Instituto Nacional de Formación
Técnico Profesional
Infotep.gob.do

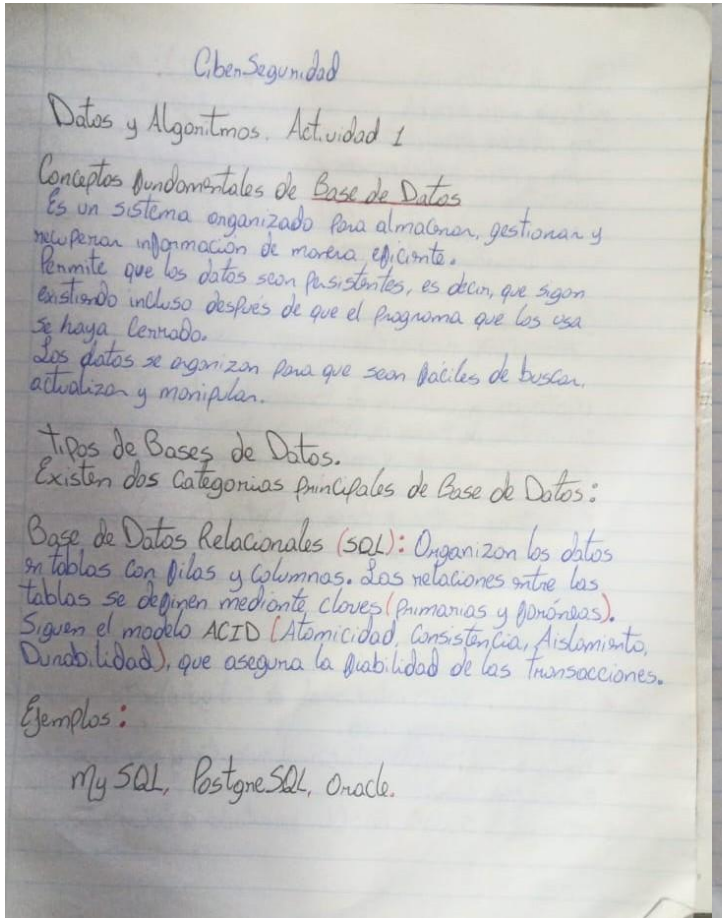
Dapiato Prosalud
A la vanguardia de la tecnologia
secomdr.com

Tecnico Ciberseguridad

Profesor: Kevin Feliz Henriquez

Nombre: Joice Pérez Paulino

Base de Datos actividad 1



CiberSeguridad

Datos y Algoritmos. Actividad 1

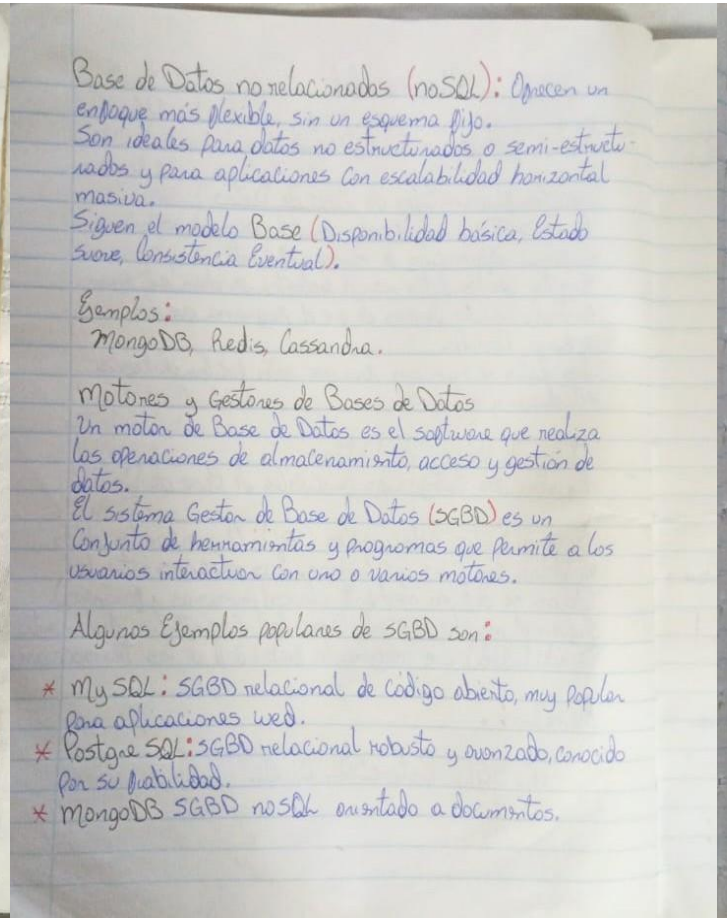
Conceptos fundamentales de Base de Datos
Es un sistema organizado para almacenar, gestionar y recuperar información de manera eficiente.
Permite que los datos sean persistentes, es decir, que sigan existiendo incluso después de que el programa que los usa se haya cerrado.
Los datos se organizan para que sean fáciles de buscar, actualizar y manipular.

