

Package database/sql

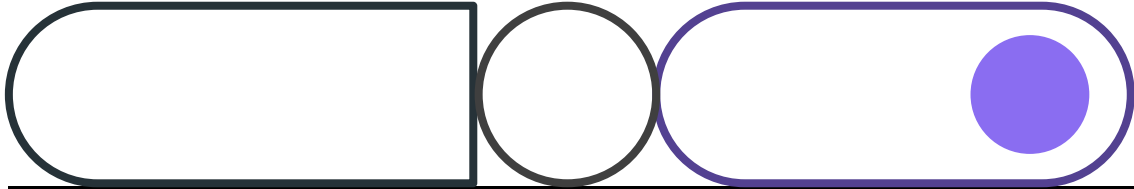
Índice

- 01** [Definición](#)
- 02** [Implementación](#)
- 03** [Consideraciones generales](#)

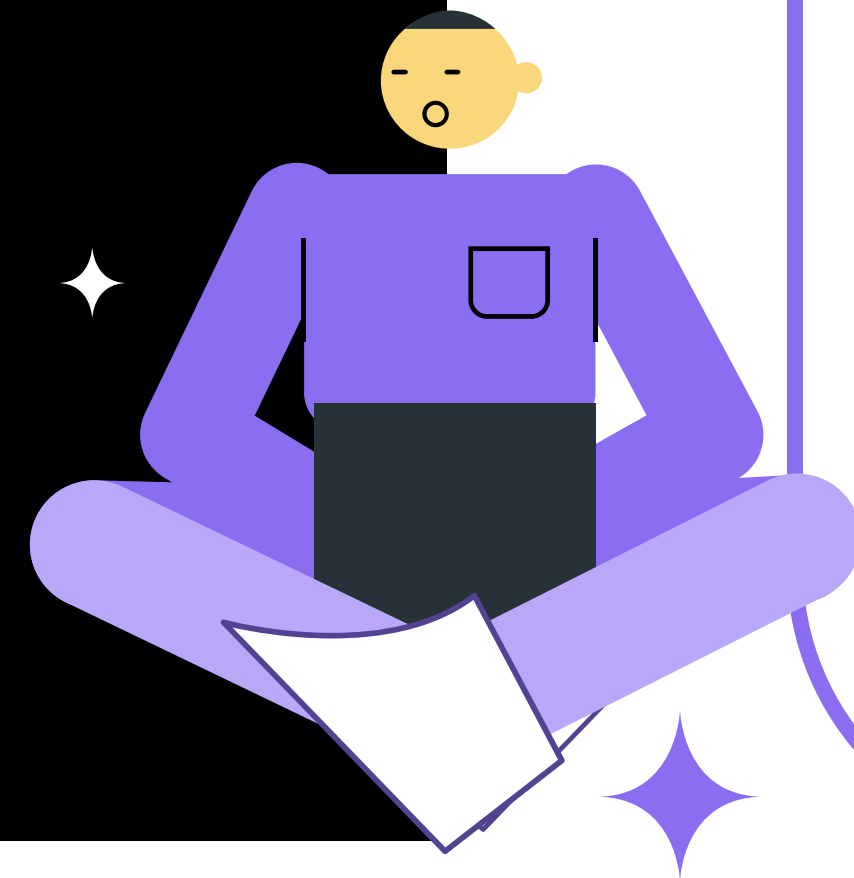


01

Definición



El paquete **database/sql** es una implementación nativa de Go, que expone una interfaz que permite gestionar la conexión y datos de una base de datos.



Características

- Forma parte de las librerías estándar de Go.
- Compatible con bases de datos SQL.
- Debe ser usado conjuntamente con un driver de base de datos.
- Hace uso del type **sql.DB** para gestionar la conexión y la ejecución.
- Puede generar una conexión o administrar un pool de conexiones.



