Регистрация с подтверждением через **email** (без реального домена-хостинга только до момента отправки link на email-login, т.е. на Vagrant, например, для завершения регистрации – в phpmyadmin изменить значение поля active на 1).

**1.migrations\2014\_10\_12\_000000\_create\_users\_table.php**

$table->integer('active')->default(0);

+ **DatabaseSeeder.php для всех users**

'active' => 1,

**remove all tables + php artisan migrate –seed**

**2.routes\web.php**

**Route::name('letter')->get('/letter', 'Front\ShopController@letter');**

**3.Controllers\Front\ShopControler.php**

/\*\*

\* Show message of sending letter to email for finish registration.

\*

\* @return void

\*/

**public function letter()**

**{**

**return view('auth.letter');**

**}**

**+ views\auth\letter.blade.php** (from lesson)

@extends('front.layout')

@section('main')

<div class="products-catagories-area clearfix">

<div class="amado-pro-catagory clearfix">

<div class="container">

<div class="row justify-content-center">

<div class="col-md-8">

<div class="card">

<div class="card-header">{{ \_\_('Finish Registration') }}</div>

<div class="card-body">

<h3>Chech your email and go to link...</h3>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

@endsection

**4.Controllers\Auth\RegisterControler.php**

...

**use Illuminate\Http\Request;**

**use Illuminate\Auth\Events\Registered;**

/\*\*

\* Create a new user instance after a valid registration.

\*

\* @param array $data

\* @return \App\User

\*/

protected function create(array $data)

{

return User::create([

'name' => $data['name'],

'email' => $data['email'],

'password' => Hash::make($data['password']),

**'api\_token' => str\_random(8),**

]);

}

/\*\*

\* Handle a registration request for the application.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function **register**(Request $request)

{

**$this->validator($request->all())->validate();**

**event(new Registered($user = $this->create($request->all())));**

//$this->guard()->login($user); **//!!!НЕТ ЛОГИНА**

//return $this->registered($request, $user)

//?: redirect($this->redirectPath()); **//!!!НЕТ РЕДИРЕКТА - БУДЕТ СВОЙ КАСТОМНЫЙ**

