**ahcme**

**《Web应用程序开发项目实训》实训报告**

**基于Laravel框架的客户信息管理系统**

系 （部）： 互联网与通信学院

班 级： 软件3194班

姓 名： 郝宏鑫

小组名称： dalao带带我

指导教师： 王军 赵宏

互联网与通信学院课程实训评价表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **自我评价**：  通过本次实训我学到了很多同时也发现了自己的很多不足，在今后的学习生活中还需巩固基础知识,提升自身综合能力理解并掌握软件开发的主流软件设计模式和设计思想 。 | | | | | |
| **教师评价**： | | | | | |
| 成绩（等级） |  | 教师签名 |  | 日期 | 2021-7-1 |

**摘 要**

Laravel是 Taylor Otwell 开发的一款基于 PHP 语言的 Web 开源框架，旨在实现的Web软件的MVC架构。让你可以通过简单、优雅的表达式语法开发出很棒的Web应用，Laravel同时拥有更富有表现力的语法、高质量的文档、丰富的扩展包，它可以让你从面条一样杂乱的代码中解脱出来。

这次实训我将学习和使用 Laravel 框架，并结合 MySQL/MariaDB数据库开发一个带有用户身份验证和数据库增、删、改、查操作的信息管理系统。

关键词: PHP、Laravel、MVC、信息管理系统

目 录

[第1章 需求分析 1](#_Toc76073911)

[1.1 需求描述 1](#_Toc76073912)

[1.2 功能描述 2](#_Toc76073913)

[第2章 总体设计 3](#_Toc76073914)

[2.1 运行环境 3](#_Toc76073915)

[2.2 软件结构设计 3](#_Toc76073916)

[第3章 界面设计 4](#_Toc76073917)

[3.1 主界面设计 4](#_Toc76073918)

[第4章 详细设计 9](#_Toc76073919)

[4.1 Laravel 生命周期 9](#_Toc76073920)

[4.2 详细设计描述 10](#_Toc76073921)

[第5章 代码设计 12](#_Toc76073922)

[第6章 总结 18](#_Toc76073923)

[**参考文献** 19](#_Toc76073924)

# 第1章 需求分析

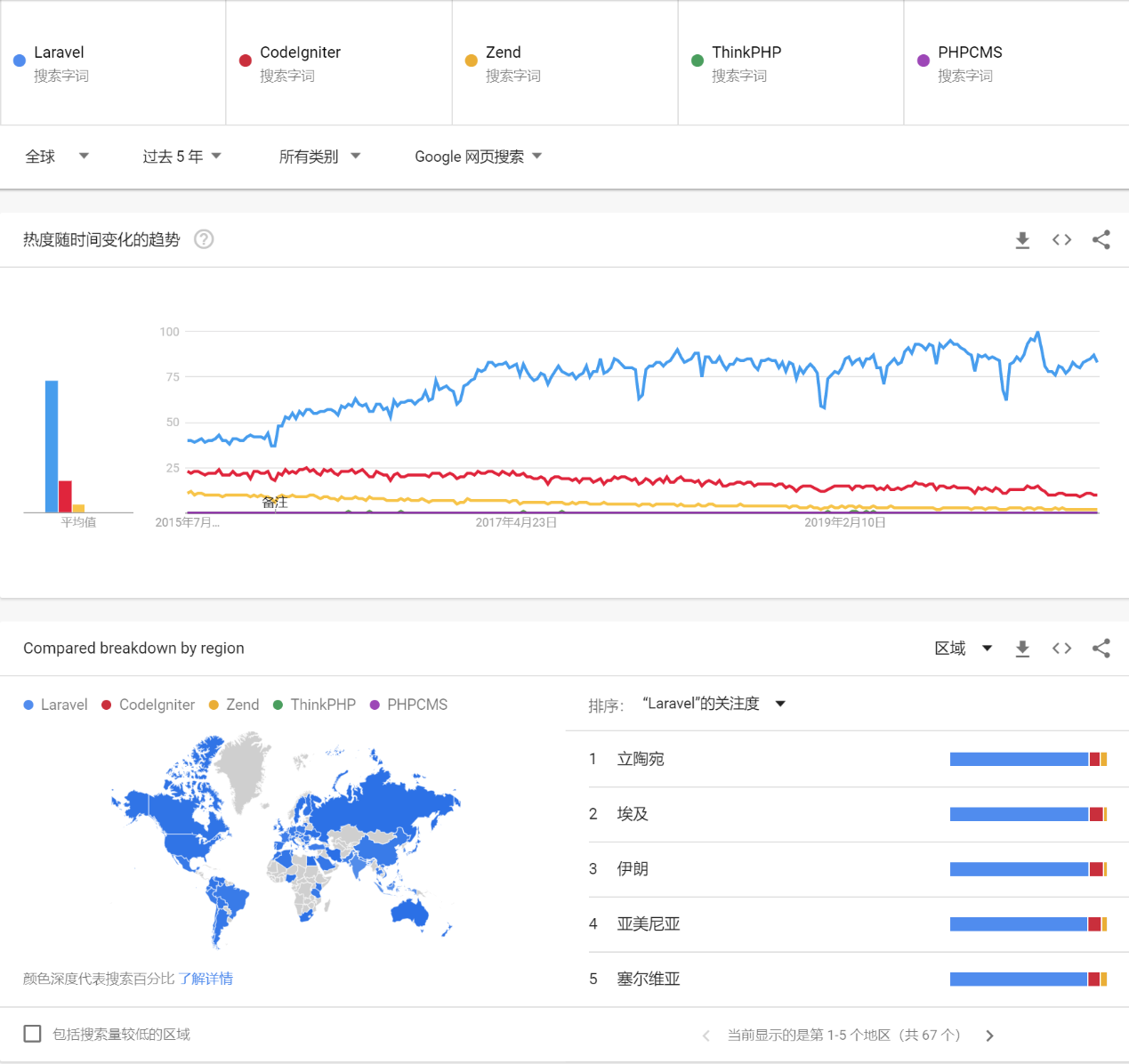
1.1 需求描述

系统分析是开发管理信息系统的关键性阶段，是一个从不断认识和逐步细化的过程，是下一阶段的工作基础，是为下一阶段进行物理方案设计、解决“怎么做”提供依据，基关键性主要体现在“理解需求”和“表达需求”两方面。

通过对系统的整体分析，我们应该更能理解系统所要实现的功能，所需的环境支持，并提出这些需求的实现条件以及需求应达到的标准，也就是确定新系统要做什么，做到什么程度。

这次实训选择的课题是基于Laravel 框架的信息管理系统。既然是信息管理系统就应该包括一套用户身份验证机制防止未授权访客用户越权操作，然后包含基本的数据库增删改查操作。

为了实现这一需求可供选择的方案大致分为两种，通过PHP 原生代码开发或通过 PHP开发框架进行开发；这里我选择使用 PHP 开发框架进行开发，通过 Google Trends 的数据来看 Laravel 近年来已经成为最受开发者欢迎的 PHP 开发框架如图 1-1。通过使用开发框架不仅能够极大的提升开发效率，降低维护成本还可以学习到 MVC 设计思想。

