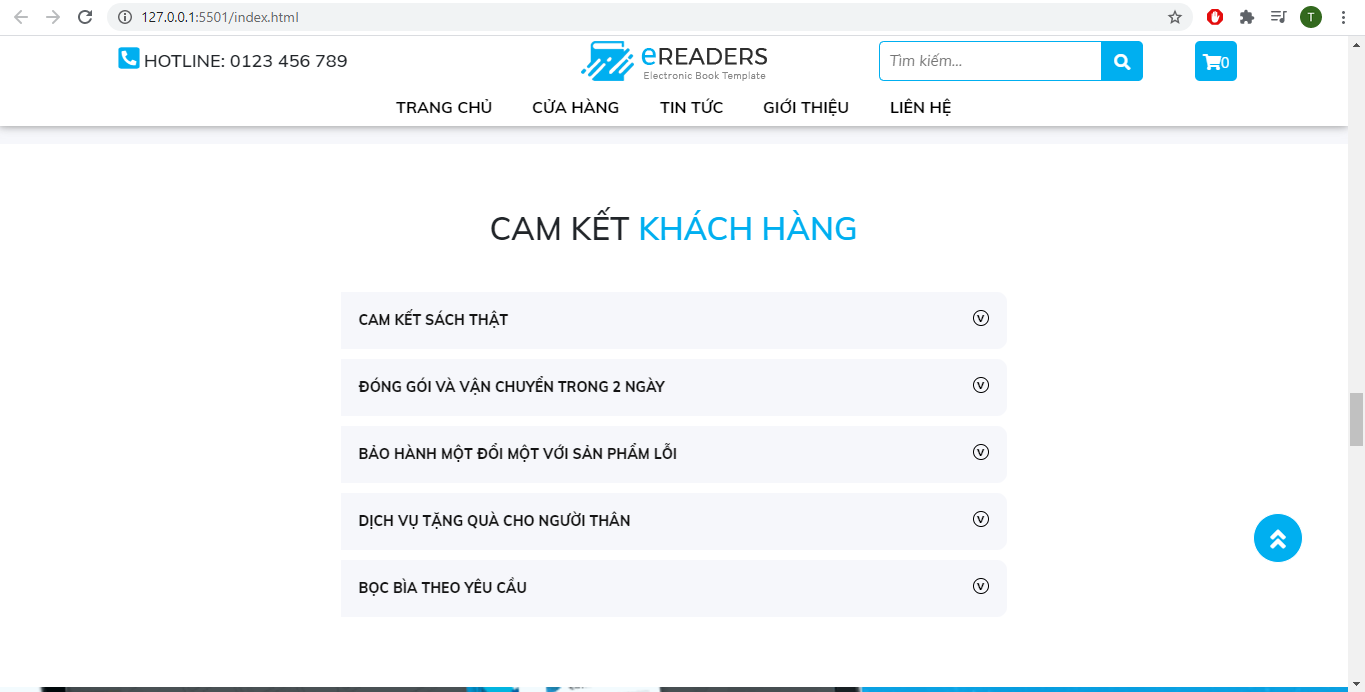


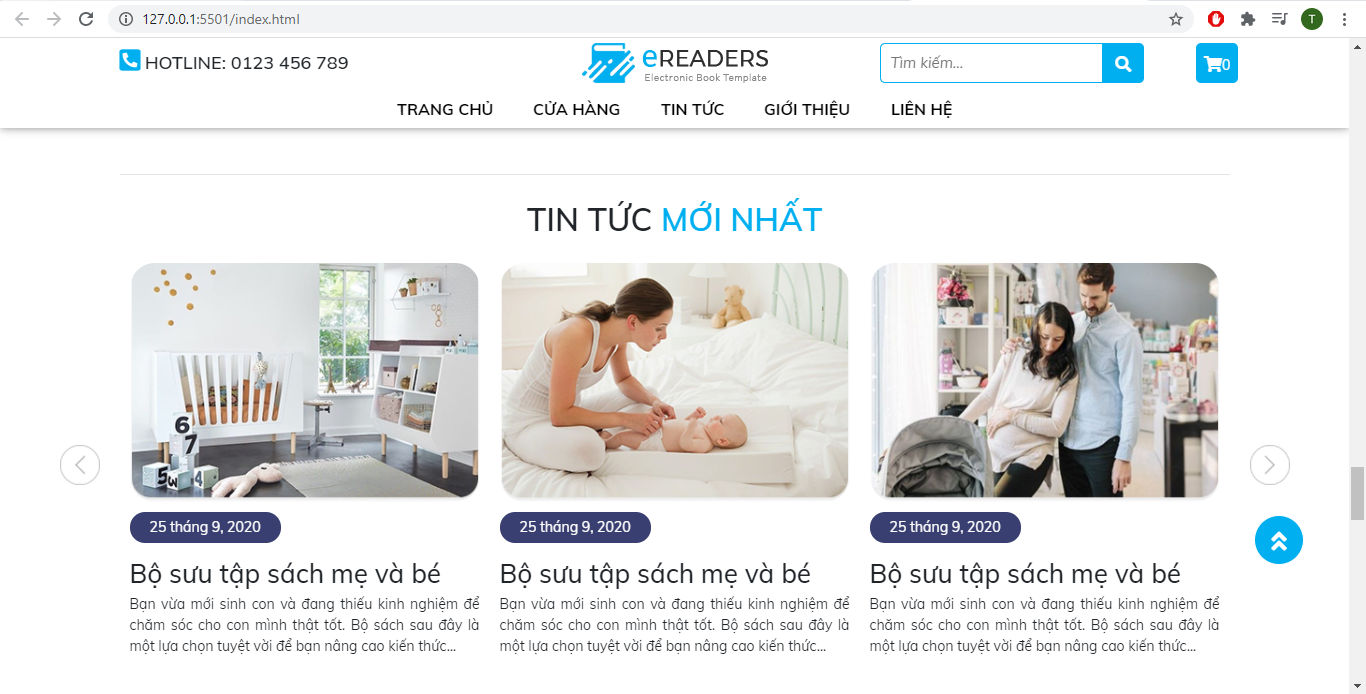


