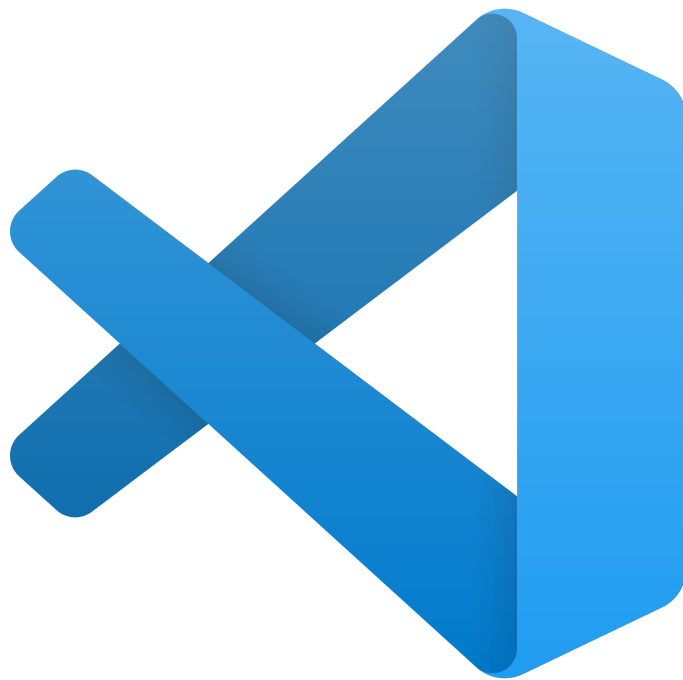


PROYECTO VISUAL STUDIO CODE Y MONGODB



ÍNDICE

1.Instalación Visual Studio Code

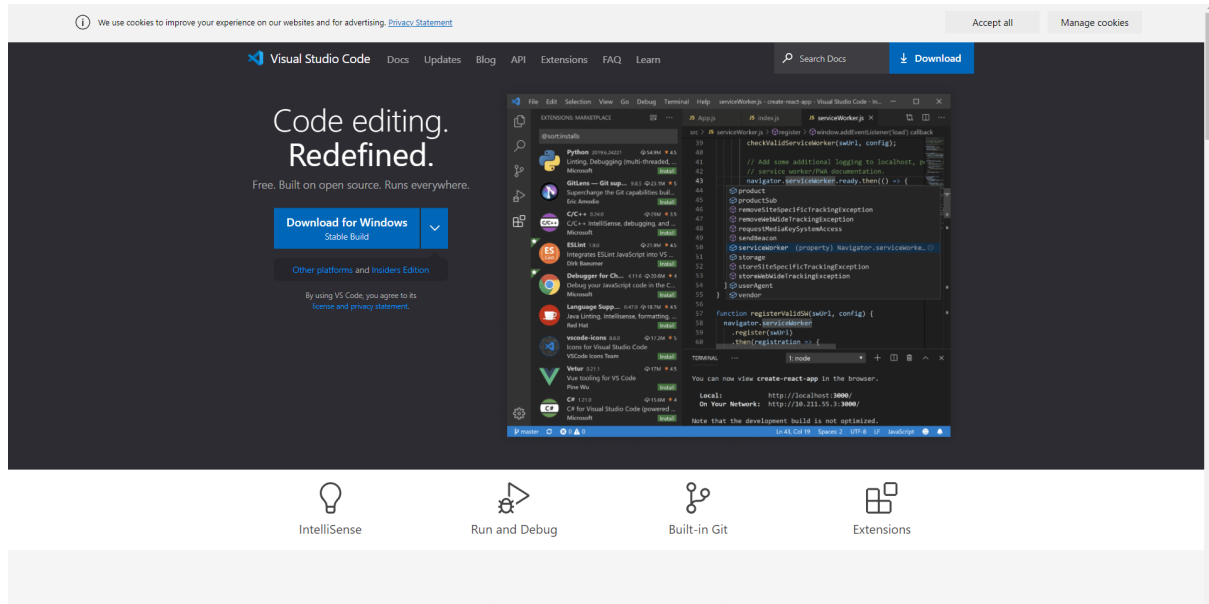
2.Instalación Mongoddb

3.Primeros comandos Mongo Shell

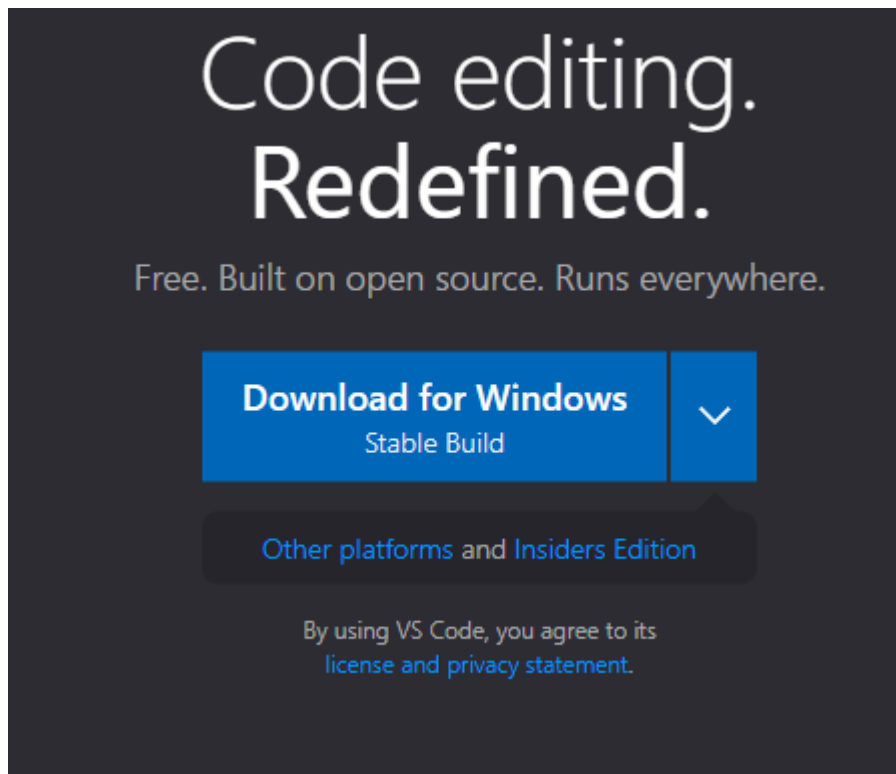
4.Explicación métodos usados

