Entrega 3

DISEÑO Y CONSTRUCCIÓN DE LA APLICACIÓN

1. Análisis y modelo conceptual

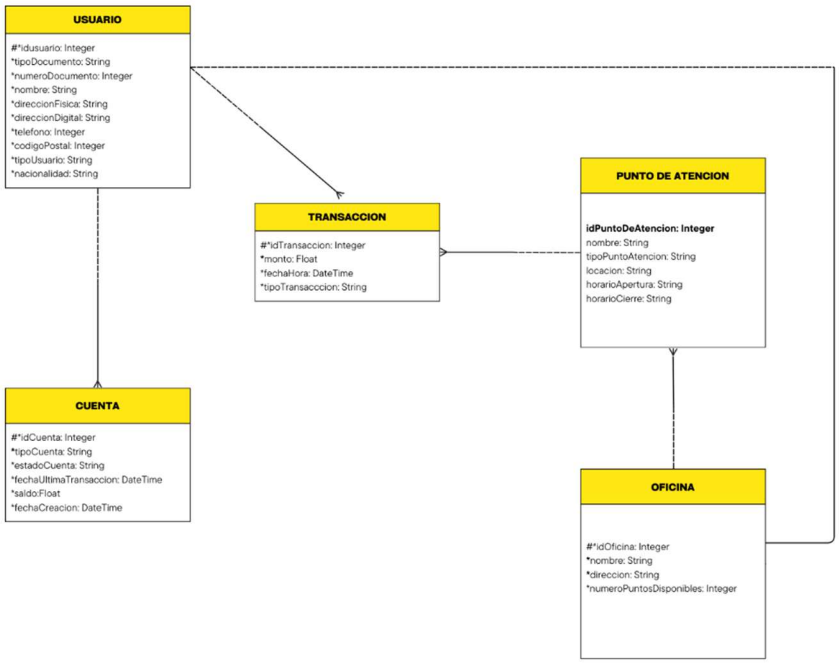


Imagen 1. Modelo relacional Banco de los Andes

2. Diseño de la base de datos
    a. Análisis de la carga de trabajo.

Entidad	**Atributos**	**Entidades (Registros)**
Usuario	Id usuario, tipo documento, numero documento, Nombre, dirección física, dirección digital, dirección, código postal, tipo usuario, nacionalidad	**1.500.000**
Cuenta	Id cuenta, tipo cuenta, estado cuenta, fecha ultima transacción, saldo, fecha creación	**2.500.000**
Transacción	Id transacción, monto, fecha hora, tipo transacción	**8.000.000**
Puntos de atención	Id punto de atención, nombre, tipo punto atención, dirección, horario apertura, horario cierre	**1500**
Oficina	Id oficina, nombre, dirección, numero puntos disponibles	**300**

Tabla 1. Identificación de entidades con sus atributos y cuantificación aproximada

Entidad	Operación	Información necesaria	Tipo
Usuario	Crear usuario	Atributos de usuario	Escritura
Oficina, Usuario	Crear Oficina	Atributos oficina + Id Usuario (gerente)	Escritura
Punto Atención, Oficina	Crear/Borrar	Atributos Punto Atención + Id punto de atención	Escritura
Cuenta, Usuario	Crear	Atributos Cuenta + Id Usuario	Escritura
Cuenta	Modificar Estado Cuenta	Id Cuenta	Escritura
Transacción, Cuenta, Usuario	Registrar operación sobre cuenta	Id Cuenta + Id Usuario+ Id Punto Atención+ Atributos transacción	Escritura
Cuenta	Consultar todas las cuentas	Atributos Cuentas	Lectura
Transacción, Cuenta	Extracto Bancario para cuenta	Id Cuenta + Saldo inicio mes + transacciones + saldo final mes	Lectura

Tabla 2. Operaciones de lectura y escritura para cada entidad

Entidad	Operación	Información necesaria	Tipo	Rate
Usuario	Crear usuario o modificar	Atributos de usuario	Escritura	200 veces por día
Usuario	Consultar usuario	Id Usuario	Lectura	500 veces por día
Oficina, Usuario	Crear Oficina o modificar	Atributos oficina + Id Usuario (gerente)	Escritura	1 vez por mes
Oficina, Usuario	Consultar Oficina	Id Oficina	Lectura	1 vez por semana
Punto Atención, Oficina	Crear, modificar o Borrar	Atributos Punto Atención + Id punto de atención	Escritura	1 vez por mes
Punto Atención	Consulta	Id punto de atención	Lectura	1 vez por semana
Cuenta, Usuario	Crear	Atributos Cuenta + Id Usuario	Escritura	500 veces por día
Cuenta	Modificar Estado Cuenta	Id Cuenta	Escritura	500 veces por día
Transacción, Cuenta, Usuario	Registrar operación sobre cuenta	Id Cuenta + Id Usuario+ Id Punto Atención+ Atributos transacción	Escritura	20,000 veces por día
Transacción, Cuenta, Usuario	Consulta operación sobre cuenta	Id Cuenta + Id usuario	Lectura	5,000 veces por dia
Cuenta	Consultar cuentas	Id Cuentas	Lectura	5,000 veces por día
Transacción, Cuenta	Extracto Bancario para cuenta	Id Cuenta + Saldo inicio mes + transacciones + saldo final mes	Lectura	1 vez por mes

