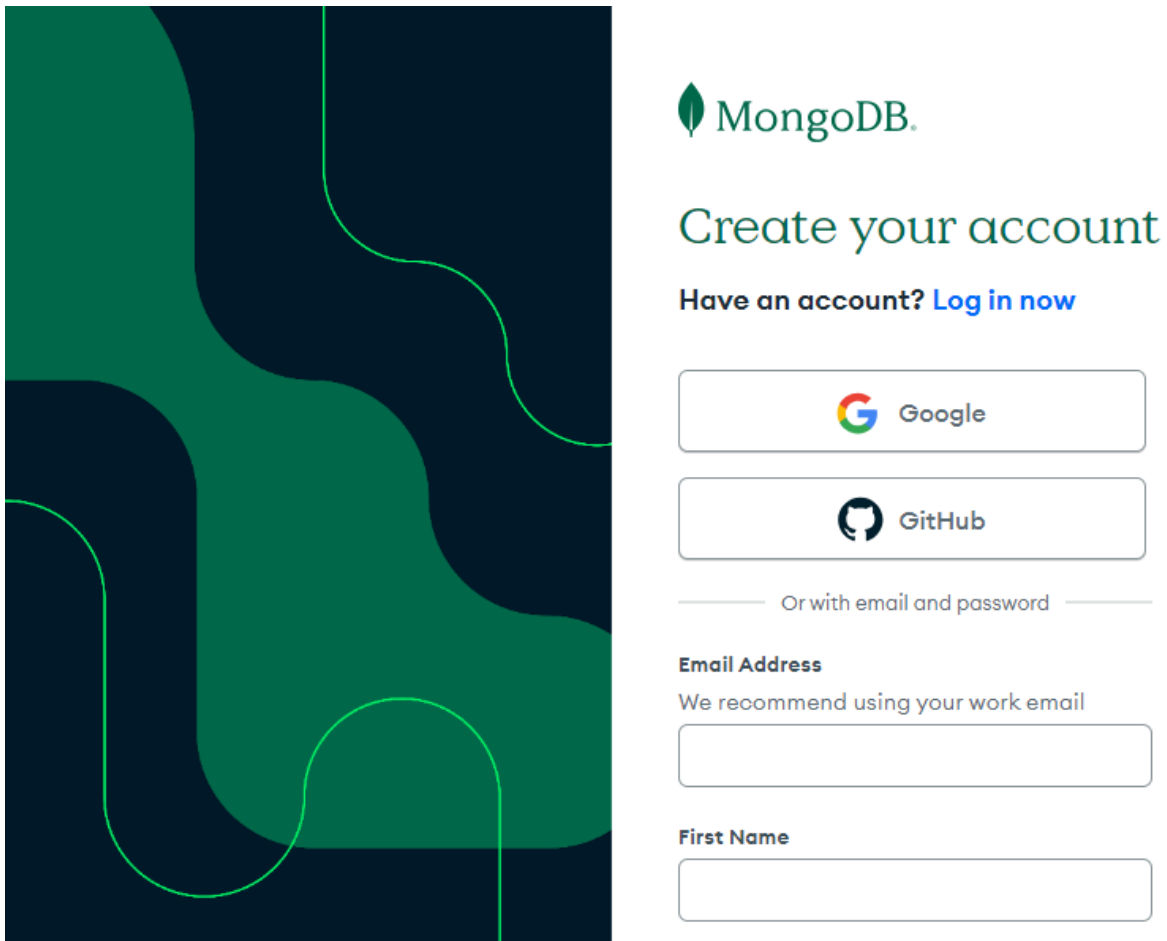


Creando la BD de Mongo:


Para crear la BD que albergará la tabla de auditoría, se usa el servicio de Mongo DB Atlas (<https://www.mongodb.com/es/atlas/database>). Allí, se crea una cuenta:




MongoDB.

Create your account

Have an account? [Log in now](#)

 Google

 GitHub

Or with email and password

Email Address
We recommend using your work email

First Name

Al crear la cuenta, se debe dirigir a la pantalla principal y se escoge la opción de New Project ubicada en la parte superior derecha.

ESTEBAN ENRIQUE'S ORG - 2023-10-20

Projects

New Project

Find a project...



