

Evidencias:

Registro de Padre/Madre de Familia: debe registrar su nombre, cedula, dirección, teléfono primario y telefono secundario. (5 PTS)

Examen3_Componentes / POST -parents-

POST ⌵ http://localhost:8080/parents

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ⌵

```
1 {
2   ...."nombre":"Jacinto mortadela",
3   ...."cedula":"1112111",
4   ...."direccion":"500 mts Sur de La Duquesa",
5   ...."telefonoPrimario":"72893123",
6   ...."telefonoSecundario":"8324234"
7 }
8
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ⌵

```
1 {
2   "nombre": "Jacinto mortadela",
3   "cedula": "1112111",
4   "direccion": "500 mts Sur de La Duquesa",
5   "telefonoPrimario": "72893123",
6   "telefonoSecundario": "8324234",
7   "children": null,
8   "id": 97
9 }
```

Debe hacer un endpoint para actualizar los datos de padre/madre (5pts)

The image shows a REST client interface with a PUT request to `http://localhost:8080/parents/97`. The request body is a JSON object with the following fields:

```
1 {
2   "nombre": "Jacinto Villalobos",
3   "cedula": "11231234",
4   "direccion": "300 mts sur de La Duquesa",
5   "telefonoPrimario": "72809671",
6   "telefonoSecundario": "8341213"
7 }
```

The response body is also shown in JSON format:

```
1 {
2   "nombre": "Jacinto Villalobos",
3   "cedula": "11231234",
4   "direccion": "300 mts sur de La Duquesa",
5   "telefonoPrimario": "72809671",
6   "telefonoSecundario": "8341213",
7   "children": [],
8   "id": 97
9 }
```

Otro endpoint para buscar los datos de un padre o madre por aproximación sobre el nombre y/o apellidos (5pts)

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/parents/name_middlename/Jacinto`. The response is displayed in the 'Body' tab, showing a JSON array of two parent objects. The first object has an ID of 65 and the second has an ID of 97. Both parents are named 'Jacinto Villalobos' and have the same phone numbers, but different addresses.

Request:

- Method: GET
- URL: `http://localhost:8080/parents/name_middlename/Jacinto`

Response (JSON):

```
[
  {
    "nombre": "Jacinto Villalobos",
    "cedula": "11231234",
    "direccion": "Ya tiene",
    "telefonoPrimario": "72809671",
    "telefonoSecundario": "8341213",
    "children": [],
    "id": 65
  },
  {
    "nombre": "Jacinto Villalobos",
    "cedula": "11231234",
    "direccion": "300 mts sur de La Duquesa",
    "telefonoPrimario": "72809671",
    "telefonoSecundario": "8341213",
    "children": [],
    "id": 97
  }
]
```

GET



http://localhost:8080/parents/name_middlename/co

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Query Params

KEY

Key

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1  [
2    {
3      "nombre": "Marco",
4      "cedula": "1112111",
5      "direccion": "No tiene",
6      "telefonoPrimario": "72893123",
7      "telefonoSecundario": "8324234",
8      "children": [],
9      "id": 1
10   }
11 ]
```

Registro de hijo/hija: debe registrar nombre, si es solo usuario de plan de lectura o de guardería o ambas y alergias (guárdelas como un solo texto) y asociarlos a sus padres. 10 pts

POST ▼ http://localhost:8080/children

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▼

```
1 {
2   ...."nombre":"Juan Barqueros Villalobo",
3   ...."planUsuario":"lectura",
4   ...."alergias":"todas",
5   ...."idParent":65
6 }
7
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1 {
2   "id_child": 65,
3   "nombre": "Juan Barqueros Villalobo",
4   "planUsuario": "lectura",
5   "alergias": "todas",
6   "idParent": 65
7 }
```

Debe hacer un endpoint que dado el id de un padre o madre liste sus datos y sus hijos o hijas 10 pts

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/parents/family/65`. The response is displayed in the 'Body' tab, formatted as JSON. The response contains a parent object and an array of children objects.

```
1 {
2   "nombre": "Jacinto Villalobos",
3   "cedula": "11231234",
4   "direccion": "Ya tiene",
5   "telefonoPrimario": "72809671",
6   "telefonoSecundario": "8341213",
7   "children": [
8     {
9       "id_child": 33,
10      "nombre": "Ortencio",
11      "planUsuario": "guardería",
12      "alergias": "todas",
13      "idParent": 65
14    },
15    {
16      "id_child": 65,
17      "nombre": "Juan Barqueros Villalobo",
18      "planUsuario": "lectura",
19      "alergias": "todas",
20      "idParent": 65
21    }
22  ],
23   "id": 65
24 }
```

Debe hacer un endpoint para actualizar los datos del niño o niña 5 pts

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/children/65`. The 'Body' tab is selected, and the request body is a JSON object. Below the request, the 'Body' tab of the response section is also selected, showing the response body in a pretty-printed JSON format.

