Audência Quive

Guião

Licenciatura em Informática

Universidade Save

Chongoene

2024

Autenticação e autorização	**Criptografia e proteção de dados**
Transações em Aplicações Laravel	**Aplicação de API**
Testes unitários com PhpUnit	**Logs de auditoria**
Single Sign On (SSO)	

1. Autententicao e Autorizacao

1.1. Autenticação

Na autenticação usei os pacotes padroes do laravel pra fazer autenticacao no login

Controlador dos usuários

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Http\Requests\Auth\LoginRequest;
use App\Mail\TwoFactorCodeMail;
use App\Providers\RouteServiceProvider;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Mail;
use Illuminate\View\View;

class AuthenticatedSessionController extends Controller
{
    /**
     * Display the login view.
     */
    public function create(): View
    {
        return view('auth.login');
    }

    /**
     * Handle an incoming authentication request.
     */
    public function store(LoginRequest $request): RedirectResponse
    {
        $request->authenticate();

        // Verifica se há um ID armazenado para 2FA
        if (session('auth.id')) {
            return redirect()->route('auth.two-factor');
        }

        $request->session()->regenerate();
```

```
        return redirect()->intended(RouteServiceProvider::HOME);
    }


    /**
     * Destroy an authenticated session.
     */
    public function destroy(Request $request): RedirectResponse
    {
        Auth::guard('web')->logout();

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

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

        return redirect('/');
    }
}
```

1.2. Autorização

Arquivo de middleware: CheckAccess.php

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Services\LogService; // Importação do LogService

class CheckAccess
{
    public function handle(Request $request, Closure $next, ...$roles)
    {
        // Verificar se o usuário está autenticado
        if (!Auth::check()) {
            // Registrar tentativa de acesso não autenticado
            LogService::record([
                'user_id' => null, // Usuário não autenticado
                'user_name' => 'Visitante',
                'model' => 'Middleware',
                'action' => 'unauthorized_access',
                'status' => 'error',
                'message' => 'Tentativa de acesso não autenticada.',
                'ip_address' => $request->ip(),
            ]);
```

```
            return redirect()->route('login')->with('error', 'Usuarios
não autenticado');
        }

        // Obter o nível de acesso do usuário autenticado
        $userRole = Auth::user()->nivel_acesso;

        if (is_null($userRole) || !in_array($userRole, $roles)) {
            // Registrar tentativa de acesso negado
            LogService::record([
                'user_id' => Auth::id(),
                'user_name' => Auth::user()->name ?? 'Usuário
Desconhecido',
                'model' => 'Middleware',
                'action' => 'access_denied',
                'status' => 'error',
                'message' => "Acesso negado. Papel do usuário: " .
($userRole ?? 'não definido') . ".",
                'ip_address' => $request->ip(),
            ]);

            return redirect()->route('dashboard')->with('error', 'Acesso
não autorizado!');
        }


        // Passar para o próximo middleware ou controlador
        return $next($request);
    }
}
```

Configuração no kernel.php

```
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{
    /**
     * The application's global HTTP middleware stack.
     *
     * These middleware are run during every request to your application.
     *
     * @var array<int, class-string|string>
     */
```

```
    protected $middleware = [
        // \App\Http\Middleware\TrustHosts::class,
        \App\Http\Middleware\TrustProxies::class,
        \Illuminate\Http\Middleware\HandleCors::class,
        \App\Http\Middleware\PreventRequestsDuringMaintenance::class,
        \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
        \App\Http\Middleware\TrimStrings::class,
        \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull:
:class,
    ];

    /**
     * The application's route middleware groups.
     *
     * @var array<string, array<int, class-string|string>>
     */
    protected $middlewareGroups = [
        'web' => [
            \App\Http\Middleware\EncryptCookies::class,
            \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::cla
ss,
            \Illuminate\Session\Middleware\StartSession::class,
            \Illuminate\View\Middleware\ShareErrorsFromSession::class,
            \App\Http\Middleware\VerifyCsrfToken::class,
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],

        'api' => [
            //
\Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class
,
            \Illuminate\Routing\Middleware\ThrottleRequests::class.':api'
,
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],
    ];

    /**
     * The application's middleware aliases.
     *
     * Aliases may be used to conveniently assign middleware to routes
and groups.
     *
     * @var array<string, class-string|string>
     */
    protected $middlewareAliases = [
        'auth' => \App\Http\Middleware\Authenticate::class,
        'auth.basic' =>
\Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        'auth.session' =>
\Illuminate\Session\Middleware\AuthenticateSession::class,
```

```
        'cache.headers' =>
\Illuminate\Http\Middleware\SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
        'password.confirm' =>
\Illuminate\Auth\Middleware\RequirePassword::class,
        'signed' => \App\Http\Middleware\ValidateSignature::class,
        'throttle' =>
\Illuminate\Routing\Middleware\ThrottleRequests::class,
        'verified' =>
\Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
        'check.access' => \App\Http\Middleware\CheckAccess::class,
    ];
}
```