图1-1 Google Trends

1.2 功能描述

1. 用户身份验证

实现用户注册、用户登录、找回密码、未登录跳转功能

2）显示客户信息

通过一个 Web 页面显示数据表中的客户信息并提供搜索功能

1. 新增客户信息

实现通过 Web 页面添加客户信息功能数据插入到数据库

1. 编辑删除客户信息

对已有的客户信息进行修改和删除操作

# 第2章 总体设计

2.1 运行环境

操作系统平台: Windows/Linux/macOS

数据库平台: MySQL/Mariadb

开发工具: PhpStrom + PHP 8.0.3

运行平台: Apache/Nginx + Laravel + PHP

2.2 软件结构设计

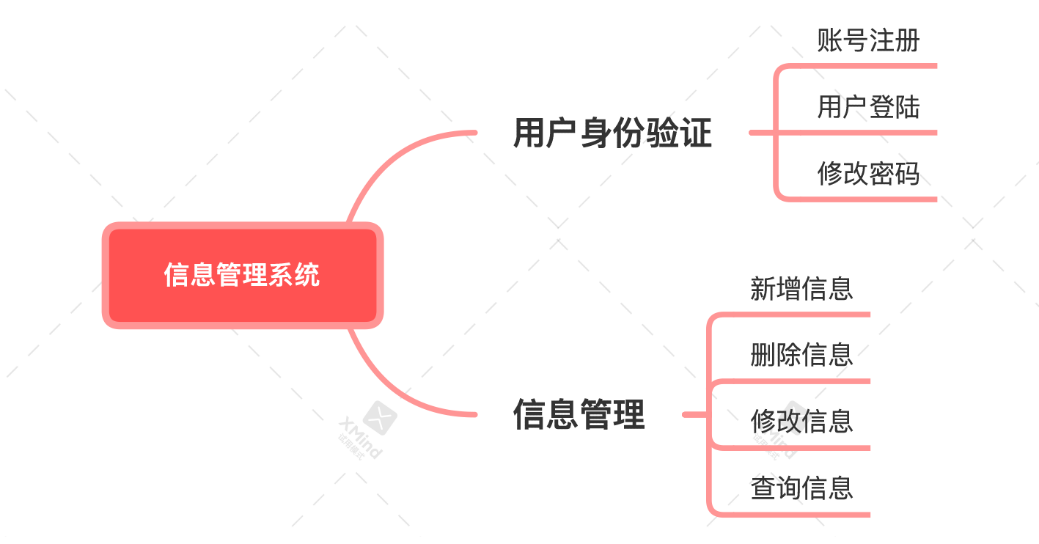


图 2-1 结构图

# 第3章 界面设计

3.1 主界面设计

1) 主界面设计如: 图 3-1、图 3-2、图3-3、图3-4、图3-5、图3-6、图3-7、图3-8

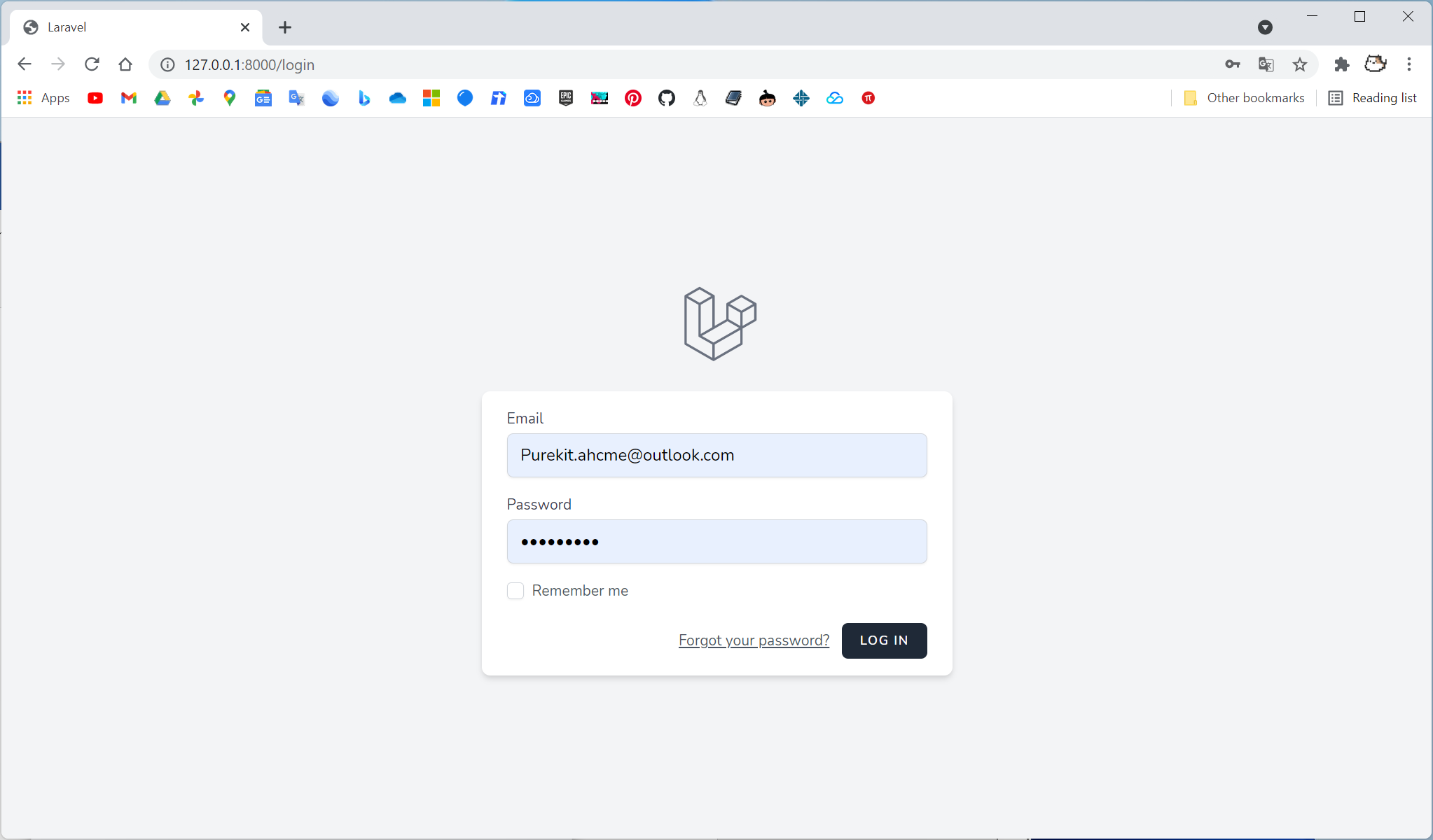


图3-1 登陆页面

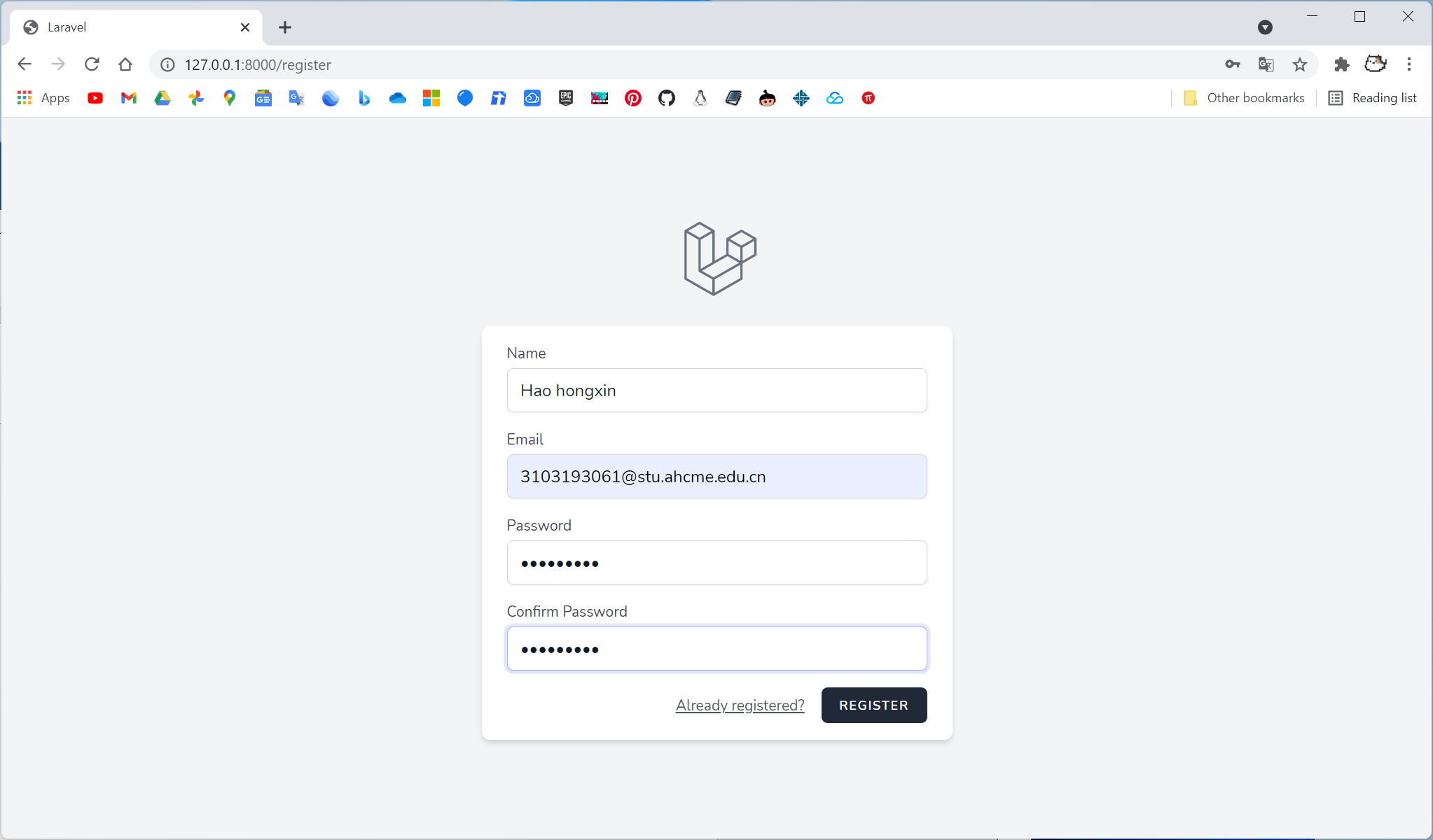
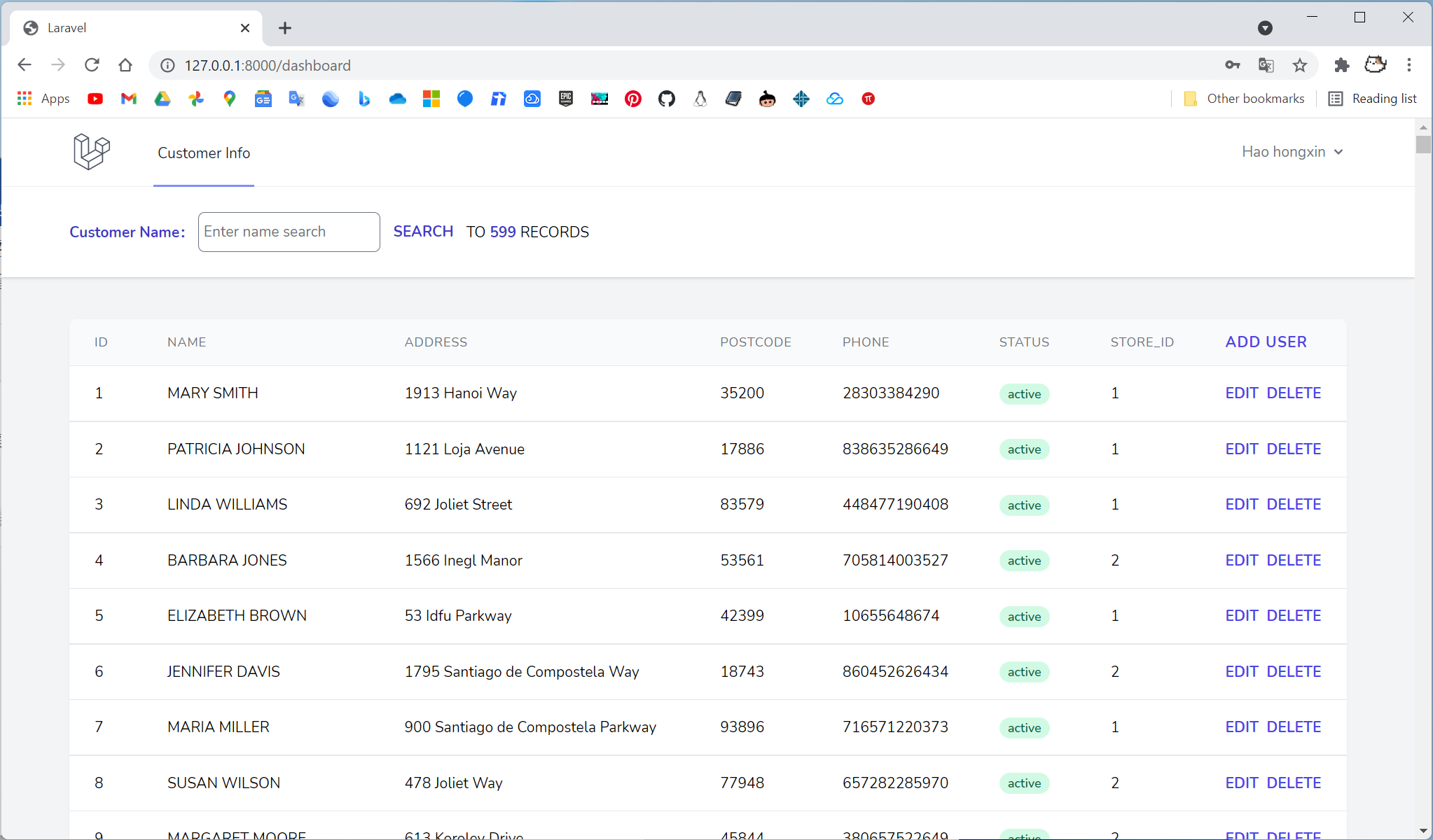
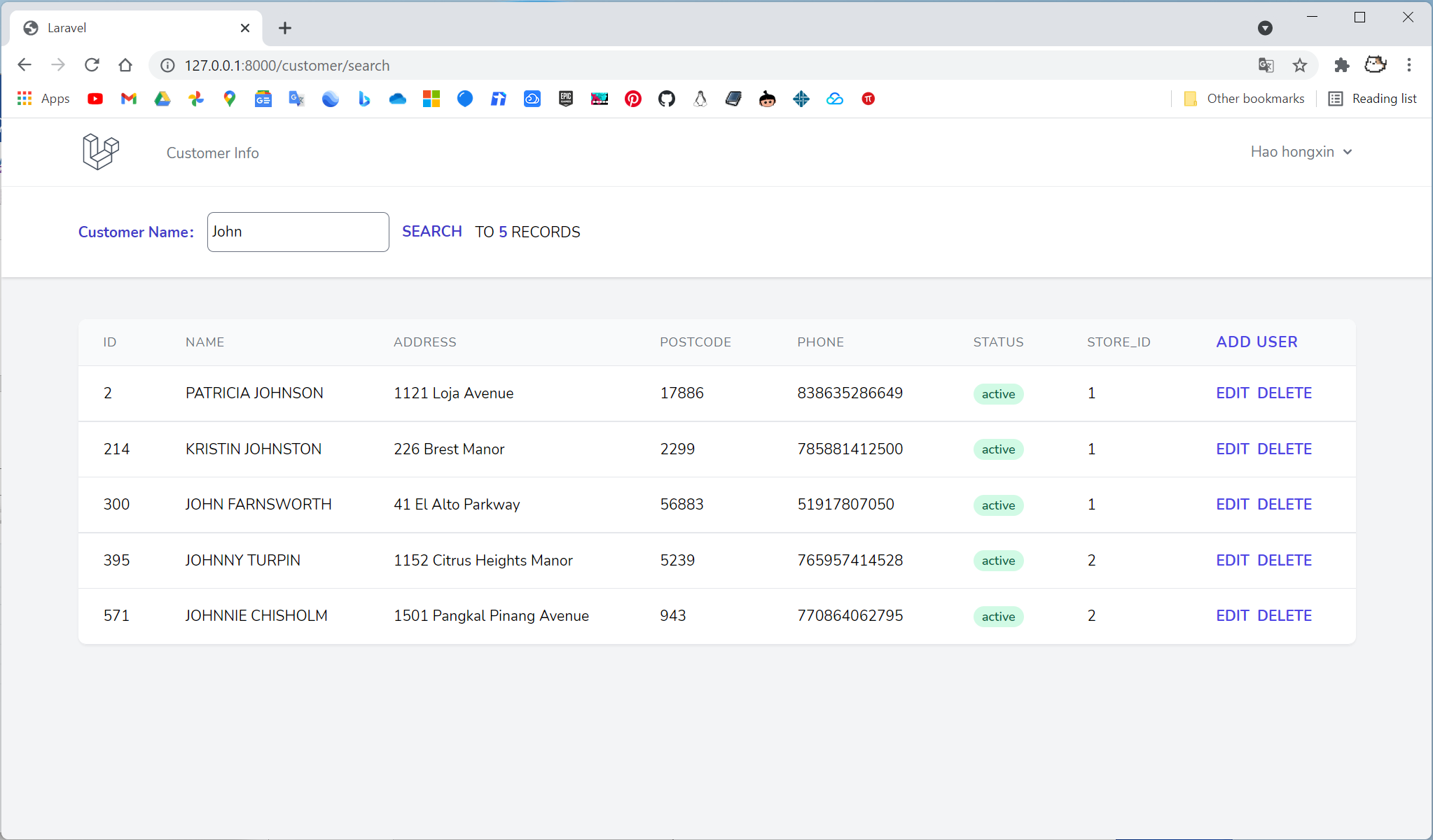


