

# 17. [MongoDB] Un mismo universo de datos

## PARTE 1 – PHP + MongoDB

### Fase 1.1 — Preparar el proyecto PHP

Instalación de la extensión PHP de MongoDB

- sudo pecl install mongodb

```
ub@forense-ai:~$ sudo pecl install mongodb
```

Habilita la extensión después de la instalación:

- echo "extension=mongodb.so" | sudo tee /etc/php/8.1/cli/conf.d/20-mongodb.ini

```
ub@forense-ai:~$ echo "extension=mongodb.so" | sudo tee /etc/php/8.1/cli/conf.d/20-mongodb.ini
tee: /etc/php/8.1/cli/conf.d/20-mongodb.ini: No existe el archivo o el directorio
extension=mongodb.so
```

- echo "extension=mongodb.so" | sudo tee /etc/php/8.1/apache2/conf.d/20-mongodb.ini

```
ub@forense-ai:~$ echo "extension=mongodb.so" | sudo tee /etc/php/8.1/apache2/conf.d/20-mongodb.ini
tee: /etc/php/8.1/apache2/conf.d/20-mongodb.ini: No existe el archivo o el directorio
extension=mongodb.so
```

Reiniciar Apache

- sudo systemctl restart apache2

```
ub@forense-ai:~$ sudo systemctl restart apache2
```

Habilitar la extensión MongoDB para PHP CLI

- sudo nano /etc/php/8.3/cli/conf.d/20-mongodb.ini

```
ub@forense-ai:~$ sudo nano /etc/php/8.3/cli/conf.d/20-mongodb.ini
```

Escriba en el archivo:

- extension=mongodb.so

```
GNU nano 7.2                               /etc/php/8.3/cli/conf.d/20-mongodb.ini
extension=mongodb.so
```

Verificación de la extensión PHP MongoDB

- php -m | grep mongodb

```
ub@forense-ai:~$ php -m | grep mongodb
mongodb
```

Apache:

- sudo nano /etc/php/8.3/apache2/conf.d/20-mongodb.ini

```
ub@forense-ai:~$ sudo nano /etc/php/8.3/apache2/conf.d/20-mongodb.ini
```

Del mismo modo, escriba el siguiente contenido en el archivo:

- extension=mongodb.so

```
GNU nano 7.2          /etc/php/8.3/apache2/conf.d/20-mongodb.ini *
extension=mongodb.so
```

A continuación, reinicie Apache:

- sudo systemctl restart apache2

```
ub@forense-ai:~$ sudo systemctl restart apache2
```

Creación de la ruta PHP

- mkdir ~/mongo\_php\_lab
- cd ~/mongo\_php\_lab

```
ub@forense-ai:~$ mkdir ~/mongo_php_lab
ub@forense-ai:~$ cd ~/mongo_php_lab
ub@forense-ai:~/mongo_php_lab$
```

Instala la biblioteca PHP oficial de MongoDB

- sudo apt install composer -y

```
ub@forense-ai:~/mongo_php_lab$ sudo apt install composer -y
- composer require mongodb/mongodb
```

```
ub@forense-ai:~/mongo_php_lab$ composer require mongodb/mongodb
./composer.json has been created
```

## Fase 1.2 — Crear el script PHP

Crear un script PHP

- sudo nano mongo\_php\_lab.php

```
ub@forense-ai:~/mongo_php_lab$ sudo nano mongo_php_lab.php
```

