



Curso de Back-End con Node.js (Inicial)

Clase 06



Temario



Temario

- Base de datos.
- Claves primarias.
- Tipos de datos.
- Instalación de MySQL.
- Acceso a MySQL a través de CLI y GUI.
- Lenguaje SQL.



Base de Datos



Base de Datos – Definición

Una base de datos es una **colección de datos que está organizada** de tal manera que pueda ser accedida, gestionada y actualizada con facilidad.

Abreviación: BD o DB.

Notar que la definición no habla de software. De hecho, con esta definición, una guía telefónica o una agenda de contactos también se pueden considerar como bases de datos.

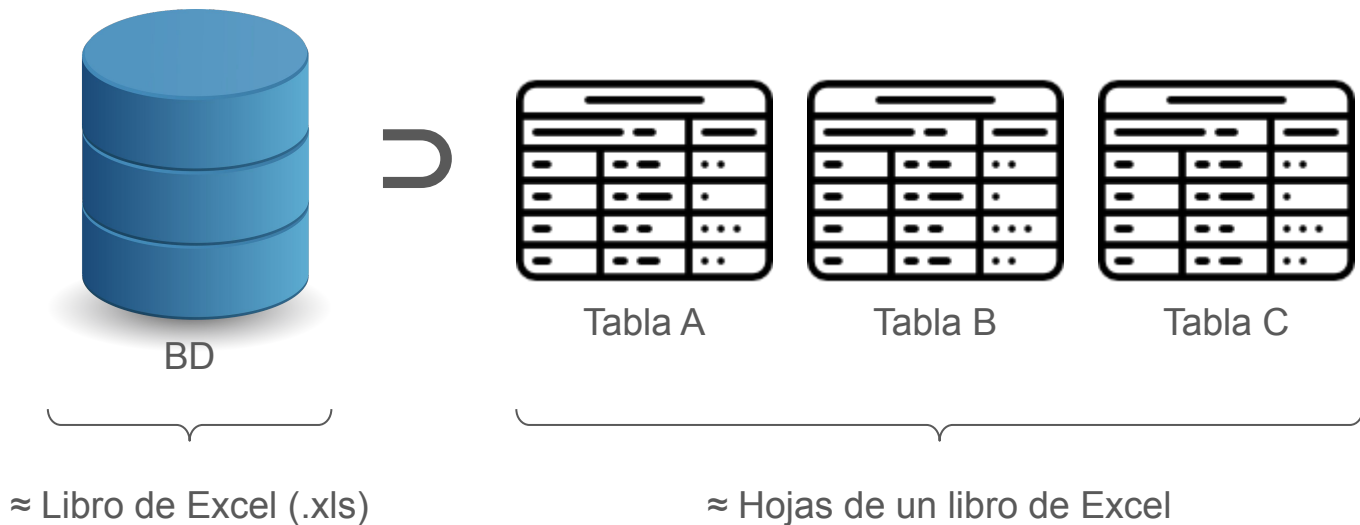




Base de Datos Relacional (1/2)

Una **base de datos relacional** es una base de datos que organiza la información en una o más **tablas**.

Una **tabla** es una colección de datos organizados en filas y columnas (similar a Excel). A veces, a las tablas se les dice *relaciones*. A las filas también se les llama **registros** o *tuplas*.





Base de Datos Relacional (2/2)

Ejemplo: **Tabla Usuarios**

Tabla

id	nombre	apellido	edad	genero
1	Juan	Pérez	23	Hombre
2	María	Sánchez	34	Mujer
3	Pedro	Álvarez	28	Hombre

Campo

**Registro
o tupla**



DBMS – Database Management System

Un **DBMS** es un **software** que permite crear y manipular bases de datos. Es como una interface entre las BD y las aplicaciones. Algunos gestores populares son:

- MySQL.
 - MariaDB.
 - SQL Server.
 - Microsoft Access.
 - Oracle.
 - PostgreSQL.
 - MongoDB.
 - Cassandra.
-
- Relacionales.
- No Relacionales.

El que usaremos en el curso.

¿Relacional vs. No Relacional?
Ver → [Link 1.](#) [Link 2.](#)

Nota: Al DBMS también se le conoce como “Motor de BD”, “Gestor de BD” o “Manejador de BD”.

SQL



Es un **lenguaje** para manipular (guardar, obtener, actualizar, borrar) datos de una base de datos relacional.

Es el lenguaje utilizado para comunicarse con el gestor de base de datos (DBMS).

