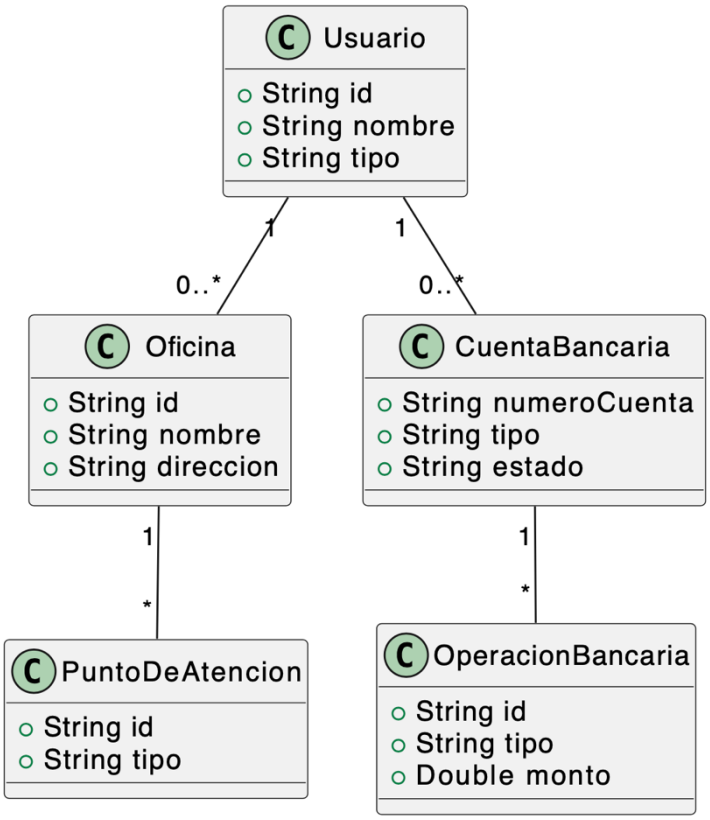
Universidad de los Andes Facultad de Ingeniería	Ingeniería de Sistemas y Computación **Pregrado** **ISIS2304 – Sistemas Transaccionales**	Accredited ABET Engineering Accreditation Commission

PROYECTO ENTREGA 3 DOCUMENTACIÓN

INTEGRANTES

Andrés Caballero	Juan José Díaz	Sebastián Palma
ac.caballero@uniandes.edu.co	jj.diazo1@uniandes.edu.co	s.palma@uniandes.edu.co
202216295	202220657	202222498

1. UML



2. DISEÑO DE LA BASE DE DATOS

2a. Identificación de Entidades y Atributos

```
Usuario: {id, nombre, tipo}
Oficina: {id, nombre, dirección}
Punto de Atención: {id, tipo}
Cuenta Bancaria: {numeroCuenta, tipo, estado}
Operación Bancaria: {id, tipo, monto}
```

2b. Cuantificación de Entidades

Usuarios: ~1.500.000

Oficinas: ~300

Puntos de Atención: ~1500

Cuentas Bancarias: ~2.500.000

Operaciones Bancarias: ~24.000.000 en 3 años

2c. Análisis de Operaciones de Lectura y Escritura

Entidad	Lecturas diarias	Escrituras diarias
Usuarios	500	200
Oficinas	1/semana	1/mes
Puntos de Atención	1/semana	1/mes
Cuentas Bancarias	5000	500
Operaciones Bancarias	5000	20000

2d. Descripción de Entidades y Relaciones en NoSQL

Entidades:

- Usuario: Representado como documento con campos id, nombre, tipo.
- Oficina: Documento con campos id, nombre, dirección.
- Punto de Atención: Documento con campos id, tipo, asociado a una Oficina.
- Cuenta Bancaria: Documento con campos númeroCuenta, tipo, estado, asociado a un Usuario.
- Operación Bancaria: Documento con campos id, tipo, monto, asociado a una Cuenta.

Relaciones y Cardinalidad:

- Usuario a Oficina: Muchos a Uno (un usuario puede gestionar muchas oficinas)
- Oficina a Punto de Atención: Uno a Muchos (una oficina tiene varios puntos de atención)
- Usuario a Cuenta Bancaria: Uno a Muchos (un usuario puede tener varias cuentas)
- Cuenta Bancaria a Operación Bancaria: Uno a Muchos (una cuenta puede tener muchas operaciones)

Análisis de Esquema de Asociación:

Usuario a Oficina:

- Tipo: Referenciado.
- Justificación: Un usuario (gerente) puede gestionar varias oficinas, pero las oficinas no cambian de gerente - frecuentemente, lo que hace innecesario embeber esta información.

Oficina a Punto de Atención:

- Tipo: Embebido.
- Justificación: Los puntos de atención rara vez se operan sin la información de la oficina asociada. Embeber esta información facilita la consulta rápida de todos los puntos de atención de una oficina específica.