Introduzca lo siguiente:

```
<?php
require 'vendor/autoload.php';
```

```
use MongoDB\Client;
```

```
// 1. Conexión al servidor MongoDB
$client = new Client("mongodb://localhost:27017");
```

```
// 2. Selección de base de datos y colección
$database = $client->getDatabase("planets");
$collection = $database->planets;
```

```
echo "<h1>PHP + MongoDB: Laboratorio de planetas</h1>";
```

```

// 3. Insertar varios planetas (insertMany)
echo "<h2>1. Insertando planetas...</h2>";

$insertResult = $collection->insertMany([
    [
        'name' => 'Vulcan',
        'species' => 'Vulcans',
        'affiliation' => 'Federation',
        'warp_capable' => true
    ],
    [
        'name' => "Qo'noS",
        'species' => 'Klingons',
        'affiliation' => 'Klingon Empire',
        'warp_capable' => true
    ],
    [
        'name' => 'Ferenginar',
        'species' => 'Ferengi',
        'affiliation' => 'Ferengi Alliance',
        'warp_capable' => true
    ]
]);
echo "Planetas insertados: " . $insertResult->getInsertedCount() . "<br>";

// 4. Listar todos los planetas
echo "<h2>2. Lista completa de planetas</h2>";

$cursor = $collection->find();

foreach ($cursor as $planet) {
    echo "Nombre: " . $planet['name'] .
        " | Especie: " . $planet['species'] .
        " | Alineación: " . $planet['affiliation'] .
        " | Warp: " . ($planet['warp_capable'] ? 'Sí' : 'No') .
        "<br>";
}

// 5. Mostrar solo los de la Federación
echo "<h2>3. Planetas de la Federación</h2>";

$cursorFed = $collection->find(['affiliation' => 'Federation']);

foreach ($cursorFed as $planet) {
    echo "Nombre: " . $planet['name'] . "<br>";
}

```

```

// 6. Cambiar un campo (warp_capable) para Vulcan
echo "<h2>4. Actualizando warp_capable de Vulcan a false</h2>";

$updateResult = $collection->updateOne(
    ['name' => 'Vulcan'],
    ['$set' => ['warp_capable' => false]]
);

echo "Documentos modificados: " . $updateResult->getModifiedCount() . "<br>";

// 7. Borrar un registro (Ferenginar)
echo "<h2>5. Borrando el planeta Ferenginar</h2>";

$deleteResult = $collection->deleteOne(['name' => 'Ferenginar']);

echo "Documentos eliminados: " . $deleteResult->getDeletedCount() . "<br>";

// 8. Lista final para comprobar cambios
echo "<h2>6. Lista final de planetas tras cambios</h2>";

$cursorFinal = $collection->find();

foreach ($cursorFinal as $planet) {
    echo "Nombre: " . $planet['name'] .
        " | Alineación: " . $planet['affiliation'] .
        " | Warp: " . ($planet['warp_capable'] ? 'Sí' : 'No') .
        "<br>";
}

```

GNU nano 1.2 mongo\_php\_lab.php

```

<?php
require 'vendor/autoload.php';

use MongoDB\Client;

// 1. Conexión al servidor MongoDB
$client = new Client("mongodb://localhost:27017");

// 2. Selección de base de datos y colección
$database = $client->fed_records;
$collection = $database->planets;

echo "<h1>PHP + MongoDB: Laboratorio de planetas</h1>";

// 3. Insertar varios planetas (insertMany)
echo "<h2>1. Insertando planetas...</h2>";

$insertResult = $collection->insertMany([

```

### Fase 1.3 — Ejecutar el script PHP

Opción A – Desde navegador:

1. Copia el proyecto a la carpeta del servidor
  - sudo cp -r ~/mongo\_php\_lab /var/www/html/

- sudo chown -R www-data:www-data /var/www/html/mongo\_php\_lab

```
ub@forense-ai:~/mongo_php_lab$ sudo cp -r ~/mongo_php_lab /var/www/html/
ub@forense-ai:~/mongo_php_lab$ sudo chown -R www-data:www-data /var/www/html/mongo_php_lab
```

## 2. En el navegador

- http://IP\_DEL\_SERVIDOR/mongo\_php\_lab/mongo\_php\_lab.php

### PHP + MongoDB: Laboratorio de planetas

#### 1. Insertando planetas...

Planetas insertados: 3

#### 2. Lista completa de planetas

```
Nombre: Vulcan | Especie: Vulcans | Alineación: Federation | Warp: Si
Nombre: Qo'nos | Especie: Klingons | Alineación: Klingon Empire | Warp: Si
Nombre: Vulcan | Especie: Vulcans | Alineación: Federation | Warp: Si
Nombre: Qo'nos | Especie: Klingons | Alineación: Klingon Empire | Warp: Si
Nombre: Ferenginar | Especie: Ferengi | Alineación: Ferengi Alliance | Warp: Si
```

