Helder Neves

hmrneves@gmail.com

Helder’s api

Uma simples API direcionada para um blog baseada na framework Laravel.

Índice

Conteúdo

[Introdução 3](#_Toc45575517)

[Planeamento 4](#_Toc45575518)

[Base de Dados 5](#_Toc45575519)

[Diagrama ER 5](#_Toc45575520)

[Funcionalidades 6](#_Toc45575521)

[Autenticação 8](#_Toc45575522)

[Documentação da API 9](#_Toc45575523)

[Autenticação 9](#_Toc45575524)

[*Registo* 9](#_Toc45575525)

[*Login* 9](#_Toc45575526)

[*Logout* 10](#_Toc45575527)

[*Logout de todos os dispositivos* 10](#_Toc45575528)

[*Reposição de Password* 10](#_Toc45575529)

[*Criar Nova Password* 11](#_Toc45575530)

[Categorias 12](#_Toc45575531)

[*Listar Todas as Categorias* 12](#_Toc45575532)

[*Listar uma Categoria* 12](#_Toc45575533)

[*Registar Categoria* 12](#_Toc45575534)

[*Atualizar Dados de uma Categoria* 13](#_Toc45575535)

[*Eliminar uma Categoria* 13](#_Toc45575536)

[Posts 14](#_Toc45575537)

[*Listar Todos os Posts* 14](#_Toc45575538)

[*Obter Dados de um Post* 14](#_Toc45575539)

[*Regista um Novo Post na Base de Dados* 14](#_Toc45575540)

[*Atualiza um Post existente na Base de Dados* 15](#_Toc45575541)

[*Elimina um post* 15](#_Toc45575542)

[Comentários 16](#_Toc45575543)

[*Registar um comentário* 16](#_Toc45575544)

[*Atualiza um Comentário* 16](#_Toc45575545)

[*Elimina um Comentário* 17](#_Toc45575546)

[Pesquisa 17](#_Toc45575547)

[*Por Keyword* 17](#_Toc45575548)

[*Por Categoria* 17](#_Toc45575549)

[Logs 18](#_Toc45575550)

[Subscritores 18](#_Toc45575551)

[*Listar Todos* 18](#_Toc45575552)

[*Subscrever* 18](#_Toc45575553)

[*Pedir Token Para Cancelamento da Subscrição* 19](#_Toc45575554)

[*Cancelar Subscrição* 19](#_Toc45575555)

[Administração – Gerir Posts 20](#_Toc45575556)

[*Listar Todos os Posts* 20](#_Toc45575557)

[*Listar Posts a Aguardar Aprovação* 20](#_Toc45575558)

[*Aprovar Post Pendente* 20](#_Toc45575559)

[*Eliminar Post Pendente* 21](#_Toc45575560)

[Administração – Gerir Comentários 22](#_Toc45575561)

[*Listar Comentários Pendentes* 22](#_Toc45575562)

[*Aprovar Comentário Pendente* 22](#_Toc45575563)

[*Elimina Comentário Pendente* 22](#_Toc45575564)

[Administração – Gerir Utilizadores 23](#_Toc45575565)

[*Listagem de Utilizadores* 23](#_Toc45575566)

[*Listagem dos Posts de um Utilizador* 23](#_Toc45575567)

[*Listagem dos Comentários de um Utilizador* 23](#_Toc45575568)

[*Atualiza o Grupo de um Utilizador* 24](#_Toc45575569)

[Administração – Grupos de Utilizadores 24](#_Toc45575570)

[*Listagem* 24](#_Toc45575571)

[Utilizadores 24](#_Toc45575572)

[*Ver Perfil de Utilizador* 24](#_Toc45575573)

[*Atualizar Email* 25](#_Toc45575574)

[*Atualizar Password* 25](#_Toc45575575)

[*Atualizar Avatar* 26](#_Toc45575576)

Introdução

Este projeto levou a cabo o desenvolvimento de uma API[[1]](#footnote-1) com base ideológica de um blog.

Planeamento

O planeamento deste projeto não seguiu a ordem habitual, ou seja, não foram definidas metas de desenvolvimento e nem prazos, uma vez que foi desenvolvido à medida da aquisição de conhecimentos.

Base de Dados

O sistema de gestão de dados deste projeto está assente em MySQL.