Usuario a Cuenta Bancaria:

- Tipo: Referenciado.
- Justificación: Los usuarios pueden tener múltiples cuentas y las cuentas pueden necesitar ser accesadas independientemente del usuario para operaciones bancarias.

Cuenta Bancaria a Operación Bancaria:

- Tipo: Embebido.

- Justificación: Las operaciones son inherentes a la cuenta y son accedidas frecuentemente junto con la cuenta. Embeber las operaciones en el documento de la cuenta puede mejorar el rendimiento de las consultas.

Ejemplos JSON:

Oficina con Puntos de Atención Embebidos:	<pre>{
	"_id": "oficina123",
	"nombre": "Oficina Principal",
	"direccion": "Calle 50, Ciudad",
	"puntosDeAtencion": [
	{
	"_id": "punto123",
	"tipo": "Cajero Automático",
	"ubicacion": "Entrada Principal"
	},
	{
	"_id": "punto124",
	"tipo": "Atención Personalizada",
	"ubicacion": "Segundo Piso"
	}
	]
	}</pre>

Usuario con Referencia a Cuentas Bancarias:	<pre>{
	"_id": "usuario456",
	"nombre": "Juan Pérez",
	"tipo": "Cliente",
	"cuentas": [
	{"$ref": "cuentaBancaria", "$id": "cuenta789"},
	{"$ref": "cuentaBancaria", "$id": "cuenta790"}
	]
	}</pre>

Cuenta Bancaria con Operaciones Embebidas:	<pre>{
	"_id": "cuenta789",
	"numeroCuenta": "100200300",
	"tipo": "Ahorros",
	"estado": "Activa",
	"operaciones": [
	{
	"_id": "op123",
	"tipo": "Depósito",
	"monto": 200.00,</pre>

	"fecha": "2024-05-01"
	},
	{
	"_id": "op124",
	"tipo": "Retiro",
	"monto": 50.00,
	"fecha": "2024-05-02"
	}
	]
	}

3. COLLECIONES MONGODB

Colección	SCRIPT

Usuarios	db.createCollection("usuarios", {
	validator: {
	$jsonSchema: {
	bsonType: "object",
	required: ["nombre", "tipo"],
	properties: {
	nombre: {
	bsonType: "string",
	description: "Debe ser una cadena y es obligatorio"
	},
	tipo: {
	bsonType: "string",
	enum: ["Cliente", "Empleado"], // Asegurarse que sólo sean tipos válidos
	description: "Puede ser 'Cliente' o 'Empleado' y es obligatorio"
	}
	}
	}
	}
	});

Oficinas	db.createCollection("oficinas", {
	validator: {
	$jsonSchema: {

	bsonType: "object",
	required: ["nombre", "direccion"],
	properties: {
	nombre: {
	bsonType: "string",
	description: "Debe ser una cadena y es obligatorio"
	},
	direccion: {
	bsonType: "string",
	description: "Debe ser una cadena y es obligatorio"
	}
	}
	}
	}
	});

Cuentas Bancarias	db.createCollection("cuentasBancarias", {
	validator: {
	$jsonSchema: {
	bsonType: "object",
	required: ["numeroCuenta", "tipo", "estado"],
	properties: {
	numeroCuenta: {
	bsonType: "string",
	description: "Debe ser una cadena y es único"
	},
	tipo: {
	bsonType: "string",
	enum: ["Ahorros", "Corriente", "AFC"],
	description: "Puede ser 'Ahorros', 'Corriente' o 'AFC' y es obligatorio"
	},
	estado: {
	bsonType: "string",
	enum: ["Activa", "Cerrada", "Desactivada"],
	description: "Puede ser 'Activa', 'Cerrada' o 'Desactivada' y es obligatorio"
	}
	}
	}
	}
	});

Operaciones Bancarias (como subdocumento de Cuentas Bancarias)	db.createCollection("operacionesBancarias", {
	  validator: {
	  $jsonSchema: {
	  bsonType: "object",
	  required: ["tipo", "monto", "fecha", "cuentaBancariaId"],
	  properties: {
	  tipo: {
	  bsonType: "string",
	  enum: ["Depósito", "Retiro", "Transferencia"],
	  description: "Debe ser 'Depósito', 'Retiro' o 'Transferencia' y es obligatorio"
	  },
	  monto: {
	  bsonType: "double",
	  description: "Debe ser un número y es obligatorio"
	  },
	  fecha: {
	  bsonType: "date",
	  description: "Debe ser una fecha y es obligatorio"
	  },
	  cuentaBancariaId: {
	  bsonType: "objectId",
	  description: "Debe ser un ID de cuenta bancaria válido y es obligatorio"
	  }
	  }
	  }
	  }
	});

4. ESCENARIOS DE PRUEBA