



ITP

**IMPLEMENTACIÓN DE BASE DE DATOS PARA LA GESTIÓN DE
UN RESTAURANTE EN MONGODB**

ENTREGADO POR:

JOHAN FERNANDO BURBANO CALPA

ENTREGADO A:

BRAYAN ARCOS

INSTITUTO TECNOLÓGICO DEL PUTUMAYO

MOCOA – PUTUMAYO

2024



Contenido

Contenido 2

Resumen Ejecutivo 3

Introducción 3

Contexto y Motivación.....3

Alcance del Informe3

Objetivos.....3

Metodología 4

Herramientas Utilizadas.....4

Procedimientos.....4

Realización del Taller:.....4

Desarrollo del Informe..... 5

Esquema de la Base de Datos 5

Modelo de Datos 7

- Relaciones7

- Cardinalidad7

Métodos de Captura..... 8

Consultas 9

Análisis y Discusión10

Interpretación de Resultados 10

Conclusiones11_____

Referencias 11

Resumen Ejecutivo

Este informe presenta el diseño, desarrollo e implementación de una base de datos para un restaurante, utilizando MongoDB como sistema de gestión de bases de datos. Se destaca la creación de una base de datos con colecciones que incluyen restaurantes, administradores, empleados, clientes, mesas, reservas, platos, pedidos y facturas. El diseño se centró en optimizar las relaciones entre estas colecciones para reflejar de manera eficiente las interacciones dentro del restaurante. Las principales decisiones de diseño incluyen la implementación de relaciones uno a uno, uno a muchos y muchos a muchos. Los resultados obtenidos muestran que la base de datos es capaz de manejar consultas complejas, actualizar datos y realizar inserciones de manera efectiva, cumpliendo con los objetivos de optimización y gestión de la información del restaurante.

Introducción

Contexto y Motivación

La base de datos fue diseñada con el objetivo de mejorar la gestión de un restaurante, permitiendo una administración eficiente de las relaciones entre empleados, clientes, mesas, reservas, pedidos y facturas. La motivación detrás de este proyecto es facilitar el análisis, la consulta y la manipulación de la información crítica del restaurante, como el seguimiento de las reservas, la gestión de empleados y el control de pedidos. Este sistema permitirá una mejora significativa en la eficiencia operativa del restaurante.

Alcance del Informe

El informe abarca el diseño de la base de datos utilizando MongoDB, el desarrollo de relaciones entre colecciones, la ejecución de consultas para la inserción y actualización de datos, así como la validación de dichas relaciones. Se incluyen ejemplos de relaciones uno a uno, uno a muchos y muchos a muchos.

Objetivos

- Diseñar una base de datos clara, escalable y eficiente.
- Implementar relaciones entre colecciones usando las mejores prácticas.
- Ejecutar consultas para la manipulación de datos (insert, update, upsert).
- Facilitar el análisis y manejo de información crítica del restaurante mediante consultas optimizadas.

Metodología

Herramientas Utilizadas

- MongoDB: Sistema NoSQL utilizado para la creación y gestión de la base de datos.
- Studio 3T: Herramienta gráfica empleada para la creación, visualización y análisis de las colecciones y las relaciones entre ellas.

Procedimientos

Creación de la Base de Datos: Se diseñó la estructura de la base de datos en MongoDB, empleando Studio 3T para la creación y manipulación visual de las colecciones.

- Colecciones principales restaurantes, administradores, empleados, clientes, mesas, reservas, platos, pedidos y facturas
- Relaciones: Se implementaron relaciones uno a uno, uno a muchos y muchos a muchos entre las colecciones, según la naturaleza de los datos.

Realización del Taller:

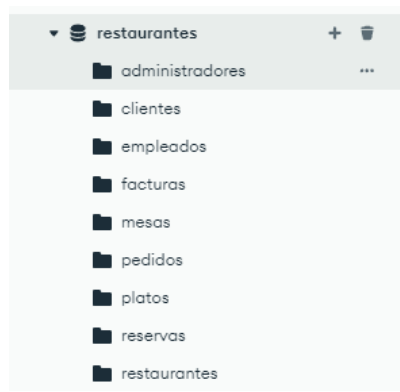
Se utilizaron archivos mediante consola para generar las colecciones, y se normalizó el modelo de datos para reducir redundancias y mejorar el rendimiento de las consultas.