Aplicação nas rotas

Por exemplo na rotas do CRUD do usuários apenas o usuário admin pode manipular as rotas.

```
<?php

use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\ConfirmablePasswordController;
use App\Http\Controllers\Auth\EmailVerificationNotificationController;
use App\Http\Controllers\Auth\EmailVerificationPromptController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth\PasswordController;
use App\Http\Controllers\Auth\PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use Illuminate\Support\Facades\Route;
use Laravel\Socialite\Facades\Socialite;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Http;

Route::middleware('guest')->group(function () {

    Route::get('login', [AuthenticatedSessionController::class,
'create'])->name('login');

    Route::post('login', [AuthenticatedSessionController::class,
'store']);

    Route::get('forgot-password', [PasswordResetLinkController::class,
'create'])->name('password.request');
```

```
    Route::post('forgot-password', [PasswordResetLinkController::class,
'store'])->name('password.email');

    Route::get('reset-password/{token}', [NewPasswordController::class,
'create'])->name('password.reset');

    Route::post('reset-password', [NewPasswordController::class,
'store'])->name('password.store');

    Route::get('/auth/google', function () {
        $query = http_build_query([
            'client_id' => env('GOOGLE_CLIENT_ID'),
            'redirect_uri' => env('GOOGLE_REDIRECT_URI'),
            'response_type' => 'code',
            'scope' => 'email profile',
            'access_type' => 'offline',
            'prompt' => 'consent',
        ]);

        return redirect('https://accounts.google.com/o/oauth2/v2/auth?' .
$query);
    })->name('google.login');

    Route::get('/auth/google/callback', function
(\Illuminate\Http\Request $request) {
        $code = $request->input('code');

        if (!$code) {
            return redirect('/login')->with('error', 'Não foi possível
autenticar com o Google.');
        }

        // Trocar o código por um token de acesso
        $response = Http::asForm()-
>post('https://oauth2.googleapis.com/token', [
            'client_id' => env('GOOGLE_CLIENT_ID'),
            'client_secret' => env('GOOGLE_CLIENT_SECRET'),
            'redirect_uri' => env('GOOGLE_REDIRECT_URI'),
            'grant_type' => 'authorization_code',
            'code' => $code,
        ]);

        $data = $response->json();

        if (!isset($data['access_token'])) {
            return redirect('/login')->with('error', 'Erro ao obter o
token de acesso.');
        }

        // Obter informações do utilizador com o token de acesso
        $userResponse = Http::withHeaders([
```

```
            'Authorization' => 'Bearer ' . $data['access_token'],
        ])->get('https://www.googleapis.com/oauth2/v2/userinfo');

        $googleUser = $userResponse->json();

        // Processar o utilizador no sistema
        $user = \App\Models\User::where('email', $googleUser['email'])-
>first();

        if (!$user) {
            $user = \App\Models\User::create([
                'name' => $googleUser['name'],
                'email' => $googleUser['email'],
                'google_id' => $googleUser['id'],
                'password' => bcrypt('senha_aleatoria'), // Opcional
            ]);
        }

        // Fazer login do utilizador
        \Illuminate\Support\Facades\Auth::login($user);

        return redirect('/dashboard');
    });

});

#, 'check.device'
Route::middleware('auth')->group(function () {
    Route::get('verify-email', EmailVerificationPromptController::class)
                ->name('verification.notice');

    Route::get('verify-email/{id}/{hash}', VerifyEmailController::class)
                ->middleware(['signed', 'throttle:6,1'])
                ->name('verification.verify');

    Route::post('email/verification-notification',
[EmailVerificationNotificationController::class, 'store'])
                ->middleware('throttle:6,1')
                ->name('verification.send');

    Route::get('confirm-password', [ConfirmablePasswordController::class,
'show'])
                ->name('password.confirm');

    Route::post('confirm-password',
[ConfirmablePasswordController::class, 'store']);

    Route::put('password', [PasswordController::class, 'update'])-
>name('password.update');
```

```
    Route::post('logout', [AuthenticatedSessionController::class,
'destroy'])->name('logout');

    Route::middleware('check.access:admin')->group(function (){
        Route::get('users', [RegisteredUserController::class, 'index'])-
>name('users.index');
        Route::post('register', [RegisteredUserController::class,
'store']);
        Route::put('users/{user}', [RegisteredUserController::class,
'update'])->name('users.update');
        Route::get('register', [RegisteredUserController::class,
'create'])->name('register');

    });
});
```

2. Transações em Aplicações Laravel

Nas transações em laravel como begin, commit e rollback implemetei no controlador que gerência funcionários.

