**DISEÑO Y CONSTRUCCIÓN DE LA APLICACIÓN:**

1. Identifiquen entidades y sus atributos:

**1.**

**Entidad Cliente:**

**Atributos:**

* “\_id” (int): Identificador único del cliente.
* “tipodocumento” (string): Tipo de documento del cliente.
* “nombre” (String): Nombre del cliente.
* “nacionalidad” (string): Nacionalidad del cliente.
* “direccionfisica” (String): Dirección física del cliente
* “direccionelectronica” (String): dirección electrónica del cliente.
* “teléfono” (string): Número de teléfono del cliente.
* “ciudad” (String): Ciudad del cliente.
* “departamento” (String): Departamento del cliente.
* “codigopostal (int): Código postal del cliente.
* “password” (string): Contraseña del cliente.
* “cuentas” (array): lista de cuentas asociadas al cliente:
* “numcuenta” (int): Número de cuenta.
* “tipocuenta” (string): Tipo de cuenta.
* “saldo” (int): Saldo de la cuenta.
* “estado” (string): Estado de la cuenta.
* “fechaultimatransaccion” (date): Fecha de la última transacción en la cuenta.
* “fechacreacion” (date): Fecha de creación de la cuenta.
* “id\_oficina” (int): Identificador de la oficina asociada a la cuenta

**2.**

**Entidad Oficina:**

**Atributos:**

* “\_id” (int): identificador único de la oficina.
* “nombre” (string): Nombre de la oficina.
* “direccion” (string): Dirección de la oficina.
* “nombregerente” (string): Nombre del gerente de la oficina.
* “numdocgerente” (int): Numero de documento del gerente.
* “puntodeatencion” (Array): Lista de puntos de atención asociados a la oficina:
* “idPunto” (int): identificador del punto de atención.
* “ubicacion” (string): Ubicación del punto de atención.
* “tipo” (string): Tipo de punto de atención.

**3.**

**Entidad Operación de Cuenta:**

**Atributos:**

* “\_id” (int): Identificador único de la operación.
* “tipo” (string): Tipo de operación (consignación, retiro, transferencia).
* “idpuntoatencion” (int): Identificador del punto de atención donde se realizó la operación.
* “valor” (double): Valor de la operación.
* “numcuentaprincipal” (int): Número de cuenta principal involucrada en la transacción.
* “fecha” (string): Fecha de la operación.
* “numcuentadestino” (int/null): Número de cuenta de destino en caso de transferencias.

1. (incluye d) Cuantifiquen las entidades

| Entidad | Cantidad de Registros | Creación/Modificación (Frecuencia) | Consulta (Frecuencia) |
| --- | --- | --- | --- |
| Oficinas | 300 | 1 vez al mes | 100 veces a la semana |
| Puntos de Atención | 1500 | 1 vez al mes | 100 veces a la semana |
| Clientes | 1.500.000 | 200 veces al día | 500 veces al día |
| Cuentas | 2.500.000 | 500 veces al día | 5000 veces al día |
| Operaciones de Cuenta | 24.000.000 (en 3 años) | 20.000 veces al día | 5000 veces al día |

1. Analicen las operaciones de lectura y escritura para cada entidad. Para ello utilicen una tabla como la del ejemplo del anexo A. Recuerden que este análisis sirve para saber qué información se accederá de manera conjunta.

|  |  |  |  |
| --- | --- | --- | --- |
| **Entidad** | **Operaciones** | **Información necesaria** | **Tipo** |
| **Cliente** | **Crear Usuario** | **Cliente details** | **write** |
| **Oficina** | **Crear oficina** | **Oficina details** | **write** |
| **PuntoAtencion** | **Crear/Borrar punto Atención** | **Oficina details + puntoatencion details** | **write** |
| **Cuenta** | **Crear Cuenta** | **Cliente Details + cuenta detilas + id\_oficina** | **write** |
| **Cuenta** | **Update estado cuenta** | **Cliente details + cuenta details** | **Write** |
| **OperacionCuenta** | **Crear operación** | **Numero cuenta + operacionCuenta details +** | **write** |
| **Cuenta** | **Consultar cuenta** | **Cuenta detail** | **read** |
| **Operación cuenta** | **Consultar operaciones de una cuenta** | **Operaciones detail + número cuenta** | **read** |

**Describan las entidades de datos y las relaciones entre ellas (NoSQL) que corresponden al modelo conceptual UML propuesto. Para ello, presenten lo siguiente:**

**1.**

**Lista de entidades:**

* Cliente: es el usuario del servicio y quien tiene productos en el banco
* Cuentas: son las cuentas de los clientes.
* Oficina: el lugar donde están los puntos de atención y Se asocian las cuentas.
* PuntosAtencion: donde se pueden crear operaciones a las cuentas.
* Operacionescuenta: son operaciones individuales de cada cuenta.

**Relaciones:**

Clilente-Cuenta: de 1 a muchos.

Oficina-Cuenta: de 1 a muchos.