Se debe seleccionar un nombre para el proyecto a crear y dar en siguiente para, de ser necesario, agregar más colaboradores para el mismo. Por último, se selecciona create Project.

Create a Project

Name Your Project > Add Members

Name Your Project

Project names have to be unique within the organization (and other restrictions).

vueltaAColombia

Cancel

Next

Create a Project

✓ Name Your Project > Add Members

Add Members and Set Permissions

Invite new or existing users via email address...

Give your members access permissions below.

esteban.cuervo@upto.edu.
co (you)


Project Owner


Back


Cancel


Create Project

Al ser un servicio gratuito, se tendrán algunas limitaciones en cuanto a funcionalidades, sin embargo, para efectos del ejercicio, los servicios gratuitos serán suficientes. Al crear el proyecto, debe aparecer una página como la que se ve a continuación:

 Atlas

Esteban Enri...



Access Manager Billing

vueltaAColombia 

Data Services

App Services

Charts

Overview

DEPLOYMENT

Database

Data Lake

SERVICES

Device Sync

Triggers

Data API

Data Federation

Search

Stream Processing

SECURITY

Backup

Database Access

ESTEBAN ENRIQUE'S ORG - 2023-10-20 > VUELTAACOLOMBIA

Overview



Create a deployment


Choose your cloud provider, region, and specs.

+ Create

Para continuar con la creación de la BD, primero debemos crear el cluster donde se almacenará. Se debe seleccionar la opción de Database Access disponible en el menú izquierdo, debe aparecernos una página donde nos indica que no tenemos usuarios y nos da la opción de agregar uno nuevo.

Database Access

Database UsersCustom Roles



Create a Database User

Set up database users, permissions, and authentication credentials in order to connect to your clusters.

Add New Database User

Debemos crear un nuevo database user para la administración de la base de datos, así que seleccionamos esta opción y lo creamos con usuario y password para mayor practicidad. Tendremos más opciones que, para efectos de practicidad del ejercicio, no son necesarias.

Add New Database User

Create a database user to grant an application or user access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding [Access Manager](#).

Authentication Method

Password	Certificate	AWS IAM (MongoDB 4.4 and up)	PREVIEW Federated Auth (MongoDB 7.0 and up)
-----------------	--------------------	--	--

MongoDB uses [SCRAM](#) as its default authentication method.

Password Authentication

e.g. new-user_31	
Enter password	SHOW
Autogenerate Secure Password	Copy

Se diligencian los campos requeridos y se le asignan privilegios al usuario a crear. Como necesitamos que nuestro usuario pueda tener rol de administrador, seleccionamos Atlas admin.

Database User Privileges

Configure role based access control by assigning database user a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. **You must choose at least one role or privilege.** [Learn more about roles.](#)

Built-in Role

Select one [built-in role](#) for this user.

0 SELECTED



Add Built In Role

Custom Roles

Select your [pre-defined custom role\(s\)](#). Create a custom role in the [Custom Roles](#) tab.



Specific Privileges

Select multiple privileges and what database and collection they are associated with. Leaving collection blank will grant this role for all collections in the database.



Restrict Access to Specific Clusters/Federated Database Instances

Enable to specify the resources this user can access. By default, all resources in this project are accessible.



Temporary User

This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week.



Built-in Role

1 SELECTED

Select one [built-in role](#) for this user.

Atlas admin

También se tiene la opción de personalizar los roles que se pueden asignar o especificar que es un usuario temporal de ser necesario. Al adecuar el usuario a nuestras necesidades, procedemos a dar click en add user.

Temporary User

This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week.