1.Instalación de Visual Studio Code

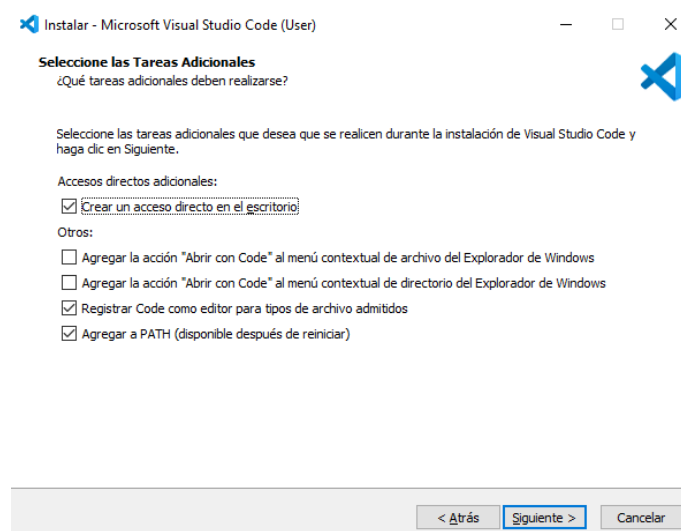
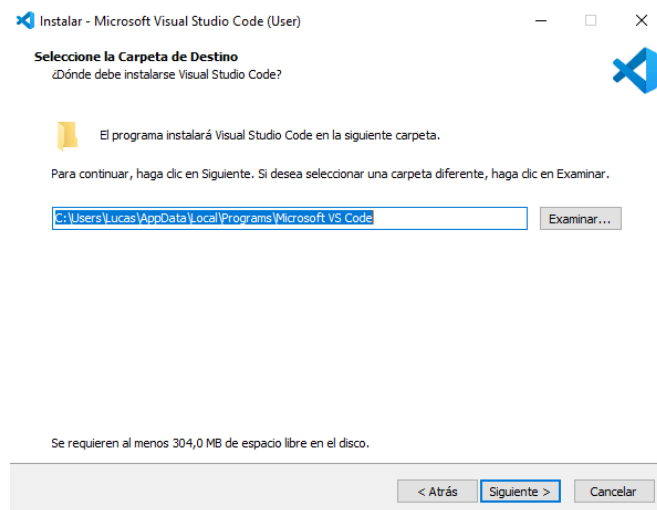
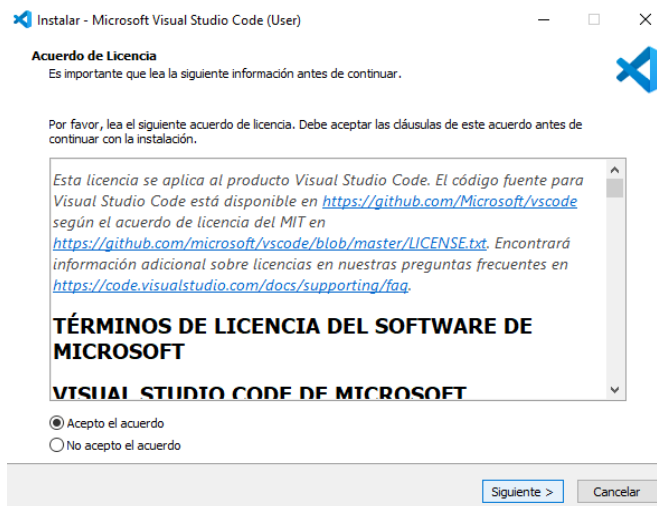
Para instalar el Visual Studio Code primeramente accedemos al siguiente enlace para ir a la página oficial y descargar la aplicación. <https://code.visualstudio.com/>