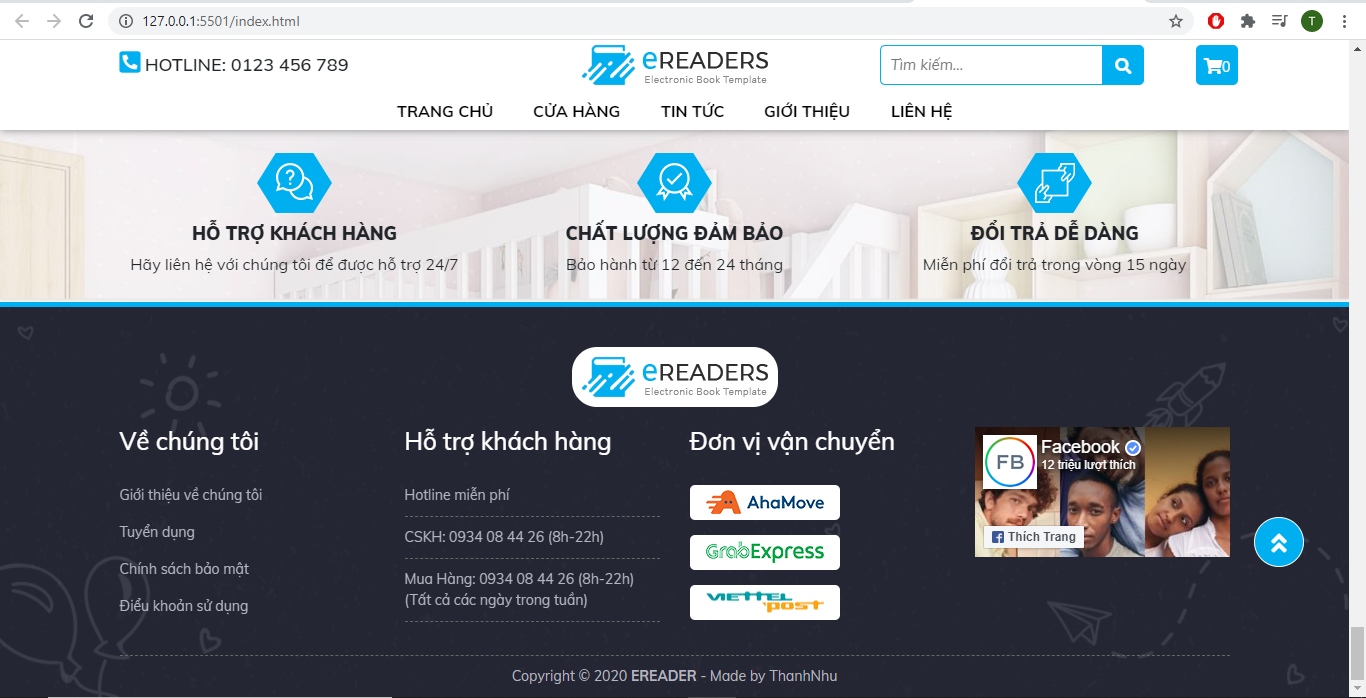






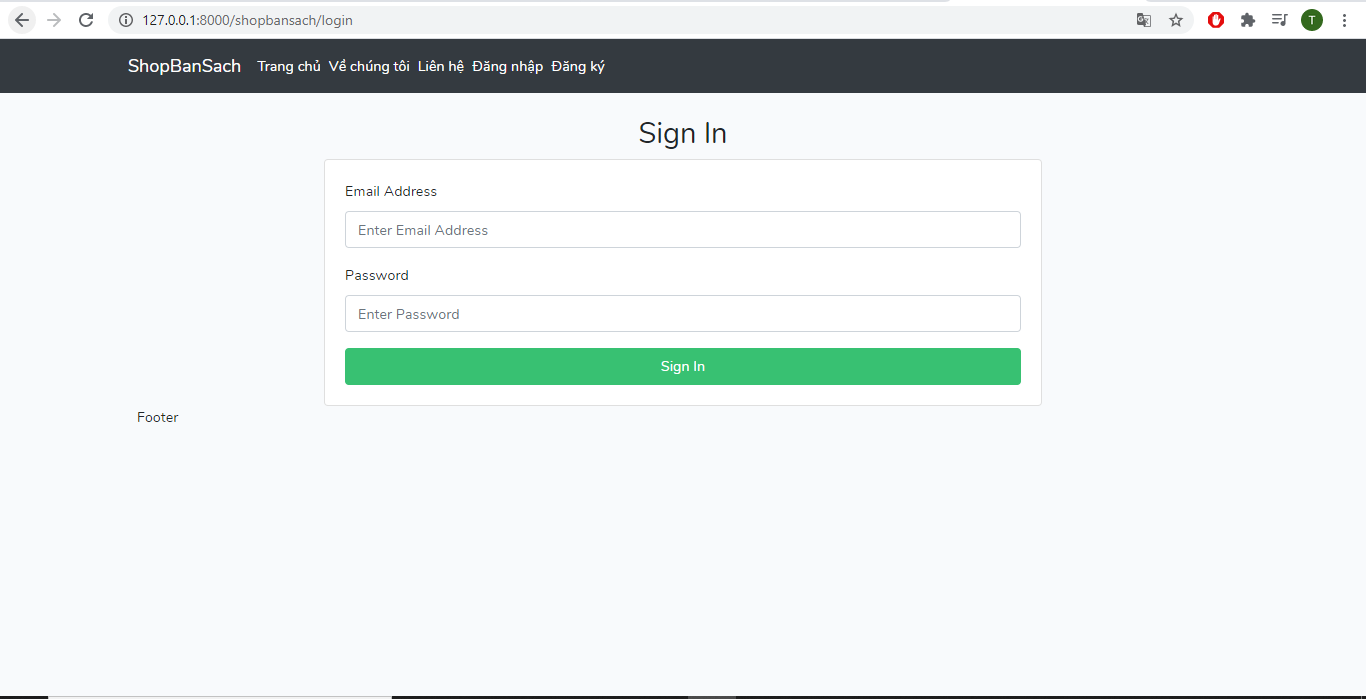




