Instituto Tecnológico de Costa Rica

Centro Académico de Alajuela

Bases de Datos II

Investigación I

Bases de Datos No Relacionales (NoSQL – MongoDB)

Grupo 8:

Marco Rodríguez Vargas – Carné: 2022149445

Kevin Carranza Jiménez – Carné: 2015100260

Jorge Esteban Benavides Castro – Carné: 2022230697

Profesor:

Alberto Shum Chan

II Semestre, 2023

Escuela de Ingeniería en Computación	**Prof. Alberto Shum Chan**
Bachillerato en Ingeniería en Computación	**Base de Datos II**
Sede Interuniversitaria de Alajuela	**Semestre II, 2023**

Tema: Trabajo de investigación

Objetivo: Realizar una investigación sobre bases de datos no relacionales (NoSQL – MongoDB) y aplicarlos conocimientos adquiridos.

Entrega en el TecDigital:

- Deben presentar un archivo PDF y los scripts del ejemplo práctico.
- Forma de trabajo en grupos de 3 personas.

En el ámbito de la gestión de bases de datos, las bases de datos relacionales (RDBMS) han sido históricamente predominantes, ofreciendo un marco robusto y estructurado para almacenar, recuperar y manipular datos. Sin embargo, con la emergencia de las Big Data y la necesidad de mayor flexibilidad y escalabilidad, las bases de datos NoSQL, incluyendo las bases de datos No Relacionales como MongoDB, han ganado una significativa atención y adopción.

Este proyecto de investigación tiene como objetivo proporcionar un análisis comparativo práctico entre las bases de datos relacionales y MongoDB, una base de datos no relacional prominente. Los estudiantes deben explorar y contrastar cómo las operaciones CRUD, la gestión de índices, procedimientos o funciones, la integridad referencial, y las restricciones son manejadas en el paradigma no relacional.

Instrucciones:

1. **Revisión Bibliográfica:**

Investigar y documentar los principios fundamentales de las bases de datos no relacionales. Reseñar las características, ventajas y desventajas. Debe incluir al menos 3 referencias. 10 pts.

2. **Implementación Práctica:**

Crear una base en MongoDB con base en 3 o 4 tablas utilizadas en el modelo de datos entregado para proyecto #1 (pueden modificar las existentes o crear tablas nuevas relacionadas con su modelo de datos.). Deben documentar el proceso de instalación (imágenes) e incluir el código respectivo. También deben incluir mensajes de error, problemas encontrados y cómo lo solucionaron (enlaces, videos, etc.). 10pts

3. **Operaciones CRUD.**

Presentar 2 ejemplos de cada operación CRUD aplicada sobre las tablas seleccionadas en el punto anterior. 8pts

4. **Índices, integridad referencial y restricciones.**

Investigar e implementar un ejemplo sobre el manejo de índices, integridad referencial y restricciones a nivel de una base de datos no relacional. 9pts.

5. **Procedimientos y funciones.**

Explique cómo es el manejo de procedimientos y funciones en una base de datos no relacional como MongoDB. 3pts

6. **Tipos de datos**

Cite y explique 5 tipos de datos existentes en MongoDB. Presente un ejemplo para cada uno.10pts

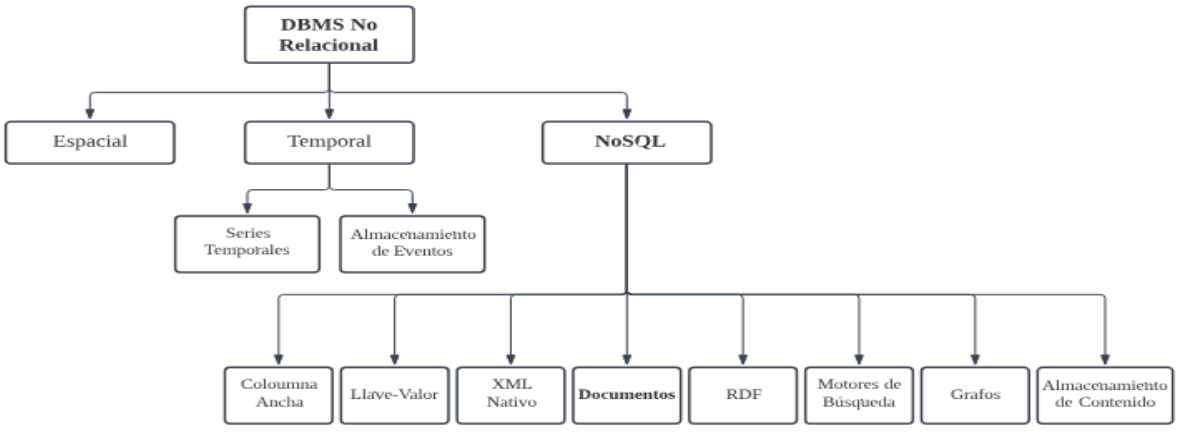
Revisión Bibliográfica

Características

Las bases de datos NoSQL, investigadas por Phiri y Kunda (2017) y Gupta et al. (2017), son una alternativa esencial en la gestión de datos actual debido a su flexibilidad, escalabilidad y capacidad para manejar diversos tipos de datos. Estos sistemas incluyen almacenes de documentos, almacenes de columnas anchas, bases de datos clave-valor y bases de datos de grafos, adaptándose a diferentes requisitos. Una característica destacada es su escalabilidad horizontal, que permite distribuir la carga en servidores económicos en lugar de actualizar costosamente el hardware. Son eficientes para manejar grandes volúmenes de datos, siendo ideales para aplicaciones de Big Data.

No obstante, enfrentan desafíos como la falta de un lenguaje de consulta estándar y preocupaciones de seguridad. La diversidad de implementaciones ha dado lugar a múltiples lenguajes e interfaces. La elección entre bases de datos relacionales y NoSQL depende de las necesidades específicas, y ambas coexisten para complementarse en diferentes contextos (Gupta et al., 2017). Las bases de datos NoSQL son versátiles y eficaces para la gestión de datos, adaptándose a la diversidad de datos y necesidades en constante evolución (Li & Li, 2018).

Figura 1.



Nota. Adaptado de *Types of DBMSs* (p. 2), de A. Akhtar, 2023, University of Pakistan.

