Proyecto

Camilo Esteban Zambrano Pereira
czambrano@unal.edu.co
MLDS 3: Big Data
Departamento de Ingeniería Eléctrica y Electrónica
Universidad Nacional de Colombia.
Bogotá. Colombia.

I. Problemática

El problema planteado es que se busca diseñar una cerradura electrónica comercial que se conecte a una base de datos con el fin de manejar que usuarios pueden abrir o cerrar la puerta, tener un historial de apertura de esta, abrirla de manera inalámbrica, entre otros. A partir de esto, es necesario diseñar una base de datos que pueda manejar la información para cada cerradura y para cada usuario.

II. Objetivos

II-A. Objetivo General

Diseñar e implementar una base de datos que maneje la información necesaria para administrar y controlar una cerradura electrónica.

II-B. Objetivos específicos

- Definir la estructura de la base de datos.
- Elegir una tecnología big data que se ajuste a las necesidades del proyecto.
- Diseñar un conjunto de datos con los que se pueda probar diferentes comandos para observar el funcionamiento de la base de datos.

III. Tecnología

Entre las tecnologías disponibles se opto por una base de datos no relacional ya que en la estructura planteada para los datos no se observan demasiadas relaciones entre los valores. Por otro lado, la tecnología elegida fue MongoDB por su velocidad de lectura y su fácil uso. Esto se debe a que para esta aplicación es importante la velocidad de lectura ya que un alto tiempo de búsqueda implica una mayor demora a la hora de abrir la cerradura. De esta manera se utilizo MongoDB Atlas, un servicio con una oferta gratuita para proyectos pequeños como es este caso. Por otro lado, no se utilizó el ofrecido por los profesores del curso ya que se busca continuar este proyecto y se tendrían algunas limitaciones.

IV. Obtención de datos

Dado que no se encontró un conjunto de datos pre diseñado para este proyecto, se realizo un dataset propio con ayuda de la librería faker de python, de esta manera, se crearon datos de nombre, nombre de usuario, fecha, hora, correo y contraseña, de forma que los datos se ajustan a las necesidades del sistema, el procedimiento de la construcción del dataset se realizará más adelante.

V. Diagrama de despliegue y descripción de los datos

La función de este proyecto es la de juntar una aplicación móvil con un microcontrolador que a su vez maneja sensores y periféricos que abren o cierran una cerradura electrónica:

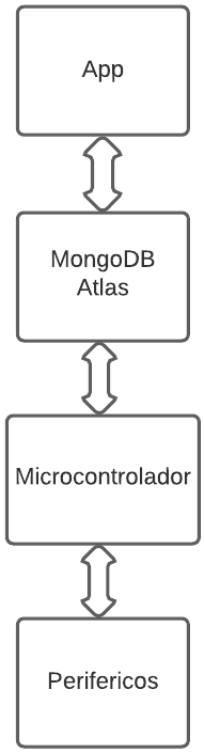

Figura 1: Diagrama de funcionamiento del sistema.

Los datos de se distribuyen en 2 colecciones de la base de datos, una con la información de las cerraduras y otra con la información de los usuarios de la siguiente forma:

Para la base de datos de la cerradura se utilizo la siguiente estructura:

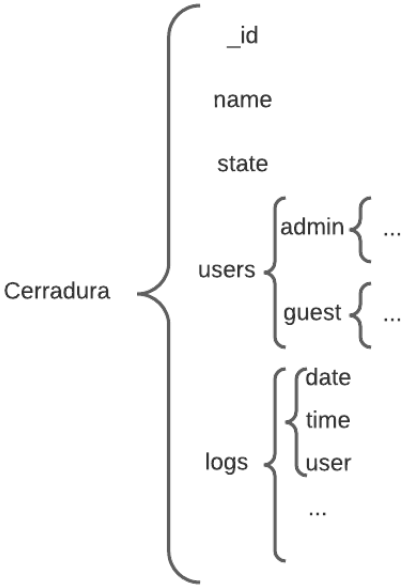

Figura 2: Estructura de la base de datos de las cerraduras.