Cancel

Add User

Al finalizar, debemos observar la siguiente interfaz:

ESTEBAN ENRIQUE'S ORG - 2023-10-20 > VUELTA COLOMBIA

Database Access

Database Users

Custom Roles

+ ADD NEW DATABASE USER

User Name	Authentication Method	MongoDB Roles	Resources	Actions
vueltaColombia	SCRAM	atlasAdmin@admin	All Resources	EDIT DELETE

Con el usuario creado, procedemos a crear la base de datos junto a la colección donde almacenaremos nuestra auditoría. En la siguiente vista, vamos a Browse Collections.

vueltaColombiaCluster

Connect

View Monitoring

Browse Collections

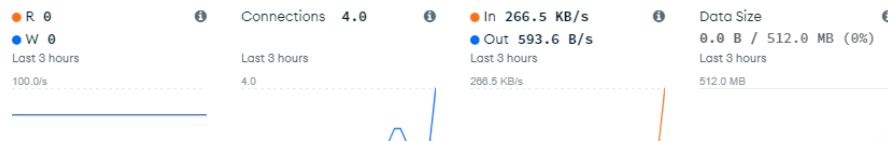
...

FREE SHARED

Enhance Your Experience

For production throughput and richer metrics, upgrade to a dedicated cluster now!

Upgrade





Explore Your Data

- **Find:** run queries and interact with documents
- **Indexes:** build and manage indexes
- **Aggregation:** test aggregation pipelines
- **Search:** build search indexes

[Load a Sample Dataset](#)[Add My Own Data](#)[Learn more in Docs and Tutorials](#)

Estando allí, nos aparecerá otra vista donde denotaremos que no contamos con bases de datos ni con colecciones. Para crear la BD junto a la colección de auditoría, seleccionamos la opción de add my own data. Allí, nos aparecerá la opción de asignarle un nombre a nuestra base de datos y a nuestra colección. Confirmamos cambios dando click en create.

Create Database



Database name ?

Collection name ?

Additional Preferences

[Cancel](#)[Create](#)

De esta forma, hemos creado la base de datos en MongoDB, al igual que la colección.

DATABASES: 1 COLLECTIONS: 1

VISUALIZE YOUR DATA

REFRESH

+ Create Database

Search Namespaces

vueltaColombia

audit_entry

vueltaColombia.audit_entry

STORAGE SIZE: 4KB LOGICAL DATA SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB

Find

Indexes

Schema Anti-Patterns

Aggregation

Search Indexes

INSERT DOCUMENT

Filter

Type a query: { field: 'value' }

Reset

Apply

More Options

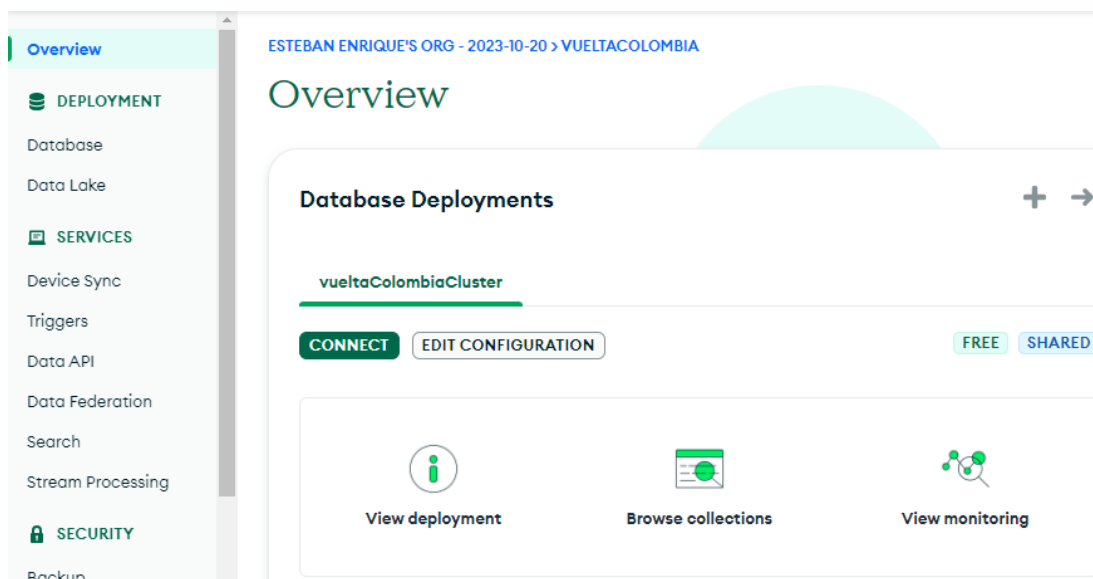
QUERY RESULTS: 0

Conectando MongoDB con el aplicativo en java:

Con el aplicativo Java creado junto a las entidades y operaciones CRUD, se plantea que, al ejecutar alguna de estas operaciones, esta se almacene en la base de datos de Mongo. Siendo así, necesitamos añadir a nuestras dependencias la correspondiente a MongoDB, esta es:

```
implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'
```

Lo siguiente a modificar es nuestro archivo application.properties. Allí, ponemos nuestra uri de conexión con el cluster y la base de datos creada. Para obtener esta uri, nos dirigimos a Atlas y allí, en la opción de Overview, seleccionamos connect:



En connect, seleccionamos la opción de drivers por cuanto necesitamos la uri de conexión.

×



Drivers

Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)



Allí, seleccionamos el driver que necesitamos y la versión. Esto nos generará una cadena de conexión, la cual, debemos añadir en nuestro proyecto, específicamente en el archivo `application.properties`.



Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver Version

Java 4.3 or later

2. Install your driver

[View MongoDB Java Driver installation instructions.](#)

3. Add your connection string into your application code

 View full code sample

```
mongodb+srv://vueltaColombia:<password>@vueltacolombiaccluster.lrlyhbf.mongodb.net/?  
retryWrites=true&w=majority
```



Replace **<password>** with the password for the **vueltaColombia** user. Ensure any option params are [URL encoded](#).

```
# Configuración de MongoDB
spring.data.mongodb.uri=mongodb+srv://vueltaColombia:uptc@vueltaColombiacluster.lrl9ybf.mongodb.net/vueltaColombia
```

Y así, realizaríamos la conexión hacia MongoDB desde nuestro aplicativo java.

Como la orientación del problema es guardar los datos de auditoría en una colección, así como se hizo para la creación de las tablas, también debemos crear una entidad para las colecciones que tendremos y un repositorio, sin embargo, estos tendrán unos leves cambios.

Para la entidad, la cual llamaremos AuditEntry, crearemos una clase AuditEntry que contendrá los campos que queremos se guarden en nuestra colección, para el caso, serán los siguientes:

```
3 usages
@Document(collection = "audit_entry")
public class AuditEntry {
    @Id
    private String id;
    4 usages
    private String action;
    4 usages
    private String table;
    4 usages
    private Map<String, Object> data;
}
```

Como podemos observar, ya no usaremos la anotación @Table por cuanto no estamos manejando una tabla, ahora usaremos @Document donde en los atributos pondremos collection y lo inicializaremos con el nombre de nuestra colección creada, para el caso, audit_entry. También tendremos:

- Id: de tipo String, el cual si va denotado con la anotación @Id y autogenerado por mongo.
- Action: operación CRUD ejecutada.
- Table: tabla sobre la cual se ejecutó la acción.
- Data: datos que se operaron en esa tabla.

Por el lado del repositorio, en este ya no se hará uso de JPA sino de MongoRepository, siendo la estructura la misma que en el repositorio de JPA, sin embargo, no se usará la anotación @Repository por cuanto MongoRepository ya define esta interface como un repository.

```
1 messages
public interface AuditEntryRepository extends MongoRepository<AuditEntry, String> {
}
```

Con estos cambios, ya podremos implementar nuestra lógica en los service de nuestras entidades para poder enviar datos a la colección en MongoDB. A continuación, un ejemplo en cuanto al ingreso de datos en la tabla Patrocinadores.

Tenemos nuestra función saveSponsor con parámetro de entrada un objeto sponsor de tipo Sponsor en nuestra clase SponsorService. En primer lugar, se debe inyectar mongoTemplate. Para crear el patrocinador, simplemente bastaría con llamar al repository.save(), sin embargo, cambiamos un poco la lógica para que, al ejecutar este método en la clase controller, inserte datos en la colección.

```

@Service
public class SponsorService {
    @Autowired
    private SponsorRepository sponsorRepository;
    @Autowired
    private MongoTemplate mongoTemplate;

    1 usage
    public Sponsor saveSponsor(Sponsor sponsor){
        Sponsor savedSponsor = sponsorRepository.save(sponsor);

        AuditEntry auditEntry = new AuditEntry();
        auditEntry.setAction("create");
        auditEntry.setTable("sponsor");
        Map<String, Object> data = new HashMap<>();
        data.put("sponsorId", savedSponsor.getSponsorId());
        data.put("sponsorName", savedSponsor.getSponsorName());

        auditEntry.setData(data);
        mongoTemplate.save(auditEntry, collectionName: "audit_entry");
        return sponsorRepository.save(savedSponsor);
    }
}