图3-2 注册页面

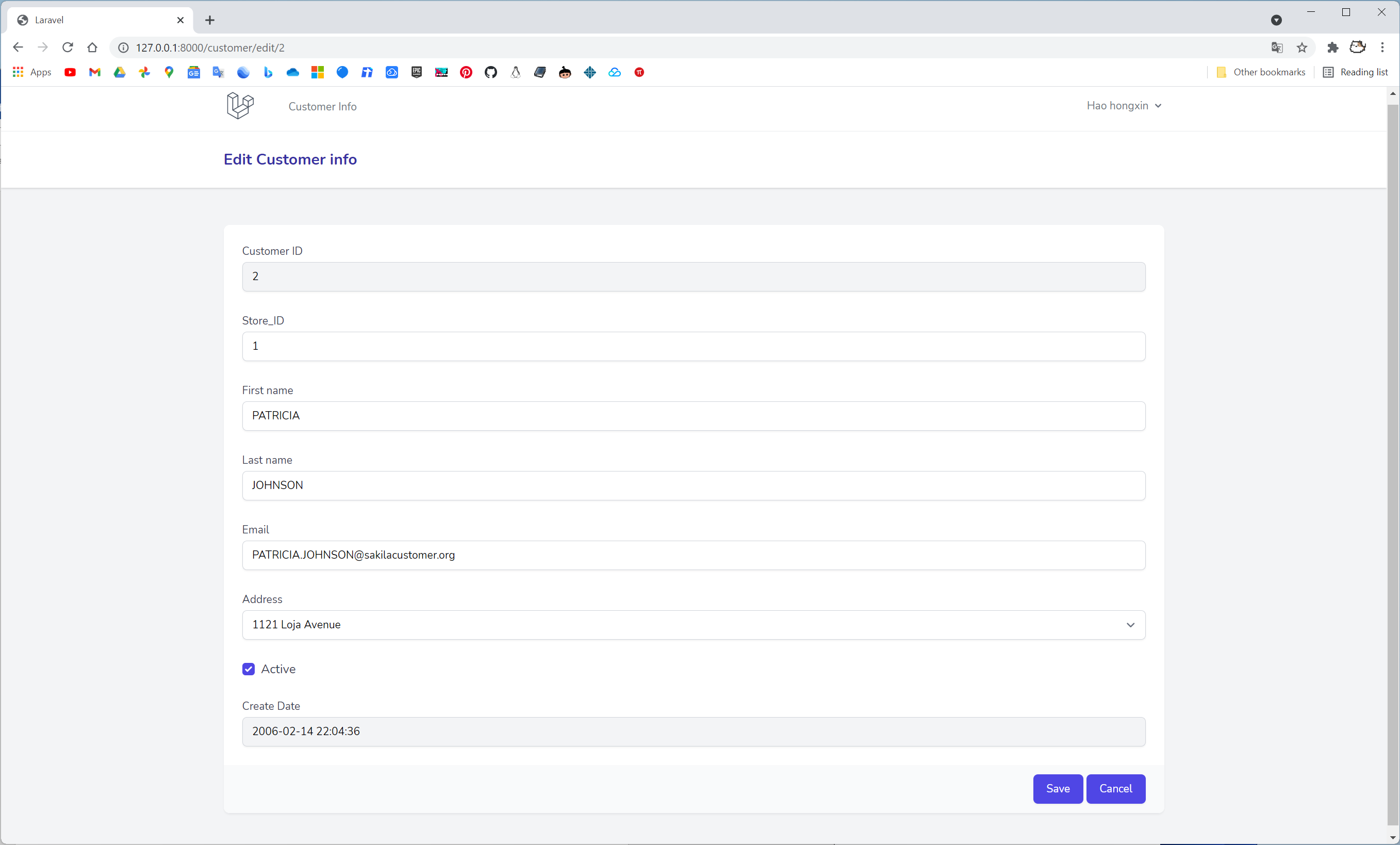
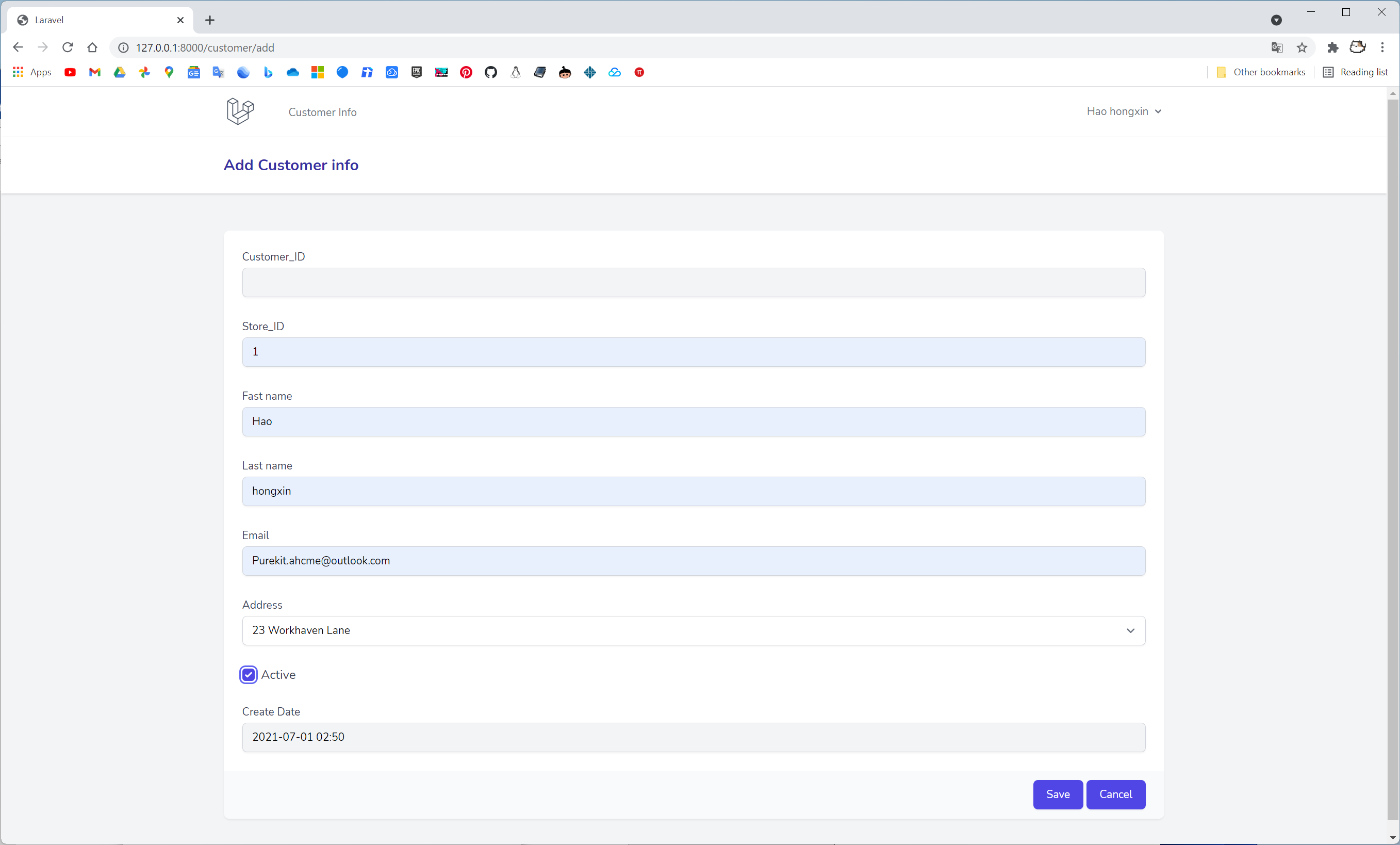