En este documento se aborda la base de datos MongoDB, que se clasifica como una base de datos NoSQL de documentos y es ampliamente conocida. MongoDB es una opción atractiva debido a su naturaleza de código abierto, lo que la hace popular en la comunidad informática (Chauhan, 2019). Una característica destacada es su modelo de almacenamiento de información, que se basa en documentos BSON (Binary JavaScript Object Notation), similares archivos JSON pero optimizados para el almacenamiento y transmisión de datos (Valverde et al., 2019). Estos documentos pueden contener campos, valores y otros documentos, lo que permite una estructura flexible y desnormalizada (Valverde et al., 2019).

Figura 2.

RANKING OF TOP 20 DBMS

Rank	DBMS	Score	Type
1	Oracle	1262.66	Relational
2	MySQL	1228.38	Relational
3	Microsoft SQL Server	981.95	Relational
4	PostgreSQL	577.15	Relational
5	MongoDB	496.16	Document
6	Redis	168.31	Key-value
7	IBM Db2	165.15	Relational
8	Elasticsearch	155.76	Search engine
9	SQLite	130.2	Relational
10	Cassandra	114	Wide column
11	Microsoft Access	113.45	Relational
12	MariaDB	97.98	Relational
13	Splunk	90.05	Search engine
14	Hive	82.68	Relational
15	Microsoft Azure SQL Database	75.22	Relational
16	Amazon DynamoDB	75.2	Multi-model
17	Teradata	68.95	Relational
18	Neo4j	57.16	Graph
19	SAP HANA	53.81	Relational
20	Solr	51.79	Search engine

Nota. *Ranking of Top 20 DBMS*, de A. Akhtar, 2023, Virtual University of Pakistan.

Las bases de datos NoSQL no cumplen con las garantías de ACID, pero cumplen con el principio BASE, que garantiza que los datos estén disponibles, incluso si no están consistentes. Las bases de datos NoSQL son adecuadas para aplicaciones que requieren escalabilidad y rendimiento, como aplicaciones de análisis de datos o aplicaciones móviles.

•**Escalabilidad:** Las bases de datos NoSQL se pueden escalar horizontalmente, lo que significa que se pueden agregar más nodos al clúster para aumentar el rendimiento y la capacidad (Abramova & Bernardino, 2013).

•**Rendimiento:** Las bases de datos NoSQL están diseñadas para proporcionar un rendimiento alto y consistente.

•**Disponibilidad:** Las bases de datos NoSQL están diseñadas para proporcionar una alta disponibilidad, incluso en caso de falla del sistema.

•**Facilidad de desarrollo:** Las bases de datos NoSQL son más fáciles de desarrollar que las bases de datos relacionales (Mapanga & Kadebu, 2013).

Ventajas

•**Flexibilidad y diversidad de modelos de datos:** Las bases de datos NoSQL permiten almacenar datos semi-estructurados y no estructurados, lo que las hace más versátiles que las bases de datos relacionales. Según Phiri y Kunda (2017), los modelos de datos más destacados en el contexto NoSQL incluyen Key-Value, Document-oriented, Column Databases y Graph Databases. Ha evolucionado a un ritmo muy alto lo cual permite diferentes acercamientos para manejar los datos (Nayak, Poriya y Poojary, 2013).

•**Escalabilidad horizontal**: Las bases de datos NoSQL se basan en la distribución de la carga en múltiples servidores de bajo costo. Esto permite mantener altos niveles de rendimiento incluso cuando la cantidad de datos y usuarios crece significativamente. Según Li y Li (2018), esta arquitectura escalable resulta en un menor costo total de propiedad.

•**Gestión eficiente de grandes volúmenes de datos:** Las bases de datos NoSQL están diseñadas para manejar grandes cantidades de información. Su capacidad para distribuir la carga de trabajo de

manera efectiva es esencial en aplicaciones intensivas en datos, como redes sociales o análisis de datos masivos.

•**Propiedades de consistencia, disponibilidad y tolerancia a fallos:** Las bases de datos NoSQL se posicionan en diferentes puntos del espectro CAP, lo que permite a los desarrolladores elegir la que mejor se adapte a sus necesidades específicas. Según Phiri y Kunda (2017), las NoSQL pueden priorizar la disponibilidad constante incluso en caso de fallos de hardware. Algunos proveedores de servicios de bases de datos NoSQL como Riak y Cassandra están programados para manejar fallas de hardware.

Desventajas

Las bases de datos NoSQL presentan desventajas significativas en comparación con las bases de datos relacionales. Estas desventajas incluyen la falta de un estándar común en cuanto a lenguaje de consulta, lo que dificulta la adopción y el aprendizaje para los desarrolladores (Phiri & Kunda, 2017). Además, la falta de una estructura relacional en algunas bases de datos NoSQL, como MongoDB, implica que las regulaciones de la información deben ser implementadas en la lógica de la aplicación, lo que puede generar procesamiento incorrecto de la información y vulnerabilidades (Chauhan, 2019).

La seguridad es otro desafío importante en las bases de datos NoSQL, ya que a menudo carecen de mecanismos de seguridad sólidos incorporados en su núcleo, lo que puede resultar en vulnerabilidades y ataques (Gupta et al., 2017). La falta de capacidades de registro adecuadas dificulta el seguimiento de amenazas y ataques internos. Además, la ausencia de soporte para operaciones JOIN dificulta la realización de consultas complejas que requieran la unión de datos de múltiples fuentes (Li & Li, 2018).

En el caso específico de MongoDB, la asincronía en las operaciones de escritura puede llevar a problemas de fiabilidad de datos, ya que una operación de escritura puede fallar, pero aún así, parte de la información se escribirá, lo que puede resultar en información inconsistente o con errores (Chauhan, 2019).

