Creare un sistema di logging delle visite per un sito web con Laravel

Creare un sistema di logging delle visite per un sito web con Laravel è un modo efficace per raccogliere dati sugli utenti, monitorare il traffico e migliorare l'esperienza utente. Con Laravel, l'implementazione diventa più semplice grazie alla sua architettura MVC e alle potenti funzionalità integrate. Vediamo come costruire un sistema di logging delle visite passo per passo.

Inizia installando una nuova applicazione Laravel, se non ne hai già una. Esegui il seguente comando:

```
composer create-project laravel/laravel visit-logger
```

Laravel semplifica la gestione dei database grazie alle migrazioni e ai modelli Eloquent. Crea una migrazione e un modello per le visite con il comando:

```
php artisan make:model Visit -m
```

Questo creerà un modello chiamato `Visit` e un file di migrazione per la tabella delle visite in `database/migrations`.

Modifica il file di migrazione generato in `database/migrations/<timestamp>_create_visits_table.php` per definire la struttura della tabella `visits`:

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateVisitsTable extends Migration
{
    public function up()
    {
        Schema::create('visits', function (Blueprint $table) {
            $table->id();
            $table->string('ip_address', 45);
```

```
            $table->text('user_agent');
            $table->string('page_url');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('visits');
    }
}
```

Descrizione dei campi:

- `ip_address`: l'indirizzo IP del visitatore.
- `user_agent`: le informazioni sul browser e sul sistema operativo.
- `page_url`: l'URL della pagina visitata.
- `timestamps`: colonne per `created_at` e `updated_at`, create automaticamente da Laravel.

Esegui la migrazione per creare la tabella nel database:

```
php artisan migrate
```

Modifica il modello `Visit` in `app/Models/Visit.php` per impostare i campi che possono essere riempiti:

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Visit extends Model
{
    use HasFactory;

    protected $fillable = [
        'ip_address',
        'user_agent',
        'page_url',
    ];
}
```

Ora possiamo creare un middleware per registrare ogni visita. Utilizza il comando:

```
php artisan make:middleware LogVisit
```

Questo creerà un middleware in `app/Http/Middleware/LogVisit.php`. Modifica il file per registrare i dati della visita:

```
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use App\Models\Visit;

class LogVisit
{
    public function handle(Request $request, Closure $next)
    {
        Visit::create([
            'ip_address' => $request->ip(),
            'user_agent' => $request->header('User-Agent'),
            'page_url' => $request->fullUrl(),
        ]);

        return $next($request);
    }
}
```

Spiegazione:

- `$request->ip()` ottiene l'indirizzo IP del visitatore.
- `$request->header('User-Agent')` recupera le informazioni sul browser e sul sistema operativo.
- `$request->fullUrl()` ottiene l'URL completo della pagina visitata.
- `Visit::create([...])` salva i dati della visita nella tabella `visits`.

Per assicurarti che ogni visita venga registrata, aggiungi il middleware `LogVisit` all’elenco dei middleware globali. Apri `app/Http/Kernel.php` e aggiungilo alla proprietà `$middleware`:

```
protected $middleware = [
    // Altri middleware...
```

