Esta clase va a ser

- grabada

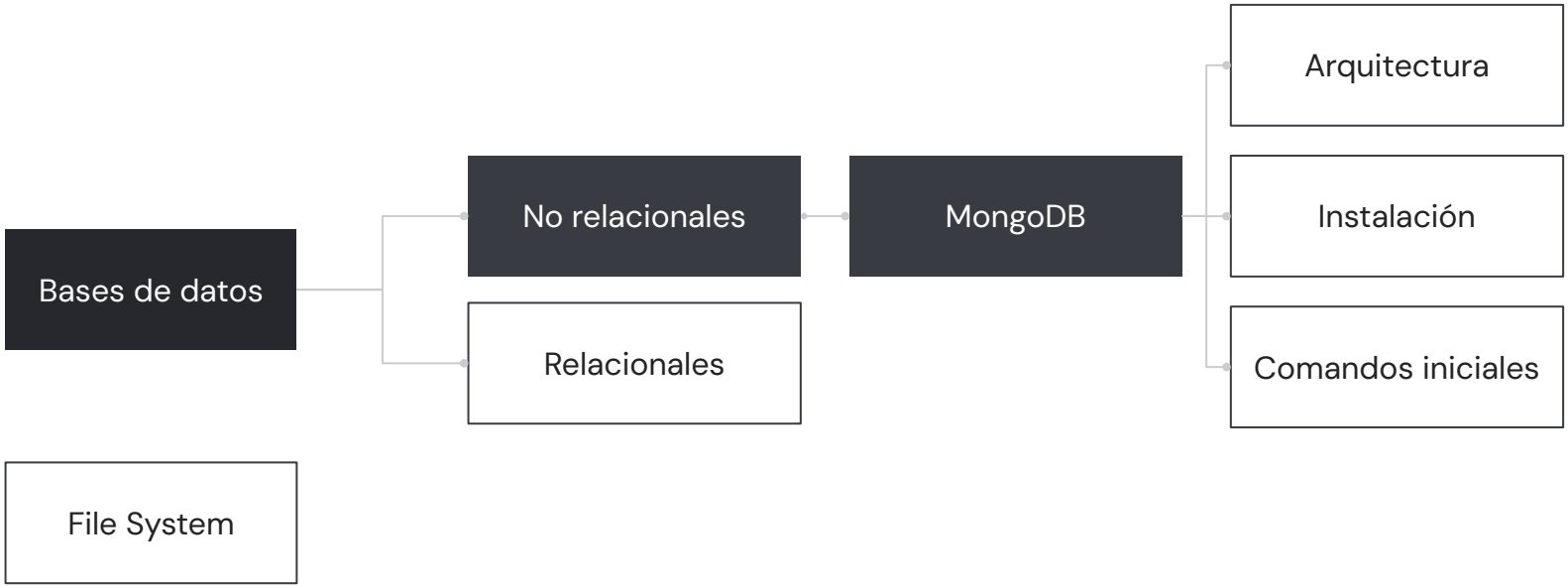
Clase 12. DESARROLLO AVANZADO DE BACKEND

MongoDB

Objetivos de la clase

- Comprender el concepto de una base de datos
- Comprender las diferencias entre bases de datos relacionales y no relaciones
- Comprender el concepto de MongoDB y su respectiva instalación

MAPA DE CONCEPTOS



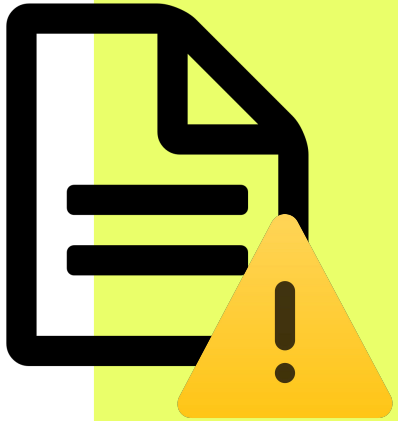
El gran problema de la persistencia

El gran problema de la persistencia: archivos

Hasta este punto del curso, hemos estado trabajando con un fileSystem (sistema de archivos) para hacer nuestros almacenamientos principales.

