



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Entrega 7

Taller - 2022-1

Introducción a Cloud Computing con AWS

13 de mayo de 2022 - v1.0

Índice

1. Creación de tabla	4
2. Creación de item	4
3. Carga de items	5
4. Operaciones sobre tabla	7
5. Operaciones sobre indice	14

Índice de figuras

1. Captura de pantalla de la consola de DynamoDB con la vista Overview de la tabla creada	4
2. Captura de pantalla mostrando el item a crear en el formulario	4
3. Captura de pantalla mostrando la tabla con el nuevo item creado	5
4. Captura de pantalla mostrando el código de la función de carga de datos .	5
5. Captura de pantalla mostrando “General Configuration” de la función de carga de datos	6
6. Captura de pantalla mostrando los permisos de ejecución del rol de la función lambda	6
7. Captura de pantalla mostrando la lista de items contenidos en la tabla luego de la ejecución del test	6

Instrucciones

El presente documento corresponde a una plantilla que incluye las informaciones que deben ser proveídas para evaluar la entrega.

La entrega se basa mayormente en capturas de pantalla de la consola, la mayoría de ellas vistas en clase más algunas acciones adicionales que deben ser descubiertas por cada estudiante.

Todos los textos en rojo a lo largo de la plantilla, junto con esta página de instrucciones, deben ser eliminadas antes de la compilación final que debe ser entregada por Moodle.

1. Creación de tabla

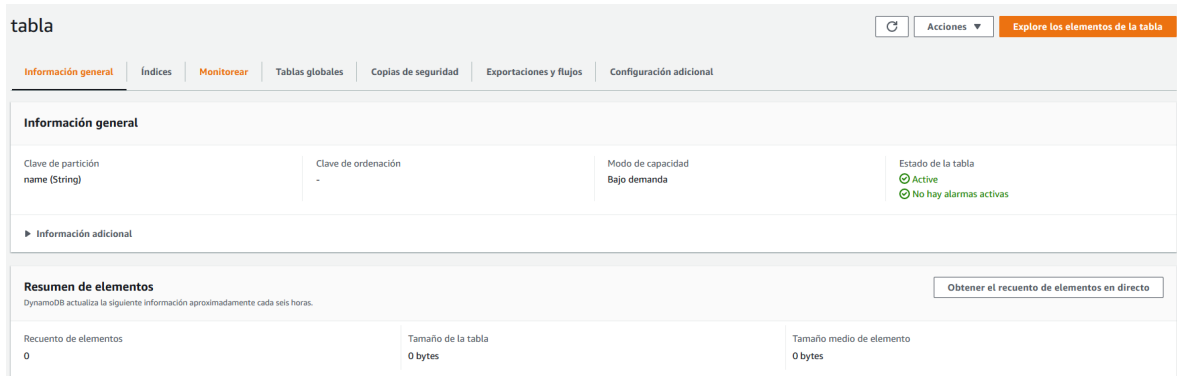


Figura 1: Captura de pantalla de la consola de DynamoDB con la vista Overview de la tabla creada

2. Creación de item

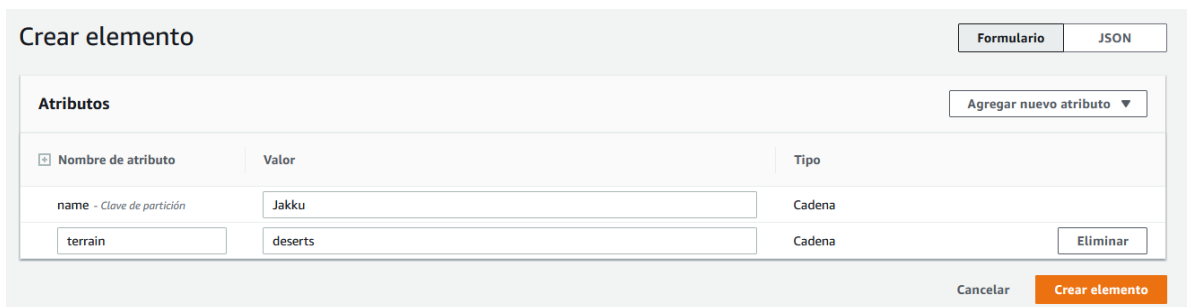


Figura 2: Captura de pantalla mostrando el item a crear en el formulario

La representación del item a ingresar en DynamoDB JSON corresponde a:

```

1 {
2   {
3     "name": {
4       "S": "Jakku"
5     },
6     "terrain": {
7       "S": "deserts"
8     }
9   }
10 }
```

