



**Clase 20.** Programación Backend

# ***DBaaS & Firebase***



## ***OBJETIVOS DE LA CLASE***

- Conocer el concepto de Base de Datos como un Servicio (DBaaS).
- Crear una cuenta en Mongo Atlas.
- Acceder a Mongo Atlas a través de clientes CLI, GUI y Web.
- Realizar un CRUD utilizando Firebase.

# ***CRONOGRAMA DEL CURSO***

Clase 19



**Mongoose**

---

---

---

---

Clase 20



**DBaaS & Firebase**

---

---

---

---

Clase 21



**Clase 21 - Trabajo con  
datos: Mocks**

---

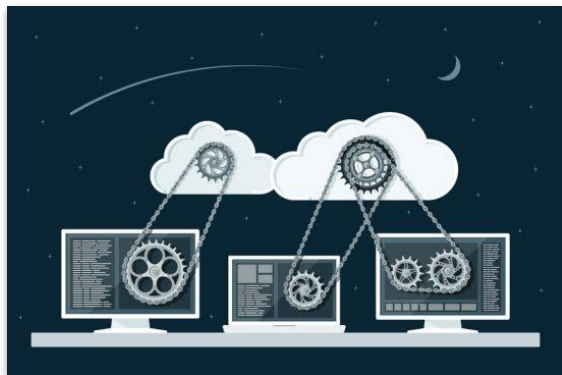
---

---

---

# ***DBaaS: DataBase As A Service***





## ***DBaaS: ¿en qué consiste?***

- Cuando hablamos de Base de Datos como Servicio (**DBaaS**) nos referimos a la **ejecución** y **gestión** de las **bases de datos**, **optimizadas** y **alojadas** en la infraestructura de un **proveedor** de **servicios cloud**.
- De esta manera, para gestionar las bases de datos en el cloud **debemos contar con un servicio «por detrás»** como PaaS o IaaS, para estar seguros de tener la infraestructura necesaria.



# ***Modalidades y Configuraciones***

Podemos definir dos modalidades de servicio bien diferenciadas

- **Modelo clásico:** el cliente hace uso de la infraestructura física del proveedor para alojar sus bases de datos.
- **Alojamiento gestionado:** el cliente se desentiende de cualquier tarea de mantenimiento y gestión avanzada de la base de datos, que asumirá el proveedor.



# ***Ventajas del modelo DBaaS***



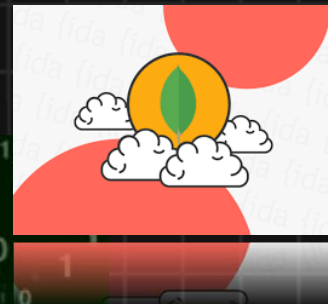
- **Se elimina la infraestructura física** de la ecuación ahorrando en costos, ya que el proveedor es responsable del mantenimiento y la disponibilidad de los sistemas. Los usuarios son responsables de sus propios datos.
- **Ahorro de costos generalizado.** Además de prescindir de las inversiones físicas, con **DBaaS** se puede tener menos personal dedicado a esta tarea, ahorrar en energía y aprovechar mejor el espacio físico.
- **Escalabilidad.** Con **DBaaS** podemos acceder a diferentes tarifas basadas principalmente en el rendimiento deseado y nuestras necesidades.
- **Personal cualificado.** A través de **DBaaS** se accede a expertos en bases de datos que se encargarán de todas las tareas de mantenimiento, actualización, seguridad y gestión.



***¿Alguna pregunta hasta  
ahora?***



# ***MongoDB Atlas***



# ***Atlas: el salto a la nube de MongoDB***

- **MongoDB Atlas** es un servicio de **Cloud Database** (Base de Datos en la Nube), que nos permite crear y administrar nuestra **MongoDB** desde cualquier lugar del mundo a través de su plataforma.
- MongoDB Atlas está orientado a ser **accesible desde el navegador** y fue desarrollado con el objetivo de aliviar el trabajo de los desarrolladores, al quitarles la necesidad de instalar y administrar entornos de Base de Datos.

