Documentação de Log para auditoria dos sistemas da Secretaria da Saúde do Estado da Bahia – **SESAB/CGTICS/DMA/COSIS**.

Andre Ottoni

Cristian dos Santos Gomes da Silva

Este módulo registra logs completos de requisições HTTP feitas por usuários autenticados em aplicações Laravel (Starter Pack). Ele captura dados como IP, rota acessada, método HTTP utilizado, código de status retornado, corpo da requisição e da resposta, além do estado anterior do recurso em operações como `PUT`, `PATCH` e `DELETE`.

A principal vantagem desse módulo é permitir que ele seja adaptado para qualquer sistema Laravel por meio de simples configurações, **sem alterar o core do código**. Ele também oferece filtros de permissão por perfil e controle de visibilidade de rotas para que apenas ações relevantes sejam registradas e exibidas.

Instalação

Para usar esse módulo, basta copiar os arquivos para seus respectivos diretórios:

- `app/Http/Middleware/AccessLogMiddleware.php` – Middleware que intercepta as requisições e registra os dados.
- `app/Http/Controllers/Api/LogController.php` – Controller que fornece endpoints para leitura dos logs.
- `app/Services/AccessLogService.php` – Serviço que contém as regras de negócio e formatação dos logs.
- `app/Models/AccessLog.php` – Model correspondente à tabela de logs.
- Configurações: copiar todos os arquivos `config/log_*.php` para a pasta `config/`.

📦 Estrutura do Módulo: `modules/LogModule/`

```
modules/
└── LogModule/
    ├── Controllers/
    │   └── Api
    │       └── LogController.php
    ├── Middleware/
    │   └── AccessLogMiddleware.php
    ├── Models/
    │   └── AccessLog.php
    ├── Services/
    │   └── AccessLogService.php
    ├── Config/
    │   ├── log_access_filter.php
    │   ├── log_http_verbs.php
    │   ├── log_models_mapping.php
    │   ├── log_routes_mapping.php
    │   └── log_status.php
    ├── database/
    │   └── migrations/
    │       └── 2025_01_01_000000_create_access_logs_table.php
    ├── routes/
    │   └── api.php
    └── README.md
```

- Migration: Crie a migration (‘php artisan make:migration CreateAccessLogsTable’) e copiar as informações do arquivo de migration `create_access_logs_table.php` para o arquivo que criado.

- Após isso, registre o middleware em `App\Http\Kernel.php`, dentro do grupo `(protected $routeMiddleware =`
  `Adicione -> 'log.access' =>`
  `\App\Http\Middleware\AccessLogMiddleware::class`),

- Insira o `log.access` no vetor de middlewares, após o `sanctum`.

```
 Route::group(['middleware' =>
['auth:sanctum','refreshTokenSanctum','log.access']]
```

- Insira as rotas no arquivo `api.php`, conforme o modelo ilustrado abaixo dentro do `Route::group` middleware

```
Route::get('logs/formatted',[LogController::class,'getFormattedLogs']);
Route::get('logs/search',[LogController::class,'search']);
```

- Insira o caminho do LogController no topo do projeto

```
 // Insira o LogController no início do arquivo

 use App\Http\Controllers\Api\LogController;
```

- Em seguida, execute o comando `php artisan migrate` para criar a tabela no banco de dados.

Configuração

A configuração da API de logs é feita exclusivamente através dos arquivos da pasta `config/`:

- `log_access_filter.php`:
- `routes_block_display`: define quais rotas devem ser ocultadas;
- `routes_permission`: define quais exigem permissão;
- `allowed_profiles`: define quais perfis de usuário podem gravar logs.

- `log_http_verbs.php`: define como cada verbo HTTP (GET, POST, PUT etc.) será descrito nos logs em casos de sucesso ou falha.

- `log_status.php`: define mensagens para códigos HTTP (200, 401, 404 etc.), com um booleano que indica sucesso ou falha.

- `log_models_mapping.php`: define qual *model* será usado para capturar o estado anterior de um recurso. Cada rota de alteração precisa estar mapeada aqui.

- `log_routes_mapping.php`: associa rotas e verbos a descrições legíveis, para que o log não exiba apenas "api/users/1 [PUT]", mas sim algo como "atualizou um usuário".

Exemplo de Registro no Log

Quando um usuário autenticado faz uma requisição do tipo PUT para `/api/users/42`, o middleware captura:

- IP e ID do usuário
- Caminho: `api/users/42`
- Método: `PUT`
- Status: `200`, `422` etc.
- Corpo da requisição (ex: os dados atualizados)
- Corpo da resposta
- Estado anterior do recurso (ex: `User::find(42)->toArray()`)

O serviço formata essas informações e salva na tabela `access_logs`.

Consulta de Logs

A leitura dos logs pode ser feita via dois endpoints:

```
GET /api/logs/formatted
```

Retorna os **50 logs** mais recentes, já formatados. Exemplo de resposta:

```
[
  {
    "message": "O(A) Ana Silva do(a) Unidade A, atualizou um perfil com
sucesso.",
    "dateTime": "14/04/2025 às 10h33"
  }
]
```

```
GET /api/logs/search
```

Aceita os seguintes filtros via query string ou JSON:

```
{
  "name": "ana",
  "start_date": "01/01/2025",
  "end_date": "02/01/2025",
  "method": "POST",
  "status": "200",
  "path": "api/users"
}
```

O sistema retorna os logs filtrados com as mesmas mensagens descritivas usadas no `getFormattedLogs()`.

Personalização

Para incluir uma nova entidade ou rota no log, você só precisa:

1. Adicionar a rota e seu respectivo ***`model`*** em `log_models_mapping.php` (com `primary_key`);

2. Adicionar a mesma rota e verbo em `log_routes_mapping.php`, com a descrição desejada (ex: `'PUT' => 'um paciente atualizado'`);

3. Se necessário, incluir o método HTTP e as mensagens no `log_http_verbs.php`.

Se quiser evitar que determinada rota apareça nos logs visuais, adicione em `routes_block_display` no `log_access_filter.php`.

Requisitos

- Laravel 8 ou superior
- PHP 7.4+
- Middleware ativado no grupo `api`
- Tabela `access_logs` migrada no banco de dados

Licença

Secretária de Saúde do Estado da Bahia - SESAB/CGTICS/DMA/COSIS