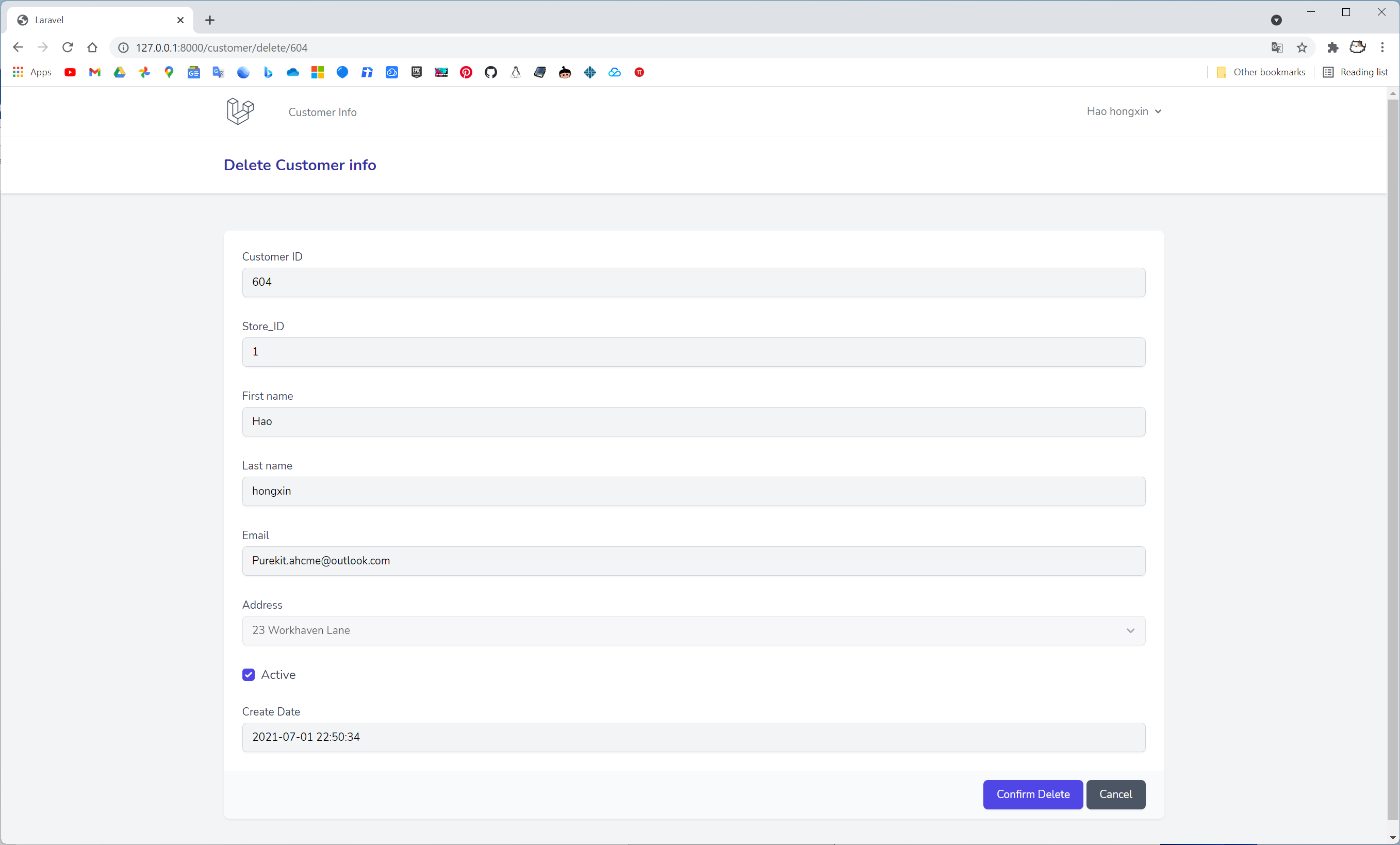
身份验证页面由 Laravel 框架自动生成包含完整的用户身份验证逻辑。