Tipos de Bases de Datos.
Existen dos categorías principales de Base de Datos:

Base de Datos Relacionales (SQL): Organizan los datos en tablas con filas y columnas. Las relaciones entre las tablas se definen mediante claves (primarias y foráneas).
Siguen el modelo ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), que asegura la fiabilidad de las transacciones.

Ejemplos:

MySQL, PostgreSQL, Oracle.



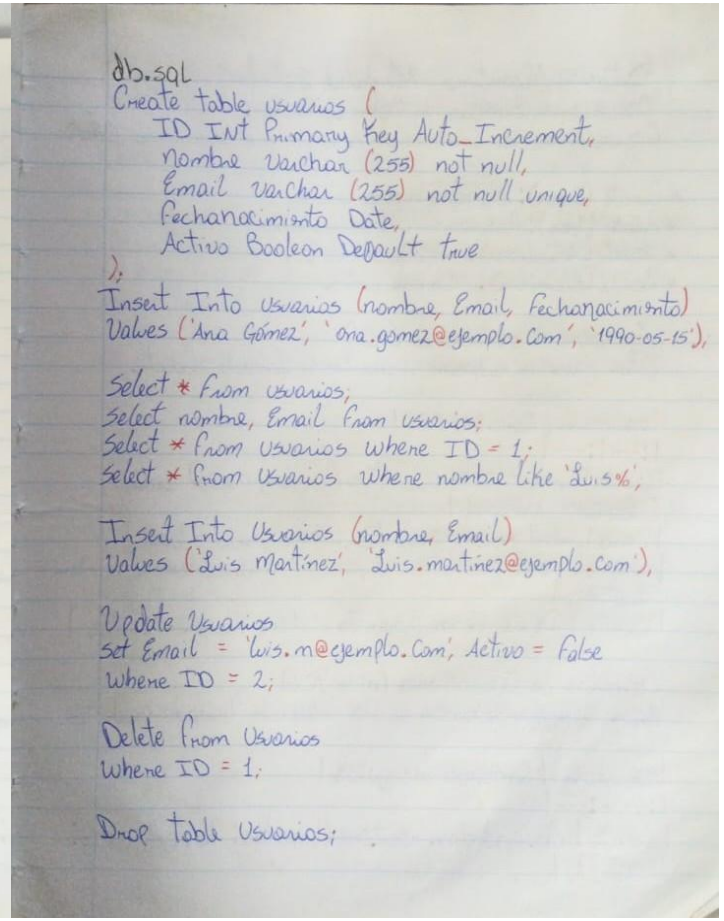
Base de Datos no relacionados (noSQL): Ofrecen un enfoque más flexible, sin un esquema fijo.
Son ideales para datos no estructurados o semi-estructurados y para aplicaciones con escalabilidad horizontal masiva.
Siguen el modelo Base (Disponibilidad básica, Estado suave, Consistencia Eventual).