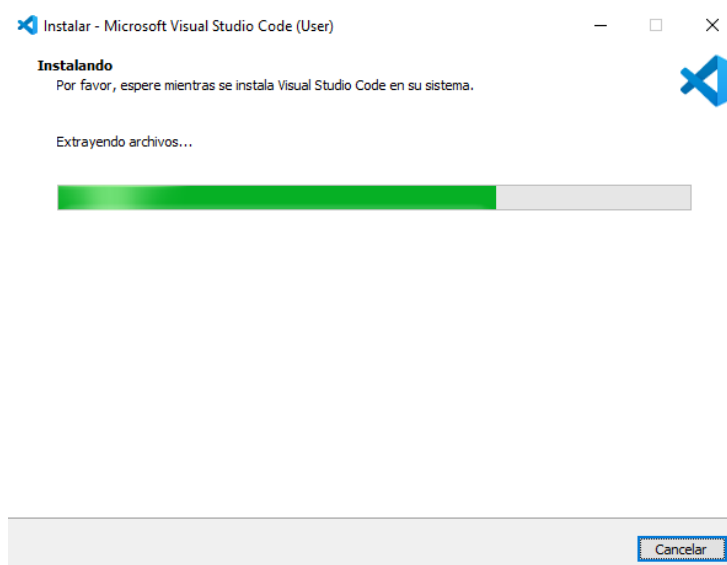
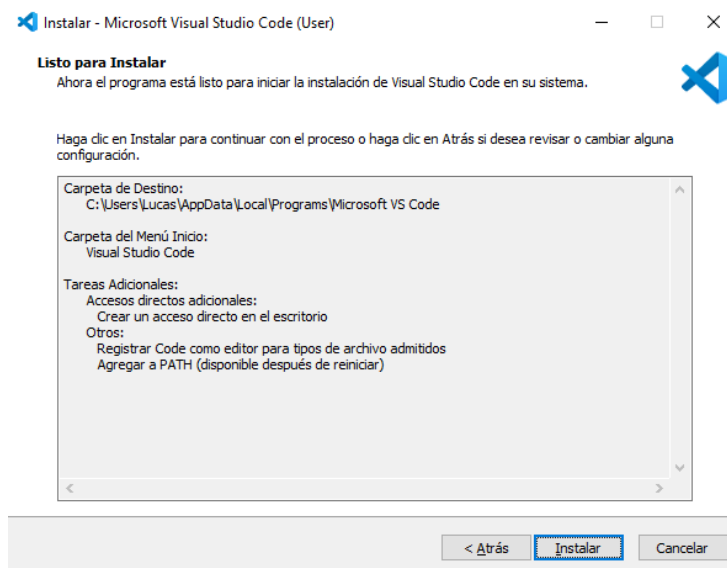
Una vez dentro de la página le daremos a “Download for Windows” para comenzar la descarga.



Ahora una vez descargado ejecutamos el archivo y comenzamos el proceso de instalación.

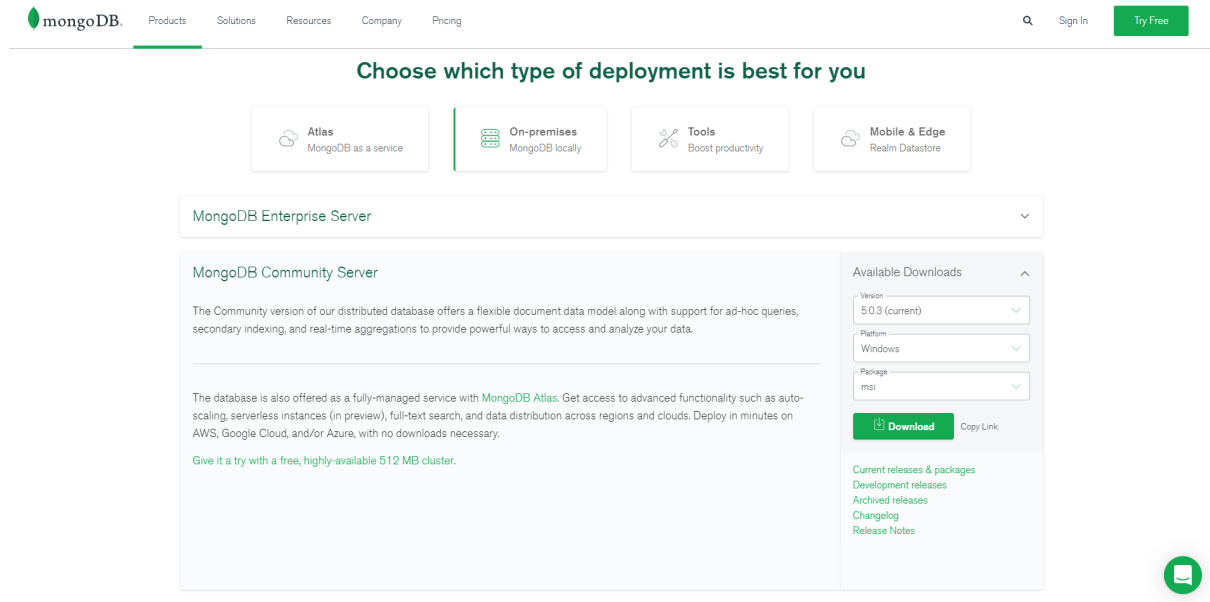


Completamos el Proceso de Instalación siguiendo los pasos, podemos indicar en qué ruta instalarlo y las tareas adicionales que queramos, seguidamente le damos a Instalar y así finalizamos la Instalación.



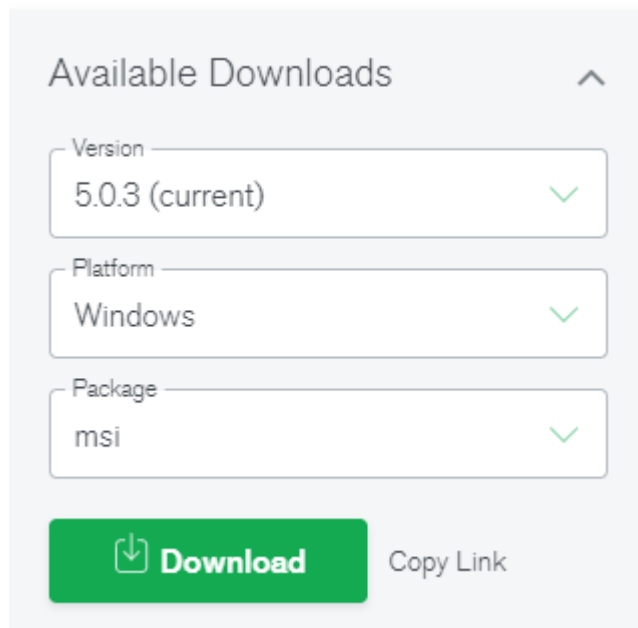
2.Instalación de MongoDB

Procedemos con la Instalación de MongoDB, para ello accedemos al siguiente enlace <https://www.mongodb.com/try/download/community>