**return redirect('/letter'); //!!!'/letter' - СВОЙ КАСТОМНЫЙ redirect**

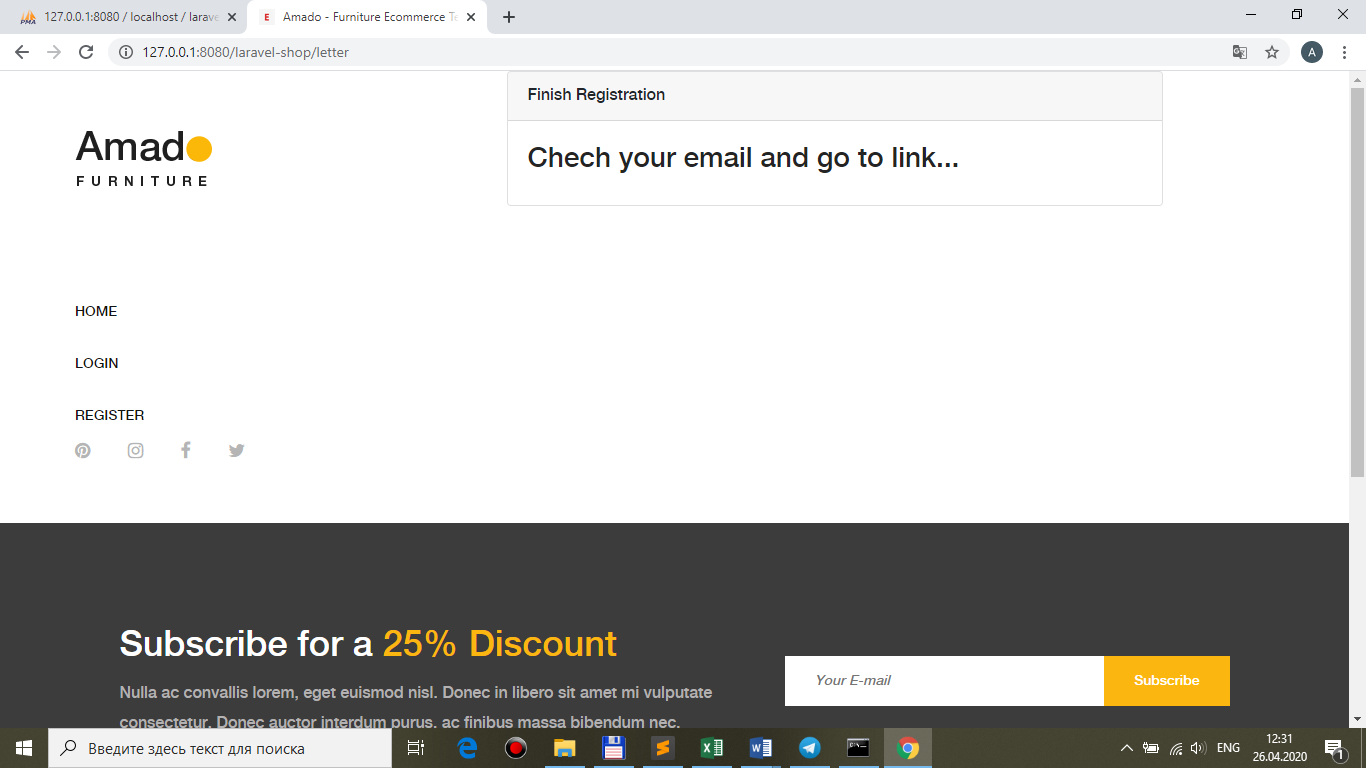
}

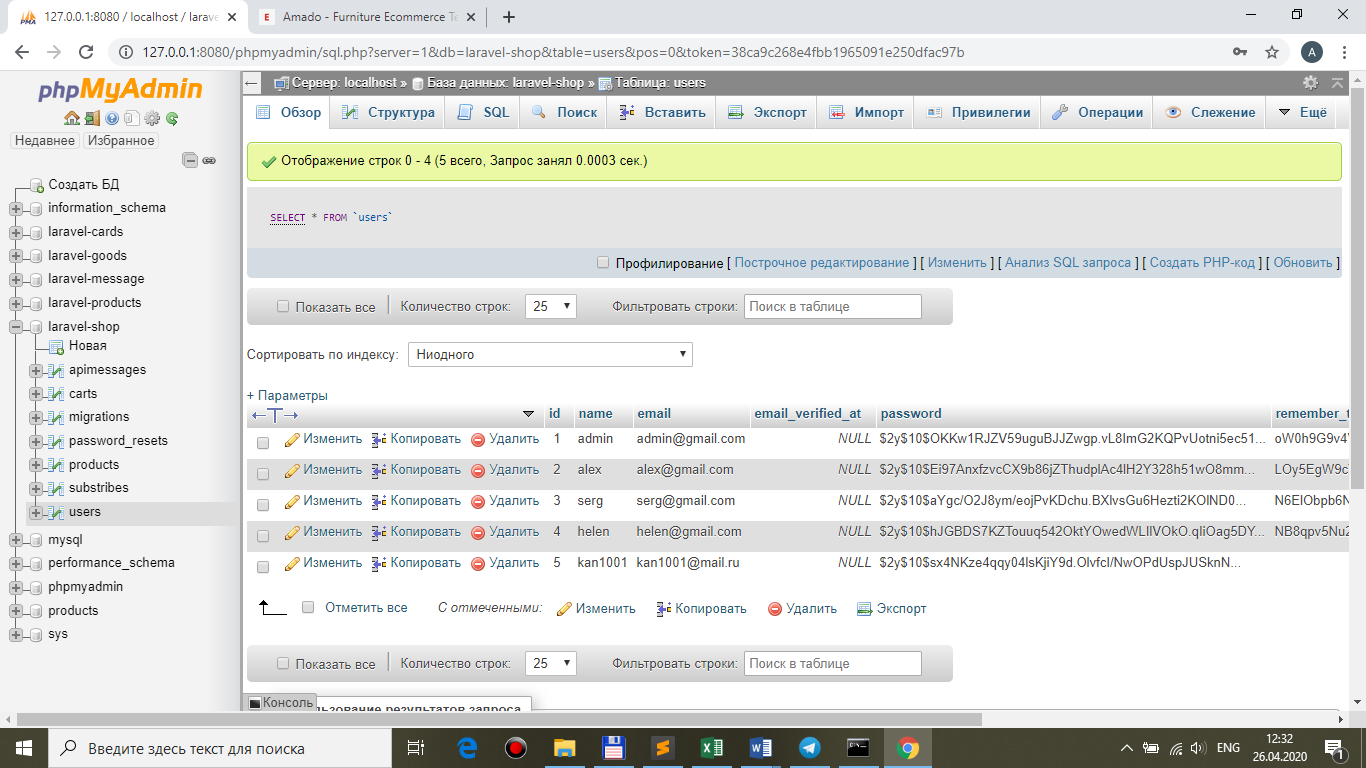
**+ Models\User.php**

protected $fillable = [

'name', 'email', 'password', **'api\_token',**

];





