

به نام خدا  
مستندات پروژه سیستم تیکتینگ

صفحه ورود:



ورود

admin@admin.com

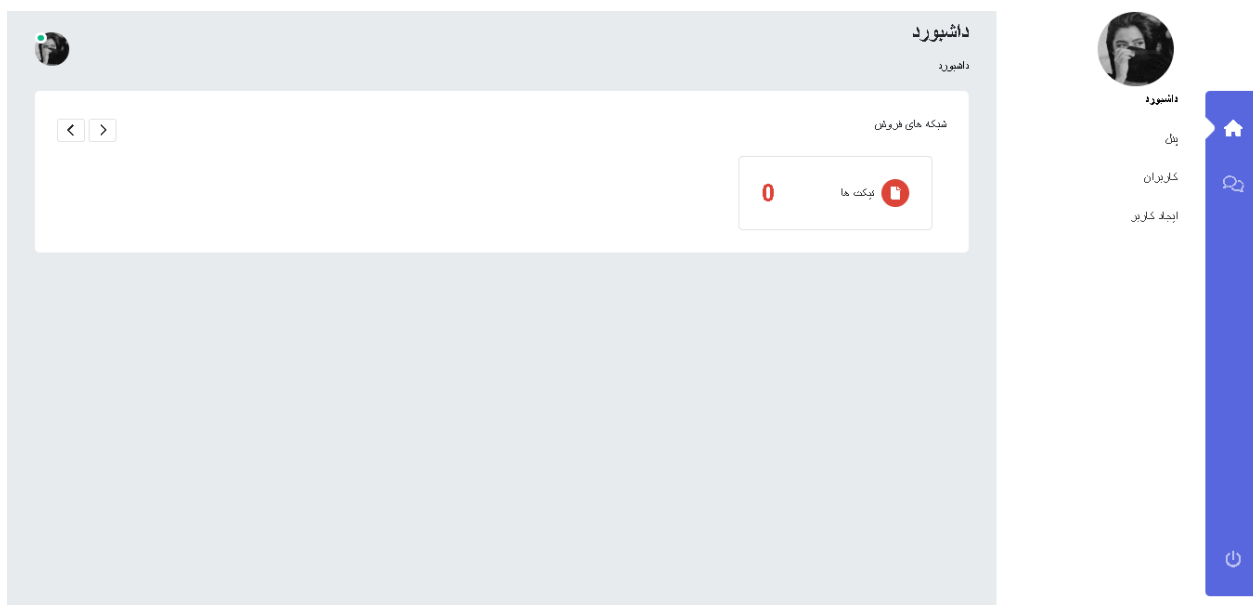
\*\*\*\*\*

☒ به خاطر بساز

ورود

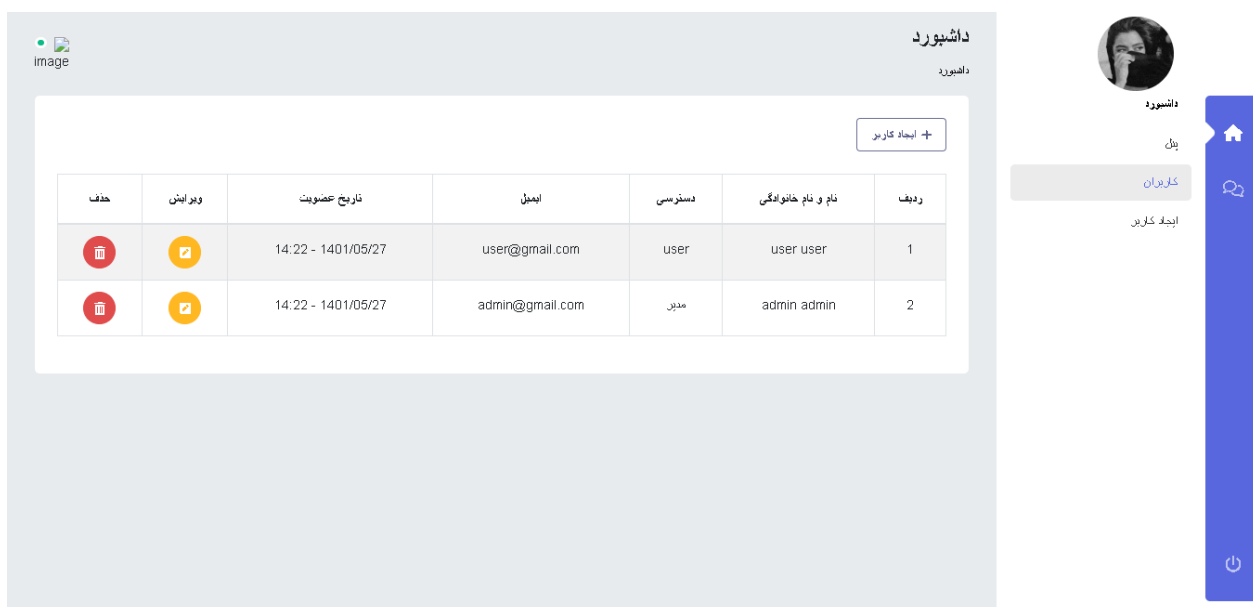
این صفحه شامل دو فیلد یوزر و پسورد میباشد که با وارد کردن مقادیر درست وارد پنل سایت می شوید.  
در صورت وارد کردن مقادیر اشتباه با ارور **نام کاربری یا پسورد اشتباه است** مواجه می شوید.

