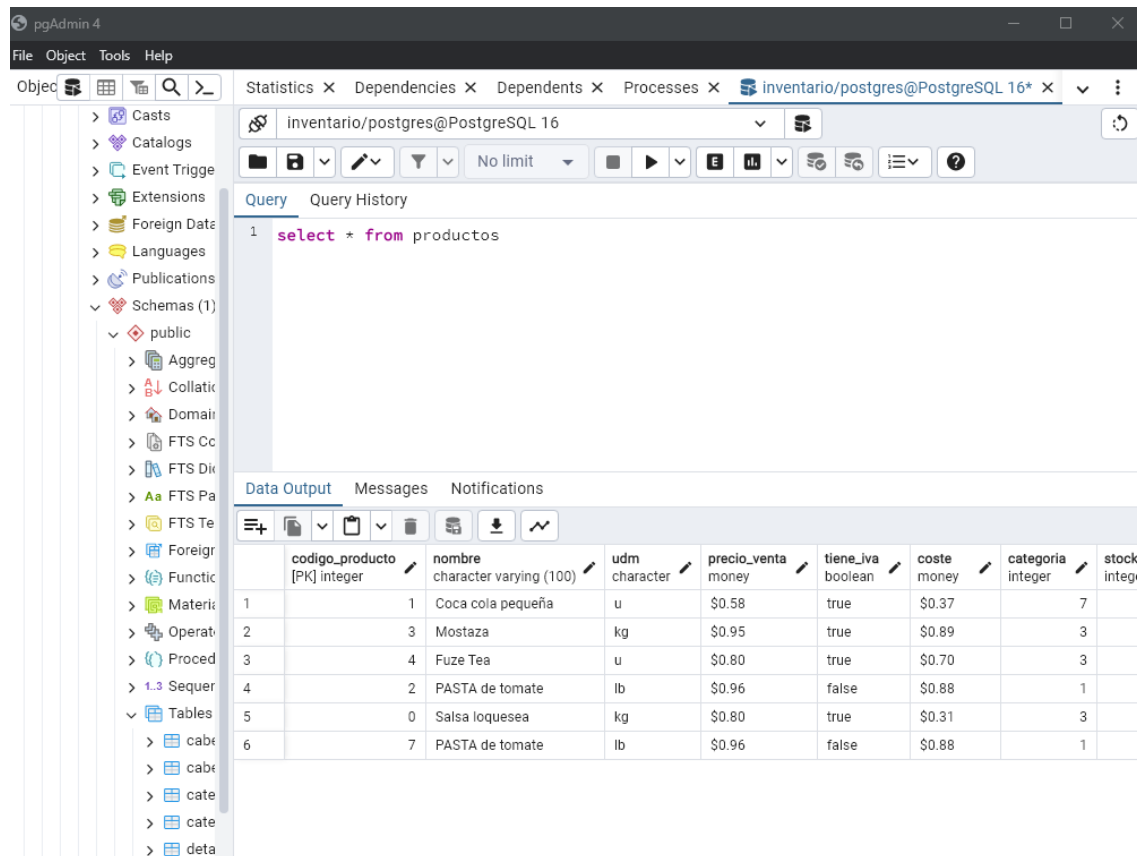


REPASO MODULO 3 – PARTE 1

VALIDACION DEL AMBIENTE

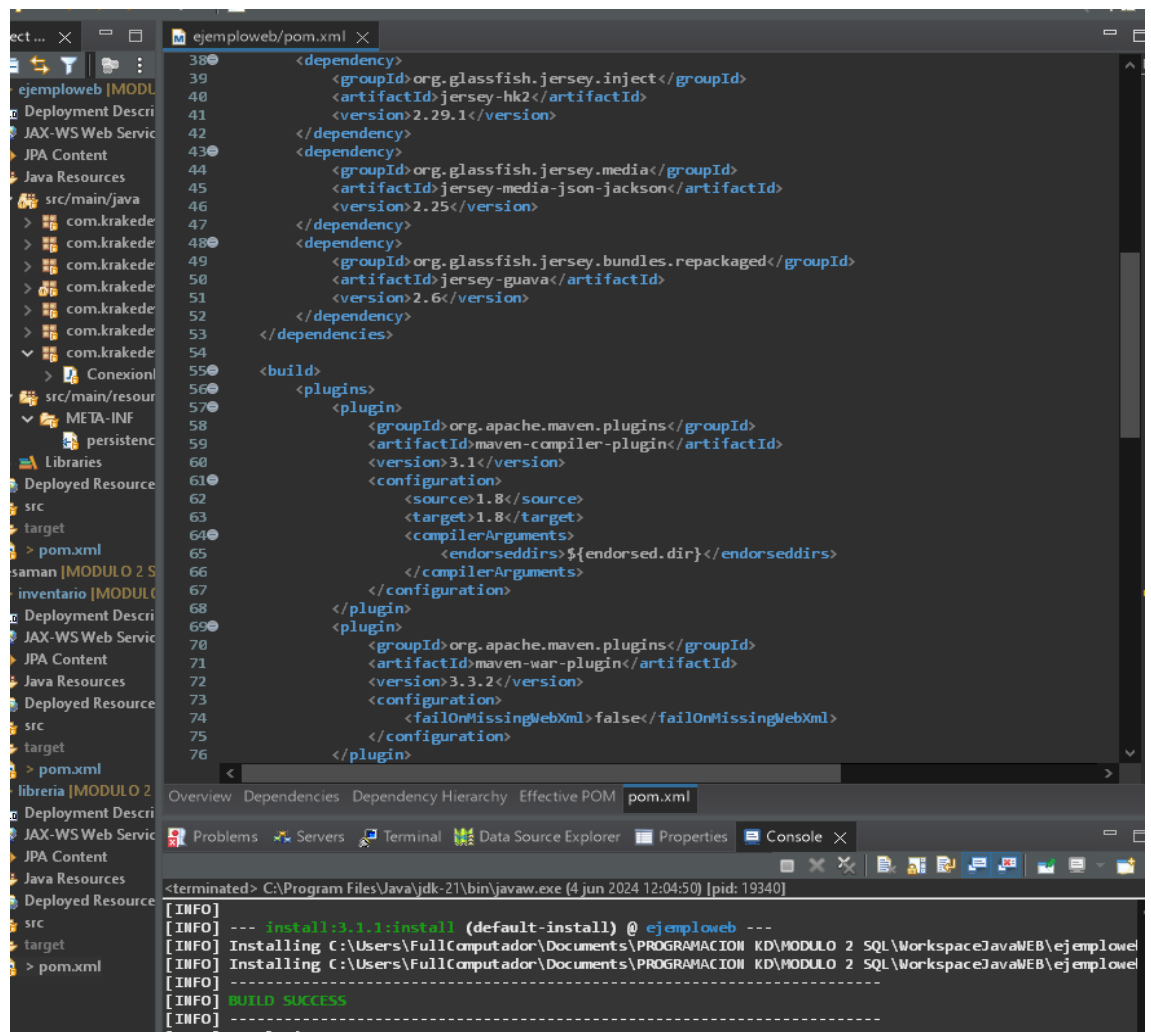
BASE DE DATOS:



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including Schemas (public), Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Plugins, Foreign Data Wrappers, Functions, Materials, Operators, Procedures, Sequences, and Tables. The main pane shows a query result for the 'productos' table. The query is 'select * from productos'. The result is a table with 10 columns: 'codigo_producto' (integer), 'nombre' (character varying (100)), 'udm' (character), 'precio_venta' (money), 'tiene_iva' (boolean), 'coste' (money), 'categoria' (integer), and 'stock' (integer). The data is as follows:

codigo_producto	nombre	udm	precio_venta	tiene_iva	coste	categoria	stock
1	Coca cola pequeña	u	\$0.58	true	\$0.37	7	
2	Mostaza	kg	\$0.95	true	\$0.89	3	
3	Fuze Tea	u	\$0.80	true	\$0.70	3	
4	PASTA de tomate	lb	\$0.96	false	\$0.88	1	
5	Salsa loquesea	kg	\$0.80	true	\$0.31	3	
6	PASTA de tomate	lb	\$0.96	false	\$0.88	1	

ECLIPSE:



The screenshot shows the Eclipse IDE. The left sidebar displays the project structure, including 'ejemploweb' (MODULO 2), 'Deployment Descriptor', 'JAX-WS Web Service', 'JPA Content', 'Java Resources', 'src/main/java', 'com.krakade', 'Conexión', 'src/main/resources', 'META-INF', 'persistencia', 'Libraries', 'Deployed Resource', 'src', 'target', 'pom.xml', 'saman' (MODULO 2), 'inventario' (MODULO 2), 'Deployment Descriptor', 'JAX-WS Web Service', 'JPA Content', 'Java Resources', 'Deployed Resource', 'src', 'target', 'pom.xml', 'libreria' (MODULO 2), 'Deployment Descriptor', 'JAX-WS Web Service', 'JPA Content', 'Java Resources', 'Deployed Resource', 'src', 'target', 'pom.xml'. The main pane shows the 'pom.xml' file for 'ejemploweb'. The content is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<groupId>org.example</groupId>
<artifactId>ejemploweb</artifactId>
<version>1.0</version>
<packaging>war</packaging>
<dependencies>
<dependency>
<groupId>org.glassfish.jersey.inject</groupId>
<artifactId>jersey-hk2</artifactId>
<version>2.29.1</version>
</dependency>
<dependency>
<groupId>org.glassfish.jersey.media</groupId>
<artifactId>jersey-media-json-jackson</artifactId>
<version>2.25</version>
</dependency>
<dependency>
<groupId>org.glassfish.jersey.bundles.repackaged</groupId>
<artifactId>jersey-guava</artifactId>
<version>2.6</version>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.1</version>
<configuration>
<source>1.8</source>
<target>1.8</target>
<compilerArguments>
<endorseddirs>${endorsed.dir}</endorseddirs>
</compilerArguments>
</configuration>
</plugin>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-war-plugin</artifactId>
<version>3.3.2</version>
<configuration>
<failOnMissingWebXml>false</failOnMissingWebXml>
</configuration>
</plugin>
</plugins>
</build>
</project>
```

The bottom pane shows the Maven console output, indicating that the build was successful.

```
<terminated> C:\Program Files\Java\jdk-21\bin\javaw.exe [4 jun 2024 12:04:50] [pid: 19340]
[INFO] --- install:3.1.1:install (default-install) @ ejemploweb ---
[INFO] Installing C:\Users\FullComputador\Documents\PROGRAMACION_KD\MODULO 2 SQL\WorkspaceJavaWEB\ejemploweb\ejemploweb-1.0.war to C:\Users\FullComputador\Documents\PROGRAMACION_KD\MODULO 2 SQL\WorkspaceJavaWEB\ejemploweb\ejemploweb-1.0.war
[INFO] BUILD SUCCESS
[INFO]
```

TOMCAT:

The collage illustrates the development of a REST API for an inventory system. It is divided into four quadrants, each showing a different aspect of the development process:

- Top-Left:** A screenshot of the Postman interface showing a GET request to `http://localhost:8080/inventarios-1.0/inventarios-1`. The response is a JSON array of inventory items, including details like `identificador`, `tipodocumento`, `nombre`, `telefono`, `correo`, and `direccion`.
- Top-Right:** A screenshot of the PostgreSQL query editor showing a query: `select * from proveedor where nombre like 'Peli'`. The results table shows two rows of data for providers named 'Pepi Pulgas' and 'Pepi Pedro'.
- Bottom-Left:** A screenshot of the Postman interface showing a POST request to `http://localhost:8080/inventarios-1.0/rest/`. The request body is a JSON object representing a new inventory item.
- Bottom-Right:** A screenshot of the PostgreSQL query editor showing a query: `select * from proveedor where nombre like 'Pa3'`. The results table shows two rows of data for providers named 'Paco' and 'Papo Pedro'.

PUT http://localhost:8080/inventarios-1.0.0/rest/... Send

Body raw JSON Beautify

```
1 {
2   "codigo": 7,
3   "nombre": "PASTA de jijas",
4   "unidadMedida": {
5     "nombre": "lb"
6   },
7   "precioVenta": 0.96,
8   "tieneIva": false,
9   "coste": 0.8800,
10  "categoria": {
11    "codigo": 1
12  }
13 }
```

Body 200 OK 6 ms 123 B Save as example

Pretty Raw Preview Visualize Text

- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (12)
- cabecera_pedidos
- cabecera_ventas
- categorias
- categorias_unidad_medida
- detalle_pedidos
- detalle_ventas
- estado_pedidos
- historial_stock

```
3 select * from tipo_de_documento
4 select * from productos
5 select * from productos where nombre like 'Sal%'
```

Data Output Messages Notifications

	codigo_producto [PK] integer	nombre character varying (100)	udm character	precio money
1	1	Coca cola pequeña	u	\$0.58
2	3	Mostaza	kg	\$0.95
3	4	Fuze Tea	u	\$0.80
4	2	PASTA de tomate	lb	\$0.96
5	0	Salsa jijas	kg	\$0.80
6	7	PASTA de jijas	lb	\$0.96

GET http://localhost:8080/inventarios-1.0.0/rest/... Send

Params Query Params

Key	Value	Des...	Bulk Edit
Key	Value	Description	

Body 200 OK 48 ms 424 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "codigo": 3,
3   "nombre": "Mostaza",
4   "unidadMedida": {
5     "nombre": "kg",
6     "descripcion": "kilogramos",
7     "categoriaUnidadMedida": {
8       "codigo": "P",
9       "nombre": null
10    }
11  }
```

- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (12)
- cabecera_pedidos
- cabecera_ventas
- categorias
- categorias_unidad_medida
- detalle_pedidos
- detalle_ventas
- estado_pedidos
- historial_stock
- productos
- proveedor
- tipo_de_documento
- unidades_de_medida
- Trigger Functions
- Types
- Views

```
3 select * from productos
4 select * from productos where
```

