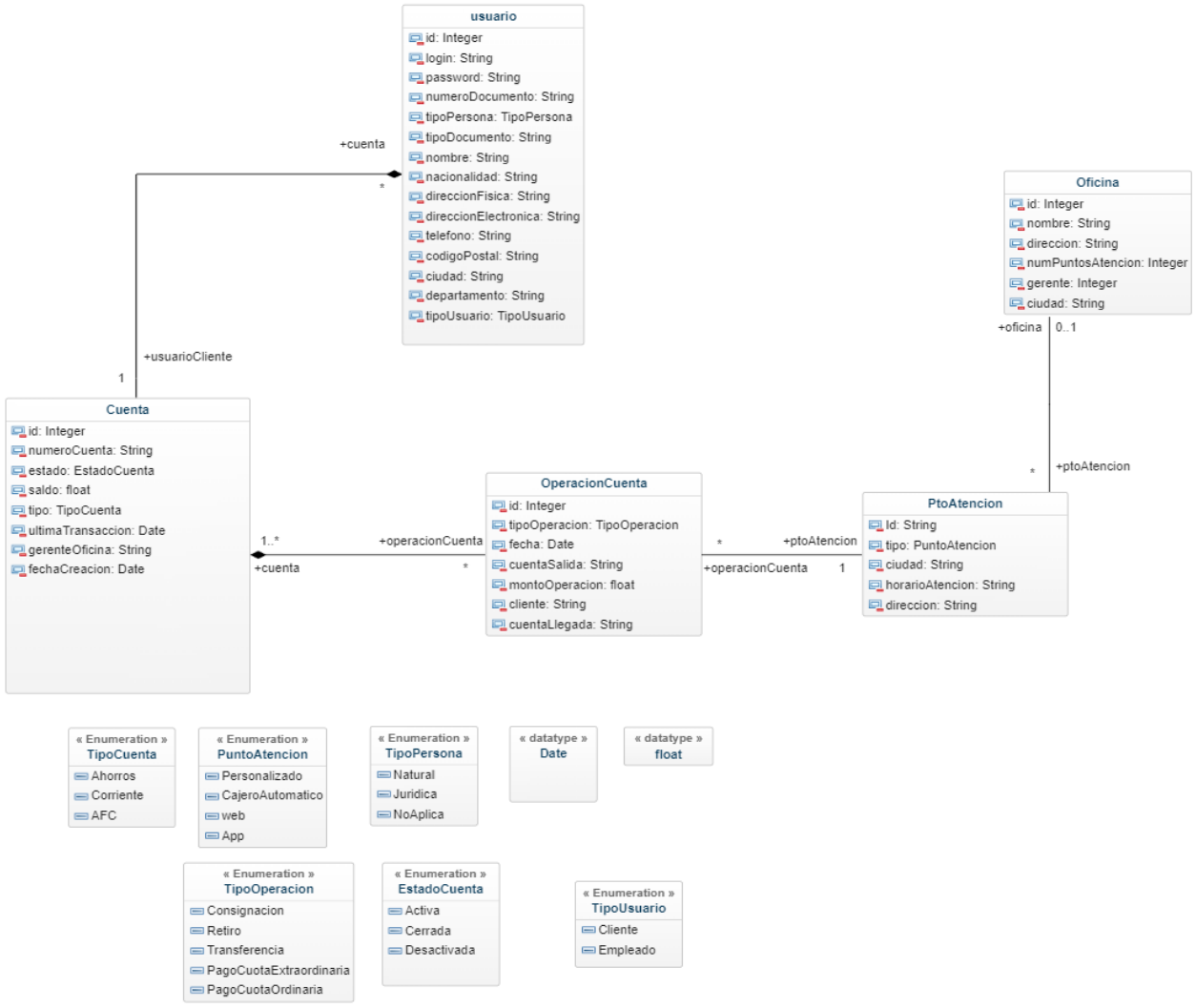
Jairo Fierro (202226326)
Marcos España (202124714)
Juan Felipe Puig (202221336)