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Crypt;
use App\Models\Funcionario;
use App\Services\LogService; // Importação do LogService

class FuncionarioController extends Controller
{
    public function index()
    {

        $funcionarios = Funcionario::all();
        return view('funcionario.index', compact('funcionarios'));
    }

    public function create()
    {

        return view('funcionarios.create');
    }

    public function store(Request $request)
    {
        $request->validate([
```

```
            'nome' => 'required|string|max:255',
            'password'=> 'required|min:8',
            'email' => 'required|email|unique:funcionarios',
            'contacto' => 'required|string|max:255',
            'idade' => 'required|integer',
            'sexo' => 'required|string',
            'bi' => 'required|string',
            'data_nascimento' => 'required|string',
            'data_expiracao' => 'required|string',
            'tipo_trabalho' => 'required|string',
            'cartao_credito' => 'required|string',
            'nib' => 'required|string',
            'rua' => 'required|string',
            'cidade' => 'required|string',
            'codigo_postal' => 'required|string',
            'pais' => 'required|string',
            'condicao_saude' => 'required|string',
            'medicamento' => 'required|string',
            'historico_comportamento' => 'required|string',
        ]);

        DB::beginTransaction();
        try {
            $funcionario = Funcionario::create($request->all());

            // Registrar log de sucesso
            LogService::record([
                'user_id' => auth()->id(),
                'user_name' => auth()->user()->name,
                'model' => Funcionario::class,
                'model_id' => $funcionario->id,
                'action' => 'create',
                'changes' => $funcionario->toArray(),
                'status' => 'success',
                'message' => 'Funcionário criado com sucesso.',
                'ip_address' => $request->ip(),
            ]);

            DB::commit(); // Confirma as alterações no banco de dados

            return redirect()->route('funcoinarios.index')-
>with('success', 'funcoinario criado com sucesso!');
        } catch (\Exception $e) {
            DB::rollBack(); // Reverte as alterações no banco de dados em
caso de erro

            // Registrar log de erro
            LogService::record([
                'user_id' => auth()->id(),
                'user_name' => auth()->user()->name,
                'model' => Funcionario::class,
```

```
                'action' => 'create',
                'status' => 'error',
                'message' => 'Erro ao criar funcionario: ' . $e->getMessage(),
                'ip_address' => $request->ip(),
            ]);

            return redirect()->back()->with('error', 'Erro ao criar funcoinario.');
        }
    }

    public function show(Funcionario $funcionario)
    {
        return view('funcionarios.show', compact('funcionario'));
    }

    public function edit(Funcionario $funcionario)
    {
        return view('funcionarios.edit', compact('funcionario'));
    }

    public function update(Request $request, Funcionario $funcionario)
    {
        // Validação dos campos
        $request->validate([
            'nome' => 'required|string|max:255',
            'password' => 'nullable|min:8',
            'email' => 'required|email|unique:funcionarios,email,' . $funcionario->id,
            'contacto' => 'required|string|max:255',
            'idade' => 'required|integer',
            'sexo' => 'required|string',
            'bi' => 'required|string',
            'data_nascimento' => 'required|string',
            'data_expiracao' => 'required|string',
            'tipo_trabalho' => 'required|string',
            'cartao_credito' => 'required|string',
            'nib' => 'required|string',
            'rua' => 'required|string',
            'cidade' => 'required|string',
            'codigo_postal' => 'required|string',
            'pais' => 'required|string',
            'condicao_saude' => 'required|string',
            'medicamento' => 'required|string',
            'historico_comportamento' => 'required|string',
        ]);

        try {
            // Iniciar transação
            DB::beginTransaction();
```

```
            $data = $request->all();

            // Encriptação do e-mail
            if ($request->filled('email')) {
                $data['email'] = Crypt::encryptString($data['email']);
            }

            // Atualização da senha apenas se fornecida
            if ($request->filled('password')) {
                $data['password'] = bcrypt($data['password']);
            }

            // Atualizar os dados do funcionário
            $funcionario->update($data);

            // Commit da transação
            DB::commit();

            return redirect()->route('funcionarios.index')-
>with('success', 'Funcionário atualizado com sucesso.');
        } catch (\Exception $e) {
            // Reverter transação em caso de erro
            DB::rollBack();
            return redirect()->route('funcionarios.index')->with('error',
'Erro ao atualizar o funcionário: ' . $e->getMessage());
        }
    }

    public function destroy(Funcionario $funcionario)
    {
        try {
            // Iniciar transação
            DB::beginTransaction();

            // Apagar o funcionário
            $funcionario->delete();

            // Commit da transação
            DB::commit();

            return redirect()->route('funcionarios.index')-
>with('success', 'Funcionário removido com sucesso.');
        } catch (\Exception $e) {
            // Reverter transação em caso de erro
            DB::rollBack();
            return redirect()->route('funcionarios.index')->with('error',
'Erro ao remover o funcionário: ' . $e->getMessage());
        }
    }
```

```
}
```

3. Single Sing On (SSO)

3.1. Criar uma conta no Google Cloud console

- **Definir as rotas do inicio do projeto**
  - **http://localhost:8000**
- **Definir a rota de callback do token de acesso**
  - **http://localhost:8000/auth/google/callback**

3.2. Criar arquivo services.php na pasta: App\Services

