

# BIG DATA Y PYTHON

**MÁSTER EN BIG DATA**

201020

GABRIEL MARÍN DÍAZ

**hola**

# Presentación

---

---

Yo mismo

**Nombre:** Gabriel Marín Díaz

**A qué me dedico...**

- Channel Enablement Manager en Sage
- Profesor Asociado UCM

**Perfil de LinkedIn:** <https://www.linkedin.com/in/gabrielmarindiaz/>

# **CONTENIDO**

# Contenido

---

---

## Resumen

Tema 1 – Visión General

Tema 2 – Introducción a SQL

Tema 3 – Introducción al Lenguaje Python

Tema 4 – HTML y Python

Tema 5 – Big Data y Python

Tema 6 – Procesamiento Distribuido (Spark)

**Prácticas** las realizaremos con Python, MySQL, MongoDB, Apache Spark

**Arrancamos?**

# **EJERCICIOS**

# Ejercicio

---

**OBTENER LA SERIE HISTÓRICA DE COVID 19  
EN FORMATO CSV, INTENTAR OBTENER EL  
INCREMENTO DIARIO DE CASOS  
DIAGNOSTICADOS, HOSPITALIZACIONES,  
UCI, FALLECIMIENTOS. LLEVAR EL  
RESULTADO A UN FORMATO EXCEL Y  
REPRESENTAR LOS DATOS EN GRÁFICOS**

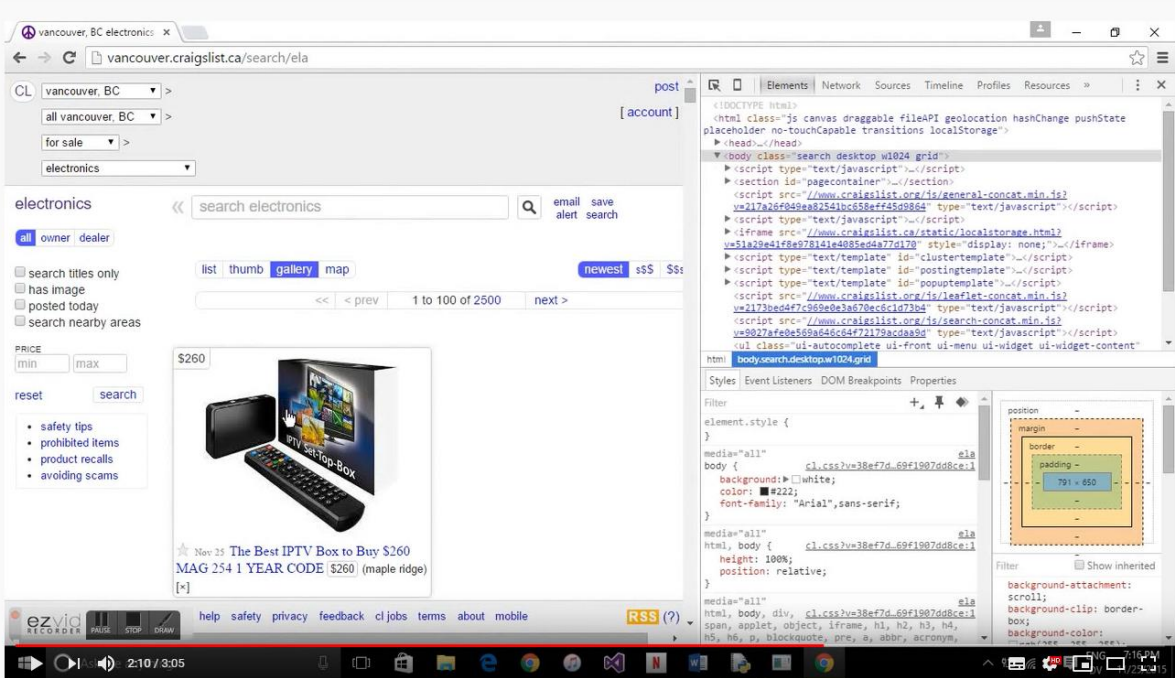


# Webscrapping

Revisar el siguiente vídeo... hacer lo mismo con una página web de compras en español.

← → ↺ 🔒 <https://www.youtube.com/watch?v=bhYulVzYRng>

≡ YouTube how to web scrape with python (selenium) 🔍



The screenshot shows a web browser window with the URL <https://www.youtube.com/watch?v=bhYulVzYRng>. The video player shows a browser window with the Craigslist website. The browser's developer tools are open, showing the 'Elements' panel with the HTML structure of the page. The video title is 'How to Web Scrape with Python (Selenium/ChromeDriver)' and it has 1239 likes and 18 comments. The video is by Charles Clayton, who has 3310 subscribers.

How to Web Scrape with Python (Selenium/ChromeDriver)

88.827 visualizaciones · 26 nov. 2015

Charles Clayton  
3310 suscriptores

SUSCRIBIRSE

Click me!

# Uso de APIs

## EJERCICIO 1

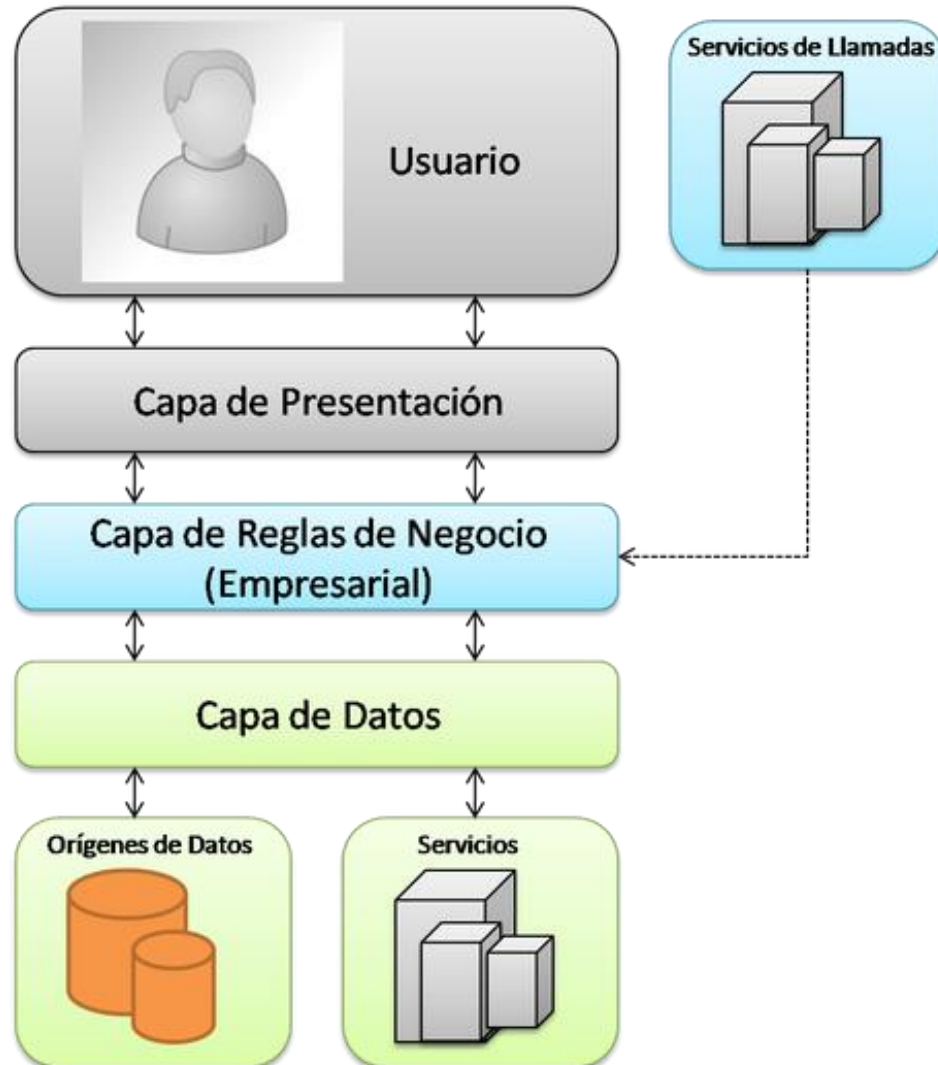
Vamos a mejorar el ejercicio de escucha en twitter...