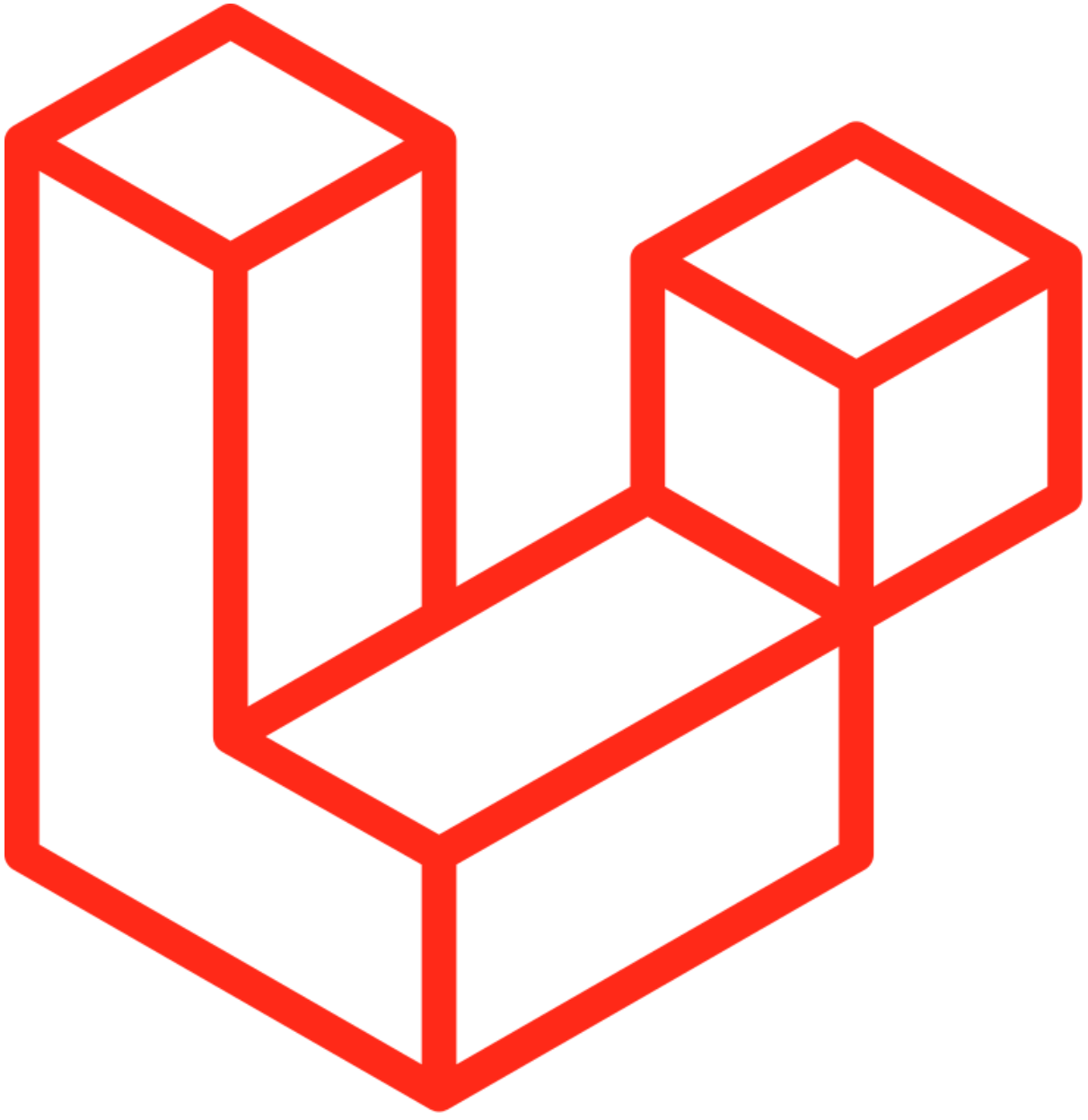
```
    \App\Http\Middleware\LogVisit::class,
];
```

Ora ogni richiesta al tuo sito passerà attraverso il middleware `LogVisit`, che registrerà i dati della visita.