Donde ”_idçorresponde al id unico de cada cerradura, ”name.es el nombre personalizado que da el usuario a la cerradura, ”state.es el estado de la cerradura, su valor es de 1 o 0 que corresponden a cerrado o abierto respectivamente, üsers”tiene 2 valores, .ªdminz ”guest”que corresponden a los usuarios que son administradores de la puerta (pueden editar parámetros como

el nombre o agregar y eliminar a otros usuarios) y los usuarios invitados que solamente pueden abrir la cerradura. El campo ”logs”hace referencia al historial de apertura de la puerta, donde en cada ”log”se almacena la hora, fecha y el usuario que abrió la puerta en ese momento.

Para la base de datos de los usuarios se maneja la siguiente estructura:

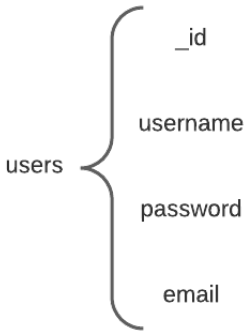

Figura 3: Estructura de la base de datos de los usuarios.

Nuevamente ”_id.es el valor unico de cada usuario, el üsername.es su nombre de usuario, .email.es el correo con el que se registró y ”password.es la contraseña.

VI. Instalación de MongoDB Atlas

Una vez registrado en MongoDB Atlas se escoge la versión gratuita que permite hasta 512 MB de almacenamiento con AWS de Amazon y se establece el nombre ”Biometrics”(este nombre se escogió dado que originalmente se planeaba guardar datos biométricos para el control de la cerradura, sin embargo, por políticas de seguridad de los dispositivos de Google y Apple esto era inviable).

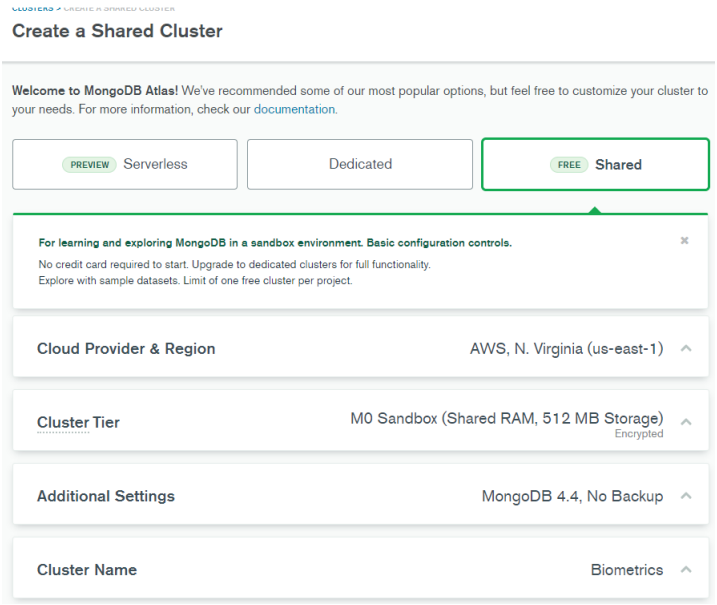

Figura 4: Creación del cluster.

Una vez que creado el cluster, se crean la bases de datos çerradura-electrónica”, se realiza la conexión con Python con el comando:

```
client = pymongo.MongoClient("mongodb+srv://czambrano:czambrano@biometrics.murkv.mongodb.net
/myFirstDatabase?retryWrites=true&w=majority")
db = client['cerradura-electronica']
```