Sin embargo, sabemos claramente que hay algunos problemas con el uso de archivos, como son:

- ✓ Tener que actualizar todo el archivo cuando hacemos algún cambio.
- ✓ Tener que leer todo el archivo cuando buscamos algún dato.
- ✓ Sin protección al momento de querer mover o modificar algo.



CODERHOUSE

Base de datos

Una base de datos no es más que una recopilación organizada de datos. Dichos datos deben compartir algún contexto y son almacenados con poder convertirse posteriormente en información útil para utilizarse dentro de algún sistema.

La base de datos sólo se encargará de almacenar dichos datos, el uso que les demos posteriormente ya no competen a ésta.

Comparaciones entre FileSystem y Base de Datos

Tarea: Aumentar la edad de todos los usuarios femeninos en 1 año

FileSystem	Base de datos
Leer el archivo	Acceder a la base de datos
Mapear los usuarios 1 a 1, actualizando la propiedad para aumentar en 1 año a los usuarios que cumplan con el género “femenino”.	Solicitar que se actualicen a los usuarios femeninos en 1 año
Guardar el arreglo mapeado en un nuevo arreglo	
Sobreescribir **todo el archivo** (incluyendo a los que no actualizamos)	

Tarea: Devolver sólo el nombre y apellido de todos los usuarios, ordenados alfabéticamente

FileSystem	Base de datos
Leer el archivo	Acceder a la base de datos
Mapear el arreglo de usuarios, pasando a todos los usuarios a un objeto nuevo que sólo cuente con las dos propiedades de interés.	Solicitar una proyección de los usuarios en nombre y apellido, solicitando en la misma instrucción por orden alfabético
Guardar el arreglo mapeado en un nuevo arreglo	Devolver usuarios obtenidos.
Aplicar sobre este arreglo un algoritmo de ordenamiento	...
Devolver usuarios obtenidos	:)

Tarea: Agregar un usuario nuevo con un id autoincremental

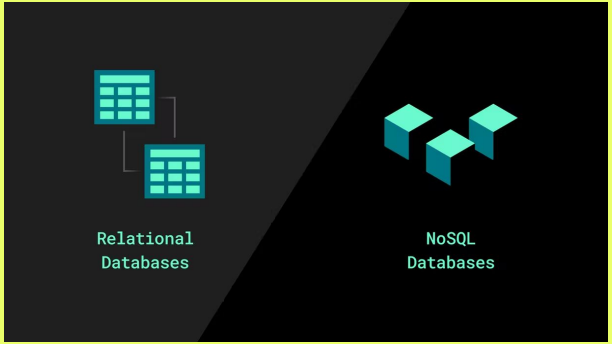
FileSystem	Base de datos
Leer el archivo	Acceder a la base de datos
Parsear el contenido del archivo en un arreglo.	Agregar el dato
Calcular el id que corresponde al último usuario del arreglo y asignarlo al usuario a agregar.	
Agregar el usuario al arreglo.	
Escribir **Todo** el arreglo (incluyendo a todos los usuarios que ya existen) para sobreescribir el archivo.	

Modelo Relacional y No Relacional

La necesidad de distintos modelos de bases de datos

Una vez que entendimos que la base de datos nos sirve para mantener los datos organizados, toca entender cuándo utilizar un modelo relacional o un modelo no relacional.

