

Como o CRUD Acontecerá para User, Post e Tag

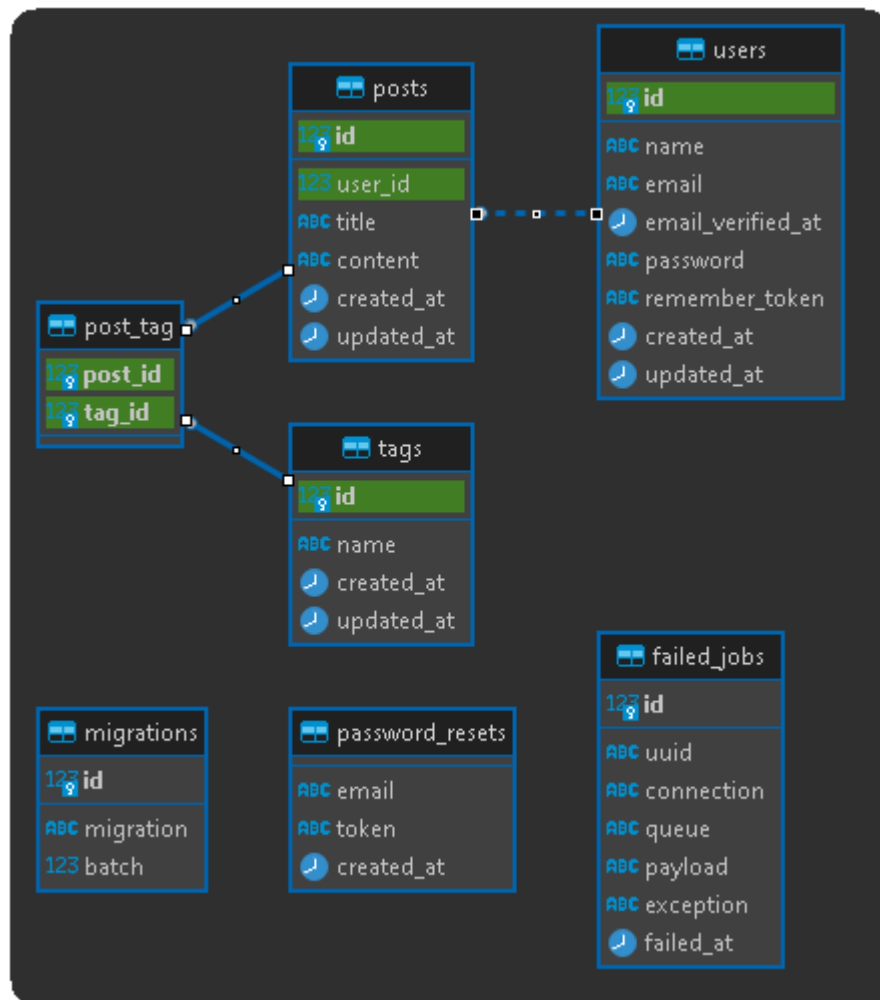
Mapa:

Método HTTP	Rota	Função Controller	Ação
GET	/api/users	index	Listar todos os usuários
GET	/api/users/{id}	show	Exibir os detalhes de um usuário específico
POST	/api/users	store	Criar um novo usuário
PUT	/api/users/{id}	update	Atualizar os dados de um usuário específico
DELETE	/api/users/{id}	destroy	Excluir um usuário específico
GET	/api/posts	index	Listar todas as postagens
GET	/api/posts/{id}	show	Exibir os detalhes de uma postagem específica
POST	/api/posts	store	Criar uma nova postagem
PUT	/api/posts/{id}	update	Atualizar os dados de uma postagem
DELETE	/api/posts/{id}	destroy	Excluir uma postagem específica
GET	/api/tags	index	Listar todas as tags
GET	/api/tags/{id}	show	Exibir os detalhes de uma tag específica
POST	/api/tags	store	Criar uma nova tag
PUT	/api/tags/{id}	update	Atualizar uma tag existente
DELETE	/api/tags/{id}	destroy	Excluir uma tag específica

Com o [apiResource](#), as rotas correspondentes às operações de CRUD para **User**, **Post** e **Tag** são automaticamente geradas e protegidas pela autenticação **JWT** (exceto no end point **User** para **login** e **cadastro**) mais detalhe JWT verifique última página.

É aconselhável realizar a inserção do **CRUD** nessa ordem (**User**, **Post** e **Tag**) , pela entidade e relacionamento do banco, obedecendo a existência de dados no relacionamento de chave estrangeira.

Diagrama entidade relacionamento



Users

id (PK) - Chave primária	id (PK) - Chave primária
name - Nome do usuário	title - Título da postagem
email - Email do usuário	content - Conteúdo da postagem
password - Senha criptografada	user_id (FK) - referenciando o <u>users(id)</u>
created_at - Data de criação	created_at - Data de criação
updated_at - Data de atualização	updated_at - Data de atualização

Posts

Tags

id (PK) - Chave primária	post_id (FK) referenciando o <u>posts(id)</u>
name - Nome da tag	tag_id (FK) referenciando o <u>tags(id)</u>
created_at - Data de criação	(PK post_id e tag_id) - Chave primária composta
updated_at - Data de atualização	

Post_tag

Regras do negócio API

User (Usuário)

- **GET - /api/users – Listar todos os usuários**

- A rota executa a função index do UserController, retornando uma lista de todos os usuários no banco de dados.