VII. Carga de datos

Para la carga de datos se realizo el siguiente script en python, que genera nombres, username, contraseñas, fechas, horas, entre otros. de esta forma, se añaden 30.000 valores aleatorios a la colección de cerradura. Para cada una de ellas hay 6 usuarios, 3 administradores y 3 invitados, de esta forma, se generan 180.000 usuarios diferentes:

```
for i in range(30000):
  user1=list(fake.profile("username").values())[0]
  user2=list(fake.profile("username").values())[0]
  user3=list(fake.profile("username").values())[0]
  user4=list(fake.profile("username").values())[0]
  user5=list(fake.profile("username").values())[0]
  user6=list(fake.profile("username").values())[0]

  cerradura = {
     "_id": i,
     "name": list(fake.profile("username").values())[1],
     "users":{

        "admin":[
              user1,
              user2,
              user3
        ],
        "guest":[
              user4,
              user5,
              user6
        ]
     },
     "state":1,
     "logs":[
           {
              "date": fake.date(),
              "time": fake.time(),
              "user": user3
           },
           {
              "date": fake.date(),
              "time": fake.time(),
              "user": user1
           },
           {
              "date": fake.date(),
              "time": fake.time(),
              "user": user5
           }
     ]
   }
  users = [
    {
      "_id": 6*i+1,
      "username": user1,
      "password": fake.password(length=8),
      "email": fake.free_email()
    },
    {
      "_id": 6*i+2,
      "username": user2,
      "password": fake.password(length=8),
      "email": fake.free_email()
    },
    {
      "_id": 6*i+3,
      "username": user3,
      "password": fake.password(length=8),
      "email": fake.free_email()
    },
    {
```

```
    "_id": 6*i+4,
    "username": user4,
    "password": fake.password(length=8),
    "email": fake.free_email()
  },
  {
    "_id": 6*i+5,
    "username": user5,
    "password": fake.password(length=8),
    "email": fake.free_email()
  },
  {
    "_id": 6*i+6,
    "username": user6,
    "password": fake.password(length=8),
    "email": fake.free_email()
  },
 ]
 db.cerradura.insert_one(cerradura)
 db.users.insert_many(users)
```

Y añadiendo un usuario adicional con valores reales para tener conocimiento de al menos el valor para realizar búsquedas más adelante:

```
# Usuario real
cerradura = {
    "_id": 30001,
    "name": "esp32",
    "users":{

       "admin":[
             "CamSP"
       ],
       "guest":[]
    },
    "state":1,
    "logs":[
          {
             "date": "2021-12-18",
             "time": "09:25:10",
             "user": "CamSP"
          },
          {
             "date": "2021-12-10",
             "time": "09:25:10",
             "user": "CamSP"
          },
          {
             "date": "2021-11-1",
             "time": "09:25:10",
             "user": "CamSP"
          }
    ]
   }
users = {
   "_id": 180001,
   "username": "CamSP",
   "password": "123456",
   "email": "czambrano@unal.edu.co"
 }
db.cerradura.insert_one(cerradura)
db.users.insert_one(users)
```

De esta forma, se observa que efectivamente existen 30.001 valores en la colección de cerraduras y 180.001 en la de usuarios:

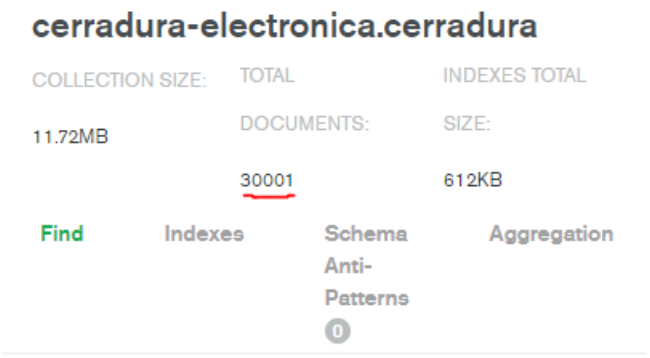

Figura 5: Cantidad de documentos en la colección de cerradura.

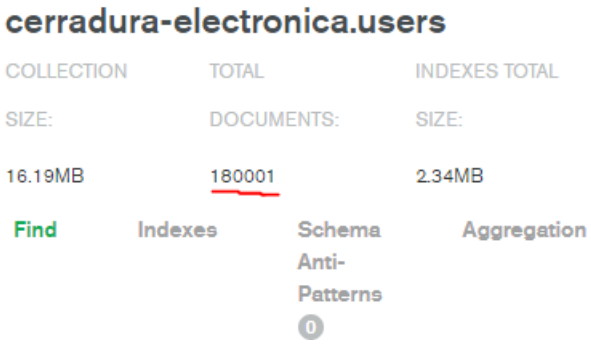

Figura 6: Cantidad de documentos en la colección de users.

VIII. Operaciones con MongoDB

Para la aplicación que se realizará en un futuro sobre este sistema es necesario hacer búsquedas en la base de datos, algunas de estas operaciones son:

Consulta de usuarios de una cerradura:

Figura 7: Consulta de usuarios de una cerradura.

