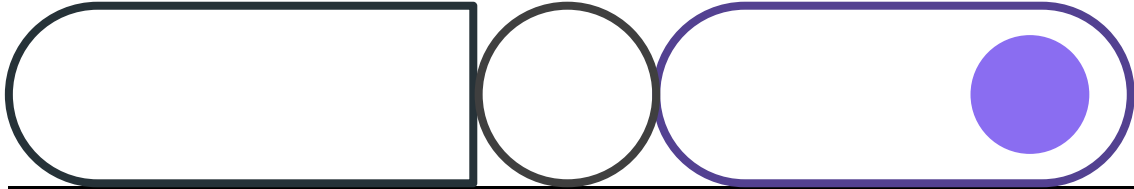
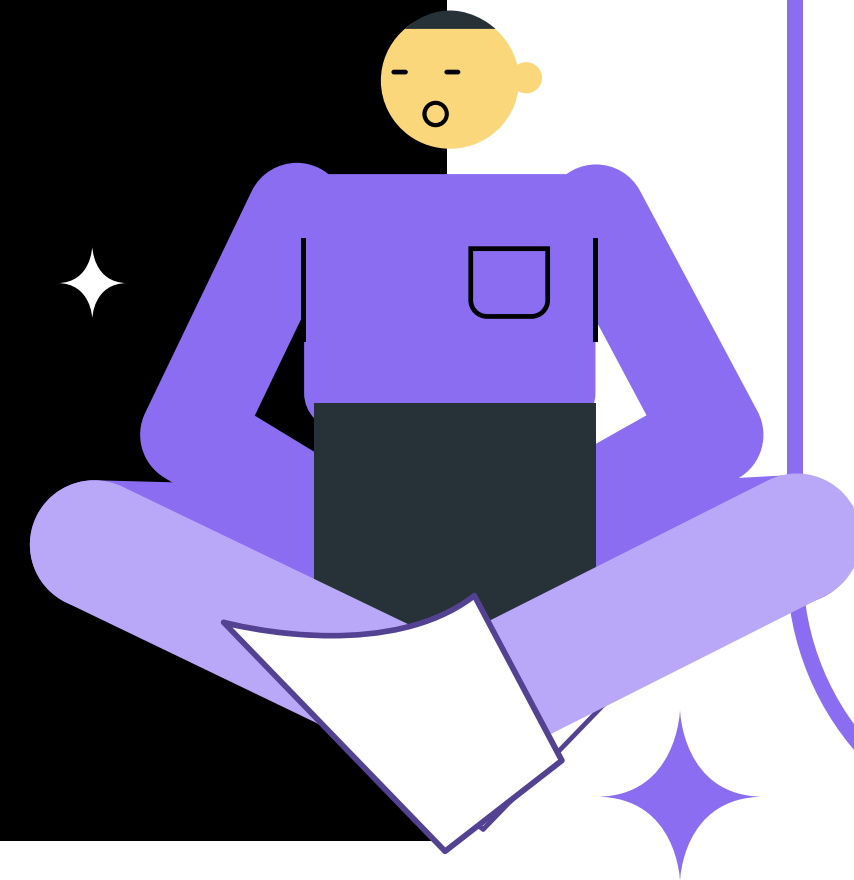


Parámetros



Un **parámetro** es una variable (en este caso, en la URL) utilizada para recibir valores. Estos valores los podemos utilizar dentro de nuestra aplicación.



Path params

Una empresa posee una lista de empleados y queremos consultar la información de estos de acuerdo a su ID. Si dirigimos la consulta al path “/**empleados/:id**”, sería como el siguiente ejemplo:

→ <http://www.empresa.com/empleados/11>

→ <http://www.empresa.com/empleados/24>

Estamos recibiendo del cliente un dato como parámetro sin nombre y le estamos dando un nombre “**id**” al recibirlo en nuestra ruta.



En el ejemplo de arriba, al recibir el valor “11”, nuestro router lo convierte en el parámetro “**id**” cuyo valor será “11”.

Query params

Ahora bien, los parámetros de ruta (**path params**) nos resultan útiles hasta cierto punto. ¿Cómo podemos aclarar que queremos una información que cumpla determinados requerimientos?

Supongamos que tenemos un servidor de una tienda virtual y queremos obtener de nuestra colección de artículos aquel ítem cuyo código sea “**12742**”:

→ <http://www.mitienda.com/articulos?codigo=12742>

Vemos que se parece a una web como las que vimos en ejemplos anteriores, pero aquí hay un detalle adicional: el signo de pregunta (?).

Query params

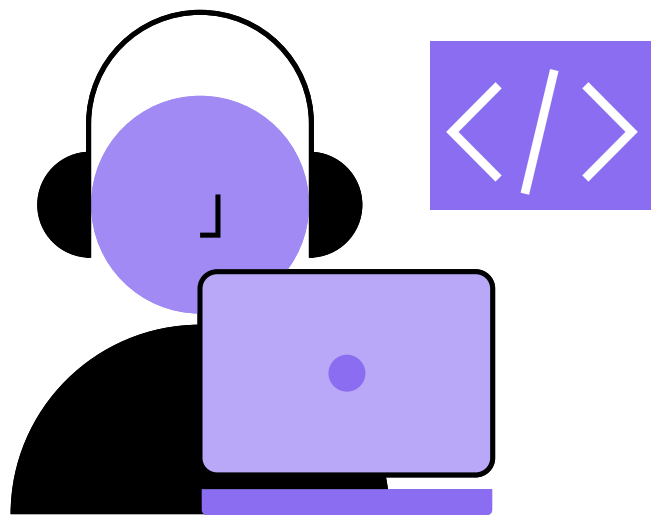
Lo que denota este signo de pregunta es el componente **query** de la URL que se pasa al servidor o recurso de gateway. Este componente se acompaña del path de la URL identificando al recurso.