图3-3 首页 (客户信息显示)

客户信息显示页面通过检索 Sakila 数据库的 customer\_list 视图获取客户信息，返回 customer.show 视图并传递检索到的客户数据。

图3-4 客户信息搜索页面

搜索框输入数据通过 POST 方式提交到 CustomerController 的 search() 方法通过 DB::table 静态方法链式调用查询包含姓名关键词的客户信息。

图3-5 客户信息修改页面图3-6 添加客户信息页面图3-7 删除信息

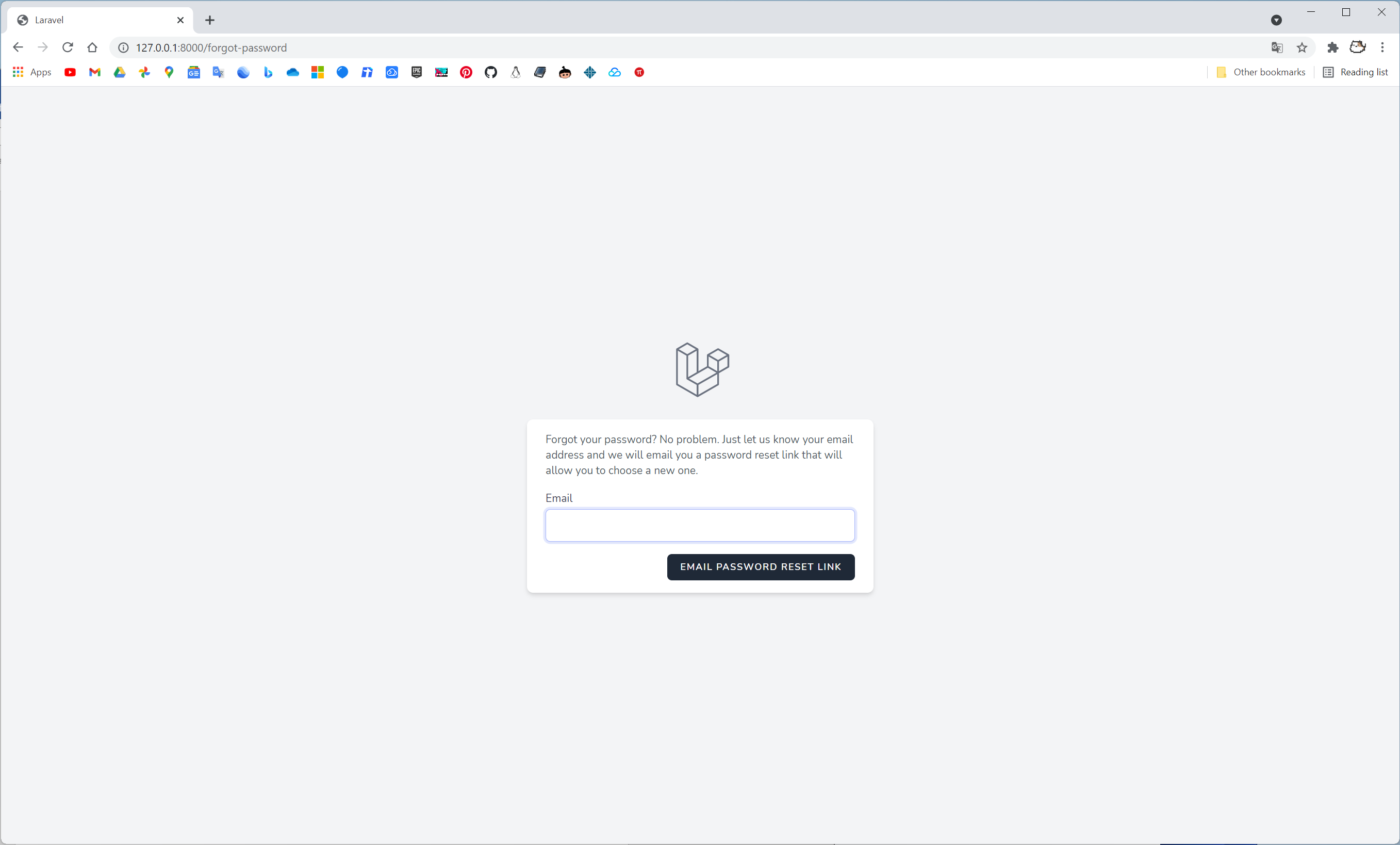


图3-8 首页 密码重置页面

# 第4章 详细设计

4.1 Laravel 生命周期

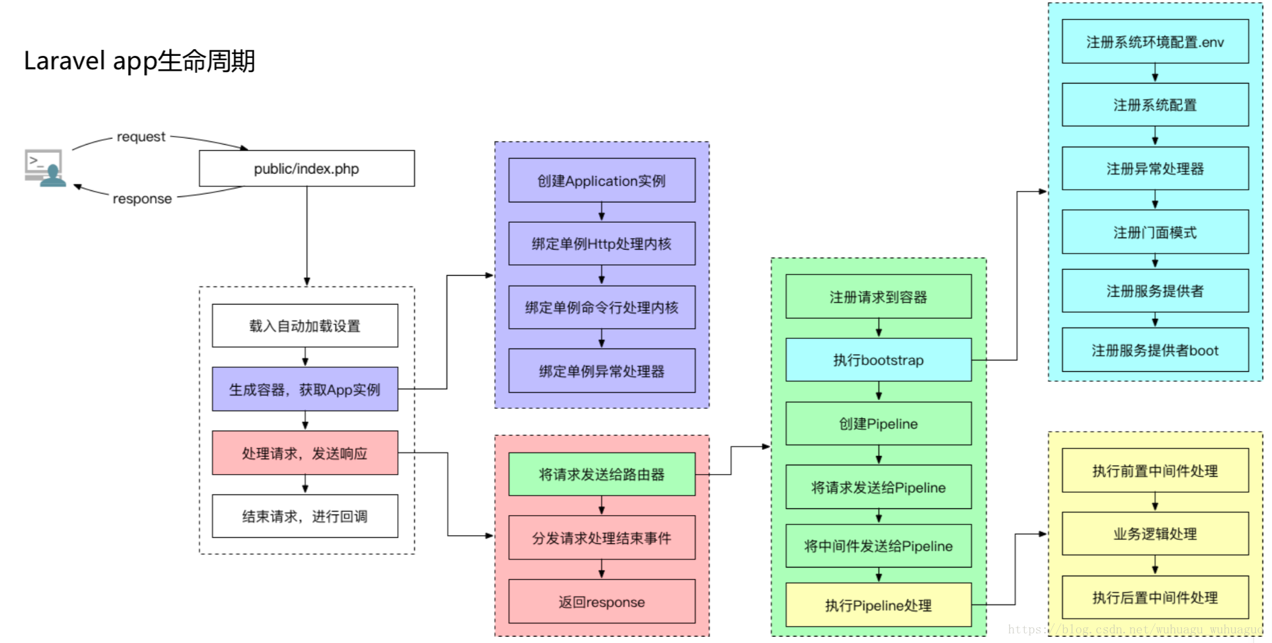


图4-1 Laravel 生命周期

Laravel 框架将所有的 Request 和 Response 的出入口设定为 public/index.php 进行管理，然后交由路由进行统一控制在路由之后数据进入业务逻辑之前会经过框架默认加载的 Middleware(中间件)继续数据合法性检测加密 Cookies 验证 CSRF 等处理，中间件又分为前置和后置中间件，我们也可以自己编写中间件来实现需要的操作。

