1. var express = require('express');
2. var bcrypt = require('bcryptjs'); // https://github.com/dcodeIO/bcrypt.js
3. var jwt = require('jsonwebtoken'); // https://github.com/auth0/node-jsonwebtoken
5. var SEED = require('../config/config').SEED;
6. var CADUCIDAD\_TOKEN = require('../config/config').CADUCIDAD\_TOKEN;
8. var app = express();
10. var Usuario = require('../models/usuario');
12. const { OAuth2Client } = require('google-auth-library');
13. const GOOGLE\_CLIENT\_ID = require('../config/config').GOOGLE\_CLIENT\_ID;
14. // const GOOGLE\_SECRET = require('../config/config').GOOGLE\_SECRET;
15. const client = new OAuth2Client(GOOGLE\_CLIENT\_ID);
17. //=============================
18. // Autenticación de Google
19. //=============================
20. app.post('/google', async (req, res) => {
21. let token = req.body.token;
22. const ticket = await client.verifyIdToken({
23. idToken: token,
24. audience: GOOGLE\_CLIENT\_ID,
25. }).catch(e => {
26. return res.status(403).json({
27. ok: false,
28. mensaje: 'Token no válido',
29. err: e
30. });
31. })
33. const googleUser = ticket.getPayload();
35. /\* res.status(200).json({
36. ok: true,
37. ticket: googleUser,
38. email: googleUser.email
39. }) \*/
41. Usuario.findOne({ email: googleUser.email }, (err, usuarioDB) => {
42. if (err) {
43. return res.status(500).json({
44. ok: false,
45. err
46. });
47. };
48. if (usuarioDB) {
49. if (usuarioDB.google === false) {
50. return res.status(400).json({
51. ok: false,
52. err: {
53. message: 'Debe de usar su autenticación normal'
54. }
55. });
56. } else {
57. console.log('No existe usuario');
59. let token = jwt.sign({ usuario: usuarioDB }, SEED, { expiresIn: CADUCIDAD\_TOKEN });
60. return res.json({
61. ok: true,
62. usuario: usuarioDB,
63. token: token,
64. id: usuarioDB.\_id
65. });
66. }
67. } else {
68. // Si el usuario no existe en nuestra base de datos
69. let usuario = new Usuario();
70. usuario.nombre = googleUser.name;
71. usuario.email = googleUser.email;
72. usuario.img = googleUser.picture;
73. usuario.google = true;
74. usuario.password = ':)';
75. usuario.save((err, usuarioDB) => {
76. if (err) {
77. return res.status(500).json({
78. ok: false,
79. err
80. });
81. };
82. let token = jwt.sign({ usuario: usuarioDB }, SEED, { expiresIn: CADUCIDAD\_TOKEN });
83. //var token = jwt.sign({ usuario: usuarioBD }, SEED, { expiresIn: 14400 }); // 4 horas
85. return res.json({
86. ok: true,
87. usuario: usuarioDB,
88. token: token,
89. id: usuarioDB.\_id
90. });
91. });
92. }
93. });
94. });

97. //=============================
98. // Autenticación normal
99. //=============================
100. app.post('/', (req, res) => {
102. var body = req.body;
104. // Verifico que existe el usuario con ese correo
105. Usuario.findOne({ email: body.email }, (err, usuarioBD) => {
107. if (err) {
108. return res.status(500).json({
109. ok: false,
110. mensaje: 'Error al buscar usuario',
111. errors: err
112. });
113. }
115. if (!usuarioBD) {
116. return res.status(400).json({
117. ok: false,
118. mensaje: 'Credenciales incorrectas - email',
119. errors: err
120. });
121. }
123. // Si la contraseña no coincide entre el password enviado con el password de BD
124. if (!bcrypt.compareSync(body.password, usuarioBD.password)) {
125. return res.status(400).json({
126. ok: false,
127. mensaje: 'Credenciales incorrectas - password',
128. errors: err
129. });
130. }
132. // Crear un token (en este punto el correo y el password ya son correctos)
133. usuarioBD.password = ':D';
135. var token = jwt.sign({ usuario: usuarioBD }, SEED, { expiresIn: CADUCIDAD\_TOKEN });
137. res.status(200).json({
138. ok: true,
139. usuario: usuarioBD,
140. token: token,
141. id: usuarioBD.\_id
142. });
144. });
146. });

149. // Lo exporto para ser usado en app.js
150. module.exports = app;

    var express = require('express');

    var bcrypt = require('bcryptjs'); // https://github.com/dcodeIO/bcrypt.js

    var jwt = require('jsonwebtoken'); // https://github.com/auth0/node-jsonwebtoken

    var SEED = require('../config/config').SEED;

    var CADUCIDAD\_TOKEN = require('../config/config').CADUCIDAD\_TOKEN;

    var app = express();

    var Usuario = require('../models/usuario');

    const { OAuth2Client } = require('google-auth-library');

    const GOOGLE\_CLIENT\_ID = require('../config/config').GOOGLE\_CLIENT\_ID;

    // const GOOGLE\_SECRET = require('../config/config').GOOGLE\_SECRET;

    const client = new OAuth2Client(GOOGLE\_CLIENT\_ID);

    //=============================

    // Autenticación de Google

    //=============================

    app.post('/google', async (req, res) => {

     let token = req.body.token;

     const ticket = await client.verifyIdToken({

     idToken: token,

     audience: GOOGLE\_CLIENT\_ID,

     }).catch(e => {

     return res.status(403).json({

     ok: false,

     mensaje: 'Token no válido',

     err: e

     });

