

**NODE.JS**

**MySQL**

wawiwa

# Base de datos MySQL

MySQL es un sistema de gestión de bases de datos relacionales basado en SQL (lenguaje de consulta estructurado). ... Sin embargo, el uso más común de MySQL es como base de datos web. Se puede utilizar para almacenar cualquier cosa, desde un único registro de información hasta un inventario completo de productos disponibles para una tienda en línea.

[mysql.com](http://mysql.com)

# Descarga MySQL

Ve a [mysql.com](http://mysql.com)

The screenshot shows the MySQL website homepage. A red arrow points from the text "Ve a mysql.com" to the "DOWNLOADS" button in the navigation bar. The page features a large blue header with the text "MySQL Database Service with HeatWave". Below the header, there are two sections: "Lower Total Cost of Ownership" and "Faster Performance", each with a bulleted list of benefits. At the bottom right, there are buttons for "Try Now", "Free Webinars", and "MySQL Enterprise Edition".

MySQL Database Service with HeatWave

Lower Total Cost of Ownership

- 2/3 the cost of Amazon RDS
- 1/2 the cost of Amazon Aurora
- 1/2 the cost of Amazon Redshift AQUA
- 1/5 the cost of Snowflake

Faster Performance

- 5400x Faster than Amazon RDS
- 1400x Faster than Amazon Aurora
- 6.5x Faster than Amazon Redshift AQUA
- 7x Faster than Snowflake

Try Now | > | Free Webinars | MySQL Enterprise Edition

Ve a [mysql.com](http://mysql.com)

MySQL | DOWNLOADS | DOCUMENTATION | DEVELOPER ZONE

Contact MySQL | Login | Register

The world's most popular open source database

MySQL.COM DOWNLOADS

# Descarga el “community server”

Ve a [mysql.com](https://www.mysql.com)

The screenshot shows the MySQL website's main navigation bar at the top, followed by a large central banner for the MySQL Database Service. The banner features a blue background with white text and a large cloud icon. It highlights "Faster Performance" (with HeatWave), "Lower Total Cost of Ownership" (with a list of cost savings), and "Contact Sales". Below the banner, there are sections for "Free Webinars", "MySQL Enterprise Edition", "MySQL Cluster CGE", and "MySQL Community (GPL) Downloads". The footer contains links for "Contact Us Online" and "More Countries".

[Get MySQL](#) | [Log In](#) | [Register](#)

MySQL Database Service

with HeatWave

**Faster Performance**

- 540x Faster than Amazon RDS
- 140x Faster than Amazon Aurora
- 6.5x Faster than Amazon Aurora AQUA
- 7x Faster than Snowflake

[Try Now](#)

**Lower Total Cost of Ownership**

- 2x the cost of Amazon RDS
- 1/2 the cost of Amazon Aurora
- 1/2 the cost of Amazon Redshift AQUA
- 1/5 the cost of Snowflake

**MySQL Enterprise Edition**

MySQL Enterprise Edition includes the most comprehensive set of advanced features, management tools and technical support for MySQL.

[Learn More](#) | [Customize](#) | [Download](#)

**MySQL Cluster CGE**

MySQL Cluster is a real-time open source transactional database designed for fast, always-on access to data under high throughput conditions.

[MySQL Cluster](#) | [MySQL Cluster Manager](#) | [Purchase everything in MySQL Enterprise Edition](#)

**Contact Sales**

Why Every Business Should Use MySQL Enterprise Edition  
MySQL Enterprise Edition  
MySQL Cluster  
MySQL Cluster Manager  
Purchase everything in MySQL Enterprise Edition

U.S.A. +1 800 221 0204  
Canada +1 866 221 0204  
Germany +49 89 143 01 240  
France +33 157 61 3157  
UK +44 207 553 8447

India +91 20 4245 1220  
China +86 10 8841 0613  
India +91 80 2056 0010  
More Countries

**MySQL Community (GPL) Downloads**

Haz click en:MySQL  
Community (GPL) Descargas »

# Descarga el “community server”



Ve a [mysql.com](https://mysql.com)



## ④ MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP
- MySQL Community Server
  - MySQL Cluster
  - MySQL Router
  - MySQL Shell
  - MySQL Workbench
- MySQL Installer for Windows
- MySQL for Visual Studio
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

Descarga: MySQL Community Server



© 2022, Oracle Corporation and/or its affiliates  
[Legal Policies](#) | [Your Privacy Rights](#) | [Terms of Use](#) | [Trademark Policy](#) | [Contributor Agreement](#) | [Cookie Preferences](#)

# Descarga mysql

Ve a [mysql.com](https://mysql.com)

Selecciona tu sistema operativo y descarga:

General Availability (GA) Releases Archives

MySQL Community Server 8.0.28

Select Operating System: macOS

Select OS Version: All

Looking for previous GA versions?