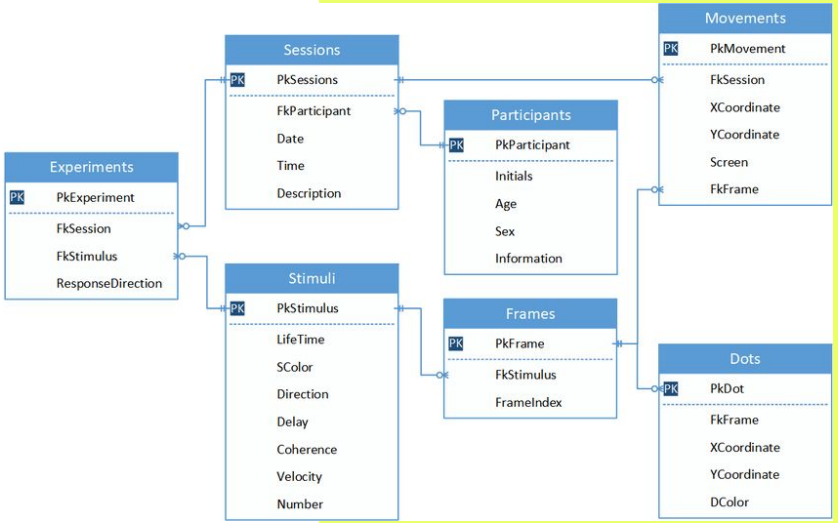
- ✓ Una base de datos **relacional** refiere a estructura, relación, dependencia y de cambio controlado.
- ✓ Una base de datos **no relacional** refiere a algo menos estructurado, con relaciones y dependencias más flexibles, y de cambios sumamente rápidos.



Modelo relacional

Refiere a modelos de datos donde se requieren estructuras más firmes y estrictas sobre los datos. Además, se utilizan en datos más controlados

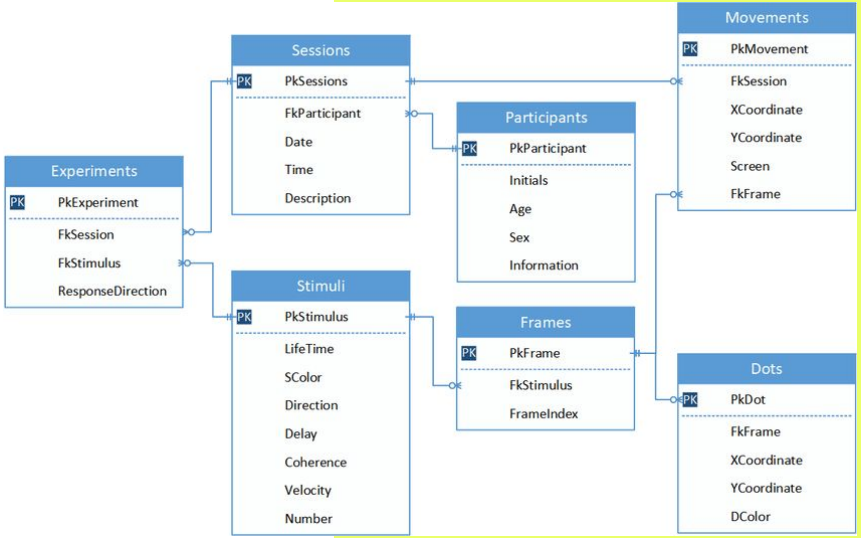
- ✓ Se basan en tablas, columnas y filas para gestionar sus datos.
- ✓ Permiten conectar las tablas a partir de “relaciones” basadas en llaves primarias y foráneas.