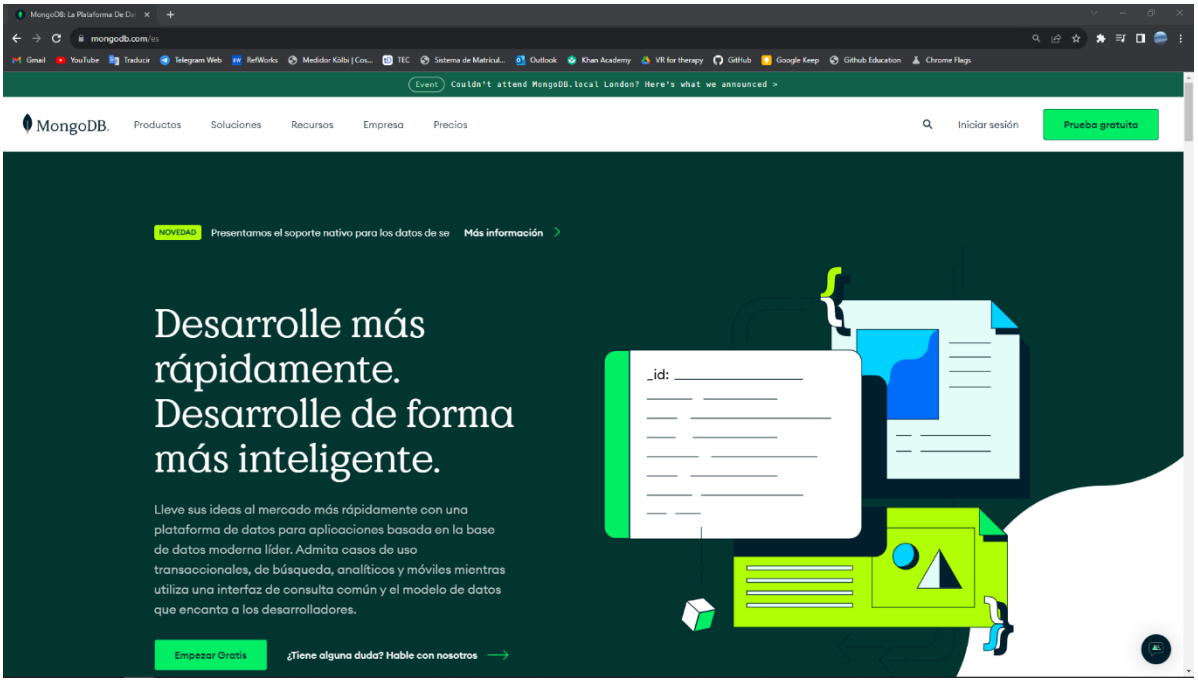
En resumen, las bases de datos NoSQL enfrentan desafíos relacionados con la falta de estandarización en el lenguaje de consulta, la seguridad, la capacidad limitada para realizar operaciones JOIN y problemas de fiabilidad de datos en casos como MongoDB. Estas desventajas deben ser cuidadosamente consideradas al elegir una plataforma de gestión de datos, dependiendo de las necesidades específicas de la aplicación (Phiri & Kunda, 2017; Gupta et al., 2017; Li & Li, 2018). Además, se ha señalado que las bases de datos NoSQL pueden ser inmaduras y carecer de una interfaz estándar, lo que también debe ser tenido en cuenta (Nayak, et al., 2013).

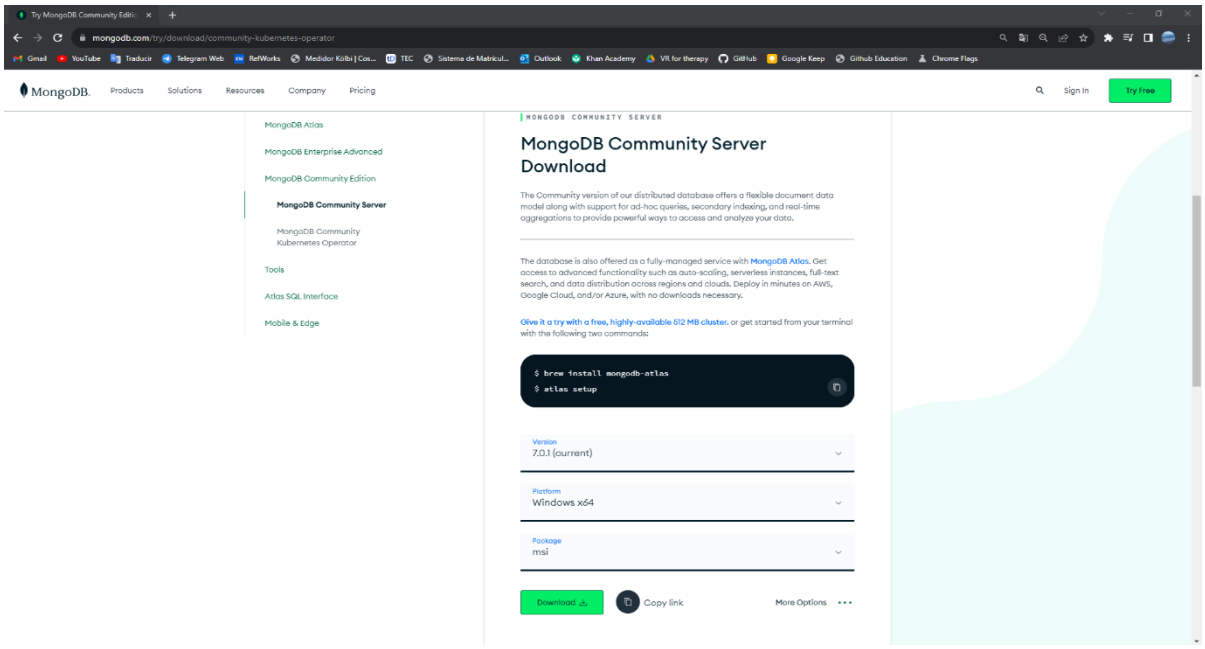
Implementación Práctica

Video de instalación usado como base:

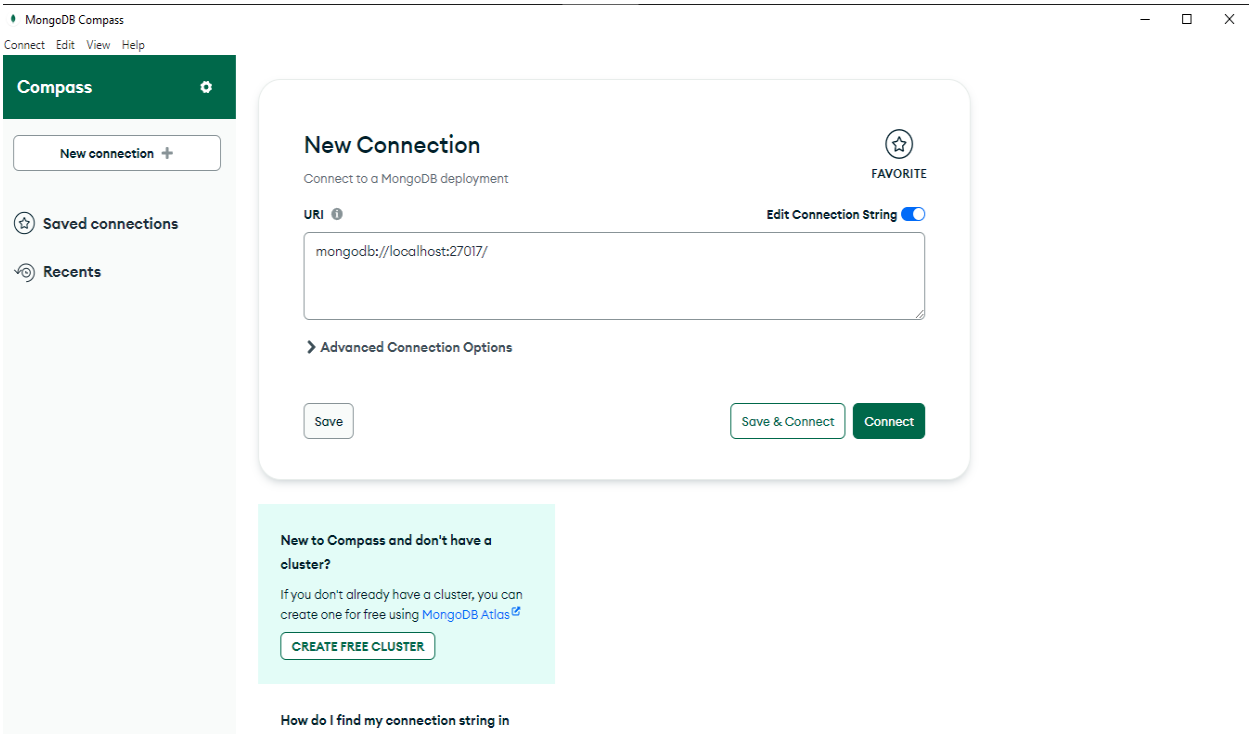
Facultad Autodidacta. (2022). *Instalar y configurar mongodb 2022* [Video]. YouTube. https://www.youtube.com/watch?v=LibtQECAR1U&ab_channel=FacultadAutodidacta