Ejemplos:
MongoDB, Redis, Cassandra.

Motores y Gestores de Bases de Datos
Un motor de Base de Datos es el software que realiza las operaciones de almacenamiento, acceso y gestión de datos.
El sistema Gestor de Base de Datos (SGBD) es un conjunto de herramientas y programas que permite a los usuarios interactuar con uno o varios motores.

Algunos Ejemplos populares de SGBD son:

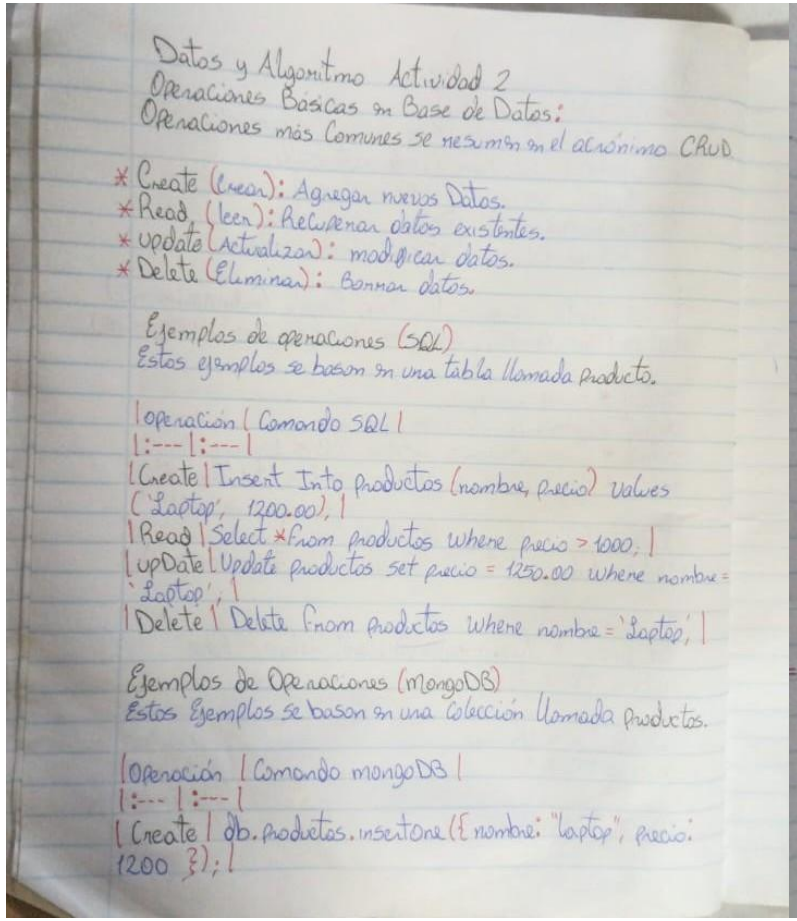
* MySQL: SGBD relacional de código abierto, muy popular para aplicaciones web.
* PostgreSQL: SGBD relacional robusto y avanzado, conocido por su fiabilidad.
* MongoDB SGBD noSQL orientado a documentos.



db.sql

```
Create table usuarios (
    ID INT Primary Key Auto_Increment,
    Nombre Varchar (255) not null,
    Email Varchar (255) not null unique,
    Fechanacimiento Date,
    Activo Boolean Default true
);
Insert Into Usuarios (nombre, Email, Fechanacimiento)
Values ('Ana Gómez', 'ana.gomez@ejemplo.com', '1990-05-15'),

Select * From usuarios;
Select nombre, Email From usuarios;
Select * From usuarios Where ID = 1;
Select * From usuarios Where nombre like 'Luis%';

Insert Into Usuarios (nombre, Email)
Values ('Luis Martínez', 'luis.martinez@ejemplo.com'),

Update Usuarios
Set Email = 'luis.m@ejemplo.com', Activo = False
Where ID = 2;

Delete From Usuarios
Where ID = 1;

Drop Table Usuarios;
```