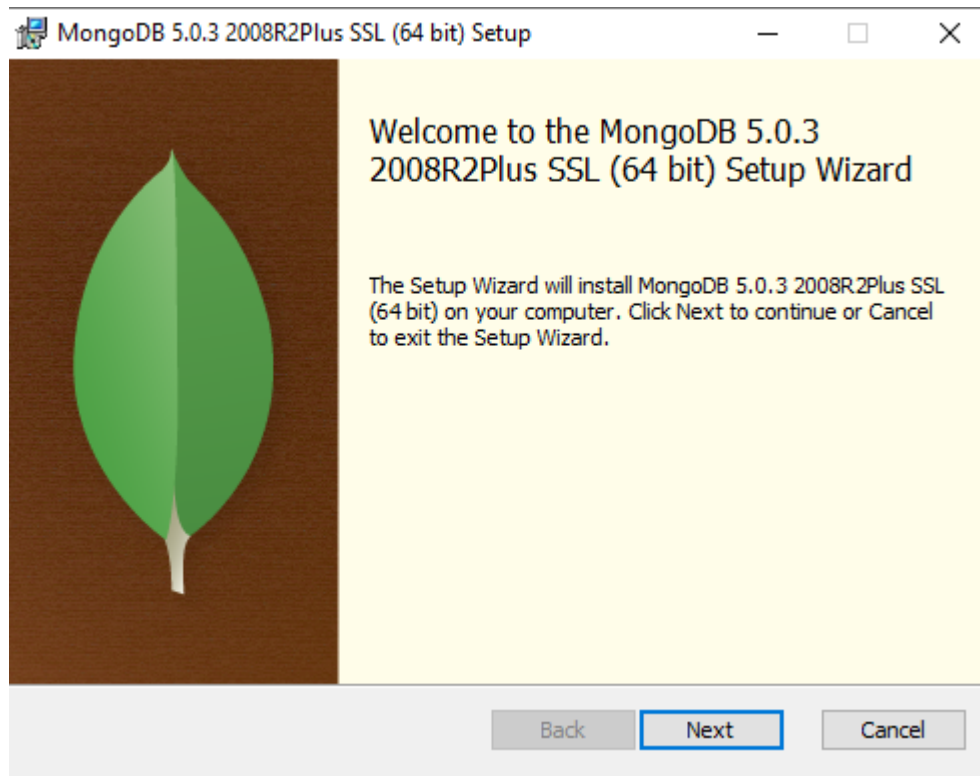
The screenshot shows the MongoDB website's deployment selection interface. At the top, the MongoDB logo is on the left, and navigation links for Products, Solutions, Resources, Company, and Pricing are in the center. On the right, there are links for Sign In and a green Try Free button. Below the navigation bar, a heading reads "Choose which type of deployment is best for you". Four deployment options are presented in boxes: Atlas (MongoDB as a service), On-premises (MongoDB locally), Tools (Boost productivity), and Mobile & Edge (Realm Datastore). The On-premises option is highlighted with a green border. Below these options, a dropdown menu shows "MongoDB Enterprise Server" selected. The main content area displays details for "MongoDB Community Server", including a description of its flexible document data model and support for ad-hoc queries, secondary indexing, and real-time aggregations. It also mentions that the database is offered as a fully-managed service with MongoDB Atlas. A green link offers a free, highly-available 512 MB cluster. On the right side of the main content area, there is a sidebar titled "Available Downloads" with dropdown menus for Version (5.0.3 (current)), Platform (Windows), and Package (msi). Below these dropdowns is a green Download button and a Copy Link button. At the bottom right of the page, there is a green chat icon.

Una vez en la página de MongoDB community server le damos “Download” y comenzará la descarga.

