Implementing Authentication

Mateo Prigl
Software developer

Implementing Authentication

Registration

Form with three fields: name, email and password

Login

Form for user credentials: email and password

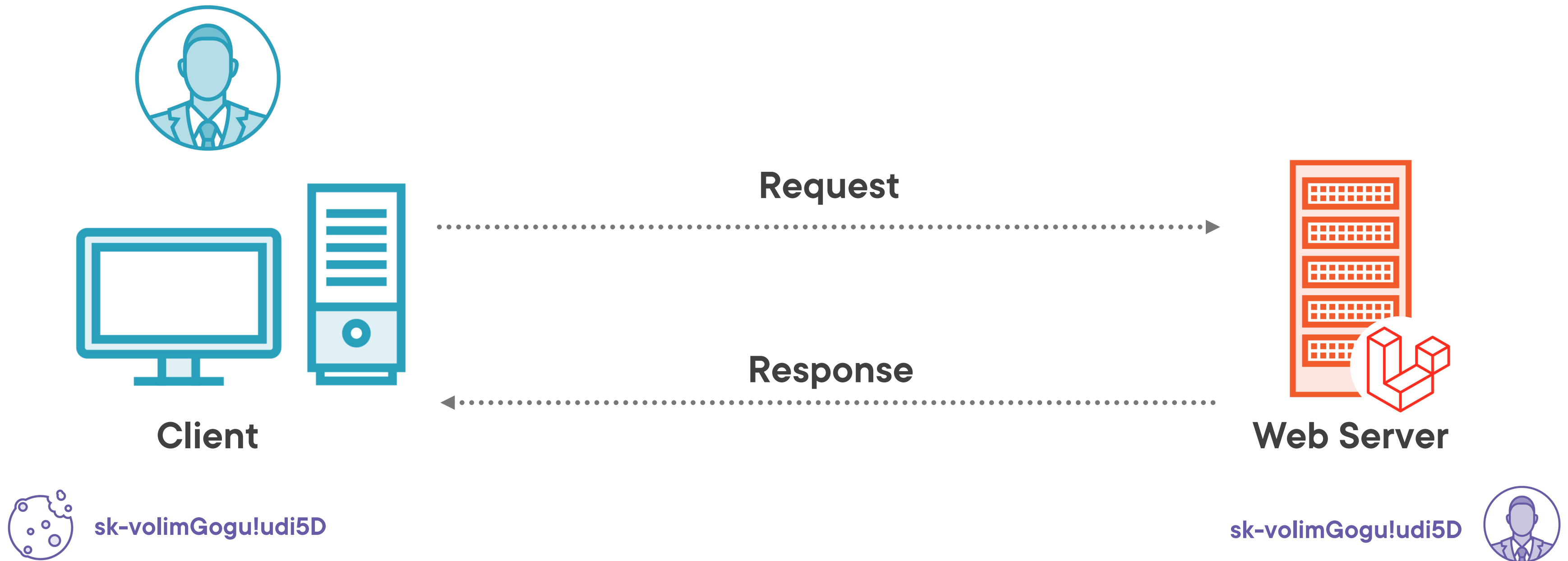**Logout**
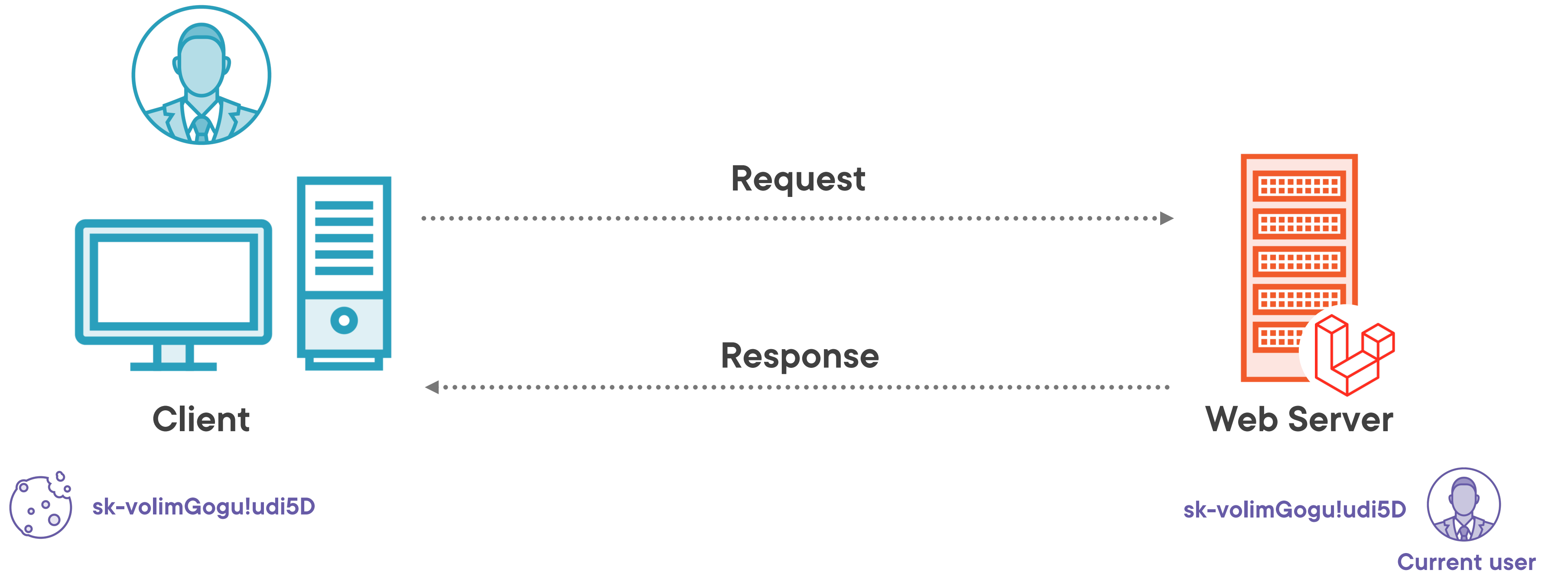
Route for removing the user session