**5.Controllers\Auth\LoginControler.php**

…

**use Illuminate\Http\Request;**

**use App\Models\User;**

public function **login**(Request $request)

{

$this->validateLogin($request);

// If the class is using the ThrottlesLogins trait, we can automatically throttle

// the login attempts for this application. We'll key this by the username and

// the IP address of the client making these requests into this application.

if (method\_exists($this, 'hasTooManyLoginAttempts') &&

$this->hasTooManyLoginAttempts($request)) {

$this->fireLockoutEvent($request);

return $this->sendLockoutResponse($request);

}

**//!!!**

**$user = User::select('active')->where('email', $request->email)->get();**

**if ($user[0]->active && $this->attemptLogin($request)) {**

**return $this->sendLoginResponse($request);**

**} else {**

**// Increment the failed login attempts and redirect back to the**

**// login form with an error message.**

**$this->incrementLoginAttempts($request);**

**return redirect()**

**->back()**

**->with('oldemail', $request->email) //!!!session('oldemail')**

**->with('active', 'You must be active to login.'); //!!!session('active')**

**}**

**//!!!**

// If the login attempt was unsuccessful we will increment the number of attempts

// to login and redirect the user back to the login form. Of course, when this

// user surpasses their maximum number of attempts they will get locked out.

**//**$this->incrementLoginAttempts($request); //!!!

return $this->sendFailedLoginResponse($request);

}

**+ views\auth\login.blade.php**

...

<div class="card-body">

**@if (session('active'))**

**@component('front.components.alert')**

**@slot('type')**

**danger**

**@endslot**

**{!! session('active') !!}**

**@endcomponent**

**@endif**

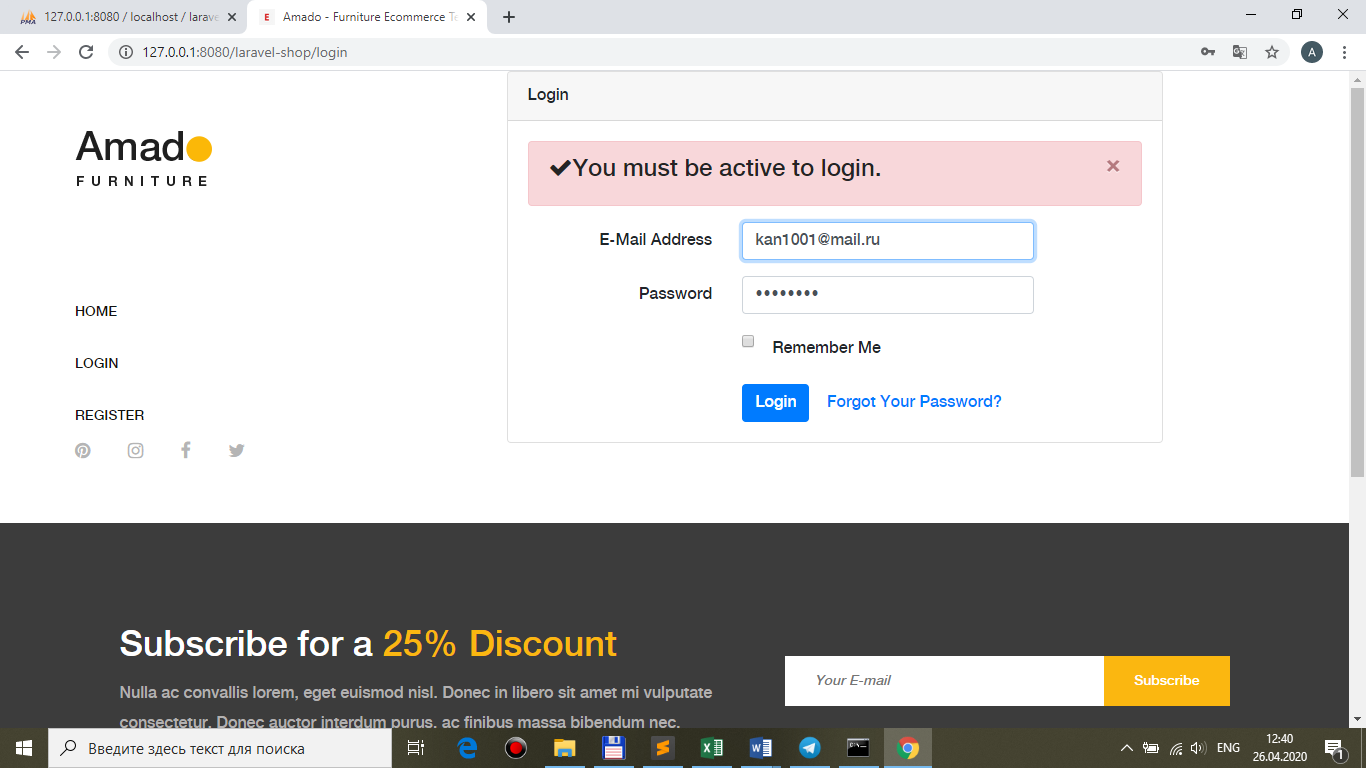
<form method="POST" action="{{ route('login') }}">