02

Implementación



Instalación

Para usar database/sql necesitamos instalar el **driver correspondiente a la BD con la que vayamos a conectar**. En este caso, vamos a hacer uso de **MySQL**. Por lo tanto, hacemos un:

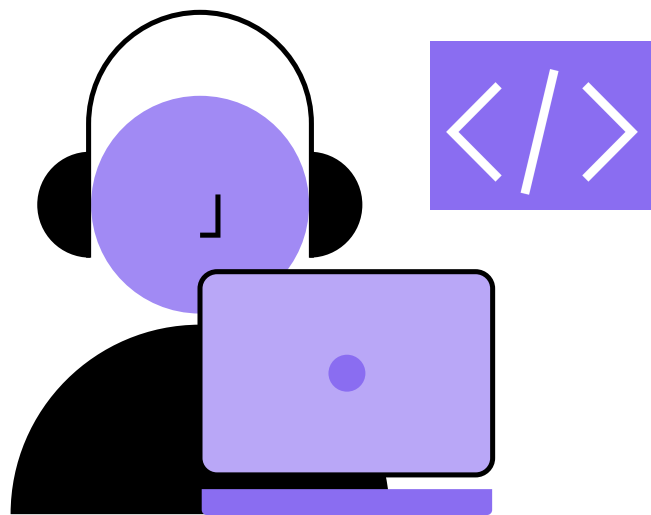
```
$ go get "github.com/go-sql-driver/mysql"
```

Al momento de importarlo, debemos hacerlo anteponiendo el guion bajo (_) al import:

```
{ } import (  
    "database/sql"  
    _ "github.com/go-sql-driver/mysql"  
)
```

Implementación

La función **Open** del package `sql` recibe por parámetro el nombre del driver (por eso debemos importar el driver), y los datos de conexión de la BD, y retorna justamente una conexión y un error, si lo hubiera.

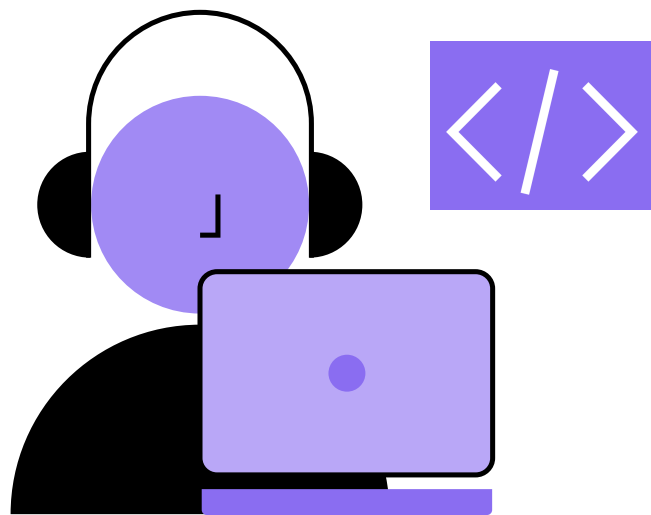


```
import (  
    "database/sql"  
  
    _ "github.com/go-sql-driver/mysql"  
)  
  
func init() {  
    dataSource := "user:pass@tcp(server:Port)/dbName"  
    // Open inicia un pool de conexiones. Solo abrir una vez  
    storageDB, err = sql.Open("mysql", dataSource)  
    if err != nil {  
        panic(err)  
    }  
}
```


Implementación

Veamos el siguiente ejemplo. En la función de inicialización se establece la conexión y se exporta la variable **StorageDB** como puntero del **type *sql.DB** creado con los datos de nuestra conexión.

De esta manera, cuando necesitemos operar la BD, solo tenemos que llamar a la variable **StorageDB**.



```
package db

import (
    "database/sql"
    "log"

    _ "github.com/go-sql-driver/mysql"
)

var (
    StorageDB *sql.DB
)

func init() {
    dataSource := "root:abcd1234@tcp(localhost:3306)/storage"
    // Open inicia un pool de conexiones. Solo abrir una vez
    var err error
    StorageDB, err = sql.Open("mysql", dataSource)
    if err != nil {
        panic(err)
    }
    if err = StorageDB.Ping(); err != nil {
        panic(err)
    }
    log.Println("database Configured")
}
```



Implementación

Los datos requeridos para la conexión a la BD son: **DB User**, **Password**, **Server**, **Port**, **DB Name**. Podemos definir los datos de conexión de la BD en una variable de tipo string. Veamos un ejemplo:



03

Consideraciones generales

Consideraciones generales

01

El package se encarga de gestionar el pool de conexiones. No debemos reutilizar el método **Open**, ya que esto generará leaks de memoria y saturación de conexiones a la base.

02

Es una buena práctica dejar la creación de la conexión en un archivo aparte.

03

Cada motor de base de datos opera de forma distinta, a pesar de que tengan elementos de funcionamiento en común. Por eso debemos siempre importar el driver correspondiente.

¡Muchas gracias!