- Introduzcamos como parámetro de escucha “COVID-19”.
- Escuchemos durante un período de tiempo razonable (30 minutos).
- El resultado obtenido lo dirigiremos a un fichero JSON (“escucha\_covid.json”).
- Utilizando la librería de Python Pandas, vamos a ir creando un estudio de los tweets generados: ubicación del tweet, texto, fuente del tweet (source), número de respuestas que ha recibido el tweet,... y todo lo que se os ocurra para obtener estadísticas, esta parte la trabajaremos esta semana.

Es importante que reviséis la estructura de un tweet, de ahí podréis obtener toda la información y que trabajéis la biblioteca Pandas para análisis de datos.

**CONTINUAMOS...**

# Arquitectura en capas



**1. Capa de Presentación:** Interacción entre el usuario y el software. Puede ser tan simple como un menú basado en líneas de comando o tan complejo como una aplicación basada en formas. Su principal función es mostrar información al usuario, interpretar los comandos de este y realizar algunas validaciones simples de los datos ingresados.

**2. Capa de Reglas de Negocio (Empresarial):** También denominada Lógica de Dominio, esta capa contiene la funcionalidad que implementa la aplicación. Involucra cálculos basados en la información dada por el usuario, datos almacenados y validaciones. Controla la ejecución de la capa de acceso a datos y servicios externos.

**3. Capa de Datos:** Esta capa contiene la lógica de comunicación con otros sistemas que llevan a cabo tareas por la aplicación. Para el caso de aplicaciones empresariales, está representado por una base de datos, que es responsable del almacenamiento persistente de información. Esta capa debe abstraer completamente a las capas superiores (negocio) del dialecto utilizado para comunicarse con los repositorios de datos (PL/SQL, Transact-SQL, etc.).

# Índice

---

- ☐ Lectura de Ficheros
- ☐ Web Scraping
- ☐ Uso de APIs
- ☐ Mongo DB
- ☐ Procesamiento Distribuido con SPARK
- ☐ Visualización de Resultados

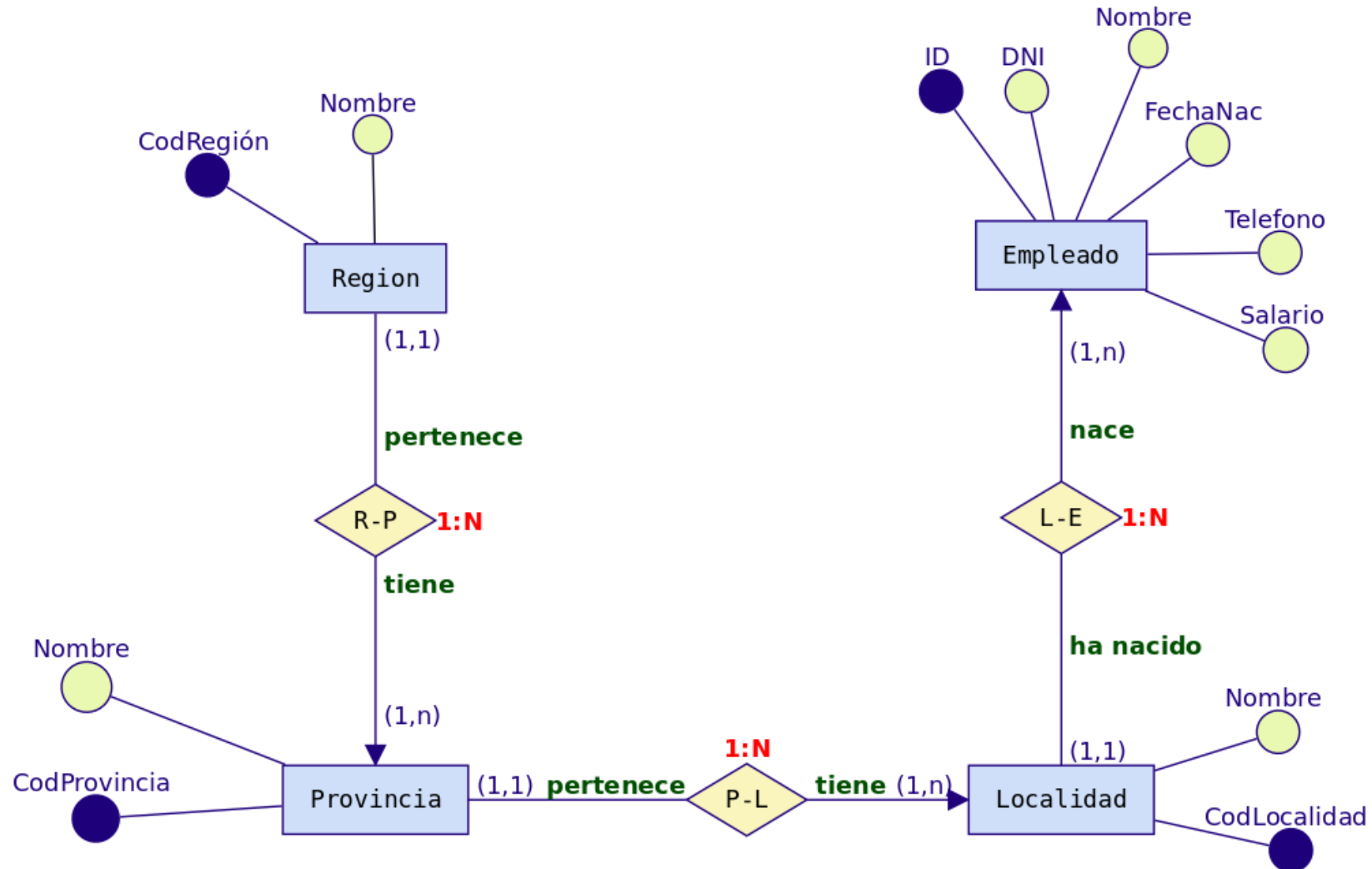
# Mongo DB

---

¿QUÉ ES **MONGO DB**?