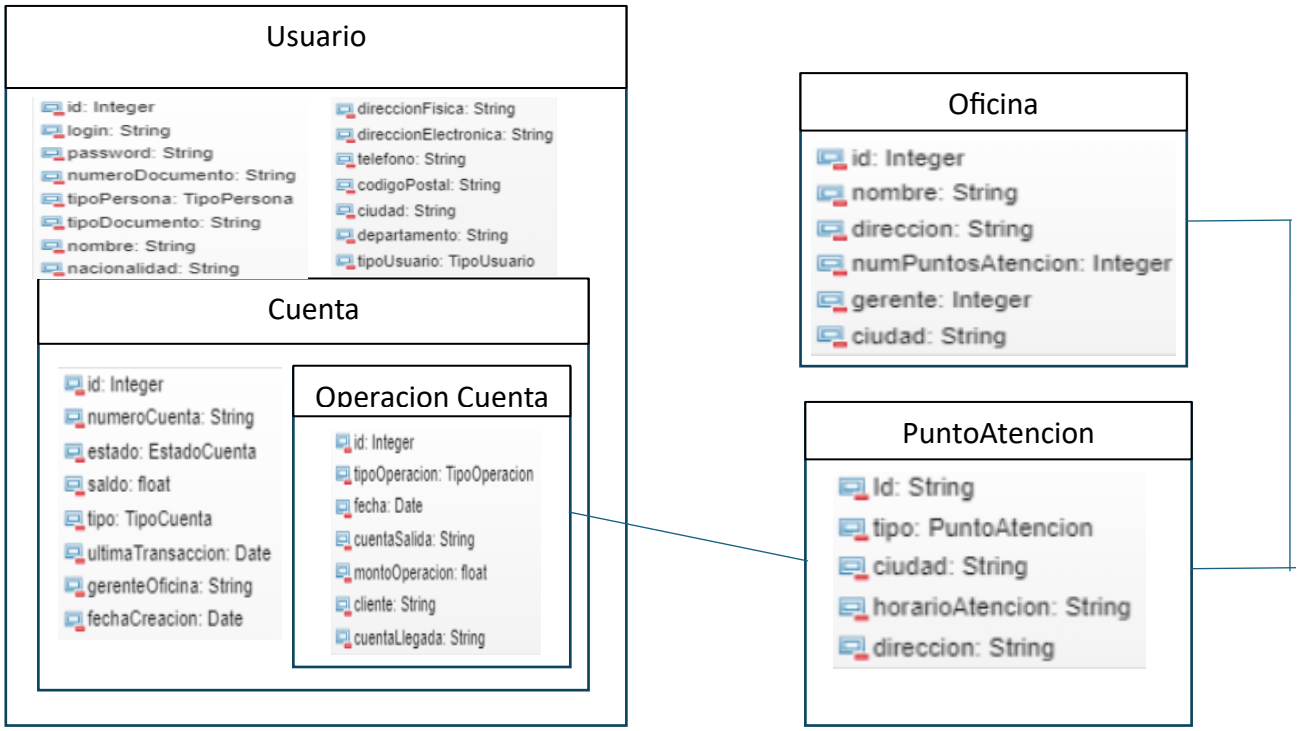
Proyecto Sistrans 3

2)Análisis y modelo conceptual

UML propuesto



Modelo NoSQL



3) Diseño de la base de datos

- Las Oficinas no se consultan, crean o modifican habitualmente. Se estima que la creación o modificación de Oficinas se haría en promedio 1 vez cada mes y su consulta 1 vez cada semana
- Los Puntos de Atención no se consultan, crean o modifican habitualmente. Se estima que la creación o modificación de Puntos de Atención se haría en promedio 1 vez cada mes y su consulta 1 vez cada semana.
- Los Usuarios del Banco se consultan, crean o modifican de manera regular. Se estima que la creación o modificación de Usuarios se haría en promedio 200 veces cada día y su consulta 500 veces cada día.
- Las Cuentas se consultan, crean y modifican habitualmente. Estas operaciones se realizan cada vez que un cliente o usuario abre una cuenta a través de alguno de los puntos de atención. Se estima que la creación o modificación de una cuenta se haría en promedio 500 vez cada día y la consulta de cuentas en promedio 5000 veces cada día.
- Las Operaciones sobre cuentas se consultan, crean y modifican muy habitualmente. Estas operaciones se realizan cada vez que un cliente realiza alguna de las operaciones señaladas en el enunciado. Se estima que el ingreso de una operación sobre cuentas se haría en promedio 20000 veces cada día y su consulta en promedio 5000 veces cada día.