پنل:



در این صفحه تعداد تیکت های موجود را مشاهده میکنید و از سایدبار سمت راست میتوانید به امکانات دیگر سایت دسترسی داشته باشید.

کاربران:



در این صفحه جدولی را مشاهده میکنید که تمام کاربران سایت در آن قابل نمایش است و با کلیک روی گزینه ویرایش میتوانید اطلاعات کامل همان کاربر را دیده و ویرایش کنید. همچنین با کلیک

روی دکمه حذف میتوانید آن کاربر را از جدول و پایگاه داده حذف کنید (این عملیات به صورت زنده و ایجکسی اتفاق می افتد).

با کلیک بر روی دکمه ایجاد کاربر وارد فرم ایجاد می شوید و میتوانید با پر کردن فرم کاربر جدید اضافه کنید.

ایجاد کاربر:

پس از پر کردن این فرم و مشخص کردن سطح دسترسی کاربر (مدیر – کاربر) میتوانید کاربر را با زدن روی دکمه ذخیره اضافه کنید. مقادیر مورد نیاز توسط **Validation** صحت سنجی می شوند. پسورد وارد شده در کنترلر هس می شود.

ویرایش کاربر:

داشبورد

داشبورد

ویرایش کاربر

نام

نام خانوادگی

سطح دسترسی

ایمیل

پسورد

ثبت ✓

داشبورد

پنل

کاربران

ایجاد کاربر

با وارد کردن فیلدها، همان کاربری که قصد ویرایشش را دارید، ویرایش می شود.  
**موارد امنیتی در ایجاد کاربر هم در این قسمت اجرا می شوند.**

حذف کاربر:

داشبورد

داشبورد

+ ایجاد کاربر

حذف	ویرایش	تاریخ عضویت
		۱۴۰۲/۰۵/۲۷ - ۱۴۰۲/۰۵/۲۷
		۱۴۰۲/۰۵/۲۷ - ۱۴۰۲/۰۵/۲۷

حذف شود؟

لغو حذفش کن

داشبورد

پنل

کاربران

ایجاد کاربر

با زدن روی گزینه حذف، یک **sweet alert** جهت اطمینان از این عملیات نمایش داده می شود و با زدن روی گزینه **حذفش کن**، سطر مورد نظر به صورت **ایجکسی** حذف می شود.

## تیکت ها:

داشبورد

داشبورد

پشتیبانی

تیکت ها

خانه

تیکت ها

پاور

ایجاد تیکت

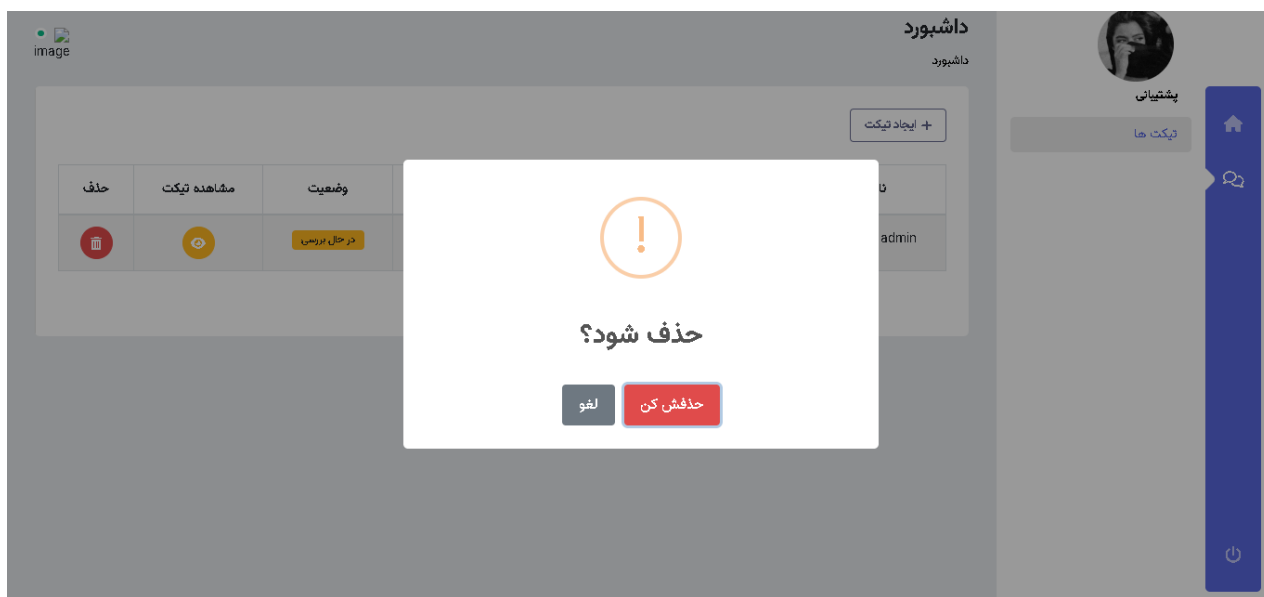
نام	موضوع	تاریخ	ساعت	کد	وضعیت	مشاهده تیکت	حذف
admin admin	test	۱۴۰۱/۰۵/۲۷	۱۶:۱۱	#۶۵۹۸۸۴۷۵۳۴	در حال بررسی		

در این صفحه شما لیست تیکت هاتون رو مشاهده میکنید و در صورتی که ادمین باشید تمام تیکت ها رو مشاهده میکنید ولی در صورتی که کاربر باشید، فقط تیکت های مربوط به خودتون رو مشاهده می کنید.