Tabla 3. Cuantificación operaciones de lectura y escritura para cada entidad

b. Descripción de las entidades de datos y las relaciones NoSQL entre ellas que corresponden al modelo Er propuesto.

Entidades:

- Usuario: Representa la información básica de un empleado y un cliente, es decir, en el caso del empleado puede ser un gerente general, un gerente de oficina o un cajero, y en el caso del cliente puede ser natural o jurídico.
- Cuenta: Es un producto del banco que pertenece a un cliente, ya sea natural o jurídico. Las cuentas pueden ser del tipo ahorros, corriente o AFC. Además, cuentan con un estado activo, cerrado o desactivado.
- Transacción: Es realizada por un cliente en un punto de atención y afecta un producto, específicamente, a una cuenta. Los tipos de transacción son abrir, cerrar, consignar, retirar, transferir y desactivar.
- Punto De Atención: Hace referencia a las entidades por medio de las cuales se crean las entidades anteriores. Se dividen en personalizada, cajeros automáticos y digitales. Los puntos de atención personalizada de tipo cajero y los cajeros automáticos están vinculados de forma obligatoria a una oficina. Los de autoservicio digital como el portal Web y la aplicación móvil no están asociados a una oficina.
- Oficina: Punto físico que agrupa puntos de atención personalizados.

Relaciones entre entidades y cardinalidad:

- Usuario y cuenta: un usuario puede tener una o varias cuentas. Y, una cuenta debe tener un único usuario asociado. Por tanto, la cardinalidad es de uno a muchos.
- Usuario y transacción: Un usuario puede realizar muchas transacciones, mientras que, una transacción debe tener un usuario asociado. Por tanto, la cardinalidad es de uno a muchos.
- Transacción y punto de atención: Una transacción puede tener un punto de atención asociado, y, un punto de atención puede tener muchas transacciones. Entonces, la cardinalidad es de uno a muchos.
- Punto de atención y oficina: Un punto de atención puede o no estar asociado a una oficina. Mientras que, una oficina puede o no tener uno o varios puntos de atención asociados a ella. En conclusión, la cardinalidad es de uno a muchos.
- Oficina y usuario: La creación de una oficina tiene como requisito la afiliación de un usuario tipo gerente, el cual ya debe existir en la base de datos. Esta relación es de uno a uno.

Análisis de selección del esquema de asociación:

- Usuario y cuenta:

Guideline	**Question**	**Respuesta**
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Reference
Go Together	Do the pieces of information have a has-a, contains, or similar relationship?	Embed
Query Atomicity	Does the application query the pieces of information together?	Embed
Update Complexity	Are the pieces of information updated together?	Reference

Archival	Should the pieces of information be archived at the same time?	Reference
Cardinality	Is there a high cardinality in the child side of the relationship?	Reference
Data Duplication	Would data duplication be too complicated to manage and undesired?	Embed
Document size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	Reference
Document growth	Would the embedded piece grow without bound?	Reference
Workload	Are the pieces of information written at different times in a Escritura-heavy workload?	Reference
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	Reference

Tabla 4. Reference vs Embed en la relación usuario y cuenta

Análisis: Al responder a las preguntas, se observa que, debido a las particularidades de la búsqueda, actualización y simplicidad, la mejor solución es utilizar una referencia en esta relación. Por lo tanto, la clase Usuario tendría una referencia hacia las cuentas. Sin embargo, esta solución podría generar problemas con el paso de los años. Por ello, optaremos por utilizar una referencia inversa con el fin de evitar un arreglo que crezca indefinidamente.

Es importante destacar que comprometemos la eficiencia con esta elección, ya que una referencia directa sería más eficiente. Sin embargo, debido a las características del negocio, se sacrifica la escalabilidad y es necesario poder almacenar las cuentas, es decir, se necesita garantizar la permanencia. Además, una lista infinita de cuentas podría generar conflictos debido a las restricciones de memoria que puede tener un documento en MongoDB.

En conclusión, utilizar una referencia para la relación entre usuarios y cuentas nos permitirá mantener un modelo de datos más manejable y optimizado para las consultas y actualizaciones, a pesar de los compromisos necesarios en términos de eficiencia y escalabilidad.