# ***MongoDB Atlas: Características Destacadas***

- **Automatización:** una manera fácil de crear, lanzar y escalar aplicaciones en MongoDB.
- **Flexibilidad:** DBaaS con todo lo necesario para las aplicaciones modernas.
- **Seguridad:** varios niveles de seguridad disponibles.
- **Escalabilidad:** gran escalabilidad sin interrumpir la actividad.
- **Alta disponibilidad:** implementaciones con tolerancia a errores y autoreparación predeterminadas.
- **Alto rendimiento:** el necesario para las cargas de trabajo exigentes.



# ***Ventajas de MongoDB Atlas***

## **Ejecución**

- Puesta en marcha de un clúster en segundos.
- Implementaciones replicadas y sin interrupción.
- Total escalabilidad: escalado horizontal o vertical sin interrumpir la actividad.
- Revisiones automáticas y actualizaciones simplificadas.

## **Protección y seguridad**

- Autenticación y cifrado.
- Copias de seguridad continuas con recuperación temporal.
- Supervisión detallada y alertas personalizadas.



# ***Ventajas de MongoDB Atlas***

## **Libertad de movimiento**

- Modelo de planes de precio según demanda: se factura por hora.
- Compatible con diferentes tipos de de servicios de nube (AWS, GCP, Azure).
- Parte de un paquete de productos y servicios para todas las fases de la aplicación.

*La configuración se realizará desde una cuenta free. MongoDB Atlas también ofrece una versión de pago.*

# MongoDB Atlas Website: <https://www.mongodb.com/cloud/atlas>

The screenshot shows the MongoDB Atlas website. The main heading is "MongoDB Atlas". Below it, a paragraph describes the service as the most innovative cloud database service on the market, with unmatched data distribution and mobility across AWS, Azure, and Google Cloud. A green "Start free" button is visible. A modal is open, showing the "Cloud Provider & Region" selection process. The modal includes a dropdown for "Cloud Provider & Region" set to "AWS, N. Virginia (us-east-1)". Below this, there are buttons for "aws", "Google Cloud", and "Azure". A section for "Multi-Cloud, Multi-Region & Workload Isolation (M10+ clusters)" is also present, with a toggle switch set to "OFF". At the bottom, a table lists recommended regions for a free tier cluster, categorized by North America, Europe, and Asia.

**MongoDB Atlas**

The most innovative cloud database service on the market, with unmatched data distribution and mobility across AWS, Azure, and Google Cloud, built-in automation for resource and workload optimization, and so much more.

**Start free**

Already have an account? [Log in here](#) →

**Cloud Provider & Region** AWS, N. Virginia (us-east-1) ▼

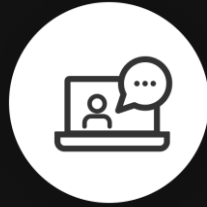
**aws** Google Cloud Azure

**Multi-Cloud, Multi-Region & Workload Isolation (M10+ clusters)**  
Distribute data across clouds or regions for improved availability and local read performance, or introduce replicas for workload isolation. [Learn more](#) ☐

Create a free tier cluster by selecting a region with **FREE TIER AVAILABLE** and choosing the M0 cluster tier below.  
★ Recommended region ⓘ

NORTH AMERICA	EUROPE	ASIA
<b>N. Virginia (us-east-1) ★</b> <b>FREE TIER AVAILABLE</b>	<b>Stockholm (eu-north-1) ★</b>	<b>Hong Kong (ap-east-1) ★</b>
<b>Ohio (us-east-2) ★</b>	<b>Ireland (eu-west-1) ★</b> <b>FREE TIER AVAILABLE</b>	<b>Tokyo (ap-northeast-1) ★</b>
<b>N. California (us-west-1) ★</b>	<b>London (eu-west-2) ★</b>	<b>Seoul (ap-northeast-2) ★</b>
<b>Oregon (us-west-2) ★</b> <b>FREE TIER AVAILABLE</b>	<b>Paris (eu-west-3) ★</b>	<b>Singapore (ap-southeast-1) ★</b> <b>FREE TIER AVAILABLE</b>
<b>Montreal (ca-central-1) ★</b>	<b>Frankfurt (eu-central-1) ★</b> <b>FREE TIER AVAILABLE</b>	<b>Mumbai (ap-south-1) ★</b> <b>FREE TIER AVAILABLE</b>