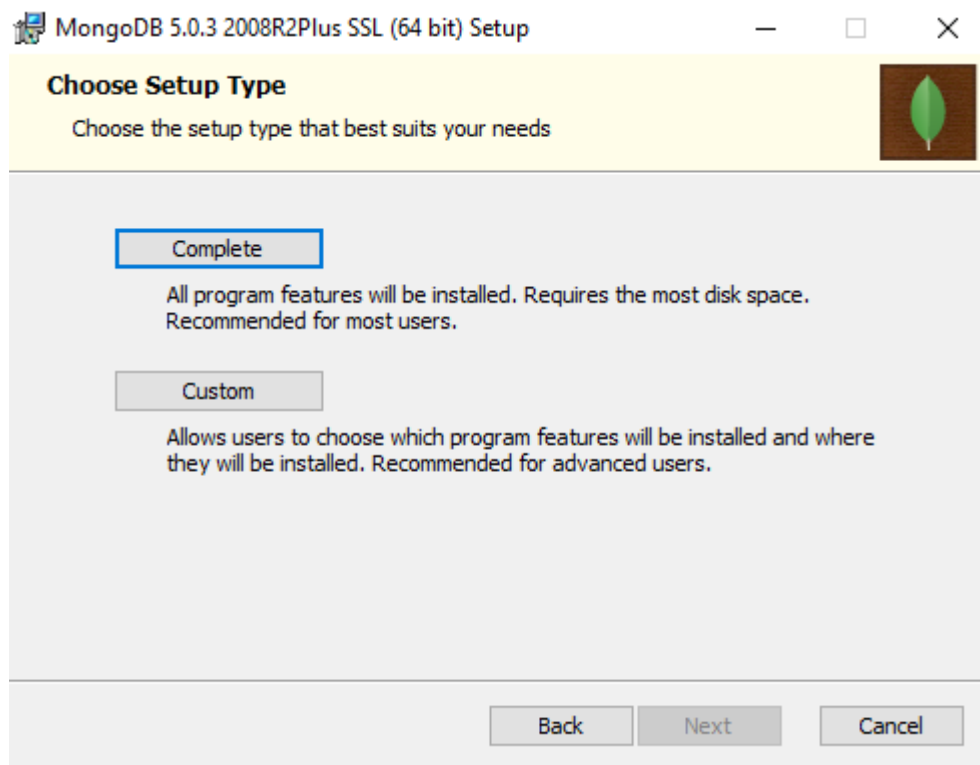


The screenshot shows a close-up of the "Available Downloads" section. It features three dropdown menus: Version (5.0.3 (current)), Platform (Windows), and Package (msi). Each dropdown has a green checkmark icon on the right. Below the dropdowns is a large green button with a download icon and the text "Download", followed by a "Copy Link" button.

Ejecutamos el archivo descargado y comenzamos con la instalación desde la siguiente interfaz.



Seleccionamos "Complete" o "Custom" según queramos una Instalación completa o una Instalación más personalizada.



A continuación Instalamos MongoDB como un servicio.

Service Configuration
Specify optional settings to configure MongoDB as a service.

☒ Install MongoDB as a Service

☒ Run service as Network Service user

☐ Run service as a local or domain user:

Account Domain:

Account Name:

Account Password:

Service Name:

Data Directory:

Log Directory:

< Back Next > Cancel

En la siguiente parte de la instalación se instala también “MongoDB Compass”

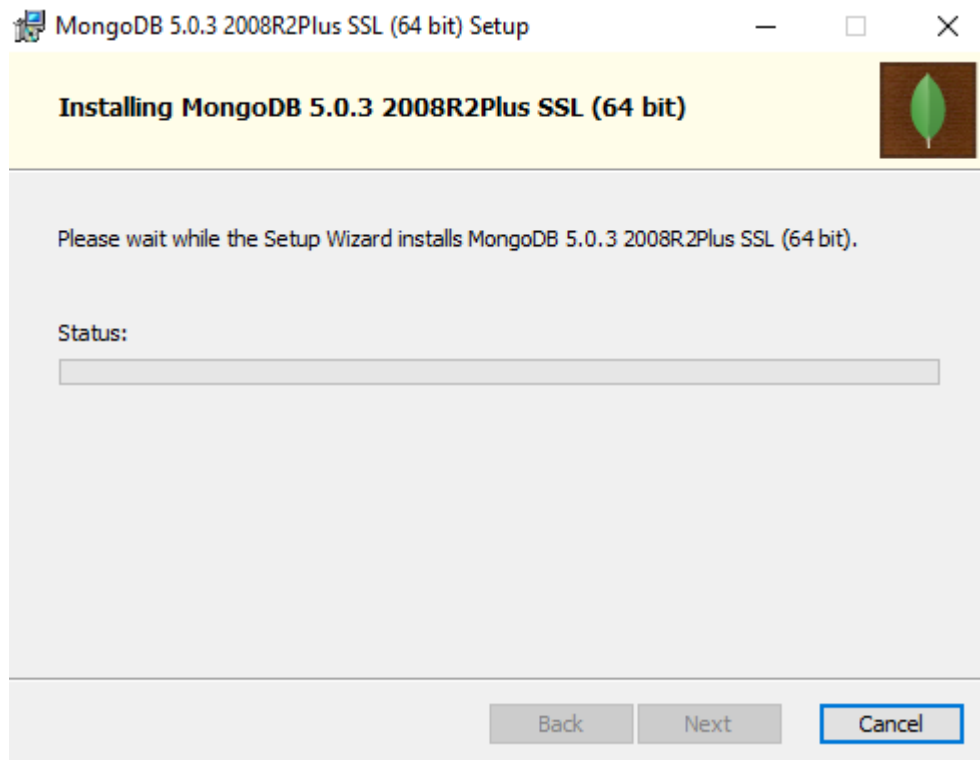
Install MongoDB Compass

MongoDB Compass is the official graphical user interface for MongoDB.