Packages for Big Sur (11) are compatible with Monterey (12)

Version	File Type	Size	MD5	Action
8.0.28	macOS 11 (ARM, 64-bit), DMG Archive	419.9M	MD5: 9fe7f6911decdd4961a45b28a9c95d   Signature	<a href="#">Download</a>
8.0.28	macOS 11 (x86, 64-bit), DMG Archive	425.5M	MD5: 27ee41c26a7644d1ddfe3a35e482617   Signature	<a href="#">Download</a>
8.0.28	macOS 11 (ARM, 64-bit), Compressed TAR Archive	168.5M	MD5: f1943053b12428edc04ed309a636fd0   Signature	<a href="#">Download</a>
8.0.28	macOS 11 (x86, 64-bit), Compressed TAR Archive	168.9M	MD5: b2d5bb57edb9281141fd61c84f1c9d6f   Signature	<a href="#">Download</a>
8.0.28	macOS 11 (ARM, 64-bit), Compressed TAR Archive	251.6M	MD5: 2028-macos11-arm64.tar.gz	<a href="#">Download</a>

# Inicia el servidor

Desde mac :

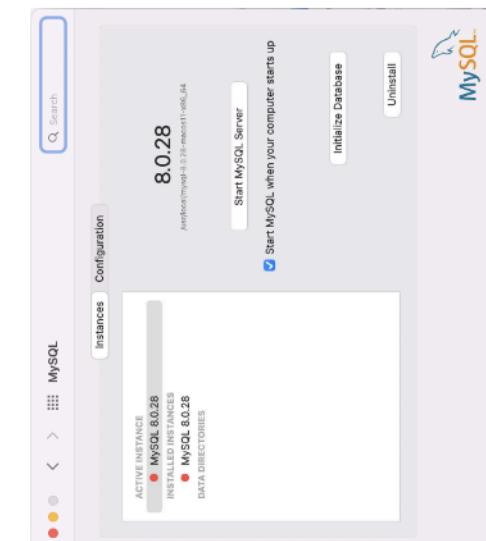
Ve a



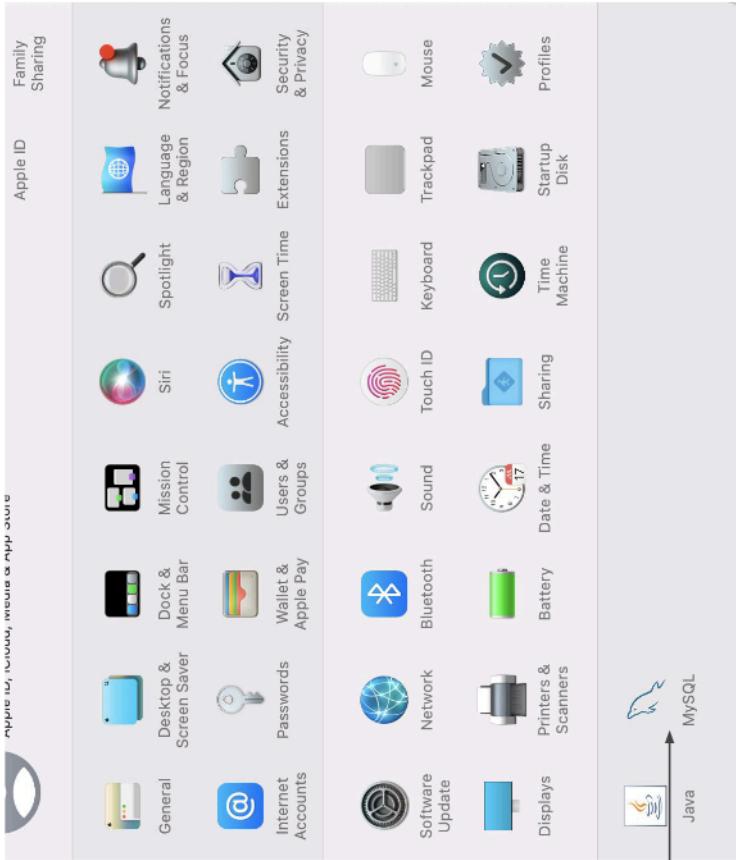
# Inicia el servidor

Apple ID, iCloud, iCloud & App Store		Apple ID				Family Sharing	
	General		Dock & Menu Bar		Mission Control		Siri
	Desktop & Screen Saver		Wallet & Apple Pay		Users & Groups		Spotlight
	Internet Accounts		Passwords		Accessibility		Language & Region
	Software Update		Displays		Bluetooth		Notifications & Focus
	Mission Control		Battery		Sound		Keyboard
	Dock & Menu Bar		Sharing		Touch ID		Trackpad
	Users & Groups		Screen Time		Extensions		Mouse
	Network		Date & Time		Accessibility		Startup Disk
	Passwords		Time Machine		Keyboard		Profiles
	Software Update		Battery		Sound		Mouse
	Displays		Sharing		Keyboard		Trackpad
	Mission Control		Date & Time		Accessibility		Mouse
	Dock & Menu Bar		Time Machine		Keyboard		Profiles
	Users & Groups		Sharing		Keyboard		Mouse
	Network		Date & Time		Keyboard		Profiles
	Passwords		Sharing		Keyboard		Mouse
	Software Update		Time Machine		Keyboard		Profiles
	Displays		Sharing		Keyboard		Mouse
	Mission Control		Date & Time		Keyboard		Profiles
	Dock & Menu Bar		Sharing		Keyboard		Mouse
	Users & Groups		Time Machine		Keyboard		Profiles
	Network		Sharing		Keyboard		Mouse
	Passwords		Time Machine		Keyboard		Profiles
	Software Update		Sharing		Keyboard		Mouse
	Displays		Time Machine		Keyboard		Profiles
	Mission Control		Sharing		Keyboard		Mouse
	Dock & Menu Bar		Time Machine		Keyboard		Profiles
	Users & Groups		Sharing		Keyboard		Mouse