**CODER HOUSE**



***¿Alguna pregunta hasta  
ahora?***

# ***¿Cómo crear y configurar una cuenta en MongoDB Atlas?***





***¿Alguna pregunta hasta  
ahora?***

# ***MongoDB Atlas con distintos clientes***



***¿Alguna pregunta hasta  
ahora?***



# ***MongoDB Atlas con distintos clientes***

*Tiempo: 10 minutos*



- 1) Conectarse a la base MongoDB Atlas con los clientes **CLI**, **Compass** y **Node.js**
- 2) Mediante la consola CLI insertar estos documentos en una base llamada 'ecommerce', colección 'usuarios':

```
[  
  { nombre: 'Lucas', apellido: 'Blanco', dni: '30355874' },  
  { nombre: 'María', apellido: 'García', dni: '29575148' },  
  { nombre: 'Tomas', apellido: 'Sierra', dni: '38654790' },  
  { nombre: 'Carlos', apellido: 'Fernández', dni: '26935670' }  
]
```

- 1) A través de un proyecto **Node.js**, listar estos datos representándolos en la consola.
- 2) Con el mismo proyecto, incorporar un usuario más:  
nombre: 'Federico', apellido: 'Perez', dni: '320118321' }

Utilizar sintaxis de Promesas con async await e import para la importación de módulos.

- 1) Con **Compass** borrar al usuario llamado Tomas.
- 2) Con **Mongo CLI** actualizar el usuario llamado 'Carlos' al nombre 'Juan Carlos' y luego listar los documentos finales.



***BREAK***

**¡5/10 MINUTOS Y VOLVEMOS!**



***Firebase***

***CODER HOUSE***



# ***¿Qué es y cómo usarlo?***

- Firebase es una **plataforma** para el **desarrollo de aplicaciones web y móviles** desarrollada por James Tamplin y Andrew Lee en 2011 y adquirida por Google en 2014, empezando con su producto base: **base de datos en tiempo real**.
- Firebase permite que, en lugar de hacer peticiones AJAX, el usuario se conecte a la base de datos y automáticamente envíe los datos.
- Firebase puede ser administrado por cualquier aplicación backend y hay múltiples dependencias disponibles para lograr la conexión en cualquier plataforma.