```

Ejecutamos nuestro aplicativo e ingresamos datos por postman:

POST http://localhost:8080/sponsor/createSponsor Send

Params Auth Headers (8) **Body** Pre-req. Tests Settings

raw JSON Beautify

```

1 {
2   "sponsorId": 2143,
3   "sponsorName": "Patrocinador 2"
4 }

```

Body 200 OK 1052 ms 213 B Save as Example

Pretty Raw Preview Visualize JSON

```

1 {
2   "sponsorId": 2143,
3   "sponsorName": "Patrocinador 2"
4 }

```

Y observamos la inserción tanto en MySQL como en MongoDB:

143 • select * from patrocinadores;

nit_patrocinador	nombre
2143	Patrocinador 2
218743	Patrocinador 1
HULL	HULL

QUERY RESULTS: 1-1 OF 1

```

_id: ObjectId('6532221ca6438277028a2ea2')
action: "create"
table: "sponsor"
data: Object
  sponsorId: 2143
  sponsorName: "Patrocinador 2"
_class: "com.uptc.frw.vuelta Colombia.jpa.entity.AuditEntry"

```

Problemas al conectar Mongo:

- org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'mongoDatabaseFactory' defined in class path resource [org/springframework/boot/autoconfigure/data/mongo/MongoDatabaseFactoryConfiguration.class]: Bean instantiation via factory method failed; nested exception is org.springframework.beans.BeanInstantiationException: Failed to instantiate [org.springframework.data.mongodb.core.MongoDatabaseFactorySupport]: Factory method 'mongoDatabaseFactory' threw exception; nested exception is java.lang.IllegalArgumentException: Database name must not be empty!

```
56 org.springframework.beans.factory.  
   BeanCreationException: Error creating bean with name  
     'mongoDatabaseFactory' defined in class path  
     resource [org/springframework/boot/autoconfigure/data  
       /mongo/MongoDatabaseFactoryConfiguration.class]: Bean  
       instantiation via factory method failed; nested  
       exception is org.springframework.beans.  
     BeanInstantiationException: Failed to instantiate [  
       org.springframework.data.mongodb.core.  
       MongoDatabaseFactorySupport]: Factory method '  
       mongoDatabaseFactory' threw exception; nested  
       exception is java.lang.IllegalArgumentException:  
       Database name must not be empty!
```

57

```
00:30:55.231 INFO 26228 --- [          main] ConditionEvaluationReportLoggingListener :  
ing ApplicationContext. To display the conditions report re-run your application with 'debug' enabled.  
00:30:55.263 ERROR 26228 --- [          main] o.s.boot.SpringApplication : Application run failed  
'framework.beans.factory.BeanCreationException' Create breakpoint : Error creating bean with name 'mongoDatabaseFactory' defined in class path resource [org/springf
```

Solución: La url que nos proporciona Atlas para conectarnos desde nuestro aplicativo es bastante acertada, sin embargo, se debe modificar para agregar el nombre de nuestra base de datos dentro del cluster.

Cadena original:

```
# Configuración de MongoDB  
spring.data.mongodb.uri=mongodb+srv://vueltaColombia:<u>uptc@vueltacolombiac</u>cluster.lrlhbf.mongodb.net/?retryWrites=true&w=majority
```

Cadena adecuada a la base de datos:

```
# Configuración de MongoDB  
spring.data.mongodb.uri=mongodb+srv://vueltaColombia:<u>uptc@vueltacolombiac</u>cluster.lrlhbf.mongodb.net/<u>vueltaColombia
```

Así, la definición de la uri para la conexión a la base de datos, sería la siguiente:

```
spring.data.mongodb.uri=mongodb+srv://<username>:<password>@<cluster>.mongodb.net/<database>
```