CODERHOUSE

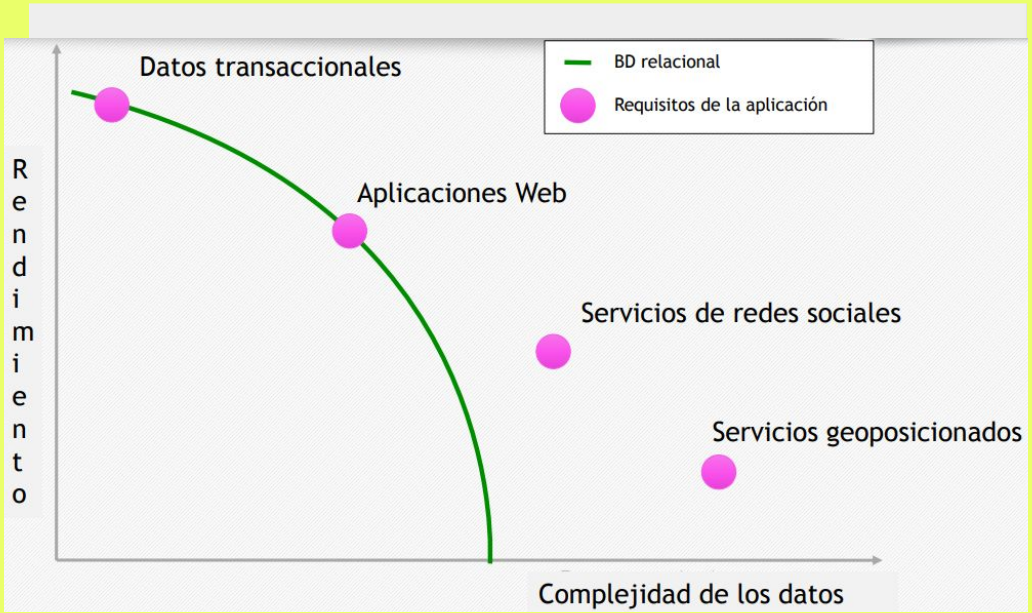
Modelo relacional

- ✓ Sirve para datos de control como:
  - ✓ Sistemas bancarios
  - ✓ Sistemas de clima (no tiempo atmosférico).
  - ✓ Sistemas de películas.
- ✓ Su lenguaje es SQL (Structured Query Language)
- ✓ Algunos sistemas SQL son:
  - ✓ PostgreSQL
  - ✓ Oracle
  - ✓ MySQL
  - ✓ MariaDB



CODERHOUSE

El problema de las BD relacionales



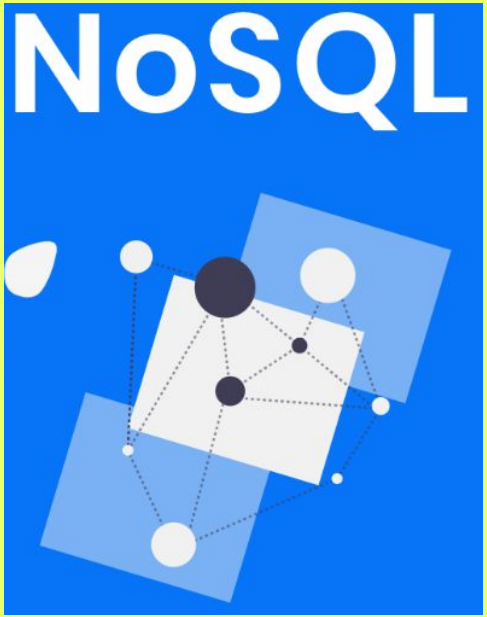
Cuando las aplicaciones que necesitamos incrementan sus requisitos, los datos cambian más rápido y son más complejos, son más inconsistentes y por lo tanto nuestra base de datos relacional comienza a volverse lenta.

Solución: introducir un modelo más flexible

Introduciendo el modelo no relacional

Se desarrolla un modelo donde los datos sean más flexibles, tanto en estructura, como en asociación.

Todo esto con el fin de crear datos pensados para desempeño, no para consistencia inmediata.



Modelo no relacional

La flexibilidad de los datos lo hace considerablemente más rápido en cuanto a su accesibilidad.

