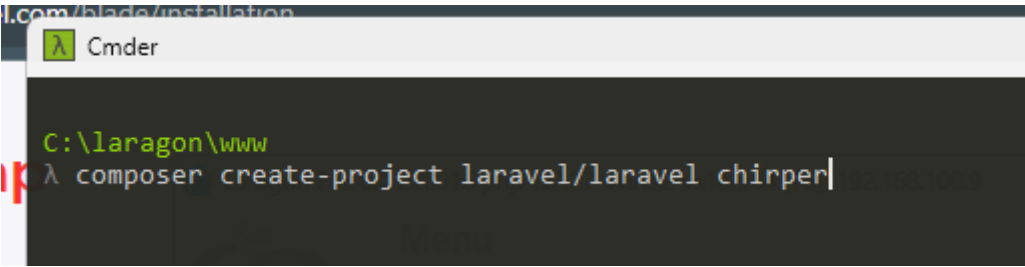


1. Installation

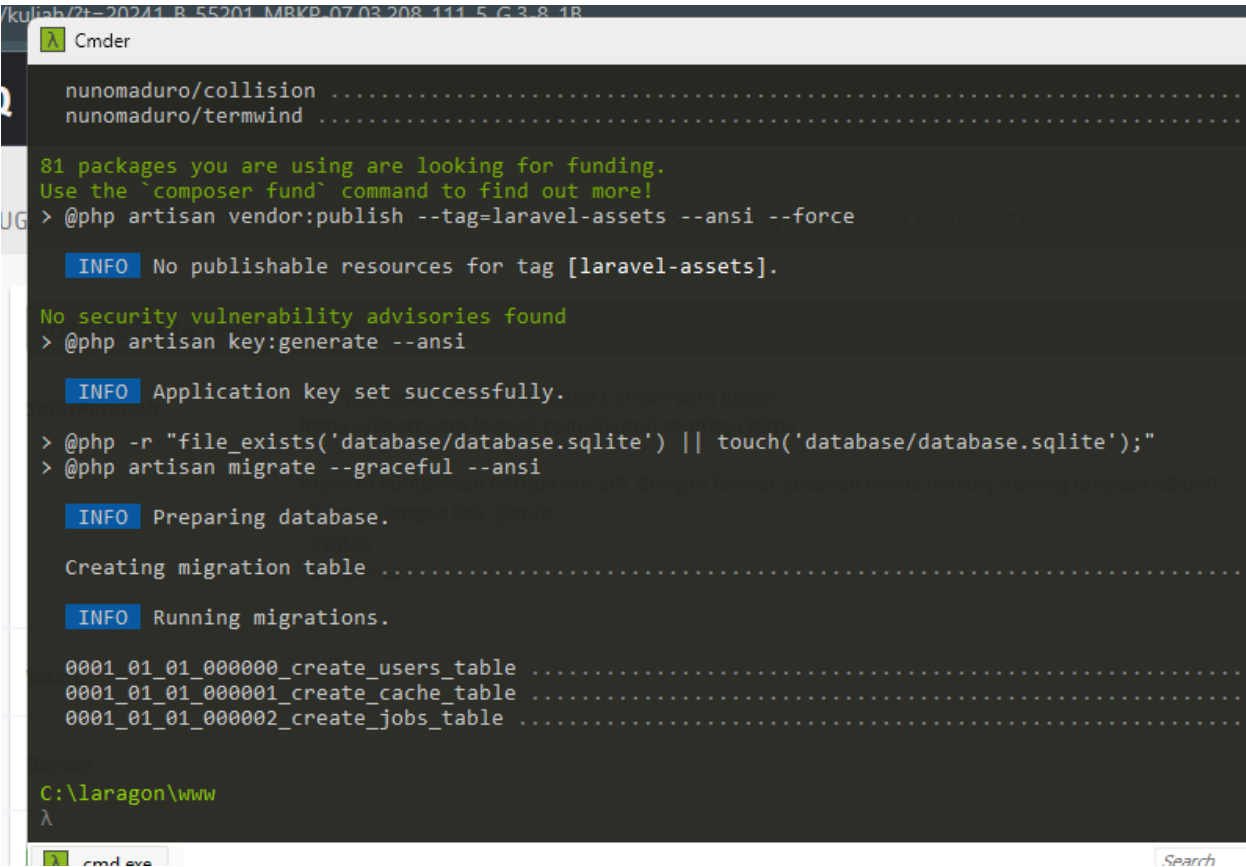
Installing Laravel

Jika sudah memiliki laragon, menginstall php version, menginstall composer lanjutkan dengan menginstall laravel via composer/ cmd di laragon



```
C:\laragon\www
λ composer create-project laravel/laravel chirper
```

Syntax diatas merupakan menginstall project laravel dengan nama project chirper, enter kemudian tunggu proses selanjutnya



```
nunomaduro/collision .....
nunomaduro/termwind .....

81 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
INFO No publishable resources for tag [laravel-assets].

No security vulnerability advisories found
> @php artisan key:generate --ansi
INFO Application key set successfully.

> @php -r "file_exists('database/database.sqlite') || touch('database/database.sqlite');"
> @php artisan migrate --graceful --ansi
INFO Preparing database.

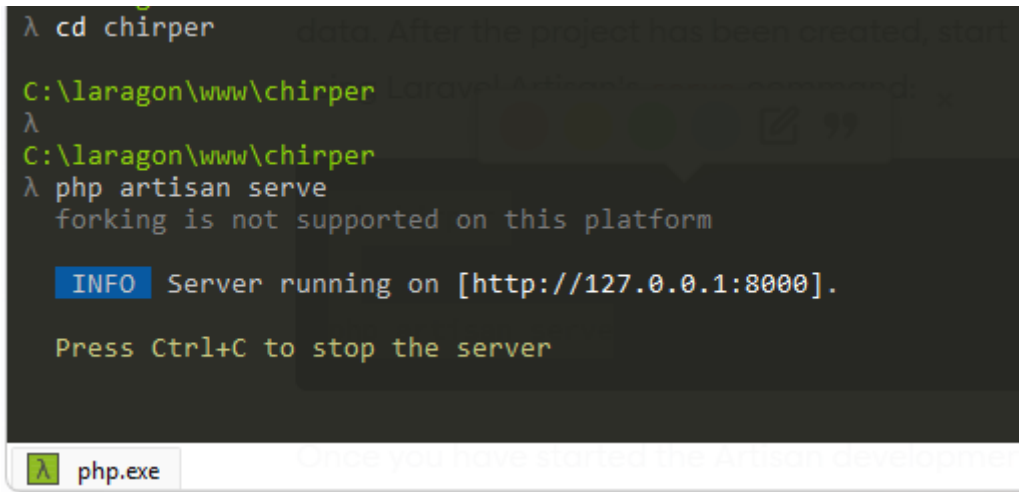
Creating migration table .....

INFO Running migrations.

0001_01_01_000000_create_users_table .....
0001_01_01_000001_create_cache_table .....
0001_01_01_000002_create_jobs_table .....

C:\laragon\www
λ
```

Proses diatas menandakan laravel sudah terinstall di folder chirper



```
λ cd chirper

C:\laragon\www\chirper
λ
C:\laragon\www\chirper
λ php artisan serve
forking is not supported on this platform

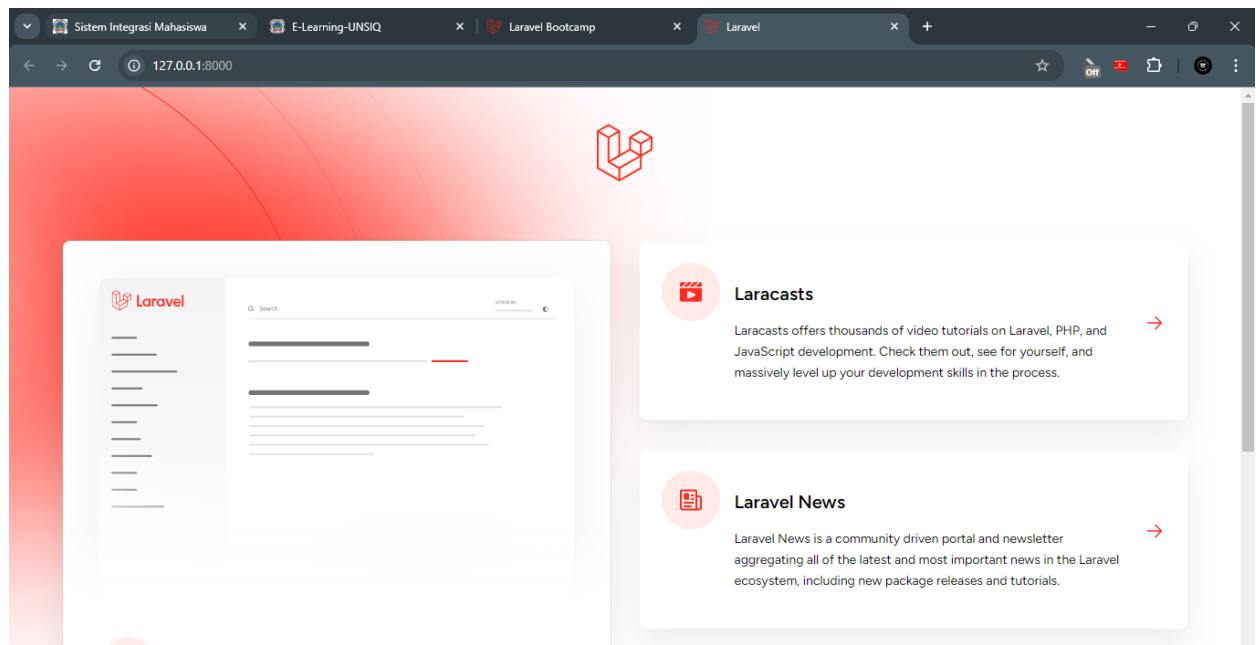
INFO Server running on [http://127.0.0.1:8000].

Press Ctrl+C to stop the server

λ php.exe
```