Diagrama ER[[2]](#footnote-2)

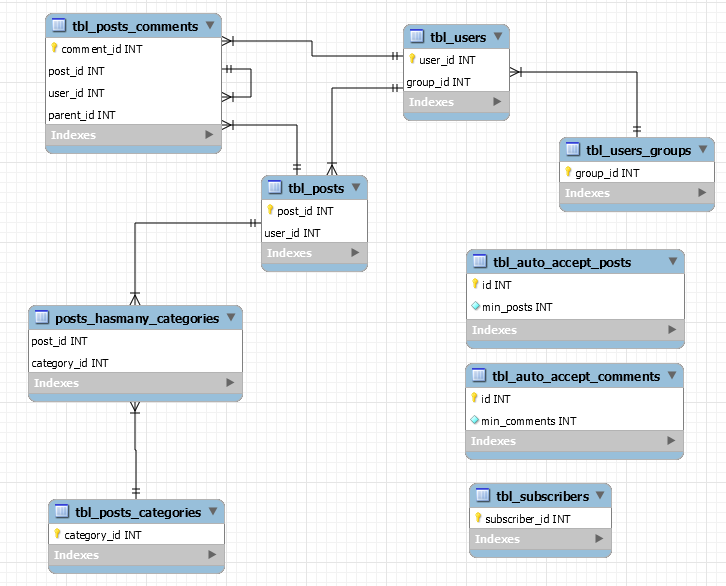


Figura 1 - Diagrama ER

Pelo diagrama, podemos verificar que o projeto conta com várias relações de 1:N[[3]](#footnote-3) e uma relação de N:M[[4]](#footnote-4).

Funcionalidades

Uma aplicação de front-end, poderá ter acesso às seguintes funcionalidades:

1. Posts – Criar, ler, atualizar e eliminar posts
2. Comentários
   1. Registar um comentário a um post
   2. Atualizar um post
   3. Eliminar um post
3. Categorias – Criar, ler, atualizar e eliminar categorias
4. Subscritores
   1. Listar todos os subscritores
   2. Subscrever
   3. Pedir token para cancelar subscrição
   4. Cancelar subscrição por token
5. Pesquisar posts
   1. Por keyword no título
   2. Por categoria
6. Utilizadores
   1. Ver perfil de utilizador
   2. Atualizar email
   3. Atualizar password
   4. Atualizar avatar
7. Logs
   1. Analisar logs de um utilizador
8. Autenticação
   1. Registar novo utilizador
   2. Login
   3. Logout do dispositivo
   4. Logout de todos os dispositivos
   5. Pedir token para efetuar reposição de password
   6. Repor password através de token
9. Gestão
   1. Gerir posts pendentes: lista-los, aprová-los e eliminá-los
   2. Gerir comentários pendentes: lista-los, aprová-los e eliminá-los
   3. Analisar atividade de um utilizador
   4. Atualizar grupo de um utilizador

Autenticação

Ao invés das sessões que as aplicações web usam, as API utilizam tokens para autenticar os utilizadores e não mantêm estados das sessões entre chamadas.

A API usa o pacote Laravel Passport[[5]](#footnote-5) para efetuar toda a gestão de autenticação por tokens.

Documentação da API

Abaixo será descrito as possíveis chamadas à API, bem como os métodos e parâmetros aceites.

Autenticação

### Registo

Ação → Regista um novo utilizador.

Postman → 1. Authentication > register()

Método → POST

URL → api/v1/auth/register

Form Data →

1. name – String
2. user\_name – String
3. password – String, Mínimo 8 caracteres
4. password\_confirm – String, Mínimo 8 caracteres
5. email – Email válido

### Login

Ação → Autentica um utilizador

Postman → 1. Authentication > login()

Método → POST

URL → api/v1/auth/login

Form Data →

1. email – Email válido
2. password – String, Mínimo 8 caracteres

### Logout

Ação → Efetua o logout do utilizador. Elimina o token obtido.

Postman → 1. Authentication > logout()

Método → POST

URL → api/v1/auth/logout

### Logout de todos os dispositivos

Ação → Efetua o logout e elimina todos os tokens atribuídos a esse utilizador.

Postman → 1. Authentication > logout\_all\_devices()

Método → POST

URL → api/v1/auth/logout-all-devices

### Reposição de Password

Ação → Pede a reposição de password e obtém um token único para o efeito.

Postman → 1. Authentication > request\_token()

Método → POST

URL → api/v1/auth/request-reset-password-token

Form Data →

1. email – Email válido

### Criar Nova Password

Ação → Permite criar uma nova password utilizando o token previamente obtido.

Postman → 1. Authentication > set\_new\_password()

Método → POST

URL → api/v1/auth/set-new-password

Form Data →

1. email – Email válido
2. token – Token obtido na chamada anterior (request\_token())
3. password – String, Mínimo 8 caracteres
4. passsword\_confirmation – String, Mínimo 8 caracteres

Categorias

### Listar Todas as Categorias

Ação → Listagem de todas as categorias.

Postman → 2. Categories > índex()

Método → GET

URL → api/v1/categories

### Listar uma Categoria

Ação → Lista os dados de uma categoria em especifico.

Postman → 2. Categories > show()

Método → GET

URL → api/v1/categories/{category\_id}

### Registar Categoria

Ação → Regista uma nova categoria na base de dados.