- Usuario y transacción:

Guideline	**Question**	**Respuesta**
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Reference
Go Together	Do the pieces of information have a has-a, contains, or similar relationship?	Embed
Query Atomicity	Does the application query the pieces of information together?	Embed
Update Complexity	Are the pieces of information updated together?	Reference

Archival	Should the pieces of information be archived at the same time?	Reference
Cardinality	Is there a high cardinality in the child side of the relationship?	Reference
Data Duplication	Would data duplication be too complicated to manage and undesired?	Reference
Document size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	Reference
Document growth	Would the embedded piece grow without bound?	Reference
Workload	Are the pieces of information written at different times in a Escritura-heavy workload?	Reference
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	Reference

Tabla 5. Reference vs Embed en la relación usuario y transaccion

Análisis: Al responder a las preguntas, se observa que, debido a las particularidades de la búsqueda, actualización y simplicidad, la mejor solución es utilizar una referencia en esta relación. Por lo tanto, la clase Transacción tendría una referencia hacia el Usuario. Sin embargo, esta solución podría generar problemas con el paso de los años. Por ello, optaremos por utilizar una referencia inversa con el fin de evitar un arreglo que crezca indefinidamente.

Es importante destacar que comprometemos la eficiencia con esta elección, ya que una referencia directa sería más eficiente. Sin embargo, debido a las características del negocio, se sacrifica la escalabilidad y es necesario poder almacenar las transacciones, es decir, se necesita garantizar la permanencia. Además, una lista infinita de transacciones podría generar conflictos debido a las restricciones de memoria que puede tener un documento en MongoDB.

En conclusión, utilizar una referencia inversa para la relación entre usuarios y transacciones nos permitirá mantener un modelo de datos más manejable y optimizado para las consultas y actualizaciones, a pesar de los compromisos necesarios en términos de eficiencia y escalabilidad

- Transacción y punto de atención:

Guideline	**Question**	**Respuesta**
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Reference
Go Together	Do the pieces of information have a has-a, contains, or similar relationship?	Embed
Query Atomicity	Does the application query the pieces of information together?	Embed
Update Complexity	Are the pieces of information updated together?	Reference
Archival	Should the pieces of information be archived at the same time?	Reference

Cardinality	Is there a high cardinality in the child side of the relationship?	Reference
Data Duplication	Would data duplication be too complicated to manage and undesired?	Embed
Document size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	Reference
Document growth	Would the embedded piece grow without bound?	Reference
Workload	Are the pieces of information written at different times in a Escritura-heavy workload?	Reference
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	Reference

Tabla 6. Reference vs Embed en la relación transaccion y punto de atencion

Análisis: Al responder a las preguntas, se observa que, debido a las particularidades de la búsqueda, actualización y simplicidad, la mejor solución es utilizar una referencia en esta relación. Por lo tanto, la clase Transacción tendría una referencia hacia el Punto de Atención. Sin embargo, esta solución podría generar problemas con el paso de los años. Por ello, optaremos por utilizar una referencia inversa con el fin de evitar un arreglo que crezca indefinidamente.

Es importante destacar que comprometemos la eficiencia con esta elección, ya que una referencia directa sería más eficiente. Sin embargo, debido a las características del negocio, se sacrifica la escalabilidad y es necesario poder almacenar las transacciones, es decir, se necesita garantizar la permanencia. Además, una lista infinita de transacciones podría generar conflictos debido a las restricciones de memoria que puede tener un documento en MongoDB.

En conclusión, utilizar una referencia inversa para la relación entre transacciones y puntos de atención nos permitirá mantener un modelo de datos más manejable y optimizado para las consultas y actualizaciones, a pesar de los compromisos necesarios en términos de eficiencia y escalabilidad

- Punto de atención y oficina:

Guideline	**Question**	**Respuesta**
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Reference
Go Together	Do the pieces of information have a has-a, contains, or similar relationship?	Embed
Query Atomicity	Does the application query the pieces of information together?	Embed
Update Complexity	Are the pieces of information updated together?	Reference
Archival	Should the pieces of information be archived at the same time?	Reference
Cardinality	Is there a high cardinality in the child side of the relationship?	Reference

Data Duplication	Would data duplication be too complicated to manage and undesired?	Embed
Document size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	Reference
Document growth	Would the embedded piece grow without bound?	Reference
Workload	Are the pieces of information written at different times in a Escritura-heavy workload?	Reference
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	Reference

Tabla 7. Reference vs Embed en la relación oficina y punto de atención

Análisis: Al responder a las preguntas, se observa que, debido a las particularidades de la búsqueda, actualización y simplicidad, la mejor solución es utilizar una referencia en esta relación. Por lo tanto, la clase Punto de Atención tendría una referencia hacia la Oficina. Sin embargo, esta solución podría generar problemas con el paso de los años. Por ello, optaremos por utilizar una referencia inversa con el fin de evitar un arreglo que crezca indefinidamente.

