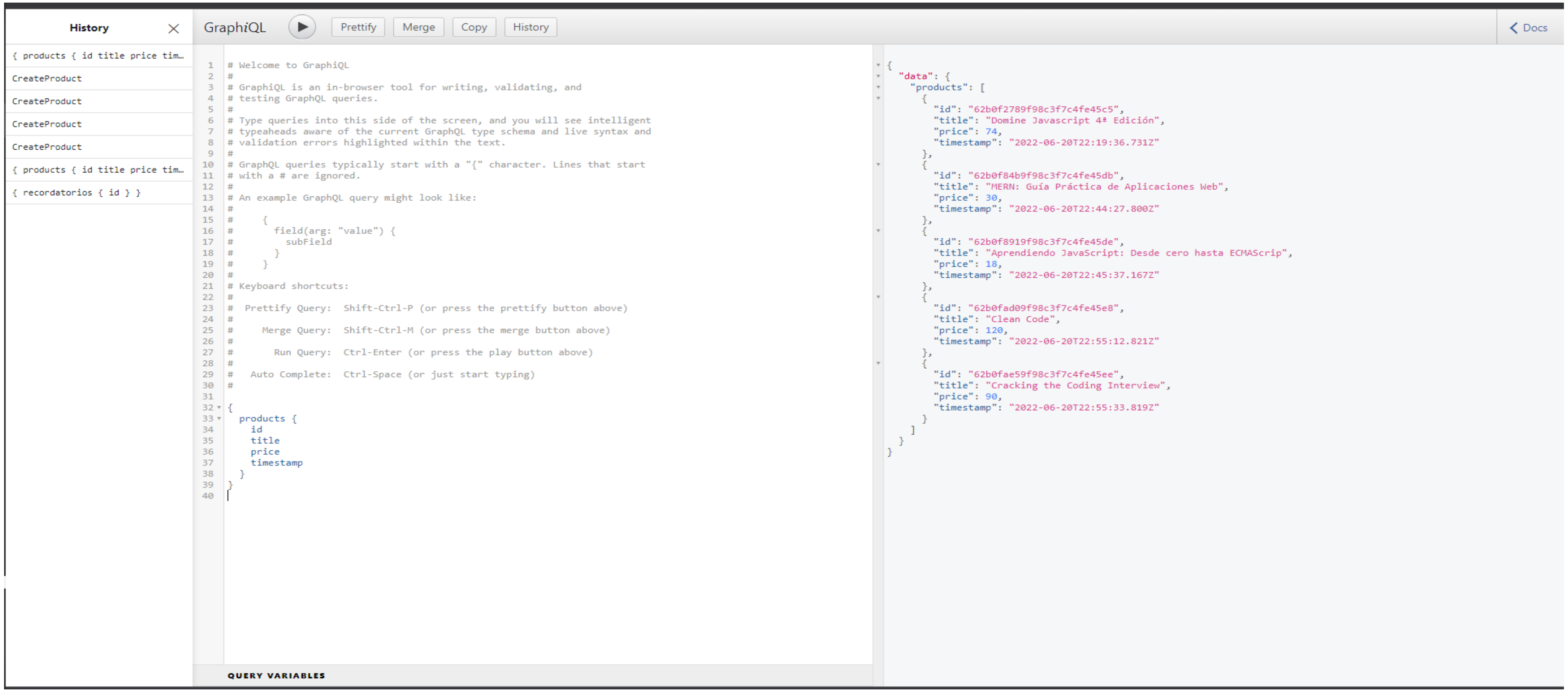
DESAFIO 21 – BACKEND CODERHOUSE

GERMÁN ALFONSO LOAIZA PINTO

CURSADA 28220

2022 - JUNIO

QUERY – LISTAR PRODUCTOS

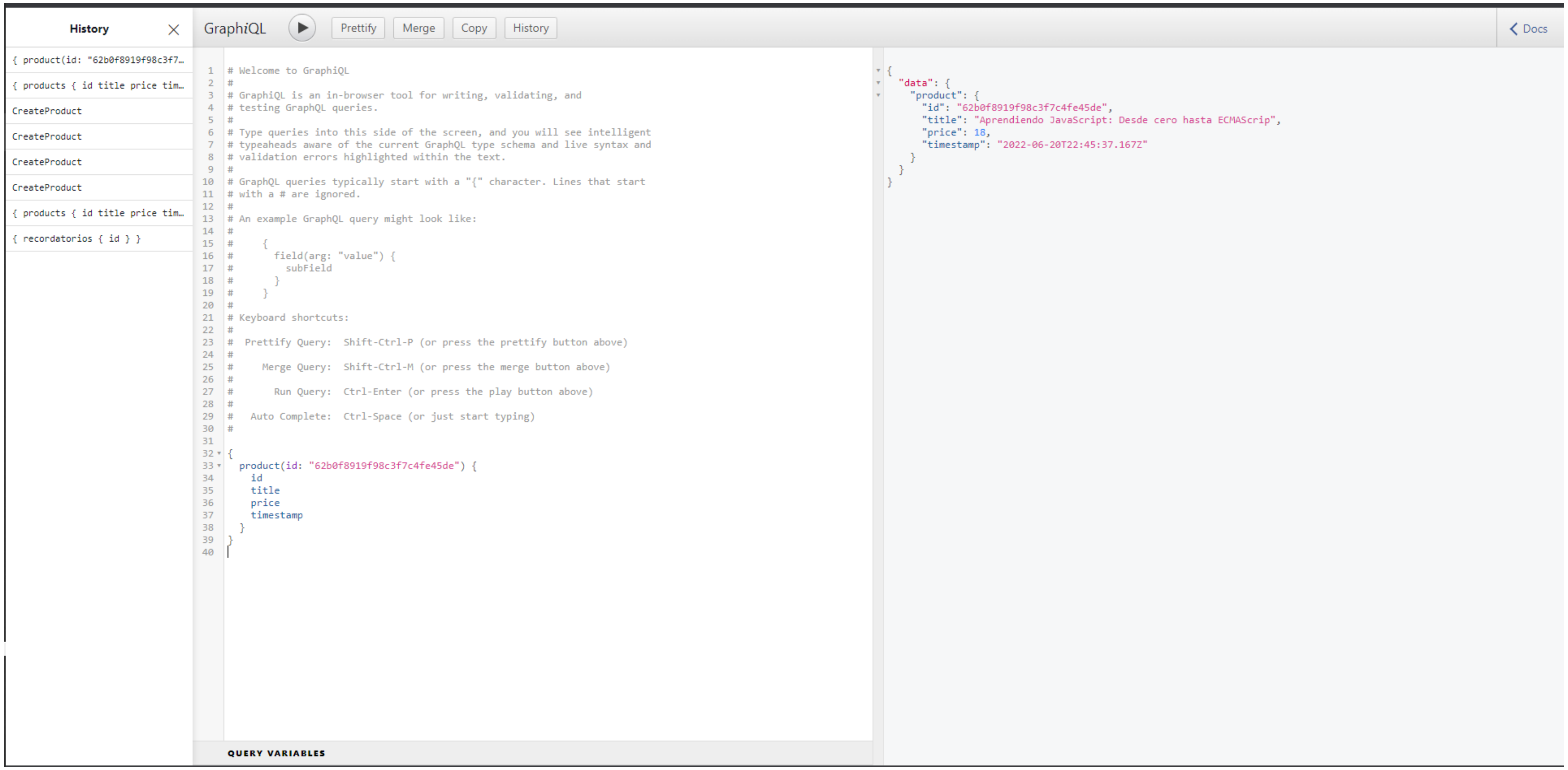


```
{ products { id title price tim…
CreateProduct
CreateProduct
CreateProduct
CreateProduct
{ products { id title price tim…
{ recordatorios { id } }
```

```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for writing, validating, and
# testing GraphQL queries.
#
# Type queries into this side of the screen, and you will see intelligent
# typeaheads aware of the current GraphQL type schema and live syntax and
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" character. Lines that start
# with a # are ignored.
#
# An example GraphQL query might look like:
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#  Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
#
#     Merge Query:  Shift-Ctrl-M (or press the merge button above)
#
#       Run Query:  Ctrl-Enter (or press the play button above)
#
#   Auto Complete:  Ctrl-Space (or just start typing)
#


{
  products {
    id
    title
    price
    timestamp
  }
}
```

```
{
  "data": {
    "products": [
      {
        "id": "62b0f2789f98c3f7c4fe45c5",
        "title": "Domine Javascript 4ª Edición",
        "price": 74,
        "timestamp": "2022-06-20T22:19:36.731Z"
      },
      {
        "id": "62b0f84b9f98c3f7c4fe45db",
        "title": "MERN: Guía Práctica de Aplicaciones Web",
        "price": 30,
        "timestamp": "2022-06-20T22:44:27.800Z"
      },
      {
        "id": "62b0f8919f98c3f7c4fe45de",
        "title": "Aprendiendo JavaScript: Desde cero hasta ECMAScrip",
        "price": 18,
        "timestamp": "2022-06-20T22:45:37.167Z"
      },
      {
        "id": "62b0fad09f98c3f7c4fe45e8",
        "title": "Clean Code",
        "price": 120,
        "timestamp": "2022-06-20T22:55:12.821Z"
      },
      {
        "id": "62b0fae59f98c3f7c4fe45ee",
        "title": "Cracking the Coding Interview",
        "price": 90,
        "timestamp": "2022-06-20T22:55:33.819Z"
      }
    ]
  }
}
```