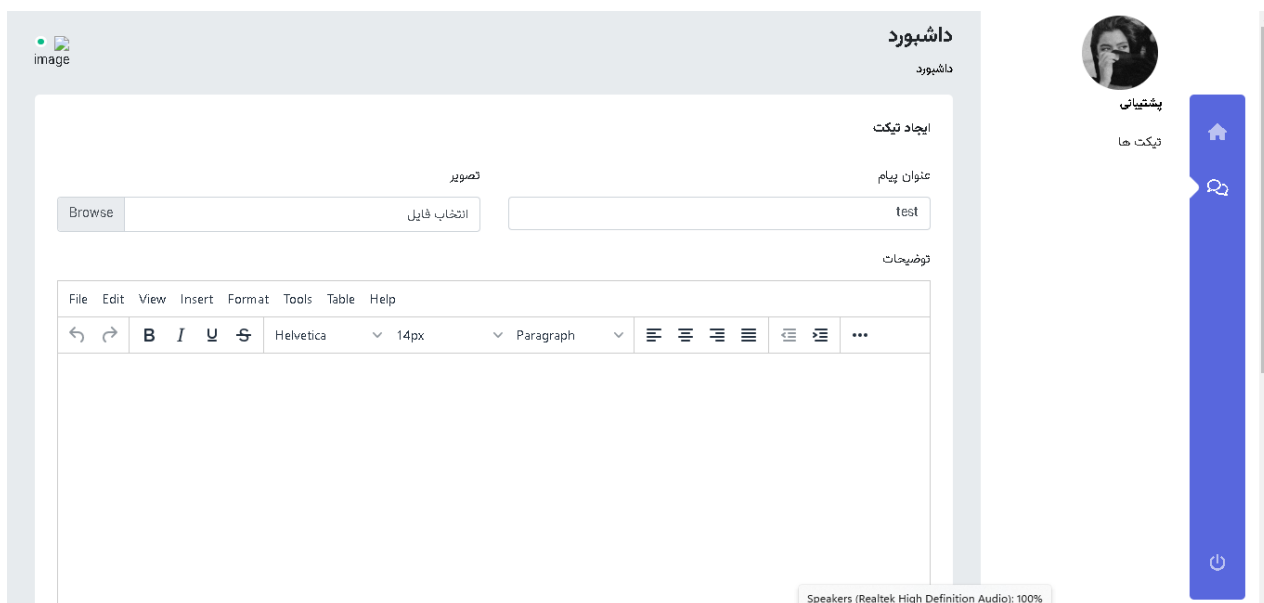
با زدن روی دکمه ایجاد تیکت وارد فرم ایجاد کردن تیکت می شوید.

با زدن روی دکمه مشاهده تیکت وارد صفحه چت تیکت می شوید.

با زدن روی دکمه حذف، یک **sweet alert** نمایش داده می شود که با زدن روی دکمه **حذفش کن**، تیکت به صورت **ایجکسی** حذف می شود.



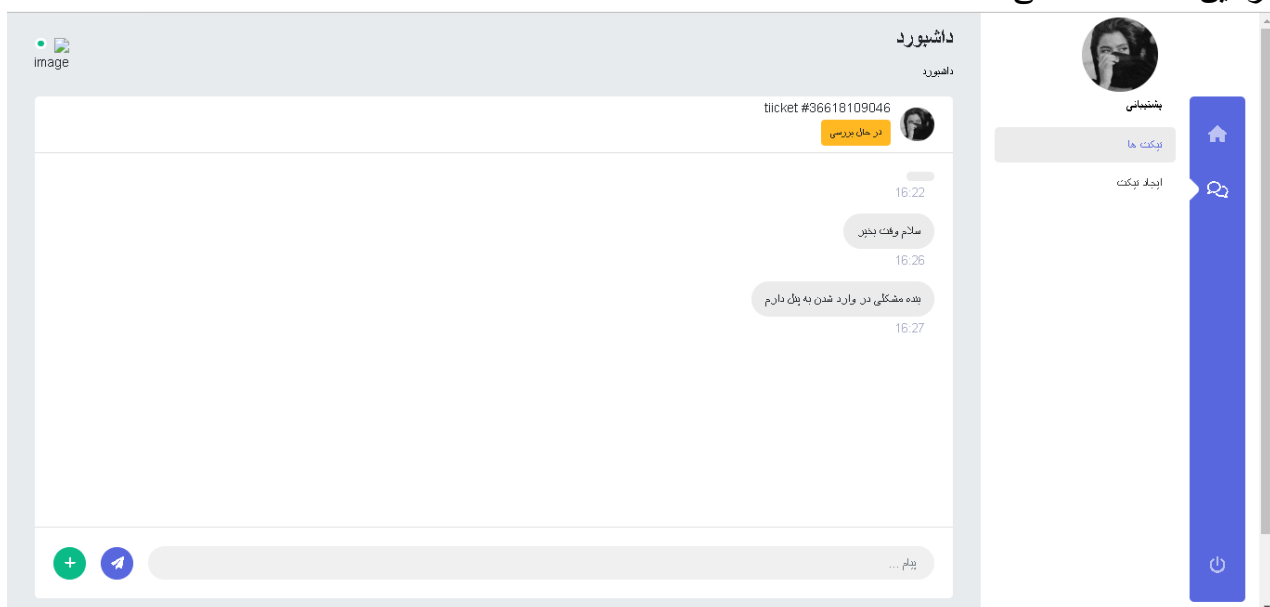
ایجاد تیکت:



در این صفحه با وارد کردن عنوان تیکت و پیام تیکت و یا تصویر ضمیمه، میتوانید تیکت را ارسال کنید. تیکت ارسالی به اولین ادمین ارجاع داده می شود.