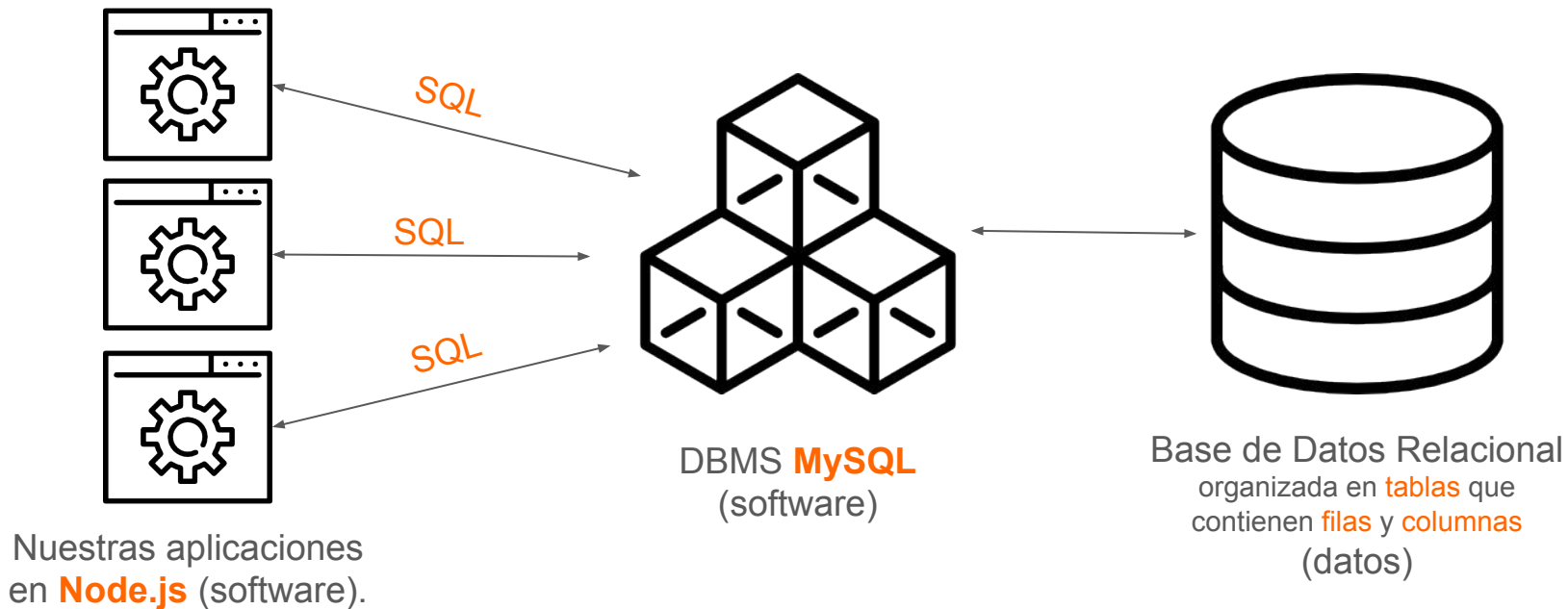
SQL = Structured Query Language.

Nota: más adelante en la clase veremos SQL en mayor detalle.



Node.js + MySQL

En este sprint vamos a crear **aplicaciones con Node.js**, que mediante el **lenguaje SQL** se comunicarán con un **gestor de bases de datos** llamado **MySQL**, que a su vez será el encargado de manipular la base de datos.





Clave primaria



Clave primaria

Es recomendable que en cada tabla exista un campo llamado `id` que permita identificar inequívocamente a cada registro. Se dice que `id` es una **clave primaria** de la tabla (y sólo puede haber una por tabla).

<code>id</code>	<code>nombre</code>	<code>apellido</code>	<code>edad</code>	<code>genero</code>
1	Juan	Pérez	23	Hombre
2	María	Sánchez	NULL	Mujer
3	Pedro	Álvarez	28	Hombre

Nota: el nombre `id` es simplemente una convención. Podría ser cualquier otro nombre.

Incluso, en el caso de una tabla de usuarios, se podría usar como clave primaria el campo *Cédula de Identidad* ya que el mismo permite identificar inequívocamente a cada persona. Sin embargo, incluso en estos casos es usual tener la columna `id`.



Tipos de Datos



Tipos de Datos

Cuando se crea un campo en una base de datos MySQL, se debe especificar el **tipo de dato** que contendrá dicho campo. Ver [documentación](#).