HASTA AHORA HABÍAMOS VISTO LOS  
SISTEMAS GESTORES DE BASES DE  
DATOS RELACIONALES...

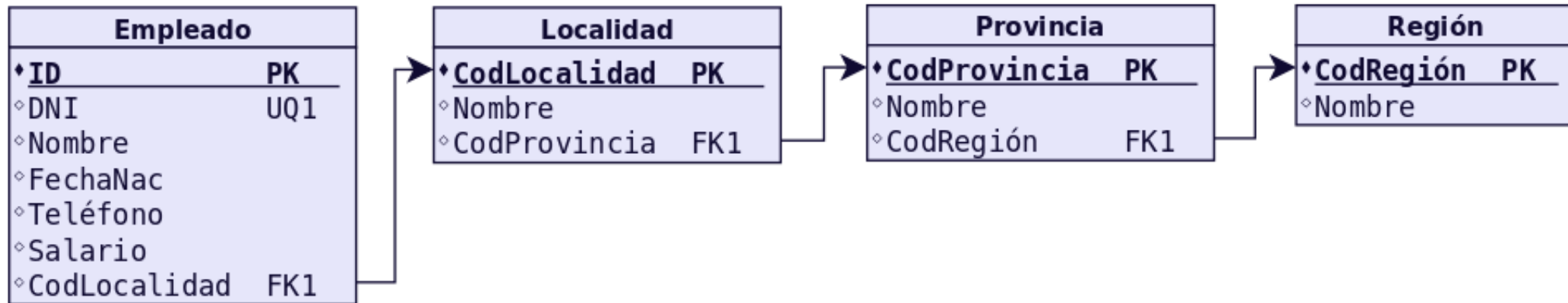
# Modelos de datos Entidad-relación





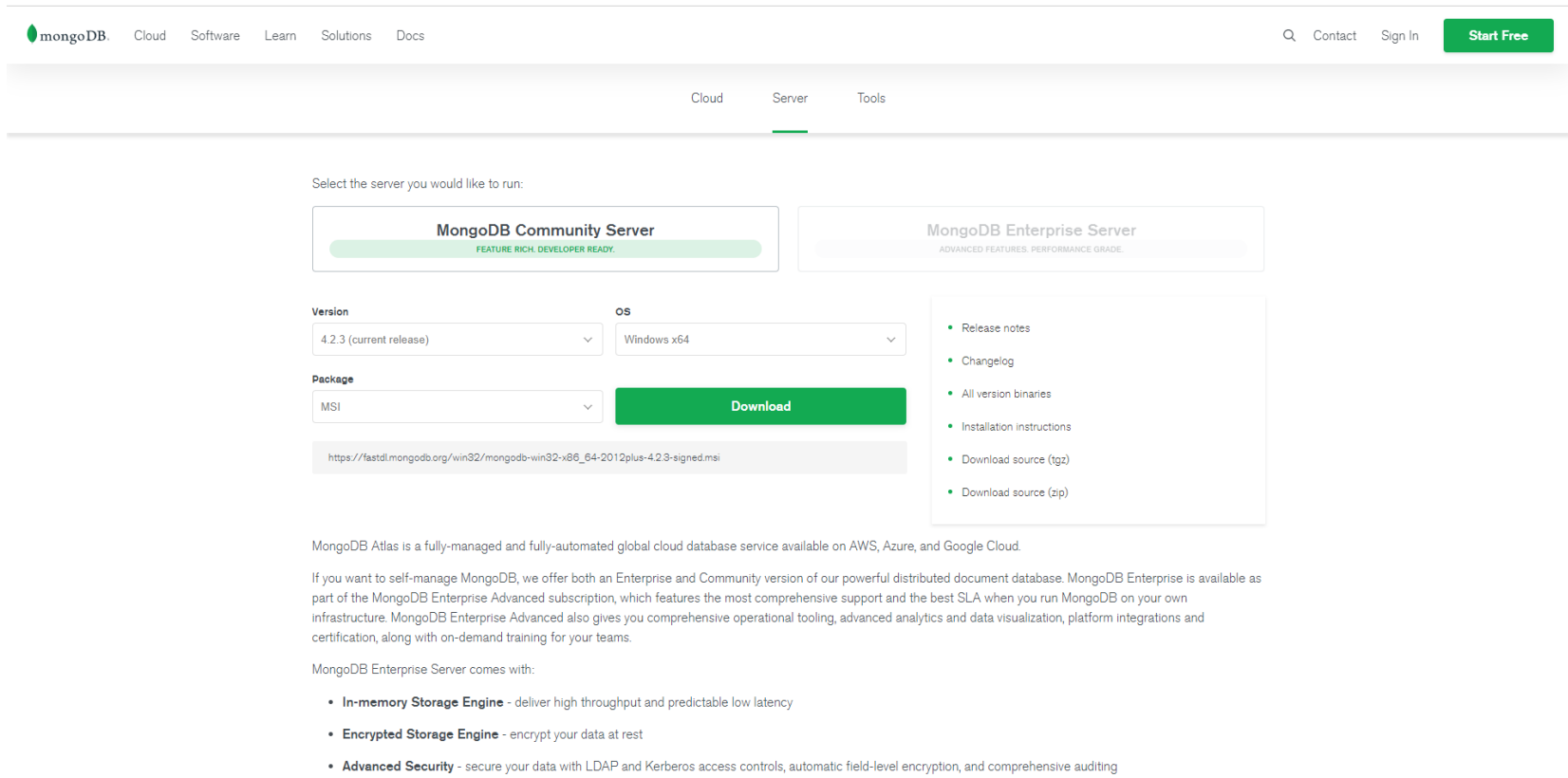
# Modelos de datos Relacional

## Ejemplo



# Mongo DB

La versión de Mongo DB que vamos a utilizar en este libro es la Community Server 4.0, disponible en la siguiente url: <https://www.mongodb.com/download-center/community>



The screenshot shows the MongoDB download center website. The navigation bar includes links for Cloud, Software, Learn, Solutions, and Docs. The main content area is titled "Select the server you would like to run:" and features two options: "MongoDB Community Server" (highlighted with a green bar) and "MongoDB Enterprise Server". Below these options, there are dropdown menus for "Version" (set to 4.2.3 (current release)) and "OS" (set to Windows x64). A "Package" dropdown is set to MSI. A green "Download" button is prominently displayed. Below the button, a URL is provided: [https://fastdl.mongodb.org/win32/mongodb-win32-x86\\_64-2012plus-4.2.3-signed.msi](https://fastdl.mongodb.org/win32/mongodb-win32-x86_64-2012plus-4.2.3-signed.msi). To the right of the download options, a list of links is provided: Release notes, Changelog, All version binaries, Installation instructions, Download source (tgz), and Download source (zip). At the bottom, there is a section for MongoDB Atlas and a description of the MongoDB Enterprise Advanced subscription.

mongoDB. Cloud Software Learn Solutions Docs

Cloud Server Tools

Select the server you would like to run:

**MongoDB Community Server**  
FEATURE RICH. DEVELOPER READY.

**MongoDB Enterprise Server**  
ADVANCED FEATURES. PERFORMANCE GRADE.

Version: 4.2.3 (current release) OS: Windows x64

Package: MSI

**Download**

[https://fastdl.mongodb.org/win32/mongodb-win32-x86\\_64-2012plus-4.2.3-signed.msi](https://fastdl.mongodb.org/win32/mongodb-win32-x86_64-2012plus-4.2.3-signed.msi)

- Release notes
- Changelog
- All version binaries
- Installation instructions
- Download source (tgz)
- Download source (zip)

MongoDB Atlas is a fully-managed and fully-automated global cloud database service available on AWS, Azure, and Google Cloud.