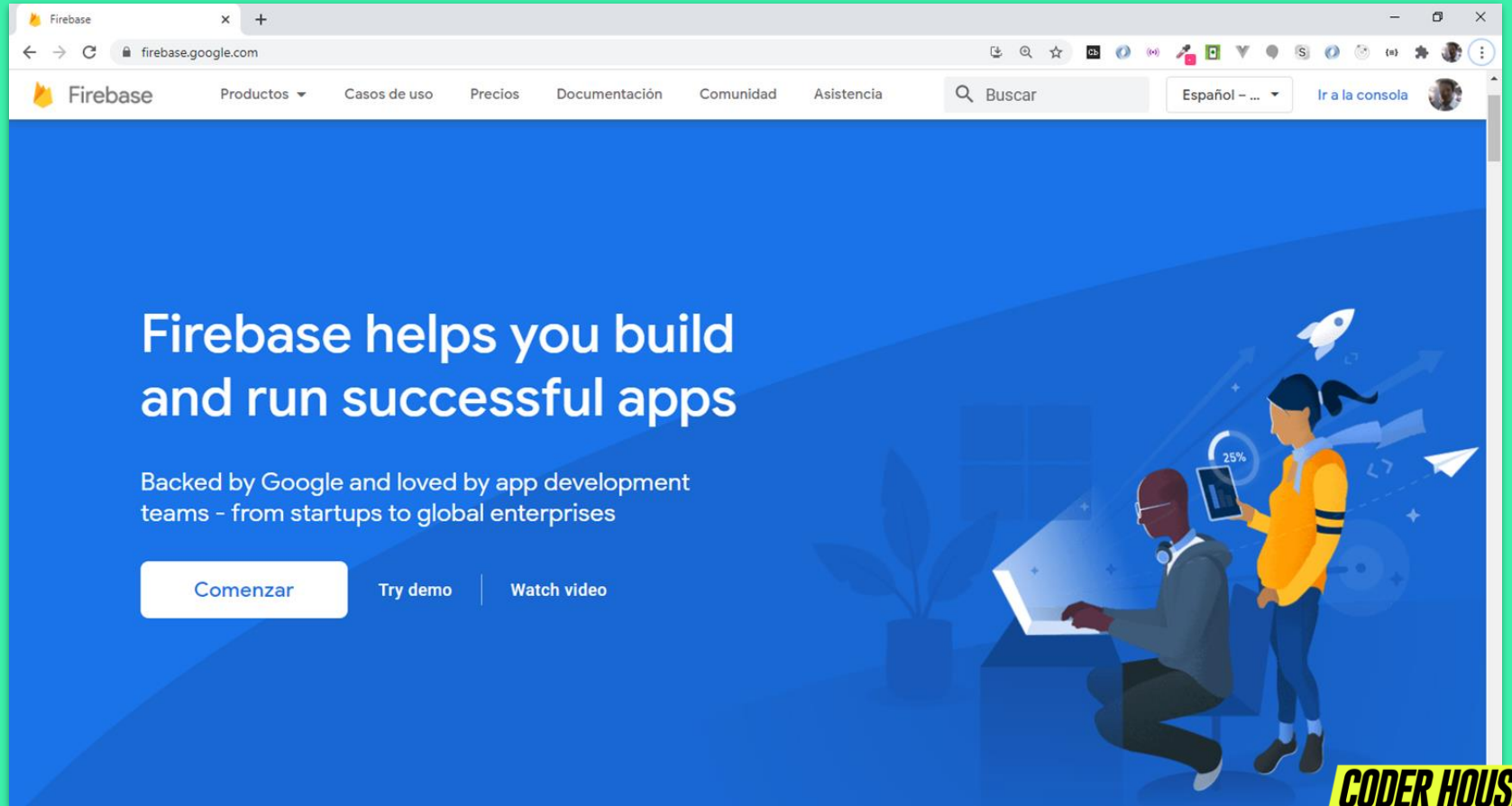


# ***Firestore: Ventajas***



- No necesitamos casarnos con Firestore, **se usa lo que se necesita** (por ejemplo quiero sólo Cloud Functions y Remote Config, pero no Real Time database).
- **Usa Cloud Storage:** base de datos para que usuarios puedan compartir ficheros e imágenes, sin necesidad de hacer bases de datos propias, que para imágenes a veces puede ser un poco 'tedioso'.
- **Usa Cloud Functions:** con esto nos ahorramos toda la infraestructura de backend. Es lo que más cobra Google, ya que sabe que es en lo que más ahorramos. Con el plan Blaze con las CF puedes hacer llamadas a tu API, no hay firewalls.

**Firestore Website:** <https://firebase.google.com/>



The image is a screenshot of the Firebase website homepage, viewed in a web browser. The browser's address bar shows the URL `firebase.google.com`. The website's navigation bar includes the Firebase logo, links for "Productos", "Casos de uso", "Precios", "Documentación", "Comunidad", and "Asistencia", a search bar with the text "Buscar", a language selector set to "Español", and a link to "Ir a la consola". The main content area has a blue background with the headline "Firestore helps you build and run successful apps". Below this, it states "Backed by Google and loved by app development teams - from startups to global enterprises". There are three buttons: "Comenzar", "Try demo", and "Watch video". An illustration on the right shows two people, a man and a woman, working together; the man is sitting and using a laptop, while the woman is standing and holding a tablet. The background of the illustration includes a rocket, arrows, and a plant.

Firestore helps you build  
and run successful apps

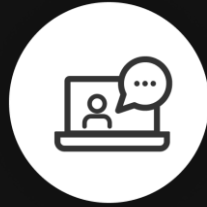
Backed by Google and loved by app development  
teams - from startups to global enterprises

Comenzar Try demo Watch video

**CODER HOUSE**

# ***Firestore y Node.js***





***¿Alguna pregunta hasta  
ahora?***

# ***Configuración del proyecto Node.js***



# ***Node.js con Firebase***

*Tiempo: 10 minutos*

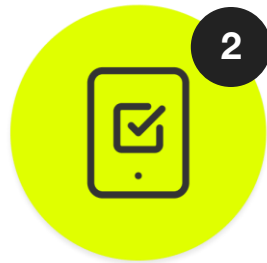


Realizar un proyecto en Node.js que acceda a una base de datos Firebase ya configurada.

- 1) **Agregar** los colores red, green, blue dentro de una colección llamada 'colores' con el formato { nombre: color }
- 2) **Listar** todos los colores disponibles.
- 3) **Modificar** el color blue por navy.
- 4) **Borrar** el color green

### A tener en cuenta:

- Implementar estas funciones utilizando Promises en las funciones de Firebase con sintaxis async/await, utilizando la importación en formato ES Modules (import)
- Verificar la información de la base de datos con la consola de Firebase.



