Aqui está o guia para implementar a comunicação com a API do TMDB (The Movie Database) utilizando Guzzle no Laravel:

Passos para configurar o projeto:

1. **Obter a chave da API do TMDB**:
   - O TMDB (The Movie Database) é uma plataforma que oferece informações sobre filmes e séries, e você pode acessar dados usando a API pública.
   - Para obter a chave de API, inscreva-se no TMDB (https://www.themoviedb.org) e crie um projeto na sua conta. Isso lhe fornecerá uma chave API.

2. **Configurar as chaves no Laravel**:
   - No arquivo `config/services.php`, adicione o seguinte código para configurar as chaves da API:

```
return [
    'tmdb' => [
        'key' => env('TMDB_API_API'),
        'read_key' => env('TMDB_API_READ_API'),
    ],
];
```

   - No arquivo `.env`, adicione as chaves que você obteve no site do TMDB:

```
TMDB_API_API=YOUR_TMDB_API_KEY
TMDB_API_READ_API=YOUR_TMDB_READ_API_KEY
```

3. **Criar o HomeController**:
   - Para permitir listagem de filmes populares e busca por filmes:

```php
<?php

namespace App\Http\Controllers;

use GuzzleHttp\Client;
use Illuminate\View\View;
use Illuminate\Http\Request;

class HomeController extends Controller
{
    public function __invoke(Request $request): View
    {
        // Verifica se a busca foi realizada
        if ($request->filled('query')) {
            $query = $request->get('query');
            $apiUrl = "https://api.themoviedb.org/3/search/movie?query={$query}&include_adult=false&include_video=false&l
        } else {
            $apiUrl = "https://api.themoviedb.org/3/discover/movie?include_adult=false&include_video=false&language=pt&pa
        }

        // Adiciona a paginação
        if ($request->has('page')) {
            $apiUrl = $apiUrl . "&page=" . $request->get('page', 1);
        }

        $client = new Client([
            'accept' => 'application/json',
            'Authorization' => "Bearer " . config('services.tmdb.read_key')
        ]);

        try {
            // Requisição GET para a API do TMDB
            $response = $client->get($apiUrl);
            $data = json_decode($response->getBody(), true);

            return view('home', ['data' => $data]);
        } catch (\Exception $e) {
            return view('error', ['error' => $e->getMessage()]);
        }
    }
}
```

4. **Views para exibir filmes**:

   - Na **view home**, liste os filmes usando o loop de resultados `$data['results']` e calcule a metadata de paginação:

```php
@php
    $page = $data['page'];
    $total_pages = $data['total_pages'];
    $total_results = $data['total_results'];
    $items_per_page = ceil($total_results / $total_pages);
    $start_index = ($page - 1) * $items_per_page + 1;
    $end_index = min($start_index + $items_per_page - 1, $total_results);
@endphp
```

   - Adicione a paginação com os botões "Anterior" e "Próxima":

```
@if ($data['total_pages'] > 1)
    <div class="d-flex mt-4">
        @php
            $page = $data['page'];
            $next_page = $page + 1;
            $previous_page = $page - 1;
            $previous_page = max($previous_page, 1);
            $next_page = min($next_page, $total_pages);
            $previous_disabled = $page == 1;
            $next_disabled = $page == $total_pages;
        @endphp
        <ul class="pagination ms-auto">
            <li class="page-item">
                <a @class(['page-link', 'disabled' => $previous_disabled]) @disabled($previous_disabled)
                   href="{{ route('home', ['page' => $previous_page]) }}" tabindex="-1">
                    Anterior
                </a>
            </li>
            <li class="page-item">
                <a @class(['page-link', 'disabled' => $next_disabled]) href="{{ route('home', ['page' => $next_page]) }}"
                   @disabled($next_disabled)>
                    Próxima
                </a>
            </li>
        </ul>
    </div>
@endif
```

5. **Criar o MovieController**:

   - Para exibir detalhes de um filme:

```
<?php

namespace App\Http\Controllers;

use GuzzleHttp\Client;
use Illuminate\View\View;

class MovieController extends Controller
{
    public function show(string $id): View
    {
        $apiUrl = "https://api.themoviedb.org/3/movie/" . $id . "?language=pt&api_key=" . config('services.tmdb.key');

        $client = new Client([
            'accept' => 'application/json',
            'Authorization' => "Bearer " . config('services.tmdb.read_key')
        ]);

        try {
            $response = $client->get($apiUrl);
            $data = json_decode($response->getBody(), true);

            return view('movies.show', ['data' => $data]);
        } catch (\Exception $e) {
            return view('error', ['error' => $e->getMessage()]);
        }
    }
}
```

6. **Exibir detalhes de um filme**:

   - Na **view movie**, exiba as informações obtidas pelo controlador de detalhes do filme. usando a variavel `$data`