

Universidad de las fuerzas armadas espe

Nombre: Ariel llumiquinga

Fecha:06/02/2026

Actividad

La empresa TecnoMega requiere un prototipo de API para compra/venta de productos tecnológicos, usando Redis como base NoSQL clave–valor para almacenar catálogos y transacciones.

Objetivo general

Construir una API REST en Node.js + Express que:

1. Use middleware para parsear JSON (express.json()).
2. Use middleware response-time para medir tiempo de respuesta.
3. Implemente endpoints SET y GET contra Redis.
4. Cargue 10 registros por cada “tabla” desde un archivo JSON (clave–valor) y los almacene en Redis.

Modelo de datos (tablas en Redis)

Se simulan 4 “tablas” (colecciones):

1. clientes
2. productos
3. pedidos
4. detalle_pedido
5. Si el estudiante quiere agregar más colecciones lo puede realizar

Reglas mínimas

- **productos:** deben incluir código, nombre, categoría, precio, stock.
- **clientes:** deben incluir cedula o dni, nombres, email, teléfono, edad, genero
- **pedidos:** deben incluir Código , clientId, fecha, subtotal, iva, total, estado.
- **detalle_pedido:** debe incluir Código, productId, cantidad, detalle, precioUnit.

Entregables

1. Repositorio con: <https://github.com/arielll669/Leccion-Redis3U-BDA.git>
- index.js (API)

- data/tecnomega.json (10 registros por tabla mínimo)

2. Evidencia:

- Capturas de Postman/Thunder Client
- Salida del tiempo de respuesta (Response-Time)

DESARROLLO

PRUEBA 1: Verifica que la API está corriendo

1. Método: **GET**

2. URL: <http://localhost:3000/>

```

{
  "message": "API TecnoMega - Redis + Node.js",
  "endpoints": {
    "seed": "POST /seed - Carga masiva desde JSON",
    "crear": "POST /collection - Guardar un registro",
    "obtener": "GET /collection/:id - Obtener un registro",
    "listar": "GET /collection - Listar todos los registros"
  },
  "collections": [
    "clientes",
    "productos",
    "pedidos",
    "detalle_pedido"
  ]
}

```

PRUEBA 2: Cargar datos masivos (SEED)

carga todos los datos del JSON a Redis.

1. Método: **POST**

2. URL: <http://localhost:3000/seed>

The screenshot shows the Postman interface. At the top, it says "http://localhost:3000/seed". Below that, a "POST" method is selected. The URL is "http://localhost:3000/seed". The "Body" tab is active, showing the raw JSON content:

```
1 | { "success": true, "message": "Datos cargados exitosamente", "totalInserted": 40, "duracion": "66ms", "collections": { "clientes": 10, "productos": 10, "pedidos": 10, "detalle_pedido": 10 } }
```

At the bottom right, the response is shown as "200 OK" with a duration of 104 ms and a size of 407 B.

Terminal de vsc:

```
PS C:\Users\andre\Documents\bdAvanzada2\Leccion3Parcial> node index.js
    Conectado a Redis
    Servidor corriendo en http://localhost:3000
    GET / - 8.13ms
    POST /seed - 67.33ms
```

PRUEBA 3: Listar todos los clientes (GET)