Página Oficial de MongoDB

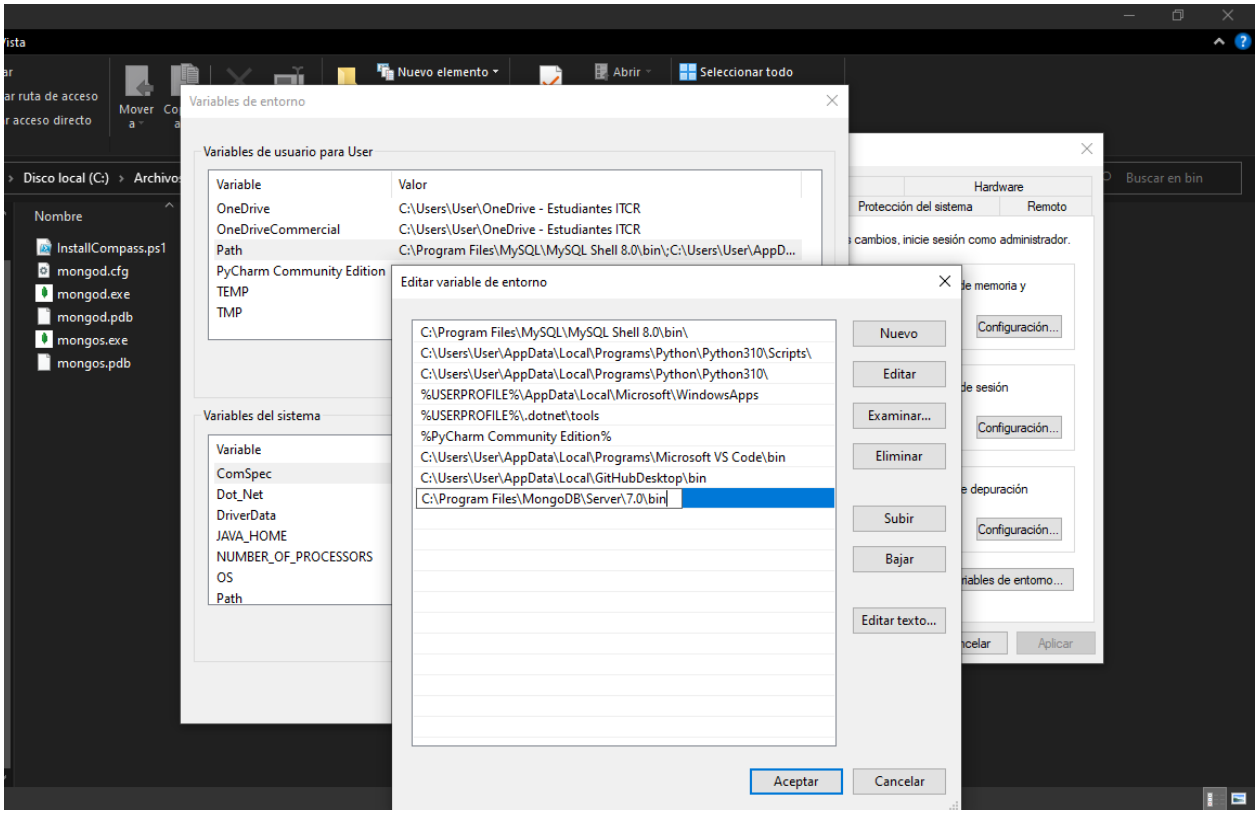




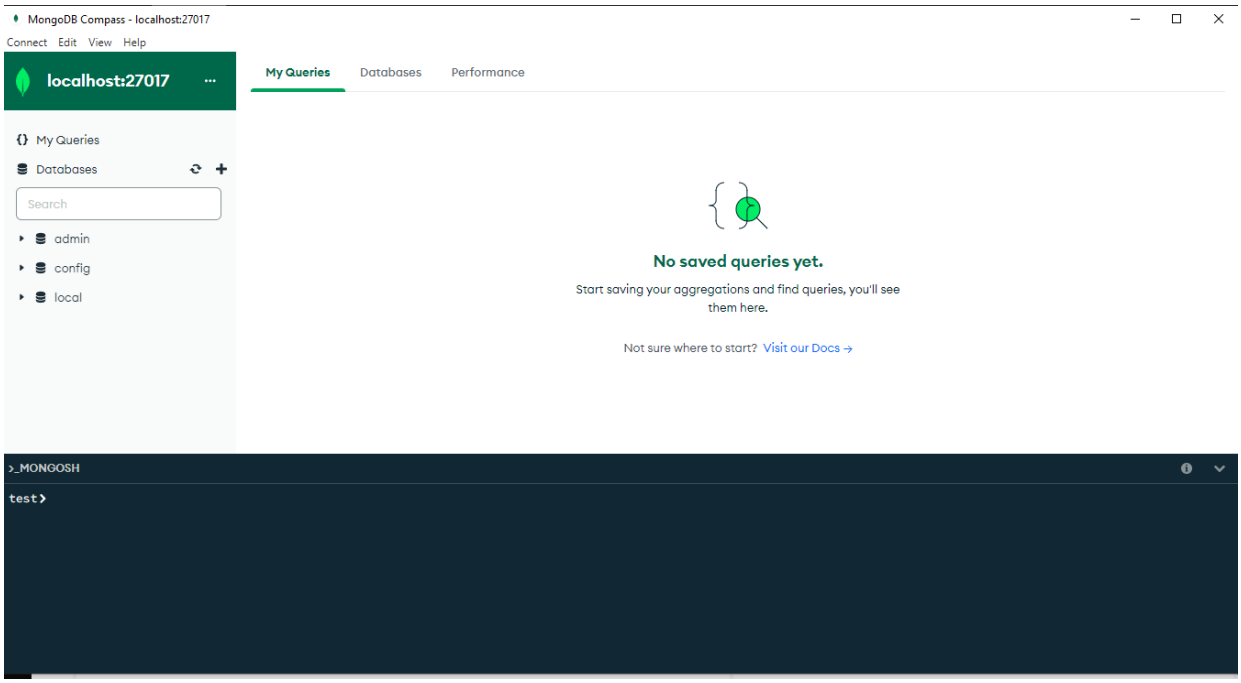
Crear la conexión de MongoDB al puerto 27017