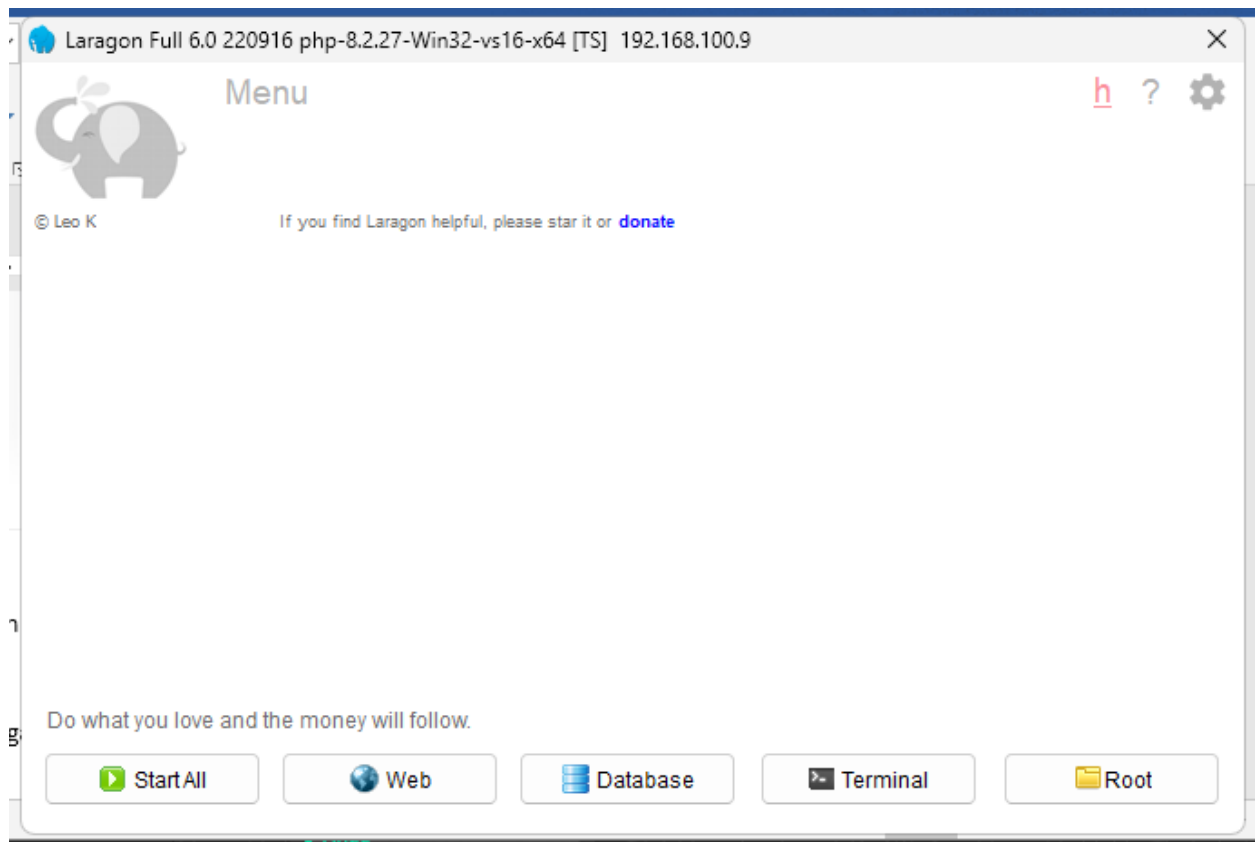
Cd chirper digunakan untuk menuju folder htdocs/chirper

Ahmad Gozali – NIM 20240150175 – Program Studi Teknik Informatika – **Building Chirper with Blade**
Php Artisan serve merupakan syntax dimana mengaktifkan laravel/webserver yang terinstall kemudian mengakses di [http://127.0.0.1:8000]



Dengan ini laravel sudah terinstall dan siap untuk digunakan.

Bisa juga apabila menggunakan laragon mengaktifkan dengan cara :



Klik start all

Kemudian klik kanan > www > pilih chirper

Installing Laravel Breeze

Di terminal buka folder project chirper dan install laravel/breeze

Gunakan syntax composer require laravel/breeze --dev

Untuk menginstall laravel/breeze dan paket di install hanya dalam lingkungan pengembangan

```
C:\laragon\www\chirper
λ composer require laravel/breeze --dev
Using version ^2.3 for laravel/breeze
./composer.json has been updated
Running composer update laravel/breeze
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking laravel/breeze (v2.3.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading laravel/breeze (v2.3.0)
- Installing laravel/breeze (v2.3.0): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

 INFO  Discovering packages.

laravel/breeze ..... DONE
laravel/pail ..... DONE
laravel/sail ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE

81 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

 INFO  No publishable resources for tag [laravel-assets].

No security vulnerability advisories found

C:\laragon\www\chirper
λ
```

Laravel/breeze sudah selesai diinstall kemudian masukkan syntax

```
php artisan breeze:install blade
```

digunakan untuk menginstall paket blade template sebagai stack front-end

```
C:\laragon\www\chirper
λ php artisan breeze:install blade

 INFO  Installing and building Node dependencies.

added 168 packages, and audited 169 packages in 28s
40 packages are looking for funding
run `npm fund` for details

found 0 vulnerabilities

> build
> vite build

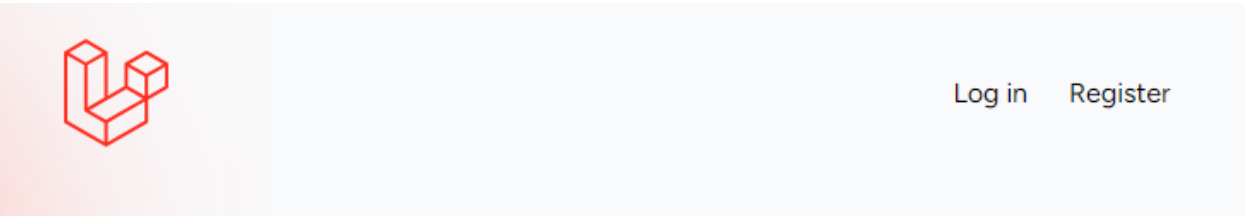
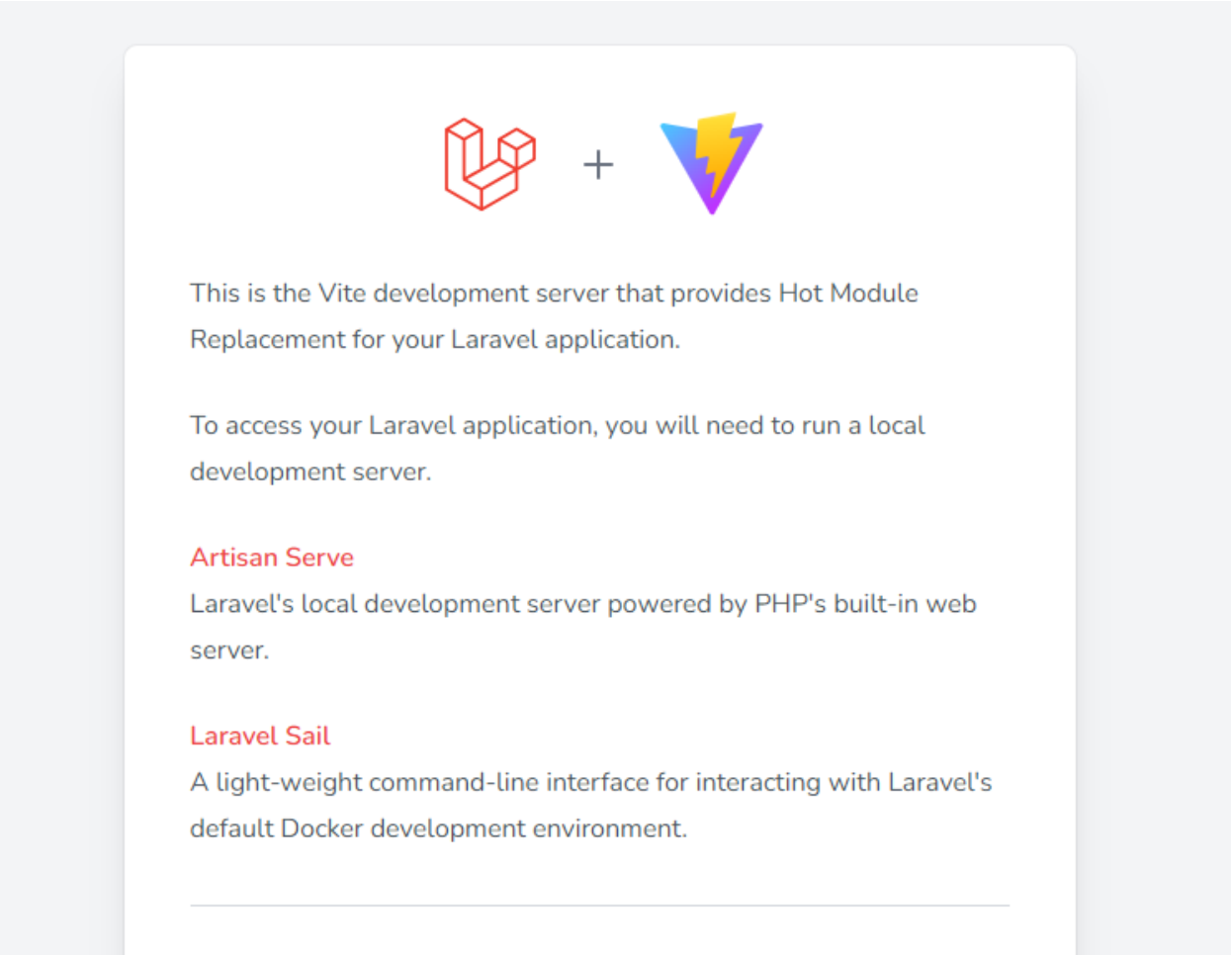
vite v6.0.6 building for production...
transforming...
✓ 54 modules transformed.
rendering chunks...
computing gzip size...
public/build/manifest.json 0.27 kB | gzip: 0.15 kB
public/build/assets/app-BZE5VnyZ.css 34.21 kB | gzip: 6.55 kB
public/build/assets/app-Cy98PJ2n.js 79.37 kB | gzip: 29.63 kB
✓ built in 6.32s

 INFO  Breeze scaffolding installed successfully.

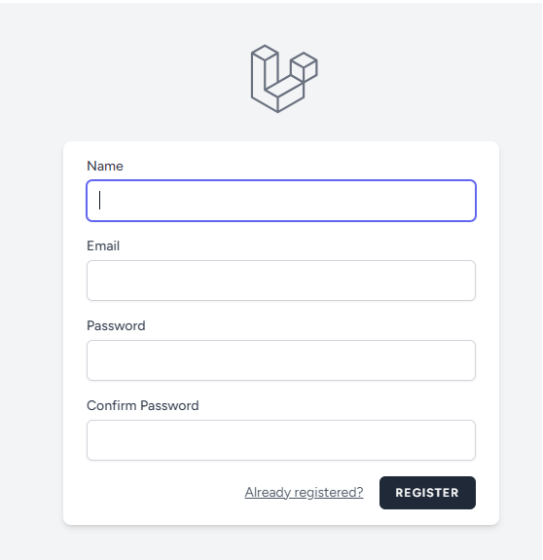
C:\laragon\www\chirper
λ
```