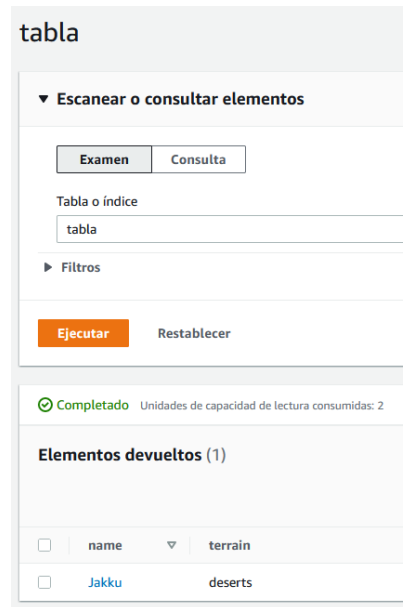


Figura 3: Captura de pantalla mostrando la tabla con el nuevo item creado

3. Carga de items

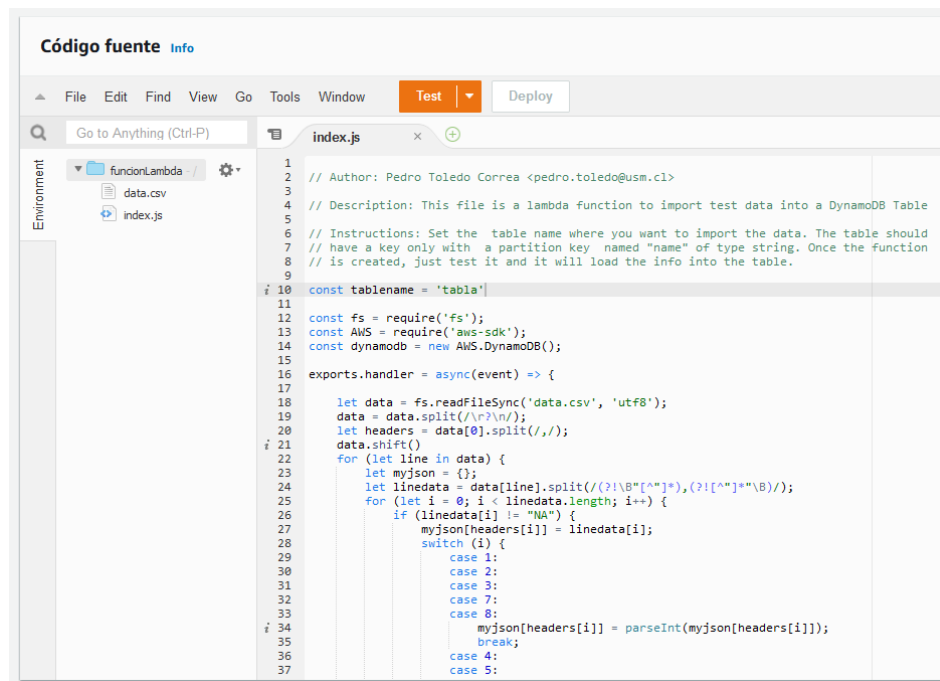


Figura 4: Captura de pantalla mostrando el código de la función de carga de datos

Configuración general Info		
Descripción	Memoria	Almacenamiento efímero
-	128 MB	512 MB
Tiempo de espera		
10 min 0 s		

Figura 5: Captura de pantalla mostrando “General Configuration” de la función de carga de datos

Políticas de permisos (2)	
Puede asociar hasta 10 políticas administradas.	
<input type="text" value="Filtre las políticas por propiedad o nombre de política y pulse Intro"/>	
<input type="checkbox"/>	Nombre de la política
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-0bfe3258-e4e6-4c6d-953f-b52d6884fd55
<input type="checkbox"/>	AmazonDynamoDBFullAccess

