REPORTE DE
PROYECTO

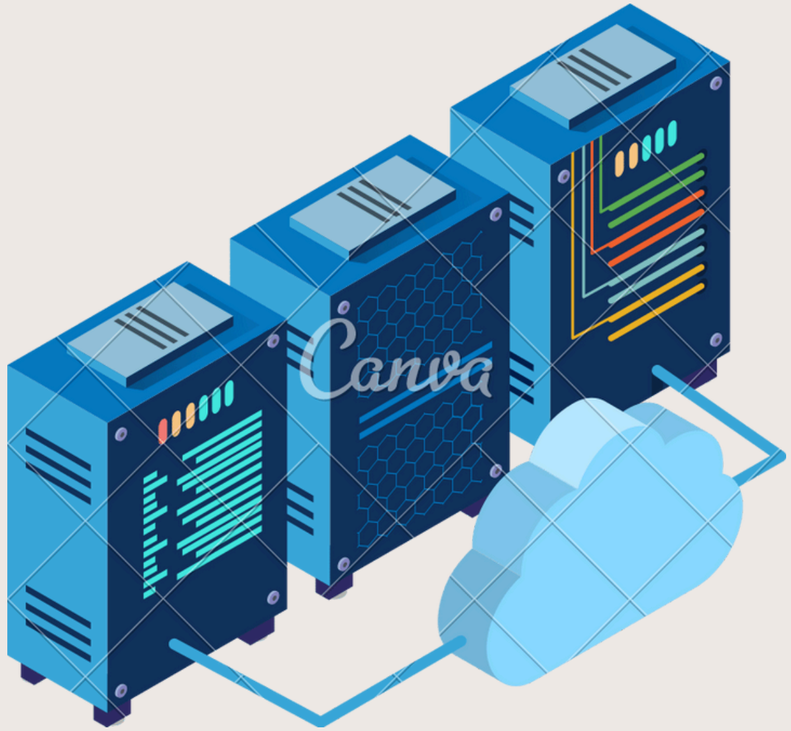
Realizado por:
Israel Campos Vázquez
Luis Erick Esperilla Mendoza
José Eduardo Valles Aguilera
Axel Damián Ortiz Simón

TABLA DE **CONTENIDO**

INTRODUCCIÓN
AL PROYECTO

Proyecto de Base de Datos Distribuidas para una agencia de viajes usando MongoDB, elegido por su escalabilidad horizontal (sharding) y flexibilidad como NoSQL.

La base gestionará reservas, clientes, empleados, reseñas, paquetes, sucursales, promociones, destinos y pagos mediante colecciones (tablas) con documentos (registros JSON) para almacenar datos de forma eficiente, ideal para múltiples sucursales.



RESEÑA

```
Resenia {
  _id: ObjectId,
  comentario: String,
  calificacion: Float,
  fecha: Date,
  reserva_id: String (REF Reserva),
  sucursal_id: Integer (REF Sucursal)
}
```

RESERVA

```
Reserva {
  _id: ObjectId,
  fecha_salida: Date,
  fecha_regreso: Date,
  estado: String,
  detalles: String,
  codigo_descuento: String,
  cliente_id: String (REF Cliente),
  destino_id: String (REF Destino),
  sucursal_id: Integer (REF Sucursal)
}
```

PAGO

```
Pago {
  _id: ObjectId,
  fecha_pago: Date,
  metodo_pago: String,
  estado: String,
  reserva_id: String (REF Reserva),
  sucursal_id: Integer (REF Sucursal)
}
```

CLIENTE

```
Cliente {
  _id: ObjectId,
  nombre: String,
  correo: String,
  telefono: String,
  direccion: Boolean,
  fecha_nacimiento: Date,
  sucursal_id: Integer (REF Sucursal)
}
```

SUCURSAL

```
Sucursal {
  _id: ObjectId,
  sucursal_id: Integer,
  nombre: String,
  direccion: String,
  telefono: String
}
```

PROMOCION

```
Promocion {
  _id: ObjectId,
  descripcion: String,
  codigo: String,
  descuento: Integer,
  fecha_inicio: Date,
  fecha_fin: Date,
  sucursal_id: Integer (REF Sucursal)
}
```