1. Método: **GET**
2. URL: <http://localhost:3000/clientes>

```

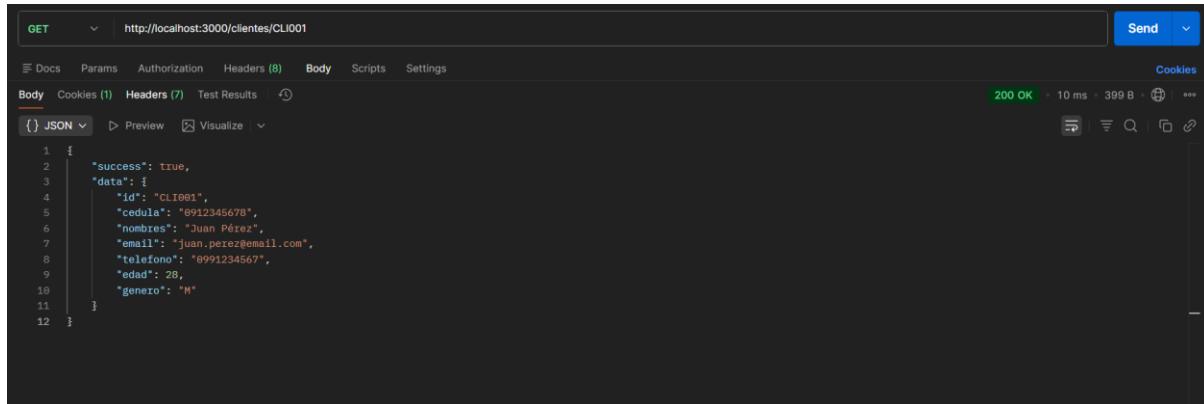
GET http://localhost:3000/clients
200 OK 19 ms 1.71 KB
Body Cookies Headers (8) Body Scripts Settings
Body Cookies Headers (7) Test Results
JSON Preview Visualize
1 {
2   "success": true,
3   "collection": "clients",
4   "count": 10,
5   "data": [
6     {
7       "id": "CLI001",
8       "cedula": "0912345678",
9       "nombres": "Juan Pérez",
10      "email": "juan.perez@email.com",
11      "telefono": "0991234567",
12      "edad": 28,
13      "genero": "M"
14    },
15    {
16      "id": "CLI002",
17      "cedula": "0923456789",
18      "nombres": "María González",
19      "email": "maria.gonzalez@email.com",
20      "telefono": "0992345678",
21      "edad": 32,
22      "genero": "F"
23    },
24    {
25      "id": "CLI003",
26      "cedula": "0934567890",
27      "nombres": "Carlos Rodríguez",
28      "email": "carlos.rodriguez@email.com",
29      "telefono": "0993456789",
30      "edad": 25,
31      "genero": "M"
32    },
33    {
34      "id": "CLI004",
35      "cedula": "0945678901",
36      "nombres": "Ana Martínez",
37      "email": "ana.martinez@email.com",
38      "telefono": "0994567890",
39      "edad": 30,
40      "genero": "F"
41    },
42    {
43      "id": "CLI005",
44      "cedula": "0956789012",
45      "nombres": "Luis Fernández",
46      "email": "luis.fernandez@email.com",
47      "telefono": "0995678901",
48      "edad": 35,
49    },
50    {
51      "id": "CLI006",
52      "cedula": "0967890123",
53      "nombres": "Sofía López",
54      "email": "sofia.lopez@email.com",
55      "telefono": "0996789012",
56      "edad": 27,
57      "genero": "F"
58    },
59    {
60      "id": "CLI007",
61      "cedula": "0978901234",
62      "nombres": "Diego Torres",
63      "email": "diego.torres@email.com",
64      "telefono": "0997890123",
65      "edad": 29,
66      "genero": "M"
67    },
68    {
69      "id": "CLI008",
70      "cedula": "0989012345",
71      "nombres": "Valentina Ruiz",
72      "email": "valentina.ruiz@email.com",
73      "telefono": "0998901234",
74      "edad": 24,
75      "genero": "F"
76    }
77  ],
78  {
79    "email": "Valentina.Ruiz@email.com",
80    "telefono": "0998901234",
81    "edad": 24,
82    "genero": "F"
83  },
84  {
85    "id": "CLI009",
86    "cedula": "0990123456",
87    "nombres": "Andrés Silva",
88    "email": "Andres.silva@email.com",
89    "telefono": "0999012345",
90    "edad": 31,
91    "genero": "M"
92  },
93  {
94    "id": "CLI010",
95    "cedula": "0901234567",
96    "nombres": "Camila Herrera",
97    "email": "Camila.herrera@email.com",
98    "telefono": "0990123456",
99    "edad": 26,
100   "genero": "F"
101 }
102 ]

```

PRUEBA 4: Obtener un cliente específico (GET)