Package blade sudah di install kemudian dibutuhkan untuk menjalankan development vite untuk recompile css dan refresh browser untuk mengubah ke blade templates

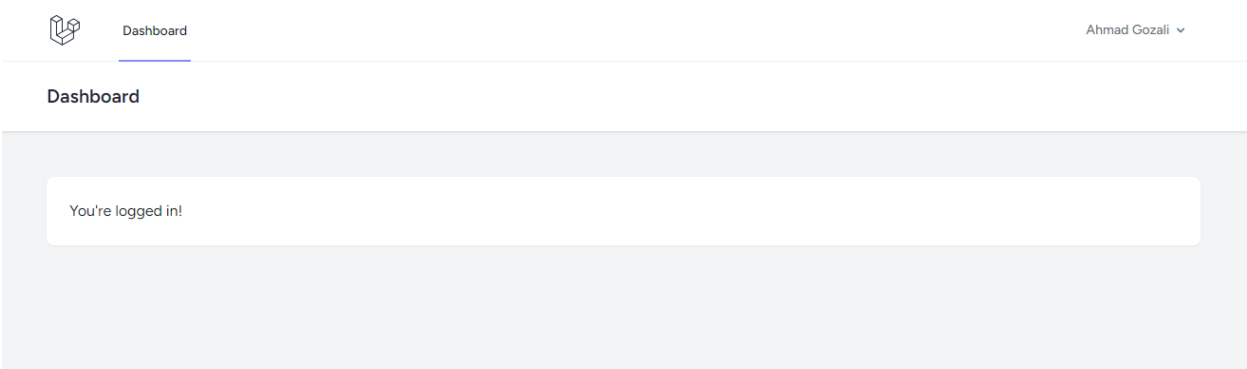
Syntax ini terkait dengan menjalankan template blade atau laravel sudah siap didevelopment



Akan muncul link register di link chirper.test



Ketika di klik akan muncul register seperti diatas. Isikan data keseluruhan kemudian akan muncul
Tampilan seperti dibawah ini



2. Creating Chirps

Model, Migrations, and Controller

Untuk memperbolehkan user posting di chirps, kita harus membuat model, migrations, dan controller

Untuk itu menggunakan syntax

```
php artisan make:model -mrc Chirp
```

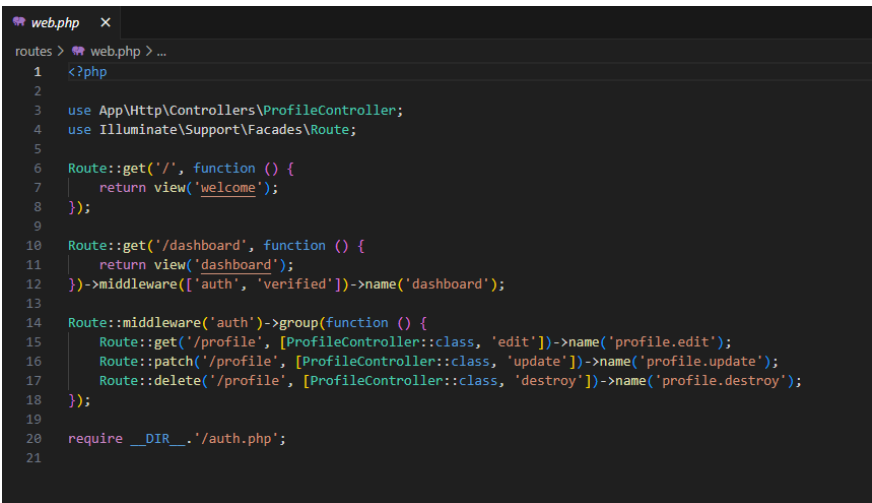
Syntax ini membuat tiga file

This command will create three files for you:

- app/Models/Chirp.php - The Eloquent model.
- database/migrations/<timestamp>_create_chirps_table.php - The database migration that will create your database table.
- app/Http/Controllers/ChirpController.php - The HTTP controller that will take incoming requests and return responses.

Routing

Untuk file di route/web.php tampilan awal seperti ini

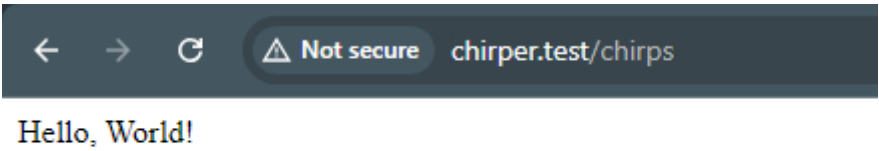



```

    * Display a listing of the resource.
    */
-   public function index()
+   public function index(): Response
    {
-       //
+       return response('Hello, World!');
    }
    ...