Se llevaron a cabo operaciones de inserción y actualización de datos utilizando las funciones `updateOne()`, `setOnInsert` y `each()` para garantizar la integridad y consistencia de los datos.

Desarrollo del Informe

Esquema de la Base de Datos

El esquema de la base de datos se diseñó para representar todas las entidades principales del restaurante y sus relaciones:



The screenshot shows the MongoDB Compass interface for a database named 'restaurantes'. A sidebar on the left lists the collections: administradores, clientes, empleados, facturas, mesas, pedidos, platos, reservas, and restaurantes. The main area displays a table with details for each collection.

| Collection Name | Documents | Avg. document size | Indexes | Total index size |
|------------------------|-----------|--------------------|---------|------------------|
| administradores | 3 | 76.00 B | 1 | 36.00 KB |
| clientes | 4 | 77.00 B | 1 | 36.00 KB |
| empleados | 15 | 94.00 B | 1 | 20.48 KB |
| facturas | 2 | 107.00 B | 1 | 20.48 KB |
| mesas | 6 | 76.00 B | 1 | 20.48 KB |
| pedidos | 4 | 170.00 B | 1 | 36.00 KB |
| platos | | | | |

- Empleados: Cada empleado está asociado a un restaurante (relación 1 a muchos).

```

_id: ObjectId('6710854ed64ce1bfb49aee78')
nombre: "milanesa"
direccion: "Avenida Colombia"
telefono: "555-1234"
administradorId: ObjectId('671083d23963da2315e2a1d1')
empleados: Array (6)
  0: Object
  1: Object
  2: Object
  3: Object
  4: Object
  5: Object

```

- Mesas y Reservas: Las reservas están vinculadas tanto a las mesas como a los clientes (relación muchos a muchos).

```

_id: ObjectId('67108635340dfddc1471c252')
clienteId: ObjectId('67108611340dfddc1471c251')
mesaId: ObjectId('671085b1340dfddc1471c248')
fecha: 2024-10-16T00:00:00.000+00:00
estado: "confirmada"

```

- Pedidos y Facturas: Los pedidos realizados por los clientes generan facturas (relación uno a muchos).

```

_id: ObjectId('67108db9875da03af476eb68')
clienteId: ObjectId('671085dc340dfddc1471c24e')
pedidos: Array (1)
  0: ObjectId('67108d71875da03af476eb66')
total: 26
fechaEmision: 2024-10-10T00:00:00.000+00:00

```

- Administrador y Restaurantes: cada administrador tiene un restaurante asignado y cada restaurante tiene un único administrador. (relación de uno a uno)

```

_id: ObjectId('6710854ed64ce1bfb49aee78')
nombre: "milanesa"
direccion: "Avenida Colombia"
telefono: "555-1234"
administradorId: ObjectId('671083d23963da2315e2a1d1')
empleados: Array (6)

```

Modelo de Datos

El modelo fue normalizado para minimizar redundancias y asegurar una estructura eficiente:

- **Relaciones:** Las relaciones entre colecciones fueron representadas mediante referencias (ObjectId), como en el caso de los empleados y su restaurante, o embebiendo documentos en casos donde se necesitaba información relacionada directamente dentro de la colección (por ejemplo, detalles del pedido dentro de la factura).

```
_id: ObjectId('67108db9875da03af476eb68')
clienteId: ObjectId('671085dc340dfddc1471c24e')
pedidos: Array (1)
  0: ObjectId('67108d71875da03af476eb66')
total: 26
fechaEmision: 2024-10-10T00:00:00.000+00:00
```

- **Cardinalidad:**
 - Uno a uno: Administrador y restaurantes.
 - Uno a muchos: Empleado y restaurante.
 - Muchos a muchos: Mesas y reservas, gestionado a través de colecciones intermediarias.

Métodos de Captura

- Se crearon las diferentes colección e inserción de datos por medio de la consola de MongoDB.

```
restaurantes> // Insertar administrador
db.administradores.insertOne({
  nombre: "Juan Pérez",
  email: "juan.perez@example.com"
});

// Insertar restaurante con referencia al administrador
db.restaurantes.insertOne({
  nombre: "Restaurante El Sabor",
  direccion: "Calle 123, Ciudad",
  telefono: "555-1234",
  administradorId: ObjectId("67108753d64ce1bfb49aee7a")
});
```

```
restaurantes> db.empleados.insertMany([
  { nombre: "Pedro Gómez", cargo: "Mesero", restauranteId: ObjectId("6710854ed64ce1bfb49aee78") },
  { nombre: "Marta López", cargo: "Cocinero", restauranteId: ObjectId("6710854ed64ce1bfb49aee78") },
  { nombre: "Juan Pérez", cargo: "Cajero", restauranteId: ObjectId("6710854ed64ce1bfb49aee78") },
  { nombre: "Luisa Torres", cargo: "Mesero", restauranteId: ObjectId("6710854ed64ce1bfb49aee78") },
  { nombre: "Carlos Vega", cargo: "Cocinero", restauranteId: ObjectId("6710854ed64ce1bfb49aee78") }
]);
```

