



---

# ESCUELA SUPERIOR DE CÓMPUTO

Investigacion sobre Ciencia de Datos

Alumno:

**Michael Adolfo Huerta Ramírez / Jesedh Guerrero Paisano**

Profesor:

**Ulises Velez Saldaña**

28 Marzo del 2023

## Índice

1. Medidas de prevencion de seguridad a realizar	2
--	---

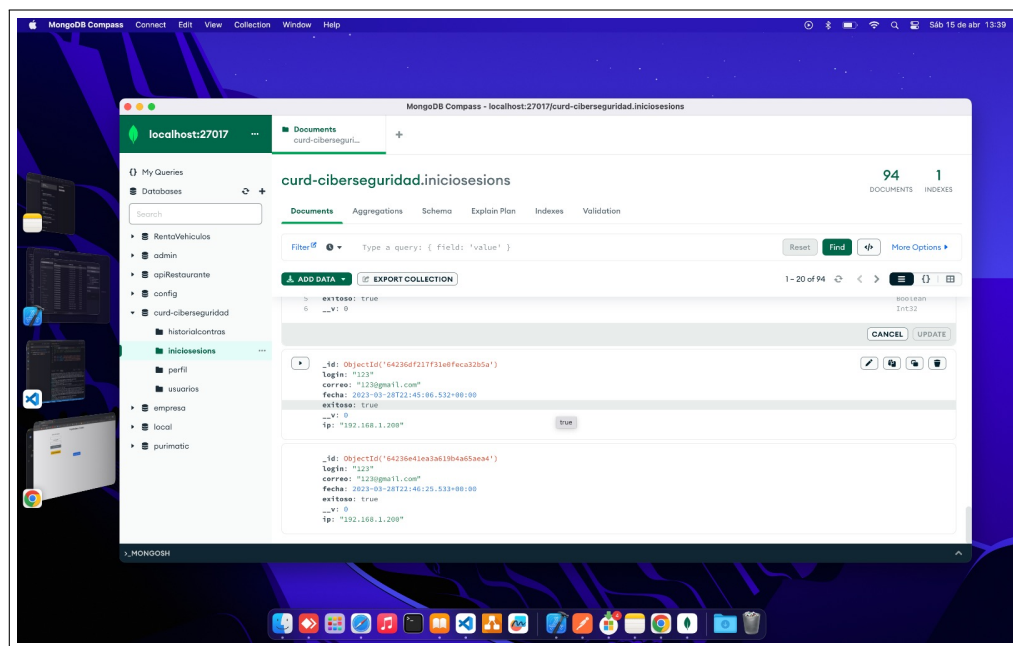
## 1. Medidas de prevencion de seguridad a realizar

- Registro de intentos Registrar los intentos con y sin exito ocurridos.
- Terminar las sesiones inactivas tras un periodo definido de inactividad (5 minutos).
- Restringir los tiempos de conexion reduciendo la ventana de oportunidad de los usuarios no autorizados (20 minutos)
- Debe tener al menos 8 caracteeres, una minúscula, una mayuscula y un caracter especial
- Forzar los cambios regulares de contraseñas y bajo peticion. Las contraseñas (cada 30 dias).
- Mantener un registro de las contraseñas usadas anteriormente y evitar su reutilizacion
- Almacenar los ficheros de contraseñas de manera separada de los datos del sistema de aplicación
- Almacenar y transmitir las contraseñas en forma protegida

