ProjectManager | MERN

Lógica de Autenticación de usuarios

1. HELPERS

`errorResponse()`

- Función que devuelve la respuesta de la API en caso de error.

```
module.exports = (res, error, method) => {
    console.log(error);
    return res.status(error.status || 500).json({
        ok : false,
        msg : error.message || `Upss, hubo un error en ${method}`
    })
}
```

`generateTokenRandom()`

- Función que devuelve un número aleatorio
- *Math* es un objeto incorporado que tiene propiedades y métodos para constantes y funciones matemáticas
- El método `Math.random()` devuelve un número de coma flotante pseudo-aleatorio, comprendido en el rango de 0 a menor que 1 (es decir, incluido el 0 pero no el 1)
- El método `toString()` tiene un param llamado `radix` que puedes pasar en números entre 2 - 36 , que convertirá los números generados en los caracteres de base que se encuentran entre el número dado. La raíz también se conoce como base y es para representar valores numéricos. A partir del 11 comenzará a introducir letras.
- `Base 32` es un sistema de numeración posicional que usa 32 como base. Es similar al sistema de numeración posicional Base64 pero usando 32 como base en lugar de 64. Para representar los números usa las 26 letras mayúsculas A-Z y los seis dígitos 2-7.
- El método `substring(2)` devuelve la parte de la string desde el índice de inicio hasta y excluyendo el índice final, o hasta el final de la cadena si no se proporciona un índice final.
- Enlaces de interés:
    - Método random()
    - Método toString()
    - Método substring()
    - Base 32

```
module.exports = () => {
    const random = Math.random().toString(32).substring(2);
    const date = Date.now().toString(32);
```

```
    return random + date
}
```

`generateJWT()`

- *JSON Web Token* es un objeto de JSON (notación de objeto de JavaScript), una herramienta de estándar abierto cuyo objetivo es establecer una transmisión de información entre dos o más campos. A partir de estos, se puede propagar información de forma segura y efectiva, que, además, es verificada, pues se firma de forma virtual. Este conjunto de información toma la referencia de web token, bajo el estándar abierto de JSON.
- Enlaces de interés
  - ¿Qué es JWT?
  - Web oficial

```
const jwt = require('jsonwebtoken');

module.exports = (payload) => jwt.sign(payload, process.env.JWT_SECRET,{
    expiresIn : '1h'
})
```

2. Controlador

- Registrar usuario | `register()`

```
    try {

            const {name,email,password} = req.body;

            if([name,email,password].includes("")){
                throw createError(400,"Todos los campos son obligatorios");
            };

            let user = await User.findOne({
                email
            });

            if(user){
                throw createError(400,"El email ya se encuentra registrado");
            }

            user = new User(req.body);
            user.token = generateTokenRandom();

            const userStore = await user.save();

            //TODO: enviar el email de confirmación con el TOKEN

            return res.status(201).json({
```

```
                ok : true,
                msg :'Usuario Registrado',
                data : userStore
            })

        } catch (error) {
            return errorResponse(res,error, "REGISTER")
        }
```

- Autenticar usuario | `login()`

```
const {email,password} = req.body;

        try {

            if([email,password].includes("")){
                throw createError(400,"Todos los campos son obligatorios");
            };

            let user = await User.findOne({
                email
            });

            if(!user){
                throw createError(403,"Credenciales inválidas | EMAIL");
            };

            if(!user.checked){
                throw createError(403,"Tu cuenta no ha sido confirmada");
            };

            if(!await user.checkedPassword(password)){
                throw createError(403,"Credenciales inválidas | PASSWORD");
            }

            return res.status(200).json({
                ok : true,
                msg :'Usuario Logueado',
                user : {
                    nombre : user.name,
                    email : user.email,
                    token : generateJWT({
                        id: user._id
                    })
                }
            })
        } catch (error) {
            return errorResponse(res,error, "LOGIN")
        }
```

- Completar la registración | `checked()`

```
const {token} = req.query;
      try {

          if(!token){
              throw createError(400,"Token inexistente");
          };

          const user = await User.findOne({
              token
          });

          if(!user){
              throw createError(400,"Token inválido");
          };

          user.checked = true;
          user.token = "";

          await user.save()

          return res.status(201).json({
              ok : true,
              msg :'Registro completado exitosamente'
          })
      } catch (error) {
          return errorResponse(res,error, "CHECKED")

      }
```

- Enviar link para reestablecer la contraseña | `sendToken()`

```
const {email} = req.body;

      try {

          let user = await User.findOne({
              email
          });

          if(!user) throw createError(400,"Email incorrecto");

          user.token = generateTokenRandom();
          await user.save();

          //TODO: Enviar email para reestablecer la contraseña

          return res.status(200).json({
              ok : true,
```

```
                msg :'Se ha enviado un email con las intrucciones'
            })
        } catch (error) {
            return errorResponse(res,error, "SEND-TOKEN")
        }
```