QUERY – LISTAR PRODUCTO POR ID

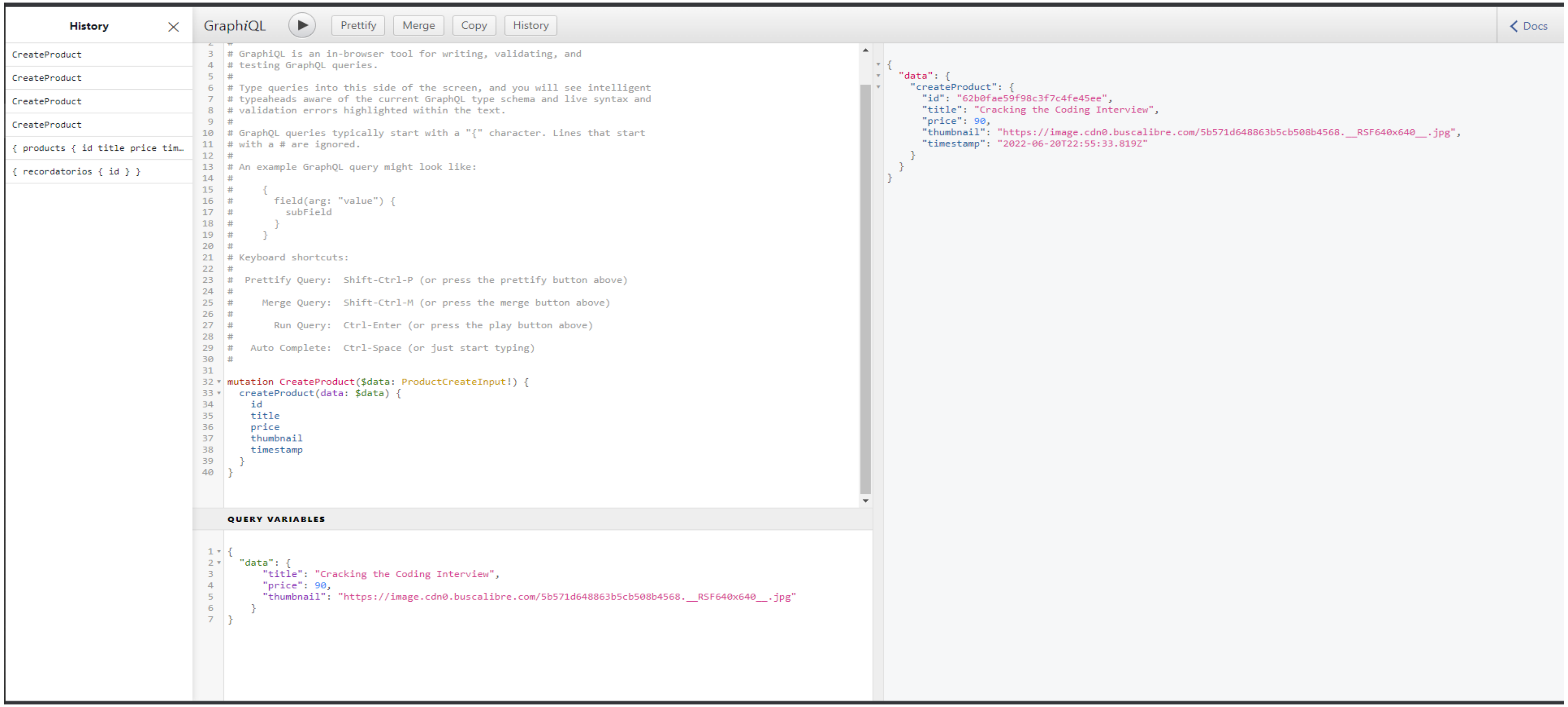


```
{ product(id: "62b0f8919f98c3f7…
{ products { id title price tim…
CreateProduct
CreateProduct
CreateProduct
CreateProduct
{ products { id title price tim…
{ recordatorios { id } }
```

```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for writing, validating, and
# testing GraphQL queries.
#
# Type queries into this side of the screen, and you will see intelligent
# typeaheads aware of the current GraphQL type schema and live syntax and
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" character. Lines that start
# with a # are ignored.
#
# An example GraphQL query might look like:
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#  Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
#
#     Merge Query:  Shift-Ctrl-M (or press the merge button above)
#
#       Run Query:  Ctrl-Enter (or press the play button above)
#
#   Auto Complete:  Ctrl-Space (or just start typing)
#

{
  product(id: "62b0f8919f98c3f7c4fe45de") {
    id
    title
    price
    timestamp
  }
}
```

```
{
  "data": {
    "product": {
      "id": "62b0f8919f98c3f7c4fe45de",
      "title": "Aprendiendo JavaScript: Desde cero hasta ECMAScrip",
      "price": 18,
      "timestamp": "2022-06-20T22:45:37.167Z"
    }
  }
}
```