```
'google' => [
    'client_id' => env('GOOGLE_CLIENT_ID'),
    'client_secret' => env('GOOGLE_CLIENT_SECRET'),
    'redirect' => env('GOOGLE_REDIRECT_URI'),
],
```

3.3. Adicionar as configurações do API no arquivo .env

```
GOOGLE_CLIENT_ID=674139171592-
davttf15d5p5lt36gtborq0sf21ve8p7.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=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
GOOGLE_REDIRECT_URI=http://localhost:8000/auth/google/callback
```

3.4. Modelo de usuários

```
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Illuminate\Support\Facades\Crypt;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
```

```
    protected $fillable = [
        'name',
        'email',
        'password',
        'nivel_acesso',
        'google_id',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
        'two_factor_enabled' => 'boolean',
        'two_factor_expires_at' => 'datetime',
    ];

}
```

3.5. Adicionar as rotas

```
<?php

use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\ConfirmablePasswordController;
use App\Http\Controllers\Auth\EmailVerificationNotificationController;
use App\Http\Controllers\Auth\EmailVerificationPromptController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth\PasswordController;
use App\Http\Controllers\Auth\PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use Illuminate\Support\Facades\Route;
use Laravel\Socialite\Facades\Socialite;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Http;


      Route::get('/auth/google', function () {
        $query = http_build_query([
            'client_id' => env('GOOGLE_CLIENT_ID'),
            'redirect_uri' => env('GOOGLE_REDIRECT_URI'),
            'response_type' => 'code',
            'scope' => 'email profile',
            'access_type' => 'offline',
```

```
            'prompt' => 'consent',
        ]);

        return redirect('https://accounts.google.com/o/oauth2/v2/auth?' .
$query);
    })->name('google.login');

    Route::get('/auth/google/callback', function
(\Illuminate\Http\Request $request) {
        $code = $request->input('code');

        if (!$code) {
            return redirect('/login')->with('error', 'Não foi possível
autenticar com o Google.');
        }

        // Trocar o código por um token de acesso
        $response = Http::asForm()-
>post('https://oauth2.googleapis.com/token', [
            'client_id' => env('GOOGLE_CLIENT_ID'),
            'client_secret' => env('GOOGLE_CLIENT_SECRET'),
            'redirect_uri' => env('GOOGLE_REDIRECT_URI'),
            'grant_type' => 'authorization_code',
            'code' => $code,
        ]);

        $data = $response->json();

        if (!isset($data['access_token'])) {
            return redirect('/login')->with('error', 'Erro ao obter o
token de acesso.');
        }

        // Obter informações do utilizador com o token de acesso
        $userResponse = Http::withHeaders([
            'Authorization' => 'Bearer ' . $data['access_token'],
        ])->get('https://www.googleapis.com/oauth2/v2/userinfo');

        $googleUser = $userResponse->json();

        // Processar o utilizador no sistema
        $user = \App\Models\User::where('email', $googleUser['email'])-
>first();

        if (!$user) {
            $user = \App\Models\User::create([
                'name' => $googleUser['name'],
                'email' => $googleUser['email'],
                'google_id' => $googleUser['id'],
                'password' => bcrypt('senha_aleatoria'), // Opcional
            ]);
```

```
        }

        // Fazer login do utilizador
        \Illuminate\Support\Facades\Auth::login($user);

        return redirect('/dashboard');
    });
```

3.6. Botão na interface

```
<!-- Botão para Google -->
            <div class="flex space-x-4">
                <a href="{{ route('google.login') }}"
                    class="w-full flex items-center justify-center bg-
white text-gray-800 border border-gray-300 py-3 rounded-lg hover:bg-gray-
100 shadow">
                    <img src="{{ asset('img/google.png') }}" alt="Google
Logo" class="h-10 w-10 mr-2">

                    <span class="font-medium">Entrar com Google</span>
                </a>
            </div>
```

4. Criptografia e protencão de dados

Modelo Funcionario foi implemtado para codificar descodificar os campos que se considera sensíveis.

```
<?php

namespace App\Models;

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

class Funcionario extends Model
{
    use HasFactory;

    // Campos permitidos para preenchimento em massa
    protected $fillable = [
        'nome',
        'password',
        'email',
        'contacto',
        'idade',
        'sexo',
        'bi',
        'data_nascimento',
```

```
        'data_expiracao',
        'tipo_trabalho',
        'cartao_credito',
        'nib',
        'rua',
        'cidade',
        'codigo_postal',
        'pais',
        'condicao_saude',
        'medicamento',
        'historico_comportamento',
    ];

    // Método para criptografar automaticamente os campos sensíveis
    public function setAttribute($key, $value)
    {
        $encryptedFields =
['password','contacto','tipo_trabalho','data_expiracao', 'bi',
'cartao_credito', 'nib', 'condicao_saude', 'medicamento',
'historico_comportamento'];

        if (in_array($key, $encryptedFields) && $value) {
            $this->attributes[$key] = encrypt($value);
        } else {
            $this->attributes[$key] = $value;
        }
    }