Data Output Messages Notifications

	codigo_producto [PK] integer	nombre character varying (100)
1	1	Coca cola pequeña
2	3	Mostaza
3	4	Fuze Tea
4	2	PASTA de tomate
5	0	Salsa jijas
6	7	PASTA de jijas

POST http://localhost:8080/inventarios-1.0.0/rest/... Send

Body raw JSON Beautify

```
1 {
2   "producto": {
3     "codigo": 1,
4     "precioVenta": 0.58
5   },
6   "cantidadSolicitada": 28
7 },
8 {
9   "producto": {
10    "codigo": 2,
11    "precioVenta": 0.95
12  },
13   "cantidadSolicitada": 56
14 }
15 }
```

Body 200 OK 16 ms 123 B Save as example

Pretty Raw Preview Visualize Text

- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (12)
- cabecera_pedidos
- cabecera_ventas
- categorias
- categorias_unidad_medida
- detalle_pedidos
- detalle_ventas
- estado_pedidos
- historial_stock
- productos
- proveedor
- tipo_de_documento
- unidades_de_medida
- Trigger Functions
- Types
- Views

```
3 select * from tipo_de_documento
4 select * from productos
5 select * from productos where nombre like 'Sal%'
6 select * from detalle_pedidos
```

Data Output Messages Notifications

	codigo_pedido [PK] integer	cabecera_pedido integer	producto integer	cantidad integer
1	1	1	1	100
2	2	1	4	50
3	3	2	1	10
4	6	10	1	28
5	7	10	2	56
6	4	9	1	100
7	5	9	2	50
8	8	11	1	28
9	9	11	2	56
10	10	12	1	28
11	11	12	2	56
12	12	14	1	28
13	13	14	2	56

PUT http://localhost:8080/inventarios-1.0.0/res...

Body

raw JSON Beautify

```
0 {
1   "producto": {
2     "codigo": 1,
3     "precioVenta": 0.58
4   },
5   "cantidadRecibida": 10
6 },
7 {
8   "codigo": 11,
9   "producto": {
10     "codigo": 2,
11     "precioVenta": 0.95
12   },
13   "cantidadRecibida": 598
14 },
15 ...
16 ]
17 }
```

body 200 OK 17 ms 123 B Save as example

Pretty Raw Preview Visualize Text

GET http://localhost:8080/inventarios-1.0.0/res...

Params

Query Params

Key	Value	Des...	Bulk Edit
Key	Value	Description	

Body

200 OK 25 ms 2.25 KB Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "codigo": 12,
3   "proveedor": {
4     "identificador": "1792285766",
5     "tipoDocumento": {
6       "codigo": "C",
7       "descripcion": "CEDULA"
8     },
9     "nombre": "Perry Pedro",
10    "telefono": "0992928777",
11    "correo": "pepe89@gmail.com",
12    "direccion": "Floresta"
13  },
14  "fecha": "2024-05-07",
15  ...
16 }
```

body 200 OK 1635 ms 123 B Save as example

Pretty Raw Preview Visualize Text

POST http://localhost:8080/inventarios-1.0.0/rest/...

Body

raw JSON Beautify

```
1 {
2   "detalles": [
3     {
4       "producto": {
5         "codigo": 1,
6         "precioVenta": 0.58,
7         "tieneIva": true
8       },
9       "cantidad": 30
10    },
11    {
12      "producto": {
13        "codigo": 2,
14        "precioVenta": 0.95,
15        "tieneIva": false
16      },
17      "cantidad": 30
18    }
19  ]
20 }
```

body 200 OK 1635 ms 123 B Save as example

Pretty Raw Preview Visualize Text

- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (12)
- cabecera_pedidos
- cabecera_ventas
- categorias
- categorias_unidad_medida
- detalle_pedidos
- detalle_ventas
- estado_pedidos
- historial_stock
- productos
- proveedor
- tipo_de_documento
- unidades_de_medida
- Trigger Functions
- Types
- Views

- Aggregates
- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (12)
- cabecera_pedidos
- cabecera_ventas
- categorias
- categorias_unidad_medida
- detalle_pedidos
- detalle_ventas
- estado_pedidos
- historial_stock
- productos
- proveedor
- tipo_de_documento
- unidades_de_medida
- Trigger Functions
- Types
- Views

- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (12)
- cabecera_pedidos
- cabecera_ventas
- categorias
- categorias_unidad_medida
- detalle_pedidos
- detalle_ventas
- estado_pedidos
- historial_stock
- productos
- proveedor
- tipo_de_documento
- unidades_de_medida
- Trigger Functions
- Types
- Views

```
3 select * from tipo_de_documento
4 select * from productos
5 select * from productos where nombre like 'Sa%'
6 select * from detalle_pedidos
7 select * from detalle_ventas
```

Data Output Messages Notifications

	codigo_pedido [PK] integer	cabecera_pedido integer	producto integer	cantidad integer	subtotal money	cantidad_recibida integer
1	1	1	1	100	\$37.29	100
2	2	1	4	50	\$11.80	50
3	3	2	1	10	\$3.73	10
4	6	10	1	28	\$16.24	0
5	7	10	2	56	\$53.20	0
6	4	9	1	100	\$29.00	50
7	5	9	2	50	\$47.50	50
8	8	11	1	28	\$29.00	50
9	9	11	2	56	\$47.50	50
10	12	14	1	28	\$16.24	0
11	13	14	2	56	\$53.20	0
12	10	12	1	28	\$5.80	10
13	11	12	2	56	\$568.10	598