框架通过这些机制把项目拆分成不同的部分，各个组件之间相互配合各司其职。规范了项目的编码风格，提升了开发效率并降低维护成本。

4.2 详细设计描述

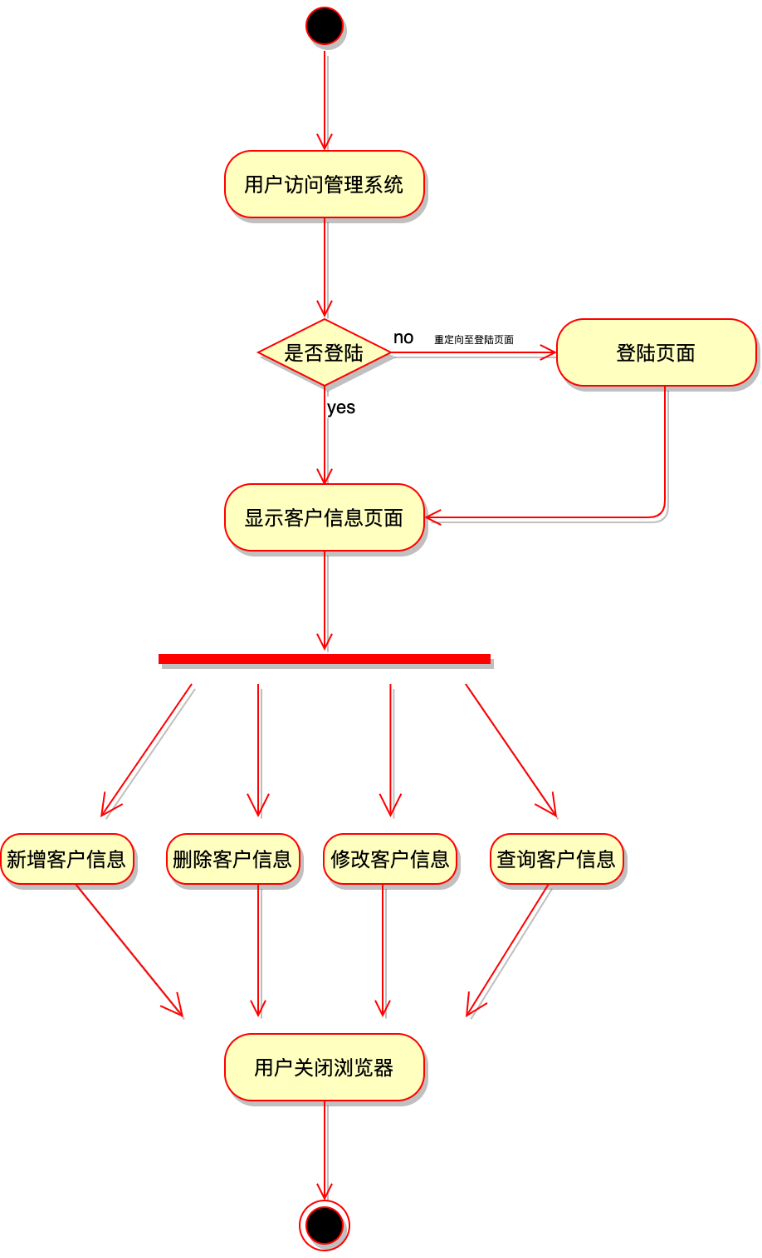


图4-2 用户活动图

客户信息管理系统分为几个子系统，分别提供用户注册登陆、添加、删除、修改、查询功能。

用户身份验证系统，使用 Laravel 提供的 Laravel/ui 包自动生成控制器视图和路由数据库迁移文件等业务逻辑。通过在路由中设定 Auth middleware 中间件即可实现安全路由功能，启用 Auth 中间件的路由必须在用户登陆之后才可访问否则会跳转回登陆页面。

页面导航栏由 Laravel 用户验证模块自动生成的 app.blade.php 布局文件提供，其他页面通过Laravel Jetstream前端UI和TailWind Css实现布局。

客户信息的增删改查操作通过 Laravel 的 Eloquent 数据模型来间接修改数据库表单，表单提交数据通过创建 Request 请求类修改 rules 方法添加字段来验证数据合法性当表单提交当数据验证失败时会重定向至提交前的页面。

整个业务逻辑代码都被单独放置在 CusotmerController 控制器中，通过路由定义所需响应调用 CustomerController 中的不同公开方法来实现身份验证增删改查等一系列操作。