    // Método para descriptografar os campos sensíveis
    public function getAttribute($key)
    {
        $encryptedFields =
['password','contacto','tipo_trabalho','data_expiracao', 'bi',
'cartao_credito', 'nib', 'condicao_saude', 'medicamento',
'historico_comportamento'];

        if (in_array($key, $encryptedFields) && isset($this-
>attributes[$key])) {
            return decrypt($this->attributes[$key]);
        }

        return $this->attributes[$key] ?? null;
    }
}
```

Integracão no controlador

```
<?php
namespace App\Http\Controllers;
```

```php
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Crypt;
use App\Models\Funcionario;
use App\Services\LogService; // Importação do LogService

class FuncionarioController extends Controller
{
    public function index()
    {

        $funcionarios = Funcionario::all();
        return view('funcionario.index', compact('funcionarios'));
    }

    public function create()
    {

        return view('funcionarios.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'nome' => 'required|string|max:255',
            'password'=> 'required|min:8',
            'email' => 'required|email|unique:funcionarios',
            'contacto' => 'required|string|max:255',
            'idade' => 'required|integer',
            'sexo' => 'required|string',
            'bi' => 'required|string',
            'data_nascimento' => 'required|string',
            'data_expiracao' => 'required|string',
            'tipo_trabalho' => 'required|string',
            'cartao_credito' => 'required|string',
            'nib' => 'required|string',
            'rua' => 'required|string',
            'cidade' => 'required|string',
            'codigo_postal' => 'required|string',
            'pais' => 'required|string',
            'condicao_saude' => 'required|string',
            'medicamento' => 'required|string',
            'historico_comportamento' => 'required|string',
        ]);

        DB::beginTransaction();
        try {
            $funcionario = Funcionario::create($request->all());

            // Registrar log de sucesso
            LogService::record([
```

```
                'user_id' => auth()->id(),
                'user_name' => auth()->user()->name,
                'model' => Funcionario::class,
                'model_id' => $funcionario->id,
                'action' => 'create',
                'changes' => $funcionario->toArray(),
                'status' => 'success',
                'message' => 'Funcionário criado com sucesso.',
                'ip_address' => $request->ip(),
            ]);

            DB::commit(); // Confirma as alterações no banco de dados

            return redirect()->route('funcoinarios.index')-
>with('success', 'funcoinario criado com sucesso!');
        } catch (\Exception $e) {
            DB::rollBack(); // Reverte as alterações no banco de dados em
caso de erro

            // Registrar log de erro
            LogService::record([
                'user_id' => auth()->id(),
                'user_name' => auth()->user()->name,
                'model' => Funcionario::class,
                'action' => 'create',
                'status' => 'error',
                'message' => 'Erro ao criar funcionario: ' . $e-
>getMessage(),
                'ip_address' => $request->ip(),
            ]);

            return redirect()->back()->with('error', 'Erro ao criar
funcoinario.');
        }
    }

    public function show(Funcionario $funcionario)
    {
        return view('funcionarios.show', compact('funcionario'));
    }

    public function edit(Funcionario $funcionario)
    {
        return view('funcionarios.edit', compact('funcionario'));
    }

    public function update(Request $request, Funcionario $funcionario)
    {
        // Validação dos campos
        $request->validate([
            'nome' => 'required|string|max:255',
```

```
            'password' => 'nullable|min:8',
            'email' => 'required|email|unique:funcionarios,email,' .
$funcionario->id,
            'contacto' => 'required|string|max:255',
            'idade' => 'required|integer',
            'sexo' => 'required|string',
            'bi' => 'required|string',
            'data_nascimento' => 'required|string',
            'data_expiracao' => 'required|string',
            'tipo_trabalho' => 'required|string',
            'cartao_credito' => 'required|string',
            'nib' => 'required|string',
            'rua' => 'required|string',
            'cidade' => 'required|string',
            'codigo_postal' => 'required|string',
            'pais' => 'required|string',
            'condicao_saude' => 'required|string',
            'medicamento' => 'required|string',
            'historico_comportamento' => 'required|string',
        ]);

        try {
            // Iniciar transação
            DB::beginTransaction();

            $data = $request->all();

            // Encriptação do e-mail
            if ($request->filled('email')) {
                $data['email'] = Crypt::encryptString($data['email']);
            }

            // Atualização da senha apenas se fornecida
            if ($request->filled('password')) {
                $data['password'] = bcrypt($data['password']);
            }

            // Atualizar os dados do funcionário
            $funcionario->update($data);

            // Commit da transação
            DB::commit();

            return redirect()->route('funcionarios.index')-
>with('success', 'Funcionário atualizado com sucesso.');
        } catch (\Exception $e) {
            // Reverter transação em caso de erro
            DB::rollBack();
            return redirect()->route('funcionarios.index')->with('error',
'Erro ao atualizar o funcionário: ' . $e->getMessage());
        }
```