Base de Datos actividad 2



Datos y Algoritmo Actividad 2
Operaciones Básicas en Base de Datos:
Operaciones más Comunes se resumen en el acrónimo CRUD

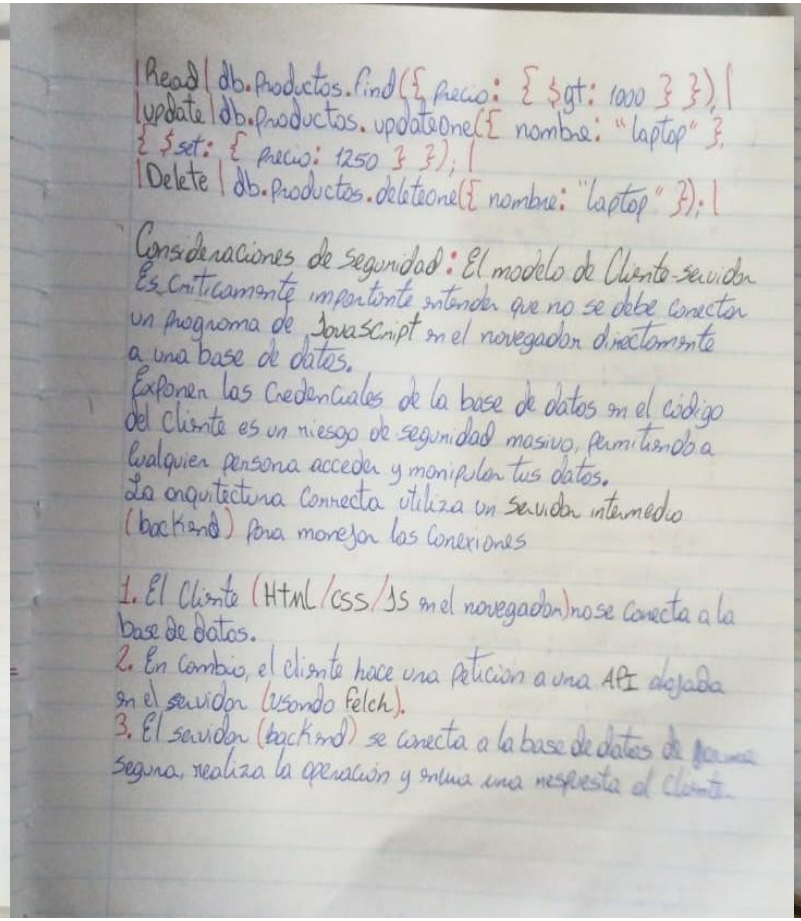
* Create (Crear): Agregar nuevos Datos.
* Read (leer): Recuperar datos existentes.
* update (Actualizar): modificar datos.
* Delete (Eliminar): Borrar datos.

Ejemplos de operaciones (SQL)
Estos ejemplos se basan en una tabla llamada Producto.

Operación	Comando SQL
Create	Insert Into productos (nombre, precio) values ('Laptop', 1200.00),
Read	Select * From productos Where precio > 1000;
upDate	Update productos set precio = 1250.00 where nombre = 'Laptop',
Delete	Delete From productos Where nombre = 'Laptop';

Ejemplos de Operaciones (MongoDB)
Estos Ejemplos se basan en una Colección llamada productos.