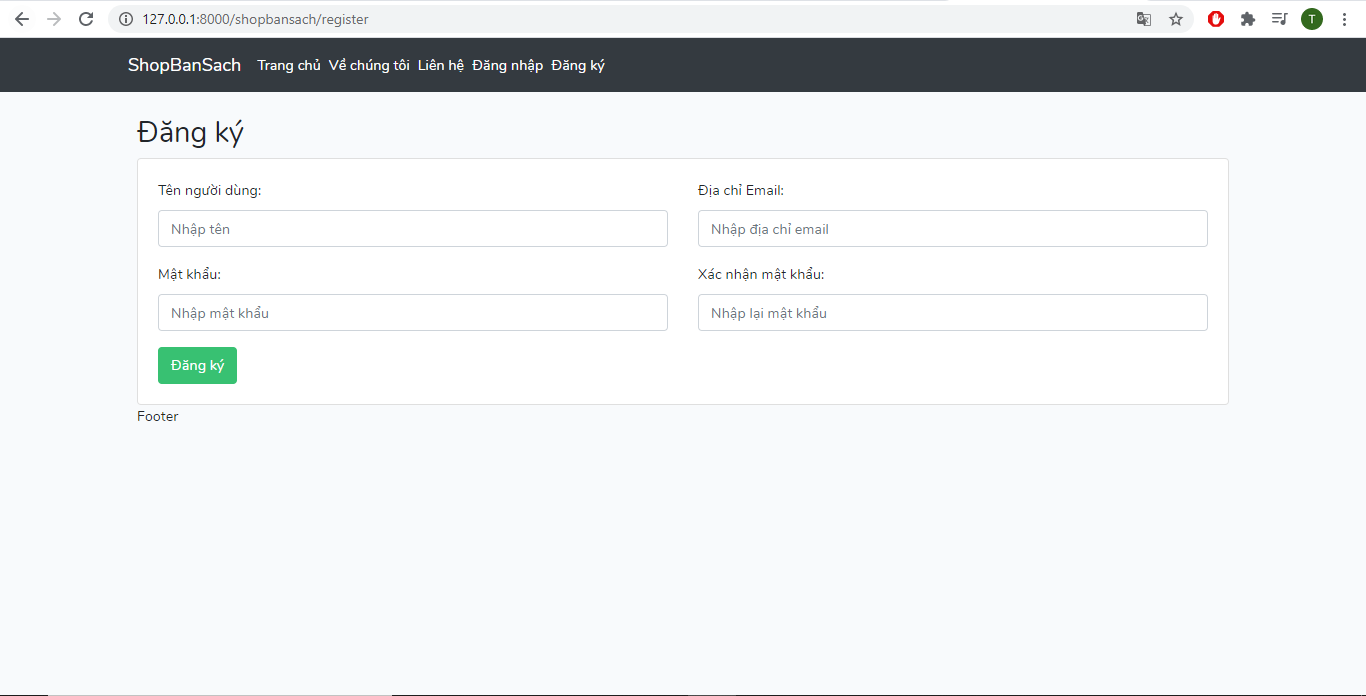


## Admin

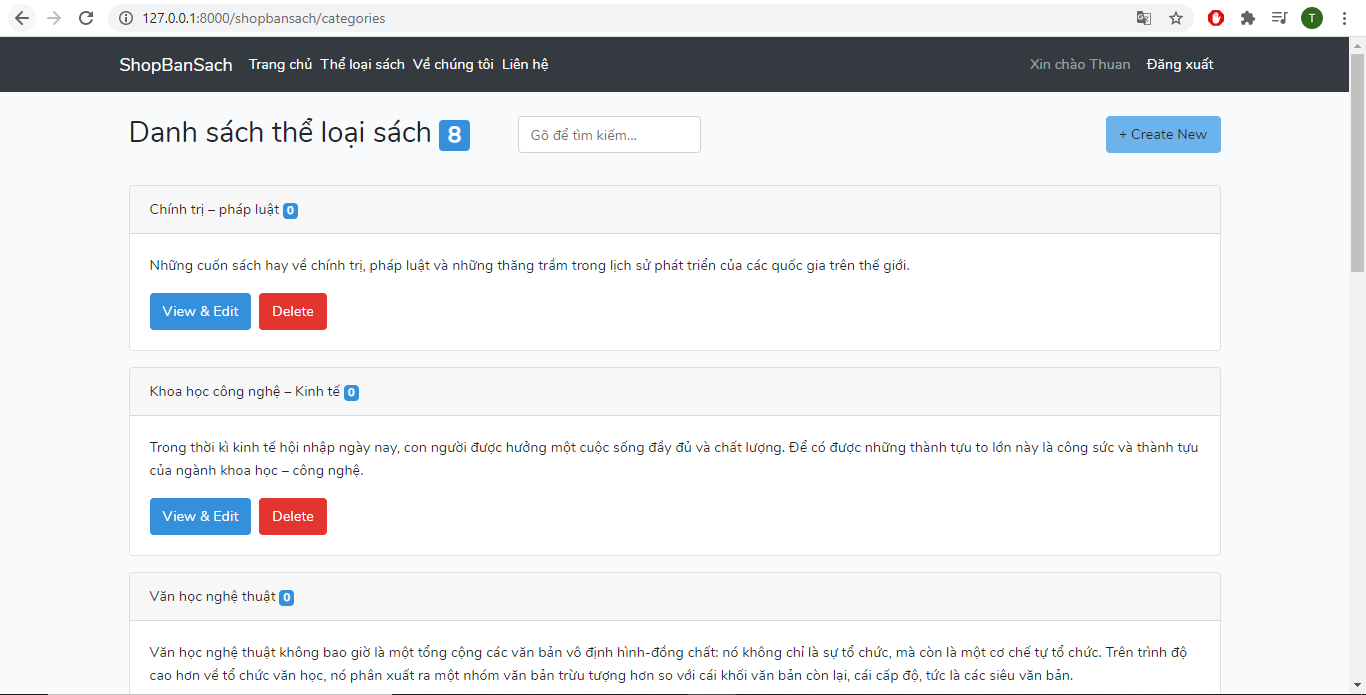