```
    }

    public function destroy(Funcionario $funcionario)
    {
        try {
            // Iniciar transação
            DB::beginTransaction();

            // Apagar o funcionário
            $funcionario->delete();

            // Commit da transação
            DB::commit();

            return redirect()->route('funcionarios.index')-
>with('success', 'Funcionário removido com sucesso.');
        } catch (\Exception $e) {
            // Reverter transação em caso de erro
            DB::rollBack();
            return redirect()->route('funcionarios.index')->with('error',
'Erro ao remover o funcionário: ' . $e->getMessage());
        }
    }

}
```

5. Testes unitários com phpUnit

Testes de controlador de usuários.

```
<?php

namespace Tests\Feature;

use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Foundation\Testing\WithFaker;
use Tests\TestCase;
use App\Models\User;

class RegisteredUserControllerTest extends TestCase
{


    public function test_user_can_be_created_successfully()
    {
        // Criar um utilizador administrador para autenticação
        $admin = User::factory()->create(['nivel_acesso' => 'admin']);
        $this->actingAs($admin);
```

```
        // Gerar um email único
        $email = 'user' . uniqid() . '@example.com';

        // Enviar a requisição para criar um utilizador
        $response = $this->post(route('register'), [
            'name' => 'John Doe',
            'email' => $email,
            'password' => 'password123',
            'password_confirmation' => 'password123',
            'nivel_acesso' => 'operador',
        ]);

        // Verificar o redirecionamento e a mensagem de sucesso
        $response->assertRedirect(route('users.index'))
                ->assertSessionHas('success', 'Utilizador criado com
sucesso!');

        // Garantir que o utilizador foi adicionado à base de dados
        $this->assertDatabaseHas('users', [
            'email' => $email,
            'name' => 'John Doe',
            'nivel_acesso' => 'operador',
        ]);
    }

    #Teste de Atualização de Utilizador
    public function test_user_can_be_updated_successfully()
    {
        $admin = User::factory()->create(['nivel_acesso' => 'admin']);
        $this->actingAs($admin);

        $user = User::factory()->create();

        $response = $this->put(route('users.update', $user), [
            'name' => 'Updated Name',
            'email' => $user->email,
            'password' => 'newpassword123',
            'password_confirmation' => 'newpassword123',
            'nivel_acesso' => 'admin',
        ]);

        $response->assertRedirect(route('users.index'))
                ->assertSessionHas('success', 'Utilizador atualizado com
sucesso!');

        $this->assertDatabaseHas('users', [
            'id' => $user->id,
            'name' => 'Updated Name',
            'nivel_acesso' => 'admin',
        ]);
    }
```

6. Logs de auditoria

Controlador de logs: LogController.php

```php
<?php
namespace App\Http\Controllers;

use App\Models\Log;

class LogController extends Controller
{
    public function index()
    {
        $logs = Log::latest()->paginate(20); // Recupera os logs com
paginação
        return view('logs.index', compact('logs')); // Passa os logs para
a view
    }
}
```

Serviços de logs: App\Services\LogService.php

```php
<?php

namespace App\Services;

use App\Models\Log;
use Illuminate\Support\Facades\Auth;

class LogService
{
    public static function record(array $data)
    {
        Log::create([
            'user_id' => $data['user_id'] ?? Auth::id(),
            'user_name' => $data['user_name'] ?? Auth::user()->name ??
'Sistema',
            'model' => $data['model'] ?? null,
            'model_id' => $data['model_id'] ?? null,
            'action' => $data['action'] ?? 'undefined',
            'changes' => $data['changes'] ?? [],
            'status' => $data['status'] ?? 'info',
            'message' => $data['message'] ?? 'Nenhuma mensagem
fornecida.',
            'ip_address' => $data['ip_address'] ?? request()->ip(),
```

```
        ]);
    }
}
```

Modelo de logs: Log.php

```php
<?php

namespace App\Models;

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

class Log extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',     // ID do usuário que realizou a ação
        'user_name',   // Nome do usuário que realizou a ação
        'ip_address',  // Endereço IP do usuário
        'model',       // Nome do modelo afetado
        'model_id',    // ID do registro afetado
        'action',      // Ação realizada (create, update, delete, etc.)
        'changes',     // Alterações feitas
        'status',      // Status da ação (success, error, info)
        'message',     // Descrição da ação
    ];

    protected $casts = [
        'changes' => 'array',
    ];

    // Relacionamento com o usuário
    public function user()
    {
        return $this->belongsTo(User::class, 'user_id');
    }

    /**
     * Obter o nome do usuário associado ao log.
     */
    public function getUserNameAttribute()
    {
        return $this->user->name ?? 'Sistema';
    }

