Laravel API Login and Registration Documentation.

User end points

Registration End-Point: POST http://localhost:8000/api/register

The registration end-point expects a body of

```
{
   "name":"Example name",
   "email": "test@example",
   "password":"123456A_",
   "password_confirmation":"123456A_"
}
```

And the response will be a body of

```
{
   "user": {
       "name": "name Example",
       "email": "test@example",
       "updated_at": "2023-12-23T00:08:24.000000Z",
       "created_at": "2023-12-23T00:08:24.000000Z",
       "id": 3
   },
   "token": "7|XijEm121Xb1XoXJWdrGDOh1RTqJYgfH72IHV9W6Rd037b85c"
}
```

Registration rules:

- The Email must be unique.
- The password field must match the password confirmation field.
- The password field must be at least 8 characters.
- The password field must contain at least one symbol.
- The password field must contain at least one number.
- The password field must contain at least one letter.

Note: the password is not returned to the user, but if it were, the response would look like such:

```
{
   "user": {
       "name": "Example name",
       "email": "test@example",
       "password": "$2y$12$q90ks89slGfVcTygeU.PReuMEhrt8IW9BBX.wxH9A9cre1duNfliW",
       "updated_at": "2023-12-23T00:16:41.000000Z",
       "created_at": "2023-12-23T00:16:41.000000Z",
       "id": 4
   },
   "token": "8|skJT81AGkQeCc7mmr81qJtnt22Qpk08jbIpXQasD72a69b99"
}
```

Login End-Point: POST http://localhost:8000/api/login

The registration end-point expects a body of

```
{
   "email": "test@example",
   "password":"123456A_"
}
```

And the response will be a body of

```
{
{
   "user": {
       "id": 1,
       "name": "example",
       "email": "test@example",
       "created_at": "2023-12-22T23:50:07.000000Z",
       "updated_at": "2023-12-22T23:50:07.000000Z"
   },
   "token": "10|GeyX2XQgNSGM9HXp7Utmti9XaNi3nQi5se1PDGDRbf48076d"
}
```

Note that each time a user logs in the old token is revoked and a new token is generated, so the old token can no longer be used for authorization, make sure to update the authorization Bearer Token each time the user relogs (more on that further into the document).

Logout End-Point: POST http://localhost:8000/api/logout

Requires Authorization Token.

If the user is authenticated, you should expect a response of.

```
{
   "success": true,
   "message": "logged out successfully"
}
```

User End-Point: GET http://localhost:8000/api/user

Requires Authorization Token.

If the User is authenticated, you can expect a body of

```
{
   "id": 6,
   "name": "example",
   "email": "test@example",
   "created_at": "2023-12-23T01:43:15.000000Z",
   "updated_at": "2023-12-23T01:43:15.000000Z"
}
```

posts end points.

To further showcase the authentication and authorization I've added a Model called posts, each user can have multiple posts, further explanation is present with each end point.

Create a post end-point: **POST** http://localhost:8000/api/post.

Requires Authorization Token.

The post end-point expects a body of

```
{
   "title":"post"
}
```

Note that the title is a unique attribute, so multiple entries with the same title will result in a duplicate entry error.

A successful post request will result in a response of


```
{
    "title": "post1",
    "user_id": 1,
    "updated_at": "2023-12-23T02:12:09.000000Z",
    "created_at": "2023-12-23T02:12:09.000000Z",
    "id": 9
}
```


Get all posts end point: **GET** http://localhost:8000/api/post

The get end-point requires no authentication and no body, meaning that anyone is able to read any post.

The response looks like this, with each post containing the info of its user, typically for a redirect to the user's profile.


```
[{
        "id": 1,
        "title": "Autem veniam voluptatem voluptatem voluptatum et error.",
        "user_id": 1,
        "created_at": "2023-12-23T02:05:40.000000Z",
        "updated_at": "2023-12-23T02:05:40.000000Z",
        "user": {
            "id": 1,
            "name": "User A",
            "email": "usera@example.com",
            "created_at": "2023-12-23T02:05:40.000000Z",
            "updated_at": "2023-12-23T02:05:40.000000Z"
        }},{
        "id": 4,
        "title": "Ipsam repudiandae accusantium omnis iure quis ut delectus.",
        "user_id": 2,
        "created_at": "2023-12-23T02:05:41.000000Z",
        "updated_at": "2023-12-23T02:05:41.000000Z",
        "user": {
            "id": 2,
            "name": "User B",
            "email": "userb@example.com",
            "created_at": "2023-12-23T02:05:41.000000Z",
            "updated_at": "2023-12-23T02:05:41.000000Z" }]
```

Get one post end-point by ID: **GET** http://localhost:8000/api/post/{id}

This end-point requires no authentication and no body, meaning that anyone is able to get any post.

Replace the {id} parameter with an existing post id.