- En cuanto a la cantidad de datos que se tendrán, se estima que para un Banco de tamaño medio la cantidad de Oficinas podría llegar a las 300, los puntos de atención a los 1500, los clientes a 1.500.000 y las cuentas a unas 2.500.000.
- Con respecto a las operaciones sobre cuentas, se estima que por año, se puede llegar a tener hasta 8.000.000 de operaciones, lo cual quiere decir que en una ventana de tiempo de 3 años se podría llegar a 24.000.000 de operaciones.

Parte A)

a.) Entidades y atributos identificados:

Oficinas	Id, nombre, dirección, número de puntos de atención, gerente, ciudad.
Puntos de atención	Id, tipo, ciudad, horario atención, dirección, oficina
Usuarios	Login, password, numero de documento, tipo de documento, tipo, nombre, nacionalidad, dirección física, dirección electrónica, teléfono, código postal, ciudad, departamento
Cuentas de Banco	Numero cuenta, estado, saldo, tipo, cliente, ultima transacción, gerente oficina, fecha creación
Operaciones entre cuentas	Id, tipo operación, fecha, cuenta salida, monto operación, cliente, cuenta llegada, punto de atencion

b)
Cuantificando entidades:

Entity	Quantity
Oficinas	300
Puntos de atención	1500
Usuarios del Banco	1.500.000
Cuentas de Banco	2.500.000
Operaciones entre cuentas	24.000.000

c)
Identificacion de Lecturas y Escrituras:

Entity	Operations	Information needed	Type	Rate
Oficinas	Crean/Modificar oficina	Datos de la oficina	Write	1/month
Oficinas	Leer una oficina	ID de la oficina	Read	1/week
Puntos de atención	Crean/Modificar Punto de atención	Datos del punto de atención	Write	1/month
Puntos de atención	Leer un punto de atención	ID del punto de atención	Read	1/week
Usuarios del Banco	Crean/Modificar Usuarios	Datos del usuario	Write	200/day
Usuarios del Banco	Leer un Usuario	ID del usuario	Read	500/day
Cuentas de Banco	Crean/Modificar Cuenta	Datos de la cuenta	Write	500/day
Cuentas de Banco	Leer una cuenta	Número de cuenta	Read	5000/day
Operaciones entre cuentas	Crean/Modificar Operaciones	Datos de la operación	Write	20000/day
Operaciones entre cuentas	Leer una operación	ID de la operación	Read	5000/day

d)

Cuantificación de lectura y escritura

Entity	**Operations**	**Information Needed**	**Type**	**Rate**	**Total Monthly Operations**
Oficinas	Crear/Modificar oficina	Datos de la oficina	Write	1/month	1 write/month
	Leer una oficina	ID de la oficina	Read	1/week	4 reads/month
Puntos de atención	Crear/Modificar Punto de atención	Datos del punto de atención	Write	1/month	1 write/month
	Leer un punto de atención	ID del punto de atención	Read	1/week	4 reads/month
Usuarios del Banco	Crear/Modificar Usuarios	Datos del usuario	Write	200/day	6000 writes/month
	Leer un Usuario	ID del usuario	Read	500/day	15000 reads/month
Cuentas de Banco	Crear/Modificar Cuenta	Datos de la cuenta	Write	500/day	15000 writes/month

	Leer una cuenta	Número de cuenta	Read	5000/day	150000 reads/month
Operaciones entre cuentas	Crear/Modificar Operaciones	Datos de la operación	Write	20000/day	600000 writes/month
	Leer una operación	ID de la operación	Read	5000/day	150000 reads/month

Parte B)

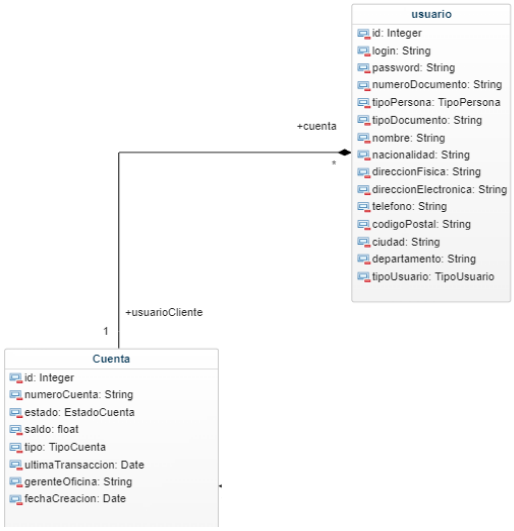
a) Lista de entidades con la descripción de cada una de ellas