	Network		Time Machine		Keyboard		Profiles
	Passwords		Sharing		Keyboard		Mouse
	Software Update		Time Machine		Keyboard		Profiles
	Displays		Sharing		Keyboard		Mouse
	Mission Control		Time Machine		Keyboard		Profiles
	Dock & Menu Bar		Sharing		Keyboard		Mouse
	Users & Groups		Time Machine		Keyboard		Profiles
	Network		Sharing		Keyboard		Mouse
	Passwords		Time Machine		Keyboard		Profiles
	Software Update		Sharing		Keyboard		Mouse
	Displays		Time Machine		Keyboard		Profiles
	Mission Control		Sharing		Keyboard		Mouse
	Dock & Menu Bar		Time Machine		Keyboard		Profiles
	Users & Groups		Sharing		Keyboard		Mouse
	Network		Time Machine		Keyboard		Profiles
	Passwords		Time Machine		Keyboard		Mouse
	Software Update		Sharing		Keyboard		Profiles
	Displays		Time Machine		Keyboard		Mouse
	Mission Control		Sharing		Keyboard		Profiles
	Dock & Menu Bar		Time Machine		Keyboard		Profiles
	Users & Groups		Sharing		Keyboard		Mouse
	Network		Time Machine		Keyboard		Profiles
	Passwords		Time Machine		Keyboard		Mouse
	Software Update		Sharing		Keyboard		Profiles
	Displays		Time Machine		Keyboard		Mouse
	Mission Control		Sharing		Keyboard		Profiles
	Dock & Menu Bar		Time Machine		Keyboard		Profiles
	Users & Groups		Sharing		Keyboard		Mouse
	Network		Time Machine		Keyboard		Profiles
	Passwords		Time Machine		Keyboard		Mouse
	Software Update		Sharing		Keyboard		Profiles
	Displays		Time Machine		Keyboard		Mouse
	Mission Control		Sharing		Keyboard		Profiles
	Dock & Menu Bar		Time Machine		Keyboard		Profiles
	Users & Groups		Sharing		Keyboard		Mouse
	Network		Time Machine		Keyboard		Profiles
	Passwords		Time Machine		Keyboard		Mouse
	Software Update		Sharing		Keyboard		Profiles
	Displays		Time Machine		Keyboard		Mouse
	Mission Control		Sharing		Keyboard		Profiles
	Dock & Menu Bar		Time Machine		Keyboard		Profiles
	Users & Groups		Sharing		Keyboard		Mouse
	Network		Time Machine		Keyboard		Profiles
	Passwords		Time Machine		Keyboard		Mouse
	Software Update		Sharing		Keyboard		Profiles
	Displays		Time Machine		Keyboard		Mouse
	Mission Control		Sharing		Keyboard		Profiles
	Dock & Menu Bar		Time Machine		Keyboard		Profiles
	Users & Groups		Sharing		Keyboard		Mouse
	Network		Time Machine		Keyboard	<img alt="Profiles icon" data-bbox="288 328 328 35	