If you want to self-manage MongoDB, we offer both an Enterprise and Community version of our powerful distributed document database. MongoDB Enterprise is available as part of the MongoDB Enterprise Advanced subscription, which features the most comprehensive support and the best SLA when you run MongoDB on your own infrastructure. MongoDB Enterprise Advanced also gives you comprehensive operational tooling, advanced analytics and data visualization, platform integrations and certification, along with on-demand training for your teams.

MongoDB Enterprise Server comes with:

- In-memory Storage Engine** - deliver high throughput and predictable low latency
- Encrypted Storage Engine** - encrypt your data at rest
- Advanced Security** - secure your data with LDAP and Kerberos access controls, automatic field-level encryption, and comprehensive auditing

# Mongo DB

En los temas anteriores hemos visto distintas fuentes de datos, páginas web, redes sociales, ficheros CSV. Toda esta información, antes de ser analizada, debe ser almacenada adecuadamente, **Mongo DB es una BD NoSQL**.

Características:

- Orientada a documento, la noción de fila, usual en las bases de datos SQL, se sustituye aquí por documento, en particular por un documento que ya conocéis JSON. Esto nos permite representar información compleja sin necesidad de pensar en formato de tablas.
- Las colecciones, conjunto de documentos, no tienen un esquema prefijado. Es decir, un documento puede contener unas claves mientras que el documento siguiente, dentro de la misma colección, puede tener una estructura completamente diferente.
- V de variedad de formato de datos, pensemos en una tienda que define un catálogo de ropa, cada una con una característica distinta, si esto es así en un SGBD relacional nos obligaría a definir una tabla por cada tipo de ropa. En el caso de Mongo DB podemos utilizar el catálogo y asociarle distintos objetos JSON.
- V de volumen, posibilidad de almacenar grandes cantidades de datos en un entorno altamente escalable, clúster de ordenadores, los datos de la misma colección pueden estar repartidos por todo el clúster.
- V de velocidad, la velocidad de lectura de datos se logra gracias a que las búsquedas se hacen en paralelo en todos los ordenadores, lo que nos proporciona escalabilidad en el tiempo de acceso.

# Mongo DB

---

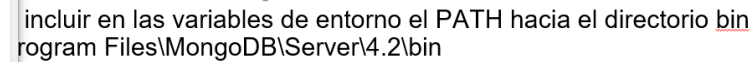
Se puede descargar e instalar el fichero (msi), indicar que se desea instalar la versión completa, prácticamente todos los componentes son interesantes y no ocupan demasiado espacio.

En un momento Mongo nos preguntará si deseamos instalar la aplicación como un servicio, si decimos que sí, el servidor Mongo se iniciará por defecto de forma automática cada vez que arranquemos Windows. De esta forma no tendremos que preocuparnos de arrancar el servicio cada vez que queramos utilizar Mongo DB.

Por otro lado, es importante que especifiquemos los directorios de datos y de log,

- C:\Program Files\MongoDB\Server\4.2\data
- C:\Program Files\MongoDB\Server\4.2\log

\_\_\_\_\_



# Mongo DB

Los comandos mongo, mongod y mongoimport deberán estar disponibles desde una ventana de comandos...

```
Seleccionar Símbolo del sistema

C:\Users\gabriel.marin>mongo
MongoDB shell version v4.2.2
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("d11880a6-cb24-4058-889c-5477360d15c3") }
MongoDB server version: 4.2.2
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
Server has startup warnings:
2020-03-21T11:13:31.864+0100 I CONTROL [initandlisten]
2020-03-21T11:13:31.864+0100 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-03-21T11:13:31.864+0100 I CONTROL [initandlisten] **           Read and write access to data and configuration is unrestricted.
2020-03-21T11:13:31.864+0100 I CONTROL [initandlisten]
MongoDB Enterprise > exit
bye

C:\Users\gabriel.marin>mongod
2020-03-21T14:09:01.599+0100 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-03-21T14:09:02.146+0100 I CONTROL [initandlisten] MongoDB starting : pid=7728 port=27017 dbpath=C:\data\db\ 64-bit host=MB06CMRGMARIN
2020-03-21T14:09:02.146+0100 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-03-21T14:09:02.147+0100 I CONTROL [initandlisten] db version v4.2.2
2020-03-21T14:09:02.147+0100 I CONTROL [initandlisten] git version: a0bbbff6ada159e19298d37946ac8dc4b497eadf
2020-03-21T14:09:02.147+0100 I CONTROL [initandlisten] allocator: tcmalloc
2020-03-21T14:09:02.148+0100 I CONTROL [initandlisten] modules: enterprise
2020-03-21T14:09:02.148+0100 I CONTROL [initandlisten] build environment:
2020-03-21T14:09:02.148+0100 I CONTROL [initandlisten]   distmod: windows-64
2020-03-21T14:09:02.148+0100 I CONTROL [initandlisten]   distarch: x86_64
2020-03-21T14:09:02.149+0100 I CONTROL [initandlisten]   target_arch: x86_64
2020-03-21T14:09:02.149+0100 I CONTROL [initandlisten] options: {}
2020-03-21T14:09:02.150+0100 I STORAGE [initandlisten] exception in initAndListen: NonExistentPath: Data directory C:\data\db\ not found., terminating
2020-03-21T14:09:02.150+0100 I NETWORK [initandlisten] shutdown: going to close listening sockets...
2020-03-21T14:09:02.151+0100 I - [initandlisten] Stopping further Flow Control ticket acquisitions.
2020-03-21T14:09:02.151+0100 I CONTROL [initandlisten] now exiting
2020-03-21T14:09:02.151+0100 I CONTROL [initandlisten] shutting down with code:100

C:\Users\gabriel.marin>mongoimport
2020-03-21T14:09:07.581+0100 no collection specified
2020-03-21T14:09:07.864+0100 using filename '' as collection
2020-03-21T14:09:07.865+0100 error validating settings: invalid collection name: collection name cannot be an empty string

C:\Users\gabriel.marin>
```