     })

     const googleUser = ticket.getPayload();

     /\* res.status(200).json({

     ok: true,

     ticket: googleUser,

     email: googleUser.email

     }) \*/

     Usuario.findOne({ email: googleUser.email }, (err, usuarioDB) => {

     if (err) {

     return res.status(500).json({

     ok: false,

     err

     });

     };

     if (usuarioDB) {

     if (usuarioDB.google === false) {

     return res.status(400).json({

     ok: false,

     err: {

     message: 'Debe de usar su autenticación normal'

     }

     });

     } else {

     console.log('No existe usuario');

     let token = jwt.sign({ usuario: usuarioDB }, SEED, { expiresIn: CADUCIDAD\_TOKEN });

     return res.json({

ok: true,

     usuario: usuarioDB,

     token: token,

     id: usuarioDB.\_id

     });

     }

     } else {

     // Si el usuario no existe en nuestra base de datos

     let usuario = new Usuario();

     usuario.nombre = googleUser.name;

     usuario.email = googleUser.email;

     usuario.img = googleUser.picture;

     usuario.google = true;

     usuario.password = ':)';

     usuario.save((err, usuarioDB) => {

     if (err) {

     return res.status(500).json({

     ok: false,

     err

     });

     };

     let token = jwt.sign({ usuario: usuarioDB }, SEED, { expiresIn: CADUCIDAD\_TOKEN });

     //var token = jwt.sign({ usuario: usuarioBD }, SEED, { expiresIn: 14400 }); // 4 horas

     return res.json({

     ok: true,

     usuario: usuarioDB,

     token: token,

     id: usuarioDB.\_id

     });

     });

     }

     });

    });

    //=============================

    // Autenticación normal

    //=============================

    app.post('/', (req, res) => {

     var body = req.body;

     // Verifico que existe el usuario con ese correo

     Usuario.findOne({ email: body.email }, (err, usuarioBD) => {

     if (err) {

     return res.status(500).json({

     ok: false,

     mensaje: 'Error al buscar usuario',

     errors: err

     });

     }

     if (!usuarioBD) {

     return res.status(400).json({

     ok: false,

     mensaje: 'Credenciales incorrectas - email',

     errors: err

     });

     }

     // Si la contraseña no coincide entre el password enviado con el password de BD

     if (!bcrypt.compareSync(body.password, usuarioBD.password)) {

     return res.status(400).json({

     ok: false,

     mensaje: 'Credenciales incorrectas - password',

     errors: err

     });

     }

     // Crear un token (en este punto el correo y el password ya son correctos)

     usuarioBD.password = ':D';

     var token = jwt.sign({ usuario: usuarioBD }, SEED, { expiresIn: CADUCIDAD\_TOKEN });

     res.status(200).json({

     ok: true,

     usuario: usuarioBD,

     token: token,

     id: usuarioBD.\_id

     });

     });

    });

    // Lo exporto para ser usado en app.js

    module.exports = app;

1. CODIGO DE PREGUNTAS Y RESPUESTAS
2. app.post('/google', (req, res) => {
3. var token = req.body.token || '';
4. const client = new OAuth2Client(GOOGLE\_CLIENT\_ID, GOOGLE\_CLIENT\_SECRET);
5. const ticket = client.verifyIdToken({
6. idToken: token,
7. audience: GOOGLE\_CLIENT\_ID
8. });
9. ticket.then(data => {
10. res.status(200).json({
11. ok: true,
12. ticket: data.payload,
13. userid: data.payload.sub
14. });
15. Usuario.findOne({ email: ticket.email }, (err, usuario) => {
16. if (err) {
17. return res.status(500).json({
18. ok: false,
19. mensaje: 'Error al buscar el usuario - login',
20. errors: err
21. });
22. }
23. if (usuario) {
24. if (usuario.google === false) {
25. return res.status(400).json({
26. ok: true,
27. mensaje: 'Debe Utilizar el método de Autenticación normal para poder acceder...'
28. });
29. } else {
30. // Crear un Token
31. usuario.password = ':)';
32. var token = jwt.sign({ usuario: usuario }, SEED, { expiresIn: 14400 });
33. res.status(200).json({
34. ok: true,
35. usuario: usuario,
36. token: token,
37. id: usuario.\_id,
38. // body: body
39. });
40. }
41. // Si el Usuario no existe en la base de datos
42. } else {
43. console.log(ticket);
44. var payload = ticket.payload;
45. var nuevoUsuario = new Usuario();
46. nuevoUsuario.nombre = ticket.name;
47. nuevoUsuario.email = ticket.email;
48. nuevoUsuario.password = ':)';
49. nuevoUsuario.img = data.picture;
50. nuevoUsuario.google = true;
51. console.log(nuevoUsuario);
52. /\*nuevoUsuario.save((err, usuarioBD) => {
53. if (err) {
54. return res.status(500).json({
55. ok: false,
56. mensaje: 'Error al crear el usuario - google',
57. errors: err
58. });
59. }
60. // Crear un Token
61. var token = jwt.sign({ usuario: usuarioBD }, SEED, { expiresIn: 14400 });
63. res.status(200).json({
64. ok: true,
65. usuario: usuarioBD,
66. token: token,
67. id: usuarioBD.\_id,
68. // body: body
69. });
70. });\*/
71. }
72. });
73. }).catch(err => {
74. if (err) {
75. return res.status(400).json({
76. ok: false,
77. mensaje: 'Token no válido',
78. errors: err
79. });
80. }
81. })
82. });