Según el tipo de dato, también será necesario especificar el largo que tendrá el campo.

Ejemplos:

INT (y similares)	Para números enteros.
VARCHAR	Para textos cortos.
TEXT	Para textos largos.
DATETIME	Para fechas/horas.
TIMESTAMP	Para fechas/horas.
BOOLEAN	Para false/true, se representan con 0 y 1.

También es posible especificar valores por defecto para cada campo y si el campo admite valores vacíos (NULL).



Resumen de lo visto hasta ahora




Resumen de lo visto hasta ahora

- Una **base de datos relacional** es una base de datos que organiza la información en una o más tablas.
- Una **tabla** es una colección de datos organizados en **filas** y **columnas**.
En general, las tablas llevan de nombre a sustantivos en plural: usuarios, autos, tareas, etc., aunque existen [grandes debates](#) al respecto.
- A las filas se les llama **registros** o tuplas.
- A las columnas se les llama **campos** y cada uno tiene su propio **tipo**.
Ejemplo: id, nombre, apellido, cedula, etc.
- Las tablas suelen tener un campo (o un conjunto de campos), que identifican a cada registro, y se le llama **clave primaria**. Ejemplos: cédula, nro. de serie, nro. de pasaporte, RUT, ISBN, etc.
- Como clave primaria se suele utilizar un campo llamado **id** que se auto-incrementa cada vez que se crea un nuevo registro.





Instalar MySQL

 Importante: lean las siguientes diapositivas con mucha atención.



Instalar MySQL (1/2)

Si bien MySQL arrancó siendo un software 100% **open-source**, la empresa que actualmente detrás del mismo (Oracle) comercializa algunas versiones que son pagas. La versión que usaremos nosotros es la **MySQL Community Edition** ([link](#)) y es gratuita. Por más detalles sobre la instalación entrar [aquí](#).

Links de descarga:

- **Windows:** <https://dev.mysql.com/downloads/installer>. Más info [aquí](#).
⚠ Durante la instalación, seleccionar la opción “[Server Only](#)” y en Authentication Method seleccionar la opción “[Legacy](#)”.
- **Mac:** <https://dev.mysql.com/doc/refman/8.0/en/macOS-installation-pkg.html>.
También pueden usar [Homebrew](#).
- **Linux:** <https://dev.mysql.com/doc/refman/8.0/en/linux-installation.html>.

👉 Si ya tienen instalado un programa como MAMP, XAMPP, WAMP o similar, es probable que ya tengan MySQL (o MariaDB) en sus máquinas, y no sería necesario hacer una instalación adicional. Sin embargo, recomendamos seguir la instalación descrita en los enlaces de arriba.

Instalar MySQL (2/2)

Por defecto, MySQL estará escuchando en el puerto 3306.



Es probable que en algún momento de la instalación se nos solicite un **usuario** y **contraseña** para la base de datos. En un ambiente de desarrollo (nuestra PC) es común poner “root” para ambos datos.

Dependiendo de la forma en que hayan instalado MySQL, a veces es necesario correr el comando `mysql_secure_installation` (en la terminal) para hacer un *setup* inicial de la base de datos. Esto no es necesario si usaron el instalador de Windows.

Una vez que MySQL se encuentre **instalado** y corriendo, se podrá interactuar con el mismo mediante el cliente de línea de comandos (**CLI**) o mediante algún software para administrar bases de datos. Para la primera opción, se deberá escribir en la terminal:

```
mysql -u root -p
```

Si se eligió otro nombre de usuario, sustituir “root” por el mismo.