```
1 select * from proveedor where nombre like 'Pa%'
2 select * from tipo_de_documento
3 select * from productos
4 select * from productos where nombre like 'Sa%'
5 select * from detalle_pedidos
6 select * from detalle_ventas
7 select * from cabecera_pedidos
```

Data Output Messages Notifications

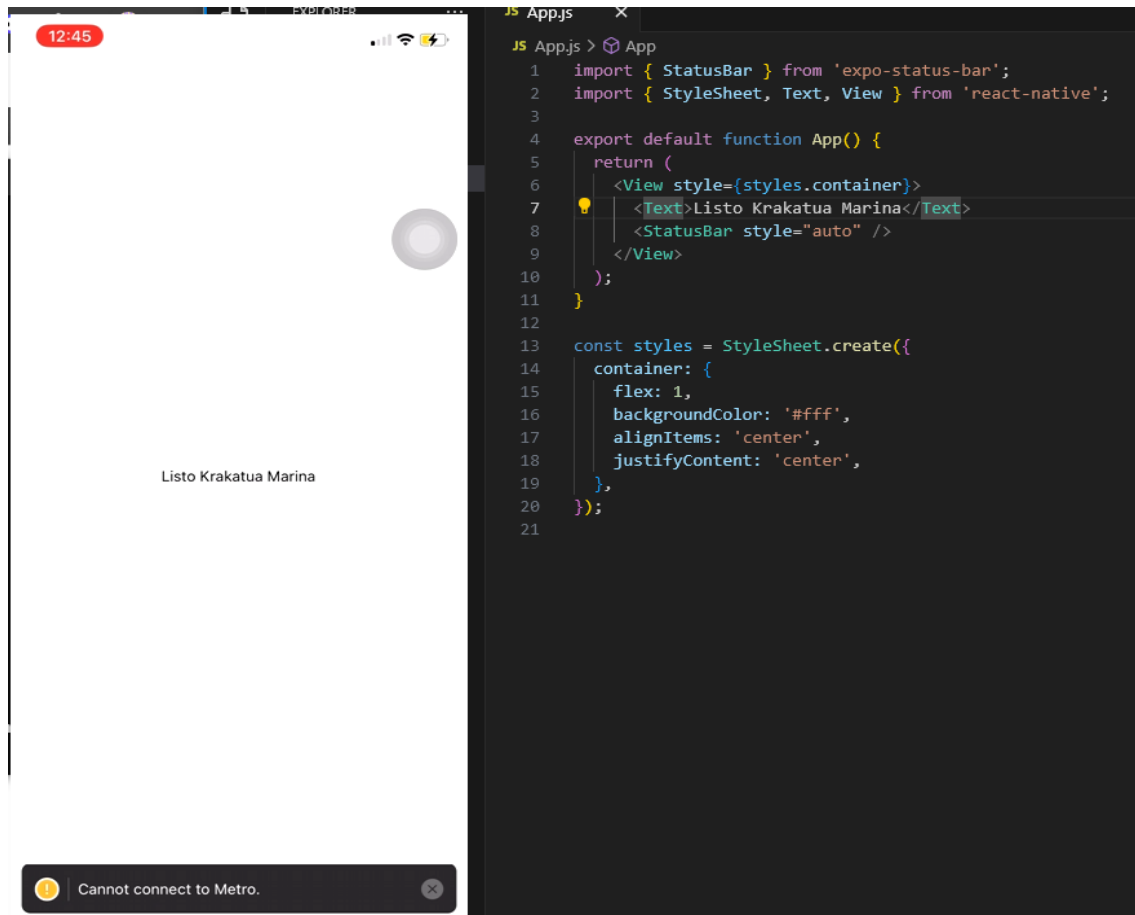
	codigo_cabecera [PK] integer	proveedor character varying (13)	fecha timestamp without time zone	estado character
1	1	1792285747	2023-11-20 00:00:00	R
2	2	1792285747	2023-11-20 00:00:00	R
3	7	1792285747001	2024-05-05 00:00:00	S
4	8	1792285755	2024-05-05 00:00:00	S
5	10	1792285755	2024-05-05 00:00:00	R
6	9	1792285755	2024-05-05 00:00:00	R
7	11	1792285766	2024-05-07 00:00:00	R
8	14	1792285744	2024-06-04 00:00:00	S
9	12	1792285766	2024-05-07 00:00:00	R

