 arielZarate / **henryAll** Public

Code

Issues

Pull requests

Actions

Projects

Wiki

Security






Insights

Settings

 main ▾

...


henryAll / FT-M4-master / 01-dbms /

 arielZarate ...	3 days ago 
..	
 homework	3 days ago
 README.json	3 days ago
 README.md	3 days ago




 README.md





Hacé click acá para dejar tu feedback sobre esta clase.



Hacé click acá completar el quiz teórico de esta lecture.

# Bases de Datos

## Necesidad de una base datos

En algún momento nuestra aplicación va a necesitar algún tipo de *persistencia* de datos, es decir que los datos queden guardados en el disco, y no importa si reinicio el servidor, o modifico mi aplicación u otras cosas que puedan ocurrir. Ya aprendimos que con `nodejs` es fácil leer y escribir archivos. Así que una forma de lograr persistencia sería escribir en archivos de texto en el disco, no? sí! pero eso **no escala**. Si lo hicieramos, nos veríamos enredados en buscar lo que queríamos dentro de cada archivo, y en poco tiempo buscar algo tardaría mucho, y nos dejaría de servir. Obviamente, si sólo tenemos que guardar cosas para luego leerlas después, este sistema nos puede servir, como por ejemplo: los logs de las páginas. Cada entrada es guardada como una línea en un archivo de texto. Otra solución, más escalable para lograr persistencia de datos es usar una *base de datos*.

- **Base de datos:** Es una colección de datos de un mismo dominio y organizada sistemáticamente para su posterior uso. Esta organización, en general, está construida de tal manera que *modele* el problema de la mejor forma.
- **DBMS** (Database Management System): es una aplicación que interactúa con el usuario, otra aplicación y la base de datos misma, de tal forma que pueda definir, crear, borrar, modificar, consultar y administrar bases de datos y datos en sí.



Lo que vamos a hacer entonces, es usar un *DBMS* para que nos ayude a guardar los datos. Y como todo el mundo utiliza estas aplicaciones, ya hay escritas muchas librerías de `nodejs` para que nos sirven como interfaz y que son fáciles de usar. Como se imaginan hay muchos sabores de DBMS para elegir. Lo primero es elegir si queremos uno que sea relacional (SQL) o uno no relacional (noSQL). Ahora vamos a empezar a ver como guardar datos en una base de datos no relacional, en particular `MongoDB`.

## Bases de Datos Relacionales

---

Como habíamos visto, la alternativa a las bases de datos `noSQL` son las bases de datos relacionales. En estas bases de datos las tablas tienen (no es obligatorio pero fuertemente recomendado) que estar normalizadas (3era forma normal). Y antes de empezar a cargar datos, tenemos que definir el modelo de datos de manera detallada, como en `mongoose` pero obligatoriamente!

Las ventajas de usar una base de datos SQL son:

- Cómo nos obliga a definir un modelo de antemano, la aplicación va a ser muy estable y difícilmente llegué un dato no deseado a la BD. El problema es que es poco flexible y hacer cambios una vez arrancado el proyecto puede ser muy costoso.
- Estas bases de datos son transaccionales, es decir que el motor de DB nos asegura que las operaciones que hagamos van a hacerse atómicamente, es decir que jamás vamos a tener datos corruptos.

- Es una tecnología muy estudiada, hace años que ya está estable, en contrapartida con las bases de datos NoSQL que son relativamente nuevas.

## Homework

---

Completa la tarea descrita en el archivo [README](#)

---