Postman → 2. Categories > store()

Método → POST

URL → api/v1/categories

Form Data →

1. name – String

### Atualizar Dados de uma Categoria

Ação → Atualiza os dados de uma categoria já registada

Postman → 2. Categories > update()

Método → POST

URL → api/v1/categories/{category\_id}

Form Data →

1. \_method = PUT
2. name – String

### Eliminar uma Categoria

Ação → Elimina uma categoria da base de dados.

Postman → 2. Categories > destroy()

Método → POST

URL → api/v1/categories/{category\_id}

Form Data →

1. \_method = DELETE

Posts

### Listar Todos os Posts

Ação → Lista todos os posts da base de dados com paginação.

Postman → 3. Posts > index()

Método → GET

URL → api/v1/posts

### Obter Dados de um Post

Ação → Lista os dados de um post, contendo também as informações do autor, cas categorias e dos comentários associados.

Postman → 3. Posts > show()

Método → GET

URL → api/v1/posts/{post\_id}

### Regista um Novo Post na Base de Dados

Ação → Regista um novo post na base de dados.

Postman → 3. Posts > store()

Método → POST

URL → api/v1/posts

Form Data →

1. title – String | Mínimo 10 Caracteres
2. body – String | Mínimo 20 caracteres
3. image – MIMES: jpg, png, gif | Tamanho máximo 3MB | Opcional
4. categories[] – Array de Id’s.

### Atualiza um Post existente na Base de Dados

Ação → Atualiza um post existente na base de dados.

Postman → 3. Posts > update()

Método → POST

URL → api/v1/posts/{post\_id}

Form Data →

1. \_method = PUT
2. title – String | Mínimo 10 Caracteres
3. body – String | Mínimo 20 caracteres
4. image – MIMES: jpg, png, gif | Tamanho máximo 3MB | Opcional
5. categories[] – Array de Id’s.

### Elimina um post

Ação → Elimina um post

Postman → 3. Posts > destroy()

Método → POST

URL → api/v1/posts/{post\_id}

Form Data →

1. \_method = DELETE

Comentários

### Registar um comentário

Ação → Regista um novo comentário a um post/comentário.

Postman → 4. Comments > store()

Método → POST

URL → api/v1/comments/store

Form Data →

1. post\_id – Campo numérico
2. comment – String | Mínimo 10 caracteres
3. parent\_id – Opcional (a preencher quando o comentário é feito a um comentário)

### Atualiza um Comentário

Ação → Atualiza um comentário

Postman → 4. Comments > update()

Método → POST

URL → api/v1/comments/{comment\_id}

Form Data →

1. \_method = PUT
2. comment – String | Mínimo 10 caracteres

### Elimina um Comentário

Ação → Elimina um comentário

Postman → 4. Comments > destroy()

Método → POST

URL → api/v1/comments/{comment\_id}

Form Data →

1. \_method = DELETE

Pesquisa

### Por Keyword

Ação → Pesquisa posts pela keyword fornecida. A pesquisa é feita pelo título.

Postman → 5. Search Posts > By Title Keyword

Método → GET

URL → api/v1/search/posts/by-keyword/{keyword}

### Por Categoria

Ação → Pesquisa posts pela categoria.

Postman → 5. Search Posts > By Category ID

Método → GET

URL → api/v1/search/posts/by-category/{category\_id}

Logs

Ação → Obtem os logs de um utilizar em especifico.

Postman → 6. Logs > Search Logs by user\_id

Método → GET

URL → api/v1/logs/search/{user\_id}

Subscritores

### Listar Todos

Ação → Lista todos os subscritores.

Postman → 7. Subscribers > index()

Método → GET

URL → api/v1/subscribers

### Subscrever

Ação → Regista nova subscrição.

Postman → 7. Subscribers > subscribe()

Método → POST

URL → api/v1/subscribers

Form Data →

1. name – String
2. email – Email válido

### Pedir Token Para Cancelamento da Subscrição

Ação → Pede um token único para efetuar o cancelamento da subscrição.

Postman → 7. Subscribers > request\_unsuscribe\_token()

Método → POST

URL → api/v1/subscribers/request-unsubscribe-token

Form Data →

1. email – Email válido

### Cancelar Subscrição

Ação → Cancela a subscrição de um utilizador utilizando o token previamente obtido.

Postman → 7. Subscribers > unsubscribeByToken()

Método → POST

URL → api/v1/subscribers/unsubscribe-by-token

Form Data →

1. email – Email válido
2. token – String

Administração – Gerir Posts

### Listar Todos os Posts

Ação → Lista todos os posts.