1. Método: GET

2. URL: <http://localhost:3000/clientes/CLI001>

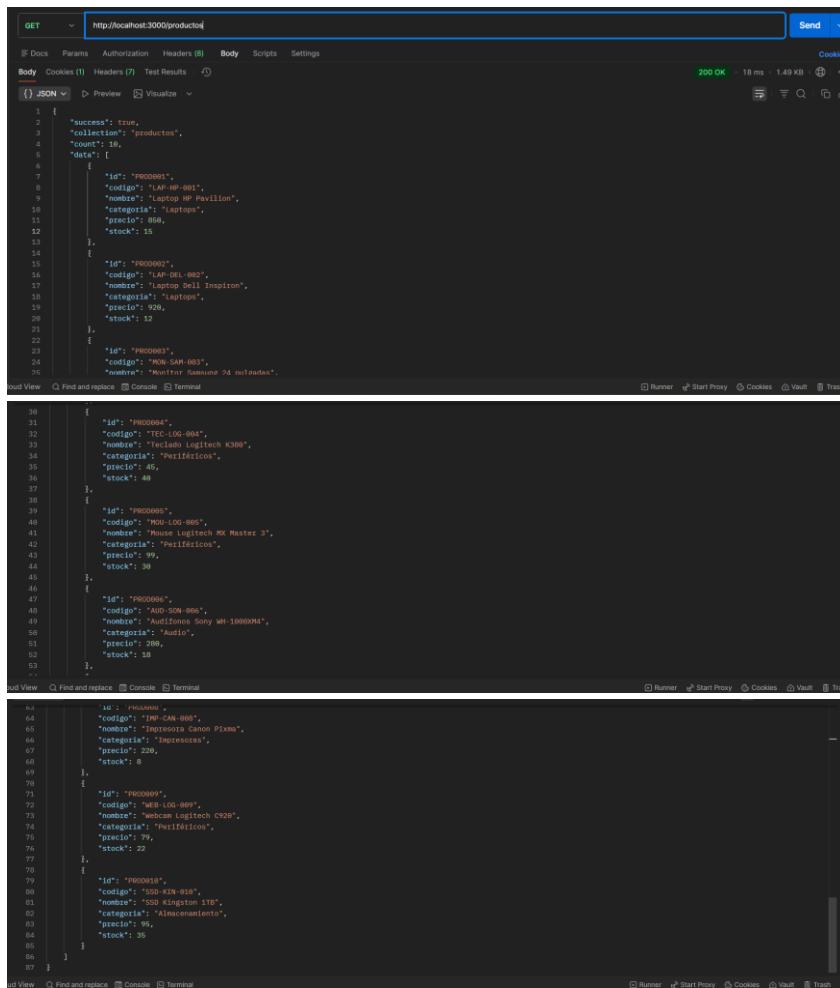


```
1 {  
2   "success": true,  
3   "data": {  
4     "id": "CLI001",  
5     "cedula": "0912345678",  
6     "nombres": "Juan Pérez",  
7     "email": "juan.perez@email.com",  
8     "telefono": "0991234567",  
9     "edad": 28,  
10    "genero": "M"  
11  }  
12 }
```

PRUEBA 5: Listar todos los productos (GET)

1. Método: GET

2. URL: <http://localhost:3000/productos>



```
1 {  
2   "success": true,  
3   "collection": "productos",  
4   "count": 10,  
5   "data": [  
6     {  
7       "id": "P0000001",  
8       "codigo": "LAP-HP-001",  
9       "nombre": "Laptop HP Pavilion",  
10      "categoria": "Laptops",  
11      "precio": 999,  
12      "stock": 15  
13    },  
14    {  
15      "id": "P0000002",  
16      "codigo": "LAP-DELL-002",  
17      "nombre": "Laptop Dell Inspiron",  
18      "categoria": "Laptops",  
19      "precio": 929,  
20      "stock": 12  
21    },  
22    {  
23      "id": "P0000003",  
24      "codigo": "MON-SAM-003",  
25      "nombre": "Monitor Samsung 24 pulgadas".  
26    }  
27  ],  
28  [  
29    {  
30      "id": "P0000004",  
31      "codigo": "TEC-LOG-004",  
32      "nombre": "Teclado Logitech K300",  
33      "categoria": "Periféricos",  
34      "precio": 45,  
35      "stock": 40  
36    },  
37    {  
38      "id": "P0000005",  
39      "codigo": "MOU-LOG-005",  
40      "nombre": "Mouse Logitech MX Master 3",  
41      "categoria": "Periféricos",  
42      "precio": 129,  
43      "stock": 30  
44    },  
45    {  
46      "id": "P0000006",  
47      "codigo": "AUD-SON-006",  
48      "nombre": "Audífonos Sony WH-1000XM4",  
49      "categoria": "Audio",  
50      "precio": 299,  
51      "stock": 18  
52    },  
53    {  
54      "id": "P0000007",  
55      "codigo": "IMP-CAN-007",  
56      "nombre": "Impresora Canon Pixma",  
57      "categoria": "Impresoras",  
58      "precio": 229,  
59      "stock": 8  
60    },  
61    {  
62      "id": "P0000008",  
63      "codigo": "WEB-LOG-008",  
64      "nombre": "Webcam Logitech C920",  
65      "categoria": "Periféricos",  
66      "precio": 99,  
67      "stock": 22  
68    },  
69  ],  
70  [  
71    {  
72      "id": "P0000009",  
73      "codigo": "SSD-KIN-009",  
74      "nombre": "SSD Kingston 1TB",  
75      "categoria": "Almacenamiento",  
76      "precio": 99,  
77      "stock": 35  
78    },  
79  ]  
80 }
```