Consulta de cerraduras de un usuario, en este caso este usuario solo tiene 1:

Consulta de cerraduras de un usuario

```
[73] Cerraduras = pd.DataFrame(db.cerradura.find({"users.admin": "CamSP"}, {"_id":1, "name":1}))
     Cerraduras
```

	_id	name
0	1001	esp32

Figura 8: Consulta de cerraduras de un usuario.

Cambiar el nombre de la cerradura, este nombre es elegido por el usuario:

Cambiar nombre de una cerradura

```
[127] list(db.cerradura.find({"_id": 456}, {"_id":0, "name":1}))

      [{'name': 'David Waters'}]
```

```
[128] db.cerradura.update_one({'_id' : 456}, {'$set' : {"name" : "Nuevo Nombre"}})
      list(db.cerradura.find({"_id": 456}, {"_id":0, "name":1}))

      [{'name': 'Nuevo Nombre'}]
```

Figura 9: Cambiar nombre de una cerradura.

Agregar usuarios a una cerradura:

Agregar usuarios a una cerradura

```
[89] db.cerradura.update_one({'_id' : 1001}, {'$set' : {"users.guest" : []}})
     list(db.cerradura.find({"_id": 1001}))

     [{'_id': 1001,
       'logs': [{'date': '2021-12-18', 'time': '09:25:10', 'user': 'CamSP'},
        {'date': '2021-12-10', 'time': '09:25:10', 'user': 'CamSP'},
        {'date': '2021-11-1', 'time': '09:25:10', 'user': 'CamSP'}],
       'name': 'esp32',
       'state': 1,
       'users': {'admin': ['CamSP'], 'guest': []}}]
```

```
[90] db.cerradura.update_one({'_id' : 1001}, {'$push' : {"users.guest" : "Nombre de Usuario"}})

     list(db.cerradura.find({"_id": 1001}))

     [{'_id': 1001,
       'logs': [{'date': '2021-12-18', 'time': '09:25:10', 'user': 'CamSP'},
        {'date': '2021-12-10', 'time': '09:25:10', 'user': 'CamSP'},
        {'date': '2021-11-1', 'time': '09:25:10', 'user': 'CamSP'}],
       'name': 'esp32',
       'state': 1,
       'users': {'admin': ['CamSP'], 'guest': ['Nombre de Usuario']}}]
```

Figura 10: Agregar usuarios a una cerradura.

Eliminar usuarios de una cerradura, se elimina el que fue agregado anteriormente:

Figura 11: Eliminar usuarios de una cerradura.

Convertir un usuario de invitado a un administrador:

Figura 12: Convertir usuario en administrador.

Convertir un usuario de administrador a un invitado:

Figura 13: Convertir usuario en invitado.

Abrir una cerradura, esta se abre por 3 segundos y luego se vuelve a cerrar:

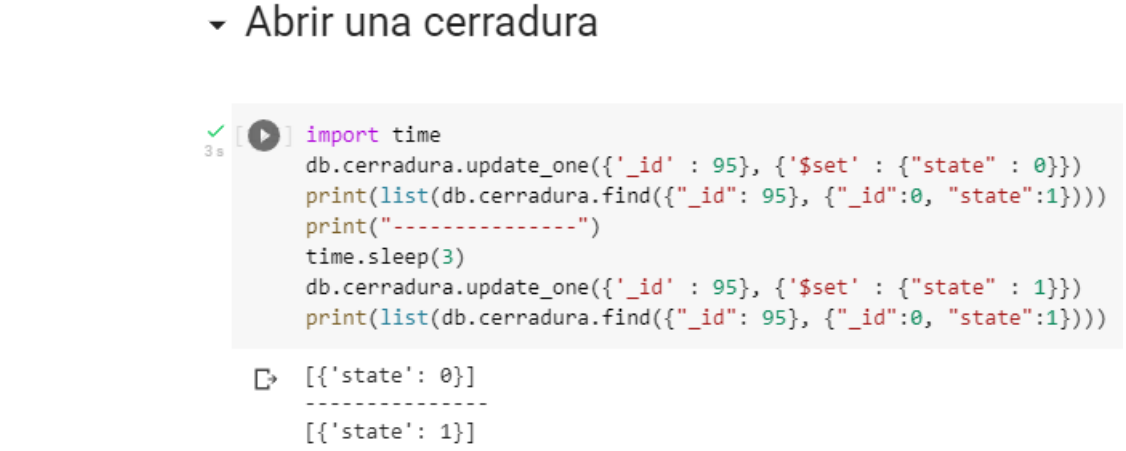

Figura 14: Abrir una cerradura.