Conclusione

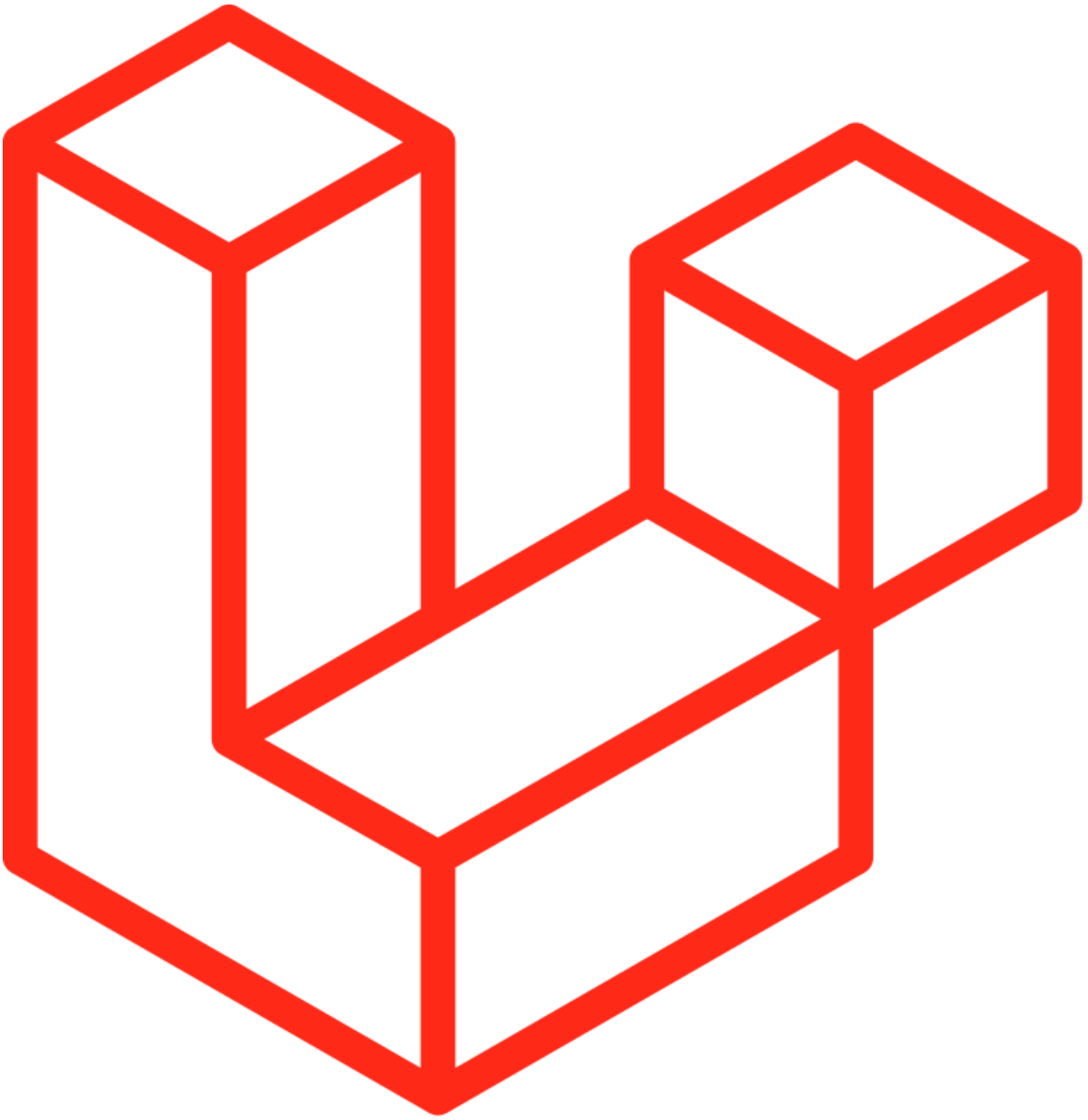
Implementare un sistema di logging delle visite con Laravel è una soluzione semplice ma potente per monitorare il traffico e raccogliere dati sugli utenti.

Applicazioni Correlate



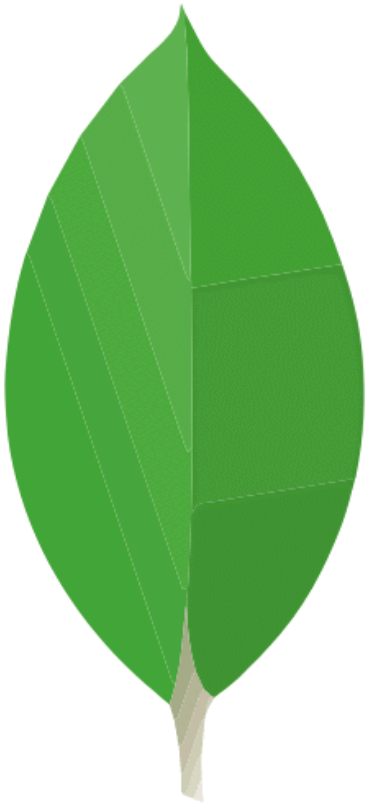
Laravel Placeholder Image

Applicazione in Laravel per creare immagini segnaposto.
DockerDocker ComposeComposerPHPLaravelJavaScript



Laravel Shopping Cart

Applicazione che fa parte del progetto Laravel E-commerce e implementa la gestione del carrello in Laravel.
DockerDocker ComposeComposerPHPLaravel



PHP MongoDB App

Applicazione basata su MongoDB con il driver PHP ufficiale.
DockerDocker ComposeComposerPHPMongoDB