چت تیکت:

در این صفحه شما می



توانید با اوپراتور مشخص شده چت کنید.  
همچنین می توانید با زدن گزینه + تصویر خود را ضمیمه کنید و ارسال کنید.  
اگر کاربر لاگین شده ادمین باشد می تواند وضعیت تیکت را به **در حال بررسی** یا **بسته شده** تغییر دهد.

مستندات کد ها :

صفحه داشبورد :

کانستراکت :

```

public function __construct()
{
    $this->middleware('auth');
}

```

در این قسمت با میدلور لاگین بودن یا نبودن کاربر را بررسی می کند. اگر کاربر لاگین نکرده باشد، آن را به صفحه لاگین ریدایرکت میکند.

ایندکس :

```

public function index()
{
    if(Gate::allows('Admin')){
        $tickets = Ticket::get();
    }else{
        $tickets = Ticket::where('sender_id', Auth::id())->orWhere('recive_id', Auth::id())->get();
    }
    return view('panel.index', compact('tickets'));
}

```

در این قسمت ابتدا سطح دسترسی کاربر را بررسی می کند. در صورتی که کاربر ادمین باشد، تمام تیکت هایی که در پایگاه داده هستند را درون یک متغیر ذخیره میکند. در صورتی که کاربر ادمین نباشد، فقط تیکت های ارسالی و دریافتی مربوط به آن کاربر را در متغیر ذخیره میکند و در انتها آن ها را به صفحه داشبورد انتقال میدهد.



نمایش کاربران:

```
public function index()  
{  
    if(Gate::allows('Admin')){  
        $data = User::orderByDesc('id')->paginate(10);  
        return view('panel.users.index',['data' => $data]);  
    }else{  
        abort(403);  
    }  
}
```

ابتدا دسترسی کاربر وارد شده بررسی می شود، اگر ادمین باشد، تمامی کاربران با مرتب سازی desc در صفحه index نمایش داده می شوند.

در صورتی که ادمین نباشد با ارور 403 که به منظور **دسترسی غیر مجاز** است مواجه میشود.

صفحه ایجاد کاربر:



```
public function create()
{
    if(Gate::allows('Admin')){
        return view('panel.users.create');
    }else{
        abort(403);
    }
}
```

با بررسی دسترسی کاربر، وارد صفحه ایجاد کاربر می شود، در غیر این صورت با ارور 403 مواجه می شود.

ذخیره کاربر:

```

public function store(User $user, CreateUsersRequest $request)
{
    if (Gate::allows('Admin')) {
        $user::create([
            'name' => $request->name,
            'family' => $request->family,
            'role' => $request->role,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);
        session()->flash('status', 'کاربر با موفقیت ایجاد شد');
        return redirect()->route('users.index');
    } else {
        abort(403);
    }
}

```

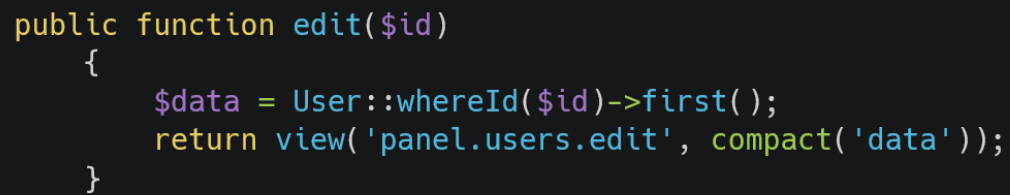
ابتدا دسترسی کاربر بررسی می شود و پس از بررسی، با الگوئنت **create** مقادیر وارد شده از ریکوئست در پایگاه داده ذخیره می شوند. سپس یک **سشن** با پیام **موفقیت آمیز** بودن عملیات ایجاد می شود.

پسورد وارد شده از سمت ریکوئست هش می شود.

پس از انجام این موارد، به **route** صفحه نمایش کاربران ریدایرکت (هدایت) می شود.

**نکته :** قبل از انجام این موارد، ریکوئست ها توسط ولیدیشن سفارشی صحتشان بررسی می شود.

ویرایش کاربر :



```
public function edit($id)
{
    $data = User::whereId($id)->first();
    return view('panel.users.edit', compact('data'));
}
```

پس از پیدا کردن کاربر با آیدی ارسال شده، فرم ویرایش را نمایش میدهد.

آپدیت کاربر :

```

public function update(User $user, Request $request)
{
    if (Gate::allows('Admin')) {
        $user->update([
            'name' => $request->name,
            'family' => $request->family,
            'role' => $request->role,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);

        session()->flash('status', 'کاربر با موفقیت ویرایش شد');
        return redirect()->route('users.index');
    }else{
        abort(403);
    }
}

```

پس از بررسی دسترسی ادمین، مقادیر وارد شده از ریکوئست وارد پایگاه داده می شوند سپس یکی سشن وضعیت ساخته شده و به روت صفحه جدول کاربران ریدایرکت می شود. موارد امنیتی لحاظ شده اما ولیدیشن به دلیل اختیاری بودن مقادیر در فرم ویرایش، لحاظ نشده است.