- URL: `http://localhost:8000/api/users`

- **GET - /api/users/{id} – Exibir um usuário específico**

- A rota executa a função show do UserController, retornando os detalhes de um usuário específico, identificado pelo seu id.

- URL: `http://localhost:8000/api/users/1`

- **POST - /api/users – Criar um novo usuário**

- A rota executa a função store do UserController, criando um novo usuário com os dados fornecidos no corpo da requisição (payload).

- URL: `http://localhost:8000/api/users`

- Cabeçalhos (Headers): Content-Type: application/json

- Ex.: Schema (Dados de entrada) - JSON:

- `{ "name": "João Silva", "email": "joao@example.com", "password": "senha123" }`

- **PUT - /api/users/{id} – Atualizar dados de um usuário**

- A rota executa a função update do UserController, atualizando as informações de um usuário existente, identificando-o pelo id.

- URL: `http://localhost:8000/api/users/1`

- Cabeçalhos (Headers): Content-Type: application/json

- URL: `http://localhost:8000/api/users/1`

- Ex.: Schema (Dados de entrada) - JSON:

- `{ "name": "João Silva2", "email": "joao@update.com", "password": "senhaupdate" }`

- **DELETE - /api/users/{id} – Excluir um usuário**

- A rota executa a função destroy do UserController, excluindo um usuário específico pelo id.

- URL: `http://localhost:8000/api/users/1`

Tag (Tag)

- **GET /api/tags – Listar todas as tags**

- A rota executa a função index do TagController, retornando uma lista de todas as tags no banco de dados.

- **GET /api/tags/{id} – Exibir uma tag específica**

- A rota executa a função show do TagController, retornando os detalhes de uma tag específica identificada pelo seu id.

- URL: `http://localhost:8000/api/tags/1`

- **POST /api/tags – Criar uma nova tag**

- A rota executa a função store do TagController, criando uma nova tag com os dados fornecidos, como o nome da tag.

- Ex.: Schema (Dados de entrada) - JSON:
- `{ "name": "Laravel" }`

- **PUT /api/tags/{id} – Atualizar uma tag existente**

- A rota executa a função update do TagController, atualizando uma tag existente identificada pelo id.

- URL: `http://localhost:8000/api/tags/1`

- Ex.: Schema (Dados de entrada) - JSON:
- `{ "name": "Laravel_updated" }`

- **DELETE /api/tags/{id} – Excluir uma tag**

- A rota executa a função destroy do TagController, excluindo uma tag específica identificada pelo id.

- URL: `http://localhost:8000/api/tags/1`

Post (Postagem)

- **GET - /api/posts – Listar todas as postagens**

- A rota executa a função index do PostController, retornando uma lista de todas as postagens no banco, incluindo as tags associadas (usando o with('tags') para carregar as tags relacionadas).

- **GET - /api/posts/{id} – Exibir uma postagem específica**

- A rota executa a função show do PostController, retornando os detalhes de uma postagem específica, incluindo suas tags.

- URL: `http://localhost:8000/api/posts/1`

- **POST /api/posts – Criar uma nova postagem**

- A rota executa a função store do PostController, criando uma nova postagem. O corpo da requisição deverá incluir os dados necessários, como title, content, user_id e, opcionalmente, as tags.

- Ex.: Schema (Dados de entrada) - JSON:
- { "title": "Meu primeiro post", "content": "Este é o conteúdo da postagem.", "user_id": 1, "tags": "1, 2" }

- **PUT /api/posts/{id} – Atualizar uma postagem existente**

- A rota executa a função update do PostController, atualizando uma postagem existente identificada pelo id. A requisição pode incluir novas tags associadas à postagem.

- Ex.: Schema (Dados de entrada) - JSON:
- { "title": "Meu primeiro post update", "content": "Este é o conteúdo da postagem update.", "user_id": 1, "tags": "3, 1" }

- **DELETE /api/posts/{id} – Excluir uma postagem**

- A rota executa a função destroy do PostController, excluindo a postagem identificada pelo id.

- URL: `http://localhost:8000/api/posts/1`

Regras do negócio JWT

Observação: As rotas **User**, **Post** e **Tag** são protegidas com autenticação **JWT**, (exceto em **User** o **login** e o **cadastro**) para testes com da API segue orientações para acesso aos end points.

1. Acesse a rota : <http://127.0.0.1:8000/api/auth/login?email=melqui@acpgroup.com&password=12345678>
(GET)

2. Estando User **cadastrado**, e for correto os dados de entrada, será gerado um token. Em posse desse token insira no Authorization, Bearer token para acessar demais end point.

[illegible]

Adicional: Login Web
(Acessando a raiz do projeto)



Somente
AUTORIZADO(s)

Login

Senha

Acessar

[Não possui cadastro? Clique!](#)