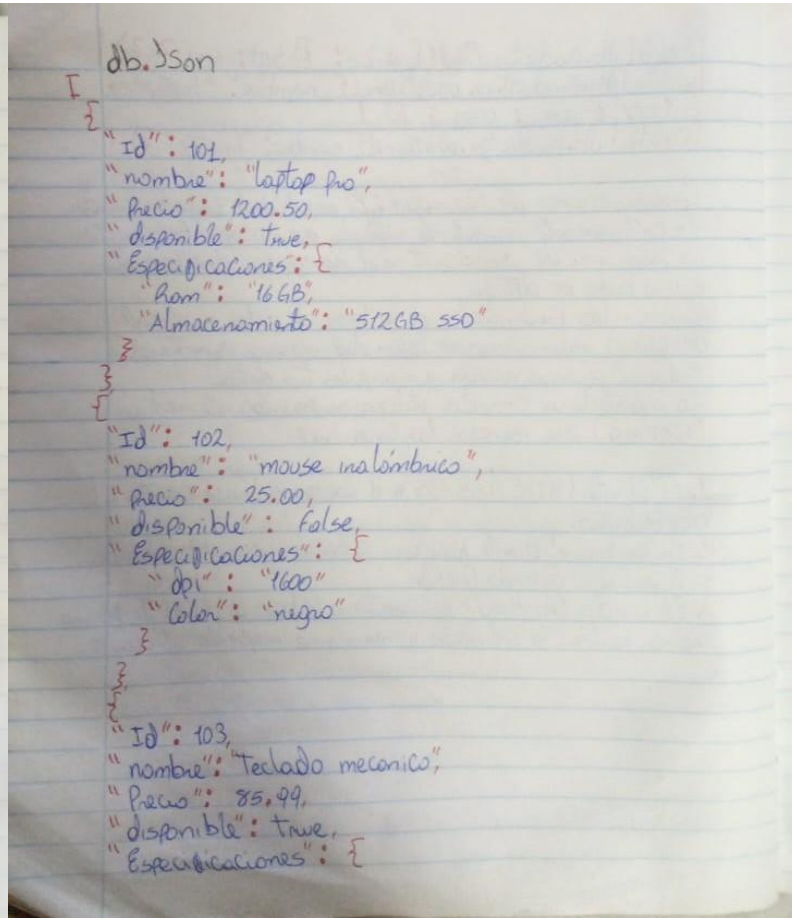
Operación	Comando mongoDB
Create	db.productos.insertOne({ nombre: "laptop", precio: 1200 });



Read	db.productos.find({ precio: { $gt: 1000 } }),
update	db.productos.updateOne({ nombre: "laptop" }, { $set: { precio: 1250 } });
Delete	db.productos.deleteOne({ nombre: "laptop" });

Consideraciones de seguridad: El modelo de Cliente-servidor
Es Criticamente importante entender que no se debe conectar un programa de JavaScript en el navegador directamente a una base de datos.
Exponen las Credenciales de la base de datos en el código del Cliente es un riesgo de seguridad masivo, permitiendo a Cualquier persona acceder y manipular tus datos.
La arquitectura Correcta utiliza un Servidor intermedio (backend) para manejar las Conexiones