- Posteriormente, se aplicaron relaciones mediante el uso de updateOne() con las opciones upsert y setOnInsert para insertar o actualizar datos cuando fuese necesario.

Consultas

- Inserciones: Se insertaron empleados, clientes, reservas y pedidos utilizando insertMany().

```
> db.administradores.insertMany([
  {
    nombre: "Luis Rodríguez",
    email: "luis.rodriguez@example.com",
    telefono: "555-6789"
  },
  {
    nombre: "Ana María Pérez",
    email: "ana.perez@example.com",
    telefono: "555-9876"
  },
  {
    nombre: "Carlos Sánchez",
    email: "carlos.sanchez@example.com",
    telefono: "555-4567"
  }
]);
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('671091dbffcaf5550d938ee4'),
    '1': ObjectId('671091dbffcaf5550d938ee5'),
    '2': ObjectId('671091dbffcaf5550d938ee6')
  }
}
```

- Actualizaciones: Se empleó updateOne() y setOnInsert para actualizar datos de clientes y reservas cuando ya existían entradas.

```
> db.restaurantes.updateOne(
  { nombre: "Restaurante El Sabor" },
  {
    $setOnInsert: { // Solo se establece si se inserta un nuevo documento
      direccion: "Calle 123, Ciudad",
      telefono: "555-1234"
    },
    $addToSet: { // Agrega un nuevo empleado solo si no existe
      empleados: {
        nombre: "PEDRO",
        cargo: "Mesero"
      }
    }
  },
  { upsert: true } // Si no se encuentra el documento, se crea uno nuevo
);
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

- Búsquedas: Se realizaron consultas para obtener la información de pedidos por cliente, reservas por mesa y facturación por restaurante.

```
db.pedidos.find(
  { clienteId: ObjectId("671085dc340dfddc1471c24e") }
).toArray();
[
  {
    _id: ObjectId('671086f2340dfddc1471c256'),
    clienteId: ObjectId('671085dc340dfddc1471c24e'),
    fecha: 2024-10-17T03:39:30.135Z,
    platos: [
      ObjectId('671086d5340dfddc1471c254'),
      ObjectId('671086d5340dfddc1471c255')
    ]
  },
  {
    _id: ObjectId('67108d71875da03af476eb66'),
    clienteId: ObjectId('671085dc340dfddc1471c24e'),
    restauranteId: ObjectId('6710854ed64ce1bfb49aee78'),
    platos: [ [Object], [Object] ],
    total: 26,
    fecha: 2024-10-10T00:00:00.000Z
  }
]
```

Análisis y Discusión

Interpretación de Resultados

- El análisis de las consultas ejecutadas y las relaciones definidas demuestra que la base de datos cumple con los objetivos propuestos. La estructura diseñada permite:
- Un manejo eficiente de las reservas y la facturación, al optimizar las relaciones entre las colecciones involucradas.
- El control y actualización de datos de manera coherente utilizando las funciones `updateOne()` con coincidencias puntuales, lo que mejora la integridad de los datos.
- Las consultas son rápidas y eficientes, gracias a la correcta definición de relaciones y la normalización de los datos.

Conclusiones

- La base de datos diseñada cubre todas las áreas críticas del restaurante, permitiendo una gestión fluida y organizada de los empleados, clientes, mesas, reservas, pedidos y facturas.
- MongoDB ha demostrado ser una solución adecuada para manejar relaciones complejas entre colecciones en un contexto de negocio como el de un restaurante.
- Las funciones de actualización y manejo de datos, como `updateOne()`, `setOnInsert`, y `each()`, permiten mantener la consistencia de los datos en todo momento.

Referencias

- MongoDB Documentación. <https://docs.mongodb.com>
- Studio 3T User Guide. <https://studio3t.com/knowledge-base/>