Postman → 8. Administration > Manage Posts > posts()

Método → GET

URL → api/v1/administration/manage-posts/posts

### Listar Posts a Aguardar Aprovação

Ação → Lista todos os posts que aguardam aprovação

Postman → 8. Administration > Manage Posts > pending\_posts()

Método → GET

URL → api/v1/administration/manage-posts/pending-posts

### Aprovar Post Pendente

Ação → Aprova um post que está pendente de aprovação.

Postman → 8. Administration > Manage Posts > approve\_posts()

Método → POST

URL → api/v1/administration/manage-posts/pending-posts/aprove

Form Data →

1. post\_id – Numérico

### Eliminar Post Pendente

Ação → Elimina um post que está pendente de aprovação, ou seja, não aprovado.

Postman → 8. Administration > Manage Posts > delete\_pending\_posts()

Método → POST

URL → api/v1/administration/manage-posts/pending-posts/delete

Form Data →

1. post\_id – Numérico

Administração – Gerir Comentários

### Listar Comentários Pendentes

Ação → Lista todos os comentários que aguardam aprovação.

Postman → 8. Administration > Manage Comments > comments()

Método → GET

URL → api/v1/administration/manage-comments/pending-comments

### Aprovar Comentário Pendente

Ação → Aprova um comentário que está pendente de aprovação.

Postman → 8. Administration > Manage Comments > approve()

Método → POST

URL → api/v1/administration/manage-comments/pending-comments/approve

Form Data →

1. comment\_id – Numérico

### Elimina Comentário Pendente

Ação → Elimina comentário pendente.

Postman → 8. Administration > Manage Comments > delete()

Método → POST

URL → api/v1/administration/manage-comments/pending-comments/delete

Form Data →

1. comment\_id – Numérico

Administração – Gerir Utilizadores

### Listagem de Utilizadores

Ação → Lista todos os utilizadores registados.

Postman → 8. Administration > Manage Users > list\_users()

Método → GET

URL → api/v1/administration/manage-users/list-users

### Listagem dos Posts de um Utilizador

Ação → Lista os posts de um utilizador

Postman → 8. Administration > Manage Users > list\_user\_posts()

Método → GET

URL → api/v1/administration/manage-users/list-user-posts

Query String Params →

1. user\_id – Numérico
2. limit – Numérico – Limite de resultados
3. order\_by – Enum [ asc / desc ]

### Listagem dos Comentários de um Utilizador

Ação → Lista os comentários de um utilizador

Postman → 8. Administration > Manage Users > list\_user\_comments()

Método → GET

URL → api/v1/administration/manage-users/list-user-comments

Query String Params →

1. user\_id – Numérico
2. limit – Numérico – Limite de resultados
3. order\_by – Enum [ asc / desc ]

### Atualiza o Grupo de um Utilizador

Ação → Atualiza o grupo de um utilizador

Postman → 8. Administration > Manage Users > update\_user\_group()

Método → POST

URL → api/v1/administration/manage-users/list-users

Form Data →

1. user\_id – Numérico
2. group\_id – Numérico

Administração – Grupos de Utilizadores

### Listagem

Ação → Lista todos os possíveis grupos de utilizador.

Postman → 8. Administration > Manage Groups > index()

Método → GET

URL → api/v1/administration/manage-user-groups/list-groups

Utilizadores

### Ver Perfil de Utilizador

Ação → Lista os dados do perfil de um utilizador.

Postman → 9. Users > índex()

Método → GET

URL → api/v1/users/{user\_id}

### Atualizar Email

Ação → Atualiza o email de um utilizador

Postman → 9. Users > update\_email()

Método → POST

URL → api/v1/users/update-email

Form Data →

1. user\_id – Numérico
2. email – Email válido | Único na tabela de utilizadores

### Atualizar Password

Ação → Atualiza a password de um utilizador

Postman → 9. Users > update\_password()

Método → POST

URL → api/v1/users/update-password

Form Data →

1. user\_id – Numérico
2. password – String | Mínimo 8 caracteres
3. password\_confirmation – String | Mínimo 8 caracteres

### Atualizar Avatar

Ação → Atualiza o avatar de um utilizador

Postman → 9. Users > update\_avatar()

Método → POST

URL → api/v1/users/update-avatar

Form Data →

1. user\_id – Numérico
2. avatar – File | MIMES: jpg, png, gif | Tamanho máximo 2MB

Ação →

Postman →

Método →

URL →

Form Data →

1. Application Programming Interface [↑](#footnote-ref-1)
2. Diagrama de Entidade-Relação [↑](#footnote-ref-2)
3. Relação 1:N – 1 para Muitos – Ex.: 1 post pode ter associados N comentários e 1 comentário só pode ter 1 post associado [↑](#footnote-ref-3)
4. Relação N:M – Muitos para Muitos – Ex.: 1 post pode ter N categorias associadas e 1 categoria pode ter N posts associados [↑](#footnote-ref-4)
5. <https://laravel.com/docs/7.x/passport> [↑](#footnote-ref-5)