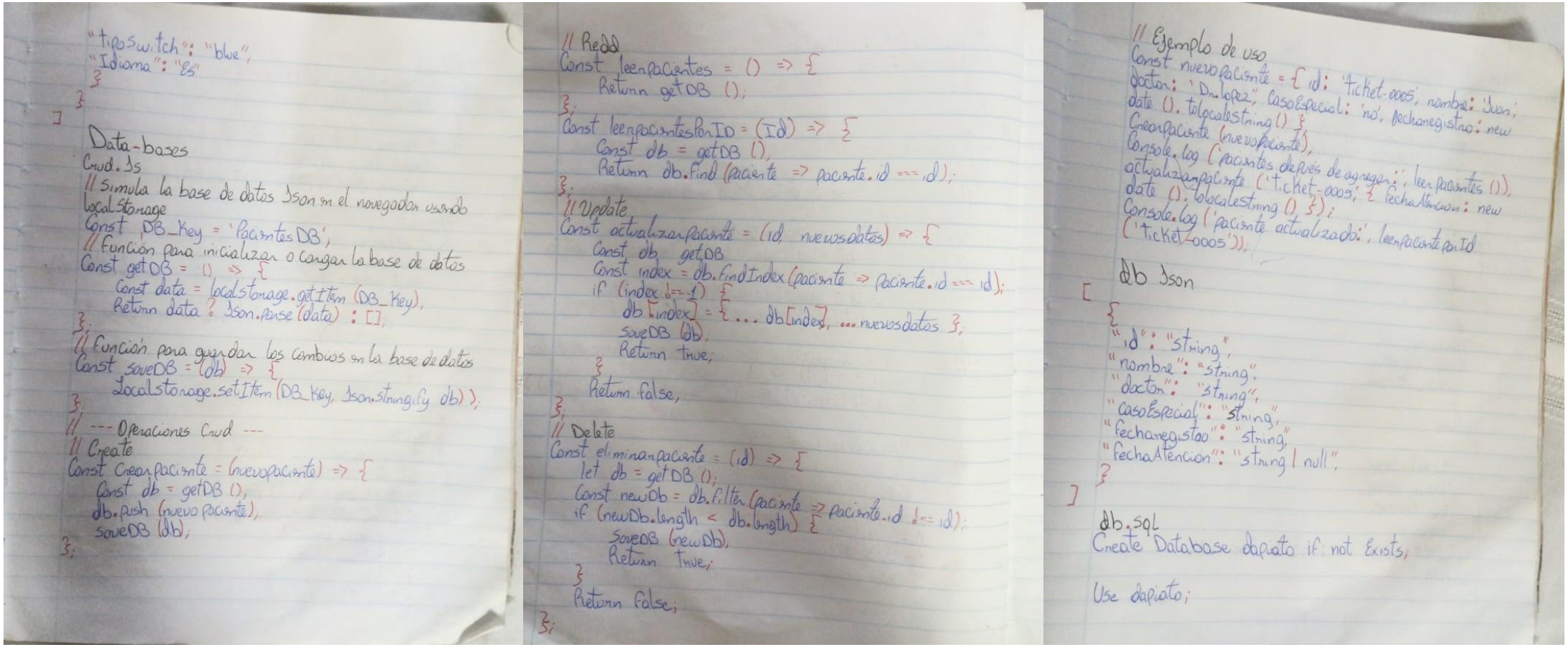
1. El Cliente (Html/css/Js en el navegador) no se Conecta a la base de datos.
2. En Cambio, el cliente hace una petición a una API alojada en el servidor (usando Fetch).
3. El servidor (backend) se conecta a la base de datos de forma segura, realiza la operación y envía una respuesta al Cliente.



db.Json

```
[
  {
    "Id": 101,
    "nombre": "laptop pro",
    "Precio": 1200.50,
    "disponible": true,
    "Especificaciones": {
      "Ram": "16GB",
      "Almacenamiento": "512GB ssd"
    }
  },
  {
    "Id": 102,
    "nombre": "mouse inalámbrico",
    "Precio": 25.00,
    "disponible": false,
    "Especificaciones": {
      "dpi": "1600"
      "Color": "negro"
    }
  },
  {
    "Id": 103,
    "nombre": "Teclado mecanico",
    "Precio": 85.99,
    "disponible": true,
    "Especificaciones": {
```