Figura 6: Captura de pantalla mostrando los permisos de ejecución del rol de la función lambda

tabla									
<input checked="" type="checkbox"/> Vista previa automática Acciones Crear elemento Actualizar la configuración de las tablas									
Escanear o consultar elementos Expandir para consultar o examinar elementos.									
Elementos devueltos (59) < 1 2 > 									
<input type="checkbox"/>	name	climate	diameter	gravity	orbital...	population	rotation...	surface...	terrain
<input type="checkbox"/>	Dantooine	[{"S":"te...	9830	[{"S":"1 st...	378	1000	25		[{"S":"oceans"}, {"S":"savanna"}, {"S":" mountains"}, {"S":" grasslands"}]
<input type="checkbox"/>	Serenno								[{"S":"rainforests"}, {"S":" rivers"}, {"S":" mountains"}]
<input type="checkbox"/>	Yavin IV	[{"S":"te...	10200	[{"S":"1 st...	4818	1000	24	8	[{"S":"jungle"}, {"S":" rainforests"}]
<input type="checkbox"/>	Cato Neim...	[{"S":"te...	0	[{"S":"1 st...	278	100000000	25		[{"S":"mountains"}, {"S":" fields"}, {"S":" forests"}, {"S":" rock arches"}]
<input type="checkbox"/>	Muunilint	[{"S":"te...	13800	[{"S":"1 st...	412	5000000000	28	25	[{"S":"plains"}, {"S":" forests"}, {"S":" hills"}, {"S":" mountains"}]
<input type="checkbox"/>	Utapau	[{"S":"te...	12900	[{"S":"1 st...	351	950000000	27	0	[{"S":"scrublands"}, {"S":" savanna"}, {"S":" canyons"}, {"S":" sinkholes"}]
<input type="checkbox"/>	Coruscant	[{"S":"te...	12240	[{"S":"1 st...	368	1000000000...	24		[{"S":"cityscape"}, {"S":" mountains"}]
<input type="checkbox"/>	Zolan								
<input type="checkbox"/>	Polis Massa	[{"S":"arti...	0	[{"S":"0.5...	590	1000000	24	0	[{"S":"airless asteroid"}]
<input type="checkbox"/>	Mygeeto	[{"S":"fig...	10088	[{"S":"1 st...	167	19000000	12		[{"S":"glaciers"}, {"S":" mountains"}, {"S":" ice canyons"}]
<input type="checkbox"/>	Naboo	[{"S":"te...	12120	[{"S":"1 st...	312	4500000000	26	12	[{"S":"grassy hills"}, {"S":" swamps"}, {"S":" forests"}, {"S":" mountains"}]
<input type="checkbox"/>	Chandila	[{"S":"te...	13500	[{"S":"1 st...	368	1200000000	20	40	[{"S":"plains"}, {"S":" forests"}]
<input type="checkbox"/>	Eriadu	[{"S":"poll...	13490	[{"S":"1 st...	360	2200000000...	24		[{"S":"cityscape"}]

Figura 7: Captura de pantalla mostrando la lista de items contenidos en la tabla luego de la ejecución del test

4. Operaciones sobre tabla

El siguiente es el código y resultado de una Query para obtener el item con llave de partición "Coruscant"

```

1 // Load the AWS SDK for Node.js
2 var AWS = require('aws-sdk');
3 // Set the region
4 AWS.config.update({region: 'sa-east-1'});
5
6 // Create the DynamoDB service object
7 var ddb = new AWS.DynamoDB();
8
9 exports.handler = async ( event ) => {
10     var params = {
11         TableName: 'tabla',
12         Key: {
13             'name': {S: 'Coruscant'}
14         }
15     };
16
17     // Call DynamoDB to read the item from the table
18     await ddb.getItem(params, function(err, data) {
19         if (err) {
20             console.log("Error", err);
21         } else {
22             console.log("Success", data.Item);
23         }
24     }). promise();
25 };

```

```

1 2022-05-10T21:03:14.161Z 489fc4fe-f085-4f5e-9221-7b734ad88b70 INFO
   Success {
2   diameter: { N: '12240' },
3   rotation_period: { N: '24' },
4   gravity: { L: [ [Object] ] },
5   climate: { L: [ [Object] ] },
6   orbital_period: { N: '368' },
7   population: { N: '1000000000000' },
8   name: { S: 'Coruscant' },
9   terrain: { L: [ [Object], [Object] ] }
10 }

```

El siguiente código es el utilizado para agregar un nuevo item con llave de partición "Hoth":

```

1 // Load the AWS SDK for Node.js
2 var AWS = require('aws-sdk');
3 // Set the region
4 AWS.config.update({region: 'sa-east-1'});
5
6 // Create the DynamoDB service object
7 var ddb = new AWS.DynamoDB();