QUERY – CREAR PRODUCTO



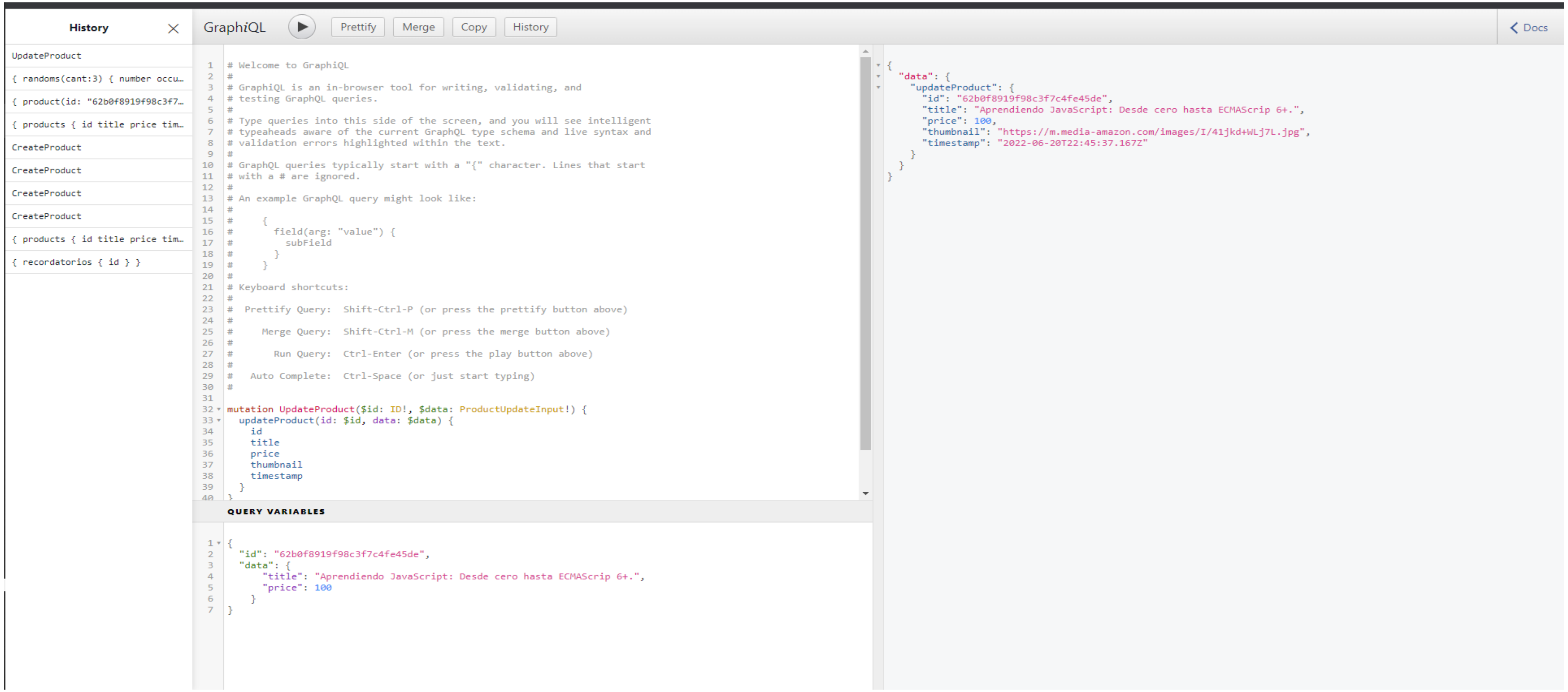
```
#
# GraphiQL is an in-browser tool for writing, validating, and
# testing GraphQL queries.
#
# Type queries into this side of the screen, and you will see intelligent
# typeaheads aware of the current GraphQL type schema and live syntax and
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" character. Lines that start
# with a # are ignored.
#
# An example GraphQL query might look like:
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#  Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
#
#     Merge Query:  Shift-Ctrl-M (or press the merge button above)
#
#       Run Query:  Ctrl-Enter (or press the play button above)
#
#   Auto Complete:  Ctrl-Space (or just start typing)
#

mutation CreateProduct($data: ProductCreateInput!) {
  createProduct(data: $data) {
    id
    title
    price
    thumbnail
    timestamp
  }
}
```