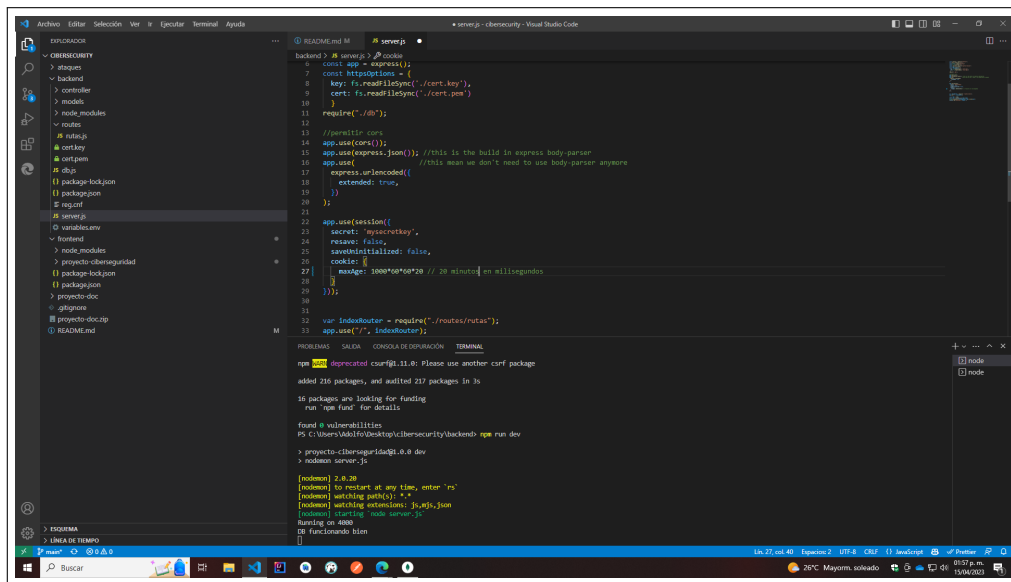
1.- Registramos en nuestra base de datos de MongoDB los siguientes datos:

1.1 login , correo , fecha , exitoso , ip

Donde login es el nombre del usuario, correo el correo del usuario , fecha es la fecha donde se realizo el intento , exitoso un booleano de exitoso o fracaso , ip donde accedio el usuario



2.- Creamos una sesion mediante node.js para crear un temporizador de 20 minutos hasta que caduco.



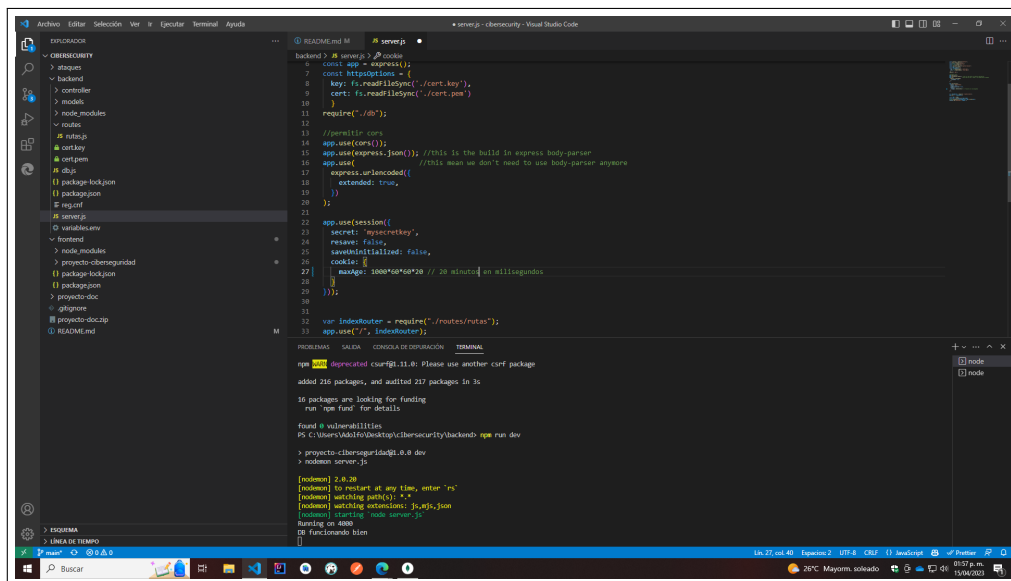
```
back&gt; M server.js & node
const app = express();
const httpsOptions = {
  key: fs.readFileSync('./cert.key'),
  cert: fs.readFileSync('./cert.pem')
};
require('./db');
//parallel cors
app.use(cors());
app.use(express.json()); //this is the build in express body-parser
app.use(
  express.urlencoded({
    extended: true,
  })
);
app.use(session({
  secret: 'supersecret',
  resave: false,
  saveinitialized: false,
  cookie: {
    maxAge: 1000*60*20 // 20 minutos en milisegundos
  }
}));
var indexRouter = require('./routes/rutas');
app.use('/', indexRouter);
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
npm WARN deprecated curl@11.0.0: Please use another curl package
added 216 packages, and audited 217 packages in 3s
16 packages are looking for funding
run npm fund for details
found 0 vulnerabilities
PS C:\Users\Adri\Documents\ciberseguridad\back&gt; node server.js
> proyecto-ciberseguridad@1.0.0 dev
> nodemon server.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: *.js,json
[nodemon] starting 'node server.js'
Running on 4000
DE funcionando bien
```