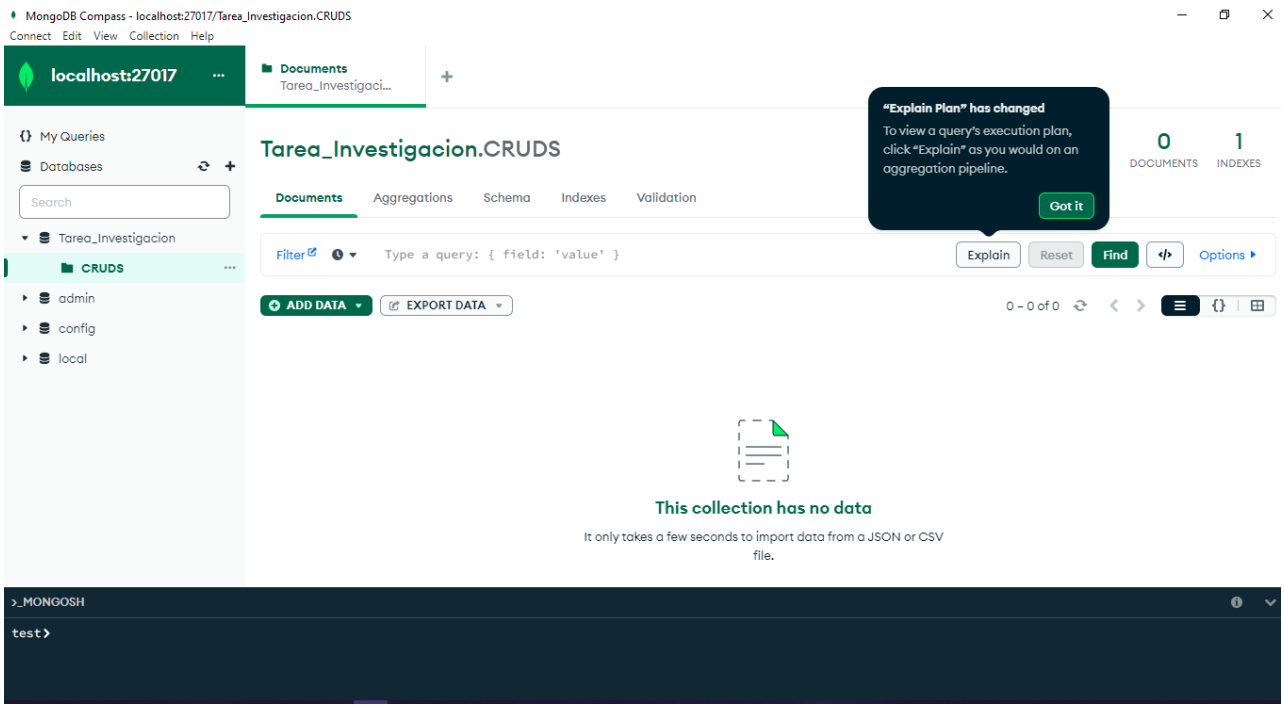
Agregar el bin al path



Crear una nueva base de datos



Agregar colecciones a la base de datos



Se crea un collection (coleccion en español) que va a almacenar los datos.

```
// Select the database to use.
use('Suministro_Construccion');

// Insert a few documents into the sales collection.
db.getCollection('Distribuidor').insertMany([
  { 'id': 0, 'nombre': 'Importadora de Materiales ContruMax', 'telefono': 12345678, 'correo':
'construmax@gmail.com' },
  { 'id': 1, 'nombre': 'Grupo Construvalle', 'telefono': 23456789, 'correo': 'construvalle@gmail.com'
},
  { 'id': 2, 'nombre': 'Suministros EcoCemento', 'telefono': 34567890, 'correo': 'eco-
cemento@gmail.com' },
  { 'id': 3, 'nombre': 'Distribuidora Arquitectura Verde', 'telefono': 45678901, 'correo':
'arquitectura-verde@gmail.com' },
  { 'id': 4, 'nombre': 'Construcentro Pacifico', 'telefono': 56789012, 'correo':
'construcentro@gmail.com' }
]);

db.getCollection('Producto').insertMany([
  { 'id': 0, 'nombre': 'Cemento', 'descripcion': 'Cemento para construccion', 'precio_unitario': 5000,
'cantidad_stock': 100 },
  { 'id': 1, 'nombre': 'Arena', 'descripcion': 'Arena para construccion', 'precio_unitario': 3000,
'cantidad_stock': 200 },
```

```
  { 'id': 2, 'nombre': 'Ladrillos', 'descripcion': 'Ladrillos de construccion', 'precio_unitario': 6000,
'cantidad_stock': 150 },
  { 'id': 3, 'nombre': 'Madera', 'descripcion': 'Madera para estructuras', 'precio_unitario': 7000,
'cantidad_stock': 50 },
  { 'id': 4, 'nombre': 'Cemento Armado', 'descripcion': 'Cemento armado para construccion',
'precio_unitario': 5500, 'cantidad_stock': 75 }
])

db.getCollection('Cliente').insertMany([
  { 'id': 112233445, 'nombre': 'Juan', 'primer_apellido': 'Perez', 'segundo_apellido': 'Fernandez',
'direccion': 'Calle Principal #123', 'telefono': 601234567, 'correo_electronico':
'juanperez@gmail.com' },
  { 'id': 223344556, 'nombre': 'Maria', 'primer_apellido': 'Solo', 'segundo_apellido': 'Solano',
'direccion': 'Avenida Central #456', 'telefono': 602345678, 'correo_electronico':
'mariasoto@gmail.com' },
  { 'id': 334455667, 'nombre': 'Luis', 'primer_apellido': 'Vargas', 'segundo_apellido': 'Jimenez',
'direccion': 'Calle de los Pinos #789', 'telefono': 603456789, 'correo_electronico':
'luisvargas@gmail.com' },
  { 'id': 445566778, 'nombre': 'Ana', 'primer_apellido': 'Rodriguez', 'segundo_apellido': 'Morales',
'direccion': 'Avenida del Sol #101', 'telefono': 604567890, 'correo_electronico':
'anarodriguez@gmail.com' },
  { 'id': 556677889, 'nombre': 'Jose', 'primer_apellido': 'Chaves', 'segundo_apellido': 'Gonzalez',
'direccion': 'Calle de las Flores #202', 'telefono': 605678901, 'correo_electronico':
'josechavez@gmail.com' }
])
```

Errores encontrados

Al usar la función de “InsertMany” se utilizaron paréntesis incorrectos debido que existe una capa adicional de paréntesis en comparación a “InsertOne”. Se solucionó revisando nuevamente el código y notando que se usaron los paréntesis equivocados.

Operaciones CRUD

Create sobre Distribuidor

```
use('Suministro_Construccion');
db.getCollection('Distribuidor').insertOne({
  'id': 5,
  'nombre': 'Importadora de Materiales ContruEstebanQuito',
  'telefono': 87654321,
  'correo': 'construestebanquito@gmail.com'
})
```

Create sobre Cliente

```
use('Suministro_Construccion');
db.getCollection('Cliente').insertMany([
  { 'id': 321459879, 'nombre': 'Bruce', 'primer_apellido': 'Wayne', 'segundo_apellido': 'Diaz',
'direccion': 'Ciudad Gotica #123', 'telefono': 607418529, 'correo_electronico':
'soybatman@incwayne.com' },
  { 'id': 789654123, 'nombre': 'Petter', 'primer_apellido': 'B', 'segundo_apellido': 'Parker',
'direccion': 'New York #456', 'telefono': 606548520, 'correo_electronico': 'spiderman@gmail.com'
}
])
```

Read sobre Cliente

```
use('Suministro_Construccion');
db.getCollection('Cliente').findOne({"nombre": 'Petter'})
```

Read sobre Distribuidor

```
use('Suministro_Construccion');
db.getCollection('Distribuidor').find({'telefono': 87654321})
```

Update sobre Producto

```
use('Suministro_Construccion');
db.getCollection('Producto').updateOne({nombre: 'Cemento'}, {$set:{descripcion: "Cemento para simientos."}})
```

Update sobre Cliente

```
use('Suministro_Construccion');
db.getCollection('Cliente').replaceOne({nombre: 'Bruce'}, {nombre: 'Bruno'})
```

Delete sobre Distribuidor

```
use('Suministro_Construccion');
db.getCollection('Distribuidor').deleteOne({nombre: 'Importadora de Materiales ContruEstebanQuito'})
```

Delete sobre Cliente

```
use('Suministro_Construccion');
db.getCollection('Cliente').deleteMany({nombre: 'Petter'})
```

Índices, integridad referencial y restricciones

Índices

Los índices respaldan la ejecución eficiente de consultas en MongoDB. Si la aplicación está ejecutando repetidamente consultas en los mismos campos, puedes crear un índice en esos campos para mejorar el rendimiento de esas consultas. Para crear un índice, utiliza el método del shell createIndex() o el método equivalente para su controlador. Esta página muestra ejemplos para el Shell de MongoDB y los controladores (MongoDB, s.f.).

```
use('Suministro_Construccion');
db.getCollection('Cliente').createIndex( {id: 1}, {name: "cliente_id"})
db.getCollection('Cliente').getIndexes()
```

Integridad referencial

MongoDB utiliza el modelo clave/valor en donde la idea principal aquí consiste en utilizar una tabla hash donde hay una clave única y un puntero a un elemento de datos específico. El modelo de clave/valor es el más simple y fácil de implementar. Sin embargo, resulta ineficiente cuando solo se tiene interés en consultar o actualizar parte de un valor, entre otras desventajas (Krishnan, et al., 2016).

Los sistemas NoSQL ofrecen mucha menos funcionalidad que los sistemas tradicionales de gestión de bases de datos relacionales, especialmente en lo que respecta al aislamiento de transacciones y las operaciones de exploración. Sin embargo, pueden ser utilizados con éxito cuando la lógica de base de datos compleja no es un objetivo, pero sí lo es una operación distribuida a gran escala (Krishnan, et al., 2016). En este caso, MongoDB no posee integridad referencial, por lo que debe ser implementada mediante una lógica adicional lo cual no es práctico.