حذف کاربر :



```
public function destroy(User $user)
{
    if (Gate::allows('Admin')) {
        $user->delete();
        return back();
    }else{
        abort(403);
    }
}
```

پس از بررسی دسترسی ادمین، کاربر مربوطه حذف شده و صفحه قبلی را برمیگرداند.

تیکت ها :

جدول نمایش تمام تیکت ها :

```

public function index()
{
    if(Gate::allows('User'))
    {
        $tickets = Ticket::where("chid" , 0)->where(function($query){
            $query->Orwhere("receive_id" , auth()->user()->id);
            $query->Orwhere("send_id" , auth()->user()->id);
        }->orderByDesc("id")->paginate(10);
    }else{
        $tickets = Ticket::where("chid" , 0)->orderByDesc("id")->paginate(10);
    }
    return view("panel.tickets.index" , ['data' => $tickets]);
}

```

در این قسمت ابتدا سطح دسترسی کاربر لاگین شده بررسی می شود. در صورتی که کاربر ادمین نباشد، تیکت های ارسالی و دریافتی مربوط به همان کاربر توسط الگونت به صورت معکوس مرتب شده و پیجینیت میشود و درون یک متغیر ذخیره می شود. در صورتی که کاربر ادمین باشد تمام تیکت ها به همین صورت درون یک متغیر ذخیره می شوند و به صفحه جدول تیکت ها منتقل می شوند.

صفحه ایجاد تیکت :

```

public function createTicket()
{
    $receive_id = User::where('role', 'admin')->first()->id;
    return view("panel.tickets.create" , ["receive_id" => $receive_id]);
}

```

ابتدا اولین ادمین جهت ارجاع تیکت گرفته می شود و آیدی آن داخل یک متغیر ذخیره می شود. سپس به صفحه ایجاد تیکت انتقال پیدا می کند.

```

public function store(Request $request)
{
    if($request->image){
        $img = UploadImg($request->image , "tickets");
    }

    $ticket = Ticket::create([
        "send_id" => auth()->user()->id,
        "receive_id" => $request->receive_id,
        "chid" => $request->chid,
        "title" => $request->title,
        "message" => $request->message,
        "image" => $img ?? "",
        "chid" => 0,
        "status" => $request->status ?? "pending",
        "code" => generateRandom(10),
    ]);

    $ticket->update([
        "ticket_id" => $ticket->id
    ]);

    return redirect()->route("tickets.edit" , $ticket->id);
}

```

ابتدا وجود عکس ضمیمه را در ریکوئست بررسی کرده و در صورت وجود از متد **UploadImg** که در هلیپر ساخته شده است جهت آپلود فایل استفاده می شود. پارامتر اول این متد فایل دریافتی از ریکوئست می باشد و پارامتر دوم این متد، نام پوشه ای که در آن باید ذخیره شود می باشد. سپس یک تیکت با مقادیر دریافت شده ساخته می شود. در این عملیات اگر فایل ضمیمه دریافت نشده باشد، این فیلد خالی میماند. اگر وضعیت تیکت دریافت نشده باشد، وضعیت در حال بررسی قرار میگیرد. برای کد تیکت از متد **generateRandom** که در هلیپر ساخته شده است استفاده می کنیم. این متد یک پارامتر میگیرد که تعداد رشته کد را تعیین میکند.



سپس فیلد `ticket_id` با الگونت آپدیت می شود و آیدی تیکت در آن قرار میگیرد.  
این آپدیت به این دلیل هست که تفاوت پیام تیکت با خود تیکت مشخص شود.  
سپس آیدی تیکت به صفحه چت تیکت انتقال پیدا می کند.

صفحه چت تیکت :

```
public function edit(Ticket $ticket)
{
    if(auth()->user()->id == $ticket->receive_id || auth()->user()->id == $ticket->send_id)
    {
        $tickets = Ticket::where("ticket_id" , $ticket->id)->orderBy("id" , "asc")->get();
        return view("panel.tickets.edit" , ["tickets" => $tickets , "data" => $ticket]);
    }else{
        abort(403);
    }
}
```

ابتدا آیدی دریافت کننده و ارسال کننده بررسی می شود که جز این دو کاربر، کاربران دیگر ای به این صفحه دسترسی نداشته باشند.  
سپس چت های مربوط به این تیکت مرتب می شوند و به ویو چت انتقال پیدا می کند.

ارسال پیام در تیکت :

```

public function update(Ticket $ticket, Request $request)
{
    if($request->image){
        $img = UploadImg($request->image , "tickets");
    }

    $data = Ticket::where("chid" , 0)->where("id" , $ticket->id)->first();

    $data->update([
        "status" => "pending"
    ]);

    Ticket::create([
        "title" => $ticket->title,
        "send_id" => auth()->user()->id,
        "receive_id" => $request->receive_id,
        "ticket_id" => $ticket->id,
        "message" => $request->message ?? "",
        "image" => $img ?? "",
        "status" => $request->status ?? "pending",
        "code" => $ticket->code,
        "chid" => $ticket->id,
    ]);

    return back();
}

```

پس از ارسال پیام در تیکت مورد نظر، ابتدا وجود عکس را بررسی می کند و در صورت موجود بودن عکس در ریکوئست دریافتی، آن را آپلود می کند. سپس تیکت مربوطه را گرفته و وضعیت آن را به درحال بررسی تغییر میدهد.