### Trang đăng nhập (tạm thời)



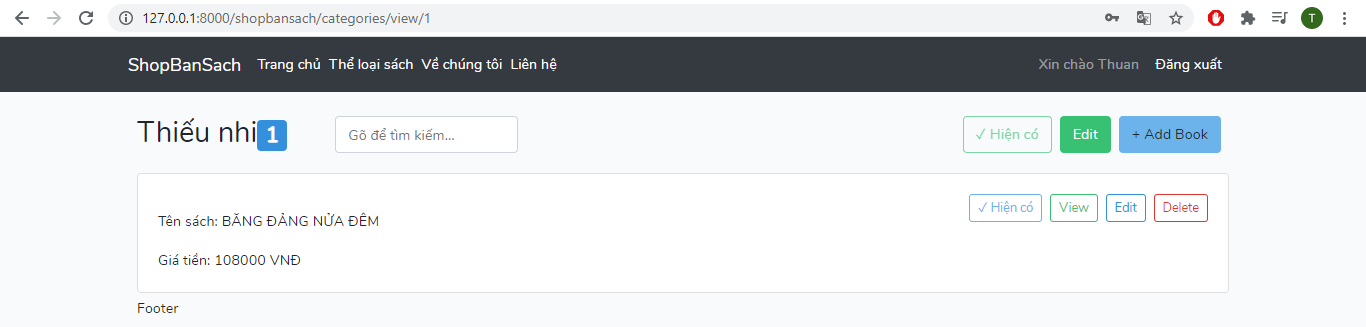
### Trang đăng ký (tạm thời)