- ✓ Puede basarse en:
  - ✓ Clave valor
  - ✓ Documentos
  - ✓ gráficos
  - ✓ memoria

Nota la diferencia en la estructura. A esto nos referimos con “flexibilidad en los datos”

CODERHOUSE

Modelo no relacional

- ✓ Son bases de datos muy útiles para organizar y gestionar información no estructurada, o cuando no se tiene una noción clara de los datos a almacenar.
- ✓ Alto grado de escalabilidad y de performance
- ✓ No utiliza SQL como lenguaje
- ✓ Algunos sistemas No SQL son
  - ✓ MongoDB
  - ✓ Redis
  - ✓ DynamoDB

Nota la diferencia en la estructura. A esto nos referimos con “flexibilidad en los datos”

CODERHOUSE

¡Importante!

El que las bases de datos no relacionales sean utilizados en aplicaciones modernas con mayor movilidad de datos, no significa que las bases de datos relacionales no sirvan ya. Las bases de datos relacionales siguen siendo ampliamente demandadas en ciertos sectores del mercado (sistemas bancarios, sistemas de películas, clima, etc)

¿Cuándo usar cada modelo?

Modelo relacional	**Modelo no relacional**
Cuando el volumen de los datos no crece, o bien lo hace poco a poco	Cuando el volumen de los datos crece rápidamente
Cuando las necesidades del proceso pueden atenderse en un solo servidor	Cuando las necesidades del proceso son tan altas y tan constantes, que se requieren multi servidores
Cuando no existen picos de uso por parte de los usuarios que utilizan el sistema.	Cuando los usuarios saturan el sistema y generan “picos de uso”.