No se puede ejemplificar dado que no existe

Restricciones a nivel de una base de datos no relacional

Falta de Integridad Referencial

Debido a que la información no puede acceder a datos que estén contenidos en un espacio diferente al suyo: no existe una forma en la que se pueda asegurar la integridad de datos relacionados al insertar o modificar información almacenada en otras tablas.

Limitaciones en las consultas complejas

Están pensadas para consultas simples, por lo que puede tener dificultades para manejar consultas que involucren relaciones entre colecciones (Calvo et al., 2017).

```
use('Suministro_Construccion');
var nombreCliente = 'Juan';
db.getCollection('Cliente').find({ 'nombre': nombreCliente })
```

Procedimientos y funciones

El manejo de procedimientos y funciones en una base de datos NoSQL como MongoDB se diferencia notablemente de cómo se gestionan en las bases de datos relacionales convencionales. En MongoDB, el enfoque se centra en la flexibilidad y la adaptabilidad a las necesidades específicas de la aplicación, lo que se refleja en su diseño sin un lenguaje SQL tradicional para procedimientos almacenados y funciones.

En MongoDB, una de las principales herramientas para realizar operaciones de procesamiento de datos es la funcionalidad de agregación. Esta característica permite realizar operaciones complejas de filtrado, transformación y agrupación de datos mediante el uso de un "pipeline de agregación", que consiste en una serie de etapas secuenciales aplicadas a los documentos almacenados. A través de esta funcionalidad, los desarrolladores pueden lograr resultados similares a los procedimientos almacenados en bases de datos relacionales, pero con un enfoque más orientado a documentos (Gupta et al., 2017).

Otra opción que ofrece MongoDB es la capacidad de ejecutar código JavaScript directamente en el servidor de la base de datos. Esto se logra a través de la función mapReduce, que permite a los desarrolladores definir operaciones personalizadas de procesamiento de datos utilizando JavaScript. Aunque no es un procedimiento almacenado en el sentido tradicional, esta capacidad proporciona flexibilidad para realizar cálculos y transformaciones complejas en los datos almacenados.

En lugar de depender de procedimientos almacenados en la base de datos, en MongoDB es común trasladar la lógica de procesamiento de datos a la capa de la aplicación. Esto significa que gran parte de las operaciones de filtrado, agregación y transformación de datos se realizan en el código de la aplicación utilizando las capacidades proporcionadas por el controlador de MongoDB. En este enfoque, la base de datos se utiliza principalmente para el almacenamiento y la recuperación

de datos, mientras que la lógica de procesamiento se implementa en la aplicación (Phiri & Kunda, 2017; Li & Li, 2018).

Tipos de datos

Fecha y Hora

La fecha y hora son tipos de datos esenciales en cualquier sistema de gestión de datos, y MongoDB no es una excepción. En MongoDB, el tipo de dato "Date" se utiliza para representar fechas y horas (MongoDB, s.f.). Puede almacenar fechas en forma de marcas de tiempo desde el 1 de enero de 1970 (UNIX Epoch; punto de partida para medir el tiempo en sistemas operativos Unix y en muchas bases de datos, incluyendo MongoDB) hasta el 19 de enero de 2038 en milisegundos (Gupta et al., 2017).

Con milisegundos se refiere a una unidad de tiempo muy pequeña para representar fechas y horas. Un milisegundo es una milésima parte de un segundo, lo que permite una precisión extremadamente alta al representar el tiempo (Battah, Iraqi & Damiani, 2021). Se utiliza esta unidad de tiempo para medir la diferencia entre el UNIX Epoch y una fecha/hora específica. Por lo tanto, cuando se dice que MongoDB puede almacenar fechas desde el 1 de enero de 1970 hasta el 19 de enero de 2038 en milisegundos, significa que puede representar y calcular fechas y horas dentro de ese rango de tiempo con gran precisión.

Un ejemplo es el seguimiento de eventos en una aplicación de registro de actividad realizada por un usuario. Supongamos que se está desarrollando una aplicación de seguimiento de actividad física que registra las actividades de los usuarios, como caminar, correr o andar en bicicleta. Cada vez que un usuario realiza una actividad, se registra la fecha y hora exactas utilizando el tipo de dato "Date" de MongoDB. Esto permite a la aplicación mostrar un historial detallado de las actividades de los usuarios, calcular estadísticas como el tiempo total dedicado al ejercicio y generar gráficos de progreso basados en marcas de tiempo precisas.

```
use('Investigacion');
db.getCollection('Tiempo').insertOne({
  usuario: "EjemploUsuario123",
  actividad: "Caminata nocturna",
  fecha_hora: new ISODate("2023-09-15T08:00:00Z")
})
```

En este caso, la marca de tiempo registrada en la base de datos permite un seguimiento preciso de cuándo se realizó la actividad, lo que es fundamental para aplicaciones en las que el registro preciso de fecha y hora sea crucial.

Geolocalización (Geospatial)

MongoDB proporciona soporte para datos geoespaciales, lo que significa que puede manejar información relacionada con ubicaciones geográficas precisas (MongoDB, s.f.). Esto se logra a través de tipos de datos y operadores geoespaciales que permiten almacenar y consultar coordenadas de longitud y latitud, polígonos y otros datos geográficos (Tampubolon, Reinhardt, Sumaryono & Tobing, 2021).

Un ejemplo es en la industria de la logística y el transporte, MongoDB se utiliza ampliamente para el seguimiento de vehículos y entregas. Supongamos que una empresa de transporte de mercancías necesita llevar un registro de la ubicación en tiempo real de sus camiones y las entregas que realizan. Utilizando datos geoespaciales en MongoDB, la empresa puede almacenar la ubicación precisa de cada camión y las coordenadas de entrega. Esto permite a la empresa realizar un seguimiento en tiempo real de la ubicación de los camiones, asignar rutas eficientes y notificar a los clientes sobre el estado de sus entregas.

```
use('Investigacion');
db.getCollection('Transporte').insertOne({
  "camion_id": "CamionIntel21",
  "ubicacion_actual": {
    "type": "Point",
    "coordinates": [ 40.7128, -74.0060 ]
  },
  "entrega": "Pedido54",
  "fecha_registro": "2023-09-25T15:30:00Z"
})
```

En este ejemplo, se ha registrado la ubicación geoespacial de un camión en un momento específico apoyándose en el ejemplo anterior y utilizando el tipo de dato geoespacial "Point." Esto permite a la empresa llevar un seguimiento preciso de la ubicación de sus activos y mejorar la eficiencia en la gestión de entregas.