```

```

8
9 exports.handler = async ( event ) => {
10   var params = {
11     TableName: 'tabla',
12     Item: {
13       'name' : {S: 'Hoth'},
14       'diameter' : {N: '7200'},
15       'rotation_period' : {N: '23'},
16       'gravity' : {L: [{
17         S: "1.1 standard"
18       }]},
19       'climate' : {L: [{
20         S: "frozen"
21       }]},
22       'surface_water' : {N: '100'},
23       'orbital_period' : {N: '549'},
24       'terrain' : {L: [
25         {S: "tundra"},
26         {S: "ice caves"},
27         {S: "mountain ranges"}
28       ]},
29     }
30   };
31
32   // Call DynamoDB to read the item from the table
33   await ddb.putItem(params, function(err, data) {
34     if (err) {
35       console.log("Error", err);
36     } else {
37       console.log("Success", data);
38     }
39   }). promise();
40 };

```

El siguiente código y resultado muestra un Query para obtener el nombre y población de todos los planetas en la lista que tienen periodo orbital entre 500 y 1000 días:

```

1 // Load the AWS SDK for Node.js
2 var AWS = require('aws-sdk');
3 // Set the region
4 AWS.config.update({region: 'sa-east-1'});
5
6 // Create the DynamoDB service object
7 var ddb = new AWS.DynamoDB();
8
9 exports.handler = async ( event ) => {
10   var params = {
11     TableName: 'tabla',
12     FilterExpression: "orbital_period <= :b and orbital_period
13       >= :a",
14     ExpressionAttributeValues: {
15       ':a': {N: '500'},
16       ':b': {N: '1000'}
17     },

```



```

17     ExpressionAttributeNames : {
18         "#c": "name",
19         "#d": "population"
20     },
21     ProjectionExpression: "#c, #d"
22 };
23
24
25 // Call DynamoDB to read the item from the table
26 await ddb.scan(params, function(err, data) {
27     if (err) {
28         console.log("Error", err);
29     } else {
30         console.log("Success", data.Items);
31     }
32 }).promise();
33 };

```

```

1 2022-05-11T23:27:58.862Z    f111298e-75ea-4d45-b9fd-5a27e310243c    INFO
    Success [
2    { name: { S: 'Polis Massa' }, population: { N: '1000000' } },
3    { name: { S: 'Bestine IV' }, population: { N: '62000000' } },
4    { name: { S: 'Hoth' } }
5 ]

```

El siguiente código re-escribe los items agregando una columna “string” que clasifica a los planetas por clase según el orden de magnitud de su población y una columna “number” que clasifica los planetas según su periodo orbital:

```

1 // Load the AWS SDK for Node.js
2 var AWS = require('aws-sdk');
3 // Set the region
4 AWS.config.update({region: 'sa-east-1'});
5
6 // Create the DynamoDB service object
7 var ddb = new AWS.DynamoDB();
8
9 exports.handler = async ( event ) => {
10     let tableContents;
11     try{
12         //get items from dynamo
13         const params = {
14             TableName: `${'tabla'}`,
15         };
16         tableContents = await scanDB(params);
17     }catch(err){
18         console.log(err);
19         return err;
20     }
21
22     let calls = [];
23     tableContents.forEach(function(value){
24         //population
25         let params;

```

```

26     if (value["population"] == undefined){
27         params = {
28             ExpressionAttributeValues: {
29                 ":newAttribute": {S: ''}
30             },
31             Key: {
32                 "name": value.name
33             },
34             TableName: '${'tabla'}',
35             UpdateExpression: "SET class_population = :newAttribute"
36         };
37     }
38     else if (Number(value["population"].N) <= 1000){
39         params = {
40             ExpressionAttributeValues: {
41                 ":newAttribute": {S: 'class A'}
42             },
43             Key: {
44                 "name": value.name
45             },
46             TableName: '${'tabla'}',
47             UpdateExpression: "SET class_population = :newAttribute"
48         };
49     }
50     else if (Number(value["population"].N) <= 1000000){
51         params = {
52             ExpressionAttributeValues: {
53                 ":newAttribute": {S: 'class B'}
54             },
55             Key: {
56                 "name": value.name
57             },
58             TableName: '${'tabla'}',
59             UpdateExpression: "SET class_population = :
60                 newAttribute",
61         };
62     }
63     else if (Number(value["population"].N) <= 1000000000){
64         params = {
65             ExpressionAttributeValues: {
66                 ":newAttribute": {S: 'class C'}
67             },
68             Key: {
69                 "name": value.name
70             },
71             TableName: '${'tabla'}',
72             UpdateExpression: "SET class_population = :
73                 newAttribute",
74         };
75     }
76     else{
77         params = {