# Mongo DB

---

# VAMOS A VER UN EJEMPLO

# Mongo DB

Una vez instalado vamos a ejecutar la instrucción **mongod**, nos abrirá el server...

```
Símbolo del sistema - mongod
Microsoft Windows [Versión 10.0.17134.1365]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\gabriel.marin>mongod
2020-03-21T07:02:42.541-0800 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-03-21T07:02:42.545-0800 I CONTROL [initandlisten] MongoDB starting : pid=25364 port=27017 dbpath=C:\data\db\ 64-bit host=MB06CMRGMARIN
2020-03-21T07:02:42.545-0800 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-03-21T07:02:42.545-0800 I CONTROL [initandlisten] db version v4.2.2
2020-03-21T07:02:42.545-0800 I CONTROL [initandlisten] git version: a0bbbff6ada159e19298d37946ac8dc4b497eadf
2020-03-21T07:02:42.545-0800 I CONTROL [initandlisten] allocator: tcmalloc
2020-03-21T07:02:42.545-0800 I CONTROL [initandlisten] modules: enterprise
2020-03-21T07:02:42.545-0800 I CONTROL [initandlisten] build environment:
2020-03-21T07:02:42.545-0800 I CONTROL [initandlisten]   distmod: windows-64
2020-03-21T07:02:42.545-0800 I CONTROL [initandlisten]   distarch: x86_64
2020-03-21T07:02:42.546-0800 I CONTROL [initandlisten]   target_arch: x86_64
2020-03-21T07:02:42.546-0800 I CONTROL [initandlisten] options: {}
2020-03-21T07:02:42.547-0800 I STORAGE [initandlisten] exception in initAndListen: NonExistentPath: Data directory C:\data\db\ not found., terminating
2020-03-21T07:02:42.547-0800 I NETWORK [initandlisten] shutdown: going to close listening sockets...
2020-03-21T07:02:42.547-0800 I - [initandlisten] Stopping further Flow Control ticket acquisitions.
2020-03-21T07:02:42.547-0800 I CONTROL [initandlisten] now exiting
2020-03-21T07:02:42.547-0800 I CONTROL [initandlisten] shutting down with code:100

C:\Users\gabriel.marin>mongod
2020-03-21T16:04:12.851+0100 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-03-21T16:04:13.325+0100 I CONTROL [initandlisten] MongoDB starting : pid=24120 port=27017 dbpath=C:\data\db\ 64-bit host=MB06CMRGMARIN
2020-03-21T16:04:13.325+0100 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-03-21T16:04:13.325+0100 I CONTROL [initandlisten] db version v4.2.2
2020-03-21T16:04:13.325+0100 I CONTROL [initandlisten] git version: a0bbbff6ada159e19298d37946ac8dc4b497eadf
2020-03-21T16:04:13.325+0100 I CONTROL [initandlisten] allocator: tcmalloc
2020-03-21T16:04:13.325+0100 I CONTROL [initandlisten] modules: enterprise
2020-03-21T16:04:13.325+0100 I CONTROL [initandlisten] build environment:
2020-03-21T16:04:13.325+0100 I CONTROL [initandlisten]   distmod: windows-64
2020-03-21T16:04:13.325+0100 I CONTROL [initandlisten]   distarch: x86_64
2020-03-21T16:04:13.326+0100 I CONTROL [initandlisten]   target_arch: x86_64
2020-03-21T16:04:13.326+0100 I CONTROL [initandlisten] options: {}
2020-03-21T16:04:13.330+0100 I STORAGE [initandlisten] wiredtiger open config: create,cache_size=7488M,cache_overflow=(file_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabl
ed=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_progress,checkpoint_progress],
2020-03-21T16:04:13.503+0100 I STORAGE [initandlisten] WiredTiger message [1584803053:503067][24120:140724178734160], txn-recover: Set global recovery timestamp: (0,0)
2020-03-21T16:04:13.523+0100 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2020-03-21T16:04:13.545+0100 I STORAGE [initandlisten] Timestamp monitor starting
2020-03-21T16:04:13.560+0100 I CONTROL [initandlisten]
2020-03-21T16:04:13.560+0100 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-03-21T16:04:13.560+0100 I CONTROL [initandlisten] **   Read and write access to data and configuration is unrestricted.
2020-03-21T16:04:13.560+0100 I CONTROL [initandlisten]
2020-03-21T16:04:13.560+0100 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2020-03-21T16:04:13.560+0100 I CONTROL [initandlisten] **   Remote systems will be unable to connect to this server.
2020-03-21T16:04:13.560+0100 I CONTROL [initandlisten] **   Start the server with --bind_ip <address> to specify which IP
2020-03-21T16:04:13.561+0100 I CONTROL [initandlisten] **   addresses it should serve responses from, or with --bind_ip_all to
2020-03-21T16:04:13.561+0100 I CONTROL [initandlisten] **   bind to all interfaces. If this behavior is desired, start the
2020-03-21T16:04:13.561+0100 I CONTROL [initandlisten] **   server with --bind_ip 127.0.0.1 to disable this warning.
2020-03-21T16:04:13.561+0100 I CONTROL [initandlisten]
2020-03-21T16:04:13.562+0100 I STORAGE [initandlisten] createCollection: admin.system.version with provided UUID: f22daefd-2ed2-48a0-8669-0a60129aae6 and options: { uuid: UUID("f22daefd-2ed2-48a0-8669-0a60129aae6") }
2020-03-21T16:04:13.583+0100 I INDEX [initandlisten] index build: done building index _id_ on ns admin.system.version
2020-03-21T16:04:13.584+0100 I SHARDING [initandlisten] Marking collection admin.system.version as collection version: <unsharded>
2020-03-21T16:04:13.584+0100 I COMMAND [initandlisten] setting featureCompatibilityVersion to 4.2
2020-03-21T16:04:13.586+0100 I SHARDING [initandlisten] Marking collection local.system.replset as collection version: <unsharded>
2020-03-21T16:04:13.586+0100 I STORAGE [initandlisten] Flow Control is enabled on this deployment.
2020-03-21T16:04:13.586+0100 I SHARDING [initandlisten] Marking collection admin.system.roles as collection version: <unsharded>
2020-03-21T16:04:13.587+0100 I STORAGE [initandlisten] createCollection: local.startup_log with generated UUID: 3500afde-af77-4b69-925c-ab22de77cccd and options: { capped: true, size: 10485760 }
2020-03-21T16:04:13.607+0100 I INDEX [initandlisten] index build: done building index _id_ on ns local.startup_log
2020-03-21T16:04:13.608+0100 I SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
2020-03-21T16:04:14.375+0100 W FTDC [initandlisten] Failed to initialize Performance Counters for FTDC: WindowsPdhError: PdhExpandCounterPathW failed with 'El objeto especificado no se encontró en el equipo.' for counter '\\Memory\\Av
ailable Bytes'
```



# Mongo DB

Y podemos utilizar la consola para hacer nuestra primera práctica...