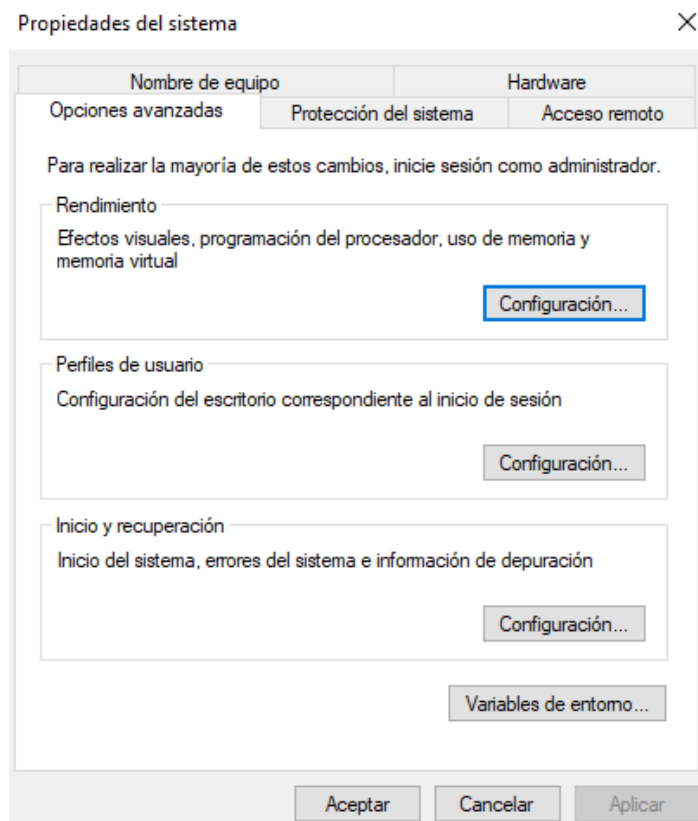
By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: <https://www.mongodb.com/products/comp...>

☒ Install MongoDB Compass Back Next Cancel

Le damos a Next y completamos la instalación.



Una vez finalizada la instalación, para tener acceso al comando “mongo” en PowerShell hay que añadir una nueva entrada en las variables de entorno del sistema “Path”.



Editamos en las Variables del sistema la variable Path.

Variables de entorno



Variables de usuario para Lucas

Variable	Valor
OneDrive	C:\Users\Lucas\OneDrive
Path	C:\Users\Lucas\AppData\Local\Microsoft\WindowsApps;;C:\Users\...
TEMP	C:\Users\Lucas\AppData\Local\Temp
TMP	C:\Users\Lucas\AppData\Local\Temp

Nueva...

Editar...

Eliminar

Variables del sistema

Variable	Valor
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\WIN...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

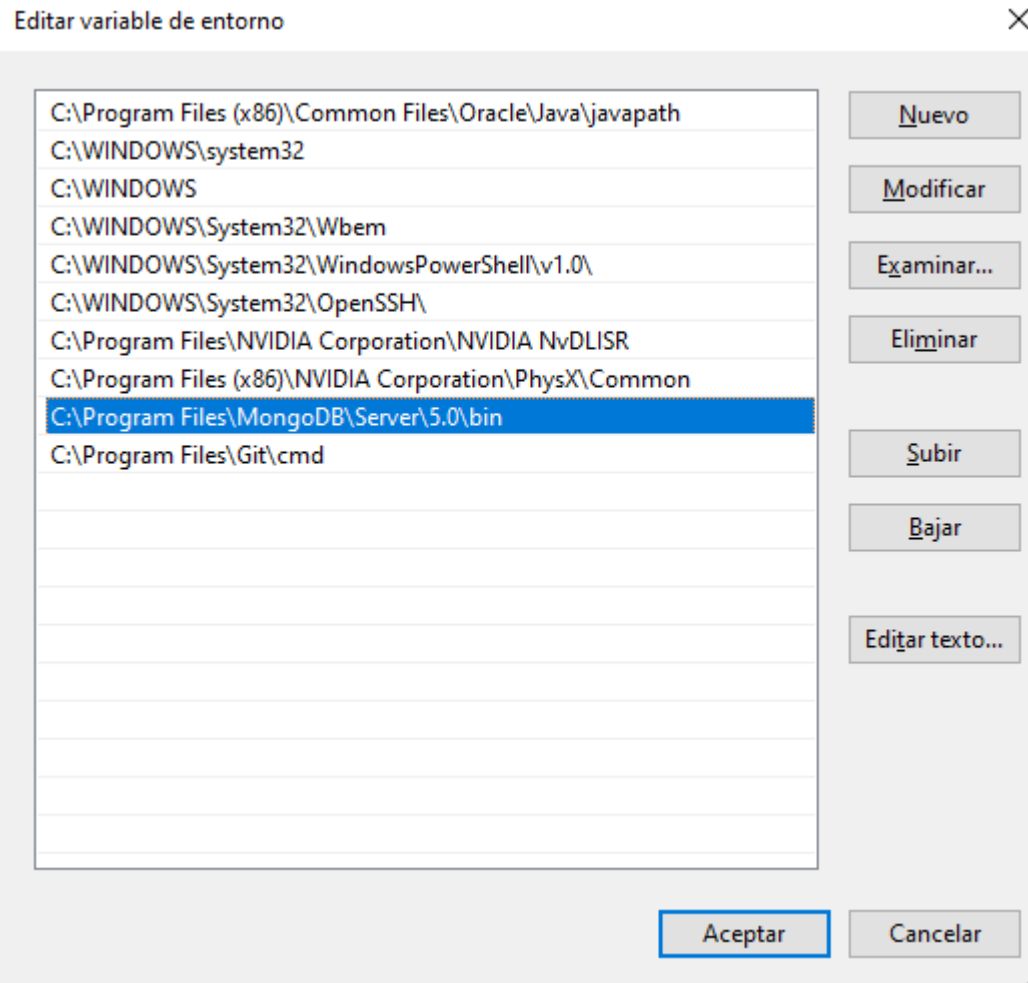
Nueva...

Editar...

Eliminar

Aceptar

Cancelar



Una vez dentro de la interfaz de editar añadimos la ruta del archivo bin de la instalación de MongoDB. En mi caso `C:\Program Files\MongoDB\Server\5.0\bin`.