页面在 Web 浏览器中的页面显示部分被抽离为一个个单独的 View 视图文件，实现显示于逻辑的分离通过 Blade 语法动态绑定参数

# 第5章 代码设计

1. 全局 Web 访问路由设定

/routes/web.php

<?php

use Illuminate\Support\Facades\Route;

use Illuminate\Http\Request;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| contains the "web" middleware group. Now create something great!

|

\*/

Route::get('/', function () {

return redirect()->route('customer.show');

});

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

// 定义 Customer 管理保护路由组

Route::prefix('customer')->middleware('auth')->group(function () {

// 定义增加客户信息路由

Route::get('add', 'CustomerController@add')

->name('customer.add');

// 定义增加客户信息路由

Route::post('insert', 'CustomerController@insert')

->name('customer.insert');

// 定义删除客户信息路由

Route::get('delete/{customer\_id}', 'CustomerController@delete')

->name('customer.delete');

// 定义编辑客户信息路由

Route::get('edit/{customer\_id}', 'CustomerController@edit')

->name('customer.edit');

// 定义更新客户信息路由

Route::post('update', 'CustomerController@update')

->name('customer.update');

// 定义显示客户信息路由

Route::get('show', 'CustomerController@show')

->name('customer.show');

// 定义搜索客户信息路由

Route::post('search', 'CustomerController@search')

->name('customer.search');

});

1. CustomerController 控制器

/app/Http/Controllers/CustomerController.php

<?php

namespace App\Http\Controllers;

use App\Http\Requests\CustomerUpdate;

use App\Models\Address;

use App\Models\Customer;

use App\Models\CustomerList;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

class CustomerController extends Controller {

/\*\*

\* 增加客户信息

\*/

public function add() {

// 获取地址列表

$address = Address::all('address\_id', 'address');

return view('customer.add', ['address' => $address]);

}

/\*\*

\* 插入客户信息

\*/

public function insert(CustomerUpdate $request) {

// 验证数据合法性

$validated = $request->validated();

$info = $request->all('store\_id', 'first\_name', 'last\_name', 'email', 'address\_id', 'active', 'create\_date');

if (!isset($info['active'])) {

$info['active'] = 0;

}

Customer::query()->insert($info);

return redirect()->route('customer.show');

}

/\*\*

\* 删除客户信息

\*

\* @param $customer\_id 客户 ID

\*/

public function delete($customer\_id) {

$customer = Customer::query()->find($customer\_id);

$customer->delete();

return redirect()->route('customer.show');

}

/\*\*

\* 编辑客户信息

\*

\* @param

\*/

public function edit($customer\_id) {

// 获取客户信息

$customer = Customer::query()->find($customer\_id);

// 获取地址列表

$address = Address::all();

return view('customer.edit', [

'customer' => $customer,

'address' => $address

]);

}

/\*\*

\* 更新客户信息

\*

\* @param Request $request

\*/

public function update(CustomerUpdate $request) {

// 验证数据合法性

$validate = $request->validated();

$info = $request->all();

if (!isset($info['active'])) {

$info['active'] = 0;

}

DB::update('update sakila.customer set store\_id=?,first\_name=?,last\_name=?,email=?,address\_id=?,active=? WHERE customer\_id=?',

[$info['store\_id'], $info['first\_name'], $info['last\_name'], $info['email'], $info['address\_id'], $info['active'], $info['id']]);

return redirect()->route('customer.edit', ['customer\_id' => $info['id']]);

}

/\*\*

\* 显示客户信息

\*

\* @return 客户信息管理视图

\*/

public function show() {

$customers = CustomerList::all()

->sortBy("ID");

return view('customer.show', ['customers' => $customers]);

}

/\*\*

\* 根据姓名查询客户信息

\*

\* @param Request $request

\* @return 客户信息管理视图

\*/

public function search(Request $request) {

// 将输入数据闪存到 Session

$request->flash();

// 获取表单提交的客户

$name = $request->input('name');

$customers = CustomerList::query()

->where('name', 'like', "%$name%")

->orderBy('id')

->get();

return view('customer.show', ['customers' => $customers]);

}

}

1. CustomerUpdate 请求类

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class CustomerUpdate extends FormRequest

{

/\*\*

\* Determine if the user is authorized to make this request.

\*

\* @return bool

\*/

public function authorize()

{

return true;

}

/\*\*

\* Get the validation rules that apply to the request.

\*

\* @return array

\*/

public function rules()

{

return [

'store\_id' => 'bail|required',

'first\_name' => 'bail|required|max:15',

'last\_name' => 'bail|required|max:15',

'email' => 'bail|required|max:50',

'address\_id' => 'bail|required',

];

}

}

1. Customer 数据表 Eloquent 数据 Model

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Customer extends Model {

protected $table = "sakila.customer";

protected $primaryKey = "customer\_id";

public $timestamps = false;

public $incrementing =false;

protected $fillable = [

'store\_id', 'first\_name', 'last\_name', 'email', 'address\_id', 'active'

];

protected $attributes = [

'active' => 0

];

}

# 第6章 总结

通过这次 PHP 实训的学习，我初步了解到了通过应用程序开发框架进行项目开发的高效快捷之处，同时也体会到了 MVC 软件开发架构的基本思想。

软件开发从汇编到面向过程乃至面向对象，是一层层的抽象我们通过抽象对现实世界里的具体事物和程序代码进行联系并简化具体细节。使程序员的注意力从具体的实现细节专注到业务逻辑之上，极大的提高了开发效率但是面对庞大的软件系统和复杂的业务逻辑。代码最终写的越来越庞大杂乱难以理解，PHP 作为一门弱类型语言，虽然在更加自由特别是编写代码的时候。但是它不具有强类型语言那样在编译时即可发现所有语法错误的好处，往往后期维护会十分痛苦，在业务逻辑越写越多的情况下，我们难以清晰的对项目本身有一个清晰的把控。

软件开发框架，通过一定的规范约束和 MVC 等分层思想将具体的业务逻辑进行分离实现低耦合高内聚。

现阶段的发展趋势逐渐倾向于 B/S 模式，NodeJS 的出现为 JavaScript 在服务端运行提供平台。Web 开发工程师所需要的掌握的技能不在局限于前端或者后端两者之间的界限越来越模糊，新技术层出不穷出现了一大堆优秀的前后端框架 Boostrap Augur.js Vue.js 等。PHP 生态似乎有所萎缩，很多主流网站的后端开始转向 NodeJS 或者 Go 语言。

在今后的学习生活中，我会选择学习以 NodeJS 为平台的后端框架如 Vue.js 等还有新兴的 Go 语言。

**参考文献**

[1] Laravel 中文社区. Laravel 8 中文文档. https://learnku.com/docs/laravel/8.x

[2] TailWind Css中文文档.

https://www.tailwindcss.cn/docs

[3] Laravel 8 Eloquent 快速入门https://learnku.com/docs/laravel/8.x/eloquent/9406