Aplicaciones para administrar MySQL

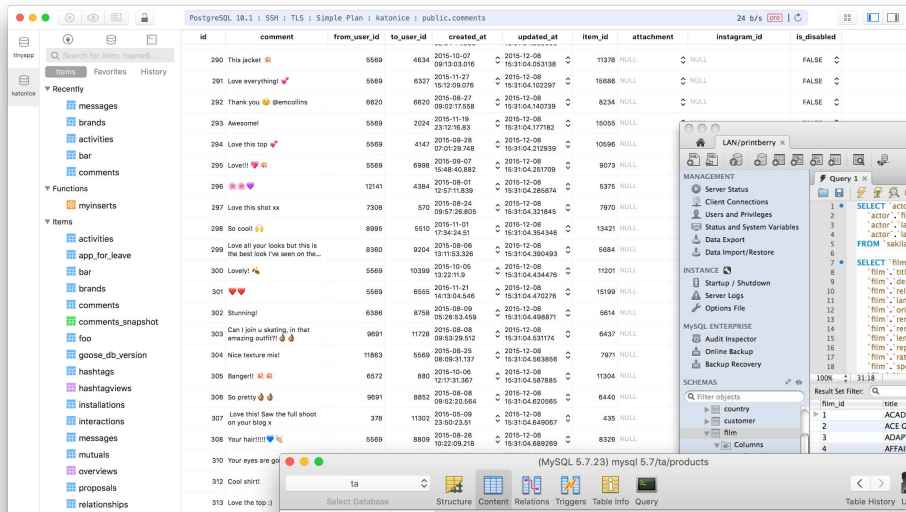


Apps para administrar MySQL

Si bien es posible interactuar con MySQL mediante la CLI, mucha gente prefiere hacerlo a través de una **GUI** para desktop. Algunas opciones son:

Nombre	Gratuito	Windows	Mac	Linux
TablePlus	Parcialmente	✓	✓	✓
Beekeeper Studio	Sí	✓	✓	✓
MySQL Workbench	Sí	✓	✓	✓
DBeaver	Sí	✓	✓	✓
Sequel Pro	Sí	✗	✓	✗
HeidiSQL	Sí	✓	✗	✗

También existen GUIs con interfaz web como [Node MySQL Admin](#), [phpMyAdmin](#) y [Adminer](#) (las dos últimas escritas en PHP), aunque no suelen ser opciones demasiado “robustas”.



TABLES	search:	productCode	=	Q	Filter
academic	productCode	productName	productLine	productScale	productVendor
author	S10_1678	1969 Harley Davidson Ultimate Ch.	Motorcycles	1:10	Min Lin Diecast
comments	S10_1949	1952 Alpine Renault 1300	Classics	1:10	Classic Metal Creations
department	S10_2016	1996 Moto Guzzi 1100	Motorcycles	1:10	Highway 66 Mini Classics
field	S10_4098	2003 Harley-Davidson Eagle Drag.	Motorcycles	1:10	Red Start Diecast
interest	S10_4757	1972 Alfa Romeo GTA	Classics	1:10	Moto City Classics
paper	S10_4962	1962 Lancia Delta 16V	Classics	1:10	Second Gear Diecast
products	S12_1099	1968 Ford Mustang	Classic Cars	1:12	AutoArt Studio Diecast
tt	S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast
untitled_table_0	S12_1666	1958 Stetco Bus	Trucks and Buses	1:12	Welly Diecast Productions
	S12_2063	2002 Ferrari F40	Classic Cars	1:12	Unimart Art Galleries
	S12_3148	1969 Corvair Monza	Classic Cars	1:18	Welly Diecast Productions
	S12_3380	1968 Dodge Charger	Classic Cars	1:18	Welly Diecast Productions
	S12_3891	1969 Ford Falcon	Classic Cars	1:18	Second Gear Diecast
	S12_3990	1970 Plymouth Hemi Cuda	Classic Cars	1:12	Studio M Art Models
	S12_4473	1957 Chevy Pickup	Trucks and Buses	1:12	Exoto Designs
	S12_4675	1969 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions
	S18_1097	1940 Ford Pickup Truck	Trucks and Buses	1:18	Studio M Art Models
	S18_1142	1937 Lincoln Continental	Classic Cars	1:18	Highway 66 Mini Classics
	S18_1367	1936 Mercedes-Benz 500K Special	Vintage Cars	1:18	Motor City Classics
	S18_1589	1965 Aston Martin DB5	Classic Cars	1:18	Studio M Art Models
	S18_1662	1980s Black Hawk Helicopter	Planes	1:18	Classic Metal Creations
	S18_1749	1917 Grand Touring Sedan	Vintage Cars	1:18	Red Start Diecast
	S18_1889	1948 Ford 356-A Roadster	Classic Cars	1:18	Welly Diecast Productions
	S18_1984	1995 Honda Civic	Classic Cars	1:18	Gearbox Collectibles
	S18_2238	1998 Chrysler Plymouth Prowler	Classic Cars	1:18	Min Lin Diecast