#### 3. Planetas de la Federación

```
Nombre: Vulcan
Nombre: Vulcan
```

#### 4. Actualizando warp\_capable de Vulcan a false

Documentos modificados: 0

#### 5. Borrando el planeta Ferenginar

Documentos eliminados: 1

#### 6. Lista final de planetas tras cambios

```
Nombre: Vulcan | Alineación: Federation | Warp: No
Nombre: Qo'nos | Alineación: Klingon Empire | Warp: Si
Nombre: Vulcan | Alineación: Federation | Warp: Si
Nombre: Qo'nos | Alineación: Klingon Empire | Warp: Si
```

Opción B – Desde línea de comandos (CLI):

- php mongo\_php\_lab.php

```
ub@forense-ai:~/mongo_php_lab$ php mongo_php_lab.php
<h1>PHP + MongoDB: Laboratorio de planetas</h1><h2>1. Insertando planetas...</h2>Planetas insertados: 3<br><h2>2. Lista completa de planetas</h2>Nombre: Vulcan | Especie: Vulcans | Alineación: Federation | Warp: Si<br>Nombre: Qo'nos | Especie: Klingons | Alineación: Klingon Empire | Warp: Si<br>Nombre: Ferenginar | Especie: Ferengi | Alineación: Ferengi Alliance | Warp: Si<br><h2>3. Planetas de la Federación</h2>Nombre: Vulcan<br><h2>4. Actualizando warp_capable de Vulcan a false</h2>Documentos modificados: 1<br><h2>5. Borrando el planeta Ferenginar</h2>Documentos eliminados: 1<br><h2>6. Lista final de planetas tras cambios</h2>Nombre: Vulcan | Alineación: Federation | Warp: No
<br>Nombre: Qo'nos | Alineación: Klingon Empire | Warp: Si<br>ub@forense-ai:~/mongo_php_lab$
```

## PARTE 2 – Python + MongoDB

### Fase 2.1 — Instalar pymongo

Crear un entorno virtual

1. Primero instalar las dependencias

- sudo apt install python3-full python3-venv

```
ub@forense-ai:~/mongo_php_lab$ sudo apt install python3-full python3-venv
```

2. Crear e ingresar a un entorno virtual

- python3 -m venv myenv
- source myenv/bin/activate

```
ub@forense-ai:~/mongo_php_lab$ python3 -m venv myenv
ub@forense-ai:~/mongo_php_lab$ source myenv/bin/activate
(myenv) ub@forense-ai:~/mongo_php_lab$ 
```

### 3. Instalación en un entorno virtual

- pip3 install pymongo

```
(myenv) ub@forense-ai:~/mongo_python_lab$ pip3 install pymongo
Collecting pymongo
  Downloading pymongo-4.16.0-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (10.0 kB)
Collecting dnspython<3.0.0,>=2.6.1 (from pymongo)
  Downloading dnspython-2.8.0-py3-none-any.whl.metadata (5.7 kB)
Downloaded pymongo-4.16.0-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (1.7 MB)
  1.7/1.7 MB 42.3 MB/s eta 0:00:00
Downloaded dnspython-2.8.0-py3-none-any.whl (331 kB)
  331.1/331.1 kB 98.2 MB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.8.0 pymongo-4.16.0
```

#### Fase 2.2 — Crear el script Python

- mkdir ~/mongo\_python\_lab
- cd ~/mongo\_python\_lab

```
(myenv) ub@forense-ai:~$ mkdir ~/mongo_python_lab
(myenv) ub@forense-ai:~$ cd ~/mongo_python_lab
(myenv) ub@forense-ai:~/mongo_python_lab$ 
```