```

Yang tanda – dihilangkan kemudian diganti yang tanda + maka akan muncul tampilan



Blade

Masih belum terkesima coba ubahlah dengan menambahkan syntax BLADE ini dimana mengupdate index method ChirpController class untuk mrender BLADE view

```

app/Http/Controllers/ChirpController.php

<?php
...
+ use Illuminate\View\View;

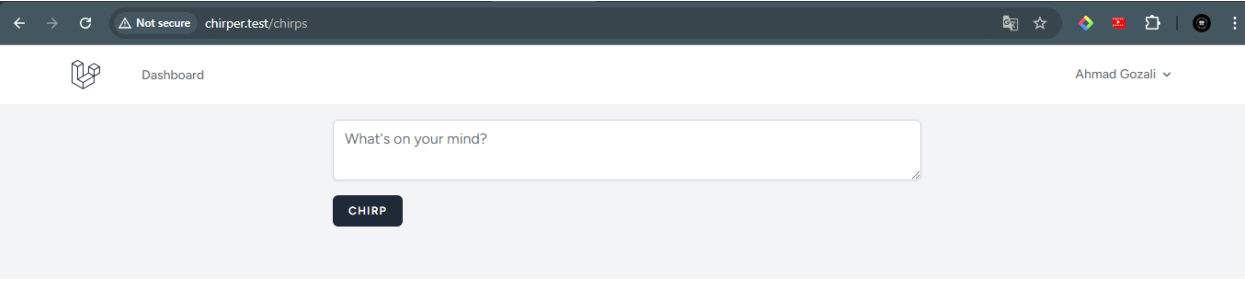
class ChirpController extends Controller
{
    /**
     * Display a listing of the resource.
     */
-   public function index(): Response
+   public function index(): View
    {
-       return response('Hello, World!');
+       return view('chirps.index');
    }
    ...
}

```

Kemudian kita dapat membuat blade view template menggunakan form untuk membuat Chirps baru

```
resources > views > chirps > index.blade.php
Copy File Path
1 <x-app-layout>
2   <div class="max-w-2xl mx-auto p-4 sm:p-6 lg:p-8">
3     <form method="POST" action="{{ route('chirps.store') }}">
4       @csrf
5       <textarea
6         name="message"
7         placeholder="{{ __('What's on your mind?') }}"
8         class="block w-full border-gray-300 focus:border-indigo-300 focus:ring focus:ring-indigo-200"
9       >{{ old('message') }}</textarea>
10      <x-input-error :messages="$errors->get('message')" class="mt-2" />
11      <x-primary-button class="mt-4">{{ __('Chirp') }}</x-primary-button>
12    </form>
13  </div>
14 </x-app-layout>
15
```

Kemudian refresh browser dan lihatlah apakah default layout provided by Breeze !



Navigation Menu

Tambahkan syntax di navigation.blade.php componen for desktop screens :

item for desktop screens:

```
resources/views/layouts/navigation.blade.php

<div class="hidden space-x-8 sm:-my-px sm:ml-10 sm:flex">
  <x-nav-link :href="route('dashboard')" :active="request()->routeIs('dashboard')">
    {{ __('Dashboard') }}
  </x-nav-link>
  + <x-nav-link :href="route('chirps.index')" :active="request()->routeIs('chirps.index')">
  +   {{ __('Chirps') }}
  + </x-nav-link>
</div>
```

Dan juga tambahkan untuk mobile screens


```
resources/views/layouts/navigation.blade.php

<div class="pt-2 pb-3 space-y-1">
    <x-responsive-nav-link :href="route('dashboard')" :active="request()-
        {{ __('Dashboard') }}
    </x-responsive-nav-link>
+   <x-responsive-nav-link :href="route('chirps.index')" :active="request
+       {{ __('Chirps') }}
+   </x-responsive-nav-link>
</div>
```

Saving Chirps

Form yang kita konfigurasi tadi post message di chirps.store route kita perbahuri store di ChirpController class untuk memvalidasi data dan membuat chirp baru

Tambahkan syntax/coding ini

```
use Illuminate\View\View;
use Illuminate\Http\RedirectResponse;
```

Dan tambahkan juga

```
public function store(Request $request): RedirectResponse
{
    $validated = $request->validate([
        'message' => 'required|string|max:255',
    ]);

    $request->user()->chirps()->create($validated);

    return redirect(route('chirps.index'));
}
```

Dengan koding diatas kita dapat mengembalikan response redirect untuk mengarahkan pengguna kembali ke rute chirps.index

Creating Relationship

Distep sebelumnya kita memanggil chirps methode di \$request->user() object. Kita membutuhkan untuk membuat metode ini di user model untuk mendefinisikan banyak hubungan/relationship

Tambahkan syntax dibawah ini

```
...  
+ use Illuminate\Database\Eloquent\Relations\HasMany;  
use Illuminate\Foundation\Auth\User as Authenticatable;  
use Illuminate\Notifications\Notifiable;  
use Laravel\Sanctum\HasApiTokens;  
  
class User extends Authenticatable  
{  
...  
+ public function chirps(): HasMany  
+ {  
+     return $this->hasMany(Chirp::class);  
+ }  
}
```

Mass Assignment Protection

Mengirimkan semua data dari sebuah request langsung ke model bisa menjadi berisiko. Bayangkan Anda memiliki halaman di mana pengguna dapat mengedit profil mereka. Jika Anda mengirim seluruh request ke model, maka pengguna dapat mengedit kolom apa pun yang mereka inginkan, seperti kolom `is_admin`. Hal ini disebut sebagai vulnerabilitas mass assignment.

Laravel melindungi Anda dari hal ini dengan memblokir mass assignment secara default. Namun, mass assignment sangat praktis karena memungkinkan Anda untuk tidak perlu menetapkan setiap atribut satu per satu. Kita dapat mengaktifkan mass assignment untuk atribut yang aman dengan menandainya sebagai "fillable".

Mari kita tambahkan properti `$fillable` ke dalam model Chirp untuk mengaktifkan mass assignment pada atribut message:

```
class Chirp extends Model  
{  
    protected $fillable = [  
        'message',  
    ];  
}
```

Updating Migration

Selama pembuatan aplikasi, Laravel sudah menerapkan migrasi default yang disertakan di dalam direktori **database/migrations**. Anda dapat memeriksa struktur database saat ini dengan menggunakan perintah berikut pada commandprompt :

Php artisan db:show

```
C:\laragon\www\chirper
λ php artisan db:show

SQLite ..... 3.39.2
Connection ..... sqlite
Database ..... C:\laragon\www\chirper\database\database.sqlite
Host .....
Port .....
Username .....
URL .....
Open Connections .....
Tables ..... 9

Table ..... Size
cache ..... -
cache_locks ..... -
failed_jobs ..... -
job_batches ..... -
jobs ..... -
migrations ..... -
password_reset_tokens ..... -
sessions ..... -
users ..... -
```

Php artisan db:table users

```
C:\laragon\www\chirper
λ php artisan db:table users

users .....
Columns ..... 8

Column ..... Type
id integer, autoincrement ..... integer
name varchar ..... varchar
email varchar ..... varchar
email_verified_at datetime, nullable ..... datetime
password varchar ..... varchar
remember_token varchar, nullable ..... varchar
created_at datetime, nullable ..... datetime
updated_at datetime, nullable ..... datetime

Index .....
primary id ..... primary
users_email_unique email ..... unique
```

Jadi, satu-satunya yang kurang adalah kolom tambahan di database kita untuk menyimpan relasi antara **Chirp** dan **User**, serta pesan itu sendiri. Ingat migrasi database yang kita buat sebelumnya? Sekarang saatnya membuka file tersebut untuk menambahkan beberapa kolom tambahan:

```
{
    Schema::create('chirps', function (Blueprint $table) {
        $table->id();
        $table->foreignId('user_id')->constrained()->cascadeOnDelete();
        $table->string('message');
        $table->timestamps();
    });
}
```

Dan kita belum memigrasi maka ketik syntax migrasi di command prompt

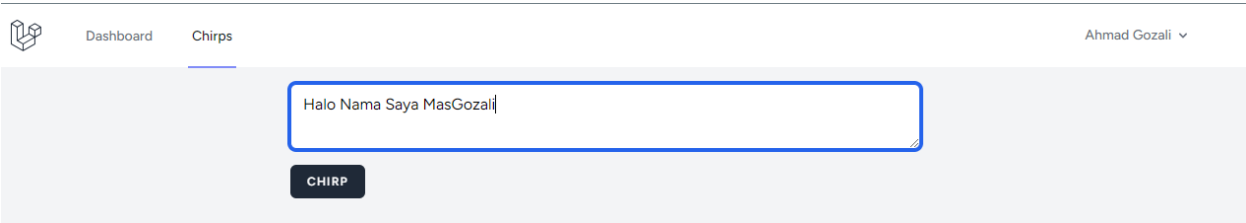
```
C:\laragon\www\chirper
λ php artisan migrate

INFO Running migrations.
2024_12_26_145138_create_chirps_table ..... 19.44ms DONE
```

Harap diperhatikan Setiap migrasi database hanya akan dijalankan satu kali. Untuk membuat perubahan tambahan pada sebuah tabel, Anda perlu membuat migrasi baru. Selama pengembangan, Anda mungkin ingin memperbarui migrasi yang belum diterapkan dan membangun ulang database dari awal menggunakan perintah **php artisan migrate:fresh**.

Testing Out

Kita sekarang akan mengetes chirp menggunakan form yang kita buat , kita tidak dapat melihat chirp sebelumnya dikarenakan tidak menampilkan chirps di page kita



Artisan Tinker

Waktunya untuk sedikit mempelajari artisan tinker. Sebuah REPL (Read Eval Print Loop) Dimana kita dapat mengeksekusi arbitrary kode php di laravel aplikasi, di cmd start tinker aplikasi baru

```
>
> App\Models\Chirp::all();
= Illuminate\Database\Eloquent\Collection {#6209
  all: [],
}

> App\Models\Chirp::all();
= Illuminate\Database\Eloquent\Collection {#6214
  all: [
    App\Models\Chirp {#6201
      id: 1,
      user_id: 1,
      message: "Halo Nama Saya MasGozali",
      created_at: "2024-12-26 16:08:48",
      updated_at: "2024-12-26 16:08:48",
    },
  ],
}
```

Halo nama Saya Masgozali telah di input menggunakan form chirps.