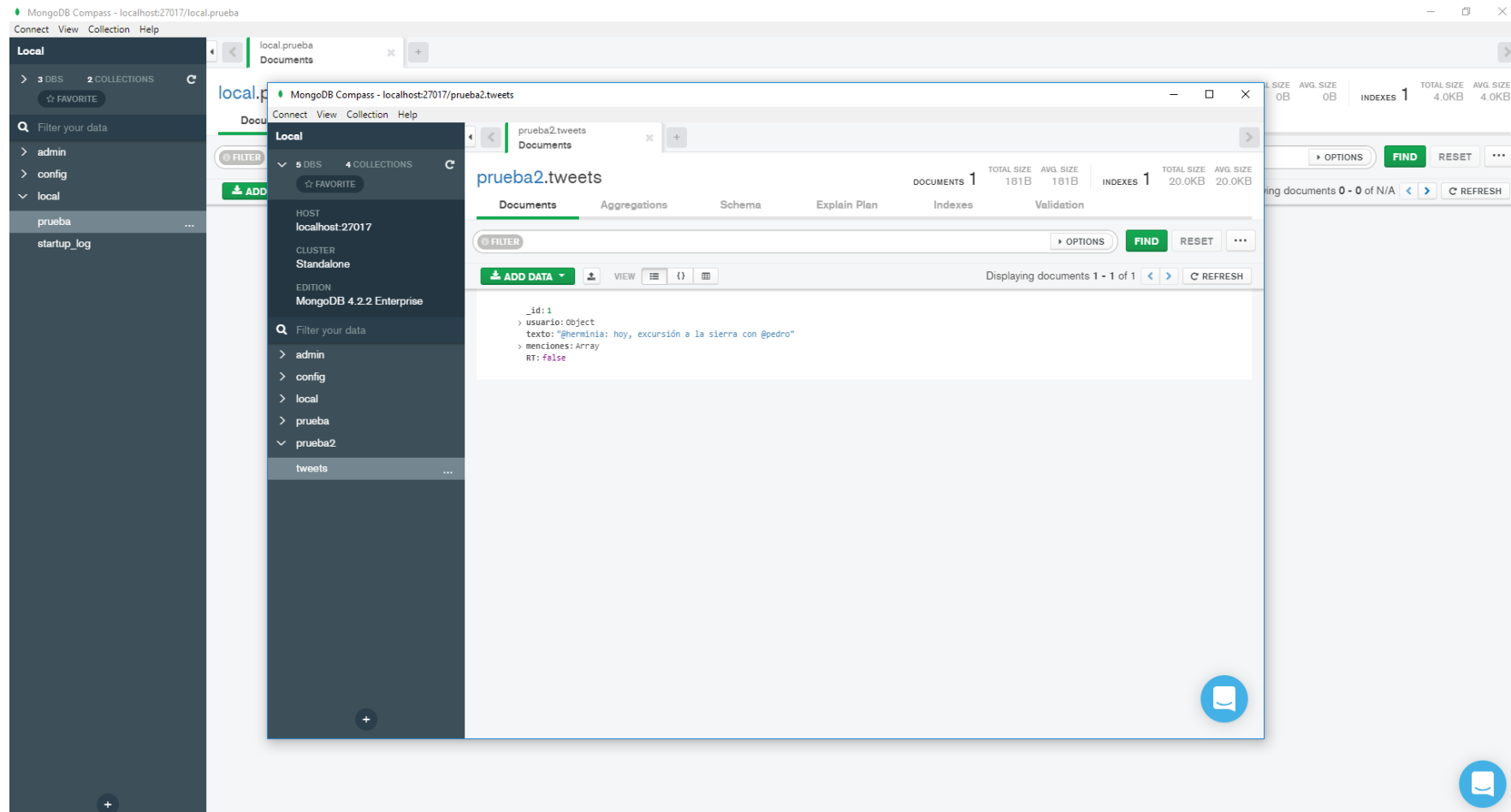
```
ca Símbolo del sistema - mongo
Microsoft Windows [Versión 10.0.17134.1365]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\gabriel.marin>mongo
MongoDB shell version v4.2.2
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("a97a79e0-7de5-4905-9bc3-6cf0d8c78588") }
MongoDB server version: 4.2.2
Server has startup warnings:
2020-03-21T11:13:31.864+0100 I  CONTROL  [initandlisten]
2020-03-21T11:13:31.864+0100 I  CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-03-21T11:13:31.864+0100 I  CONTROL  [initandlisten] **           Read and write access to data and configuration is unrestricted.
2020-03-21T11:13:31.864+0100 I  CONTROL  [initandlisten]
MongoDB Enterprise > show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
prueba 0.000GB
MongoDB Enterprise > use prueba2
switched to db prueba2
MongoDB Enterprise > db.tweets.insertOne({_id: 1, usuario: {nick:"Gabriel", seguidores: 1400}, texto: "@herminia: hoy, excursión a la sierra con @pedro", menciones: ["herminia", "pedro"], RT: false})
{ "acknowledged" : true, "insertedId" : 1 }
MongoDB Enterprise >
```

```
db.tweets.insertOne({_id:1, ususario: {nick: "Gabriel", seguidores:1400}, texto:" @herminia: hoy, excursión a la sierra con @pedro",
menciones: ["herminia", "pedro"], RT: false})
```

# Mongo DB

Y visualizar el contenido de lo creado...



# Mongo DB

Vamos a hacer esto mismo desde Python...

Lo primero que deberemos hacer es importar la biblioteca **pymongo**, para ello abriremos **spyder** y en la consola introduciremos la instrucción **pip install pymongo**.

```
Terminal de IPython
Terminal 1/A
In [2]: pip install pymongo
Collecting pymongo
  Downloading https://files.pythonhosted.org/packages/5b/df/d0f82279467c72dd0c8cd1908e04a7fb56145a5d222704722e2593af79f1/pymongo-3.10.1-cp37-cp37m-win_amd64.whl (354kB)
Installing collected packages: pymongo
Successfully installed pymongo-3.10.1
Note: you may need to restart the kernel to use updated packages.

In [3]: from pymongo import MongoClient

In [4]: client = MongoClient('localhost', 27017)

In [5]: from pymongo import MongoClient

In [6]: client = MongoClient('localhost', 27017)

In [7]: db = client['prueba']

In [8]: tweets = db['tweets']

In [9]: tweet = {'_id': 2, 'usuario': {'nick': "herminia", 'seguidores': 5320},
File "<ipython-input-9-835d22f1080b>", line 1
    tweet = {'_id': 2, 'usuario': {'nick': "herminia", 'seguidores': 5320},
SyntaxError: unexpected EOF while parsing

In [10]:

In [10]: tweet = {'_id': 2, 'usuario': {'nick': "herminia", 'seguidores': 5320}, 'texto': "RT:
@herminia: hoy, excursión a la sierra con @pedro", 'menciones': ["herminia", "pedro"], 'RT': True,
'origen': 1}
```

# Mongo DB

Y el siguiente código:

```
from pymongo import MongoClient
```

```
client = MongoClient('localhost', 27017)
```

```
db = client ['prueba2']
```

```
tweets = db ['tweets']
```

```
tweet = {'_id': 2, 'usuario': {'nick': "herminia", 'seguidores': 5320}, 'texto':"RT: @herminia: hoy, excursión a la sierra  
con @pedro", 'menciones': ["herminia", "pedro"], 'RT': True, 'origen': 1}
```

```
insertado = tweets.insert_one(tweet)
```

```
print(insertado.inserted_id)
```

Podemos ir a continuación a MongoDB Compass y analizar si se ha insertado o no el tweet en la base de datos prueba, que habíamos creado desde la consola.