Crear un script en Python

- sudo nano mongo\_python\_lab.py

```
(myenv) ub@forense-ai:~/mongo_python_lab$ sudo nano mongo_python_lab.py
```

Contenido:

```
from pymongo import MongoClient

# 1. Conexión al servidor MongoDB
client = MongoClient("mongodb://localhost:27017")

# 2. Seleccionar base de datos y colección
db = client["fed_records"]
planets = db["planets"]

print("PYTHON + MongoDB: Laboratorio de planetas\n")

# 3. Insertar nuevos planetas
print("1) Insertando nuevos planetas...\n")

insert_result = planets.insert_many([
    {
        "name": "Andoria",
        "species": "Andorians",
        "affiliation": "Federation",
        "warp_capable": True
    },
    {
        "name": "Cardassia Prime",
```

```

        "species": "Cardassians",
        "affiliation": "Cardassian Union",
        "warp_capable": True
    }
])

print("IDs insertados:", insert_result.inserted_ids, "\n")

# 4. Listar todos los planetas
print("2) Lista completa de planetas:")

for planet in planets.find():
    print(f"- {planet['name']} ({planet['affiliation']}) | Warp: {planet.get('warp_capable', 'N/A')}")


print()

# 5. Filtrar solo la Federación
print("3) Planetas de la Federación:")

for planet in planets.find({"affiliation": "Federation"}):
    print(f"- {planet['name']}")

print()

# 6. Actualizar: poner warp_capable = True de nuevo en Vulcan (si existe)
print("4) Actualizando warp_capable de Vulcan a True...\n")

update_result = planets.update_one(
    {"name": "Vulcan"},
    {"$set": {"warp_capable": True}}
)

print("Documentos modificados:", update_result.modified_count, "\n")

# 7. Borrar un planeta concreto: Cardassia Prime
print("5) Borrando Cardassia Prime...\n")

delete_result = planets.delete_one({"name": "Cardassia Prime"})
print("Documentos eliminados:", delete_result.deleted_count, "\n")

# 8. Agregación por affiliation
print("6) Agregación: número de planetas por affiliation:\n")

pipeline = [
    {"$group": {"_id": "$affiliation", "total": {"$sum": 1}}},
    {"$sort": {"total": -1}}
]

```

```

for group in planets.aggregate(pipeline):
    print(f'{group["_id"]}: {group["total"]} planetas')

```

GNU nano 7.2 mongo\_python\_lab.py \*

```

from pymongo import MongoClient

# 1. Conexión al servidor MongoDB
client = MongoClient("mongodb://localhost:27017")

# 2. Seleccionar base de datos y colección
db = client["fed_records"]
planets = db["planets"]

print("PYTHON + MongoDB: Laboratorio de planetas\n")

# 3. Insertar nuevos planetas
print("1) Insertando nuevos planetas...\n")

```

### Fase 2.3 — Ejecutar el script Python

Ejecutar script Python

- python3 mongo\_python\_lab.py

```
(myenv) ub@forense-ai:~/mongo_python_lab$ python3 mongo_python_lab.py
PYTHON + MongoDB: Laboratorio de planetas

1) Insertando nuevos planetas...

IDs insertados: [ObjectId('6973cb2012f48739fc5d09f0'), ObjectId('6973cb2012f48739fc5d09f1')]

2) Lista completa de planetas:
- Vulcan (Federation) | Warp: False
- Qo'noS (Klingon Empire) | Warp: True
- Vulcan (Federation) | Warp: True
- Qo'noS (Klingon Empire) | Warp: True
- Andoria (Federation) | Warp: True
- Cardassia Prime (Cardassian Union) | Warp: True

3) Planetas de la Federación:
- Vulcan
- Vulcan
- Andoria

4) Actualizando warp_capable de Vulcan a True...

Documentos modificados: 1

5) Borrando Cardassia Prime...

Documentos eliminados: 1

6) Agregación: número de planetas por affiliation:

Federation: 3 planetas
Klingon Empire: 2 planetas
(myenv) ub@forense-ai:~/mongo_python_lab$ 
```