3. Showing Chirps

Retrieving Chirps

Perbaharui index metode di ChirpController untuk melanjutkan Chirps dari semua user ke index page

```
+         return view('chirps.index', [
+             'chirps' => Chirp::with('user')->latest()->get(),
+         ]);
```

Di sini, kami menggunakan metode **with** dari Eloquent untuk **eager-load** setiap **Chirp** yang terhubung dengan **user**-nya. Kami juga menggunakan **scope latest** untuk mengembalikan record dalam urutan terbalik secara kronologis.

Mengembalikan semua **Chirp** sekaligus tidak akan skala di produksi. Lihatlah **paginasi** yang kuat dari Laravel untuk meningkatkan performa.

Connecting User to Chirps

Untuk mengeluarkan data author chirps, user relationship belum didefinisikan, untuk memperbaiki ini tambahkan “BelongsTo” relationship ke chirps model

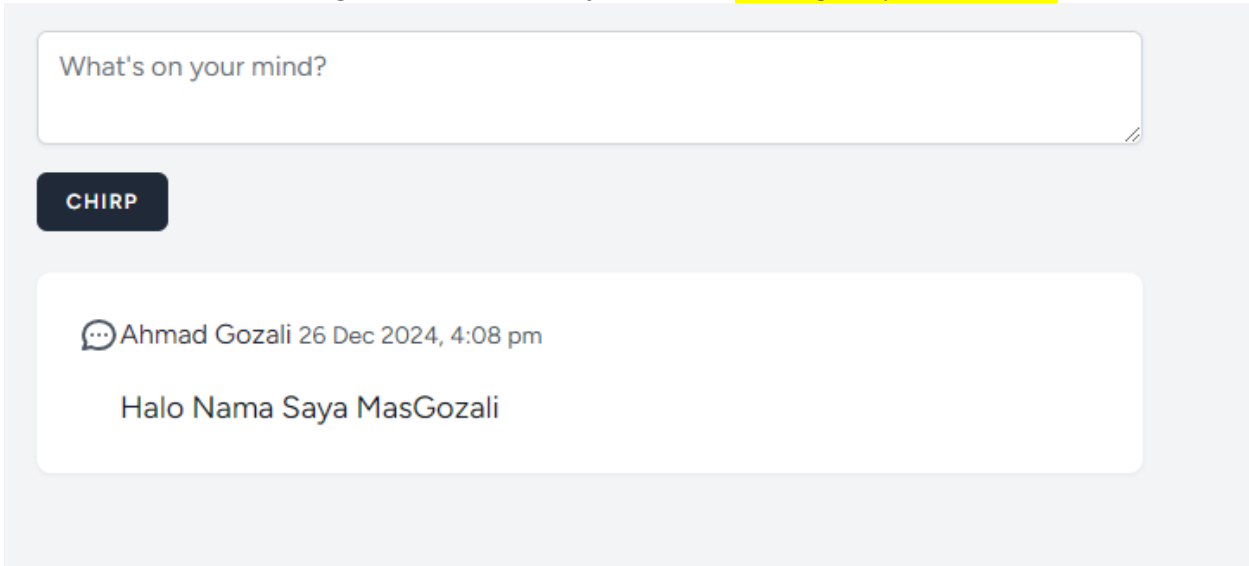
```
7 use Illuminate\Database\Eloquent\Relations\BelongsTo;
8
9 class Chirp extends Model
10 {
11     use HasFactory;
12     protected $fillable = [
13         'message',
14     ];
15
16     public function user(): BelongsTo
17     {
18         return $this->belongsTo(User::class);
19     }
20 }
```

Updating our View

Kemudian update tampilan di index.chirp component untuk mendisplay tambahkan syntax dibawah ini

```
+ <div class="mt-6 bg-white shadow-sm rounded-lg divide-y">
+     @foreach ($chirps as $chirp)
+         <div class="p-6 flex space-x-2">
+             <svg xmlns="http://www.w3.org/2000/svg" class="h-6 w-6">
+                 <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 14l2-2m0 0l2-2m-2 2l-2-2m0 0l-2-2M12 10l2 2m0 0l2 2m-2 2l-2 2m0 0l-2 2" />
+             </svg>
+             <div class="flex-1">
+                 <div class="flex justify-between items-center">
+                     <div>
+                         <span class="text-gray-800">{{ $chirp->user->name }}</span>
+                         <small class="ml-2 text-sm text-gray-600">@{{ $chirp->user->username }}</small>
+                     </div>
+                     <div>
+                         <p class="mt-4 text-lg text-gray-900">{{ $chirp->message }}</p>
+                     </div>
+                 </div>
+             </div>
+         </div>
+     @endforeach
+ </div>
```

Sekarang kembali ke chirps.tes/chirps periksa tampilan dengan refresh



Coba register dengan akun lain, agar bisa melihat perubahannya

4. Editing Chirps

Routing

Ayo menambahkan edit, kita tambahkan route file untuk memperbolehkan enable the chirps.edit dan chirps.update untuk kontrol resource. Chirps edit akan ditampilkan untuk dapat mengedit sedangkan chirps.update untuk menerima data dari form

Tambahkan syntax dibawah di route/web.php

```
->only(['index', 'store'])
+>only(['index', 'store', 'edit', 'update'])
```

Routing akan menjadi seperti ini

Verb	URI	Action	Route Name
GET	/chirps	index	chirps.index
POST	/chirps	store	chirps.store
GET	/chirps/{chirp}/edit	edit	chirps.edit
PUT/PATCH	/chirps/{chirp}	update	chirps.update

Linking to the edit page

Berikutnya, mari kita tautkan rute baru kita, yaitu chirps.edit. Kita akan menggunakan komponen x-dropdown yang sudah disediakan oleh Breeze, yang hanya akan ditampilkan kepada penulis Chirp. Kita juga akan menampilkan indikasi jika sebuah Chirp telah diedit dengan membandingkan tanggal created_at dari Chirp dengan tanggal updated_at-nya:

```
+         @unless ($chirp->created_at->eq($chirp->u
+         <small class="text-sm text-gray-600">
+         @endunless
+     </div>
+     @if ($chirp->user->is(auth()->user()))
+         <x-dropdown>
+             <x-slot name="trigger">
+                 <button>
+                     <svg xmlns="http://www.w3.org
+                     <path d="M6 10a2 2 0 11-4
+                     </svg>
+                 </button>
+             </x-slot>
+             <x-slot name="content">
+                 <x-dropdown-link :href="route('ch
+                     {{ __('Edit') }}
+                 </x-dropdown-link>
+             </x-slot>
+         </x-dropdown>
+     @endif
```

Creating the edit form

Mari kita buat tampilan Blade baru dengan sebuah formulir untuk mengedit sebuah Chirp. Formulir ini mirip dengan formulir untuk membuat Chirp, tetapi kita akan mengirim data ke `route chirps.update` dan menggunakan direktif `@method` untuk menentukan bahwa kita sedang melakukan permintaan "PATCH". Selain itu, kita juga akan mengisi terlebih dahulu kolom input dengan pesan Chirp yang sudah ada:

```
resources/views/chirps/edit.blade.php

<x-app-layout>
    <div class="max-w-2xl mx-auto p-4 sm:p-6 lg:p-8">
        <form method="POST" action="{{ route('chirps.update', $chirp) }}">
            @csrf
            @method('patch')
            <textarea
                name="message"
                class="block w-full border-gray-300 focus:border-indigo-300"
            >{{ old('message', $chirp->message) }}</textarea>
            <x-input-error :messages="$errors->get('message')" class="mt-2" />
            <div class="mt-4 space-x-2">
                <x-primary-button>{{ __('Save') }}</x-primary-button>
                <a href="{{ route('chirps.index') }}">{{ __('Cancel') }}</a>
            </div>
        </form>
    </div>
</x-app-layout>
```

Updating our Controller

Mari kita perbarui metode edit pada ChirpController untuk menampilkan formulir kita. Laravel secara otomatis akan memuat model Chirp dari database menggunakan route model binding, sehingga kita dapat langsung meneruskannya ke tampilan.

Selanjutnya, kita akan memperbarui metode update untuk memvalidasi permintaan dan memperbarui data di database.

Meskipun kita hanya menampilkan tombol edit kepada penulis Chirp, kita tetap perlu memastikan bahwa pengguna yang mengakses rute-rute ini memiliki otorisasi yang sesuai:

```
+ use Illuminate\Support\Facades\Gate;

+ public function edit(Chirp $chirp): View
+
+     Gate::authorize('update', $chirp);
+
+     return view('chirps.edit', [
+         'chirp' => $chirp,
+     ]);

+ public function update(Request $request, Chirp $chirp): RedirectResp
+
+     Gate::authorize('update', $chirp);
+
+     $validated = $request->validate([
+         'message' => 'required|string|max:255',
+     ]);
+
+     $chirp->update($validated);
+
+     return redirect(route('chirps.index'));
```

Anda mungkin menyadari bahwa aturan validasi tersebut terduplikasi dengan metode store. Anda bisa mempertimbangkan untuk mengekstraknya menggunakan Form Request Validation Laravel, yang mempermudah penggunaan kembali aturan validasi dan menjaga controller tetap ringan.