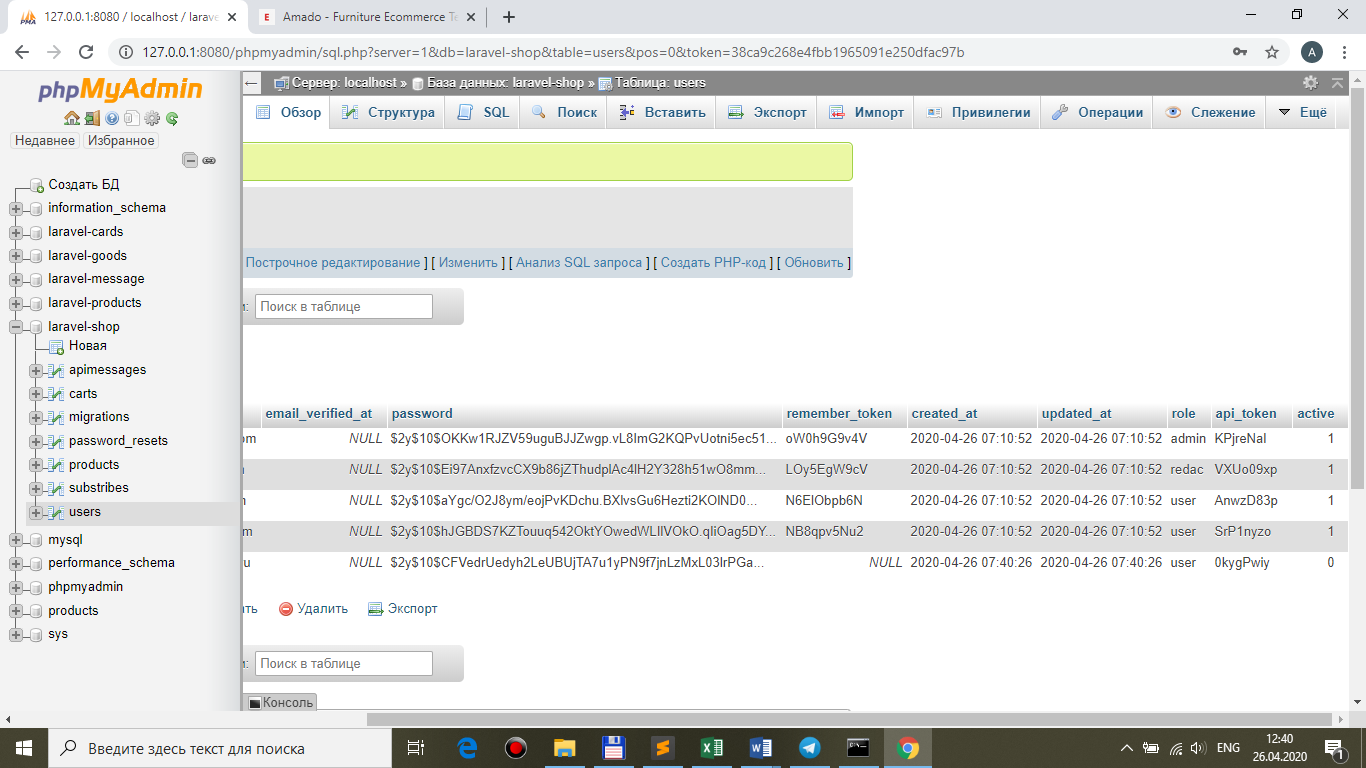
...

<div class="col-md-6">

<input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="@if(old('email')) {{ old('email') }} **@elseif(session('active')){{session('oldemail')}}**@else admin@gmail.com @endif" required autocomplete="email" autofocus>

...