## PARTE 3 – Verificación en MongoDB Compass

### Fase 3.1 — Conectarse con Compass

The screenshot shows the MongoDB Compass interface with a connection named 'ub'. The left sidebar lists connections, and the main area shows the 'ub' database with its collections: admin, andoria\_sector, config, fed\_records, and local. The 'fed\_records' collection is expanded, showing a single document:

```
_id: ObjectId('6973c5ecc90e32e69309f3c2')
name: "Vulcan"
species: "Vulcans"
affiliation: "Federation"
warp_capable: true
```

### Fase 3.2 — Explorar la base de datos fed\_records

The screenshot shows the 'fed\_records' database with the 'planets' collection selected. The interface displays four documents:

- ```
_id: ObjectId('6973c5ecc90e32e69309f3c2')
name: "Vulcan"
species: "Vulcans"
affiliation: "Federation"
warp_capable: true
```
- ```
_id: ObjectId('6973c5ecc90e32e69309f3c3')
name: "Qo'noS"
species: "Klingons"
affiliation: "Klingon Empire"
warp_capable: true
```
- ```
_id: ObjectId('6973c71be43a3574ec0d4962')
name: "Vulcan"
species: "Vulcans"
affiliation: "Federation"
warp_capable: true
```
- ```
_id: ObjectId('6973c71be43a3574ec0d4963')
name: "Qo'noS"
species: "Klingons"
affiliation: "Klingon Empire"
```

### Fase 3.3 — Comprobar coherencia de los cambios

#### 1. Verifica que:

- Vulcan existe y tiene (último cambio lo hizo Python).warp\_capable = true

The screenshot shows the 'planets' collection with two documents. Both documents have 'warp\_capable' set to true, which is explicitly mentioned in the list item above.

- ```
_id: ObjectId('6973c5ecc90e32e69309f3c2')
name: "Vulcan"
species: "Vulcans"
affiliation: "Federation"
warp_capable: true
```
- ```
_id: ObjectId('6973c71be43a3574ec0d4962')
name: "Vulcan"
species: "Vulcans"
affiliation: "Federation"
warp_capable: true
```

- Ferenginar no está (PHP lo borró).

The screenshot shows the Compass interface with a sidebar containing connections like 'ub', 'admin', 'andoria\_sector', 'config', 'fed\_records', 'planets', and 'local'. A search bar at the top has the query '{ "name": "Ferenginar" }'. Below it are buttons for 'ADD DATA', 'UPDATE', 'DELETE', 'EXPORT DATA', and 'EXPORT CODE'. The results panel shows '25 0 - 0 of 0' and a 'No results' message with a green icon.

- Cardassia Prime no está (Python lo borró).

The screenshot shows the Compass interface with a sidebar containing connections like 'ub', 'admin', 'andoria\_sector', 'config', 'fed\_records', 'planets', and 'local'. A search bar at the top has the query '{ "name": "Cardassia Prime" }'. Below it are buttons for 'ADD DATA', 'UPDATE', 'DELETE', 'EXPORT DATA', and 'EXPORT CODE'. The results panel shows '25 0 - 0 of 0' and a 'No results' message with a green icon.

## 2. Modifica un documento desde Compass, por ejemplo:

- Edita y cambia a .Qo'no Saffiliation"Klingon Empire (Updated)"

Introduzca sus criterios de filtrado en el cuadro de búsqueda de Compass para localizar Qo'noS:

- { "name": "Qo'noS" }

The screenshot shows the Compass interface with a sidebar containing connections like 'ub', 'admin', 'andoria\_sector', 'config', 'fed\_records', 'planets', and 'local'. A search bar at the top has the query '{ "name": "Qo'noS" }'. Below it are buttons for 'ADD DATA', 'UPDATE', 'DELETE', 'EXPORT DATA', and 'EXPORT CODE'. The results panel shows '25 1 - 2 of 2' and two document cards. The first card shows the original document: '\_id: ObjectId('6973c5ecc90e32e69309f3c3'), name: "Qo'noS", species: "Klingons", affiliation: "Klingon Empire", warp\_capable: true'. The second card shows the updated document: '\_id: ObjectId('6973c71be43a3574ec0d4963'), name: "Qo'noS", species: "Klingons", affiliation: "Klingon Empire (Updated)", warp\_capable: true'.