Ahora probaremos si funciona en PowerShell.

```
PS C:\Users\Lucas> mongo --version
MongoDB shell version v5.0.3
Build Info: {
  "version": "5.0.3",
  "gitVersion": "657fea5a61a74d7a79df7aff8e4bcf0bc742b748",
  "modules": [],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "windows",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
PS C:\Users\Lucas>
```

Como podemos observar el comando funciona y nos enseña la versión de mongo, y por tanto habremos terminado su instalación.

3. Primeros comandos Mongo Shell

Para comenzar a usar MongoDB entramos en la Powershell de Windows y ponemos el directorio de la carpeta de nuestro repositorio usando el comando “cd”.

 Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Lucas> cd C:\Users\Lucas\Documents\PROYECTO_02
PS C:\Users\Lucas\Documents\PROYECTO_02>
```

Una vez dentro usamos el comando “mongo” para poder usar sus comandos.

```
PS C:\Users\Lucas\Documents\PROYECTO_02> mongo
MongoDB shell version v5.0.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("44eba783-7ca2-4bf2-9a45-21498e74609b") }
MongoDB server version: 5.0.3
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
---
The server generated these startup warnings when booting:
  2021-10-17T18:10:40.330+02:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

Ya podemos empezar a usar los comandos de MongoDB.

En primer lugar crearemos una nueva base de datos en la cual trabajaremos, para ello usamos el comando “use <nombre de la base de datos>”

```
> use proyecto_02
switched to db proyecto_02
> █
```

Para comprobar que estamos usando la nueva base de datos creada usamos el comando “db” si nos sale el nombre de la base de datos que anteriormente hemos creado es que estamos usándola.

```
> db
proyecto_02
```

En segundo lugar crearemos en esta nueva base de datos una nueva colección donde guardaremos nuestros documentos, para ello usaremos el siguiente comando “db.<nombre de la nueva colección>.insertOne({})”

```
> db.jugadoresLOL.insertOne({})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("616c6045d29858a5175d3f5a")
}
>
```

Estamos usando el comando insertOne para crear la colección añadiendo un documento vacío.

Para borrar todos los documentos creados en una colección usamos el comando "db.<nombre de la colección>.deleteMany({})"

```
> db.jugadoresLOL.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 1 }
>
```

Podemos observar como se ha borrado el documento que anteriormente insertamos en la colección.

Podemos usar dos comandos para insertar documentos a las colecciones el:

db.nombredelacolección.insertOne({}) que sirve para insertar un único documento

db.nombredelacolección.insertMany({}) que sirve para insertar múltiples documentos.

En el siguiente ejemplo solo uso insertMany porque voy a meter varios documentos.

```
> db.jugadoreslol1.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 5 }
> db.jugadoreslol2.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 5 }
> db.jugadoreslol1.insertMany([
...   { nombre:"Rekkles", edad: 25, equipo:"G2", trofeos: {LEC:5 , MSI:0} },
...   { nombre:"Selfmade", edad: 22, equipo:"VITALITY", trofeos: {LEC:1 , MSI:0}, KDA:3} ,
...   { nombre:"Inspired", edad:23, equipo:"ROGUE", trofeos: {LEC:0 ,MSI:0} },
...   { nombre:"Hanssama", edad:22, equipo:"ROGUE", trofeos: {LEC:0 ,MSI:0}, KDA:4 },
...   { nombre:"Carzzy", edad:20, equipo:"MADLIONS", trofeos: {LEC:4 ,MSI:0} },
... ])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("616c79501cdf5a1534cccc"),
    ObjectId("616c79501cdf5a1534cccd"),
    ObjectId("616c79501cdf5a1534ccce"),
    ObjectId("616c79501cdf5a1534cccf"),
    ObjectId("616c79501cdf5a1534ccc0")
  ]
}
> db.jugadoreslol2.insertMany([
...   { nombre:"Bwipo", edad: 23, equipo:"FNATIC", trofeos:{LEC:4 , MSI:0}},
...   { nombre:"Caps", edad: 22, equipo:"G2", trofeos:{LEC:6 , MSI:1}, KDA:3},
...   { nombre:"Jankos", edad:25, equipo:"G2", trofeos:{LEC:4 ,MSI:1}},
...   { nombre:"Nisqy", edad:25, equipo:"FNATIC", trofeos:{LEC:2 ,MSI:0}, KDA:1.5},
...   { nombre:"Elyoya", edad:20, equipo:"MADLIONS", trofeos:{LEC:2 ,MSI:0}},
... ])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("616c79581cdf5a1534ccc1"),
    ObjectId("616c79581cdf5a1534ccc2"),
    ObjectId("616c79581cdf5a1534ccc3"),
    ObjectId("616c79581cdf5a1534ccc4"),
    ObjectId("616c79581cdf5a1534ccc5")
  ]
}
>
```

De esta manera habríamos creado dos colecciones con 5 documentos cada una.

Si queremos ver cuántas colecciones hay en una base de datos usaremos el comando "show collections"

```
> show collections
jugadoreslol1
jugadoreslol2
>
```

4.Explicación métodos usados

En este proyecto hemos usado los comandos find() con diferentes operadores, para buscar diferentes documentos dentro de las colecciones que hemos creado.