PAQUETE

```
Paquete {
  _id: ObjectId,
  nombre: String,
  precio: Float,
  servicios: [String],
  duracion: Integer,
  destinos: [String] (REF Destino)
}
```

EMPLEADO

```
Empleado {
  _id: ObjectId,
  nombre: String,
  correo: String,
  telefono: String,
  activo: Boolean,
  cargo: String,
  fecha_ingreso: Date,
  sucursal_id: Integer (REF Sucursal)
}
```

DESTINO

```
Destino {
  _id: ObjectId,
  nombre: String,
  pais: String,
  descripcion: String,
  precioBase: Float,
  temporadaAlta: String,
  cupoMaximo: Integer,
  promocion_id: String (REF Promocion),
  sucursal_id: Integer (REF Sucursal)
}
```

LOGTRANSACCION

```
{
"_id": "<ObjectId10>",
"idSucursal": "<ObjectId2>",
"tipo": "Pago",
"detalle": "Pago realizado por reserva ID <ObjectId5>",
"fecha": "2025-04-01",
"usuarioResponsable": "María López"
}
```

OBJETIVOS

01

Objetivo 01

Analizar el modelo de negocio de la agencia de viajes para definir la estructura del sharding.

02

Objetivo 02

Diseñar el esquema en MongoDB, fragmentando datos por ID de sucursales.

03

Objetivo 03

Implementar sharding en modo prueba (config servers, shards y mongos).

ALCANCES

1. Distribución eficiente de datos: Uso de sharding en MongoDB para mejorar disponibilidad y escalabilidad.

2. Escalabilidad horizontal: Capacidad de agregar nuevas sucursales sin afectar el sistema.

3. Tolerancia a fallos: Configuración de Replica Sets para respaldo automático en caso de fallos.

LIMITACIONES

Complejidad técnica

Mayor dificultad en implementación vs. bases de datos centralizadas.

Costos de infraestructura

Requiere más hardware y conectividad en red.

Dependencia de shards

Si un shard falla, puede afectar el acceso a parte de los datos.

REPLICA SETS

Grupo de mínimo 3 procesos mongod que replican los mismos datos para garantizar alta disponibilidad y tolerancia a fallos.

- Nodo primario: Único que acepta escrituras (con write concern "majority") y registra cambios en su oplog.
- Nodos secundarios: Replican el oplog del primario para mantener datos consistentes. Si el primario falla, un secundario asume su rol.

SHARDING

Técnica para distribuir datos en múltiples servidores (escalado horizontal), ideal para grandes volúmenes de datos y alto rendimiento.

- Ventajas:
  - Mejor desempeño en consultas al dividir la carga.
  - Escalabilidad flexible (añadir más servidores según demanda).
- Alternativa: Escalado vertical (mejorar hardware del servidor), pero con límites físicos.

COMPONENTES CLAVE

Shards (Fragmentos):