Request:

```
PUT http://localhost:8080/children/65

{
  "nombre": "Marito mortadela",
  "planUsuario": "guardería",
  "alergias": "todas"
}
```

Response:

```
{
  "id_child": 65,
  "nombre": "Marito mortadela",
  "planUsuario": "guardería",
  "alergias": "todas",
  "idParent": 65
}
```

Debe registrar los libros que ha leído un niño 5pts

Examen3_Componentes / **POST** -records-

POST ⌵ <http://localhost:8080/records>

Params Authorization Headers (9) **Body** ● Pre-request Script

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary

```
1 {  
2   ... "idBook": 3,  
3   ... "idChild": 65  
4 }  
5
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ⌵

```
1 {  
2   "id": 129,  
3   "idBook": 3,  
4   "nameBook": "Harry Potter y la cámara secreta",  
5   "idChild": 65,  
6   "nameChild": "Marito mortadela"  
7 }
```


Debe hacer un endpoint que liste los libros que ha leído un niño 5pts

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/records/child/65`. The response is displayed in the 'Body' tab, showing a JSON array of three book records. The 'Params' tab is also visible, showing a 'KEY' parameter with a value of 'Key'.

GET `http://localhost:8080/records/child/65`

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	Value
Key	Value

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1  []
2  {
3    "id": 3,
4    "name": "Harry Potter y la cámara secreta",
5    "author": "J. K. Rowling",
6    "genre": "Ficción"
7  },
8  {
9    "id": 2,
10   "name": "Harry Potter y la piedra filosofal",
11   "author": "J. K. Rowling",
12   "genre": "Ficción"
13 },
14 {
15   "id": 1,
16   "name": "Don Quijote de la Mancha",
17   "author": "Miguel de Cervantes",
18   "genre": "Ficción"
19 }
20 []
```

Debe hacer un endpoint que liste los nombres de los niños y cuantos libros ha leído 10pts

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/records/childrenBookcount`. The response is displayed in the 'Body' tab, formatted as JSON. The JSON array contains three objects, each representing a child and the number of books they have read.

```
1  [
2    {
3      "childName": "Ortencio",
4      "count": 0
5    },
6    {
7      "childName": "Ortencio",
8      "count": 2
9    },
10   {
11     "childName": "Marito mortadela",
12     "count": 3
13   }
14 ]
```

CON GRAPHQL

POST

```
mutation {  
  createBook(name: "Harry potter", author: "J. K. K. Rowling", genre: "Ficci3n", status: "activo")  
  {  
    id  
  }  
}
```

```
{  
  "data": {  
    "createBook": {  
      "id": "33"  
    }  
  }  
}
```

GET

```
{  
  books(count: 10) {  
    id  
    name  
    author  
    genre  
    status  
  }  
}
```

Query.books(count: Int)

```
{  
  "data": {  
    "books": [  
      {  
        "id": "1",  
        "name": "banASADS",  
        "author": "123queso",  
        "genre": "toyota",  
        "status": "activo"  
      },  
      {  
        "id": "33",  
        "name": "Harry potter",  
        "author": "J. K. K. Rowling",  
        "genre": "Ficci3n",  
        "status": "activo"  
      },  
      {  
        "id": "34",  
        "name": "Harry potter 2",  
        "author": "J. K. K. Rowling",  
        "genre": "Ficci3n",  
        "status": "activo"  
      }  
    ]  
  }  
}
```

GET BY ID

```
{  
  book(id: 34) {  
    id  
    name  
    author  
    genre  
    status  
  }  
}
```

```
{  
  "data": {  
    "book": {  
      "id": "34",  
      "name": "Harry potter 2",  
      "author": "J. K. K. Rowling",  
      "genre": "Ficci3n",  
      "status": "activo"  
    }  
  }  
}
```

PUT

```
mutation {  
  updateBook(id: 34, name: "Don Quijote", author: "Miguel", genre: "Ficción", status: "activo") {  
    id  
  }  
}
```

```
{  
  "data": {  
    "updateBook": {  
      "id": "34"  
    }  
  }  
}
```

```
{  
  "data": {  
    "book": {  
      "id": "34",  
      "name": "Don Quijote",  
      "author": "Miguel",  
      "genre": "Ficción",  
      "status": "activo"  
    }  
  }  
}
```

DELETE

```
mutation {  
  deleteBook(id: 34) {  
    id  
  }  
}
```

```
{  
  "data": {  
    "deleteBook": {  
      "id": "34"  
    }  
  }  
}
```

```
{  
  "data": {  
    "book": {  
      "id": "34",  
      "name": "Don Quijote",  
      "author": "Miguel",  
      "genre": "Ficción",  
      "status": "inactivo"  
    }  
  }  
}
```