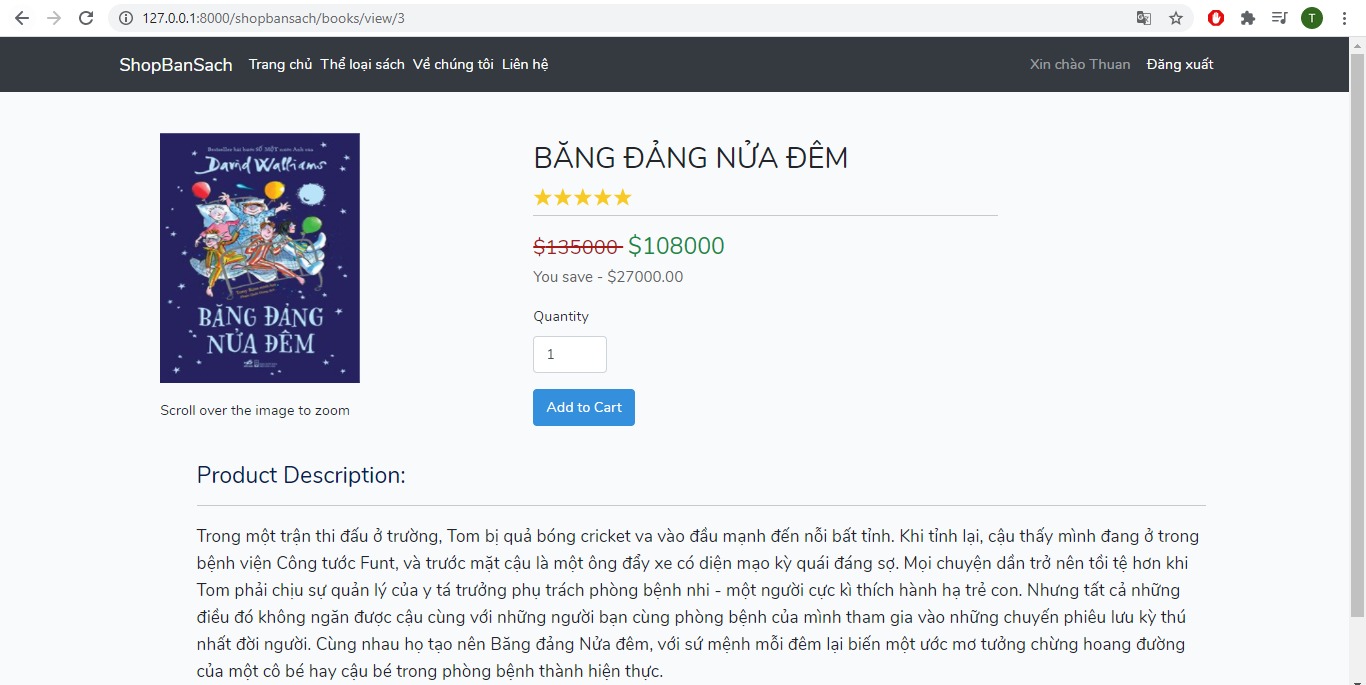
### Trang quản lý thể loại sách



### Trang quản lý sách của thể loại sách



### Trang xem thông tin chi tiết sách



# Source code vài chức năng chính

## Chức năng CRUD cho thể loại sách (table Categories)

### CategoryRepository

<?php

namespace App\repositories;

use App\interfaces\CrudInterface;

use App\Models\Category;

use Illuminate\Http\Request;

class CategoryRepository implements CrudInterface{

    public function getAll(){

        $categories = Category::withCount('books')->with('books')->orderBy('id','desc')->get();

        return $categories;

    }

    public function findById($id){

        $categories = Category::with('books')

            ->find($id);

        return $categories;

    }

    public function create(Request $request){

        $categories = new Category();

        $categories->name=$request->name;

        $categories->description=$request->description;

        $categories->status=$request->status;

        $categories->user\_id=$request->user\_id;

        $categories->save();

        return $categories;

    }

    public function edit(Request $request, $id){

        $categories = $this->findById($id);

        $categories->name=$request->name;

        $categories->description=$request->description;

        $categories->status=$request->status;

        $categories->user\_id=$request->user\_id;

        $categories->save();

        return $categories;

    }

    public function delete($id){

        $categories = $this->findById($id);

        $categories->books()->delete();

        $categories->delete();

        return $categories;