- **Oficinas:** Esta entidad permite representar cada una de las oficinas que pertenece al banco. Sus atributos son los siguientes: Id, nombre, dirección, número de puntos de atención, gerente, ciudad.
- **Puntos de atención:** Representa los puntos de atención al cliente que pueden ser cajeros, cajeros automáticos, la aplicación o la pagina web. Algunos puntos de atención están relacionados con las oficinas, por ejemplo, los cajeros y los cajeros automáticos. A continuación, los atributos de esta entidad: Id, tipo, ciudad, horario atención, dirección, oficina.
- **Usuario:** Permite representar los usuarios del banco, en esta entidad se modelan los datos del cliente. Esta entidad se relaciona con la entidad cuentas para modelando la regla de negocio en donde un cliente tiene cuentas. Esta entidad cuenta con los siguientes atributos: login, password Numero de documento, tipo de documento, tipo, nombre, nacionalidad, dirección física, dirección electrónica, teléfono, código postal, ciudad, departamento.
- **Cuentas:** Esta entidad permite modelar la información de cada cuenta. Esta entidad se relaciona con la entidad cliente, para representar la regla de negocio donde cada cliente puede tener una cuenta de banco. Estos son los atributos de la entidad: Numero cuenta, estado, saldo, tipo, cliente, ultima transacción, gerente oficina, fecha creación.
- **operacionesCuentas:** Por medio de esta entidad se modelan las operaciones sobre las cuentas, para llevar un registro de las operaciones. Esta entidad se relaciona con la entidad cuenta. Esta entidad cuenta con los siguientes atributos: Id, tipo operación, fecha, cuenta salida, monto operación, cliente, cuenta llegada, punto de atención.

b,c,d)Las relaciones entre entidades y su cardinalidad. Análisis de selección de esquema de asociación (referenciado o embebido) para cada relación entre

entidades. Descripción grafica JSON

- La relación entre la entidad Cuenta y usuarioCliente es one-to-many, pues según las reglas de negocio un cliente puede tener varias cuentas, pero una cuenta solo pertenece a un cliente



Análisis de selección de esquema de asociación:

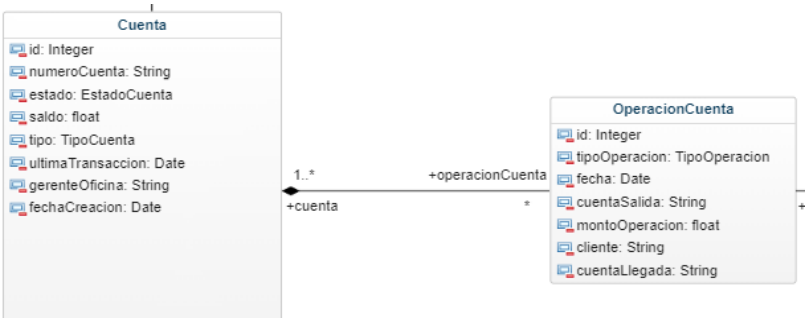
Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are the pieces of information updated together?	Yes	No
Archival	Should the pieces of information be archived at the same time?	Yes	No
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document Growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes

Embebido y solo guardo el id de la cuenta, pero creo una colleccion cuenta
Por tanto, la relación se hará mediante referenciación en donde la entidad cuenta hará referencia a usuario cliente.

```
_id: 2
login : "ja.fierro"
password : "1234"
numero_documento : "1022668766"
tipo_persona : "Natural"
tipo_documento : "CC"
nombre : "Jairo Fierro"
nacionalidad : "Colombiana"
direccion_fisica : "Calle 19"
direccion_electronica : "ja.fierro@uniandes.edu.co"
telefono : "3107889550"
codigo_postal : "233455"
ciudad : "Bogota"
departamento : "Bogota"
tipo_usuario : "Empleado"
cuentas : Array (2)
  0: Object
  1: Object
_class : "com.example.mdbspringboot.modelo.Usuario"
```

```
0: Object
    _id : 2
    numero_cuenta : "2"
    estado : "Activa"
    saldo : 1100
    tipo : "Corriente"
    ultima_transaccion : 2024-05-26T05:00:00.000+00:00
    gerente_oficina : 1
    fecha_creacion : 2024-05-26T05:00:00.000+00:00
  operaciones_cuenta : Array (empty)
```


- Relacion entre cuenta y OperacionCuenta
  La relación entre Cuenta y OperacionCuenta es uno a uno, la obligatoriedad para la clase cuenta es de cero o muchos pues una cuenta puede o no tener una operación y una operación si o si debe tener una cuenta asociada.