```

```

76     ExpressionAttributeValues: {
77         ":newAttribute": {S: 'clase D'}
78     },
79     Key: {
80         "name": value.name
81     },
82     TableName: '${'tabla'}',
83     UpdateExpression: "SET class_population = :newAttribute",
84     };
85 }
86 calls.push(ddb.updateItem(params).promise());
87
88 //orbital_period
89 let params2;
90 if (value["orbital_period"] == undefined){
91     params2 = {
92         ExpressionAttributeValues: {
93             ":newAttribute": {S: ''}
94         },
95         Key: {
96             "name": value.name
97         },
98         TableName: '${'tabla'}',
99         UpdateExpression: "SET class_orbital = :newAttribute",
100     };
101 }
102 else if (Number(value["orbital_period"].N) <= 250){
103     params2 = {
104         ExpressionAttributeValues: {
105             ":newAttribute": {S: 'clase A'}
106         },
107         Key: {
108             "name": value.name
109         },
110         TableName: '${'tabla'}',
111         UpdateExpression: "SET class_orbital = :newAttribute",
112     };
113 }
114 else if (Number(value["orbital_period"].N) <= 500){
115     params2 = {
116         ExpressionAttributeValues: {
117             ":newAttribute": {S: 'clase B'}
118         },
119         Key: {
120             "name": value.name
121         },
122         TableName: '${'tabla'}',
123         UpdateExpression: "SET class_orbital = :newAttribute",
124     };
125 }
126 else if (Number(value["orbital_period"].N) <= 750){
127     params2 = {
128         ExpressionAttributeValues: {
129             ":newAttribute": {S: 'clase C'}

```

```

130         },
131         Key: {
132             "name": value.name
133         },
134         TableName: `${'tabla'}`,
135         UpdateExpression: "SET class_orbital = :newAttribute",
136     };
137 }
138 else if(Number(value["orbital_period"].N) <= 1000){
139     params2 = {
140         ExpressionAttributeValues: {
141             ":newAttribute": {S: 'class D'}
142         },
143         Key: {
144             "name": value.name
145         },
146         TableName: `${'tabla'}`,
147         UpdateExpression: "SET class_orbital = :newAttribute",
148     };
149 }
150 else{
151     params2 = {
152         ExpressionAttributeValues: {
153             ":newAttribute": {S: 'class E'}
154         },
155         Key: {
156             "name": value.name
157         },
158         TableName: `${'tabla'}`,
159         UpdateExpression: "SET class_orbital = :newAttribute",
160     };
161 }
162 calls.push(ddb.updateItem(params2).promise());
163 });
164
165 let response;
166 try{
167     response = await Promise.all(calls);
168 }catch(err){
169     console.log(err);
170 }
171 return response;
172 };
173
174 async function scanDB(params) {
175     let dynamoContents = [];
176     let items;
177     do{
178         items = await ddb.scan(params).promise();
179         items.Items.forEach((item) => dynamoContents.push(item));
180         params.ExclusiveStartKey = items.LastEvaluatedKey;
181     }while(typeof items.LastEvaluatedKey != "undefined");
182     return dynamoContents;
183 }

```

El siguiente código y resultado muestra un Scan para obtener el nombre y diámetro de todos los planetas en la lista que son de clase B o C:

```

1 // Load the AWS SDK for Node.js
2 var AWS = require('aws-sdk');
3 // Set the region
4 AWS.config.update({region: 'sa-east-1'});
5
6 // Create the DynamoDB service object
7 var ddb = new AWS.DynamoDB();
8
9 exports.handler = async ( event ) => {
10     var params = {
11         TableName: 'tabla',
12         FilterExpression: 'contains(class_orbital, :B) OR
                           contains(class_orbital, :C) OR contains(
                           class_population, :B) OR contains(class_population, :
                           C)',
13         ExpressionAttributeValues: {
14             ':B': {
15                 S: 'clase B'
16             },
17             ':C': {
18                 S: 'clase C'
19             }
20         },
21         ExpressionAttributeNames : {
22             "#n": "name",
23             "#d": "diameter"
24         },
25         ProjectionExpression: "#n, #d"
26     };
27
28     await ddb.scan(params, function(err, data) {
29         if (err) console.log(err); // an error occurred
30         else console.log(data.Items); // successful response
31     }).promise();
32 };