    }

}

### CategoryController

<?php

namespace App\Http\Controllers\API;

use App\Http\Controllers\Controller;

use App\repositories\CategoryRepository;

use Illuminate\Http\Request;

class CategoriesController extends Controller

{

    public $categoryRepository;

    public function \_\_construct(CategoryRepository $categoryRepository) {

        $this->categoryRepository = $categoryRepository;

    }

    public function index()

    {

        $categories = $this->categoryRepository->getAll();

        return response()->json([

            'success' => true,

            'message' => 'Category List',

            'data'    => $categories

        ]);

    }

    public function show($id)

    {

        $categories = $this->categoryRepository->findById($id);

        if(is\_null($categories))

        {

            return response()->json([

                'success' => false,

                'message' => 'Category Details',

                'data'    => null

            ]);

        }

        return response()->json([

            'success' => true,

            'message' => 'Category Details',

            'data'    => $categories

        ]);

    }

    public function store(Request $request)

    {

        $formData = $request->all();

        $validator = \Validator::make($formData, [

            'name' => 'required',

            'description' => 'required',

            'user\_id' => 'required'

        ], [

            'name.required' => 'Please give category name',

            'description.required' => 'Please give category description',

        ]);

        if ($validator->fails()) {

            return response()->json([

                'success' => false,

                'message' => $validator->getMessageBag()->first(),

                'errors' => $validator->getMessageBag(),

            ]);

        }

        $categories = $this->categoryRepository->create($request);

        return response()->json([

            'success' => true,

            'message' => 'Category Stored',

            'data'    => $categories

        ]);

    }

    public function update(Request $request, $id)

    {

        $categories = $this->categoryRepository->findById($id);

        if (is\_null($categories)) {

            return response()->json([

                'success' => false,

                'message' => 'Category Not found',

                'data' => null,

            ]);

        }

        $formData = $request->all();

        $validator = \Validator::make($formData, [

            'name' => 'required',

            'description' => 'required',

            'user\_id' => 'required'

        ], [

            'name.required' => 'Please give category name',

            'description.required' => 'Please give category description',

        ]);

        if ($validator->fails()) {

            return response()->json([

                'success' => false,

                'message' => $validator->getMessageBag()->first(),

                'errors' => $validator->getMessageBag(),

            ]);

        }

        $categories = $this->categoryRepository->edit($request, $id);

        return response()->json([

            'success' => true,

            'message' => 'Category Updated',

            'data'    => $categories

        ]);

    }

    public function destroy($id)

    {

        $categories = $this->categoryRepository->findById($id);

        if (is\_null($categories)) {

            return response()->json([

                'success' => false,

                'message' => 'Category Not found',

                'data' => null,

            ]);

        }

        $categories = $this->categoryRepository->delete($id);

        return response()->json([

            'success' => true,

            'message' => 'Category Deleted',

            'data'    => $categories

        ]);

    }

}

### CategoryModel

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Category extends Model

{

    public function books()

    {

        return $this->hasMany(Book::class);

    }

}

### Tạo API cho table Categories

import Axios from "axios";

export const getCategoryList = async () => {

  return await Axios.get('http://127.0.0.1:8000/api/categories').then(

    (res) => {

      return res.data;

    }

  );

};

export const storeNewCategory = async (data) => {

    data.user\_id= 1;

    return await Axios.post('http://127.0.0.1:8000/api/categories', data)

    .then((res) => {

      return res.data;

    });

}

export const updateCategory = async (id, data) => {

  data.user\_id= 1;

  return await Axios.put(`http://127.0.0.1:8000/api/categories/${id}`, data)

  .then((res) => {

    return res.data;

  });

}

export const deleteCategory = async (id) => {

  return await Axios.delete(`http://127.0.0.1:8000/api/categories/${id}`)

  .then((res) => {

    return res.data;

  });

}

## Chức năng CRUD cho sách (table Books)

### BookRepository

<?php

namespace App\repositories;

use App\interfaces\CrudInterface;

use App\Models\Book;

use Illuminate\Http\Request;

class BookRepository implements CrudInterface{

    public function getAll(){

        $books = Book::orderBy('bookId','desc')->get();

        return $books;

    }

    public function findById($id){

        $books = Book::with('category')

            ->find($id);

        return $books;

    }

    public function create(Request $request){

        $books = new Book();

        $books->name=$request->name;

        $books->description=$request->description;

        $books->originalPrice=$request->originalPrice;

        $books->price=$request->price;

        $books->quantity=$request->quantity;

        $books->ratings=$request->ratings;

        $books->image=$request->image;

        $books->status=$request->status;

        $books->category\_id=$request->category\_id;

        $books->save();

        return $books;

    }

    public function edit(Request $request, $id){

        $books = $this->findById($id);

        $books->name=$request->name;

        $books->description=$request->description;

        $books->originalPrice=$request->originalPrice;

        $books->price=$request->price;

        $books->quantity=$request->quantity;

        $books->ratings=$request->ratings;

        $books->image=$request->image;

        $books->status=$request->status;