3.- Gracias a nuestro diseño de la BD , guardamos la ip ->en cuanto se realice tres intentos fallidos se revisa la ip de ingreso para bloquear su acceso



```
back&gt; M server.js & node
const app = express();
const httpsOptions = {
  key: fs.readFileSync('./cert.key'),
  cert: fs.readFileSync('./cert.pem')
};
require('./db');
//parallel cors
app.use(cors());
app.use(express.json()); //this is the build in express body-parser
app.use(
  express.urlencoded({
    extended: true,
  })
);
app.use(session({
  secret: 'supersecret',
  resave: false,
  saveinitialized: false,
  cookie: {
    maxAge: 1000*60*20 // 20 minutos en milisegundos
  }
}));
var indexRouter = require('./routes/rutas');
app.use('/', indexRouter);
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
npm WARN deprecated curl@11.0.0: Please use another curl package
added 216 packages, and audited 217 packages in 3s
16 packages are looking for funding
run npm fund for details
found 0 vulnerabilities
PS C:\Users\Adri\Documents\ciberseguridad\back&gt; node server.js
> proyecto-ciberseguridad@1.0.0 dev
> nodemon server.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: *.js,json
[nodemon] starting 'node server.js'
Running on 4000
DE funcionando bien
```

## 4.- Validamos mediante regex las contraseñas ingresadas

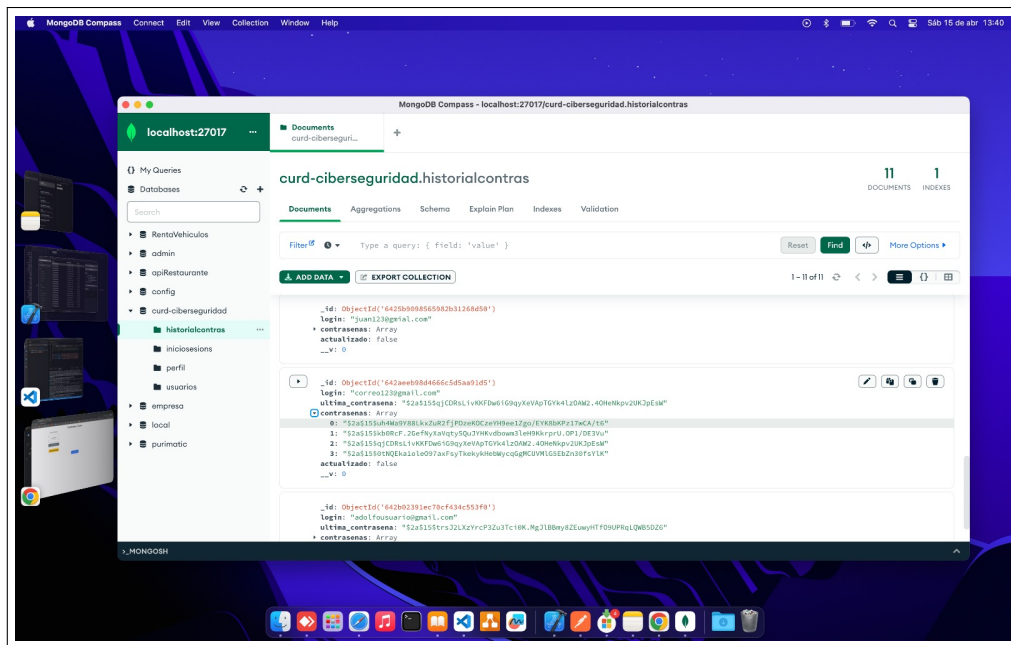
```
33 }
34 };
35
36 exports.registrar = async (req, res) => {
37   console.log(req.body);
38   if (req.body) {
39     let { nombre, perfil, correo, password, confirmar } = req.body;
40     console.log(nombre);
41     if (password !== confirmar) {
42       res.status(400).json({ mensaje: "Las contraseñas no coinciden" });
43       return;
44     }
45
46     // Expresión regular para validar la contraseña
47     const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*!@#$%^&*()_+~=-[\]{};':"\\|,.<\/>?]).+$/;
48
49     // Validar la contraseña
50     const isValidPassword = passwordRegex.test(password);
51
52     if (!isValidPassword) {
53       res.status(400).json({
54         mensaje:
55           "La contraseña debe tener al menos una letra mayúscula, una minúscula y un carácter especial",
56       });
57       return;
58     }
59
60     // cifrar la contraseña
61     const salt = await bcryptjs.genSalt(15);
62     password = await bcryptjs.hash(password, salt);
63
64     const correo_existente = await Usuario.find({ correo: correo });
65     console.log(correo_existente);
66     if (correo_existente.length > 0) {
67       res.status(230).json({ mensaje: "Ya existe un usuario con este correo" });
```

5.- Creamos una opcion para cambiar la contraseña y para periodicamente reiniciar las contraseñas.