Toda query se compone siempre de una clave y un valor. Si deseamos enviar más de una dupla clave/valor, debemos separar los pares por un ampersand (&), como vemos en el siguiente ejemplo:

→ <http://www.mitienda.com/check-inventario.com?item=28&marca=Amaizing>

Ejemplo de path params

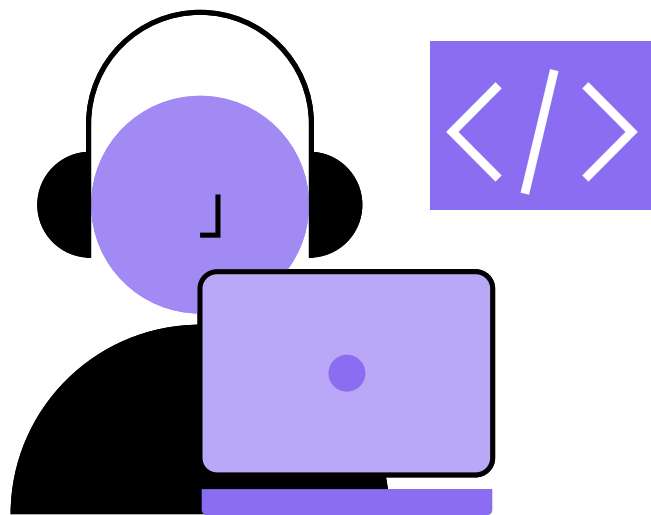
Ahora bien, vimos qué es una query y qué son los paths. Veamos cómo podemos hacerlo funcionar con **Gin**.



```
//Definimos una pseudo base de datos donde consultaremos la
información.
var empleados = map[string]string{
    "644"    : "Empleado A",
    "755"    : "Empleado B",
    "777"    : "Empleado C",
}

func main() {
    server := gin.Default()
    server.GET("/", PaginaPrincipal)
    server.GET("/empleados/:id", BuscarEmpleado)
    server.Run(":8085")
}
```

Ejemplo de path params #2

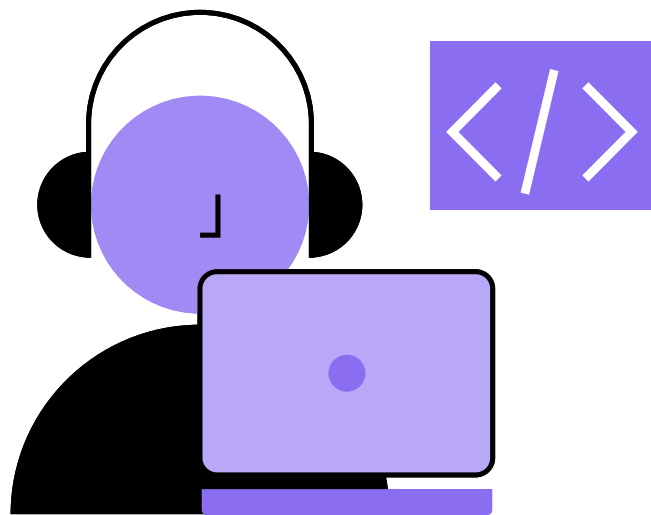


```
//Este handler se encargará de responder a /.
func PaginaPrincipal(ctxt *gin.Context) {
    ctxt.String(200, "¡Bienvenido a la Empresa Gophers!")
}

//Este handler verificará si la id que pasa el cliente existe
en nuestra base de datos.
func BuscarEmpleado(ctxt *gin.Context) {
    empleado, ok := empleados[ctxt.Param("id")]
    if ok{
        ctxt.String(200, "Información del empleado %s, nombre:
%s", ctxt.Param("id"), empleado)
    } else{
        ctxt.String(404, "Información del empleado ¡No existe!")
    }
}
```

¿Y si mi param no existe en la URL?

Aquí vemos cómo Gin maneja los parámetros de una request. El método **Param** es un atajo a **ParamsByName** y devuelve el valor del primer parámetro cuya clave coincide con el nombre dado. Si no se encuentra ningún parámetro coincidente, se devuelve una cadena vacía.



```
//Retorna el valor del "param" de La URL.  
//Es un atajo para c.ParamsByName(key)  
func (c *Context) Param(key string) string {  
    return c.ParamsByName(key)  
}  
  
//Definición de "Params"  
type Params []Param  
  
// Es un slice de parámetros dados por La URL.  
//Estos "Param" ocupan el mismo orden que en La URL.  
  
type Param struct{  
    Key string  
    Value string  
}
```




Trabajemos con los query params

La forma más sencilla de acceder a aquellos parámetros que hayan sido provistos como query params (aquellos con la forma **?clave=valor**), es con el método **Query** de nuestro context:

```
{ } valorDeNuestraKey := ctx.Query("key")
```

¡Muchas gracias!