Profundizando sobre bases de datos no relacionales: MongoDB

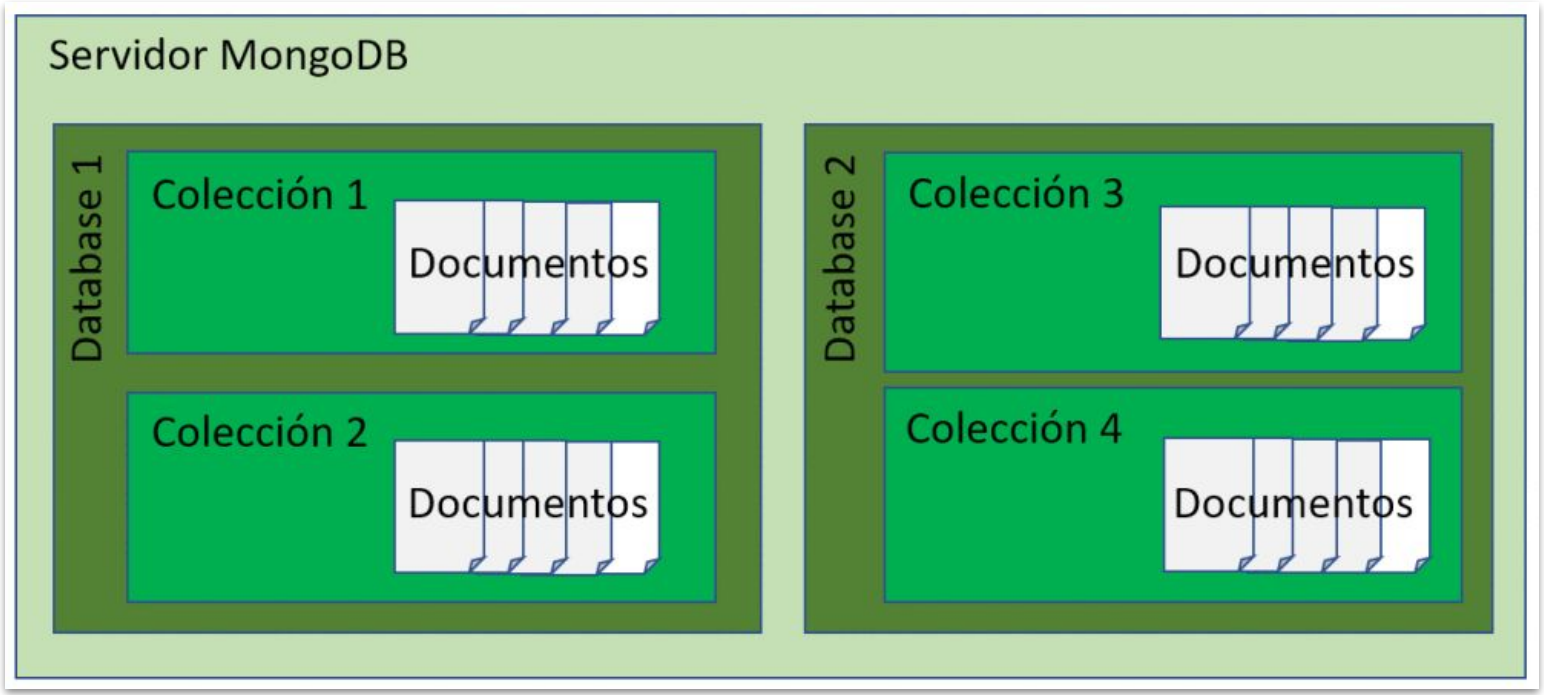
MongoDB

- ✓ Base de datos no relacional **orientada a documentos**
- ✓ En lugar de tablas, opta por utilizar **colecciones**,
- ✓ Cada documento que ingresamos a una colección puede tener diferente estructura
- ✓ Puede utilizarse en modo local o en la nube

MongoDB: características

- ✓ Almacena datos en **documentos** flexibles similares a **JSON**: la estructura de datos se puede cambiar con el tiempo.
- ✓ **El modelo** de documento se asigna a los objetos en el código de su aplicación para facilitar el trabajo con los datos.
- ✓ Las consultas *ad hoc*, la indexación y la agregación en tiempo real ofrecen maneras potentes de acceder a los datos y analizarlos.
- ✓ MongoDB es una base de datos distribuida en su núcleo, por lo que la **alta disponibilidad**, la **escalabilidad** horizontal y la distribución geográfica están integradas y son fáciles de usar.
- ✓ MongoDB es de **uso gratuito**.

MongoDB: arquitectura



Documentos

Una de las grandes ventajas de un documento es que éste se basa en el concepto de ***clave-valor***, esto, como sabrás, se asemeja muchísimo a un objeto como el que has trabajado tanto tiempo en javascript

No son propiamente un “objeto” como para llamarlos de tal forma, sino que MongoDB trabaja con una extensión de los archivos conocidos como BSON. Esto es lo que realmente permanece en la base de datos.

Los esquemas de una base de datos en MongoDB, con ayuda de elementos como mongoose, son fácilmente manipulables, ya que permite definirlos con una estructura casi idéntica a la de un objeto.

JSON (BSON)

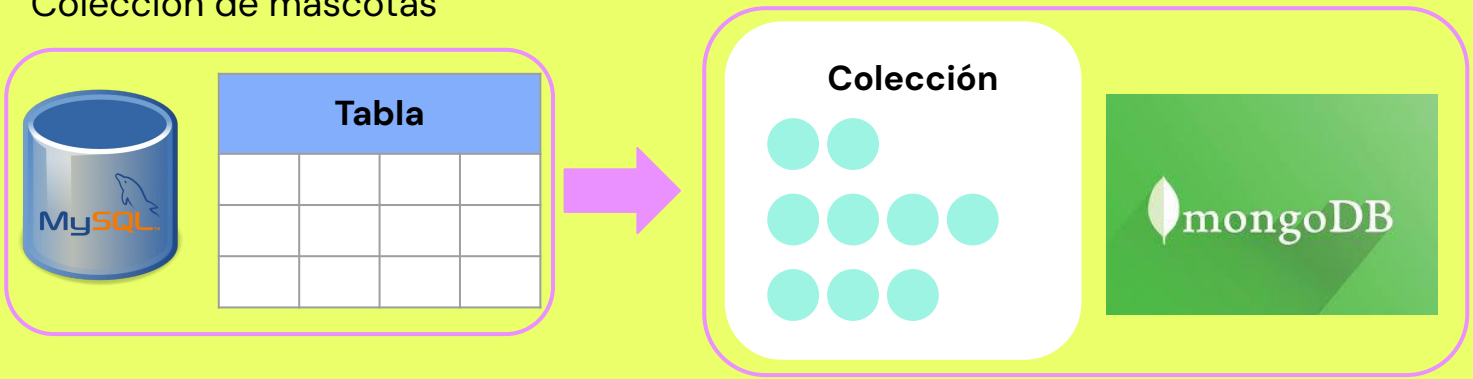
```
{
    “nombre”: “Juan”,
    “edad”: 25
    “dirección”:
         {
              “ciudad”: “Barcelona”
         },
    “aficiones”:[
         {“nombre”: “Fútbol” },
         {“nombre”: “Esquí” }
    ]
}
```