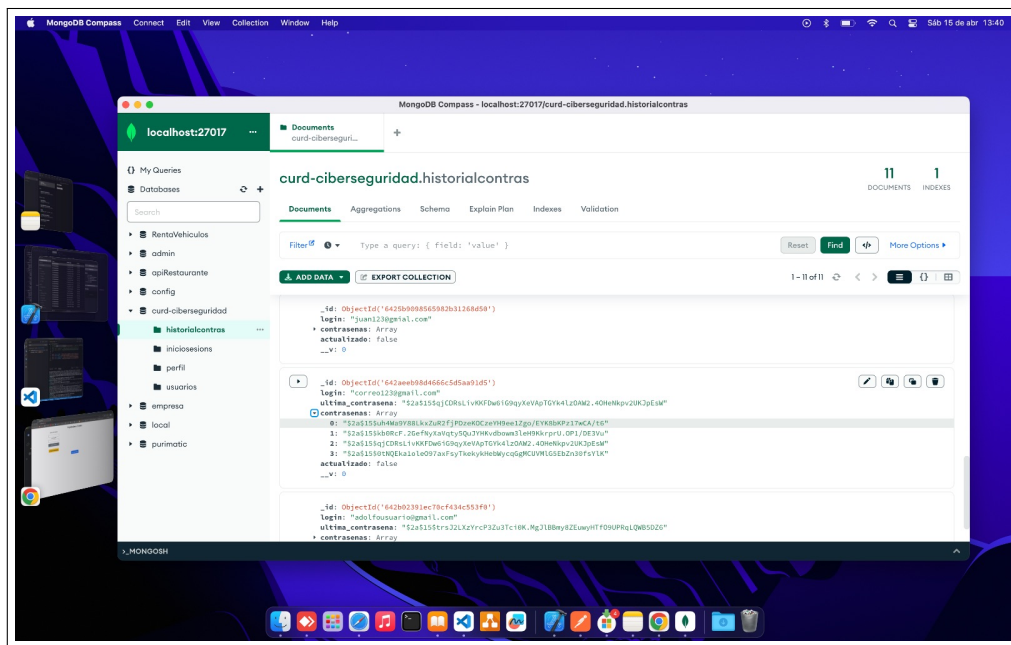


```
169 };
170
171 exports.recuperarContrasena = async (req, res) => {
172   console.log(req.body);
173   let { correo } = req.body;
174
175   //generar una contraseña aleatoria
176   const caracteres =
177     "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
178   let longitud = 8;
179   let contrasena = "";
180   for (let i = 0; i < longitud; i++) {
181     contrasena += caracteres.charAt(
182       Math.floor(Math.random() * caracteres.length)
183     );
184   }
185
186   const resultado = await Usuario.find({ correo: correo });
187
188   if (resultado) {
189     if (resultado.length > 0) {
190       let mailOptions = {
191         from: "nathanael91@ethereal.email",
192         to: correo,
193         subject: "Recuperación de contraseña",
194         text:
195           "Hola, solicitaste recuperar tu contraseña, tu nueva contraseña es: " +
196           contrasena +
197           ", te recomendamos cambiarla lo antes posible",
198       };
199
200       transporter.sendMail(mailOptions, function (error, info) {
201         if (error) {
202           console.log(error);
203         } else {
204           console.log("Correo electrónico enviado: " + info.response);
205           res.json({ mensaje: "Correo enviado correctamente a " + correo });
206         }
207       });
208     }
209     //actualizar la contraseña
210     const salt = await bcryptjs.genSalt(15);
211     contrasena = await bcryptjs.hash(contrasena, salt);
212     const resultado2 = await Usuario.updateOne(
213       { correo: correo },
214       { password: contrasena },
215       { new: true }
216     );
217   }
218 }
```

6.- Para Mantener un registro de las contraseñas usadas anteriormente y evitar su reutilizacion creamos este diccionario