QUERY VARIABLES

```
{
  "data": {
      "title": "Cracking the Coding Interview",
      "price": 90,
      "thumbnail": "https://image.cdn0.buscalibre.com/5b571d648863b5cb508b4568.__RSF640x640__.jpg"
    }
}
```

```
{
  "data": {
    "createProduct": {
      "id": "62b0fae59f98c3f7c4fe45ee",
      "title": "Cracking the Coding Interview",
      "price": 90,
      "thumbnail": "https://image.cdn0.buscalibre.com/5b571d648863b5cb508b4568.__RSF640x640__.jpg",
      "timestamp": "2022-06-20T22:55:33.819Z"
    }
  }
}
```

QUERY – ACTUALIZAR/MODIFICAR PRODUCTO



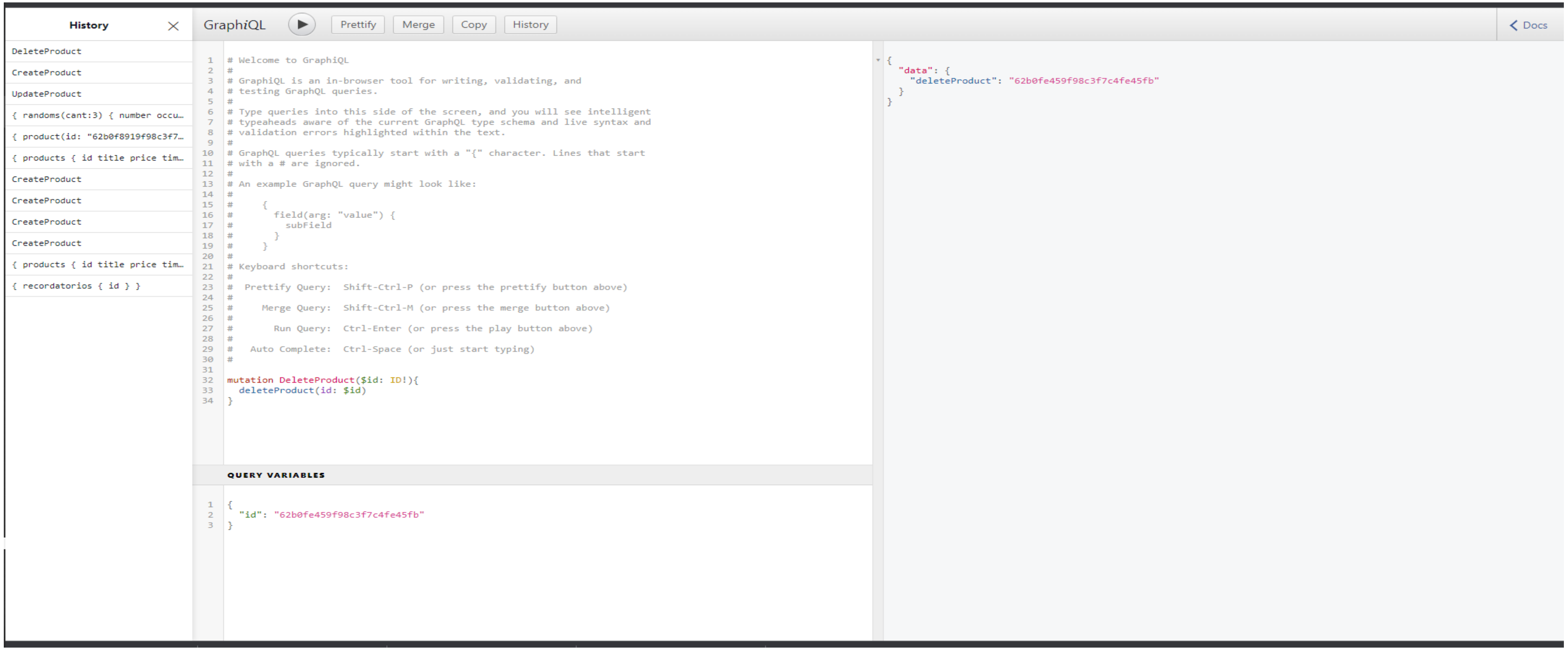
```
UpdateProduct
{ randoms(cant:3) { number occu…
{ product(id: "62b0f8919f98c3f7…
{ products { id title price tim…
CreateProduct
CreateProduct
CreateProduct
CreateProduct
{ products { id title price tim…
{ recordatorios { id } }
```

```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for writing, validating, and
# testing GraphQL queries.
#
# Type queries into this side of the screen, and you will see intelligent
# typeaheads aware of the current GraphQL type schema and live syntax and
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" character. Lines that start