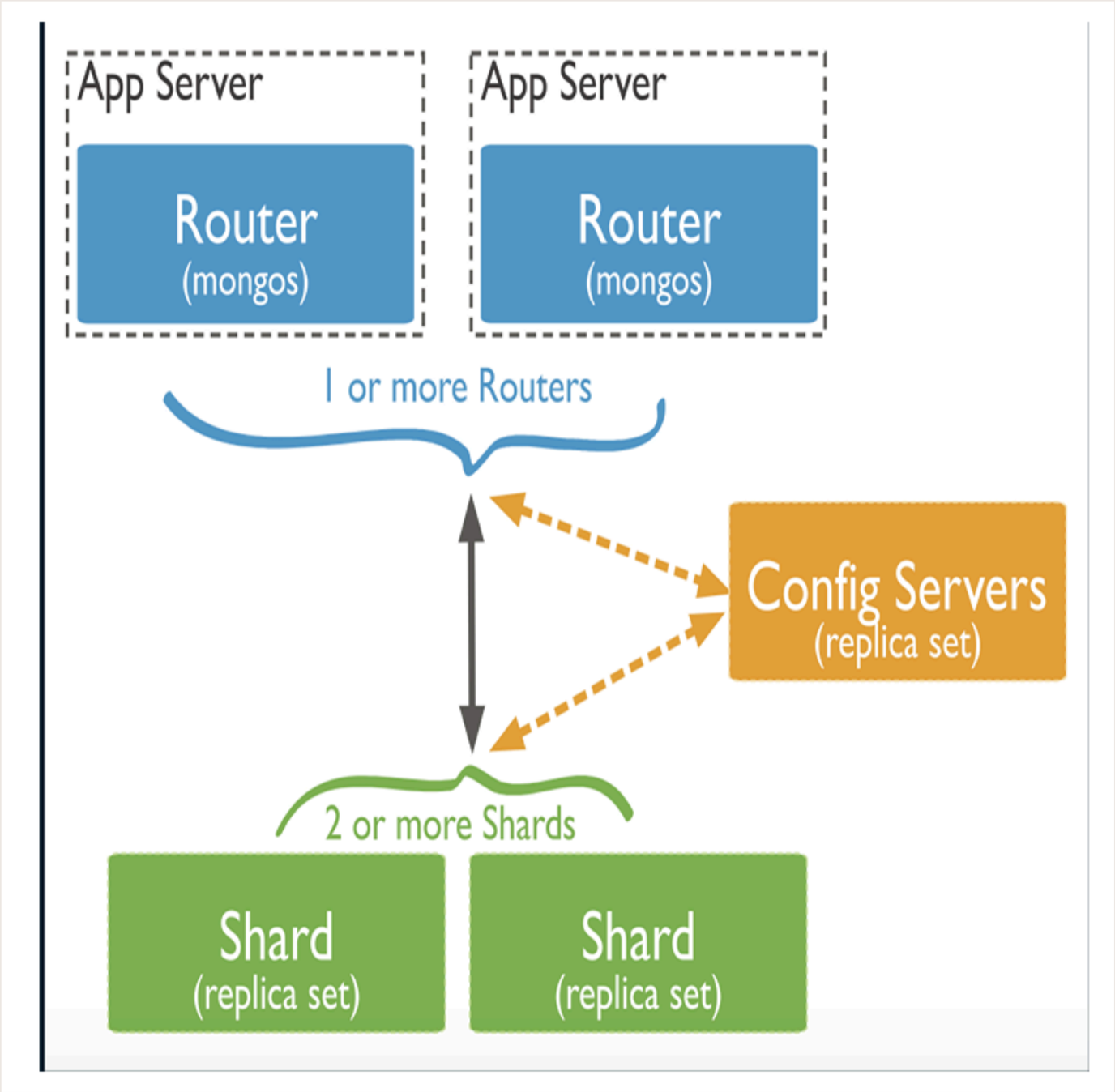
- Almacenan subconjuntos de datos fragmentados.
- Cada shard debe ser un Replica Set (para alta disponibilidad).

COMPONENTES CLAVE

Router (mongos):

Enruta consultas entre aplicaciones cliente y el sharded cluster.

- Interfaz única para interactuar con el cluster (tanto colecciones fragmentadas como no fragmentadas).
- Los clientes nunca se conectan directamente a un shard.


App Server
Router
(mongos)
App Server
Router
(mongos)
1 or more Routers
Config Servers
(replica set)
2 or more Shards
Shard
(replica set)
Shard
(replica set)

COMPONENTES CLAVE

Config Servers:

- Almacenan metadatos y configuración del cluster.
- Deben implementarse como un Replica Set (para evitar pérdida de configuración).

SHARD KEY

Determina cómo se distribuyen los documentos entre shards (ej., por sucursal_id).

Requisitos:

- Para colecciones pobladas: Debe existir un índice previo que incluya la shard key.
- Para colecciones vacías: MongoDB crea automáticamente el índice necesario.
- Documentos sin shard key: Se tratan con valor null en la distribución, pero no en consultas.

PARTICIÓN DE DATOS

- Los datos se dividen en chunks (trozos) basados en la shard key.
- Cada shard maneja un rango específico de valores (límite inferior inclusivo, superior exclusivo).

MUCHAS GRACIAS