7.- Para Almacenar los ficheros de contraseñas de manera separada de los datos del sistema de aplicación creamos otra tabla



8.- Almacenar y transmitir las contraseñas en forma protegida , para eso creamos un servicio que requiere un certificado SSL

```

backend >  cert.key
1  -----BEGIN PRIVATE KEY-----
2  MIIIEvQIBADANBgkqhkiG9w0BAQEFAASCBAcwggSjAgEAAoIBAQQDH1hEW+ypVERUz
3  Z9dl92bLIrKYoeT7UCkYDqjhOnWv1Uwe8pFBLHCXn/onG93Vezb6UvnsnCs7hGLB
4  uKvVvVADwxOKB2wANf42NvT6ZFxoJm5LHECI0F9DJ3+LSeI719p1Pt3an/9/rgeK
5  55CE1sA9m8tJMthfKwLPC/KaxILy7mvJt+gdkccFhwjXDIzZ0u1RnC+8wMrYrZhF
6  PLJ1m7R7UnYZyrQnIrCngvTga5crX/OwCxXkCIzLwbz76GpNQdfSa6+vRk5wr/F
7  mixbj9w3kiDI53GpQd8En24CH8NqE8H9elp2nWstB1/CrgWaY7ZFpSBqnsRbTM/0
8  E3xZz6z1AgMBAECggEBAJY1DMF2ZzprzcDHA/wr+0MjQJVD02cnuMvRcRViQMUj
9  9oWo1JA1hDgkUpaE9riZlhiQuDUMqcRh3Q157weKLsTVI+GzdHwWwH/XGV1vChjOy
10 Eas5gDSWU5S5ZouAi1/9dX1aM65uCeJ0Rpa++IT4IVZwtfh72LfdRnEuDUocMwNUy
11 Fla//PkNjNax3juqF1gt3mg9PBCT91qZSswDEz2D1Fltk04Fp7eBNz5BqMMz8NON
12 sI3LR7fjAU++i/rLAbaroTdgub6sP63srKzOBjzcXM4v199oq4wD6+31p/sy4VDN
13 xPPudKyAHsd1846xf0Zosxlp3/si3+VHXBnJRFUsIECgYEA8jUfjb5qgD5G0We2
14 6P0byvL2ycIiU02tXkMfRPrXX/dKIuSvCVW0nmTxpqhpz4rT7S05SDLHKocKufnb
15 /OLUM2FSY1kzR1P2pCtVJo89Z6wVTR6ee2Jk22iBB7dMLBmuyFrPq73pDifYy868
16 Se9uDJaKepnwMeUXcYXd7rcDvFUCgYEA0zdCPPZ2A45qgZqkmKabuHvykxfsdLEz
17 A2rCSOLBiPOwcleRv+Y1CE+W4C7jh8zb0PomnDLrihNewGU5rgJ2DDnyl7S+Ts8
18 dwRmg4/A36NKx/+SfJhcyc/BowiJaL3sA/aJM02CqwbML81Ii8JIdKu4MH8jeN3n
19 BDdmjowozIECgYAmzxZIOvuLou7Toeev+eC7qySZ/W5M1MSztu0ax0qfqNsLLkNi
20 UI5JVv1Vb+TMaX09oRy+EcZr5qN0WzHBmoLP3Mc7lP1fc3ewpHTXYdo/81XSbpaM
21 RJ70fTc3iko9Qrw7xD1EVoCyUPYu6ehHnIKwp6p2rJXNGN0hxoZiCpChpQKBgBlU
22 EBj1b8N/aZA/LivMFvJpkq5G+SkJMtffDotU/eZHz5H+wXCkt4lWmeSpPFSPs0hN
23 eXBDxHkPBjNfG08LlMqjGriu5Ffu3axcD7sJGxT2bPpM0JE2aDIQtY9KNUxqULU
24 ZqFCah/+QbAdUCDKXiv5J65RILzWx2Q/DIJczuBhAoGAJ23jn/6/sK4Ybf54C8t0
25 PDPpIexvddTt+w+0170L1gNfDdpU2jiS3fZpeDVvFFkPyqCm49kLkGsJYDbZ1m40
26 /UFZB9kuivlIF1QZs6/Pwsm0s7QMTpmjTw1GDkwjqLznxbf2WMUtFn56w9t7JHqg
27 ByM0Lg8okiUTGB+HDUSdTXw=
28  -----END PRIVATE KEY-----
29

```