

Nombre: Yomar Guzman

## 2do Parcial TECNOWEB II

Agregar un comentario:

GraphiQL

Prettify

Merge

Copy

History

```
1 # Welcome to GraphiQL
2 #
3 # GraphiQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeaheads aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that start
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 # {
16 #   field(arg: "value") {
17 #     subfield
18 #   }
19 # }
20 #
21 # Keyboard shortcuts:
22 #
23 # Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
24 #
25 # Merge Query:    Shift-Ctrl-M (or press the merge button above)
26 #
27 # Run Query:      Ctrl-Enter (or press the play button above)
28 #
29 # Auto Complete:  Ctrl-Space (or just start typing)
30 #
31
32 > mutation {
33   agregarComentario(
34     comentario: "Comentario desde GraphQL",
35     documento_id_documento: 2,
36     fecha: "2025-05-03",
37     publicado: true
38   ) {
39     id_comentario
40     comentario
41     documento_id_documento
42     fecha
43     publicado
44   }
45 }
46
```

query

mutation

subscription

fragment

{

Self descriptive.

```
{
  "data": {
    "agregarComentario": {
      "id_comentario": 1,
      "comentario": "Comentario desde GraphQL",
      "documento_id_documento": 2,
      "fecha": "1746230400000",
      "publicado": true
    }
  }
}
```

## Obtener todos los comentarios

GraphiQL

Prettify

Merge

Copy

History

```
1 # Welcome to GraphiQL
2 #
3 # GraphiQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeheads aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that start
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 # {
16 #   field(arg: "value") {
17 #     subfield
18 #   }
19 # }
20 #
21 # Keyboard shortcuts:
22 #
23 # Prettify Query: Shift-Ctrl-P (or press the prettify button above)
24 #
25 # Merge Query: Shift-Ctrl-M (or press the merge button above)
26 #
27 # Run Query: Ctrl-Enter (or press the play button above)
28 #
29 # Auto Complete: Ctrl-Space (or just start typing)
30 #
31
32 query {
33   comentarios {
34     id_comentario
35     comentario
36     documento_id_documento
37     fecha
38     publicado
39     isdeleted
40   }
41 }
42
43 query
mutation
subscription
fragment
{
Self descriptive.
```

```
{
  "data": {
    "comentarios": [
      {
        "id_comentario": 1,
        "comentario": "Comentario desde GraphQL",
        "documento_id_documento": 2,
        "fecha": "1746230400000",
        "publicado": true,
        "isdeleted": false
      },
      {
        "id_comentario": 2,
        "comentario": "Comentario desde GraphQL",
        "documento_id_documento": 2,
        "fecha": "1746230400000",
        "publicado": true,
        "isdeleted": false
      },
      {
        "id_comentario": 3,
        "comentario": "Comentario 3",
        "documento_id_documento": 2,
        "fecha": "1746230400000",
        "publicado": true,
        "isdeleted": false
      }
    ]
  }
}
```