سپس یک تیکت ساخته و **ticket\_id** و **chid** را پر میکند.  
سپس به صفحه چت تیکت مجدد برمیگرداند.

حذف تیکت :

```

public function destroy(Ticket $ticket)
{
    $ticket->delete();
    return back();
}

```

در این قسمت ابتدا تیکت مربوطه را به عنوان پارامتر دریافت کرده و سپس حذف میکند و صفحه قبلی را برمیگرداند.

تغییر وضعیت تیکت :

```

public function changeStatus($id)
{
    $data = Ticket::where("chid" , 0)->where("ticket_id" , $id)->first();

    $data->update([
        "status" => request()->status
    ]);

    return back();
}

```

در این قسمت ابتدا تیکت مربوطه را با پارامتر آیدی پیدا کرده و درون یک متغیر ذخیره میکند.

سپس وضعیت تمان تیکت را نسبت به وضعیت ریکوئست دریافت شده آپدیت میکند و پس از آن به صفحه قبل باز میگردد. این عملیات به صورت ایجکسی انجام می شود.

هلیپر :

هلیپر برای تعریف متد هایی ساخته شده است که عملیات های تکراری را بدون تکرار کد با حجم بالا هندل میکند. متد پیدا کردن کاربر :

```
<?php

use App\Models\User;
use Carbon\Carbon;
use Illuminate\Support\Facades\Auth;

if (!function_exists('findUser')) {
    function findUser($info = true, $id = null, $parameter = 'id')
    {
        if($id == null){
            $id = Auth::id();
        }
        if (empty($id)) {
            $output = 'تعریف نشده است';
        } else {
            $user = User::where('id', $id)->first();
            if ($user != null) {
                if ($info) {
                    $output = $user->name . ' ' . $user->family;
                } else {
                    $output = $user->$parameter;
                }
            } else {
                $output = 'تعریف نشده است';
            }
        }
        return $output;
    }
}
```

ابتدا با یک شرط چک شده است که این فانکشن یا متد قبلا وجود داشته است یا خیر، اگر وجود نداشته باشد آن را ایجاد میکند. این فانکشن سه پارامتر میگیرد.

در ابتدای این هلیپر چک می شود که پارامتر آیدی **null** است یا خیر، اگر خالی باشد کاربر لاگین شده را در خود ذخیره میکند در غیر این صورت آیدی خود پارامتر را در خود ذخیره میکند. سپس بررسی میکند که پارامتر آیدی خالی است یا خیر، اگر خالی باشد. پیام تعریف نشده است را در متغیر **output** ذخیره میکند، در غیر این صورت کاربر را نسبت به پارامتر آیدی پیدا میکند. سپس بررسی میکند که کاربر وجود داشته باشد در غیر این صورت پیام تعریف نشده است را در متغیر خروجی ذخیره میکند.

اگر کاربر وجود داشته باشد، بررسی میکند که پارامتر **info** در وضعیت **true** است یا خیر، اگر **true** باشد نام و نام خانوادگی کاربر را گرفته و در متغیر خروجی ذخیره میکند، ولی اگر پارامتر اینفو **false** باشد پارامتر **parameter** را از کاربر دریافت میکند که این پارامتر میتواند نام یکی از ستون های جدول کاربران باشد، سپس آن را درون متغیر خروجی ذخیره میکند.

در انتها متغیر خروجی را برمیگرداند.

متد آپلود عکس یا فایل :

```
if (!function_exists('UploadImg')) {
    function UploadImg($file , $name)
    {
        if ($file) {
            $filename = time() . $file->getClientOriginalName();
            $year = Carbon::now()->year;
            $month = Carbon::now()->month;
            $path = "uploads/{$name}/{ $year}/{ $month}/";
            $file->move($path, $filename);
            $img = "/uploads/{$name}/{ $year}/{ $month}/" . $filename;
            return $img;
        }
    }
}
```

ابتدا بررسی میشود که متد وجود دارد یا خیر.

این متد دو پارامتر میگیرد. ابتدا بررسی میشود پارامتر اول که فایل است خالی است یا خیر، اگر خالی نباشد نام اصلی فایل را گرفته و یک تایم استمپ زمان حال حاضر به اول آن اضافه میکند و آن را درون یک متغیر با نام **filename** ذخیره میکند.

سپس سال و ماه را گرفته و در متغیر های مربوطه ذخیره میکند.  
سپس یک متغیر به نام **path** ساخته و مسیر آپلود فایل را به همراه پوشه نام و سال و ماه رو ذخیره میکند.

سپس فایل مربوطه را با نام اصلی فایل در مسیر تعریف شده ذخیره میکند و در انتها یک متغیر به نام **img** ساخته شده و تمام مسیر فایل و نام اصلی فایل را در آن ذخیره میکند و آن را برمیگرداند.