The response looks like this, with the post containing the info of its user, typically for a redirect to the user's profile.


```
[
    {
        "id": 1,
        "title": "Autem veniam voluptatem voluptatem voluptatum et error.",
        "user_id": 1,
        "created_at": "2023-12-23T02:05:40.000000Z",
        "updated_at": "2023-12-23T02:05:40.000000Z",
        "user": {
            "id": 1,
            "name": "User A",
            "email": "usera@example.com",
            "created_at": "2023-12-23T02:05:40.000000Z",
            "updated_at": "2023-12-23T02:05:40.000000Z"
        }
    }
]
```


Get posts per user end-point by ID: **GET** http://localhost:8000/api/user/posts/{id}

This end-point requires no authentication and no body, meaning that anyone is able to get any post.

Replace the {id} parameter with an existing user id.

The response looks like this, the info of the user and all his posts, typically for a redirect to the user's profile.


```
{
    "user": {
        "id": 1,
        "name": "User A",
        "email": "usera@example.com",
        "created_at": "2023-12-23T02:05:40.000000Z",
        "updated_at": "2023-12-23T02:05:40.000000Z"
    },
    "post": {
        "0": {
            "id": 1,
            "title": "Autem veniam voluptatem voluptatem voluptatum et error.",
            "user_id": 1,
            "created_at": "2023-12-23T02:05:40.000000Z",
            "updated_at": "2023-12-23T02:05:40.000000Z"
        },
        "1": {
            "id": 2,
            "title": "Labore voluptatum qui eum quia sint labore.",
            "user_id": 1,
            "created_at": "2023-12-23T02:05:40.000000Z",
            "updated_at": "2023-12-23T02:05:40.000000Z"      }}
```

Delete a post end-point: **DELETE** http://localhost:8000/api/post/{id}

Requires Authorization Token.

Only the user that created a post is able to delete the post, replace the id parameter with an existing post id related to the authorized user. This end-point doesn't require a body.

A successful deletion will return a response of


```
{
   "message": "post with id = (3) deleted successfully",
   "id": "3"
}
```


Update a post end-point: **PATCH** http://localhost:8000/api/post/{id}

Requires Authorization Token.

Only the user that created a post is able to update the post, replace the id parameter with an existing post id related to the authorized user.

It requires a body of


```
{
   "title":"updated post"
}
```


Keep in mind that you can't change the title of a post to an already existing name as it is unique

A successful update will return the updated post, typically to swap for the old post without having to send a separate get request. The response will look like such:


```
{
   "id": 1,
   "title": "updated post",
   "user_id": 1,
   "created_at": "2023-12-23T02:05:40.000000Z",
   "updated_at": "2023-12-23T02:35:35.000000Z"
}
```

Authorization Token.

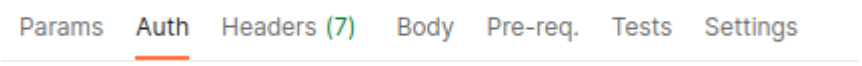
Sanctum was used for this project's token-based authentication.

Sanctum is ideal for mobile apps and SPAs.

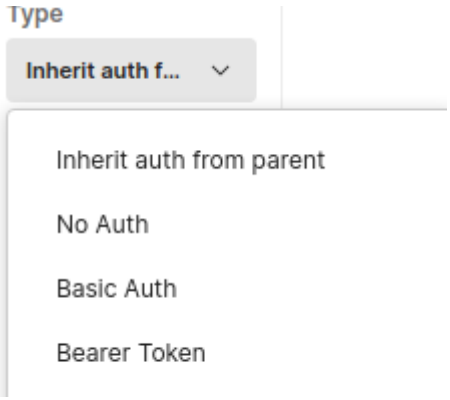
A new token is generated each time the user logs in, so make sure you grab the new token to use it.

To send the authorization token (assuming postman is used)

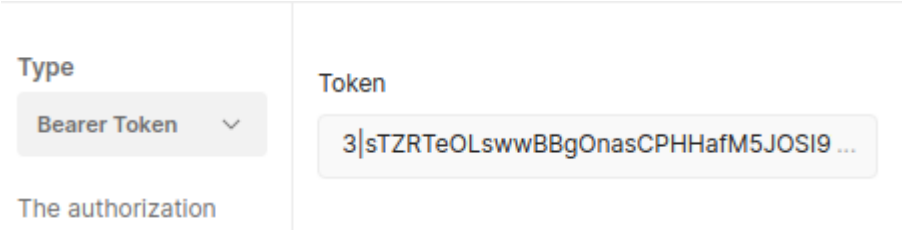
- Go to the Auth tab



- In Type pick bearer Token .



- In the Token field paste the token received from the login or registration body.



User Tests.

Run php artisan test to run the following tests

`test_users_can_authenticate_using_the_login_screen()`

`test_users_can_not_authenticate_with_invalid_password()`

`Test_new_users_can_register()`