# Mongo DB

## EJERCICIO 1

Tras ejecutar el código anterior tenemos ya dos tweets en la colección prueba. Para tener un conjunto mayor y utilizarlo en el resto de ejercicios, vamos a generar de forma aleatoria 100 tweets al azar desde Python. El código para lograr este programa comienza estableciendo la conexión, seleccionando la base de datos y la colección y asegurándose de que la colección está vacía (drop).

Además de la biblioteca pymongo para conectar con MongoDB, se utilizará la biblioteca random, para generar los valores aleatorios, y str para las operaciones que permiten generar el texto del tweet. El marcador RT se elige al azar, excepto para el primer tweet, que lógicamente no puede ser retweeteado. Si RT es true, se añade la clave origen con el \_id del de uno cualquiera de los tweets anteriores, simulando que ese tweet anterior es el que se está reemitiendo.

# Mongo DB

---

## EJERCICIO 1

# SOLUCIÓN...

Vamos a ver si lo podemos hacer aquí...

Click me!

# Mongo DB

## CONSULTAS SIMPLES

Ahora tenemos que ser capaces de extraer información...

Vamos a utilizar los operadores *find*, pero antes vamos a ver cómo son los operadores de MongoDB y cómo se pueden utilizar mediante ejemplos en la consola, ojo antes hay que hacer un use de la colección.

*use prueba2*

La forma más simple de ver el contenido de una colección (documentos)...

*db.tweets.find ()* – *Esto nos mostrará los 20 primeros elementos de twitter.*

Si la colección tiene más de 20 documentos, podemos teclear *it* para ver los 20 siguientes...

# Mongo DB

## CONSULTAS SIMPLES

El formato en que se muestran los documentos no es demasiado agradable, en el caso de JSON puede ser de utilidad emplear *pretty*.

```
db.tweets.find().pretty()
```

Podemos también saltarnos los primeros documentos con *skip*, para ver todos los documentos empezando por el segundo.

```
db.tweets.find().skip(1).pretty()
```

También podemos utilizar *limit()* para mostrar los n primeros documentos.

```
db.tweets.find().skip(1).limit(2).pretty()
```



# Mongo DB

## CONSULTAS SIMPLES

El orden en que se muestran los documentos es el de inserción, para mostrarlo en otro orden lo mejor es emplear la función `sort()`. Para comenzar con los tweets ordenados por el de mayor id, se puede utilizar...

```
db.tweets.find().sort({'_id':-1}).pretty()
```

Si queremos ordenar por el número de seguidores, de mayor a menor.

```
db.tweets.find().sort({'usuario.seguidores':-1})
```

`sort()` también permite que se ordene por variables claves.

```
db.tweets.find().sort({'usuario.seguidores':-1, '_id':-1})
```

# Mongo DB

## CONSULTAS SIMPLES

Si queremos ver los datos de cada tweet, excepto los datos del usuario, podríamos utilizar...

```
db.tweets.find({}, {usuario:0})
```

Si queremos ver el id del tweet y el texto...

```
db.tweets.find({}, {_id:1, text:1})
```

Si no queremos ver el \_id tendríamos que hacer...

```
db.tweets.find({}, {text:1, _id:0})
```

# Mongo DB

## CONSULTAS SIMPLES

La utilidad principal del find es el uso de la cláusula where en SQL... Para ver los textos de tweets que son retweets.

```
db.tweets.find({RT: true}, {text:1, _id:0})
```

El primer argumento selecciona los tweets con el indicador *RT* a true, mientras el segundo indica que de los documentos que verifican esto solo se debe mostrar el campo *text*.

Otros operadores de comparación y lógicos...



Click me!

# Mongo DB

## ARRAYS

Si queremos ver la clave mentions de aquellos tweets que mencionan a gabrimarin, habrá que escribir...

```
db.tweets.find({mentions: 'gabrimarin'}, {mentions:1})
```

Esto nos permite seleccionar documentos cuyos arrays contienen elementos concretos de forma sencilla. Los tweets que no mencionan a gabrimarin...

```
db.tweets.find({mentions: {$ne: 'gabrimarin'}}, {mentions:1})
```

La función `$exists` selecciona los documentos que tienen este campo... ordena por la clave origen, pero si no existe, muestra a estos los primeros.

```
db.tweets.find().sort({origen:1})
```

```
db.tweets.find({origen:{$exists: 1}}).sort({origen:1})
```

**OTRO EJEMPLO**

# Mongo DB

Y el siguiente código:

```
use astronomia
```

```
db.estelar.insert({_id:1, nombre:"Sirio", tipo: "estrella", espectro: "A1V"})
```

```
db.estelar.insert({_id:2, nombre:"Saturno", tipo: "planeta"})
```

```
db.estelar.insert({_id:3, nombre:"Plutón", tipo: "planeta"})
```

Ahora queremos cambiar el tipo de Plutón a "Planeta enano". Podemos hacer:

```
db.estelar.update({_id:3}, {tipo: "Planeta enano"})
```

```
db.estelar.find({_id:3})
```

Este tipo de actualizaciones utilizando la consola, parece que no son muy útiles, lo mejor es hacerlo desde Python...



Click me!

**OTRO EJEMPLO**

# Mongo DB

---

Abrimos la línea de comandos y ejecutamos:

# Lanzamos el servidor desde una ventana de consola (cmd), si no lo estaba ya y ejecutamos,

```
>>Mongod
```

En la ventana de consola nos situamos en el directorio donde hemos descargado los Tweets de escucha en RRSS (recordad el otro día...).

A continuación ejecutamos la siguiente línea de comandos para crear una base de datos y una colección:

```
>> mongoimport --db=CargarTweets --collection=final --file=Tweets.json
```

Veamos ahora en MongoDB Compass qué ha pasado...



Click me!