متد ساخت عدد تصادفی :

```
if (!function_exists('generateRandom')) {  
    function generateRandom($len=10)  
    {  
        $number = range("0" , "9");  
        $password = "";  
        for($i = 0; $i <= $len; $i++)  
        {  
            $password .= $number[rand(0 , count($number) -1)];  
        }  
        return $password;  
    }  
}
```

ابتدا وجود متد بررسی میشود سپس متد با یک پارامتر ایجاد می شود.  
سپس یک عدد تصادفی ساخته می شود و درون یک متغیر ذخیره می شود.  
سپس یک حلقه ایجاد شده و یک عدد تصادفی در یک متغیر به نام پسورد ذخیره میشود و این حلقه تا زمانی که شمارنده آن به عدد پارامتر دریافتی نرسیده است متوقف نمیشود.  
**نکته : پارامتر دریافتی نشان دهنده طول دلخواه عدد تصادفی می باشد.**

خروجی هلیو :

```

<?php

use App\Models\User;
use Carbon\Carbon;
use Illuminate\Support\Facades\Auth;

if (!function_exists('findUser')) {
    function findUser($info = true, $id = null, $parameter = 'id')
    {
        if($id == null){
            $id = Auth::id();
        }
        if (empty($id)) {
            $output = 'تعریف نشده است';
        } else {
            $user = User::where('id', $id)->first();
            if ($user != null) {
                if ($info) {
                    $output = $user->name . ' ' . $user->family;
                } else {
                    $output = $user->$parameter;
                }
            } else {
                $output = 'تعریف نشده است';
            }
        }
        return $output;
    }
}

if (!function_exists('UploadImg')) {
    function UploadImg($file , $name)
    {
        if ($file) {
            $filename = time() . $file->getClientOriginalName();
            $year = Carbon::now()->year;
            $month = Carbon::now()->month;
            $path = "uploads/{$name}/{ $year}/{ $month}/";
            $file->move($path, $filename);
            $img = "/uploads/{$name}/{ $year}/{ $month}/" . $filename;
            return $img;
        }
    }
}

if (!function_exists('generateRandom')) {
    function generateRandom($len=10)
    {
        $number = range("0" , "9");
        $password = "";
        for($i = 0; $i <= $len; $i++)
        {
            $password .= $number[rand(0 , count($number) -1)];
        }
        return $password;
    }
}

```

روت ها :

ایندکس :

```
Route::get('/', function(){
    return redirect()->route('panel');
})->name('index');
```

به دلیل نداشتن فرانت سایت اصلی، این روت کاربر را به روت پنل هدایت می کند.

گروه بندی روت ها :

```
Route::group(['middleware' => 'auth' , 'prefix' => "panel"], function(){
    //Routes
});
```

این گروه بندی شامل یک میدلور می شود که لاگین بودن کاربر را بررسی کند، روت هایی که در این گروه قرار میگیرند نیازمند لاگین بودن کاربر هستند. همچنین یک پیشوند پنل دریافت میکنند، روت هایی که در این گروه هستند در قسمت پنل سایت قرار دارند.



پنل :

```
Route::get('/',[HomeController::class, 'index']->name('panel'));
```

این روت کاربر را به متد ایندکس HomeController انتقال میدهد.

کاربران :

```
Route::resource('users', UserController::class);
```

این روت ریسورس است که روت های صفحات جدول و ایجاد و ویرایش را ایجاد میکند. علاوه بر آن روت ذخیره سازی و آپدیت و حذف کاربر را نیز ایجاد میکند.

تیکت ها :

```
Route::resource('tickets', TicketController::class);
Route::get("tickets/create" , [TicketController::class , "createTicket"]->name("tickets.createTicket"));
Route::group(['middleware' => 'can:Admin'],function(){
    Route::post("tickets/changeStatus/{id}" , [TicketController::class , "changeStatus"]->name("tickets.changeStatus"));
});
```

ابتدا یک گروه بندی ساخته شده که بتوان میدلور Gate را در آن اعمال نمود. سپس مشخص میکند که فقط سطح دسترسی ادمین به این روت ها دسترسی دارند.

سپس یک روت ریسورس برای تیکت ساخته شده که روت های مربوطه به ریسورس را ایجاد میکند.

سپس یکی روت تغییر وضعیت تیکت مشاهده می کنید که از نوع post می باشد، این روت ریکوئست دریافت شده را به متد تغییر وضعیت کنترلر تیکت ارسال میکند.

روت بعدی روتی می باشد که کاربر را به صفحه ایجاد تیکت انتقال میدهد.

روت های لاگین :

```
Route::get('login', [LoginController::class, 'showLoginForm'])->name('loginForm');
Route::post('login', [LoginController::class, 'login'])->name('login');
Route::post('logout', [LoginController::class, 'logout'])->name('logout');
```