```

```

1 2022-05-13T00:09:57.147Z 4ec8b4d0-f087-43b6-b755-4bd73bd950a7 INFO
  [
2   { diameter: { N: '9830' }, name: { S: 'Dantooine' } },
3   { diameter: { N: '0' }, name: { S: 'Cato Neimoidia' } },
4   { diameter: { N: '13800' }, name: { S: 'Muunilinst' } },
5   { diameter: { N: '12900' }, name: { S: 'Utapau' } },
6   { diameter: { N: '12240' }, name: { S: 'Coruscant' } },
7   { diameter: { N: '0' }, name: { S: 'Polis Massa' } },
8   { diameter: { N: '10088' }, name: { S: 'Mygeeto' } },
9   { diameter: { N: '12120' }, name: { S: 'Naboo' } },
10  { diameter: { N: '13500' }, name: { S: 'Chandrila' } },
11  { diameter: { N: '13490' }, name: { S: 'Eriadu' } },
12  { diameter: { N: '19720' }, name: { S: 'Kamino' } },
13  { diameter: { N: '8900' }, name: { S: 'Dagobah' } },
14  { diameter: { N: '11030' }, name: { S: 'Mon Cala' } },

```

```

15 { name: { S: 'Champala' } },
16 { diameter: { N: '10120' }, name: { S: 'Haruun Kal' } },
17 { diameter: { N: '7549' }, name: { S: 'Rodia' } },
18 { diameter: { N: '12780' }, name: { S: 'Sullust' } },
19 { diameter: { N: '13850' }, name: { S: 'Kalee' } },
20 { diameter: { N: '12765' }, name: { S: 'Kashyyyk' } },
21 { diameter: { N: '13400' }, name: { S: 'Dorin' } },
22 { diameter: { N: '11370' }, name: { S: 'Geonosis' } },
23 { diameter: { N: '14900' }, name: { S: 'Vulpter' } },
24 { name: { S: 'Iktotch' } },
25 { name: { S: 'Ojom' } },
26 { diameter: { N: '6400' }, name: { S: 'Bestine IV' } },
27 { diameter: { N: '4200' }, name: { S: 'Mustafar' } },
28 { diameter: { N: '14920' }, name: { S: 'Saleucami' } },
29 { diameter: { N: '4900' }, name: { S: 'Endor' } },
30 { diameter: { N: '0' }, name: { S: 'Trandosha' } },
31 { diameter: { N: '10465' }, name: { S: 'Tatooine' } },
32 { diameter: { N: '12150' }, name: { S: 'Nal Hutta' } },
33 { diameter: { N: '118000' }, name: { S: 'Bespin' } },
34 { diameter: { N: '12500' }, name: { S: 'Alderaan' } },
35 { diameter: { N: '0' }, name: { S: 'Socorro' } },
36 { diameter: { N: '10600' }, name: { S: 'Ryloth' } },
37 { name: { S: 'Cerea' } },
38 { name: { S: 'Skako' } },
39 { diameter: { N: '11000' }, name: { S: 'Corellia' } },
40 { name: { S: 'Iridonia' } },
41 { diameter: { N: '7200' }, name: { S: 'Hoth' } },
42 { diameter: { N: '15600' }, name: { S: 'Glee Anselm' } },
43 { diameter: { N: '9100' }, name: { S: 'Felucia' } },
44 { diameter: { N: '7900' }, name: { S: 'Toydaria' } },
45 { diameter: { N: '14050' }, name: { S: 'Ord Mantell' } },
46 { diameter: { N: '10480' }, name: { S: 'Dathomir' } }
47 ]

```

5. Operaciones sobre indice

El siguiente código y resultado muestra un Query para obtener todos los planetas población clase C y periodo orbital clase B:

```

1 // Load the AWS SDK for Node.js
2 var AWS = require('aws-sdk');
3 // Set the region
4 AWS.config.update({region: 'sa-east-1'});
5
6 // Create the DynamoDB service object
7 var ddb = new AWS.DynamoDB();
8
9 exports.handler = async ( event ) => {
10     var params = {
11         TableName: 'tabla',

