





## 17. MongoDB - Un mismo universo de datos

 No seguro 10.188.194.49/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: 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: 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

#### 3. Planetas de la Federación

Nombre: Vulcan
Nombre: Vulcan
Nombre: Andoria
Nombre: Vulcan

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

Documentos modificados: 1

#### 5. Borrando el planeta Ferenginar

Documentos eliminados: 1

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

Nombre: Vulcan	Alineación: Federation	Warp: No
Nombre: Vulcan	Alineación: Federation	Warp: Sí
Nombre: Qo'noS	Alineación: Klingon Empire	Warp: Sí
Nombre: Andoria	Alineación: Federation	Warp: Sí
Nombre: Vulcan	Alineación: Federation	Warp: Sí
Nombre: Qo'noS	Alineación: Klingon Empire	Warp: Sí

```

py
(mi_entorno) rui11@rui-server:~/mongo_python_lab$ python3 mongo_python_lab.py
PYTHON + MongoDB: Laboratorio de planetas

1) Insertando nuevos planetas...

IDs insertados: [ObjectId('69734365d45db079c2f4eff4'), ObjectId('69734365d45db079c2f4eff5')]

2) Lista completa de planetas:
- Vulcan (Federation) | Warp: False
- 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...

```

## Práctica Guiada:

“Un mismo universo de datos: PHP, Python y MongoDB Compass”

Antes de empezar, debe estar listo:

1. **Servidor MongoDB** instalado y funcionando en el servidor Linux: `sudo systemctl status mongod`
2. **PHP** instalado (por ejemplo PHP 8.x) y un servidor web (Apache) o PHP CLI.
3. **Composer** instalado: `composer --version`
4. **Python 3** instalado: `python3 --version`
5. **MongoDB Compass** instalado en el equipo del alumno o en alguna máquina con entorno gráfico.

## PARTE 1 – PHP + MongoDB

### Fase 1.1 — Preparar el proyecto PHP

1. Crea una carpeta para el proyecto, por ejemplo:  
`mkdir ~/mongo_php_lab cd ~/mongo_php_lab`
2. Instala el **driver oficial de MongoDB para PHP** vía Composer: `composer require mongodb/mongodb`
3. Asegúrate de tener instalada y activada la **extensión nativa** de MongoDB en PHP:
  - a. Instalar extensión (si no está): `sudo pecl install mongodb`
  - b. Añadir en el `php.ini` correspondiente (CLI o Apache): `extension=mongodb.so`
  - c. Reiniciar Apache si usas servidor web: `sudo systemctl restart apache2`
  - d. Comprobar desde CLI: `php -m | grep mongodb`

Debe aparecer `mongodb` en la lista de módulos.

### Fase 1.2 — Crear el script PHP

Crea el archivo `mongo_php_lab.php` dentro de `~/mongo_php_lab` con este contenido:

```
<?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([
    [
        '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>";
}
```

![[Pasted image 20260123110421.png]]

### Fase 1.3 — Ejecutar el script PHP

#### Opción A – Desde navegador (si usas Apache):

1. Copia el proyecto a la carpeta del servidor, por ejemplo: `sudo cp -r ~/mongo_php_lab /var/www/html/`
2. En el navegador, abre:  
[http://IP\\_DEL\\_SERVIDOR/mongo\\_php\\_lab/mongo\\_php\\_lab.php](http://IP_DEL_SERVIDOR/mongo_php_lab/mongo_php_lab.php)

#### Opción B – Desde línea de comandos (CLI):

```
cd ~/mongo_php_lab  
php mongo_php_lab.php
```

(La salida será texto plano, pero funcional para comprobar inserciones y cambios.)

Cuando termine esta parte, en la colección `fed_records.planets` deberían existir al menos:

- Vulcan (`warp_capable = false` tras el update)
- Qo'noS

Y **no** debería estar Ferenginar (borrado).

## PARTE 2 – Python + MongoDB

Ahora toca hacer lo mismo, pero desde Python, y además añadir una **agregación por affiliation**.

### Fase 2.1 — Instalar pymongo

En el servidor o equipo donde ejecutes Python:

```
pip install pymongo
```

(si usas `pip3`, cámbialo por `pip3`)

### Fase 2.2 — Crear el script Python

En tu home (o donde quieras), crea para el lab:

```
mkdir ~/mongo_python_lab  
cd ~/mongo_python_lab
```

Crea el archivo `mongo_python_lab.py` con este 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")
```

### Fase 2.3 — Ejecutar el script Python

Desde la carpeta del proyecto:

```
cd ~/mongo_python_lab  
python3 mongo_python_lab.py
```

Observa:

- Insertará **Andoria** y **Cardassia Prime**.
- Luego listará todo lo que haya en `fed_records.planets` (incluyendo lo creado por PHP).
- Volverá a poner Vulcan con `warp_capable = True`.
- Borrará Cardassia Prime.
- Mostrará la agregación por `affiliation`.

Con esto ya habrán hecho la misma lógica (y algo más) que en PHP, pero desde Python.

![[Pasted image 20260123110653.png]]

## PARTE 3 – Verificación en MongoDB Compass

*(La parte chula para cerrar el círculo)*

### Fase 3.1 — Conectarse con Compass

1. Abre **MongoDB Compass**.
2. En la pantalla inicial de conexión, usa:
  - a. Si Compass está en el mismo servidor que MongoDB:
    - i. `mongodb://localhost:27017`
  - b. Si Compass está en otro equipo de la red:
    - i. `mongodb://IP_DEL_SERVIDOR:27017`
3. Pulsa **Connect**.

### Fase 3.2 — Explorar la base de datos `fed_records`

1. En la barra lateral izquierda, localiza la base de datos **`fed_records`**.
2. Entra en la colección **`planets`**.

Ahí deberías ver documentos provenientes de:

- Las inserciones del **script PHP**.
- Las inserciones del **script Python**.

Es decir, una mezcla de Vulcan, Qo'noS, Andoria, etc.

### Fase 3.3 — Comprobar coherencia de los cambios

1. Verifica que:

- a. Vulcan existe y tiene `warp_capable = true` (último cambio lo hizo Python).
  - b. Ferenginar **no** está (PHP lo borró).
  - c. Cardassia Prime **no** está (Python lo borró).
2. Modifica un documento desde Compass, por ejemplo:
  - a. Edita Qo'noS y cambia `affiliation` a `"Klingon Empire (Updated)"`.
  - b. Guarda los cambios.
3. Vuelve a ejecutar el script Python: `cd ~/mongo_python_lab python3 mongo_python_lab.py` En la parte de “lista completa de planetas” verás la nueva `affiliation` de Qo'noS leída desde Python.
4. Si vuelves a ejecutar el script PHP, también verá la misma realidad.

Con esto se ve claramente que **la fuente de verdad es MongoDB**, y PHP / Python / Compass son solo distintas formas de mirar y manipular ese mismo universo de datos.