Se hace Embebido.

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are the pieces of information updated together?	Yes	No
Archival	Should the pieces of information be archived at the same time?	Yes	No
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document Growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	Yes
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	No	Yes

Por tanto, la mejor forma de representar esta relación es embebida

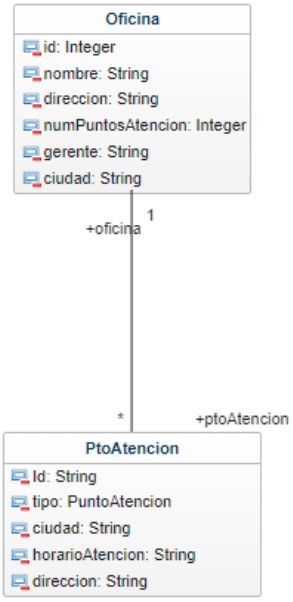
- Análisis de la relación entre operacionCuenta y ptoAtencion

Referenciado

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	~~Yes~~	No
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	~~Yes~~	No
Query Atomicity	Does the application query the pieces of information together?	Yes	~~No~~
Update Complexity	Are the pieces of information updated together?	Yes	~~No~~
Archival	Should the pieces of information be archived at the same time?	Yes	~~No~~
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	~~Yes~~
Data Duplication	Would data duplication be too complicated to manage and undesired?	~~No~~	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	~~Yes~~
Document Growth	Would the embedded piece grow without bound?	No	~~Yes~~
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	~~Yes~~
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	~~No~~	Yes

Por lo tanto, la mejor forma sería hacerlo referenciado.

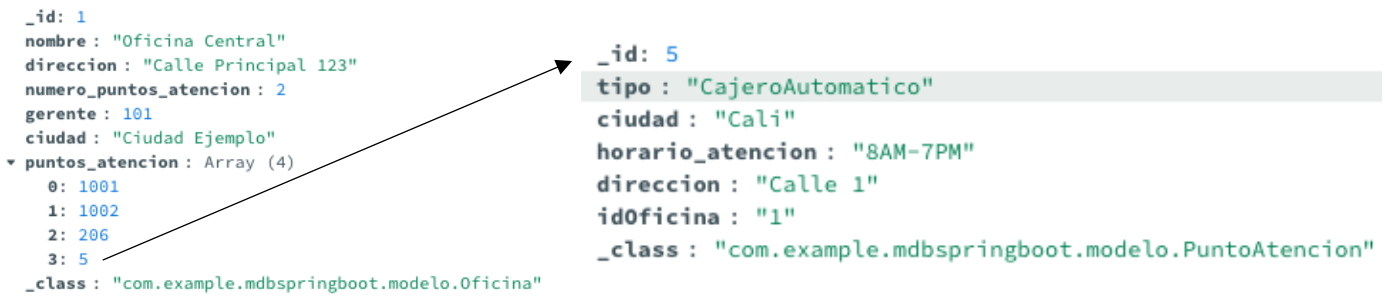
- Análisis de la relación entre oficina y ptoAtencion



Referenciado.

Guideline Name	Question	Embed	Reference
Simplicity	Would keeping the pieces of information together lead to a simpler data model and code?	Yes	No
Go Together	Do the pieces of information have a "has-a," "contains," or similar relationship?	Yes	No
Query Atomicity	Does the application query the pieces of information together?	Yes	No
Update Complexity	Are the pieces of information updated together?	Yes	No
Archival	Should the pieces of information be archived at the same time?	Yes	No
Cardinality	Is there a high cardinality (current or growing) in the child side of the relationship?	No	Yes
Data Duplication	Would data duplication be too complicated to manage and undesired?	No	Yes
Document Size	Would the combined size of the pieces of information take too much memory or transfer bandwidth for the application?	No	Yes
Document Growth	Would the embedded piece grow without bound?	No	Yes
Workload	Are the pieces of information written at different times in a write-heavy workload?	No	No
Individuality	For the children side of the relationship, can the pieces exist by themselves without a parent?	Yes	Yes

De acuerdo con el anterior análisis la mejor forma de representar esta relación es mediante el modelo embebido. Cada oficina tendrá embebidos sus puntos de atención.