PRUEBA 6: Crear un nuevo producto (POST)

1. Método: **POST**

2. URL: <http://localhost:3000/productos>

The screenshot shows a Postman request configuration. The method is set to **POST** and the URL is <http://localhost:3000/productos>. The **Body** tab is selected, showing a raw JSON payload:

```
1 {
2   "id": "PROD011",
3   "codigo": "RAM-CORS-011",
4   "nombre": "RAM Corsair Vengeance 16GB",
5   "categoria": "Componentes",
6   "precio": 120.00,
7   "stock": 20
8 }
```

The response status is **200 OK** with a response time of 10 ms and a size of 431 B. The response body is identical to the request body.

PRUEBA 7: Verificar que el nuevo producto se guardó

1. Método: **GET**

2. URL: <http://localhost:3000/productos/PROD011>

The screenshot shows a Postman request configuration. The method is set to **GET** and the URL is <http://localhost:3000/productos/PROD011>. The **Body** tab is selected, showing a raw JSON response:

```
1 {
2   "success": true,
3   "data": {
4     "id": "PROD011",
5     "codigo": "RAM-CORS-011",
6     "nombre": "RAM Corsair Vengeance 16GB",
7     "categoria": "Componentes",
8     "precio": 120,
9     "stock": 20
10   }
11 }
```

The response status is **200 OK** with a response time of 9 ms and a size of 388 B. The response body is identical to the request body.