Authorization

Secara default, metode authorize akan mencegah semua orang untuk dapat memperbarui Chirp. Kita dapat menentukan siapa yang diizinkan untuk memperbaruinya dengan membuat **Model Policy** menggunakan perintah berikut:

`php artisan make:policy ChirpPolicy --model=Chirp`

```
C:\laragon\www\chirper  php artisan make:policy ChirpPolicy --model=Chirp
λ php artisan make:policy ChirpPolicy --model=Chirp

INFO Policy [C:\laragon\www\chirper\app\Policies\ChirpPolicy.php] created successfully.

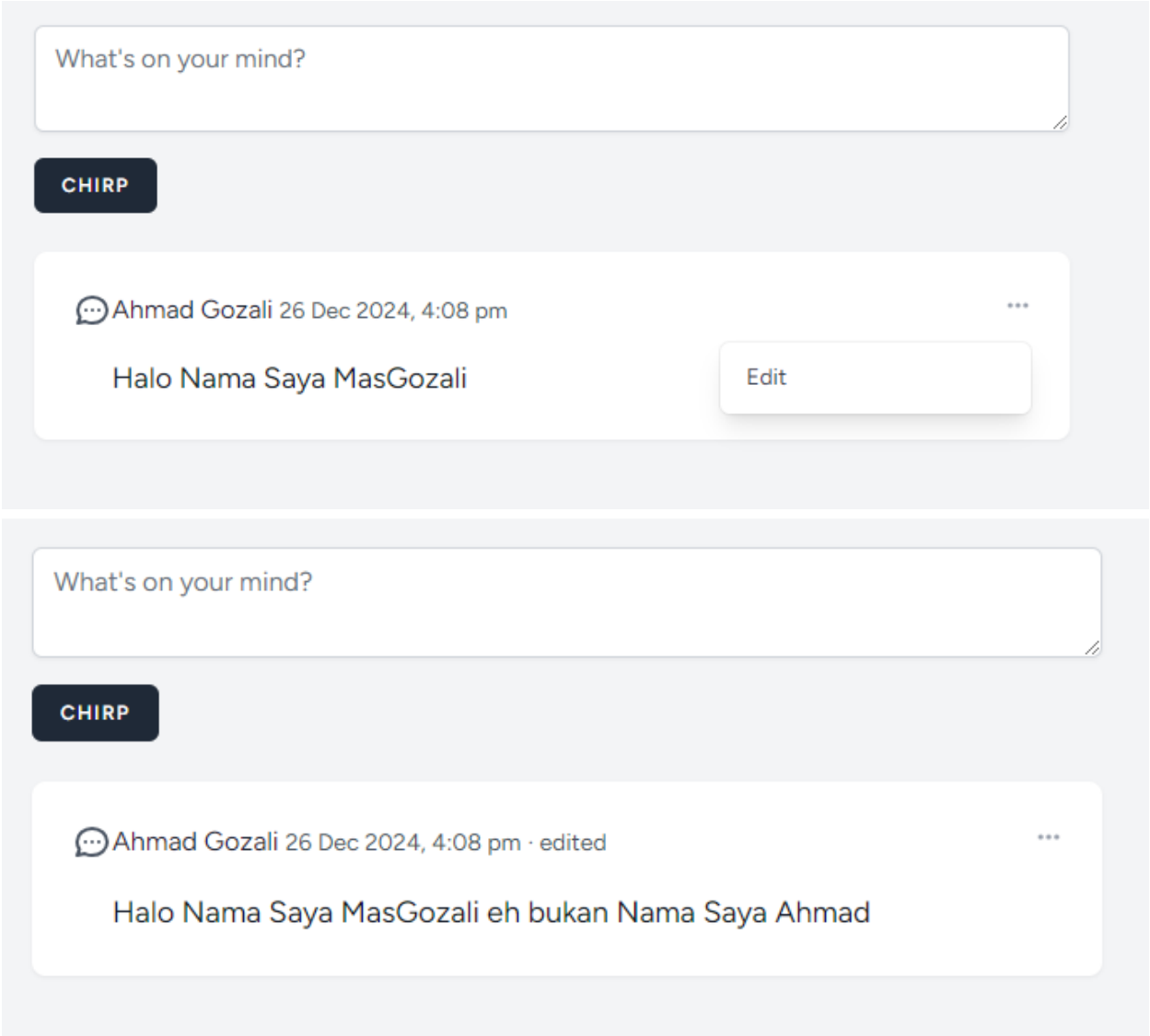
C:\laragon\www\chirper
λ |
```

Perintah ini akan membuat kelas kebijakan (policy class) di `app/Policies/ChirpPolicy.php`, yang dapat kita perbarui untuk menentukan bahwa hanya penulis yang diizinkan untuk memperbarui sebuah Chirp:

```
+     return $chirp->user()->is($user);
```


Testing Out

Saatnya mencobanya! Silakan edit beberapa Chirp menggunakan menu dropdown. Jika Anda mendaftarkan akun pengguna lain, Anda akan melihat bahwa hanya penulis sebuah Chirp yang dapat mengeditnya.



5. Deleting Chirps

Routing

Kita akan mulai lagi dengan memperbarui rute kita untuk mengaktifkan chirps.destroy rute:

```
Route::resource('chirps', ChirpController::class)
-   ->only(['index', 'store', 'edit', 'update'])
+   ->only(['index', 'store', 'edit', 'update', 'destroy'])
```

Tabel rute kami untuk pengontrol ini sekarang terlihat seperti ini:

Verb	URI	Action	Route Name
GET	/chirps	index	chirps.index
POST	/chirps	store	chirps.store
GET	/chirps/{chirp}/edit	edit	chirps.edit
PUT/PATCH	/chirps/{chirp}	update	chirps.update
DELETE	/chirps/{chirp}	destroy	chirps.destroy

Updating our controller

Sekarang kita dapat memperbarui destroy metode di ChirpController kelas kita untuk melakukan penghapusan dan kembali ke indeks Chirp:

Untuk tanda – itu dihapus di kodingan kita dan tambahkan yang +

```
- public function destroy(Chirp $chirp)
+ public function destroy(Chirp $chirp): RedirectResponse
{
- //
+ Gate::authorize('delete', $chirp);
+
+ $chirp->delete();
+
+ return redirect(route('chirps.index'));
}
```

Authorization

Seperti halnya pengeditan, kami hanya ingin penulis Chirp dapat menghapus Chirp mereka, jadi mari perbarui delete metode di ChirpPolicy kelas kami:

```
+ return $this->update($user, $chirp);
```