    /**
     * Formatar a data de criação do log.
     */
```

```
    public function getFormattedCreatedAtAttribute()
    {
        return $this->created_at->format('d/m/Y H:i:s');
    }
}
```

Implementação nos controladores para registar todas as interacoes

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use App\Providers\RouteServiceProvider;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules;
use Illuminate\View\View;
use App\Services\LogService; // Importação do LogService

class RegisteredUserController extends Controller
{
    /**
     * Exibir todos os utilizadores.
     */
    public function index(Request $request)
    {
        $users = User::all(); // Lista de utilizadores
        $editUser = null;

        if ($request->has('edit')) {
            $editUser = User::find($request->edit); // Carrega o
utilizador a ser editado
        }

        return view('funcionario.user_index', compact('users',
'editUser'));
    }

    /**
     * Exibir a página de registo.
     */
    public function create(): View
    {
```

```
        return view('auth.register');
    }

    /**
     * Criar um novo utilizador.
     */
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email',
'max:255', 'unique:users,email'],
            'password' => ['required', 'confirmed',
Rules\Password::defaults()],
            'nivel_acesso' => ['required', 'in:admin,gerente,operador'],
// Validação para o nível de acesso
        ]);

        DB::beginTransaction();
        try {
            $user = User::create([
                'name' => $request->name,
                'email' => $request->email,
                'password' => Hash::make($request->password),
                'nivel_acesso' => $request->nivel_acesso,
            ]);

            event(new Registered($user));

            // Registar log da criação do utilizador
            LogService::record([
                'user_id' => Auth::id(),
                'user_name' => Auth::user()->name,
                'model' => User::class,
                'model_id' => $user->id,
                'action' => 'create',
                'changes' => $user->getAttributes(),
                'message' => "Utilizador {$user->name} criado com
sucesso.",
                'ip_address' => $request->ip(),
                'status' => 'success',
            ]);

            DB::commit(); // Confirma as alterações
            return redirect()->route('users.index')->with('success',
'Utilizador criado com sucesso!');
        } catch (\Exception $e) {
            DB::rollBack(); // Reverte as alterações

            // Registar log de erro
            LogService::record([
```

```
                'user_id' => Auth::id(),
                'user_name' => Auth::user()->name,
                'model' => User::class,
                'action' => 'create',
                'message' => "Erro ao criar utilizador: {$e-
>getMessage()}",
                'ip_address' => $request->ip(),
                'status' => 'error',
            ]);

            return redirect()->back()->with('error', 'Erro ao criar
utilizador: ' . $e->getMessage());
        }
    }

    /**
     * Atualizar um utilizador existente.
     */
    public function update(Request $request, User $user):
RedirectResponse
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email',
'max:255', 'unique:users,email,' . $user->id],
            'nivel_acesso' => ['required', 'in:admin,gerente,operador'],
// Validação para o nível de acesso
            'password' => ['nullable', 'confirmed',
Rules\Password::defaults()],
        ]);

        DB::beginTransaction();
        try {
            $oldAttributes = $user->getAttributes();

            $user->update([
                'name' => $request->name,
                'email' => $request->email,
                'nivel_acesso' => $request->nivel_acesso,
                'password' => $request->password ? Hash::make($request-
>password) : $user->password,
            ]);

            // Registar log da atualização
            LogService::record([
                'user_id' => Auth::id(),
                'user_name' => Auth::user()->name,
                'model' => User::class,
                'model_id' => $user->id,
                'action' => 'update',
```

```
                'changes' => ['before' => $oldAttributes, 'after' =>
$user->getAttributes()],
                'message' => "Utilizador {$user->name} atualizado com
sucesso.",
                'ip_address' => $request->ip(),
                'status' => 'success',
            ]);

            DB::commit(); // Confirma as alterações
            return redirect()->route('users.index')->with('success',
'Utilizador atualizado com sucesso!');
        } catch (\Exception $e) {
            DB::rollBack(); // Reverte as alterações

            // Registar log de erro
            LogService::record([
                'user_id' => Auth::id(),
                'user_name' => Auth::user()->name,
                'model' => User::class,
                'action' => 'update',
                'message' => "Erro ao atualizar utilizador: {$e-
>getMessage()}",
                'ip_address' => $request->ip(),
                'status' => 'error',
            ]);

            return redirect()->back()->with('error', 'Erro ao atualizar
utilizador: ' . $e->getMessage());
        }
    }

    /**
     * Excluir um utilizador.
     */
    public function destroy(User $user): RedirectResponse
    {
        DB::beginTransaction();
        try {
            $userName = $user->name;
            $user->delete();

            // Registar log da exclusão
            LogService::record([
                'user_id' => Auth::id(),
                'user_name' => Auth::user()->name,
                'model' => User::class,
                'model_id' => $user->id,
                'action' => 'delete',
                'message' => "Utilizador {$userName} excluído.",
                'ip_address' => request()->ip(),
                'status' => 'success',
```

```
            ]);

            DB::commit(); // Confirma as alterações
            return redirect()->route('users.index')->with('success',
'Utilizador excluído com sucesso!');
        } catch (\Exception $e) {
            DB::rollBack(); // Reverte as alterações

            // Registar log de erro
            LogService::record([
                'user_id' => Auth::id(),
                'user_name' => Auth::user()->name,
                'model' => User::class,
                'action' => 'delete',
                'message' => "Erro ao excluir utilizador: {$e-
>getMessage()}",
                'ip_address' => request()->ip(),
                'status' => 'error',
            ]);

            return redirect()->back()->with('error', 'Erro ao excluir
utilizador: ' . $e->getMessage());
        }
    }
}
```