PRUEBA 8: Listar todos los pedidos

1. Método: **GET**

2. URL: <http://localhost:3000/pedidos>

GET http://localhost:3000/pedidos

Body Cookies Headers (7) Test Results

{} JSON Preview Visualize

```
1 {  
2   "success": true,  
3   "collection": "pedidos",  
4   "count": 10,  
5   "data": [  
6     {  
7       "id": "PED001",  
8       "codigo": "ORD-2025-001",  
9       "clienteId": "CLI001",  
10      "fecha": "2025-01-15",  
11      "subtotal": 850,  
12      "iva": 102,  
13      "total": 952,  
14      "estado": "completado"  
15    },  
16    {  
17      "id": "PED002",  
18      "codigo": "ORD-2025-002",  
19      "clienteId": "CLI002",  
20      "fecha": "2025-01-16",  
21      "subtotal": 225,  
22    },  
23    {  
24      "id": "PED003",  
25      "codigo": "ORD-2025-003",  
26      "clienteId": "CLI003",  
27      "fecha": "2025-01-17",  
28      "subtotal": 920,  
29      "iva": 110.4,  
30      "total": 1030.4,  
31      "estado": "completado"  
32    },  
33    {  
34      "id": "PED004",  
35      "codigo": "ORD-2025-004",  
36      "clienteId": "CLI004",  
37      "fecha": "2025-01-18",  
38      "subtotal": 379,  
39      "iva": 45.48,  
40      "total": 424.48,  
41      "estado": "en_proceso"  
42    },  
43    {  
44      "id": "PED005",  
45      "codigo": "ORD-2025-005",  
46      "clienteId": "CLI005",  
47      "fecha": "2025-01-19",  
48      "subtotal": 650,  
49      "iva": 78,  
50      "total": 728,  
51      "estado": "completado"  
52    },  
53    {  
54      "id": "PED006",  
55      "codigo": "ORD-2025-006",  
56      "clienteId": "CLI006",  
57      "fecha": "2025-01-20",  
58      "subtotal": 144,  
59      "iva": 17.28,  
60      "total": 161.28,  
61      "estado": "pendiente"  
62    },  
63  ],  
64}  
65
```

```

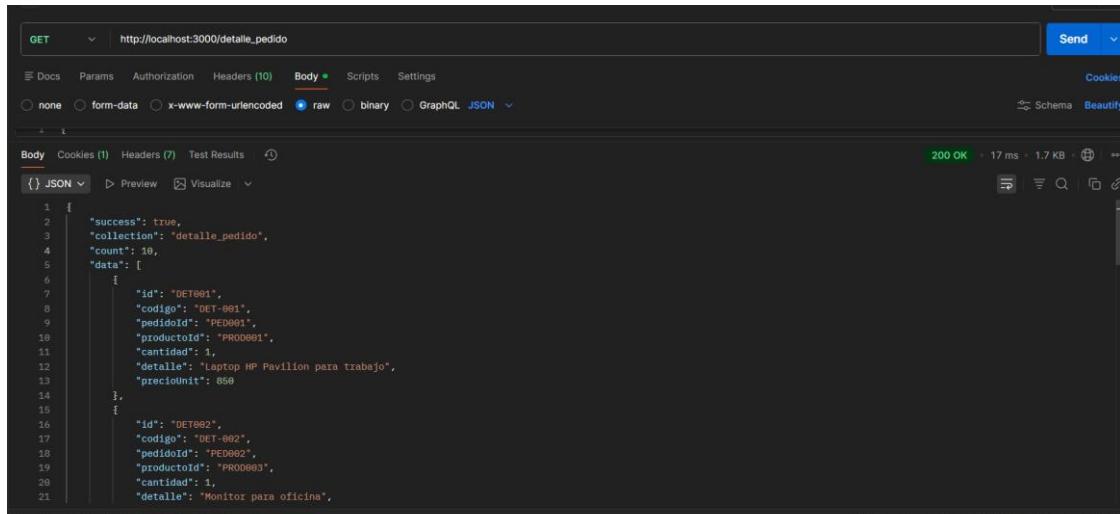
65      },
66      {
67        "id": "PED007",
68        "codigo": "ORD-2025-007",
69        "clienteId": "CLI007",
70        "fecha": "2025-01-21",
71        "subtotal": 208,
72        "iva": 33.6,
73        "total": 313.6,
74        "estado": "completado"
75      },
76      {
77        "id": "PED008",
78        "codigo": "ORD-2025-008",
79        "clienteId": "CLI008",
80        "fecha": "2025-01-22",
81        "subtotal": 220,
82        "iva": 26.4,
83        "total": 246.4,
84        "estado": "en_proceso"
85      },
86    ],
87    [
88      {
89        "id": "PED009",
90        "codigo": "ORD-2025-009",
91        "clienteId": "CLI009",
92        "fecha": "2025-01-23",
93        "subtotal": 174,
94        "iva": 29.88,
95        "total": 194.88,
96        "estado": "completado"
97      },
98      {
99        "id": "PED010",
100       "codigo": "ORD-2025-010",
101       "clienteId": "CLI010",
102       "fecha": "2025-01-24",
103       "subtotal": 95,
104       "iva": 11.4,
105       "total": 106.4,
106       "estado": "pendiente"
107     ]
108   ]

```

PRUEBA 9: Listar todos los detalles de pedido

1. Método: GET

2. URL: http://localhost:3000/detalle_pedido



The screenshot shows a POSTMAN interface with a successful API call to `http://localhost:3000/detalle_pedido`. The response status is `200 OK` with a response time of `17 ms` and a size of `1.7 KB`. The response body is a JSON object:

```

1  {
2    "success": true,
3    "collection": "detalle_pedido",
4    "count": 10,
5    "data": [
6      {
7        "id": "DET001",
8        "codigo": "DET-001",
9        "pedidoId": "PED001",
10       "productId": "PRO0001",
11       "cantidad": 1,
12       "detalle": "Laptop HP Pavilion para trabajo",
13       "precioUnit": 850
14     },
15     {
16       "id": "DET002",
17       "codigo": "DET-002",
18       "pedidoId": "PED002",
19       "productId": "PRO0003",
20       "cantidad": 1,
21       "detalle": "Monitor para oficina",
22     }
23   ],
24   {
25     "id": "DET003",
26     "codigo": "DET-003",
27     "pedidoId": "PED002",
28     "productId": "PRO0004",
29     "cantidad": 1,
30     "detalle": "Teclado inalámbrico",
31     "precioUnit": 45
32   },
33   {
34     "id": "DET004",
35     "codigo": "DET-004",
36     "pedidoId": "PED003",
37     "productId": "PRO0002",
38     "cantidad": 1,
39     "detalle": "Laptop Dell para estudiante",
40     "precioUnit": 920
41   },
42   {
43     "id": "DET005",
44   }
45 ]