Es importante destacar que comprometemos la eficiencia con esta elección, ya que una referencia directa sería más eficiente. Sin embargo, debido a las características del negocio, se sacrifica la escalabilidad y es necesario poder almacenar los puntos de atención, es decir, se necesita garantizar la permanencia. Además, una lista infinita de puntos de atención podría generar conflictos debido a las restricciones de memoria que puede tener un documento en MongoDB.

En conclusión, utilizar una referencia inversa para la relación entre oficinas y puntos de atención nos permitirá mantener un modelo de datos más manejable y optimizado para las consultas y actualizaciones, a pesar de los compromisos necesarios en términos de eficiencia y escalabilidad

- Oficina y usuario:

Guideline	**Question**	**Respuesta**
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Reference
Go Together	Do the pieces of information have a has-a, contains, or similar relationship?	Embed
Query Atomicity	Does the application query the pieces of information together?	Embed
Update Complexity	Are the pieces of information updated together?	Reference
Archival	Should the pieces of information be archived at the same time?	Reference
Cardinality	Is there a high cardinality in the child side of the relationship?	Embed

Data Duplication	Would data duplication be too complicated to manage and undesired?	Embed
Document size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	Reference
Document growth	Would the embedded piece grow without bound?	Embed
Workload	Are the pieces of information written at different times in a Escritura-heavy workload?	Reference
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	Reference

Tabla 8. Reference vs Embed en la relación oficina y usuario

Análisis: Al responder a las preguntas, se observa que, debido a las particularidades de la búsqueda, actualización y simplicidad, la mejor solución es utilizar una referencia en esta relación. Por lo tanto, la clase Usuario tendría una referencia hacia la Oficina. Sin embargo, esta solución podría generar problemas con el paso de los años. Por ello, optaremos por utilizar una referencia inversa con el fin de evitar un arreglo que crezca indefinidamente.

Es importante destacar que comprometemos la eficiencia con esta elección, ya que una referencia directa sería más eficiente. Sin embargo, debido a las características del negocio, se sacrifica la escalabilidad y es necesario poder almacenar los usuarios, es decir, se necesita garantizar la permanencia. Además, una lista infinita de usuarios podría generar conflictos debido a las restricciones de memoria que puede tener un documento en MongoDB.

En conclusión, utilizar una referencia inversa para la relación entre oficinas y usuarios nos permitirá mantener un modelo de datos más manejable y optimizado para las consultas y actualizaciones, a pesar de los compromisos necesarios en términos de eficiencia y escalabilidad