CODERHOUSE

Colecciones en MongoDB

Cada vez que pensemos en un grupo de datos, lo mencionaremos como una **colección**, en ésta almacenaremos cada documento individual (similar a una tabla con sus registros).

- ✓ Colección de usuarios
- ✓ Colección de productos
- ✓ Colección de mascotas



Instalar MongoDB

Corroborando la instalación

Break

¡10 minutos y volvemos!

¿Quieres aprender más sobre SQL?
¡Tenemos un curso para ti!

Ahora que ya instalamos MongoDB...

probemos comandos que te pueden resultar útiles

CODERHOUSE

ACTIVIDAD EN CLASE

Primeros pasos con Mongo

- ✓ Una vez que corrobores que mongo está instalado en computador, a partir del cliente CLI, crear una base de datos de nombre “estudiantes”
- ✓ Agregar 5 estudiantes diferentes con los campos “nombre”, “apellido”, “curso”, “correo”. Puedes utilizar db.collection.insertMany()
- ✓ Una vez agregados, listar a los estudiantes de dicha colección y corroborar su persistencia.

Segunda pre-entrega de tu Proyecto final

Websockets.
Integrar vistas y sockets a nuestro servidor actual.

CODERHOUSE

Websockets

Consigna

- ✓ Configurar nuestro proyecto para que trabaje con Handlebars y websocket.

Aspectos a incluir

- ✓ Configurar el servidor para integrar el motor de plantillas Handlebars e instalar un servidor de socket.io al mismo.
- ✓ Crear una vista “home.handlebars” la cual contenga una lista de todos los productos agregados hasta el momento
- ✓ Además, crear una vista “realTimeProducts.handlebars”, la cual vivirá en el endpoint “/realtimeproducts” en nuestro views router, ésta contendrá la misma lista de productos, sin embargo, ésta trabajará con websockets.
  - ✓ Al trabajar con websockets, cada vez que creemos un producto nuevo, o bien cada vez que eliminemos un producto, se debe actualizar automáticamente en dicha vista la lista.

ENTREGA DEL PROYECTO FINAL

Sugerencias

- ✓ Ya que la conexión entre una consulta HTTP y websocket no está contemplada dentro de la clase. Se recomienda que, para la creación y eliminación de un producto, Se cree un formulario simple en la vista realTimeProducts.handlebars. Para que el contenido se envíe desde websockets y no HTTP. Sin embargo, esta no es la mejor solución, leer el siguiente punto.
- ✓ Si se desea hacer la conexión de socket emits con HTTP, deberás buscar la forma de utilizar el servidor **io** de Sockets dentro de la petición POST. ¿Cómo utilizarás un emit dentro del POST?

Formato de entrega

- ✓ Link al repositorio de Github, el cual debe contar con todo el proyecto.
- ✓ No incluir node_modules

Testing de este entregable

CODERHOUSE

¿Preguntas?

Opina y valora esta clase

Muchas gracias.