

DESAFIO 21 – BACKEND CODERHOUSE

GERMÁN ALFONSO LOAIZA PINTO

CURSADA 28220

2022 - JUNIO

QUERY – LISTAR PRODUCTOS

History

{ products { id title price tim...

CreateProduct

CreateProduct

CreateProduct

CreateProduct

{ products { id title price tim...

{ recordatorios { id } }

GraphiQL

Prettify

Merge

Copy

History

1 # Welcome to GraphQL

2 #

3 # GraphQL 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 {

33 products {

34 id

35 title

36 price

37 timestamp

38 }

39 }

40 |

QUERY VARIABLES

< Docs

▼

{

"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

History

X

GraphQL

▶

Prettify

Merge

Copy

History

< Docs

{ product(id: "62b0f8919f98c3f7...

{ products { id title price tim...

CreateProduct

CreateProduct

CreateProduct

CreateProduct

{ products { id title price tim...

{ recordatorios { id } }

1 # Welcome to GraphQL

2 #

3 # GraphQL 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 {

33 product(id: "62b0f8919f98c3f7c4fe45de") {

34 id

35 title

36 price

37 timestamp

38 }

39 }

40 }

{

"data": {

"product": {

"id": "62b0f8919f98c3f7c4fe45de",

"title": "Aprendiendo JavaScript: Desde cero hasta ECMAScrip",

"price": 18,

"timestamp": "2022-06-20T22:45:37.167Z"

}

}

}

QUERY VARIABLES

QUERY – CREAR PRODUCTO

History

×

CreateProduct

CreateProduct

CreateProduct

CreateProduct

{ products { id title price tim...

{ recordatorios { id } }

GraphQL

▶

Prettify

Merge

Copy

History

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

1

2

3

4

5

6

7

GraphQL 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"

}

}

< Docs

{

"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

History

UpdateProduct

{ randomness(cant:3) { number occu...

{ product(id: "62b0f8919f98c3f7...

{ products { id title price tim...

CreateProduct

CreateProduct

CreateProduct

CreateProduct

{ products { id title price tim...

{ recordatorios { id } }

GraphiQL

Prettify

Merge

Copy

History

1 # Welcome to GraphQL

2 #

3 # GraphQL 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 #

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 UpdateProduct(\$id: ID!, \$data: ProductUpdateInput!) {

33 updateProduct(id: \$id, data: \$data) {

34 id

35 title

36 price

37 thumbnail

38 timestamp

39 }

40 }

QUERY VARIABLES

1 {

2 "id": "62b0f8919f98c3f7c4fe45de",

3 "data": {

4 "title": "Aprendiendo JavaScript: Desde cero hasta ECMAScrip 6+.",

5 "price": 100

6 }

7 }

{

"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

History

×

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 } }

GraphQL

▶

Prettify

Merge

Copy

History

1 # Welcome to GraphQL

2 #

3 # GraphQL 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 DeleteProduct(\$id: ID!){

33 deleteProduct(id: \$id)

34 }

QUERY VARIABLES

1 {

2 "id": "62b0fe459f98c3f7c4fe45fb"

3 }

{

"data": {

"deleteProduct": "62b0fe459f98c3f7c4fe45fb"

}

}

QUERY – NUMEROS RANDOMS

History

×

{ randomness(cant:3) { number occu...

{ product(id: "62b0f8919f98c3f7...

{ products { id title price tim...

CreateProduct

CreateProduct

CreateProduct

CreateProduct

{ products { id title price tim...

{ recordatorios { id } }

GraphQL

▶

Prettify

Merge

Copy

History

1 # Welcome to GraphQL

2 #

3 # GraphQL 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 #

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 * {

33 randomness(cant:3) {

34 number

35 occurrency

36 }

37 }

38

⌵

{

"data": {

"randoms": [

{

"number": "100",

"occurrency": 1

},

{

"number": "582",

"occurrency": 1

},

{

"number": "923",

"occurrency": 1

}

]

}

}

⌵

QUERY VARIABLES