```
3 select * from tipo_de_documento
4 select * from productos
5 select * from productos where nombre like 'Sa%'
6 select * from detalle_pedidos
7 select * from detalle_ventas
```

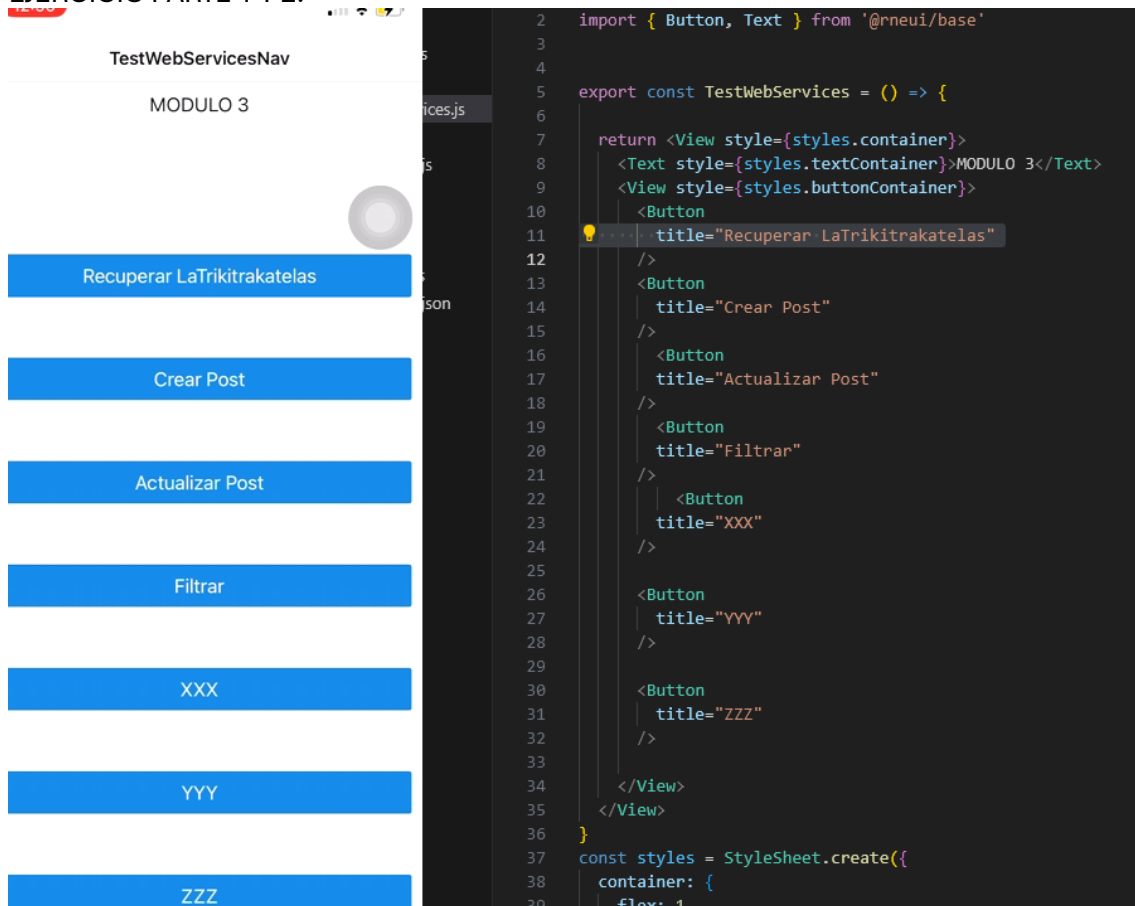
Data Output Messages Notifications

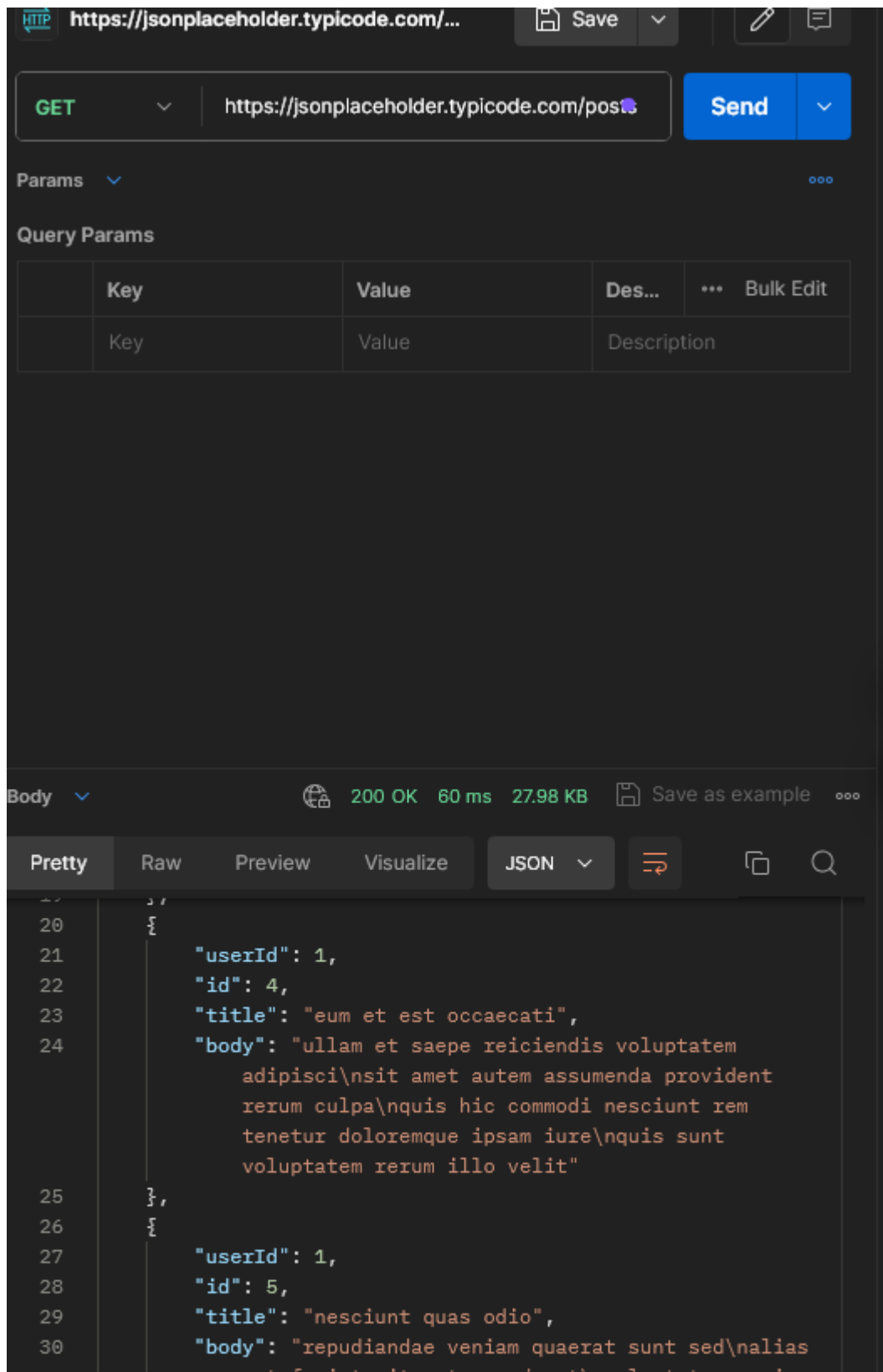
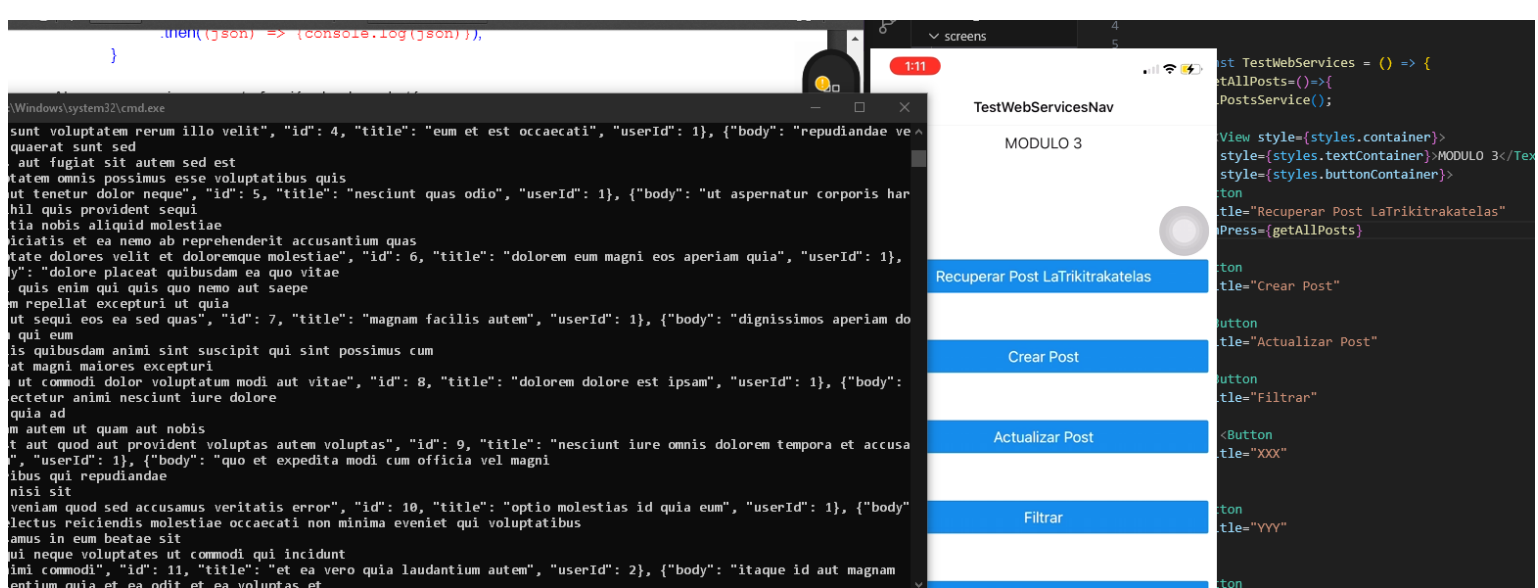
	codigo_detalle [PK] integer	cabecera_ventas integer	producto integer	cantidad integer	precio_venta money	subtotal money	subtotal_cor money
16	16	8	2	5	\$0.95	\$4.75	\$4.75
17	17	9	1	5	\$0.58	\$2.90	\$0.35
18	18	9	2	5	\$0.95	\$4.75	\$4.75
19	19	10	1	5	\$0.58	\$2.90	\$0.35
20	20	10	2	5	\$0.95	\$4.75	\$4.75
21	21	11	1	5	\$0.58	\$2.90	\$0.35
22	22	11	2	5	\$0.95	\$4.75	\$4.75
23	23	12	1	5	\$0.58	\$2.90	\$3.25
24	24	12	2	5	\$0.95	\$4.75	\$4.75
25	25	13	1	15	\$0.58	\$8.70	\$9.74
26	26	13	2	15	\$0.95	\$14.25	\$14.25
27	27	14	1	20	\$0.58	\$11.60	\$12.99
28	28	14	2	20	\$0.95	\$19.00	\$19.00
29	29	15	1	30	\$0.58	\$17.40	\$19.49
30	30	15	2	30	\$0.95	\$28.50	\$28.50
31	31	16	1	30	\$0.58	\$17.40	\$19.49
32	32	16	2	30	\$0.95	\$28.50	\$28.50

REACT NATIVE:



EJERCICIO PARTE 1 Y 2:





EJERCICIO PARTE 3:

```
expedita dolores amet  
sed temporibus distinctio magnam saepe deleniti  
omnis facilis nam ipsum natus sint similique omnis", "id": 94, "title": "qui qui voluptates illo iste minima", "userId":  
10}, {"body": "earum voluptatem facere provident blanditiis velit laboriosam  
pariatur accusamus odio saepe  
cumque dolor qui a dicta ab doloribus consequatur omnis  
corporis cupiditate eaque assumenda ad nesciunt", "id": 95, "title": "id minus libero illum nam ad officis", "userId":  
10}, {"body": "in non odio excepturi sint eum  
labore voluptates vitae quia qui et  
inventore itaque rerum  
veniam non exercitationem delectus aut", "id": 96, "title": "quaerat velit veniam amet cupiditate aut numquam ut sequi",  
"userId": 10}, {"body": "eum non blanditiis soluta porro quibusdam voluptas  
vel voluptates qui placeat dolores qui velit aut  
vel inventore aut cumque culpa explicabo aliquid at  
perspiciatis est et voluptatem dignissimos dolor itaque sit nam", "id": 97, "title": "quas fugiat ut perspiciatis vero p  
resident", "userId": 10}, {"body": "doloremque ex facilis sit sint culpa  
soluta assumenda eligendi non ut eius  
sequi ducimus vel quasi  
veritatis est dolores", "id": 98, "title": "laboriosam dolor voluptates", "userId": 10}, {"body": "quo deleniti praesent  
ium dicta non quod  
aut est molestias  
molestias et officia quis nihil  
itaque dolores quia", "id": 99, "title": "temporibus sit alias delectus eligendi possimus magni", "userId": 10}, {"body":  
: "cupiditate quo est a modi nesciunt soluta  
ipsum voluptate error itaque dicta in  
autem qui minus nesciunt et distinctio eum  
accusamus ratione error aut", "id": 100, "title": "at nam consequatur ea labore ea harum", "userId": 10}]  
6ms C:\Users\FullComputador\Desktop\archivos\consumoRestUS\node_modules\expoAppEntry.js (1 module)  
100 {"body": "suerte en su evaluaci3n", "id": 101, "title": "mensaje", "userId": 1}
```

TestWebServicesNav

MODULO 3

Recuperar Post LaTrik/trakatelas

Crear Post

Actualizar Post

Filtrar

XXX

YYY

ZZZ

```
import { createPostService } from 'TestWebServices';  
  
const config = {  
  headers: {  
    'Content-type': 'application/json',  
  },  
};  
  
fetch('https://jsonplaceholder.typicode.com/posts', config)  
  .then((response) => (return response.json()))  
  .then((json) => (console.log(json)));
```

Rest web services / JSON POST

POST https://jsonplaceholder.typicode.com/posts Send

Params

Query Params

Key	Value	Des...	Bulk Edit
Key	Value	Description	

Body 201 Created 188 ms 1.25 KB Save as example

Pretty Raw Preview Visualize JSON

```
1 {  
2   "title": "foo",  
3   "body": "bar",  
4   "userId": 1,  
5   "id": 101  
6 }
```


EJERCICIO PARTE 4:

The image shows a REST client interface at the top and an Android application interface at the bottom.

REST Client Interface:

- URL: `https://jsonplaceholder.typicode.com/...`
- Method: **PUT**
- Endpoint: `https://jsonplaceholder.typicode.com/posts/1`
- Body (JSON):

```
{
  "id": 1,
  "title": "mensaje final",
  "body": "Hasta la vista baby",
  "userId": 1
}
```
- Status: **200 OK**, 355 ms, 1.18 KB
- Response Body (Pretty):

```
{
  "id": 1,
  "title": "mensaje final",
  "body": "Hasta la vista baby",
  "userId": 1
}
```

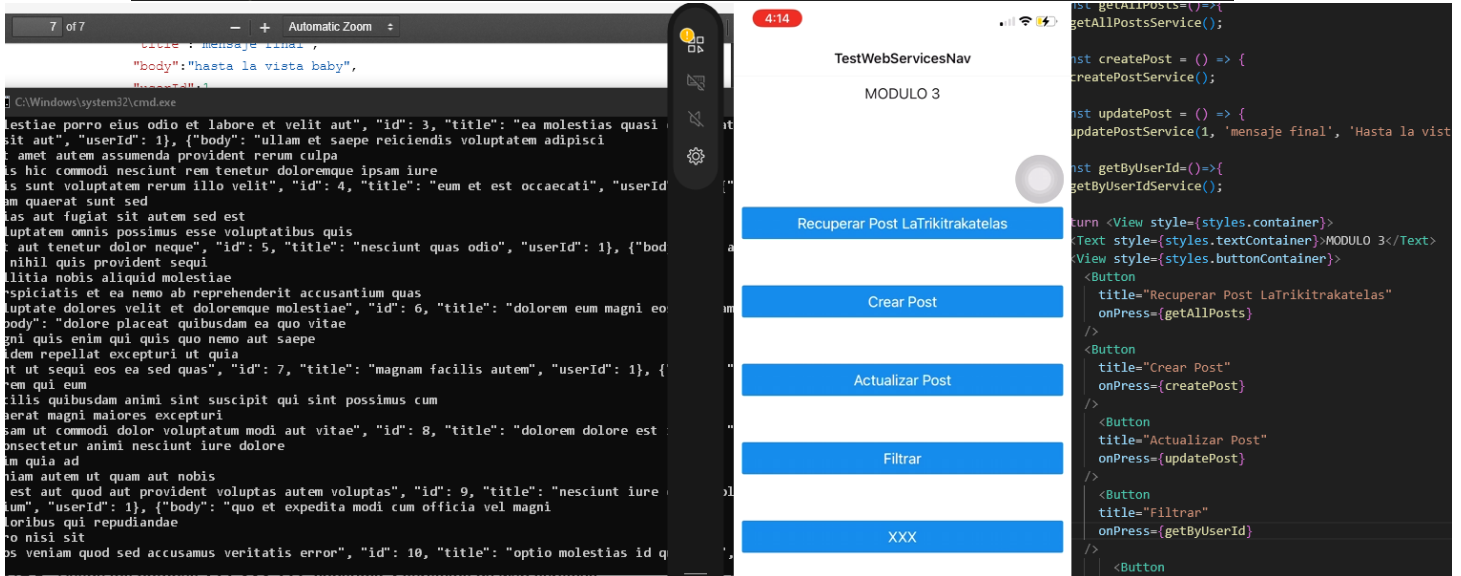
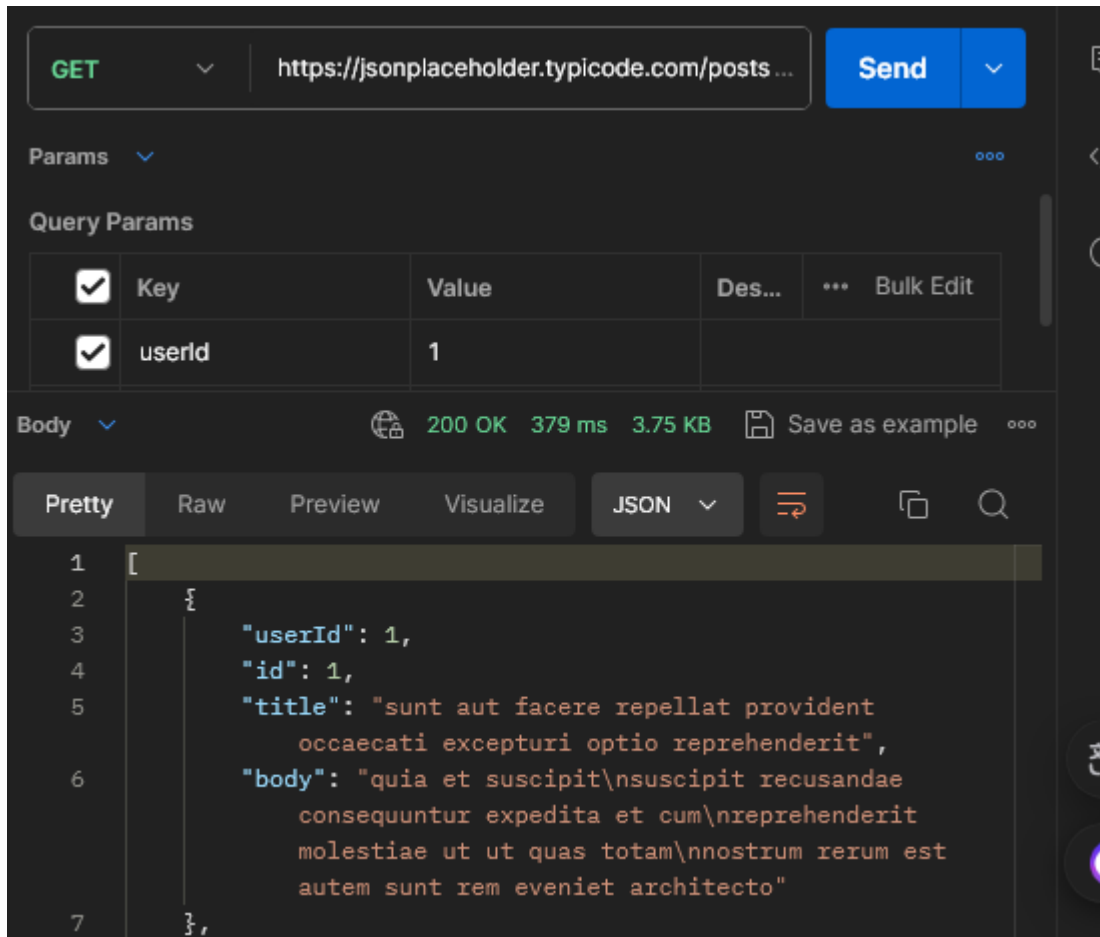
Android Application Interface:

- App Name: **TestWebServicesNav**
- Module: **MODULO 3**
- Buttons:
 - Recuperar Post LaTrikitrakatelas
 - Crear Post
 - Actualizar Post
 - Filtrar

Terminal Output:

```
CA\Windows\system32\cmd.exe
Metro waiting on exp://192.168.100.8:8081
Scan the QR code above with Expo Go (Android) or the Camera app (iOS)
Using Expo Go
Press s | switch to development build
Press a | open Android
Press w | open web
Press j | open debugger
Press r | reload app
Press m | toggle menu
Press o | open project code in your editor
Press ? | show all commands
Logs for your project will appear below. Press Ctrl+C to exit.
LOG {body: "Hasta la vista baby", id: 1, title: "mensaje final", userId: 1}
```

EJERCICIO PARTE 5:



EJERCICIO PARTE 6:

Zippered Pockets: 2 Zippered Hand Pockets, 2 Zippered Pockets on Chest (enough to keep cards or keys)and 1 Hidden Pocket Inside.Zippered Hand Pocket keep your things secure, Humanized Design: Adjustable and Detachable Hood and Adjustable cuff to prevent the wind and water, for a comfortable fit. 3 in 1 Detachable Design provide more convenience, you can separate the coat and inner as needed, or wear it together. It is suitable for different season and help you adapt to different climates", "id": 15, "image": "https://fakestoreapi.com/img/51V5MI-I5tL-AC_UX679-.jpg", "price": 56.99, "rating": {"count": 235, "rate": 2.6}, "title": "BIYLACLESEN Women's 3-in-1 Snowboard Jacket Winter Coats", {"category": "women's clothing", "description": "100% POLYURETHANE(shell) 100% POLYESTER(lining) 75% POLYESTER 25% COTTON (SMEATER), Faux leather material for style and comfort / 2 pockets of front, 2-For-One Hooded denim style faux leather jacket, Button detail on waist / Detail stitching at sides, HAND WASH ONLY / DO NOT BLEACH / LINE DRY / DO NOT IRON", "id": 16, "image": "https://fakestoreapi.com/img/81XH0e8fgf-AC_UY879-.jpg", "price": 99.99, "rating": {"count": 340, "rate": 2.9}, "title": "Lock and Love Women's Removable Hooded Faux Leather Moto Biker Jacket", {"category": "women's clothing", "description": "Lightweight perfect for trip or casual wear---Long sleeve with hooded, adjustable drawstring waist design. Button and zipper front closure raincoat, fully stripes lined and The Raincoat has 2 side pockets are a good size to hold all kinds of things, it covers the hips, and the hood is generous but doesn't overdo it.Attached Cotton Lined Hood with Adjustable Drawstrings give it a real styled look.", "id": 17, "image": "https://fakestoreapi.com/img/71HblAHs-bxL-AC_UY879-.2.jpg", "price": 39.99, "rating": {"count": 679, "rate": 3.8}, "title": "Rain Jacket Women Windbreaker Striped Climbing Raincoats", {"category": "women's clothing", "description": "95% RAYON 5% SPANDEX, Made in USA or Import ed, Do Not Bleach, Lightweight fabric with great stretch for comfort, Ribbed on sleeves and neckline / Double stitching on bottom hem", "id": 18, "image": "https://fakestoreapi.com/img/71z3kpMAYsL-AC_UY879-.jpg", "price": 9.85, "rating": {"count": 130, "rate": 4.7}, "title": "MBJ Women's Solid Short Sleeve Boat Neck V", {"category": "women's clothing", "description": "100% Polyester, Machine wash, 100% cationic polyester interlock, Machine Wash & Pre Shrunk for a Great Fit , Lightweight, roomy and highly breathable with moisture wicking fabric which helps to keep moisture away, Soft Lightweight Fabric with comfortable V-neck collar and a slimmer fit, delivers a sleek, more feminine silhouette and Added Comfort", "id": 19, "image": "https://fakestoreapi.com/img/51eg5euhm5kd-AC_UX679-.jpg", "price": 7.95, "rating": {"count": 146, "rate": 4.5}, "title": "Ogna Women's Short Sleeve Moisture", {"category": "women's clothing", "description": "95% Cotton 5%Spandex, Features: Casual, Short Sleeve, Letter Print,V-Neck,Fashion Tees, The fabric is soft and has some stretch. Occasion: Casual/Office/Beach/School/Home/Street. Season: Spring,Summer,Autumn,Winter.", "id": 20, "image": "https://fakestoreapi.com/img/61PHEJMNML-AC_UX679-.jpg", "price": 12.99, "rating": {"count": 145, "rate": 3.6}, "title": "DAIHOU Womens T Shirt Casual Cotton Short"]]

logs for your project will appear below. Press Ctrl+C to exit.

LOG [electron]: 3860ms C:\Users\FullComputer\Desktop\archivos\consumoRestWS\node_modules\expo\AppEntry.js (1018 ms)

LOG ["electronics", "jewelry", "men's clothing", "women's clothing"]

LOG [{"category": "electroniczzz", "description": "es una prueba pa", "id": 21, "image": "https://i.pravatar.cc", "price": 13.5, "title": "el main producto"}]

LOG [{"category": "electronics", "description": "que te lleva a todos laos", "id": 1, "image": "https://via.placeholder.com/150", "price": 49.99, "title": "la bici"}]

de los botones XXX YYY ZZZ, consumir 3 servicios de web services

usar los siguientes:

pi.com/docs

orris.io/

el

Filtrar

XXX

YYY

ZZZ

products/\${id}, config)

on())

//get en la fake store

export const getFakeStoreProductsService = () => {

fetch('https://fakestoreapi.com/products/categories')

.then(res=>res.json())

.then(json=>console.log(json))

};

// post en la fake store

export const postFakeStoreProductsService = () => {

const config = {

method: 'POST',

body: JSON.stringify({

title: 'el main producto',

price: 13.5,

description: 'es una prueba pa',

image: 'https://i.pravatar.cc',

category: 'electronicozzz'

}),

headers: {

'Content-type': 'application/json',

},

};

return fetch('https://fakestoreapi.com/products', config)

.then((response) => {return response.json()})

.then((json) => {console.log(json)});

};

//put en la fake store

export const putFakeStoreProductsService = (id, title, price, description,

const config = {

method: 'PUT',

body: JSON.stringify({

id,

title,

price,

description,

image,

category,

}),

headers: {

'Content-type': 'application/json',

},

};

return fetch(`https://fakestoreapi.com/products/\${id}`, config)

.then((response) => {return response.json()})

.then((json) => {console.log(json)});