**6.Controllers\Auth\RegisterControler.php**

public function **register**(Request $request)

{

$this->validator($request->all())->validate();

event(new Registered($user = $this->create($request->all()))); //!!!$user

**$this->funcMailer($user); //!!!$user**

return redirect('/letter');

}

**public function funcMailer($user) //!!!from lesson**

**{**

**$title = 'Finish of registration - ' . date('d-m-Y H:i:s');**

**$message = 'For finish of registration go to this link <a href="http://fullstack.25one.com.ua/api/apimessages?api\_token=' . $user->api\_token . '">http://fullstack.25one.com.ua/api/apimessages?api\_token=' . $user->api\_token . '</a>';**

**$client = new \GuzzleHttp\Client([**

**'headers' => [**

**]**

**]);**

**$response = $client->request('GET', 'http://api.25one.com.ua/api\_mail.php?email\_to=' . $user->email . '&title=' . $title . '&message=' . $message, //!!!$user->emai**

**[**

**//...**

**]);**

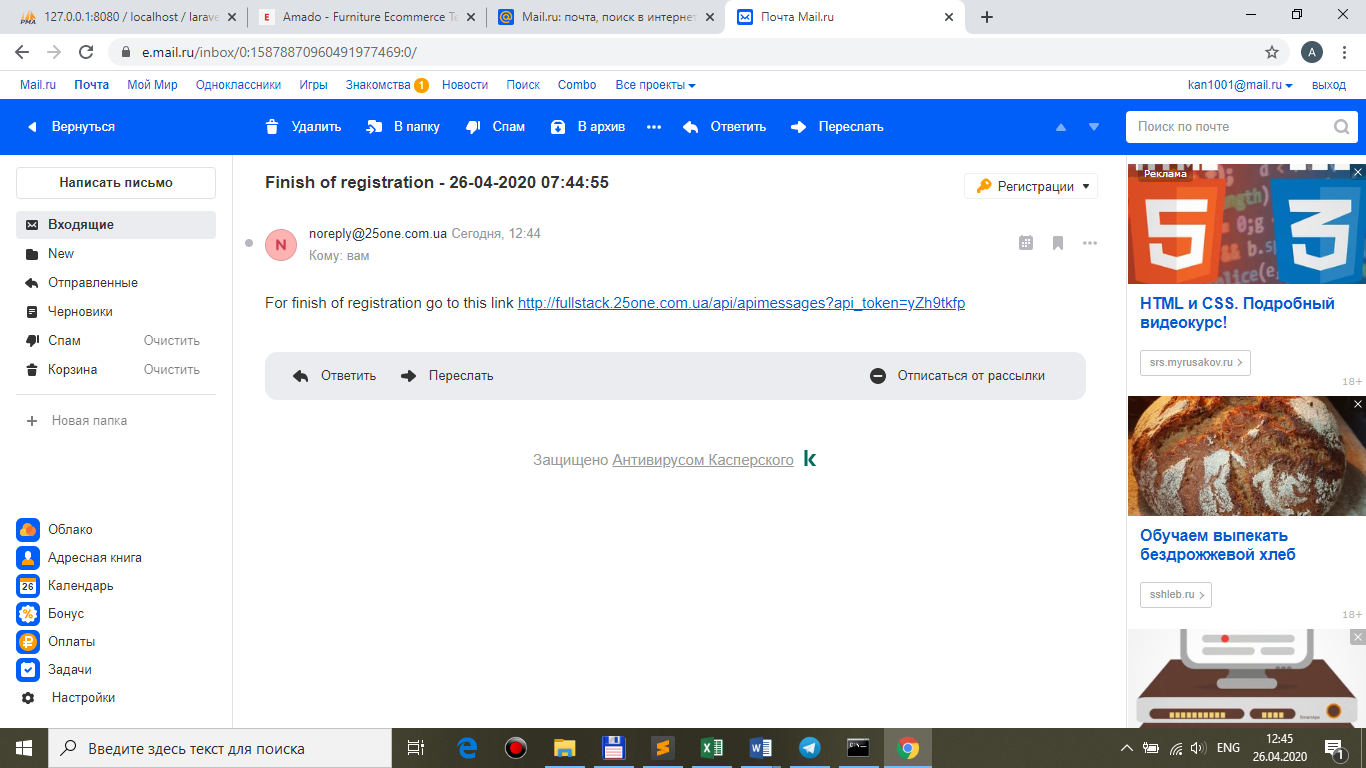
**//return json\_decode($response->getBody()->getContents());**

**return response()->json([**

**'answer' => $response->getBody()->getContents(),**

**]);**

**}**



--------------------------- **ДАЛЕЕ ТОЛЬКО ДЛЯ РЕАЛЬНОГО ДОМЕНА-ХОСТА**

**ApiController.php**

/\*\*

\* Finish of user-registration.

\*

\* @return void

\*/

public function index() //!!!index for GET-RESTful api

{

$user = User::find(\Auth::guard('api')->user()->id);

$user->active = 1;

$user->save();

return redirect(route('login'));

}