Json de cada relación entre entidades

- Usuario y cuenta: Referenciado

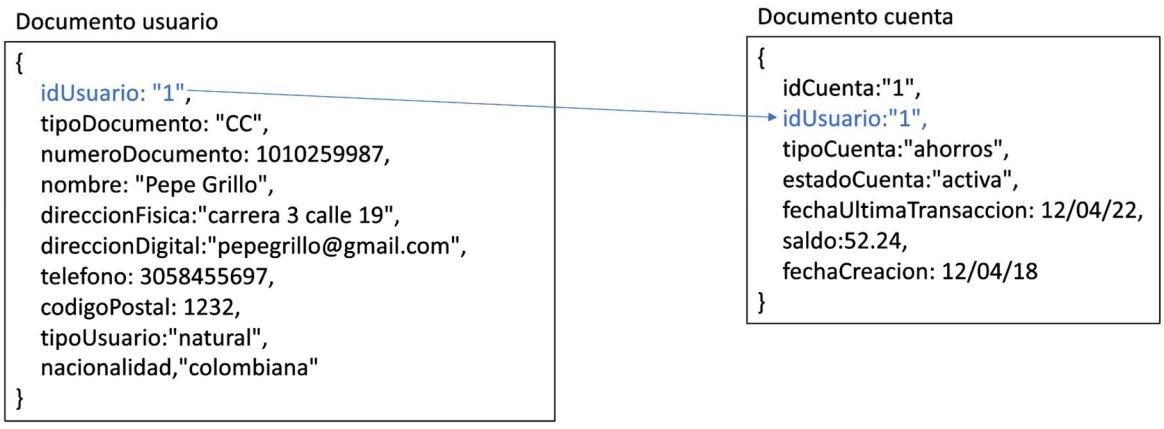


Imagen 2. Diagrama Json Usuario y Cuenta

- Usuario y transacción: Referenciado

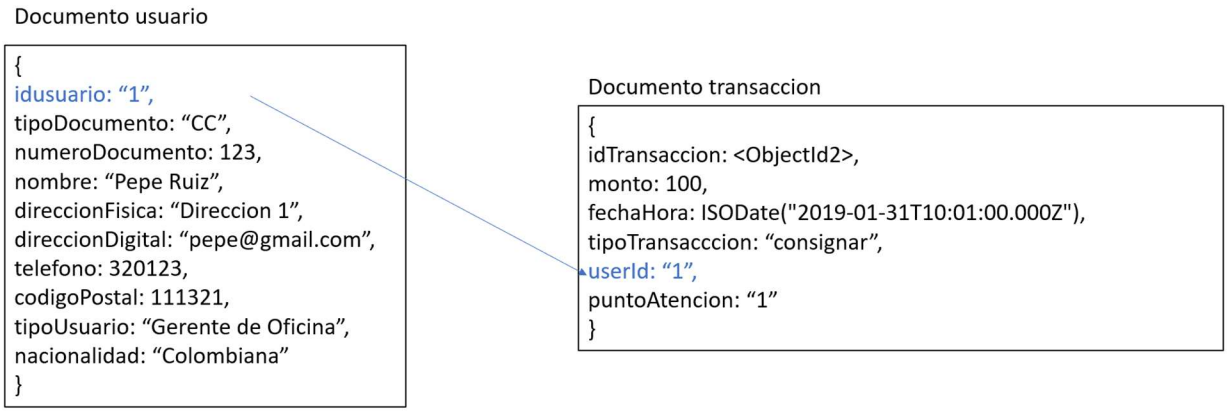


Imagen 3. Diagrama Json Usuario y Transacción

- Transacción y punto de atención: Referenciado

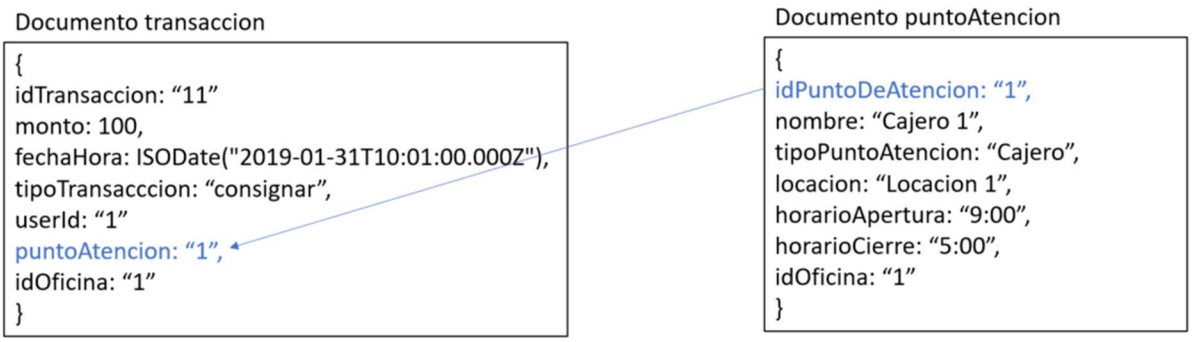


Imagen 4. Diagrama Json Transacción y Punto De Atención

- Punto de atención y oficina: Referenciado

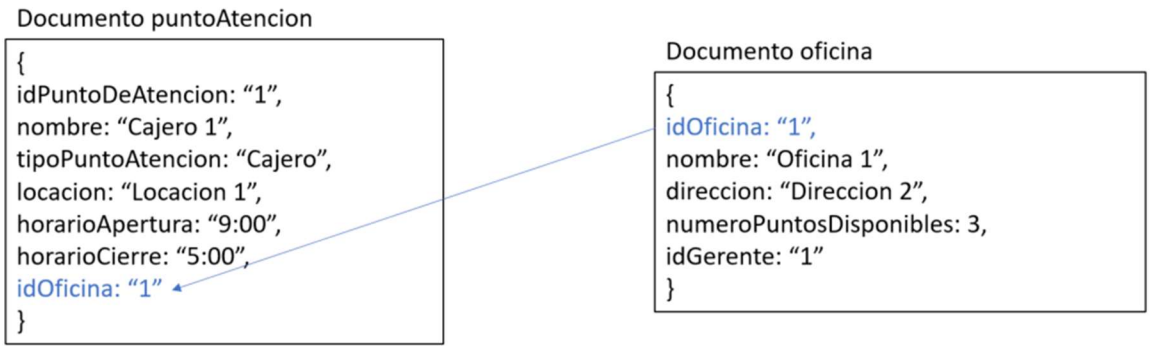


Imagen 5. Diagrama Json Punto de Atención y Oficina

- Oficina y usuario: Referenciado

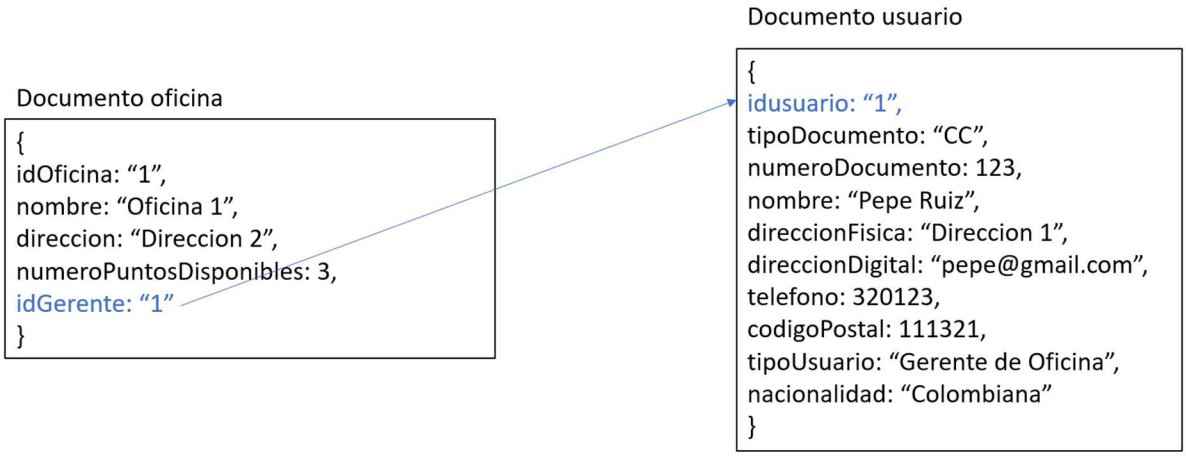


Imagen 6. Diagrama Json Oficina y Usuario

Creación de colecciones en mongo

Usuario:

```
db.createCollection("usuario", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["tipoDocumento", "numeroDocumento", "nombre", "direccionFisica", "direccionDigital",
"telefono", "codigoPostal", "tipoUsuario", "nacionalidad"],
      properties: {
```

```
        tipoDocumento: { bsonType: "string" },
        numeroDocumento: { bsonType: "int" },
        nombre: { bsonType: "string" },
        direccionFisica: { bsonType: "string" },
        direccionDigital: { bsonType: "string" },
        telefono: { bsonType: "int" },
        codigoPostal: { bsonType: "int" },
        tipoUsuario: { bsonType: "string" },
        nacionalidad: { bsonType: "string" }
      }
    }
  }
});
});
```

Punto de atención:

```
db.createCollection("puntoAtencion", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["nombre", "tipoPuntoAtencion", "horarioApertura", "horarioCierre"],
      properties: {
        nombre: { bsonType: "string" },
        tipoPuntoAtencion: { bsonType: "string" },
        locacion: { bsonType: "string" },
        horarioApertura: { bsonType: "string" },
