User Remember Token

Ruta: (post) api/v1/user/remembertoken (middlewares que observan esta ruta [`AuthenticationChannel`]), la cual apunta al controlador UserController, método RememberToken.

Primero que todo, nos encontramos una clase proporcionada por Laravel para crear una instancia que compruebe los parámetro que incluyamos que tengan con los requisitos que definamos, esto quiere que decir que en esta api deberá llegar un parámetro remember_token y decimos que es requerido para de esta manera si no va incluido lo capture como un error(img 1).

```php
$validator = Validator::make($request->all(), [
    'remember_token' => 'required'
]);

if ($validator->fails()) {
    $error = 1;
    $message_validator = $message_validator->merge($validator->errors());
}

if ($error == 1) {
    $array['error'] = 400;
    $array['error_description'] = 'The fields are not the required format.';
    $array['error_inputs'][0] = $message_validator;
```

img 1

Comprobado que esa remeber_token en la consulta, donde dentro del modelo user tenemos la función userRememberToken que permite acceder a tokos los remember tokens asociado a un usuario(img 2 y 3).

```php
else {

  $remember_token = $request->input('remember_token');

  $user = User::whereHas('userRemeberToken', function
```

img 2

```php
public function userRemeberToken()
{
    return $this->hasMany('App\Models\UserRememberToken', 'user_id');
}
public function documentType()
{
```

img 3

Ahora realizamos la extracción del usuario donde el campo token de esta tabla de user remember tokens sea igual al remember_token pasado como parámetro, además de que dicho usuario tendrá que tener el email verificado,, sin el login bloqueado y el que no hay sido bloqueado o que la fecha del bloqueo sea anterior al día actual (img 4).

```php
$user = User::whereHas('userRemeberToken', function ($query) use ($remember_token) {
    $query->where('token', $remember_token);
})
    ->where('verified_email', '=', 1)
    ->where('bloqued_login', '=', 1)
    ->where(function ($query) {
        $query->whereNull('bloqued_to')->orWhere('bloqued_to', '<', Carbon::now());
    })->first();
```

img 4

Una vez superada dicha validación/query ya tendremos en una variable al usuario con la data de todos sus atributos de la tabla lp_users que tenemos almacenada en nuestra base de datos.

Siguiente, si el usuario existe, extraemos la compañía ligada al usuario (img 5), la cual deberá estar activa con una fecha de comienzo anterior al día actual, y una fecha de fin posterior al día actual.

```php
if ($user) {
    $company = Company::where('id', $user->company_id)
        ->where('lp_companies.date_start', '<=', Carbon::now()->format('Y-m-d'))
        ->where('lp_companies.date_end', '>=', Carbon::now()->format('Y-m-d'))
        ->first();
}
```

img 5

Por último, si el usuario y la compañía existen, creamos con las funcionalidades que nos proporciona eloquent una variables (img 6) llamadas:

token, remember_token, time_expired, expired_at, refresh_token, refresh_expired_at.

time_expired, usa la configuración seteada en el campo nombre(token_validate) y aplica el valor que dicho campo contenga y un parse int para evitar problemas (se volvía loco eloquent y lo tomaba como que era string).

```php
if ($user && $company) {
    $token = base64_encode(Str::random(200));
    $remember_token = base64_encode(Str::random(200));

    $time_expired = intval(Setting::where('name', 'token_validate')->first()->value);
    $expired_at = Carbon::now()->addMinutes($time_expired);


    $refresh_token = base64_encode(Str::random(200));
    $refresh_expired_at = Carbon::now()->addMinutes(2 * $time_expired);
```

img 6

Ahora ya nos vamos a la tabla lp_user_remember_tokens y cogemos el que era igual al remember_token enviado como param y hacemos un update al registro que coincida en el cual el campo token valda el remember_token recién creado y la ip, método nativo que se puede realizar a la request ->ip() (img 7).

```php
UserRememberToken::where('token', '=', $request->input('remember_token'))->update([
    'token' => $remember_token,
    'ip' => $request->ip(),
]);
```

img 7

Creamos nuestro registro de en user tokens (img 8) con los valores generados anteriormente (img 6)

```php
UserToken::create([
    'user_id' => $user->id,
    'token' => $token,
    'expired_at' => $expired_at,
    'refresh_token' => $refresh_token,
    'refresh_expired_at' => $refresh_expired_at,
]);
```

img 8

Cambiamos ademas el valor de last login del user (img 9) y mandamos la respuesta al front ademas de con los datos generados con la info de si tiene el perfil completado (img 10).

```php
User::where('id', $user->id)->update([
    'last_login' => Carbon::now()
]);
```

img 9


```
{
    "error": 200,
    "data": [
        {
            "user_id": 8,
            "token": "Bearer
                MDhOVjJpWWRQVEhiNXd4Nk42d3REMnplODZZVEFzRTVqNWlsdHZzZWxNSXJvMThxd3BSdWVrQ1hMbHFTY3Y3Z0x3YVlrZFQxaVd6RTg2OTB2TmFra3BkMUlvaDdyUEh4NzFOUDBUN0kxZG1xQWZHRTB
                VUlFGMXZrQlBqOEpLblNyQ0dWWFFpY1NUSzdxUmpmWXM3R1hTQjdxYUdsRjBKeXZBT2lpTG9qazBicnhUckdONnlwSUpibUs4dTBYbjh4NHZlNFFSVVQ=",
            "remember_token":
                "dUVqSlJJVXNQQ2JmT1I5T1Nkc0tBa0EzNXBzRUlNMDFVeXpCT0V4dk1zTEhUWllETFRYWWU5Vml3elBrVmJKQ0ZFQWhsSnoyRExXMVpZZUtsZ2pVbElHcmFNckxXTjBTYWhpV2czZm9YdExxWcWRDRl
                Fsb0FWRTRtQ3Bzc04ycFg3VzdMRW1pRDQzbjM5RjhtS2ljaTByb3NpSUVjbTJaazhsOTI4ZjVVa2JuQzNkc06UUGljRFBIVDZPNlo4SjdxMHFpd3d3ZEo=",
            "refresh_token":
                "T21BUFFIRjJXM2NGS0cydElWOW51eTFYbUVKYWo4YU9HOHI3ZGFRaUdYUlVpTm9RMUxGS1I2endlczlrRXVCN0Vla05QZmRPZkRCeWxrejhISFZqMTVSUHdJbHJDb2N6MDJQQlhuT01OcnBmWWJXSE
                w0UktKOEpDc1RkcHEzT1h0Z05FQkRiOEVLc3NmeGNqRnhCWkZEallEaGduZGpzMk1hUm1BWmlta3lIRTYyRFNqcEo2Um1TT3Nwb0ZVVVFoaWZzMHhRbkQ=",
            "profile_completed": 1
        }
    ]
}
```


img 10