# ***SEGUNDA ENTREGA DEL PROYECTO FINAL***

Deberás entregar el avance de tu aplicación eCommerce Backend correspondiente a la segunda entrega de tu proyecto final.



## ***SEGUNDA ENTREGA DEL PROYECTO FINAL***

**Formato:** link a un repositorio en Github con el proyecto cargado.

**Sugerencia:** no incluir los node\_modules

Proyecto  
Final



**>>Consigna:** Basándose en los contenedores ya desarrollados (memoria, archivos) desarrollar dos contenedores más (que cumplan con la misma interfaz) que permitan realizar las operaciones básicas de CRUD en MongoDB (ya sea local o remoto) y en Firebase. Luego, para cada contenedor, crear dos clases derivadas, una para trabajar con Productos, y otra para trabajar con Carritos.

# SEGUNDA ENTREGA DEL PROYECTO FINAL

**Formato:** link a un repositorio en Github con el proyecto cargado.

**Sugerencia:** no incluir los node\_modules

Proyecto  
Final



## >>Aspectos a incluir en el entregable:

- A las clases derivadas de los contenedores se las conoce como DAOs (Data Access Objects), y pueden ir todas incluidas en una misma carpeta de 'daos'.
- En la carpeta de daos, incluir un archivo que importe todas las clases y exporte una instancia de dao de productos y una de dao de carritos, según corresponda. Esta decisión se tomará en base al valor de una variable de entorno cargada al momento de ejecutar el servidor (opcional: investigar el uso de imports dinámicos).
- Incluir un archivo de configuración (config) que contenga los datos correspondientes para conectarse a las bases de datos o medio de persistencia que corresponda.

## >>Opcional:

- Hacer lo mismo para bases de datos relacionales: MariaDB/SQLite3.

# SEGUNDA ENTREGA DEL PROYECTO FINAL

**Formato:** link a un repositorio en Github con el proyecto cargado.

**Sugerencia:** no incluir los node\_modules

Proyecto  
Final



## Ejemplos

```
import { promises as fs } from 'fs'

class ContenedorArchivo {

  constructor(ruta) {
    this.ruta = ruta;
  }

  async listar(id) {
    const objs = await this.listarAll()
    const buscado = objs.find(o => o.id === id)
    return buscado
  }

  async listarAll() {
    try {
      const objs = await fs.readFile(this.ruta)
      return JSON.parse(objs)
    } catch (error) {
      return []
    }
  }
}
```

```
import ContenedorArchivo from "../..../contenedores/ContenedorArchivo.js"

class ProductosDaoArchivo extends ContenedorArchivo {

  constructor() {
    super('DB/productos.json')
  }

  async desconectar() {
    // ...
  }
}

export default ProductosDaoArchivo
```

```
src
├── contenedores
│   ├── ContenedorArchivo.js
│   ├── ContenedorFirebase.js
│   ├── ContenedorMemoria.js
│   └── ContenedorMongoDb.js
├── daos
│   ├── carritos
│   │   ├── CarritosDaoArchivo.js
│   │   ├── CarritosDaoFirebase.js
│   │   ├── CarritosDaoMem.js
│   │   └── CarritosDaoMongoDb.js
│   └── productos
│       ├── ProductosDaoArchivo.js
│       ├── ProductosDaoFirebase.js
│       ├── ProductosDaoMem.js
│       └── ProductosDaoMongoDb.js
├── index.js
├── utils
├── config.js
├── main.js
└── server.js
```

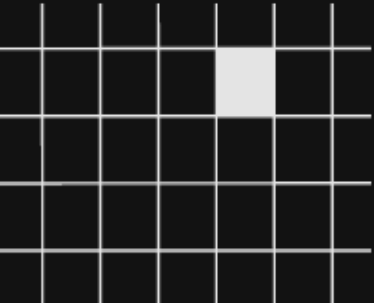
***¿PREGUNTAS?***





# ***¡MUCHAS GRACIAS!***

Resumen de lo visto en clase hoy:

- DBaaS
  - Mongo Atlas
  - Firebase
  - Integración con Node.js
- 



***OPINA Y VALORA ESTA CLASE***

***#DEMOCRATIZANDO LA EDUCACIÓN***