# Inicia el servidor

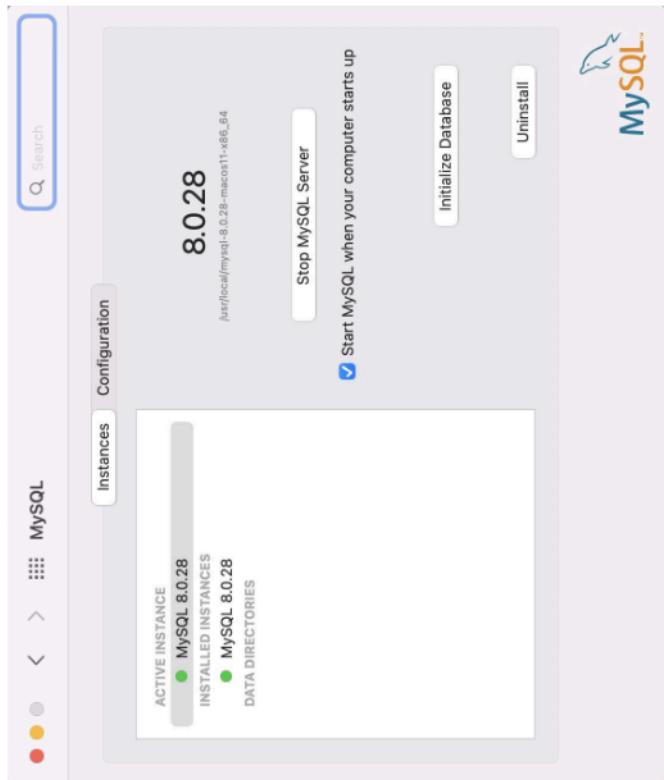


Click on mysql



# Después de instalar, inicial el servidor

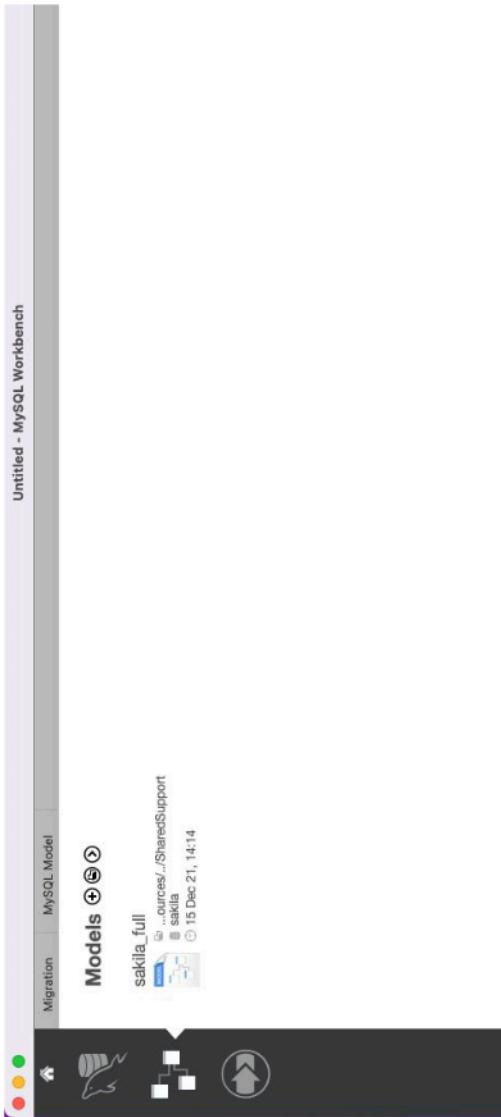
Ve a [mysql.com](http://mysql.com)



# MySQL - descarga workbench

<https://dev.mysql.com/downloads/workbench/>

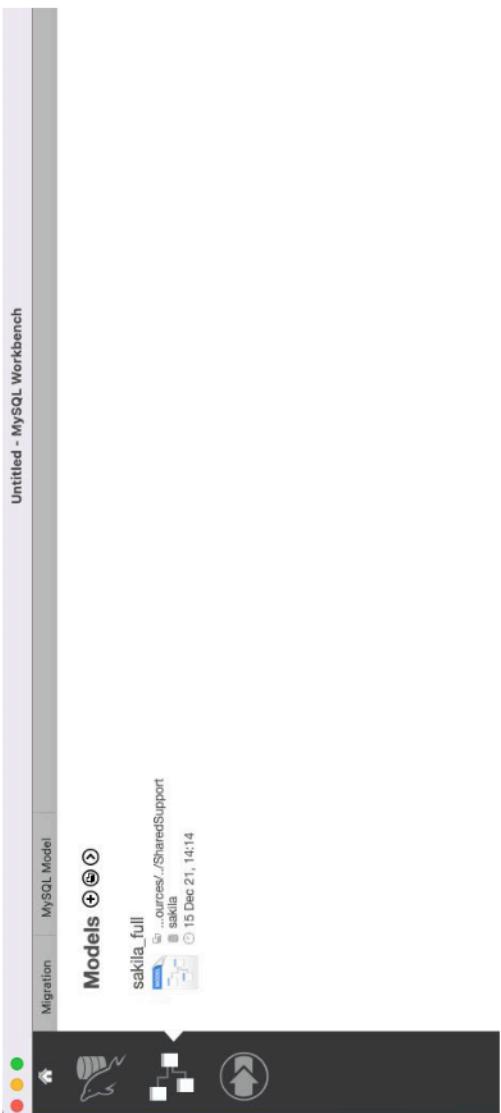
Después de instalarlo, ábrelo:



# MySQL

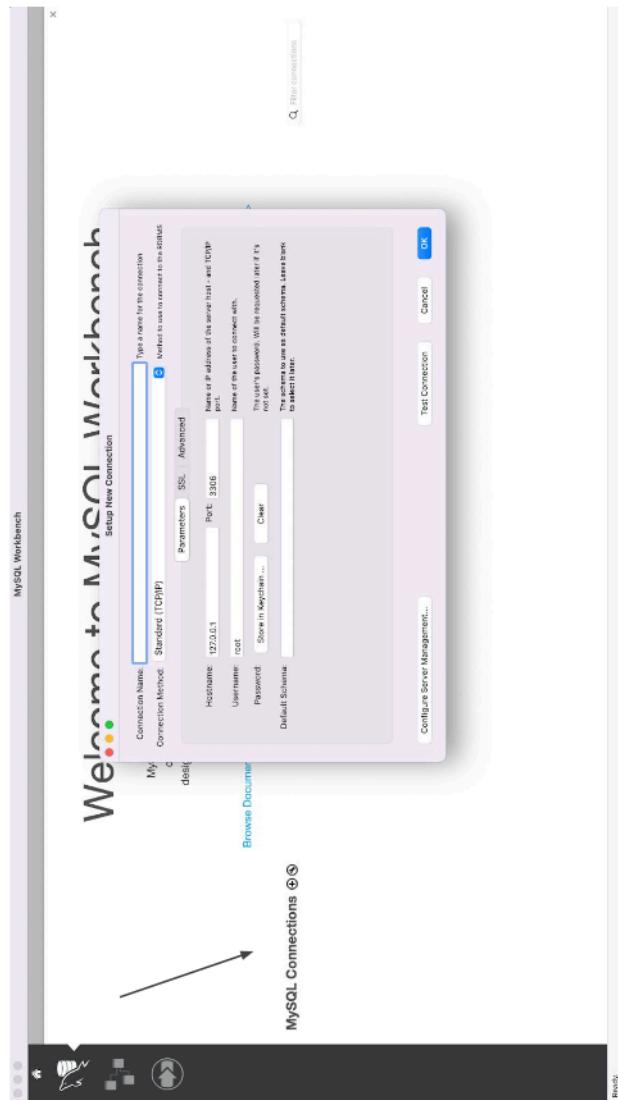
<https://dev.mysql.com/downloads/workbench/>

Después de instalarlo, ábrelo:



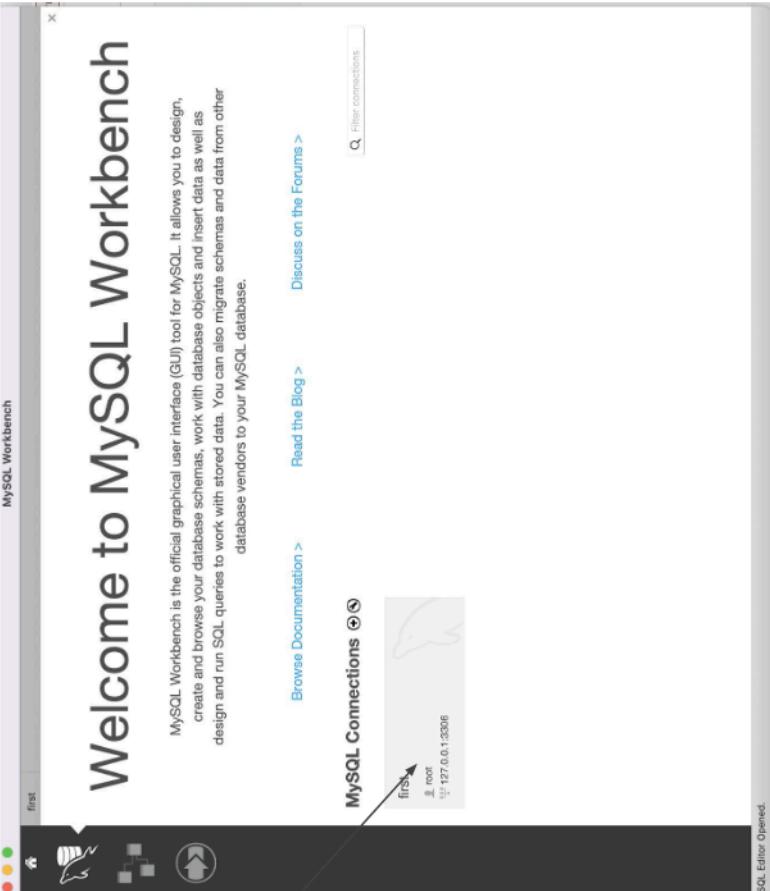
# Agrega un módulo

Haz click en el ícono + y agrega un módulo



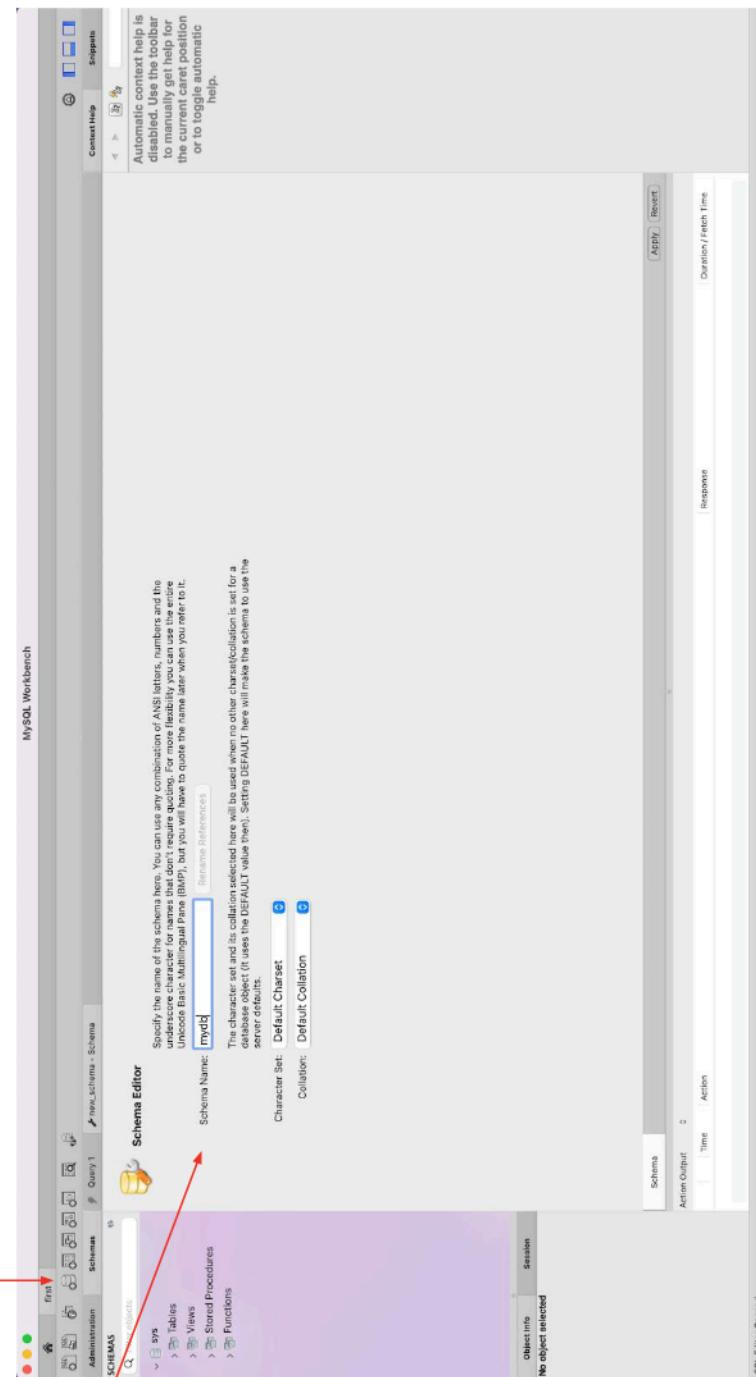
# Agrega una conexión

Da click en connection

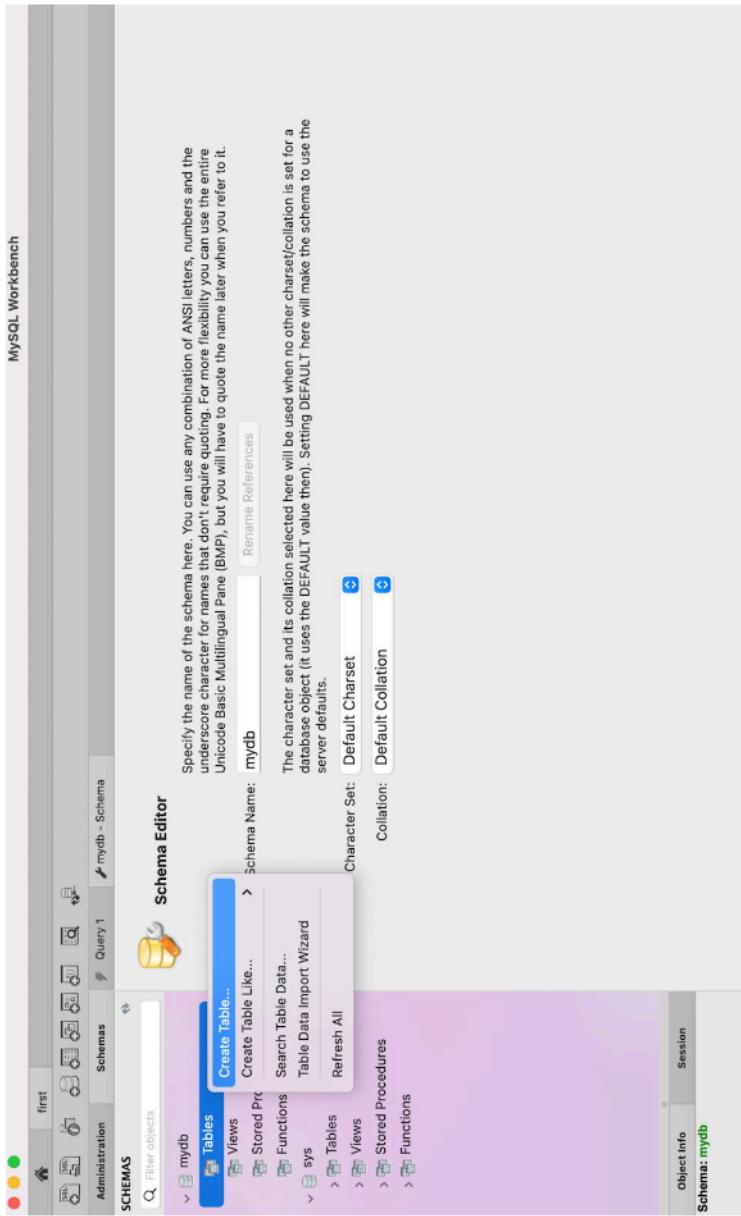


# Crea una base de datos

Crea base  
de datos



# Crea una tabla



# Aggrega campos

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Administration' is selected under 'Schemas'. A search bar at the top right contains the text 'Name: tblPerson'. Below the schema list, there's a table for defining the new table 'tblPerson'.

Column	Datatype	PK	NN	UQ	B..	UN	ZF	AI	G	Default / Expression
PersonId	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<click to edit>				
fname	VARCHAR(45)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
lname	VARCHAR(45)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
salary	VARCHAR(45)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

The 'salary' column has a tooltip: '<click to edit> INT VARCHAR() DECIMAL() DATETIME BLOB'. The table also includes a 'Comments:' section and tabs for 'Object Info' and 'Session'.

# Aggrega campos



MySQL Workbench

Query 1 mydb - Schema Name: tblPerson

Review SQL Script to be Applied on the Database

Please review the following SQL script that will be applied to the database. Note that once applied, these statements may not be revertible without losing some of the data. You can also manually change the SQL statements before execution.

Online DDL Algorithm: Default Lock Type: Default

```
1 CREATE TABLE `tblPerson` (
  `personId` INT NOT NULL AUTO_INCREMENT,
  `firstname` VARCHAR(45) NULL,
  `lastname` VARCHAR(45) NULL,
  `salary` VARCHAR(45) NULL,
  PRIMARY KEY (`personId`));

```

<click to edit>

Column details 'salary'

Column Name: salary  
Charset/Collation: Default Charset  
Comments:

STORED  
Key  
Not NULL  
Unsigned  
Generate  
Default

100% ◁ 1:1

Go Back Apply



# NODEJS Y MYSQL

# Crea una conexión

```
const express = require('express')
const mysql = require('mysql');
const app = express()
const PORT = process.env.PORT || 3000
app.get('/', (req, res) => {
  res.send('Hello Heroku!')
})

const con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "asaf7452702"