```

```

40      "precioUnit": 920
41    },
42  },
43  {
44    "id": "DET005",
45    "codigo": "DET-005",
46    "pedidoid": "PED004",
47    "productoid": "PROD006",
48    "cantidad": 1,
49    "detalle": "Audifonos con cancelación de ruido",
50    "precioUnit": 280
51  },
52  {
53    "id": "DET006",
54    "codigo": "DET-006",
55    "pedidoid": "PED004",
56    "productoid": "PROD005",
57    "cantidad": 1,
58    "detalle": "Mouse ergonómico",
59    "precioUnit": 99
60  },
61  {
62    "id": "DET007",
63    "codigo": "DET-007",
64    "pedidoid": "PED005",
65    "productoid": "PROD007",
66    "cantidad": 1,
67    "detalle": "Tablet para diseño",
68    "precioUnit": 650
69  },
70  {
71    "id": "DET008",
72    "codigo": "DET-008",
73    "pedidoid": "PED006",
74    "productoid": "PROD004",
75    "cantidad": 2,
76    "detalle": "Teclados para equipo",
77    "precioUnit": 45
78  },
79  {
80    "id": "DET009",
81    "codigo": "DET-009",
82    "pedidoid": "PED006",
83    "productoid": "PROD009",
84    "cantidad": 2,
85    "detalle": "Webcams para reuniones",
86    "precioUnit": 79
87  },
88  {
89    "id": "DET010",
90    "codigo": "DET-010",
91    "pedidoid": "PED007",
92    "productoid": "PROD006",
93    "cantidad": 1,
94    "detalle": "Audifonos premium",
95    "precioUnit": 280
96  },
97

```

View Find and replace Console Terminal Runner Start Proxy Cookies Vault Trash

Captura de la Terminal:

```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
PS C:\Users\andre\Documents\bdAvanzada2\Leccion3Parcial> node index.js
GET / - 8.13ms
POST /seed - 67.33ms
GET /clientes - 10.12ms
GET /clientes/CLI001 - 2.10ms
GET /productos - 8.73ms
POST /productos - 0.72ms
POST /productos - 2.29ms
GET /productos/PROD011 - 1.32ms
GET /pedidos - 8.84ms
GET /detalle_pedido - 8.60ms
GET /detalle_pedido - 8.37ms

```

Lín. 370, col. 2 Espacios: 4 UTF-8 CRLF { } JSON ↻

Evidencias Ubuntu

Ver todas las claves almacenadas:

```
andre@BEYKER:~$ redis-cli
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> KEYS *
1) "clientes:CLI005"
2) "pedidos:PED010"
3) "productos:PROD011"
4) "estudiante:"
5) "estudiante:1724966849"
6) "pedidos:PED003"
7) "detalle_pedido:DET004"
8) "clientes:CLI003"
9) "estudiante:1850598010"
10) "productos:PROD003"
11) "detalle_pedido:DET005"
12) "estudiante:1725600819"
13) "clientes:CLI001"
14) "pedidos:PED001"
15) "detalle_pedido:DET002"
16) "clientes:index"
17) "productos:PROD004"
18) "detalle_pedido:index"
19) "empleado:1"
20) "estudiante:1754459160"
21) "detalle_pedido:DET001"
22) "clientes:CLI002"
23) "estudiante:1722651567"
24) "clientes:CLI007"
25) "clientes:CLI009"
26) "estudiante:1723167514"
27) "key"
28) "detalle_pedido:DET003"
29) "test"
30) "productos:PROD005"
31) "estudiante:1850677764"
32) "clientes:CLI008"
33) "user:1002"
34) "productos:PROD009"
35) "pedidos:PED007"
36) "pedidos:PED006"
37) "estudiante:1727641407"
```

Ver un registro específico (JSON)

JSON almacenado del cliente

```
127.0.0.1:6379> GET clientes:CLI001
"{"id":"CLI001","cedula":"0912345678","nombres":"Juan Perez","email":"juan.perez@email.com","telefono":"0991234567","edad":28,"genero":"M"}"
```

Ver el contenido de un índice (Set)

todos los IDs de clientes

```
127.0.0.1:6379> SMEMBERS clientes:index
1) "CLI001"
2) "CLI002"
3) "CLI003"
4) "CLI004"
5) "CLI005"
6) "CLI006"
7) "CLI007"
8) "CLI008"
9) "CLI009"
10) "CLI010"
```

Contar cuántas claves hay

total de claves

```
127.0.0.1:6379> DBSIZE
(integer) 72
```