Base de Datos actividad 2

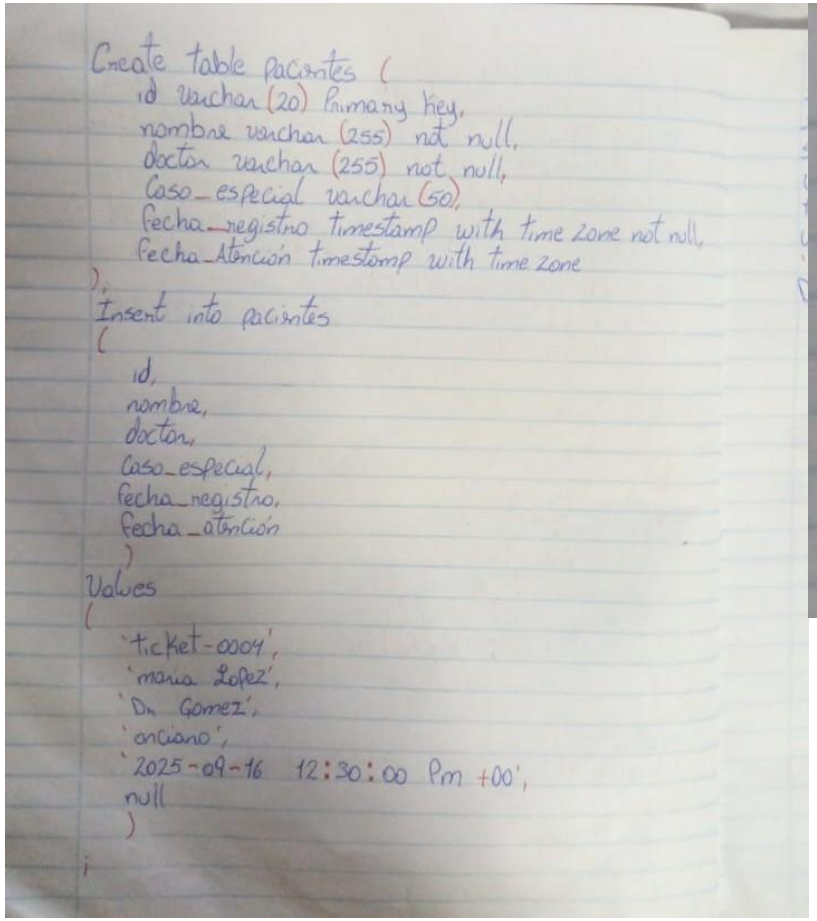


```
"tipoSwitch": "blue",
"Idioma": "Es"
}
]

Data-bases
Crud.js
// Simula la base de datos Json en el navegador usando localStorage
Const DB_Key = 'PacientesDB';
// Función para inicializar o cargar la base de datos
Const getDB = () => {
    Const data = localStorage.getItem (DB_Key),
    Return data ? Json.Parse (data) : [],
};
// Función para guardar los cambios en la base de datos
Const saveDB = (db) => {
    localStorage.setItem (DB_Key, Json.stringify db) );
};
// --- Operaciones Crud ---
// Create
Const CrearPaciente = (nuevoPaciente) => {
    Const db = getDB (),
    db.push (nuevo paciente),
    saveDB (db),
};
```