Cadena de texto (String)

Dentro de MongoDB se ofrece la utilización de datos de tipo string codificado en UTF-8, lo que significa que puede manejar información relacionada con texto y cadenas de caracteres. Esto último se logra a través de tipos de datos y operadores que permiten almacenar y consultar texto de manera eficiente en documentos de la base de datos(MongoDB, s.f.).

Para un sistema de gestión de estudiantes universitarios, se puede utilizar cadenas de texto (strings) para almacenar información relevante sobre los estudiantes. Esto puede ser útil si lo que se quiere crear son registros de admisión universitaria sin importar un mapeo relacional.

```
use('Investigacion');
db.getCollection('Universidad').insertMany([
        { "nombre": "Jorge Esteban Benavides Castro",
          "carrera": "Ing. en computación",
          "correo": "est.benavides@estudiantec.cr"},
        {"nombre": "P. F. Kvist D.",
          "carrera": "Ing. en computación",
          "correo": "pau.kvist@estudiantec.cr"}
])
```

Punto flotante (Double)

Se utiliza para representar números de punto flotante de doble precisión. Es útil para cuando se necesita almacenar números decimales con una precisión alta, como cantidades monetarias o valores científicos (MongoDB, s.f.).

Un ejemplo es en una institución bancaria. La cual puede almacenar transacciones en una fecha y hora específica. El monto es representado por el tipo de dato double para garantizar una representación precisa del valor monetario. Esto permite un registro detallado de transacciones financieras en la industria bancaria

```
use('Investigacion');
db.getCollection('Transacciones').insertMany([
          {
            "nombre": "George Steven Benson Cast",
            "saldo_bancario": 1500.50
          },
          {
            "nombre": "Adam Walker",
```

```
            "saldo_bancario": 2500.75
        }
]);
```

Arreglo (Array)

Es un tipo de estructura de datos en la que se permiten almacenar distintos elementos en cada una de sus posiciones. Cada posición del arreglo almacena un elemento distintos tipos de datos, en una secuencia ordenada de posiciones o índices y dentro de cada uno se contiene un elemento individual (MongoDB, s.f.).

```
use('Investigacion');
db.getCollection('PizzaPlanet').insertOne({
        "scrum_master": "Cristopher Chanto",
        "tareas_sprint": [
                "Levantar requerimientos",
                "Validar requerimientos con el cliente",
                "Hacer una reunion con el developer team",
                "Sprint retrospective"
        ]
})
```

Referencias

Phiri, H., & Kunda, D. (2017). *Comparative Study of NoSQL and Relational Database.* Zambia ICT Journal, 1, 1-4. https://www.researchgate.net/publication/326019759_A_Comparative_Study_of_NoSQL_and_Relational_Database

Li, Junshan & Li, Jianjun. (2018). *Research on NoSQL Database Technology*. 10.2991/icmess-18.2018.252

Gupta, A., Tyagi, S., Panwar, N., Sachdeva, S., & Saxena, U. (2017). *NoSQL databases: Critical analysis and comparison. In Proceedings of the 2017 International Conference on Computer*, Communication and Computational Sciences (pp. 293-299). https://doi.org/10.1109/IC3TSN.2017.8284494

Nayak, A., Poriya, A., & Poojary, D. (2013). *Type of NoSQL databases and its comparison with relational databases*. International Journal of Applied Information Systems, 5, 16-19. https://www.researchgate.net/publication/302557703_Article_Type_of_nosql_databases_and_its_comparison_with_relational_databases

Facultad Autodidacta. (2022). *Instalar y configurar mongodb 2022* [Video]. https://www.youtube.com/watch?v=LibtQECAR1U&ab_channel=FacultadAutodidacta

Chauhan, A. (2019*). A review on various aspects of MongoDb databases.* Int. J. Eng. Res. Sci. Technol, 8(5), 90-92. https://www.ijert.org/a-review-on-various-aspects-of-mongodb-databases

Akhtar, A. (2023). *Popularity Ranking of Database Management Systems. arXiv preprint* arXiv:2301.00847.

Valverde, V., Portalanza, N., & Mora, P. (2019). Análisis descriptivo de base de datos relacional y no relacional. Revista Atlante: Cuadernos de Educación y Desarrollo, 3.

Cookson, R. (2019). *virtual-dba*. https://www.virtualdba.com/blog/pros-and-cons-of-mongodb/

Calvo, K., Durán, J., Quirós, E., & Malinowski, E. (2017). *MongoDB: alternativas de implementar y consultar documentos. In IX Congreso Internacional de Computación y Telecomunicaciones, COMTEL, Lima* (pp. 48-49). https://www.researchgate.net/publication/323184317_MongoDB_alternativas_de_implementar_y_consultar_documentos

Abramova, V., & Bernardino, J. (2013). *NoSQL databases: MongoDB vs Cassandra. Proceedings of the International C* Conference on Computer Science and Software Engineering,* 14-22. https://web.cs.wpi.edu/~cs585/s17/StudentsPresentations/This%20Year/Week14/mongodb_vs_cassandra.pdf

Mapanga, I., & Kadebu, P. (2013). *Database management systems: A NoSQL analysis.* International Journal of Modern Communication Technologies & Research (IJMCTR),https://www.academia.edu/79216143/Database_Management_Systems_A_NoSQL_Analysis

Tampubolon, W., Reinhardt, W., Sumaryono, S., & Tobing, S. (2021*). NoSQL Standard and Approach for Geospatial Database Collection. En Seminar Nasional Geomatika* (p. 321). https://doi.org/10.24895/SNG.2020.0-0.1147

Krishnan, H., Elayidom, M. S., & Santhanakrishnan, T. (2016). *MongoDB – a comparison with NoSQL databases.* International Journal of Scientific and Engineering Research, 7, 1035-1037. https://www.researchgate.net/publication/327120267_MongoDB_-_a_comparison_with_NoSQL_databases

Battah, A., Iraqi, Y., & Damiani, E. (2021). *Blockchain-Based Reputation Systems: Implementation Challenges and Mitigation. Electronics*, 10, 289. https://doi.org/10.3390/electronics10030289

MongoDB. (s.f.). *BSON Types.* MongoDB. https://www.mongodb.com/docs/manual/reference/bson-types/