        $books->category\_id=$request->category\_id;

        $books->save();

        return $books;

    }

    public function delete($id){

        $books = $this->findById($id);

        $books->delete();

        return $books;

    }

}

### BookController

<?php

namespace App\Http\Controllers\API;

use App\Http\Controllers\Controller;

use App\repositories\BookRepository;

use Illuminate\Http\Request;

class BooksController extends Controller

{

    public $bookRepository;

    public function \_\_construct(BookRepository $bookRepository) {

        $this->bookRepository = $bookRepository;

    }

    public function index()

    {

        $books = $this->bookRepository->getAll();

        return response()->json([

            'success' => true,

            'message' => 'Book List',

            'data'    => $books

        ]);

    }

    public function show($id)

    {

        $books = $this->bookRepository->findById($id);

        if(is\_null($books))

        {

            return response()->json([

                'success' => false,

                'message' => 'Book Details',

                'data'    => null

            ]);

        }

        return response()->json([

            'success' => true,

            'message' => 'Book Details',

            'data'    => $books

        ]);

    }

    public function store(Request $request)

    {

        $formData = $request->all();

        $validator = \Validator::make($formData, [

            'name' => 'required',

            'description' => 'required',

            'originalPrice' => 'required',

            'price' => 'required',

            'quantity' => 'required',

            'ratings' => 'required',

            'image' => 'required',

            'status' => 'required',

            'category\_id' => 'required'

        ], [

            'name.required' => 'Please give book name',

            'description.required' => 'Please give book description',

            'originalPrice' => 'Please give book originalPrice',

            'price.required' => 'Please give book price',

            'quantity.required' => 'Please give book quantity',

            'ratings.required' => 'Please give book ratings',

            'image.required' => 'Please give book image',

            'status.required' => 'Please give book status',

        ]);

        if ($validator->fails()) {

            return response()->json([

                'success' => false,

                'message' => $validator->getMessageBag()->first(),

                'errors' => $validator->getMessageBag(),

            ]);

        }

        $books = $this->bookRepository->create($request);

        return response()->json([

            'success' => true,

            'message' => 'Book Stored',

            'data'    => $books

        ]);

    }

    public function update(Request $request, $id)

    {

        $books = $this->bookRepository->findById($id);

        if (is\_null($books)) {

            return response()->json([

                'success' => false,

                'message' => 'Book Not found',

                'data' => null,

            ]);

        }

        $formData = $request->all();

        $validator = \Validator::make($formData, [

            'name' => 'required',

            'description' => 'required',

            'originalPrice' => 'required',

            'price' => 'required',

            'quantity' => 'required',

            'ratings' => 'required',

            'image' => 'required',

            'status' => 'required',

            'category\_id' => 'required'

        ], [

            'name.required' => 'Please give book name',

            'description.required' => 'Please give book description',

            'originalPrice' => 'Please give book originalPrice',

            'price.required' => 'Please give book price',

            'quantity.required' => 'Please give book quantity',

            'ratings.required' => 'Please give book ratings',

            'image.required' => 'Please give book image',

            'status.required' => 'Please give book status',

        ]);

        if ($validator->fails()) {

            return response()->json([

                'success' => false,

                'message' => $validator->getMessageBag()->first(),

                'errors' => $validator->getMessageBag(),

            ]);

        }

        $books = $this->bookRepository->edit($request, $id);

        return response()->json([

            'success' => true,

            'message' => 'Book Updated',

            'data'    => $books

        ]);

    }

    public function destroy($id)

    {

        $books = $this->bookRepository->findById($id);

        if (is\_null($books)) {

            return response()->json([

                'success' => false,

                'message' => 'Book Not found',

                'data' => null,

            ]);

        }

        $books = $this->bookRepository->delete($id);

        return response()->json([

            'success' => true,

            'message' => 'Book Deleted',

            'data'    => $books

        ]);

    }

}

### BookModel

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Book extends Model

{

    protected $table = "books";

    protected $primaryKey = "bookId";

    public function category()

    {

        return $this->belongsTo(Category::class);

    }

}

### Tạo API cho table Books

import Axios from "axios";

export const getBookList = async () => {

  return await Axios.get("http://127.0.0.1:8000/api/books").then(

    (res) => {

      return res.data;

    }

  );

};

export const storeNewBook = async (data) => {

    data.category\_id = parseInt(data.category\_id);

    return await Axios.post('http://127.0.0.1:8000/api/books', data)

    .then((res) => {

      return res.data;

    });

}

export const updateBook = async (id, data) => {

  data.category\_id = parseInt(data.category\_id);

  return await Axios.put(`http://127.0.0.1:8000/api/books/${id}`, data)

  .then((res) => {

    return res.data;

  });

}

export const deleteBook = async (id) => {

  return await Axios.delete(`http://127.0.0.1:8000/api/books/${id}`)

  .then((res) => {

    return res.data;

  });