        horarioCierre: { bsonType: "string" },
       oficina: {bsonType: "string" }
      }
    }
  }
});
```

Oficina:

```
db.createCollection("oficina", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["nombre", "direccion", "numeroPuntosDisponibles", "idGerente"],
      properties: {
        nombre: { bsonType: "string" },
        direccion: { bsonType: "string" },
        numeroPuntosDisponibles: { bsonType: "int" },
        idGerente: { bsonType: "string" }
      }
    }
  }
});
```

Cuenta:

```
db.createCollection("cuenta", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
```

```
      required: ["idUsuario", "tipoCuenta", "estadoCuenta", "fechaUltimaTransaccion", "saldo",
"fechaCreacion"],
      properties: {
        idUsuario: {bsonType: "string"},
        tipoCuenta: { bsonType: "string" },
        estadoCuenta: { bsonType: "string" },
        fechaUltimaTransaccion: { bsonType: "date" },
        saldo: { bsonType: "double" },
        fechaCreacion: { bsonType: "date" }

      }
    }
  }
});
```

Escenarios de prueba

1. CRUD entidades:

Usuario:

```
CRUD USUARIO

Read
Prueba web
PruebaWeb
prueba
Juan Perez
Ana Torres
Carlos Lopez
Carolina Silva

Create

Se creó el usuario:
Usuario [id=66538c5c0b9aab36aa6f5407, tipoDocumento=CC, numeroDocumento=123444, nombre=pruebita 1, direccionFisica=direccion de pruebita, direcc
ionDigital=@email, telefono=320470, codigoPostal=123321, tipoUsuario=CLIENTE, nacionalidad=colombiana]

Se encontró el nuevo usuario
Nombre: pruebita 1
Número de Documento: 123444

Update

Usuario modificado:
Usuario [id=66538c5c0b9aab36aa6f5407, tipoDocumento=CC, numeroDocumento=12313210, nombre=pruebita 1, direccionFisica=direccion de pruebita, dire
ccionDigital=@email, telefono=320470, codigoPostal=123321, tipoUsuario=CLIENTE, nacionalidad=colombiana]

Delete
```