})

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
}) ;

app.listen(PORT, () => {
  console.log(`Example app listening on port ${PORT}`)
})
```

```
asafamir@Asafs-MBP chap18-heroku-app % node server
Example app listening on port 3000
Connected!
```

# Crea una base de datos

Creamos una base de datos en MySQL, pero también lo puedes hacer con código desde node.

```
con.connect(function(err) {  
  if (err) throw err;  
  
  console.log("Connected!");  
  
  con.query("CREATE DATABASE mydb", function (err, result) {  
    if (err) throw err;  
  
    console.log("Database created");  
  });  
});
```

# Crea una tabla

Creamos una tblPerson en MySQL, pero también lo puedes hacer con código desde node.

```
con.connect(function(err) {  
  if (err) throw err;  
  console.log("Connected!");  
  createTablePerson()  
});  
  
function createTablePerson() {  
  let sql = `CREATE TABLE IF NOT EXISTS mydb.tblPersons (personId INT  
NOT NULL AUTO_INCREMENT,  
firstname VARCHAR(45) NULL, lastname VARCHAR(45) NULL,  
salary INT NULL, PRIMARY KEY (personId))`;  
  con.query(sql, function (err, result) {  
    if (err) throw err;  
    console.log("Table created");  
  });  
}
```

**Inserta un registro**

```
try {
    let sql = `INSERT INTO mydb.tblPersons (firstname, lastname, salary) VALUES ('${req.body.firstname}', '${req.body.lastname}', ${req.body.salary})`;
    con.query(sql, function (err, result) {
        if (err) throw err;
        res.status(201).json({
            status: "success",
            data: result
        });
    });
}

catch (err) {
    res.status(400).json({
        status: "fail",
        message: "error: " + err
    });
}
```

# Actualiza un registro

```
app.patch('/api/v1/persons', function(req, res, next) {
  console.log(req.body)

  try{
    const sql = `UPDATE mydb.tblPersons SET firstname = '${req.body.firstname}' WHERE
    personId = ${req.body.personId}`;
    console.log(sql)
    con.query(sql, function (err, result) {
      if (err) throw err;
      res.status(201).json({
        status: "success",
        data:result
      })
    });
  }
  catch(err){
    res.status(400).json({
      status: "fail",
      message: "error: " + err
    })
  }
});
```