## Obtener comentario por ID

GraphiQL

Prettify

Merge

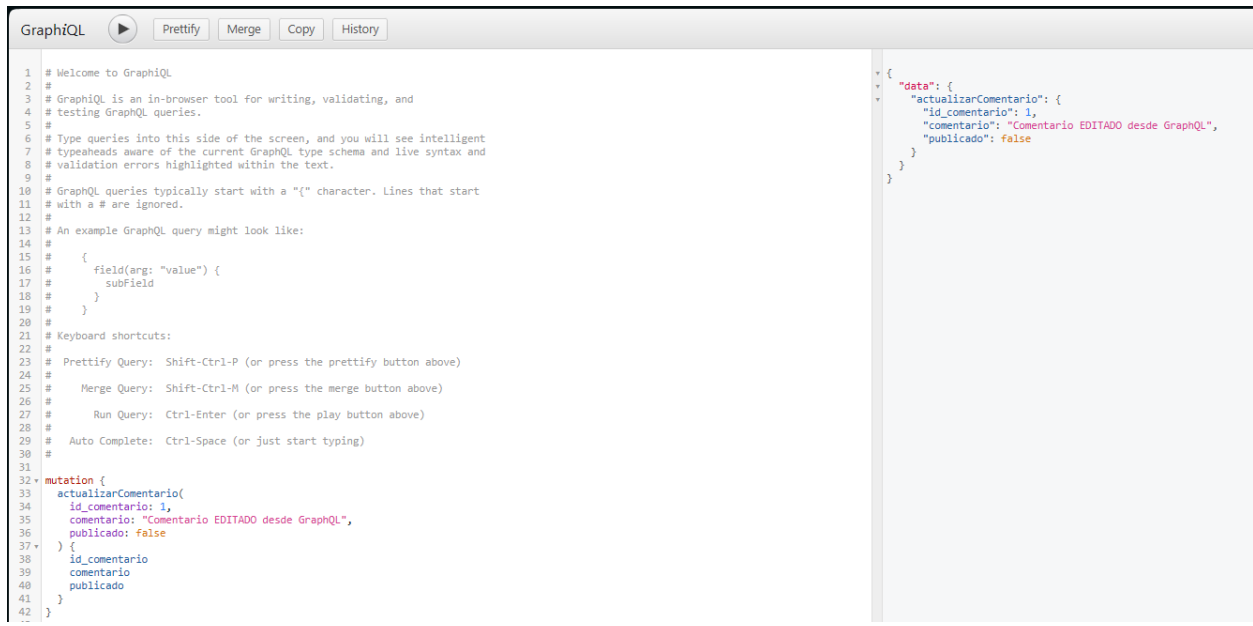
Copy

History

```
1 # Welcome to GraphiQL
2 #
3 # GraphiQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeheads aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that start
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 # {
16 #   field(arg: "value") {
17 #     subfield
18 #   }
19 # }
20 #
21 # Keyboard shortcuts:
22 #
23 # Prettify Query: Shift-Ctrl-P (or press the prettify button above)
24 #
25 # Merge Query: Shift-Ctrl-M (or press the merge button above)
26 #
27 # Run Query: Ctrl-Enter (or press the play button above)
28 #
29 # Auto Complete: Ctrl-Space (or just start typing)
30 #
31
32 query {
33   comentario(id: 3) {
34     id_comentario
35     comentario
36     documento_id_documento
37     fecha
38     publicado
39     isdeleted
40   }
41 }
42
```

```
{
  "data": {
    "comentario": {
      "id_comentario": 3,
      "comentario": "Comentario 3",
      "documento_id_documento": 2,
      "fecha": "1746230400000",
      "publicado": true,
      "isdeleted": false
    }
  }
}
```