Punto de Atención:

```
Read
cajero Uniandes
cajero 001
mod1
cajero automatico 01
Cajero 003
Cajero Automático Plaza Central
Autoservicio BancAndes 1
Atención Personalizada Norte
Cajero Automático Gran Estación
Autoservicio BancAndes 2

Create

Se creó el punto de atención

Se encontró el nuevo puntoAtencion
Información: PuntoAtencion [id=665390c9d12d7c065f71d1ae, nombre=pruebita 1, tipoPuntoAtencion=PERSONALIZADA, locacion=locacion, horarioApertura=
8, horarioCierre=9, idOficina=66492c81e0c54e311bf65333]

Update

Punto de Atención modificado:
PuntoAtencion [id=665390c9d12d7c065f71d1ae, nombre=pruebita mod, tipoPuntoAtencion=mod, locacion=mod, horarioApertura=8, horarioCierre=9, idOfi
ina=66492c81e0c54e311bf65333]

Delete

No se encontró el nuevo punto de Atención
Initializing Spring DispatcherServlet 'dispatcherServlet'
Initializing Servlet 'dispatcherServlet'
Completed initialization in 0 ms
```

Oficina:

```
CRUD OFICINA

Read
Oficina Uno
Oficina Dos
Oficina Tres
prueba
PRUEBA 2
Sucursal BancAndes Norte
Sucursal BancAndes Sur
Sucursal BancAndes Este
Sucursal BancAndes Oeste
Sucursal BancAndes Central

Create

Se creó la nueva oficina: Oficina [id=66539424e170473c63e994b4, nombre=pruebita 1, direccion=direccion, numeroPuntosDisponibles=1, idGerente=664
92bcce0c54e311bf65327]
Se encontró el nuevo puntoAtencion
Información: Oficina [id=66539424e170473c63e994b4, nombre=pruebita 1, direccion=direccion, numeroPuntosDisponibles=1, idGerente=66492bcce0c54e31
1bf65327]

Update

oficina modificada:
PuntoAtencion [id=66539424e170473c63e994b4, nombre=pruebita 1, tipoPuntoAtencion=null, locacion=null, horarioApertura=null, horarioCierre=null,
idOficina=null]

Delete

No se encontró la nueva Oficina.
Initializing Spring DispatcherServlet 'dispatcherServlet'
Initializing Servlet 'dispatcherServlet'
Completed initialization in 1 ms
```

Cuenta:

```
Read
6648fcbde5646c5bf7110794
6648f944eee82b69156ddf5a
66495a54574a6f38e62f73ea
6648cd7f7538112f97c3d5af
664a27e5ee329627fb97e119
6648cd7f7538112f97c3d5af
665107d3c9f50d4d43c2312b
6648cd7f7538112f97c3d5af
665107d4c9f50d4d43c2312c
6648f944eee82b69156ddf5a
665107d4c9f50d4d43c2312d
6649208444cac055f22317ae
665107d4c9f50d4d43c2312e
664925927f14aa3b70750207
665107d4c9f50d4d43c2312f
66492637a783d505d9bdc111
665107d4c9f50d4d43c23130
66492bcce0c54e311bf65325
665107d5c9f50d4d43c23131
66492bcce0c54e311bf65326

Create

Se creó la nueva cuenta: Cuenta [id=6653982fc3a18d5b76d5a20e, idUsuario=66492bcce0c54e311bf65327, tipoCuenta=AHORROS, estadoCuenta=ACTIVA, fecha
UltimaTransaccion=Sun May 26 15:14:39 COT 2024, saldo=12132.0, fechaCreacion=Sun May 26 15:14:39 COT 2024]
Se encontró el nuevo puntoAtencion
Información: Cuenta [id=6653982fc3a18d5b76d5a20e, idUsuario=66492bcce0c54e311bf65327, tipoCuenta=AHORROS, estadoCuenta=ACTIVA, fechaUltimaTransa
ccion=Sun May 26 15:14:39 COT 2024, saldo=12132.0, fechaCreacion=Sun May 26 15:14:39 COT 2024]

Update

Cuenta modificada:
Cuenta [id=6653982fc3a18d5b76d5a20e, idUsuario=66492bcce0c54e311bf65327, tipoCuenta=AHORROS, estadoCuenta=CERRADA, fechaUltimaTransaccion=Sun Ma
y 26 15:14:39 COT 2024, saldo=0.0, fechaCreacion=Sun May 26 15:14:39 COT 2024]

Delete

No se encontró la nueva cuenta.
Initializing Spring DispatcherServlet 'dispatcherServlet'
Initializing Servlet 'dispatcherServlet'
Completed initialization in 2 ms
```