# **EJERCICIOS CON MONGODB**

# Ejercicio

---

## EJERCICIO 1

**COMO SABEMOS REALIZAR ESCUCHAS EN  
TWITTER, VAMOS A INTENTAR ESCUCHAR  
CUALQUIER PALABRA O CONJUNTO DE PALABRAS  
EN TWITTER Y LAS LLEVAMOS A MONGODB,  
PODÉIS HACERLO COMO LO HE HECHO YO,  
DESCARGANDO EL JSON O BIEN DIRECTAMENTE  
EN MONGODB**

# Ejercicio

## EJERCICIO 2

Ejercicio 2. A partir de los datos expresados en la siguiente tabla:

CUSTOMERNUMBER	CUSTOMERNAME	CONTACTLASTNAME	CONTACTFIRSTNAME	PHONE	ADDRESSLINE1
103	Atelier graphique	Schmitt	Carine	40.32.2555	54, rue Royale
112	Signal Gift Stores	King	Sue	7025551838	8489 Strong St.
114	Australian Collectors, Co.	Ferguson	Peter	03 9520 4555	636 St Kilda Road
119	La Rochelle Gifts	Labrune	Janine	40.67.8555	67, rue des Cinquante Otages

ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	SALESREPEMPLYEENUMBER	CREDITLIMIT
NULL	Nantes	NULL	44000	France	1370	21000
NULL	Las Vegas	NV	83030	USA	1166	71800
Level 3	Melbourne	Victoria	3004	Australia	1611	117300
NULL	Nantes	NULL	44000	France	1370	118200
NULL	Stavern	NULL	4110	Norway	1504	81700

- Crea la base de datos “Sales” y la colección “Customers” en MongoDB
- Realiza una query para que devuelva la dirección completa ("ADDRESSLINE1", "ADDRESSLINE2", "CITY", "STATE", "POSTALCODE") de todos los documentos de la colección
- Actualiza la clave “CREDITLIMIT” de todos los documentos de la colección, atendiendo a la siguiente regla: Si el límite de crédito es menor o igual de 21.000€ lo incrementaremos en 100€. Si es superior a 21.000€ e inferior a 75.000€ lo incrementaremos un 200€. Si es igual o mayor a 75.000€ lo decrementaremos en 100€.

# Ejercicio

## EJERCICIO 3

Ejercicio 3. Diseña una base de datos en MongoDB para representar la información de un nuevo sistema solar.

Cuestiones:

a) Crea la base de datos astronomía y la colección planetas. Inserta, además, la siguiente información:

Id	Nombre	Masa	Volumen	Composición	Perihelio	Afelio
1	P111	1.9	3	H	0.5	
2	P222	3.1	4.5	O	0.07	1.7
3	S111	0.2				
4	S222	0.3	1.1			
5	S333		0.02			
6	C111	0.05		H	0.01	25
2	P333	3.8	4.9	He		

S111, S222 y S333 son satélites de P222. Añeade al documento que contenga la información de P222 una clave, llamada satélites, cuyo valor sea un array con la información de cada uno de los satélites.

b) Muestra por pantalla únicamente los nombres de aquellos objetos que tengan en la composición H o N o bien su masa sea mayor que 3.2. La consulta, que solo se realizará una vez, debe ser eficiente.

c) Actualiza el nombre de C111 a Z111.

Nota: perihelio y afelio son la distancia más corta y más larga en una órbita, respectivamente, a una estrella.

# Ejercicio

## EJERCICIO 4

Ejercicio 4. Una compañía telefónica necesita almacenar información de sus clientes. Diseña las siguientes operaciones en MongoDB:

- a) Crea la base de datos *movilmongo* y la colección *clientes*. Inserta, además, la siguiente información:

dni	nombre	telefono	direccion	edad
111	pepe	1111	Madrid	
222	ana	2222	Barcelona	27
333	juan	3333		
444	maria	4444		38

- b) Todos los *dni* y *nombre* de cada documento en la base de datos (sin visualizar el campo "id" ni el documento entero).
- c) Todos los documentos cuyos teléfonos sean 1111 o 4444, o bien su *nombre* sea *maria* o bien su *edad* sea menor o igual que 30.
- d) Pepe tiene dos teléfonos más (uno para casa: 1112 y otro para el trabajo: 1113). Inserta en el documento de *pepe* un campo llamado *adicionales* que contenga un array de documentos embebidos. Cada documento embebido tendrá dos claves: *casa* y *trabajo*.
- e) Añade un campo *antigüedad* para *ana* inicializado a cero. A continuación, increméntalo en 1.

# Ejercicio

## EJERCICIO 5

Ejercicio 5. En el contexto de una base datos en mongoDB para gestionar una academia:

1. Realiza las siguientes operaciones:

- Crea una base de datos llamada *academia* y una colección llamada *alumnos*.
- Añade la información de dos alumnos de nombre *Ana* y *Juan* cuyas edades son 19 y 18.
- Añade una clave llamada *lengua* a la información de *Ana* que sea un array cuyos valores sean 6, 7 y 9. Haz lo mismo con Juan para añadir la información de *lengua* (con notas 6, 7 y 8) y *matemáticas* (con notas 6, 7 y 8).

2. Realiza las siguientes consultas:

- Sólo el *nombre* de todos los alumnos que han sacado un 8 en un examen de *lengua*.
- Sólo el *nombre* de todos los alumnos que han sacado un 8 en algún examen.

3. Añade un 10 a las notas de *lengua* de Ana.

4. Ana se ha dado de baja en la academia. Sustituye su información (con una única instrucción) por la de *Javier* de *edad* 22 y con notas en *lengua* 5 y 6.

# Ejercicio

## EJERCICIO 6

Ejercicio 6. Científicos españoles han encontrado vida en un exoplaneta. Necesitan almacenar alguna información sobre las especies recién descubiertas. Diseña las siguientes operaciones en MongoDB:

- a) Crea la base de datos `exoplaneta` y la colección `especies`. Inserta, además, la siguiente información:

Código de especie	Especie más parecida en la Tierra	Tamaño	Hábitat	Esperanza de vida
111	hipopótamo	15	Marino	
222	abeja	2		2
333	gato	33	Marino	1400
444	caballo			328

Nota: puedes utilizar las siguientes abreviaturas: *codigo* por “Código de especie”, *parecida* por “Especie más parecida en la Tierra” y *esperanza* por “Esperanza de vida”.

- b) Todos los códigos de especie y la especie más parecida en la Tierra de cada documento en la base de datos (sin visualizar el campo `"id"` ni el documento entero).
- c) Todos los códigos de animales que sean parecidos a hipopótamos o caballos, o bien sean marinos o bien su esperanza de vida sea mayor o igual que 500.
- d) La especie 111 se parece a dos animales más (rinoceronte y jirafa en grado 0.8 y 0.9 respectivamente). Inserta en el documento de 111 un campo llamado `adicionales` que contenga un array de documentos embebidos. Cada documento embebido tendrá dos claves: `rinoceronte` y `jirafa` y sus correspondientes grados como valores.

**REALIZAD LOS EJERCICIOS UTILIZANDO PYTHON  
CON MONGODB Y SE PRESENTARÁN EN LA CLASE  
DEL JUEVES... DEJAMOS HASTA EL LUNES  
PRÓXIMO...**



# !Muchas Gracias!

---

GABRIEL MARÍN DÍAZ  
LCDO. CIENCIAS FÍSICAS UCM

[www.linkedin.com/in/gabrielmarindiaz/](https://www.linkedin.com/in/gabrielmarindiaz/)