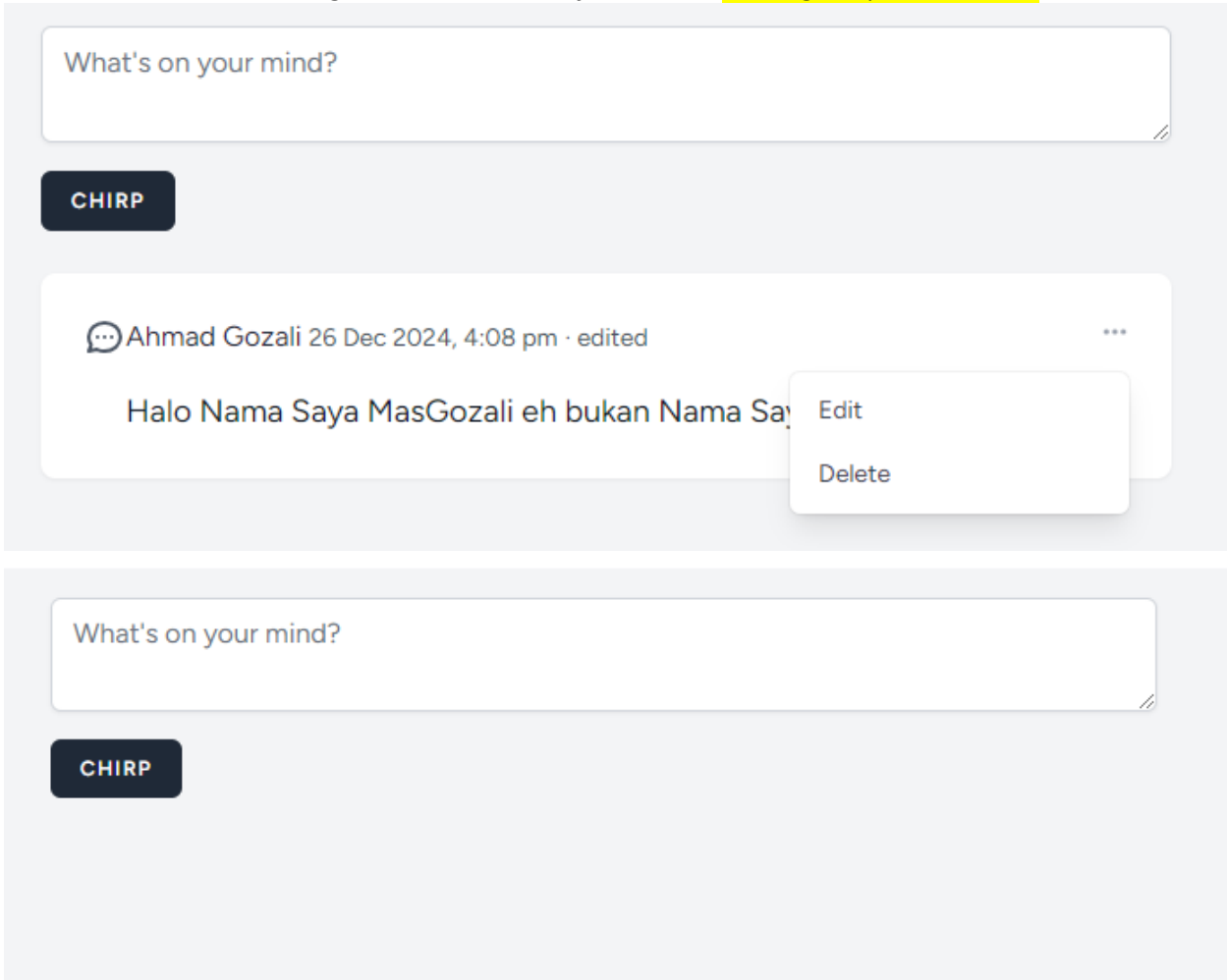
Updating our view

Terakhir, kita dapat menambahkan tombol hapus ke menu dropdown yang kita buat sebelumnya di chirps.index tampilan kita:

```
+ <form method="POST" action="{{ route('chirps.destroy', $chirp) }}">
+     @csrf
+     @method('delete')
+     <x-dropdown-link :href="route('chirps.destroy', $chirp)">
+         {{ __('Delete') }}
+     </x-dropdown-link>
+ </form>
```

Testing Out

Jika Anda menge-Chirp sesuatu yang tidak Anda sukai, cobalah menghapusnya!



6. Notifications & Events

Mari bawa Chirper ke tingkat berikutnya dengan mengirimkan pemberitahuan email saat Chirp baru dibuat.

Selain dukungan untuk mengirim email, Laravel menyediakan dukungan untuk mengirim notifikasi melalui berbagai saluran pengiriman, termasuk email, SMS, dan Slack. Ditambah lagi, berbagai saluran notifikasi buatan komunitas telah dibuat untuk mengirim notifikasi melalui lusinan saluran yang berbeda! Notifikasi juga dapat disimpan dalam basis data sehingga dapat ditampilkan di antarmuka web Anda.

Creating the Notification

Artisan dapat melakukannya, sekali lagi, melakukan semua kerja keras untuk kita dengan perintah berikut:

```
C:\laragon\www\chirper
λ php artisan make:notification NewChirp

[INFO] Notification [C:\laragon\www\chirper\app\Notifications\NewChirp.php] created successfully.
```

Ini akan membuat pemberitahuan baru `app/Notifications/NewChirp.php` yang siap untuk kita sesuaikan.

Mari kita buka `NewChirp` class dan izinkan untuk menerima Chirp yang baru saja dibuat, lalu sesuaikan pesan untuk menyertakan nama penulis dan cuplikan pesan:

```
+ use App\Models\Chirp;
use Illuminate\Bus\Queueable;
+ use Illuminate\Support\Str;
```

```
+         ->subject("New Chirp from {$this->chirp->user->name}");
+         ->greeting("New Chirp from {$this->chirp->user->name}");
+         ->line(Str::limit($this->chirp->message, 50));
+         ->action('Go to Chirper', url('/'))
+         ->line('Thank you for using our application!');
```

Kita dapat mengirim notifikasi langsung dari storemetode di ChirpController class kita, namun hal itu akan menambah pekerjaan pengontrol, yang pada gilirannya dapat memperlambat permintaan, terutama karena kita akan melakukan kueri pada basis data dan mengirim email.

Sebagai gantinya, mari kita kirimkan suatu kejadian yang dapat kita dengarkan dan proses dalam antrean latar belakang agar aplikasi kita tetap berjalan cepat.

Creating an event

Peristiwa merupakan cara hebat untuk memisahkan berbagai aspek aplikasi Anda, karena satu peristiwa dapat memiliki beberapa pendengar yang tidak bergantung satu sama lain.

Mari buat acara baru kita dengan perintah berikut:

```
C:\laragon\www\chirper
λ php artisan make:event ChirpCreated

INFO Event [C:\laragon\www\chirper\app\Events\ChirpCreated.php] created successfully.
```

Ini akan membuat kelas acara baru di app/Events/ChirpCreated.php.

Karena kami akan mengirimkan peristiwa untuk setiap Chirp baru yang dibuat, mari perbarui ChirpCreated peristiwa untuk menerima yang baru dibuat Chirp sehingga kami dapat meneruskannya ke notifikasi kami:

```
+ use App\Models\Chirp;

+ public function __construct(public Chirp $chirp)
```

Dispatching the event

Sekarang setelah kita memiliki kelas event, kita siap untuk mengirimkannya kapan saja Chirp dibuat. Anda dapat mengirimkan event di mana saja dalam siklus hidup aplikasi Anda, tetapi karena event kita terkait dengan pembuatan model Eloquent, kita dapat mengonfigurasi Chirpmodel kita untuk mengirimkan event tersebut bagi kita.

```
+ use App\Events\ChirpCreated;

+ protected $dispatchesEvents = [
+     'created' => ChirpCreated::class,
+ ];
```

Sekarang kapan pun sesuatu yang baru Chirp dibuat, ChirpCreated akan dikirim.

Creating an event listener

Sekarang setelah kami mengirimkan suatu kejadian, kami siap mendengarkan kejadian tersebut dan mengirimkan pemberitahuan kami.

Mari membuat pendengar yang berlangganan ChirpCreated acara kita:

```
C:\laragon\www\chirper
λ php artisan make:listener SendChirpCreatedNotifications --event=ChirpCreated

INFO Listener [C:\laragon\www\chirper\app\Listeners\SendChirpCreatedNotifications.php] created successfully.
```

Pendengar baru akan ditempatkan di app/Listeners/SendChirpCreatedNotifications.php.

Mari perbarui pendengar untuk mengirimkan notifikasi kita.

```
+ use App\Models\User;
+ use App\Notifications\NewChirp;

+ class SendChirpCreatedNotifications implements ShouldQueue
+ {
+     //
+     foreach (User::whereNot('id', $event->chirp->user_id)->cursor()
+         $user->notify(new NewChirp($event->chirp));
+ }
```

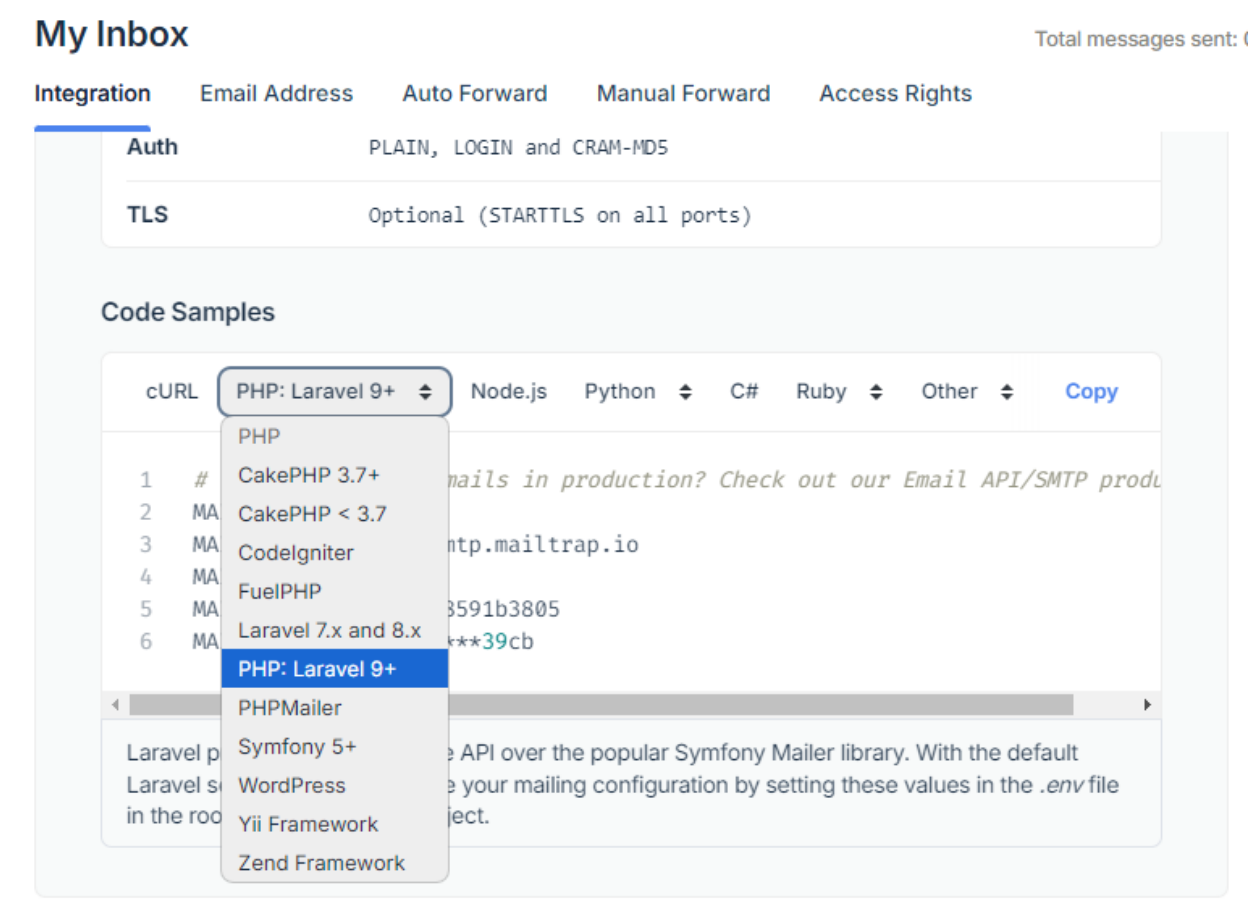
Kami telah menandai listener kami dengan ShouldQueueinterface, yang memberi tahu Laravel bahwa listener harus dijalankan dalam antrian . Secara default, antrian "database" akan digunakan untuk memproses pekerjaan secara asinkron. Untuk mulai memproses pekerjaan yang diantrekan, Anda harus menjalankan php artisan queue:workperintah Artisan di terminal Anda.