}

## Chức năng Đăng ký/Đăng nhập

### AuthRepository

<?php

namespace App\repositories;

use App\interfaces\AuthInterface;

use App\Models\Category;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

use App\Models\User;

class AuthRepository implements AuthInterface{

    public function checkIfAuthenticated(Request $request)

    {

        if (Auth::attempt(['email' => $request->email, 'password' => $request->password])) {

            return true;

        }

        return false;

    }

    public function registerUser(Request $request)

    {

        $user = new User();

        $user->name = $request->name;

        $user->email = $request->email;

        $user->password = Hash::make($request->password);

        $user->save();

        return $user;

    }

    public function findUserByEmailAddress($email)

    {

        $user = User::where('email', $email)->first();

        return $user;

    }

}

### AuthAPIController

<?php

namespace App\Http\Controllers\API\Auth;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use App\Models\User;

use App\repositories\AuthRepository;

class AuthAPIController extends Controller

{

    public $authRepository;

    public function \_\_construct(AuthRepository $authRepository)

    {

        $this->authRepository = $authRepository;

    }

    public function createToken()

    {

        $user = User::first();

        $accessToken = $user->createToken('Token Name')->accessToken;

        return $accessToken;

    }

    public function login(Request $request)

    {

        $formData = $request->all();

        $validator = \Validator::make($formData, [

            'email' => 'required',

            'password' => 'required',

        ], [

            'email.required' => 'Please give your email address',

            'password.required' => 'Please give your password',

        ]);

        if ($validator->fails()) {

            return response()->json([

                'success' => false,

                'message' => $validator->getMessageBag()->first(),

                'errors' => $validator->getMessageBag(),

            ]);

        }

        if ($this->authRepository->checkIfAuthenticated($request)) {

            $user = $this->authRepository->findUserByEmailAddress($request->email);

            $accessToken = $user->createToken('authToken')->accessToken;

            return response()->json([

                'success' => true,

                'message' => 'Logged in successully !!',

                'user' => $user,

                'access\_token' => $accessToken,

            ]);

        } else {

            return response()->json([

                'success' => false,

                'message' => 'Sorry Invalid Email and Password',

                'errors' => null,

            ]);

        }

    }

    public function register(Request $request)

    {

        $formData = $request->all();

        $validator = \Validator::make($formData, [

            'name' => 'required|min:2|max:30',

            'email' => 'required|email|max:100|unique:users',

            'password' => 'required|confirmed|min:6',

        ], [

            'name.required' => 'Hãy điền tên.',

            'name.min' => 'Tên người dùng phải có 2-30 ký tự.',

            'name.max' => 'Tên người dùng phải có 2-30 ký tự.',

            'email.required' => 'Hãy điền địa chỉ email.',

            'email.max' => 'Địa chỉ email chứa nhiều nhất 100 ký tự.',

            'email.email' => 'Địa chỉ email không hợp lệ.',

            'email.unique' => 'Địa chỉ Email bạn đăng ký đã tồn tại. Hãy đăng ký tài khoản với địa chỉ Email khác.',

            'password.required' => 'Hãy điền mật khẩu.',

            'password.min' => 'Mật khẩu phải có ít nhất 6 ký tự.',

            'password.confirmed' => 'Mật khẩu không trùng khớp.',

        ]);

        if ($validator->fails()) {

            return response()->json([

                'success' => false,

                'message' => $validator->getMessageBag()->first(),

                'errors' => $validator->getMessageBag(),

            ]);

        }

        $user = $this->authRepository->registerUser($request);

        if(!is\_null($user)){

            $user = $this->authRepository->findUserByEmailAddress($request->email);

            $accessToken = $user->createToken('authToken')->accessToken;

            return response()->json([

                'success' => true,

                'message' => 'Đăng ký tài khoản thành công',

                'user' => $user,

                'access\_token' => $accessToken,

            ]);

        }else{

            return response()->json([

                'success' => false,

                'message' => 'Đăng ký tài khoản thất bại',

                'errors' => null,

            ]);

        }

    }

}

### UserModel

<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Illuminate\Support\Facades\Log;

use Laravel\Passport\HasApiTokens;

class User extends Authenticatable

{

    use HasApiTokens, HasFactory, Notifiable;

    protected $primaryKey = "userId";

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array

     \*/

    protected $fillable = [

        'name',

        'email',

        'password',

    ];

    /\*\*

     \* The attributes that should be hidden for arrays.

     \*

     \* @var array

     \*/

    protected $hidden = [

        'password',

        'remember\_token',

    ];

    public function shoppingCartItems(){

        return $this->hasMany('App\Models\ShoppingCart', 'userId', 'userId')

                    ->where('expired', false);

                    // ->where('wishList', false);

    }

    /\*\*

     \* The attributes that should be cast to native types.

     \*

     \* @var array

     \*/

    protected $casts = [

        'email\_verified\_at' => 'datetime',

    ];

}

# Các thư viện dự kiến sử dụng