110 rows in table

LAN/printserver ✕

MySQL Workbench

MANAGEMENT

Server Status

Client Connections

Users and Privileges

Status and System Variables

Data Export

Data Import/Restore

INSTANCES

Startup / Shutdown

Server Logs

Options File

MYSQL ENTERPRISE

Audit Inspector

Online Backup

Backup Recovery

SCHEMAS

Filter objects

country

customer

film

Columns

Query 1

1 SELECT actor_id, actor_id, id

2 actor_id, first_name,

3 actor_id, last_name,

4 actor_id, last_update

5 FROM sakila, actor;

6

7 SELECT 'film', 'film_id',

8 'film', 'title',

9 'film', 'description',

10 'film', 'language_id',

11 'film', 'original_language_id',

12 'film', 'rental_duration',

13 'film', 'rental_rate',

14 'film', 'length',

15 'film', 'replacement_cost',

16 'film', 'rating',

17 'film', 'special_features',

18

Result set: 3118

film_id

title

description

release_year

language_id

original_language_id

rental_rate

rental_rate

length

1 ACADAMY D... A Epic Drama ... 2006 1 00000 6 0.99 86

2 ACADAMY D... A Coming True ... 2006 1 00000 7 2.99 48

3 ACADAMY D... A Antourding ... 2006 1 00000 7 2.99 50

4 ADAPTA PREJ... A Farcful Doc... 2006 1 00000 6 2.99 110

Table History

Users

Console

Filter

Apply

Revert

Topic: SELECT

Syntax:

SELECT

[ALL | DISTINCT |

[DISTINCTROW |

[HIGH_PRIORITY]

[SQL_BUFFER_RESULT]

[SQL_CALC_FOUND_ROWS]

[SQL_NO_CACHE]

[SQL_SMALL_RESULT]

[SQL_WARNINGS]

expr [,

expr ...]

FROM table_references

[WHERE where_condition]

[GROUP BY col_name]

[HAVING where_condition]

[ORDER BY col_name]

[LIMIT row_count

[OFFSET row_count]

]

SELECT

FROM table_references

[WHERE where_condition]

[GROUP BY col_name]

[HAVING where_condition]

[ORDER BY col_name]

[LIMIT row_count

[OFFSET row_count]

]



SQL



SQL

Es un **lenguaje** para manipular (guardar, obtener, actualizar, borrar) datos de una base de datos relacional.

Es el lenguaje utilizado para comunicarse con el gestor de base de datos (DBMS).

SQL = Structured Query Language.

👉 Recomendamos hacer el siguiente tutorial (online e interactivo) sobre SQL:
<https://www.sqlteaching.com>.



SQL – Consultas

A cada porción de código SQL que se utiliza para interactuar con la BD se le llama *consulta*.

Ejemplo:

```
SELECT *  
FROM usuarios;
```



Esta consulta lo que hace es obtener todos los registros (con todos los campos) de la tabla `usuarios`. El texto en azul son palabras clave de SQL. Lo veremos en detalle a continuación.

Nota: En general hay que tener cuidado al hacer consultas que retornen demasiados registros ya que pueden demorar (y se puede trancar todo).



SQL – SELECT (1/2)

Sintaxis:

```
SELECT columna_1, columna_2, ...,columna_n  
FROM nombre_tabla;
```

Toda consulta de tipo `SELECT` retorna como resultado una tabla temporal que contiene los registros resultantes de la consulta.

A esta tabla se le puede llamar "tabla-resultado".



SQL – SELECT (2/2)

Ejemplo:

```
SELECT id, apellido  
FROM usuarios;
```

Retorna:

id	apellido
1	Pérez
2	Sánchez
3	Álvarez

} Tabla *resultado*; es temporal.



SQL – Cláusula WHERE (1/4)

Sintaxis:

```
SELECT columna_1, columna_2, ...,columna_n  
FROM nombre_tabla  
WHERE condición;
```

La consulta retorna como resultado una "*tabla-resultado*", con los registros de *nombre_tabla* que cumplen con la condición.



SQL – Cláusula WHERE (2/4)

Ejemplo:

```
SELECT *  
FROM usuarios  
WHERE apellido = "Pérez";
```

La consulta retorna todos los registros de la tabla `usuarios` que tengan apellido "Pérez".



SQL – Cláusula WHERE (3/4)

Ejemplo:

```
SELECT *  
FROM usuarios  
WHERE telefono = 27065597;
```

La consulta retorna todos los registros de la tabla `usuarios` que tienen asignado el teléfono 27065597.



SQL – Cláusula WHERE (4/4)

Ejemplo:

```
SELECT *  
FROM usuarios  
WHERE birth_date >= "1980-01-01";
```

La consulta retorna todos los registros de la tabla `usuarios` que tienen fecha de nacimiento posterior al 1 de enero de 1980.