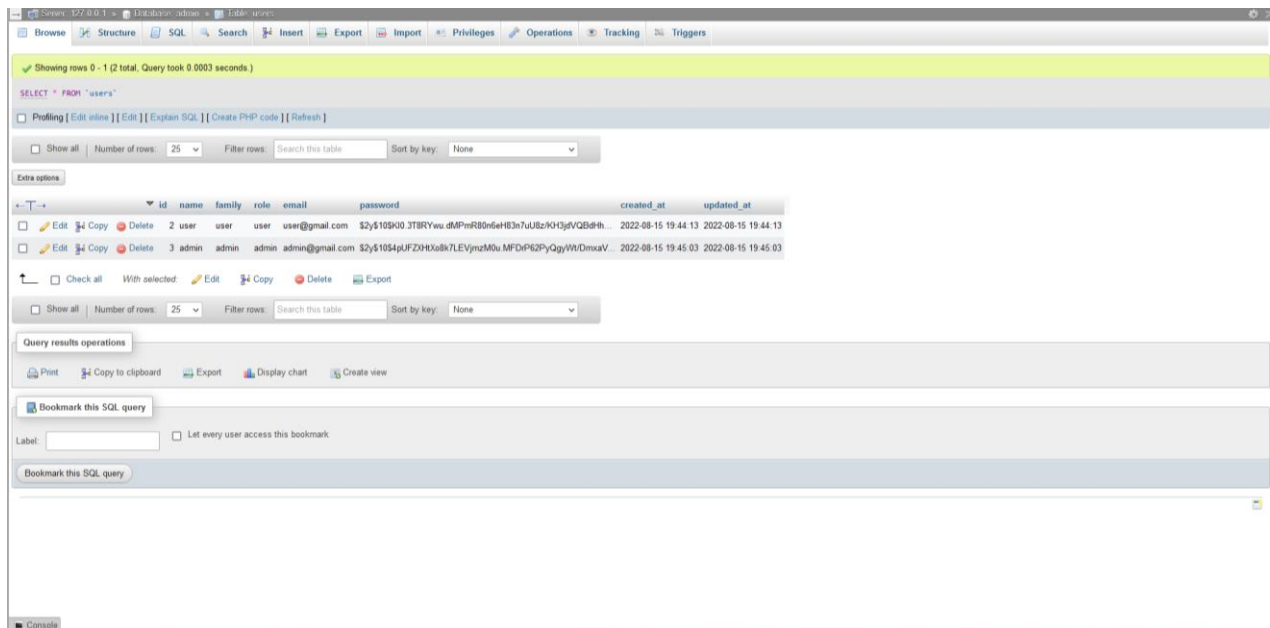
روت اول کاربر را به متد مربوط به صفحه لاگین هدایت می کند.

روت دوم ریکوئست دریافتی برای لاگین شدن را به متد لاگین LoginController ارسال میکند.

روت سوم ریکوئست دریافتی را به متد لاگ اوت کنترلر لاگین ارسال میکند.

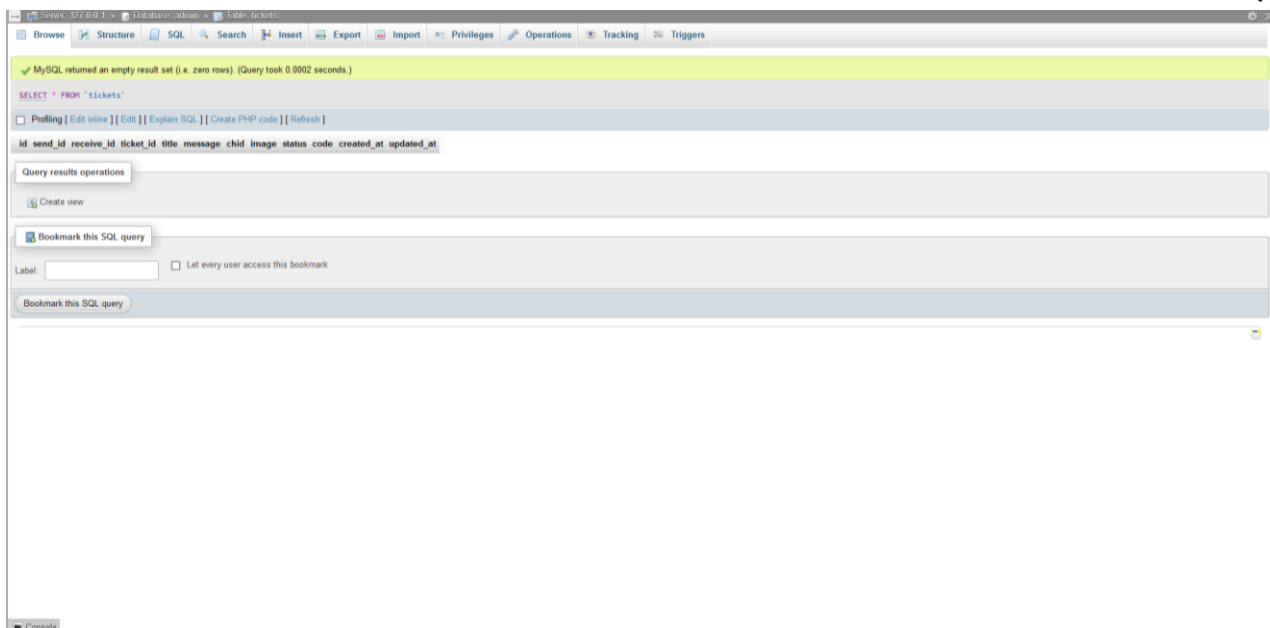
پایگاه داده :

کاربران :



در این جدول فیلد های نام، نام خانوادگی، سطح دسترسی، ایمیل، پسونرد و تاریخ ساخت اکانت و آپدیت اکانت می باشد.

تیکت ها :



این جدول شامل آیدی ارسال کننده، آیدی دریافت کننده، آیدی تیکت، عنوان تیکت، پیام تیکت، وضعیت تیکت، کد تیکت، زمان ساخت و آپدیت تیکت می باشد.

اگر تیکت برای بار اول ایجاد شود، فیلد 0 chid میماند ولی اگر پیام داخل تیکت ارسال شود، در فیلد chid آیدی تیکت والد قرار میگیرد.