# with a # are ignored.
#
# An example GraphQL query might look like:
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#  Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
#
#     Merge Query:  Shift-Ctrl-M (or press the merge button above)
#
#       Run Query:  Ctrl-Enter (or press the play button above)
#
#   Auto Complete:  Ctrl-Space (or just start typing)
#

mutation UpdateProduct($id: ID!, $data: ProductUpdateInput!) {
  updateProduct(id: $id, data: $data) {
    id
    title
    price
    thumbnail
    timestamp
  }
}
```

```
{
  "id": "62b0f8919f98c3f7c4fe45de",
  "data": {
      "title": "Aprendiendo JavaScript: Desde cero hasta ECMAScrip 6+.",
      "price": 100
    }
}
```

```
{
  "data": {
    "updateProduct": {
      "id": "62b0f8919f98c3f7c4fe45de",
      "title": "Aprendiendo JavaScript: Desde cero hasta ECMAScrip 6+.",
      "price": 100,
      "thumbnail": "https://m.media-amazon.com/images/I/41jkd+WLj7L.jpg",
      "timestamp": "2022-06-20T22:45:37.167Z"
    }
  }
}
```

QUERY – BORRAR PRODUCTO



```
DeleteProduct
CreateProduct
UpdateProduct
{ randoms(cant:3) { number occu…
{ product(id: "62b0f8919f98c3f7…
{ products { id title price tim…
CreateProduct
CreateProduct
CreateProduct
CreateProduct
{ products { id title price tim…
{ recordatorios { id } }
```