Para buscar todos los documentos de una colección usamos

```
> db.jugadoreslol1.find()
{ "_id" : ObjectId("616c79501cde5a1534cccec"), "nombre" : "Rekkles", "edad" : 25, "equipo" : "G2", "trofeos" : { "LEC" : 5, "MSI" : 0 } }
{ "_id" : ObjectId("616c79501cde5a1534cccd"), "nombre" : "Selfmade", "edad" : 22, "equipo" : "VITALITY", "trofeos" : { "LEC" : 1, "MSI" : 0 }, "KDA" : 3 }
{ "_id" : ObjectId("616c79501cde5a1534cccee"), "nombre" : "Inspired", "edad" : 23, "equipo" : "ROGUE", "trofeos" : { "LEC" : 0, "MSI" : 0 } }
{ "_id" : ObjectId("616c79501cde5a1534cccf"), "nombre" : "Hanssama", "edad" : 22, "equipo" : "ROGUE", "trofeos" : { "LEC" : 0, "MSI" : 0 }, "KDA" : 4 }
{ "_id" : ObjectId("616c79501cde5a1534cccf0"), "nombre" : "Carzzy", "edad" : 20, "equipo" : "MADLIONS", "trofeos" : { "LEC" : 4, "MSI" : 0 } }
>
```

Dentro del comando find he usado el operador \$eq para buscar un documento con un campo con un valor especificado.

```
> db.jugadoreslol1.find( { equipo: { $eq:"G2" } } )
{ "_id" : ObjectId("616c79501cde5a1534cccec"), "nombre" : "Rekkles", "edad" : 25, "equipo" : "G2", "trofeos" : { "LEC" : 5, "MSI" : 0 } }
>
```

El operador \$gt para buscar un documento con un campo con un valor mayor al especificado.

```
> db.jugadoreslol2.find( { edad: { $gt: 22 } })
{ "_id" : ObjectId("616c79501cde5a1534cccf1"), "nombre" : "Bwipo", "edad" : 23, "equipo" : "FNATIC", "trofeos" : { "LEC" : 4, "MSI" : 0 } }
{ "_id" : ObjectId("616c79501cde5a1534cccf3"), "nombre" : "Jankos", "edad" : 25, "equipo" : "G2", "trofeos" : { "LEC" : 4, "MSI" : 1 } }
{ "_id" : ObjectId("616c79501cde5a1534cccf4"), "nombre" : "Nisqy", "edad" : 25, "equipo" : "FNATIC", "trofeos" : { "LEC" : 2, "MSI" : 0 }, "KDA" : 1.5 }
>
```

El operador \$lt para buscar un documento con un campo con un valor menor al especificado.

```
> db.jugadoreslol1.find( { edad: { $lt: 22 } })
{ "_id" : ObjectId("616c79501cde5a1534cccf0"), "nombre" : "Carzzy", "edad" : 20, "equipo" : "MADLIONS", "trofeos" : { "LEC" : 4, "MSI" : 0 } }
>
```

El operador \$gte para buscar un documento con un campo con un calor mayor o igual al especificado.

```
> db.jugadoreslol1.find( { edad: { $gte: 22 } })
{ "_id" : ObjectId("616c79501cde5a1534cccec"), "nombre" : "Rekkles", "edad" : 25, "equipo" : "G2", "trofeos" : { "LEC" : 5, "MSI" : 0 } }
{ "_id" : ObjectId("616c79501cde5a1534cccd"), "nombre" : "Selfmade", "edad" : 22, "equipo" : "VITALITY", "trofeos" : { "LEC" : 1, "MSI" : 0 }, "KDA" : 3 }
{ "_id" : ObjectId("616c79501cde5a1534cccee"), "nombre" : "Inspired", "edad" : 23, "equipo" : "ROGUE", "trofeos" : { "LEC" : 0, "MSI" : 0 } }
{ "_id" : ObjectId("616c79501cde5a1534cccf"), "nombre" : "Hanssama", "edad" : 22, "equipo" : "ROGUE", "trofeos" : { "LEC" : 0, "MSI" : 0 }, "KDA" : 4 }
>
```

El operador \$in para buscar los documentos que en un campo tengan uno de los dos valores especificados.

```
> db.jugadoreslol1.find( { equipo: { $in: ["G2", "ROGUE"] } } )
{ "_id" : ObjectId("616c79501cde5a1534cccec"), "nombre" : "Rekkles", "edad" : 25, "equipo" : "G2", "trofeos" : { "LEC" : 5, "MSI" : 0 } }
{ "_id" : ObjectId("616c79501cde5a1534cccee"), "nombre" : "Inspired", "edad" : 23, "equipo" : "ROGUE", "trofeos" : { "LEC" : 0, "MSI" : 0 } }
{ "_id" : ObjectId("616c79501cde5a1534cccf"), "nombre" : "Hanssama", "edad" : 22, "equipo" : "ROGUE", "trofeos" : { "LEC" : 0, "MSI" : 0 }, "KDA" : 4 }
>
```

Para buscar un documento que tiene un campo que a su vez contiene varios campos valores usamos el siguiente comando con find()

```
> db.jugadoreslol2.find( { trofeos: { LEC:4, MSI:0 } } )
{ "_id" : ObjectId("616c79501cde5a1534cccf1"), "nombre" : "Bwipo", "edad" : 23, "equipo" : "FNATIC", "trofeos" : { "LEC" : 4, "MSI" : 0 } }
>
```

Y por último para buscar un documento con dos campos valores especificados usaremos el siguiente comando

```
> db.jugadoreslol2.find( { $and: [ { edad: { $eq: 25 } }, { KDA: { $exists: true } } ] } )
{ "_id" : ObjectId("616c79501cde5a1534cccf4"), "nombre" : "Nisqy", "edad" : 25, "equipo" : "FNATIC", "trofeos" : { "LEC" : 2, "MSI" : 0 }, "KDA" : 1.5 }
>
```

En este último comando hemos especificado que busque los documentos que cumplan con esos dos campos especificados usando el operador \$and y usando también el operador \$exists que busca todos los documentos que tengan el campo especificado.