SQL – Operador AND

Ejemplo:

```
SELECT *  
FROM usuarios  
WHERE birth_date >= "1980-01-01"  
AND birth_date <= "1989-12-31";
```

La consulta retorna los registros de la tabla `usuarios` que nacieron en al década del '80.



SQL – Operador OR

Ejemplo:

```
SELECT *  
FROM usuarios  
WHERE departamento = "Montevideo"  
OR departamento = "Canelones";
```

La consulta retorna todos los registros de la tabla `usuarios` que tienen asignado el departamento Montevideo o Canelones.

Es decir, el resultado contiene a todos los usuarios de Montevideo y a los de Canelones.



SQL – Operador AND y OR

Ejemplo:

```
SELECT *  
FROM usuarios  
WHERE birth_date >= "1980-01-01"  
AND birth_date <= "1989-31-12"  
AND (departamento = "Montevideo" OR departamento = "Canelones" OR  
departamento = "San José");
```

La consulta retorna los registros de la tabla `usuarios` que hayan nacido en la década del '80 y que sean de Montevideo, Canelones o San José.



SQL – Operador NOT

Ejemplo:

```
SELECT *  
FROM usuarios  
WHERE NOT departamento = "Montevideo";
```

La consulta retorna los registros de la tabla `usuarios` que no tienen asignado el departamento Montevideo.

Nota: Esta consulta no retorna registros que tengan departamento NULL. Esto es así porque MySQL interpreta el NULL como algo desconocido. Por lo tanto, no se puede decir que un registro con departamento NULL es distinto de "Montevideo".



SQL – ORDER BY

Ejemplo:

```
SELECT *  
FROM usuarios  
ORDER BY birth_date DESC;
```

La cláusula `ORDER BY` permite ordenar los registros de una consulta según determinado campo, de forma ascendente (`ASC`) o descendente (`DESC`), según se indique.



SQL – IS NULL

Ejemplo:

```
SELECT *  
FROM usuarios  
WHERE telefono IS NULL;
```



SQL – LIKE

Ejemplo:

```
SELECT *  
FROM usuarios  
WHERE email LIKE "%@gmail.com";
```

Retorna todos los registros de la tabla `usuarios` cuyo email termine en "@gmail.com".

El símbolo de `%` se usa para indicar que en ese lugar puede ir cualquier conjunto de caracteres.



SQL – INSERT INTO (1/2)

Sintaxis:

```
INSERT INTO nombre_tabla (cdb, columna_2, ..., columna_n)  
VALUES (valor_1, valor_2, ..., valor_n);
```

Esta consulta inserta en la tabla `nombre_tabla` un registro nuevo con los valores indicados.



SQL – INSERT INTO (2/2)

Ejemplo:

```
INSERT INTO usuarios (nombre, apellido, telefono)
VALUES ("María", "González", 27065597);
```

Esta consulta inserta en la tabla `usuarios` un registro nuevo con los valores indicados.



Ejercicio 1



Ejercicio 1

1. Instalar MySQL.
2. Instalar alguna GUI de su preferencia (ej: TablePlus o MySQL Workbench), a no ser que se sientan más cómodos trabajando con la CLI.
3. Crear una base de datos llamada `db_test`. Seleccionar como cotejamiento (encoding): `utf8mb4_unicode_ci`. Leer más al respecto [aquí](#).
4. Crear una tabla llamada “usuarios”.

Ejercicio 1 (cont)

5. Crear las siguientes columnas:
 - a. `id` (BIGINT, auto-incremental, clave primaria).
 - b. `nombre` (VARCHAR, longitud 100).
 - c. `apellido` (VARCHAR, longitud 100).
 - d. `edad` (INT).
 - e. `password` (VARCHAR, longitud 150).
6. Todos los datos deben ser **obligatorios**.
7. Insertar 2 usuarios en la base de datos. Verificar que los datos hayan sido ingresados correctamente. ¿Qué se puede decir sobre guardar el campo `password` de esta forma?



Ejercicio 2

Ejercicio 2

Completar el siguiente entrenamiento sobre SQL:

<https://www.sqlteaching.com>.

Este sitio tiene la ventaja de que permite practicar y aprender sobre SQL sin necesidad de instalar nada en su computadora. Es 100% online y gratuito.

👉 Sugerencia: hacer los primeros ejercicios de este sitio en clase (entre todos) y dejar el resto para practicar el fin de semana (o en algún otro tiempo libre).