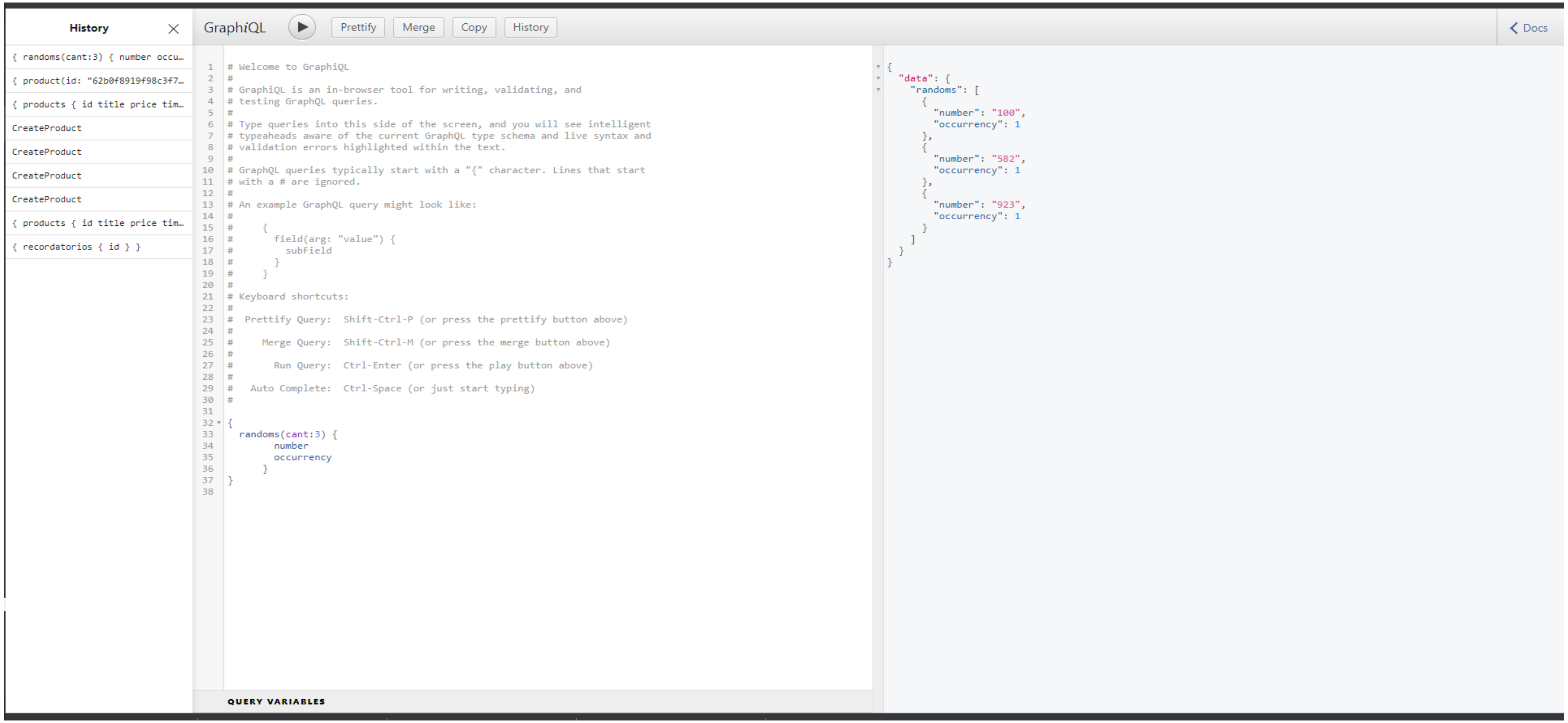
```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for writing, validating, and
# testing GraphQL queries.
#
# Type queries into this side of the screen, and you will see intelligent
# typeaheads aware of the current GraphQL type schema and live syntax and
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" character. Lines that start
# with a # are ignored.
#
# An example GraphQL query might look like:
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#  Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
#
#     Merge Query:  Shift-Ctrl-M (or press the merge button above)
#
#       Run Query:  Ctrl-Enter (or press the play button above)
#
#   Auto Complete:  Ctrl-Space (or just start typing)
#

mutation DeleteProduct($id: ID!){
  deleteProduct(id: $id)
}
```

QUERY VARIABLES

```
{
  "id": "62b0fe459f98c3f7c4fe45fb"
}
```

```
{
  "data": {
    "deleteProduct": "62b0fe459f98c3f7c4fe45fb"
  }
}
```

QUERY – NUMEROS RANDOMS



```
{ randoms(cant:3) { number occu…
{ product(id: "62b0f8919f98c3f7…
{ products { id title price tim…
CreateProduct
CreateProduct
CreateProduct
CreateProduct
{ products { id title price tim…
{ recordatorios { id } }
```

```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for writing, validating, and
# testing GraphQL queries.
#
# Type queries into this side of the screen, and you will see intelligent
# typeaheads aware of the current GraphQL type schema and live syntax and
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" character. Lines that start
# with a # are ignored.
#
# An example GraphQL query might look like:
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#  Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
#
#     Merge Query:  Shift-Ctrl-M (or press the merge button above)
#
#       Run Query:  Ctrl-Enter (or press the play button above)
#
#   Auto Complete:  Ctrl-Space (or just start typing)
#


{
  randoms(cant:3) {
        number
        occurrency
      }
}

```

```
{
  "data": {
    "randoms": [
      {
        "number": "100",
        "occurrency": 1
      },
      {
        "number": "582",
        "occurrency": 1
      },
      {
        "number": "923",
        "occurrency": 1
      }
    ]
  }
}
```