Material de consulta: Query Parameters (Query Params)

¿Qué son los Query Params?

Los *query parameters* o parámetros de consulta son una parte importante de las URL en las solicitudes HTTP. Se utilizan para enviar información adicional al servidor en la URL, que se puede usar para personalizar la respuesta. Estos parámetros aparecen después del signo de interrogación `?` y se separan por el símbolo `&`.

Por ejemplo, en la URL:

```
http://localhost:1235/api/products/search=laptop&minPrice=500&maxPrice=1500&page=2&limit=5
```

- `search=laptop`
- `minPrice=500`
- `maxPrice=1500`
- `page=2`
- `limit=5`

Los parámetros de consulta proporcionan valores clave que el servidor puede usar para filtrar, ordenar, paginar, o personalizar los datos de la solicitud.

¿Cuándo se podrían utilizar?

Los query params son útiles en una variedad de casos. Aquí hay algunos ejemplos comunes:

1. **Filtrar resultados**: Si tienes una lista de elementos (productos, usuarios, etc.), puedes permitir que el usuario filtre esos resultados según ciertos criterios, como precio, categoría, nombre, etc.
2. **Paginación**: Si tienes una lista grande de resultados, puedes permitir que el usuario pagine los datos para no sobrecargar la respuesta con demasiados elementos a la vez.
3. **Ordenar resultados**: Puedes ordenar los resultados de la consulta, por ejemplo, por precio, fecha de creación, o cualquier otro atributo.
4. **Buscar o hacer consultas textuales**: Puedes permitir a los usuarios buscar datos específicos que coincidan con un término de búsqueda.

Ejemplo práctico: Buscando productos con parámetros de consulta

En el siguiente código, la función `searchProducts` busca productos en una base de datos de acuerdo con los parámetros enviados a través de la URL. Estos parámetros pueden incluir una búsqueda de texto, un rango de precios, paginación, y la opción de ordenar los resultados.

```
const searchProducts = async (req, res) => {
  try {
    // Obtenemos los parámetros de consulta
    const { search, minPrice, maxPrice, page = 1, limit = 3, sort =
'price_asc' } = req.query;

    const query = {};

    // Filtro por búsqueda de texto (si se proporciona un término de
búsqueda)
    if (search) {
      query.name = { $regex: search, $options: 'i' };  // Búsqueda
insensible a mayúsculas y minúsculas
    }

    // Filtro por rango de precios (si se proporciona minPrice o
maxPrice)
    if (minPrice || maxPrice) {
      query.price = {};
      if (minPrice) query.price.$gte = +minPrice;  // mayor o igual
que minPrice
      if (maxPrice) query.price.$lte = +maxPrice;  // menor o igual
que maxPrice
    }

    // Ordenar los resultados por precio
    const sortOrder = sort === 'price_asc' ? { price: 1 } : { price:
-1 };

    // Realizamos la consulta con los filtros, paginación y orden
    const products = await Product.find(query)
      .sort(sortOrder)
      .skip((page - 1) * limit)  // Paginación (salta los productos
anteriores a la página actual)
      .limit(+limit);  // Límite de productos por página

    console.log(query);  // Muestra el query para ver los parámetros
utilizados

    res.status(200).json(products);  // Devuelve los productos
encontrados en formato JSON
```

```
  } catch (error) {
    res.status(500).json({ message: "Error al buscar productos",
error: error.message });
  }
};
```

Descripción del código:

Obteniendo los parámetros de consulta:

```
const { search, minPrice, maxPrice, page = 1, limit = 3, sort =
'price_asc' } = req.query;
```

1. Aquí se extraen los parámetros de consulta de la solicitud. Si no se proporcionan, se usan valores predeterminados (`page = 1`, `limit = 3`, `sort = 'price_asc'`).
2. **Creando la consulta para la base de datos:**
    - Si se proporciona un término de búsqueda, se crea un filtro utilizando una expresión regular para buscar productos cuyo nombre coincida con el término dado.
    - Si se proporciona un rango de precios (`minPrice` o `maxPrice`), se agrega un filtro para el precio de los productos.
3. **Ordenando los resultados:** Se ordenan los productos según el parámetro `sort`, que puede ser `price_asc` (de menor a mayor precio) o `price_desc` (de mayor a menor precio).
4. **Paginación:** Usando los parámetros `page` y `limit`, se determina cuántos productos mostrar en una página. Se usa `skip` para omitir los productos anteriores a la página actual y `limit` para limitar la cantidad de productos por página.

¿Cómo se usan los Query Params en una URL?

Si quisieras buscar productos con el término "laptop" que cuesten entre $500 y $1500, en la página 2, limitando los resultados a 5 productos y ordenados por precio descendente, la URL podría ser:

```
http://localhost:1235/api/products/search?search=laptop&minPrice=500
&maxPrice=1500&page=2&limit=5&sort=price_desc
```

Resumen

- **Query Params** permiten personalizar las solicitudes HTTP mediante la inclusión de parámetros en la URL.
- Son útiles para **filtrar**, **ordenar**, **buscar** y **paginación** de datos.
- Se extraen en el servidor con `req.query` y se pueden usar para ajustar la consulta a la base de datos.