Parte C

Esquemas de validación

Oficina	**Puntos de atencion**	**Usuario**
{ $jsonSchema: { bsonType: 'object', required: ['nombre', 'direccion', 'numero_puntos_atencion', 'gerente', 'ciudad'], properties: { id: { bsonType: 'int' }, nombre: { bsonType: 'string' }, direccion: { bsonType: 'string' }, numero_puntos_atencion: { bsonType: 'int' }, gerente: { bsonType: 'int' }, ciudad: { bsonType: 'string' }, puntos_atencion: { bsonType: 'array', items: { bsonType: 'int'	{ $jsonSchema: { bsonType: 'object', required: ['tipo', 'ciudad', 'horario_atencion', 'direccion'], properties: { id: { bsonType: 'int' }, tipo: { bsonType: 'string' }, ciudad: { bsonType: 'string' }, horario_atencion: { bsonType: 'string' }, direccion: { bsonType: 'string' } } } }	{ $jsonSchema: { bsonType: 'object', required: ['login', 'password', 'numero_documento', 'tipo_persona', 'tipo_documento', 'nombre'], properties: { id: { bsonType: 'int' }, login: { bsonType: 'string' }, password: { bsonType: 'string' }, numero_documento: { bsonType: 'string' }, tipo_persona: { bsonType: 'string' }, tipo_documento: { bsonType: 'string' }, nombre: { bsonType: 'string' }, nacionalidad: { bsonType: 'string' }, direccion_fisica: {

```
        }
      }
    }
  }
}
```

```
bsonType: 'string'
},
direccion_electronica: {
 bsonType: 'string'
},
telefono: {
 bsonType: 'string'
},
codigo_postal: {
 bsonType: 'string'
},
ciudad: {
 bsonType: 'string'
},
departamento: {
 bsonType: 'string'
},
tipo_usuario: {
 bsonType: 'string'
},
cuentas: {
 bsonType: 'array',
 items: {
  bsonType: 'object',
  required: [
   '_id',
   'numero_cuenta',
   'estado',
   'saldo',
   'tipo',
   'fecha_creacion'
  ],
  properties: {
   _id: {
    bsonType: 'int'
   },
   numero_cuenta: {
    bsonType: 'string'
   },
   estado: {
    bsonType: 'string'
   },
   saldo: {
    bsonType: 'double'
   },
   tipo: {
    bsonType: 'string'
   },
   ultima_transaccion: {
    bsonType: 'date'
   },
   gerente_oficina: {
    bsonType: 'int'
```

		},
		fecha_creacion: {
		bsonType: 'date'
		},
		operaciones_cuenta: {
		bsonType: 'array',
		items: {
		bsonType: 'object',
		required: [
		'id',
		'tipo_operacion',
		'fecha',
		'monto_operacion',
		'cliente'
		],
		properties: {
		id: {
		bsonType: 'int'
		},
		tipo_operacion: {
		bsonType: 'string'
		},
		fecha: {
		bsonType: 'date'
		},
		monto_operacion: {
		bsonType: 'double'
		},
		cliente: {
		bsonType: 'int'
		},
		cuenta_llegada: {
		bsonType: 'int'
		},
		punto_atencion: {
		bsonType: 'int'
		}
		}
		}
		}
		}
		}
		}
		}
		}

Pruebas

En la carpeta scripts que se encuentra dentro de la carpeta docs, están los scripts para hacer pruebas a las validaciones pruebas de validación

-En usuarioFailed1 falta el por el campo nombre

-En el script usuarioFailed2 falla por tipo de dato incorrecto, ya que el saldo debería ser tipo double.

-En usuarioFailed3 falla por falta de un campo en el un subdocumento. En este caso falta el campo fecha en operaciones_cuenta

-El script puntoAtencionFailed1 falla porque falta el campo requerido 'direccion'

puntoAtencionFailed2 falla porque el campo '_id' debería ser tipo int y en este caso es tipo String

-El script oficinasFailed1 hace una prueba donde falla porque falta el campo 'direccion'

-El script oficinasFailed2 hace una prueba donde enseña que falla porque el campo numero_puntos_atencion debería ser de tipo int, pero se proporcionó como una cadena.