# Antes de actualizar el registro

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'mydb' schema with the 'Tables' node selected. The main area shows the 'tblPersons' table with 1 row. A context menu is open over the table, with the 'Select Rows - Limit 1000' option highlighted. Other options in the menu include 'Table Inspector', 'Copy to Clipboard', 'Table Data Export Wizard', 'Table Data Import Wizard', 'Send to SQL Editor', 'Create Table...', 'Alter Table...', 'Table Maintenance...', 'Drop Table...', 'Truncate Table...', 'Search Table Data...', 'Refresh All', and 'Object Info'.

personId	firstname	lastname	salary
1	Mikel	Lewis	100
2	David	DeKlin	200

# Selecciona todos los registros

```
app.get('/api/v1/persons', (req, res) => {
  try {
    let sql = `SELECT * FROM mydb.tblPersons`;
    con.query(sql, function (err, result) {
      if (err) throw err;
      res.status(201).json({
        status: "success",
        data: result
      })
    });
  }
  catch(err){
    res.status(400).json({
      status: "fail",
      message: "error: " + err
    })
  }
});
```

localhost:3000/api/v1/persons/

localhost:3000/api/v1/persons/

GET

Params Authorization Headers (7)

none form-data x-www-form-urlencoded

Body

Pre-request Script Tests Settings

raw binary GraphQL

This request does not have a body

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1   "status": "success",
2   "data": [
3     {
4       "personId": 1,
5       "firstname": "Ron",
6       "lastname": "Lewis",
7       "salary": 100
8     },
9     {
10       "personId": 2,
11       "firstname": "David",
12       "lastname": "Debkin",
13       "salary": 200
14     },
15     {
16       "personId": 3,
17       "firstname": "John",
18       "lastname": "Smith",
19       "salary": 300
20     }
21   ]
```

# Borra

```

app.delete('/api/v1/persons', function(req, res, next) {
  console.log(req.body)

  try{
    const sql = `delete from mydb.tblPersons WHERE personId = ${req.body.personId}`;

    con.query(sql, function (err, result) {
      if (err) throw err;
      res.status(201).json({
        status: "success",
        data:result
      })
    });
  }
}

catch(err){
  res.status(400).json({
    status:"fail",
    message:"error: 😞" + err
  })
}

```

localhost:3000/api/v1/persons/

**DELETE** localhost:3000/api/v1/persons/

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON ↻

1	{
2	"personId":1
3	}

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ↻

1	{"status": "success", "data": {}}
2	"fieldCount": 0,
3	"affectedRows": 1,
4	"insertId": 0,
5	"serverStatus": 2,
6	"warningCount": 0,
7	"message": "",
8	"protoCol": 141,
9	"changedRows": 0}
10	
11	
12	
13	

# Borra

```

app.delete('/api/v1/persons', function(req, res, next) {
  console.log(req.body)

  try{
    const sql = `delete from mydb.tblPersons WHERE personId = ${req.body.personId}`;

    con.query(sql, function (err, result) {
      if (err) throw err;
      res.status(201).json({
        status: "success",
        data:result
      })
    });
  }
  catch(err){
    res.status(400).json({
      status:"fail",
      status:"Fail",
      message:"error: "+ err
    })
  }
});

```

localhost:3000/api/v1/persons/

**DELETE** localhost:3000/api/v1/persons/

Authorization:  Headers (9)  Body  Pre-request Script  Tests  Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON

1

2

3

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

1 {  
2 "status": "success",  
3 "data": {  
4 "fieldCount": 0,  
5 "affectedRows": 1,  
6 "insertId": 0,  
7 "serverStatus": 2,  
8 "warningCount": 0,  
9 "message": "",  
10 "protoInfo": true,  
11 "changesRows": 0  
12 },  
13}

# Más ejemplos

```
con.query("SELECT * FROM mydb.tblPersons ORDER BY firstname", function (err, result) {}  
  
con.query("DROP TABLE IF EXISTS mydb.tblPersons", function (err, result) {}  
  
con.query("SELECT * FROM mydb.tblPersons LIMIT 10\"", function (err, result) {}
```

## Ejemplos de “Join”

```
con.query("SELECT mydb.tblPersons.firstname AS firstname,  
mydb.tblPersons.lastname AS lastname FROM mydb.tblPersons JOIN  
cities ON mydb.tblPersons.cityId = mydb.cities.cityId", function  
(err, result) {}
```

# Hazlo tú mismo

Ingresá la siguiente tabla en MySQL,  
tblProduct (productId, name, description, price, category, quantity)

Escribe las siguientes solicitudes en node.js:

1. Ingresá 5 “products” en la tabla agregando un comando
2. Muestra todos los “products” en la tabla.
3. Ve todas las “categories”
4. Muestra solamente “productId” y “price” .
5. Ve todos los “products” en la “category” 8
6. Ve todos los “products” que su código es 10 o 20 y su “price” entre 100 y 200
7. Ve todos los “products” ordenados por su “name” .
8. Borra todos los “products” que su “quantity” sea igual a 0

wawiwa

¿Preguntas?