Observar el historial de aperturas de la cerradura:

Figura 15: Observar el historial de aperturas de una cerradura.

Añadir nuevo log a la cerradura:

▾ Añadir nuevo log de la puerta

```
[155] from datetime import date
      from datetime import datetime

      db.cerradura.update_one({"_id": 256}, {"$push": {"logs": {"date": date.today().strftime("%Y-%m-%d"), "time": datetime.now().strftime("%H:%M:%S"), "user": "Usuario10"}}})

      pd.DataFrame.from_dict(db.cerradura.find({"_id":256}, {"_id":0, "date":"$logs.date","time":"$logs.time", "user":"$logs.user"})).apply(pd.Series.explode).reset_index(drop=True)
```

	date	time	user
0	2006-09-13	13:46:40	nicholasdiaz
1	2000-02-07	00:59:40	kramirez
2	1979-08-18	01:11:42	jamiesimpson
3	2021-12-19	03:41:35	Usuario10

Figura 16: Añadir nuevo log de la puerta.

Obtener información de un usuario:

▾ Obtener información de un usuario

```
[119] usuario = pd.DataFrame(db.users.find({"_id":1250}))
      usuario
```

	_id	username	password	email
0	1250	adrianwright	#9xpNn)N	saranichols@yahoo.com

Figura 17: Obtener la información de un usuario.
Cambiar el nombre de usuario:

▾ Editar nombre de usuario

```
[120] db.users.update_one({'_id' : 1250}, {'$set' : {"username" : "Nuevo_Nombre"}})
      pd.DataFrame(db.users.find({"_id":1250}))
```

	_id	username	password	email
0	1250	Nuevo_Nombre	#9xpNn)N	saranichols@yahoo.com

Figura 18: Editar nombre de usuario.

Cambiar el correo:

▾ Editar correo

```
db.users.update_one({'_id' : 1250}, {'$set' : {"email" : "test@test.com"}})
pd.DataFrame(db.users.find({"_id":1250}))
```

	_id	username	password	email
0	1250	Nuevo_Nombre	#9xpNn)N	test@test.com

Figura 19: Editar correo.

Cambiar la contraseña:

Editar contraseña

```
[122] db.users.update_one({'_id' : 1250}, {'$set' : {"password" : "123456789"}})
      pd.DataFrame(db.users.find({"_id":1250}))
```

	_id	username	password	email
0	1250	Nuevo_Nombre	123456789	test@test.com

Figura 20: Editar contraseña.

Registrar un nuevo usuario:

Registrar un nuevo usuario

```
[181] newUser = {
          "_id": db.users.find().count()+1,
          "username": "Nuevo_Usuario",
          "password": "contraseña",
          "email": "email@email.com"
      }
      db.users.insert_one(newUser)
```

```
/usr/local/lib/python3.7/dist-packages/ipykerr

<pymongo.results.InsertOneResult at 0x7fb55a45
```

```
[182] list(db.users.find({"_id":180002}))
```

```
[{'_id': 180002,
  'email': 'email@email.com',
  'password': 'contraseña',
  'username': 'Nuevo_Usuario'}]
```

Figura 21: Registrar un nuevo usuario.

Eliminar el usuario registrado:

Eliminar usuario

```
[183] db.users.delete_one({"_id":180002})
      list(db.users.find({"_id":180002}))
```

```
[]
```

Figura 22: Eliminar usuario.

IX. Conclusiones

- Se logró diseñar una base de datos en MongoDB que se ajusta a las necesidades del proyecto, con 2 colecciones y un total de 210.002 datos.
- Se generaron y cargaron los datos en la base de datos de forma correcta, y se evidencio la eficiencia de MongoDB al buscar un dato en especifico a una gran velocidad.

- Se implementaron funciones de lectura y escritura sobre la base de datos que servirán posteriormente en el montaje de la aplicación móvil y el sistema embebido de la cerradura.