7. Aplicação de API

Nesta funcionalidade implementei um API de clima.

7.1 configurando o API Key no .env

```
TOMORROW_API_KEY=4mwFS66dox2YlxzcK5emQWAXXz3jDbiC

GEOCODING_API_KEY=ea9e9663b10d4910a57e65e229f42946
```

7.2. Criando a rota

```
Route::get('/clima', [WeatherController::class, 'showWeather'])-
>name('weather.show');
```

7.3. Configurando o controlador

```
<?php
```

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Cache;

class WeatherController extends Controller
{
    public function showWeather(Request $request)
    {
        $city = $request->input('city', 'Chibuto'); // Cidade padrão
        $apiKey = env('TOMORROW_API_KEY'); // Chave da API da Tomorrow.io

        // Obter as coordenadas da cidade
        $coordinates = $this->getCoordinates($city);

        if (!$coordinates) {
            return view('weather.index', [
                'error' => "Não foi possível obter as coordenadas para a
cidade: {$city}",
                'city' => $city,
                'weather' => null,
            ]);
        }

        // Cache para armazenar a resposta da API por 1 hora (3600
segundos)
        $weather = Cache::remember("weather_{$city}", 3600, function ()
use ($coordinates, $apiKey) {
            $response =
Http::get("https://api.tomorrow.io/v4/weather/realtime", [
                'location' =>
"{$coordinates['lat']},{$coordinates['lon']}", // Use lat e lon
                'apikey' => $apiKey,
                'headers' => [
                    'Accept' => 'application/json',
                ],
            ]);

            // Verificar se a resposta foi bem-sucedida e retornar os
dados
            if ($response->successful()) {
                return $response->json();
            }

            return null;
        });

        // Verifica se a API retornou dados válidos
        if (!$weather || !isset($weather['data'])) {
            return view('weather.index', [
```

```
                'error' => 'Não foi possível obter os dados de previsão
do clima.',
                'city' => $city,
                'weather' => null,
            ]);
        }

        // Dados do clima em tempo real
        $weatherData = $weather['data']['values'];

        return view('weather.index', [
            'weather' => $weatherData,
            'city' => $city,
            'error' => null,
        ]);
    }

    private function getCoordinates($city)
    {
        // Exemplo usando a API OpenCage para obter coordenadas
        $response =
Http::get('https://api.opencagedata.com/geocode/v1/json', [
            'q' => $city,
            'key' => env('GEOCODING_API_KEY'),
            'limit' => 1,
        ]);

        if ($response->successful() && isset($response-
>json()['results'][0]['geometry'])) {
            return [
                'lat' => $response-
>json()['results'][0]['geometry']['lat'],
                'lon' => $response-
>json()['results'][0]['geometry']['lng'], // OpenCage usa 'lng' em vez de
'lon'
            ];
        }

        return null; // Retorna null se não houver dados válidos
    }
}
```

7.4. COnfiguracao da view

```
@extends('layouts.app')

@section('content')
```

```
<div class="container py-4">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card text-white">
                <div class="card-body text-center" style="color: black;
background-image: url('/images/weather-bg.jpg'); background-size: cover;
background-position: center; border-radius: 15px;">
                    <h2 class="mb-4">Previsão do Clima em {{
ucfirst($city) }}</h2>

                    @if($error)
                        <p class="text-danger">{{ $error }}</p>
                    @elseif(isset($weather))
                        <p class="h4 mb-3">
                            <i class="fas fa-temperature-high"></i>
Temperatura: {{ $weather['temperature'] }}°C
                        </p>
                        <p class="h5 mb-3">
                            <i class="fas fa-tint"></i> Humidade: {{
$weather['humidity'] }}%
                        </p>
                        <p class="h5">
                            <i class="fas fa-wind"></i> Velocidade do
Vento: {{ $weather['windSpeed'] }} m/s
                        </p>
                    @else
                        <p class="text-danger">Não foi possível obter os
dados do clima.</p>
                    @endif

                    <form method="GET" action="{{ route('weather.show')
}}" class="mt-4">
                        <div class="input-group">
                            <input type="text" name="city" class="form-
control" placeholder="Digite o nome da cidade" value="{{ old('city',
$city) }}">
                            <button class="btn btn-primary"
type="submit">Procurar</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```