```

```

12     IndexName: "class_population-class_orbital-index",
13     FilterExpression: 'contains(class_orbital, :B) AND
14         contains(class_population, :C)',
15     ExpressionAttributeValues: {
16         ':B': {
17             S: 'clase B'
18         },
19         ':C': {
20             S: 'clase C'
21         }
22     };
23
24     await ddb.scan(params, function(err, data) {
25         if (err) console.log(err); // an error occurred
26         else console.log(data.Items); // successful response
27     }).promise();
28 };

```

```

1 2022-05-13T23:17:02.248Z 08b6d988-1570-4950-95b6-ce67b12e7f1c INFO
2  [
3  {
4     diameter: { N: '0' },
5     population: { N: '10000000' },
6     class_population: { S: 'clase C' },
7     name: { S: 'Cato Neimoidia' },
8     class_orbital: { S: 'clase B' }
9  },
10 {
11     diameter: { N: '12900' },
12     population: { N: '95000000' },
13     class_population: { S: 'clase C' },
14     name: { S: 'Utapau' },
15     class_orbital: { S: 'clase B' }
16 },
17 {
18     diameter: { N: '19720' },
19     population: { N: '1000000000' },
20     class_population: { S: 'clase C' },
21     name: { S: 'Kamino' },
22     class_orbital: { S: 'clase B' }
23 },
24 {
25     diameter: { N: '12765' },
26     population: { N: '45000000' },
27     class_population: { S: 'clase C' },
28     name: { S: 'Kashyyyk' },
29     class_orbital: { S: 'clase B' }
30 },
31 {
32     diameter: { N: '14900' },
33     population: { N: '421000000' },
34     class_population: { S: 'clase C' },
35     name: { S: 'Vulpter' },

```

```

35     class_orbital: { S: 'clase B' }
36   },
37   {
38     diameter: { N: '4900' },
39     population: { N: '30000000' },
40     class_population: { S: 'clase C' },
41     name: { S: 'Endor' },
42     class_orbital: { S: 'clase B' }
43   },
44   {
45     diameter: { N: '0' },
46     population: { N: '42000000' },
47     class_population: { S: 'clase C' },
48     name: { S: 'Trandosha' },
49     class_orbital: { S: 'clase B' }
50   },
51   {
52     diameter: { N: '0' },
53     population: { N: '300000000' },
54     class_population: { S: 'clase C' },
55     name: { S: 'Socorro' },
56     class_orbital: { S: 'clase B' }
57   },
58   {
59     population: { N: '450000000' },
60     class_population: { S: 'clase C' },
61     name: { S: 'Cerea' },
62     class_orbital: { S: 'clase B' }
63   }
64 ]

```

El siguiente código y resultado muestra las operaciones necesarias para obtener todos los datos del planeta con menor densidad de población cuya población es de clase D y periodo orbital clase B:

```

1  // Load the AWS SDK for Node.js
2  var AWS = require('aws-sdk');
3  // Set the region
4  AWS.config.update({region: 'sa-east-1'});
5
6  // Create the DynamoDB service object
7  var ddb = new AWS.DynamoDB();
8
9  exports.handler = async ( event ) => {
10     var params = {
11         TableName: 'tabla',
12         IndexName: "class_population-class_orbital-index",
13         FilterExpression: 'contains(class_orbital, :B) AND
14                           contains(class_population, :D)',
15         ExpressionAttributeValues: {
16             ':B': {
17                 S: 'clase B'
18             },
19             ':D': {
20                 N: '450000000'
21             }
22         }
23     };
24     return ddb.query(params).promise();
25 }

```



```
18         'D': {
19             S: 'clase D'
20         }
21     }
22 };
23 let items;
24 let poblacion = Number.POSITIVE_INFINITY;
25 let item_final;
26 items = await ddb.scan(params).promise();
27 items.Items.forEach(item => {
28     if(Number(item['population'].N) < poblacion){
29         poblacion = Number(item['population'].N);
30         item_final = item;
31     }
32 });
33 return item_final;
34 };
```

```
1 {
2     "diameter": {
3         "N": "13500"
4     },
5     "population": {
6         "N": "1200000000"
7     },
8     "class_population": {
9         "S": "clase D"
10    },
11    "name": {
12        "S": "Chandrigla"
13    },
14    "class_orbital": {
15        "S": "clase B"
16    }
17 }
```