2. Inserción de entidades a través de MongoDB Shell una tupla que no cumpla con su esquema de validación.

Usuario:

```
switched to db Data
db.usuario.insertOne({
    tipoDocumento: 12345,  // Debería ser un string, pero es un número.
    nombre: "Juan Pérez",  // Falta el campo "numeroDocumento".
    direccionFisica: 789,  // Debería ser un string, pero es un número.
    direccionDigital: "juan@example.com",
    telefono: "1234567890",  // Debería ser un número, pero es un string.
    codigoPostal: "ABCDE",  // Debería ser un número, pero es un string.
    tipoUsuario: "cliente",
    nacionalidad: "mexicana"
 });
▸ MongoServerError: Document failed validation
```

Punto de atención:

```
> db.puntoAtencion.insertOne({
    nombre: 12345,  // Debería ser un string, pero es un número.
    tipoPuntoAtencion: "Sucursal",
    // Falta el campo requerido "horarioApertura".
    horarioCierre: "18:00",
    locacion: "Centro",  // Este campo es opcional y correcto.
    horarioApertura: 9,  // Debería ser un string, pero es un número.
    oficina: true  // Debería ser un string, pero es un booleano.
 });
```

```
MongoServerError: Document failed validation
```

Oficina:

```
> db.oficina.insertOne({
    nombre: 98765,  // Debería ser un string, pero es un número.
    direccion: "123 Calle Principal",
    numeroPuntosDisponibles: "20",  // Debería ser un número entero, pero es un string.
    // Falta el campo requerido "idGerente".
    idGerente: null  // Este campo es requerido pero se proporciona como nulo.
 });
```

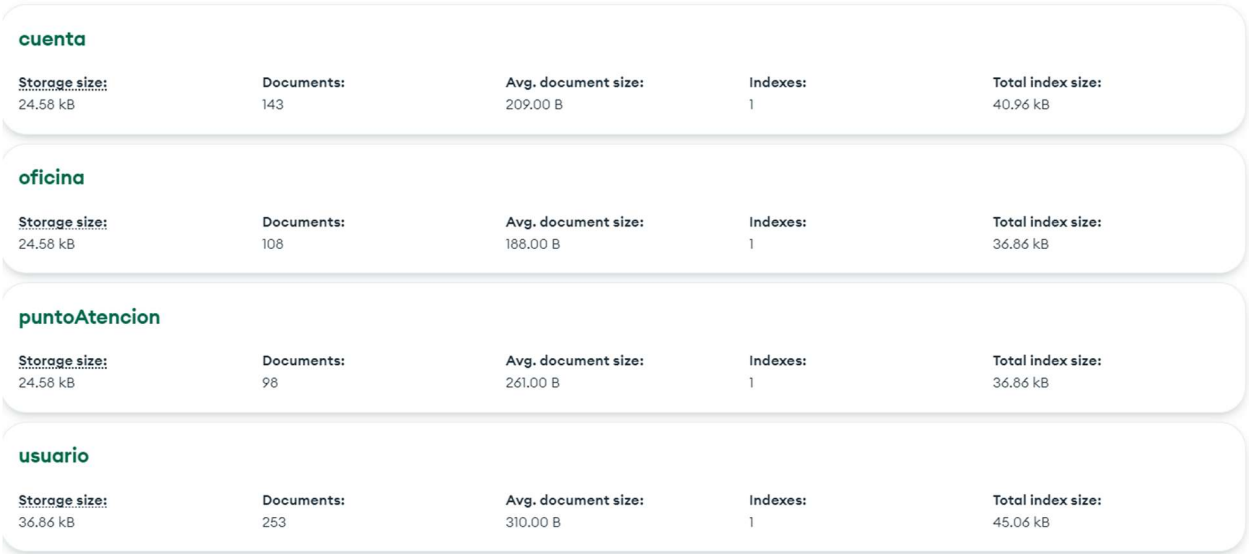
```
MongoServerError: Document failed validation
```

Cuenta:

```
> db.cuenta.insertOne({
    idUsuario: 12345,  // Debería ser un string, pero es un número.
    tipoCuenta: "Ahorro",
    estadoCuenta: "Activa",
    fechaUltimaTransaccion: "2024-05-25T14:00:00Z",  // Debería ser un tipo date, pero es un string.
    saldo: "1000.50",  // Debería ser un tipo double, pero es un string.
    // Falta el campo requerido "fechaCreacion".
 });
```

```
MongoServerError: Document failed validation
```

3. Inserción



cuenta

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
24.58 kB	143	209.00 B	1	40.96 kB

oficina

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
24.58 kB	108	188.00 B	1	36.86 kB

puntoAtencion

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
24.58 kB	98	261.00 B	1	36.86 kB

usuario

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
36.86 kB	253	310.00 B	1	45.06 kB