Kami juga telah mengonfigurasi listener kami untuk mengirim notifikasi ke setiap pengguna di platform, kecuali penulis Chirp. Kenyataannya, hal ini mungkin mengganggu pengguna, jadi Anda mungkin ingin menerapkan fitur "following" sehingga pengguna hanya menerima notifikasi untuk akun yang mereka ikuti.

Kami menggunakan kursor basis data untuk menghindari memuat setiap pengguna ke dalam memori sekaligus.

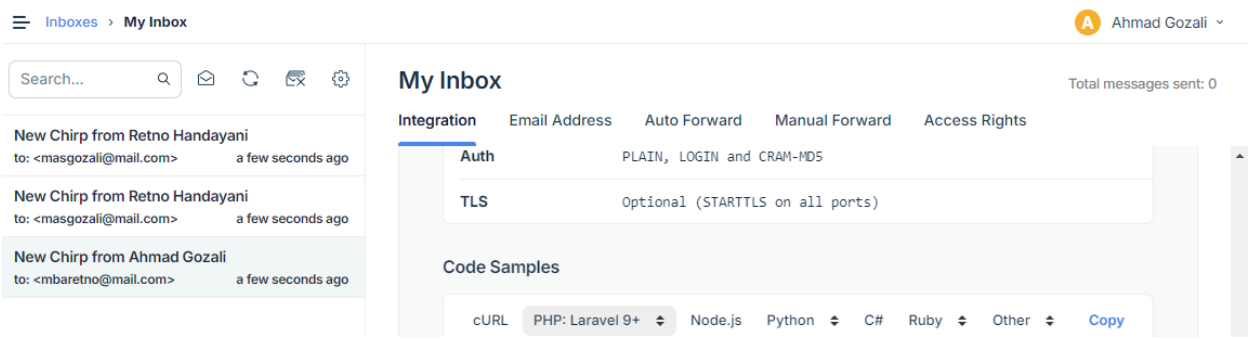
Testing Out

Setting smtp menggunakan mailtrap.io dengan menggunakan code pilih laravel 9+



Salin atau ubah settingan smtp sesuaikan dengan setting mailtrap

```
51 # Looking to send emails in production? Check out our Email API/SMTP product!
52 MAIL_MAILER=smtp
53 MAIL_HOST=sandbox.smtp.mailtrap.io
54 MAIL_PORT=2525
55 MAIL_USERNAME=1dd5e8591b3805
56 MAIL_PASSWORD=a2238055c739cb
57 MAIL_ENCRYPTION=null
58 MAIL_FROM_ADDRESS="hello@example.com"
59 MAIL_FROM_NAME="${APP_NAME}"
60
```



Sending Email

New Chirp from Retno Handayani



From: Laravel <hello@example.com>
To: <masgozali@mail.com>

2024-12-26 18:18, 12 KB

Show Headers

HTML HTML Source Text Raw Spam Analysis HTML Check 18 Tech Info



New Chirp from Retno Handayani

Apa Kabar

Go to Chirper

Thank you for using our application!

Regards,
Laravel

```
PS C:\laragon\www\chirper> php artisan queue:work
```

```
INFO Processing jobs from the [default] queue.
```

```
2024-12-26 18:18:25 App\Listeners\SendChirpCreatedNotifications ..... RUNNING
2024-12-26 18:18:39 App\Listeners\SendChirpCreatedNotifications ..... 13s DONE
2024-12-26 18:18:39 App\Listeners\SendChirpCreatedNotifications ..... RUNNING
2024-12-26 18:18:40 App\Listeners\SendChirpCreatedNotifications ..... 1s DONE
2024-12-26 18:18:55 App\Listeners\SendChirpCreatedNotifications ..... RUNNING
2024-12-26 18:18:57 App\Listeners\SendChirpCreatedNotifications ..... 1s DONE
```

```
MINGW64:/c:/laragon/www/tugaschirper
```

```
egov02@egov02 MINGW64 /c:/laragon/www/tugaschirper (main)
$ git add .
warning: in the working copy of 'package.json', CRLF will be replaced by LF the
next time Git touches it
warning: in the working copy of 'resources/views/chirps/edit.blade.php', CRLF wi
ll be replaced by LF the next time Git touches it
warning: in the working copy of 'resources/views/chirps/index.blade.php', CRLF w
ill be replaced by LF the next time Git touches it
```

```
egov02@egov02 MINGW64 /c:/laragon/www/tugaschirper (main)
$ git commit -m "First Commit"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'egov02@egov02.(none)')
```

```
egov02@egov02 MINGW64 /c:/laragon/www/tugaschirper (main)
$ git config --global user.email "masgozali.bna@gmail.com"

egov02@egov02 MINGW64 /c:/laragon/www/tugaschirper (main)
$ git config --global user.name "masgozali"

egov02@egov02 MINGW64 /c:/laragon/www/tugaschirper (main)
$ git commit -m "First Commit"
```

```
egov02@egov02 MINGW64 /c:/laragon/www/tugaschirper (main)
$ git commit -m "First Commit"
[main (root-commit) 7e9d2be] First Commit
119 files changed, 15708 insertions(+)
create mode 100644 .editorconfig
create mode 100644 .env.example
create mode 100644 .gitattributes
create mode 100644 .gitignore
create mode 100644 README.md
create mode 100644 app/Events/ChirpCreated.php
create mode 100644 app/Http/Controllers/Auth/AuthenticatedSessionController.php
create mode 100644 app/Http/Controllers/Auth/ConfirmablePasswordController.php
create mode 100644 app/Http/Controllers/Auth/EmailVerificationNotificationContr
oller.php
create mode 100644 app/Http/Controllers/Auth/EmailVerificationPromptController.
php
create mode 100644 app/Http/Controllers/Auth/NewPasswordController.php
create mode 100644 app/Http/Controllers/Auth/PasswordController.php
create mode 100644 app/Http/Controllers/Auth/PasswordResetLinkController.php
create mode 100644 app/Http/Controllers/Auth/RegisteredUserController.php
create mode 100644 app/Http/Controllers/Auth/VerifyEmailController.php
create mode 100644 app/Http/Controllers/ChirpController copy.php
create mode 100644 app/Http/Controllers/ChirpController.php
create mode 100644 app/Http/Controllers/Controller.php
create mode 100644 app/Http/Controllers/ProfileController.php
create mode 100644 app/Http/Requests/Auth/LoginRequest.php
create mode 100644 app/Http/Requests/ProfileUpdateRequest.php
create mode 100644 app/Listeners/SendChirpCreatedNotifications.php
create mode 100644 app/Models/Chirp.php
create mode 100644 app/Models/User.php
create mode 100644 app/Notifications/NewChirp.php
create mode 100644 app/Policies/ChirpPolicy.php
create mode 100644 app/Providers/AppServiceProvider.php
create mode 100644 app/View/Components/AppLayout.php
create mode 100644 app/View/Components/GuestLayout.php
create mode 100644 artisan
create mode 100644 bootstrap/app.php
create mode 100644 bootstrap/cache/.gitignore
create mode 100644 bootstrap/providers.php
create mode 100644 composer.json
create mode 100644 composer.lock
create mode 100644 config/app.php
create mode 100644 config/auth.php
create mode 100644 config/cache.php
create mode 100644 config/database.php
create mode 100644 config/filesystems.php
```

```
egov02@egov02 MINGW64 /c:/laragon/www/tugaschirper (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 155, done.
Counting objects: 100% (155/155), done.
Delta compression using up to 8 threads
Compressing objects: 100% (139/139), done.
Writing objects: 100% (155/155), 115.65 KiB | 1.68 MiB/s, done.
Total 155 (delta 21), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (21/21), done.
To https://github.com/masgozali/tugaschirper.git
 * [new branch]      main -> main
```

Document/Source Code dapat di akses di

<https://github.com/masgozali/tugaschirper.git>