```
onChangeText={() => {
    setValue(value);
    setSubject(value);
}}
>
</>
<Input
    placeholder="MENSAJE"
    value={message}
    onChangeText={() => {
        setMessage(value);
    }}
/>
</>
<Button
    title="Guardar"
    onPress={createPost}
/>
</>
</>
```

The image is a composite showing a mobile application interface on the left and its corresponding JavaScript code on the right.

Mobile App Interface (Left):

- Header:** NUEVO MENSAJE
- Form:** Contains two text input fields labeled "Gata" and "Para", and a blue "Guardar" button at the bottom.
- Confirmation Dialog:** A modal box titled "CONFIRMACION" with the message "Se ha ingresado un nuevo POST" and an "OK" button.

JavaScript Code (Right):

```
POST',
SON.stringify({
: post.title,
post.body,
d: 1,},

alert("CONFIRMACION","Se ha ingresado un nuevo POST")

: {
ent-type': 'application/json',

tps://jsonplaceholder.typicode.com/posts', config)
response) => {return response.json()})
json) => {console.log(json);fnExito();});

t updatePostService = (id, title, body, userId) => {
fig = {
'PUT',
SON.stringify({
,
d,
```

The screenshot shows a terminal window on the left and a mobile app interface on the right. The terminal displays network traffic details, including a successful connection to a local IP address (192.168.100.37) and a REST API endpoint. The mobile app interface shows a list of document types: XXX, YYY, and ZZZ, with a button labeled 'TIPOS DE DOCUMENTOS'.

PARTE 6 y 7:

The screenshot shows a REST client interface on the left and a database query result on the right.

REST Client: The URL is `http://192.168.100.8:8080/inventarios-1.0.0/...`. The method is **POST**. The body is in **JSON** format:

```
1 {
2   "codigo": "F",
3   "descripcion": "Difunto"
4 }
```

Database Query: The query is `select * from tipo_de_documento`. The result is a table with 3 rows:

codigo_doc [PK] character	descripcion character varying (100)
1	CEDULA
2	RUC
3	Difunto

The screenshot shows two Java files: `TipodocumentoBDD.java` and `ServicioTipoDocumentos.java`.

TipodocumentoBDD.java: This class handles database operations. It includes methods for inserting, updating, and retrieving document types. The `insertarTipoDocumento` method is highlighted.

```
43 public boolean insertarTipoDocumento(TipoDocumento tipoDocumento) {
44     String sql = "insert into tipo_de_documento (codigo_doc, descripcion) values (?, ?)";
45     try (Connection conn = ConexionBDD.obtenerConexion();
46          PreparedStatement pstmt = conn.prepareStatement(sql)) {
47         pstmt.setString(1, tipoDocumento.getCodigo());
48         pstmt.setString(2, tipoDocumento.getDescripcion());
49         int rowsAffected = pstmt.executeUpdate();
50         return rowsAffected > 0;
51     } catch (SQLException e) {
52         e.printStackTrace();
53         return false;
54     } catch (KrakeDevException e1) {
55         e1.printStackTrace();
56         return false;
57     }
58 }
59 }
```

ServicioTipoDocumentos.java: This class provides the service layer for document types. It includes methods for inserting and recovering document types. The `insertarTipoDocumento` method is highlighted.

```
21 @Path("insertar")
22 @POST
23 @Consumes(MediaType.APPLICATION_JSON)
24 public Response insertarTipoDocumento(TipoDocumento tipoDocumento) {
25     boolean exito = tipoDocBDD.insertarTipoDocumento(tipoDocumento);
26     if (exito) {
27         return Response.status(Response.Status.CREATED).entity(tipoDocumento).build();
28     } else {
29         return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();
30     }
31 }
32
33 @Path("recuperar")
34 @GET
35 @Produces(MediaType.APPLICATION_JSON)
36 public Response recuperar() {
37     ArrayList<TipoDocumento> documentos = null;
38     try {
39         documentos = tipoDocBDD.recuperar();
40         return Response.ok(documentos).build();
41     } catch (KrakeDevException e) {
42         e.printStackTrace();
43         return Response.serverError().build();
44     }
45 }
46 }
```

PARTE 8:

The screenshot shows a web application interface with a form and a confirmation dialog.

Form: The form is titled "NUEVO MENSAJE" and has a text input field for "Ingrese tipo de documento". The input field has a placeholder "TIPO".

Confirmation Dialog: The dialog is titled "CONFIRMACION" and contains the text "Se ha ingresado un nuevo tipo de documento". It has an "OK" button.

Database Query: The query is `select * from tipo_de_documento`. The result is a table with 3 rows:

codigo_doc [PK] character	descripcion character varying (100)
1	CEDULA
2	RUC
3	Pasaporte