## Mutación para actualizar un comentario

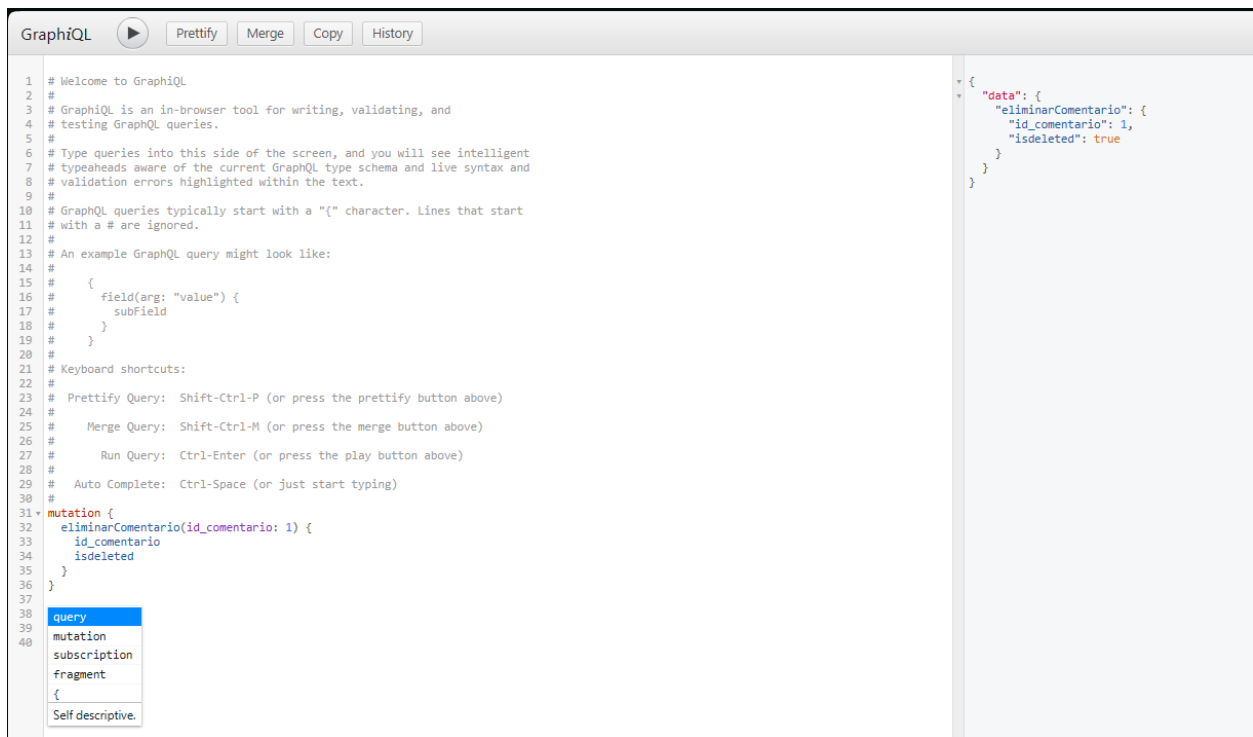


The screenshot shows the GraphiQL interface with a mutation query to update a comment. The left pane contains the query text, and the right pane shows the JSON response.

```
1 # Welcome to GraphiQL
2 #
3 # GraphiQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeahead aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that start
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 # {
16 #   field(arg: "value") {
17 #     subField
18 #   }
19 # }
20 #
21 # Keyboard shortcuts:
22 #
23 #   Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
24 #   Merge Query:    Shift-Ctrl-M (or press the merge button above)
25 #   Run Query:      Ctrl-Enter (or press the play button above)
26 #   Auto Complete:  Ctrl-Space (or just start typing)
27 #
31
32 mutation {
33   actualizarComentario(
34     id_comentario: 1,
35     comentario: "Comentario EDITADO desde GraphQL",
36     publicado: false
37   ) {
38     id_comentario
39     comentario
40     publicado
41   }
42 }
```

```
{
  "data": {
    "actualizarComentario": {
      "id_comentario": 1,
      "comentario": "Comentario EDITADO desde GraphQL",
      "publicado": false
    }
  }
}
```

## Mutación para eliminar un comentario (soft delete)



The screenshot shows the GraphiQL interface with a mutation query to soft delete a comment. The left pane contains the query text, and the right pane shows the JSON response. A dropdown menu is open over the query text, showing options like 'query', 'mutation', 'subscription', 'fragment', '{', and 'Self descriptive'.

```
1 # Welcome to GraphiQL
2 #
3 # GraphiQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeahead aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that start
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 # {
16 #   field(arg: "value") {
17 #     subField
18 #   }
19 # }
20 #
21 # Keyboard shortcuts:
22 #
23 #   Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
24 #   Merge Query:    Shift-Ctrl-M (or press the merge button above)
25 #   Run Query:      Ctrl-Enter (or press the play button above)
26 #   Auto Complete:  Ctrl-Space (or just start typing)
27 #
31
32 mutation {
33   eliminarComentario(id_comentario: 1) {
34     id_comentario
35     isdeleted
36   }
37 }
38 query
39 mutation
40 subscription
41 fragment
42 {
43 Self descriptive.
```

```
{
  "data": {
    "eliminarComentario": {
      "id_comentario": 1,
      "isdeleted": true
    }
  }
}
```