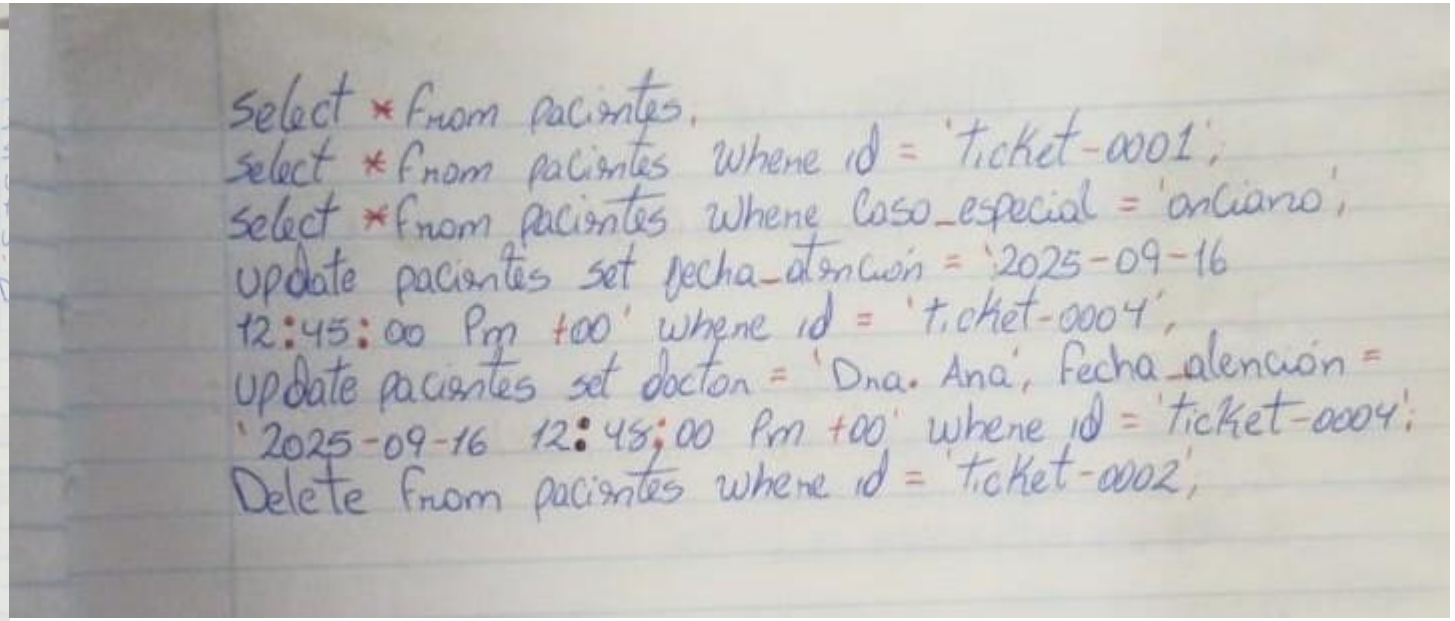
```
// Redd
Const leerPacientes = () => {
    Return getDB (),
};
Const leerPacientesPorID = (Id) => {
    Const db = getDB (),
    Return db.Find (paciente => paciente.id === id);
};
// Update
Const actualizarPaciente = (id, nuevosDatos) => {
    Const db  getDB
    Const index = db.FindIndex (paciente => paciente.id === id);
    if (index !== -1) {
        db [index] = { ... db[index], ... nuevosDatos };
        saveDB (db),
        Return true;
    }
    Return false,
};
// Delete
Const eliminarPaciente = (id) => {
    let db = getDB ();
    Const newDb = db.filter (paciente => paciente.id !== id);
    if (newDb.length < db.length) {
        saveDB (newDb),
        Return True;
    }
    Return false;
};
```

```
// Ejemplo de uso
Const nuevoPaciente = { id: 'ticket-0005', nombre: 'Juan',
doctor: 'Dr.Lopez', CasoEspecial: 'no', fecharegistro: new
date (). tolocalestring () }
CrearPaciente (nuevopaciente),
Console.log ('pacientes despues de agregar:', leerPacientes ()),
actualizarPaciente ('ticket-0005', { fechaAtencion: new
date (). tolocalestring () });
Console.log ('paciente actualizado:', leerPacientePorId
('ticket-0005')),

db Json
[
  {
    "id": "string",
    "nombre": "string",
    "doctor": "string",
    "CasoEspecial": "string",
    "fecharegistro": "string",
    "fechaAtencion": "string | null",
  }
]

db.sql
Create Database dapiato if not Exists,

Use dapiato;
```

Base de Datos actividad 2



```
Create table pacientes (
    id varchar(20) Primary key,
    nombre varchar (255) not null,
    doctor varchar (255) not null,
    Caso_especial varchar(50),
    fecha_registro timestamp with time zone not null,
    Fecha_Atención timestamp with time zone
);
Insert into pacientes
(
    id,
    nombre,
    doctor,
    Caso_especial,
    fecha_registro,
    fecha_atención
    )
Values
(
    'ticket-0004',
    'maria Lopez',
    'Dr Gomez',
    'anciano',
    '2025-09-16  12:30:00 Pm +00',
    null
    )
;
```



```
Select * From pacientes.
Select * From pacientes Where id = 'ticket-0001';
Select * From pacientes Where Caso_especial = 'anciano';
Update pacientes set fecha_atencion = '2025-09-16
12:45:00 Pm +00' where id = 'ticket-0004',
Update pacientes set doctor = 'Dra. Ana', Fecha_atencion =
'2025-09-16 12:45:00 Pm +00' where id = 'ticket-0004';
Delete From pacientes where id = 'ticket-0002',
```