HTTP Protocol

Request

Response

Client

Web Server

HTTP Protocol

Request

Response

Client

Web Server

HTTP Protocol

Request

Response

Client

Web Server

sk-volimGogu!udi5D

HTTP Protocol

Request

Response

Client

Web Server

sk-volimGogu!udi5D

sk-volimGogu!udi5D

HTTP Protocol

Client

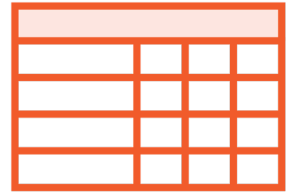Request

Response
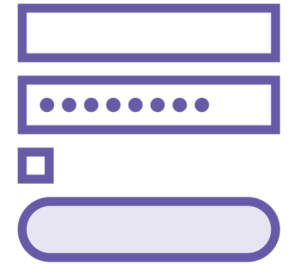
Web Server

sk-volimGogu!udi5D

sk-volimGogu!udi5D

Current user

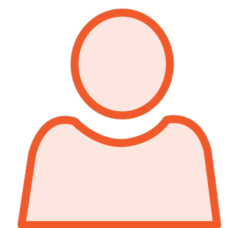# Implementing Authentication

Create the **"users"** table

Register users with the **User** model

Login users with **Auth::attempt()**

Logout users with **Auth::logout()**

Get the current user with **Auth::user()**

Summary

Configuring the Authentication System

```php
'guards' => [
        'web' => [
                'driver' => 'session',
                'provider' => 'users',
        ],
    ],
...
'providers' => [
        'users' => [
                'driver' => 'eloquent',
                'model' => App\Models\User::class,
        ],
```

config/auth.php

HTTP Protocol

Client

Request

Response

Web Server

sk-volimGogu!udi5D

sk-volimGogu!udi5D

Current user

Registering New Users

```php
public function register(Request $request)
{
        $request->validate([
            'name' => 'required',
            'email' => 'required|email|unique:users',
            'password' => 'required|min:6',
        ]);

        User::create([
            'name' => $request->input('name'),
            'email' => $request->input('email'),
            'password' => Hash::make($request->input('password')),
        ]);
        ...
}
```

Logging Users In

```php
public function login(Request $request)
{

        $credentials = $request->validate([
            'email' => 'required',
            'password' => 'required',
        ]);


        if (Auth::attempt($credentials)) {
            return redirect()
                ->route('home')
                ->with('success', 'You are logged in!');
        }
        return redirect()
            ->route('login')
            ->withErrors('Provided login information is not valid.');

}
```

Logging Users Out

```php
public function logout(Request $request)
{
        Auth::logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();


    return redirect()
        ->route('home')
        ->with('success', 'You are logged out.');

}
```

Authentication Blade Directives

```
@auth
  <li><a>Link for authenticated users</a></li>
  <p>Logged in as <b>{{ Auth::user()->name }}</b></p>
@else
  <li><a>Link for guest visitors</a></li>
@endauth
```

Authentication Middleware

```php
Route::resource('posts', PostController::class)
->middleware('auth');

Route::match(['get', 'post'], '/login',
[AuthController::class, 'login'])->name('login')
->middleware('guest');
```

Matching Multiple HTTP Verbs to One Route

Use the *match* method to allow access to the route with more than one HTTP verb/method

```php
Route::match(['get', 'post'], '/login', [AuthController::class, 'login']);
```

routes/web.php

Check which HTTP verb/method was used with the requests *isMethod* method

```php
if ($request->isMethod('get')) {
    // return something for the GET request
}
```

app/Http/Controllers/AuthController.php   (inside of the *"login"* action)

Up Next:
Restricting Access with User Authorization