### Modificar Saffiliation

The screenshot shows the Compass interface with a sidebar containing connections like 'ub', 'admin', 'andoria\_sector', 'config', 'fed\_records', 'planets', and 'local'. A search bar at the top has the query '{ "name": "Qo'noS" }'. Below it are buttons for 'ADD DATA', 'UPDATE', 'DELETE', 'EXPORT DATA', and 'EXPORT CODE'. The results panel shows '25 1 - 2 of 2' and two document cards. The first card shows the original document: '\_id: ObjectId('6973c5ecc90e32e69309f3c3'), name: "Qo'noS", species: "Klingons", affiliation: "Klingon Empire", warp\_capable: true'. The second card shows the updated document: '\_id: ObjectId('6973c71be43a3574ec0d4963'), name: "Qo'noS", species: "Klingons", affiliation: "Klingon Empire (Updated)", warp\_capable: true'. A modal dialog titled 'Modificar Saffiliation' is open, showing the updated document code. The code is: 1 \_id: ObjectId('6973c5ecc90e32e69309f3c3') 2 name : "Qo'noS\_" 3 species : "Klingons\_" 4 affiliation : "Klingon Empire (Updated)" 5 warp\_capable : true. The right side of the dialog shows the schema: ObjectId, String, String, String, Boolean. At the bottom, there are 'CANCEL' and 'UPDATE' buttons.

- Guarda los cambios.

The screenshot shows a MongoDB interface with the following document structure:

```

_id: ObjectId('6973c5ecc90e32e69309f3c3')
name : "Qo'noS"
species : "Klingons"
affiliation : "Klingon Empire (Updated)"
warp_capable : true

```

3. Vuelve a ejecutar el script Python: En la parte de “lista completa de planetas” verás la nueva de leída desde Python.

- cd ~/mongo\_python\_lab
- python3 mongo\_python\_lab.py

```
(myenv) ub@forense-ai:~/mongo_python_lab$ python3 mongo_python_lab.py
PYTHON + MongoDB: Laboratorio de planetas

1) Insertando nuevos planetas...

IDs insertados: [ObjectId('6973d071a7d28f69f4b60fb'), ObjectId('6973d071a7d28f69f4b60fbe')]

2) Lista completa de planetas:
- Vulcan (Federation) | Warp: True
- Qo'noS (Klingon Empire (Updated)) | Warp: True ←
- vulcan (federation) | warp: true
- Qo'noS (Klingon Empire) | Warp: True
- Andoria (Federation) | Warp: True
- Andoria (Federation) | Warp: True
- Cardassia Prime (Cardassian Union) | Warp: True
```

4. Si vuelves a ejecutar el script PHP, también verá la misma realidad.



## PHP + MongoDB: Laboratorio de planetas

### 1. Insertando planetas...

Planetas insertados: 3

### 2. Lista completa de planetas

```
Nombre: Vulcan | Especie: Vulcans | Alineación: Federation | Warp: Sí
Nombre: Qo'noS | Especie: Klingons | Alineación: Klingon Empire (Updated) | Warp: Sí ←
Nombre: Vulcan | Especie: Vulcans | Alineación: Federation | Warp: Sí
Nombre: Qo'noS | Especie: Klingons | Alineación: Klingon Empire | Warp: Sí
Nombre: Andoria | Especie: Andorians | Alineación: Federation | Warp: Sí
Nombre: Andoria | Especie: Andorians | Alineación: Federation | Warp: Sí
Nombre: Vulcan | Especie: Vulcans | Alineación: Federation | Warp: Sí
Nombre: Qo'noS | Especie: Klingons | Alineación: Klingon Empire | Warp: Sí
Nombre: Ferenginar | Especie: Ferengi | Alineación: Ferengi Alliance | Warp: Sí
```