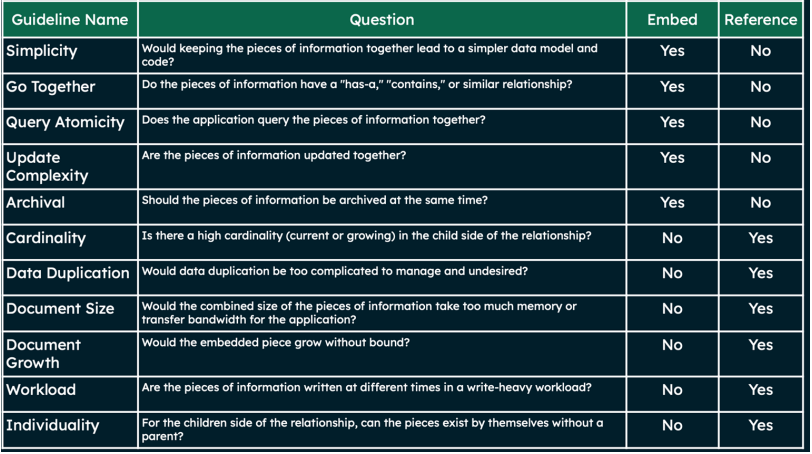
Oficina-PuntoAtencion: 1 a muchos.

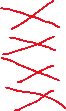
PuntosAtencion-OperacionesCuenta: 1 a muchos.

Cuenta-OperacionesCuenta: muchos a muchos.

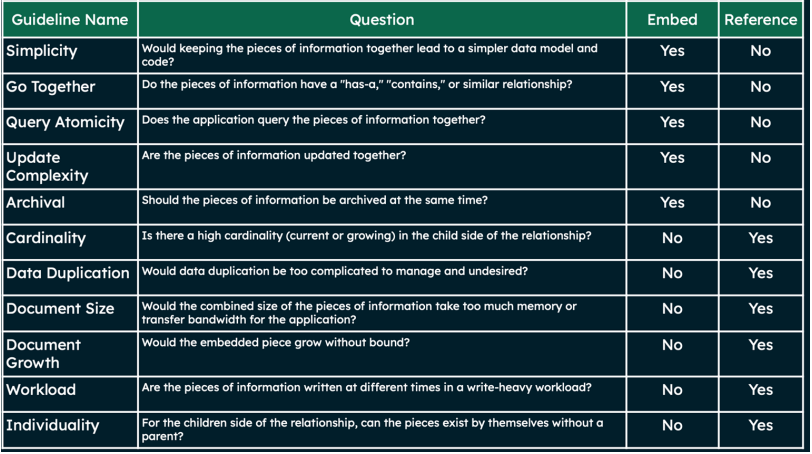
**ANALISIS DE SELECCIÓN DE ESQUEMA DE ASOCIACIÓN:**

**Cliente-Cuenta:**



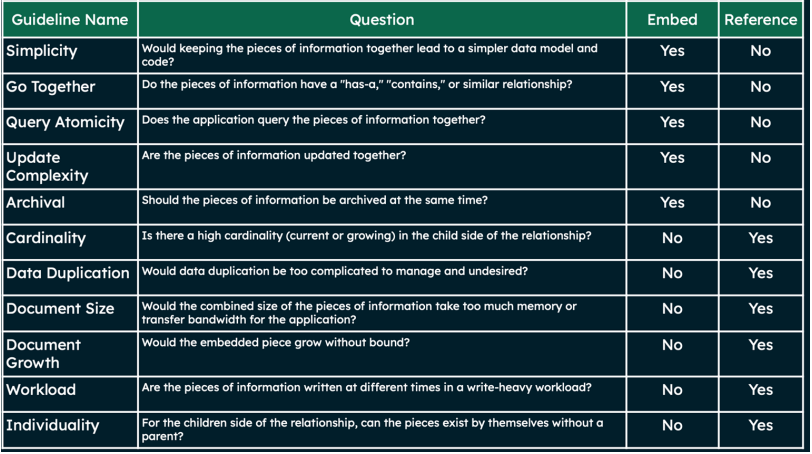


**Cuentas-OperacionesCuenta:**



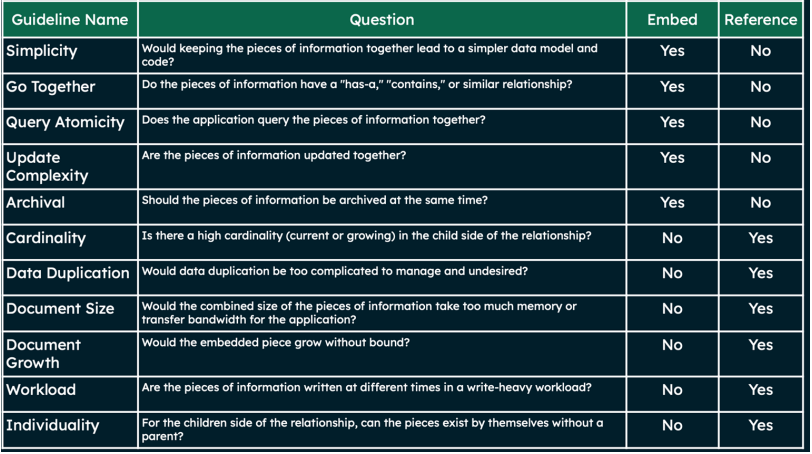


**Oficinas-PuntoAtencion:**





**Cuentas-Oficina:**





|  |  |  |
| --- | --- | --- |
| **Relación** | **Esquema** | **Justificación** |
| Cliente-Cuentas | Embedido | Tiene un acceso frecuente y en conjunto, al usar el esquema embedido simplificamos las consultas y las modificaciones. |
| Cuentas-OperacionCuenta | Referenciado | Puede llegar a existir un gran volumen operaciones para cada cuenta, entonces al usar refernciado podemos tener un tamaño manejable de documentos de cuentas. |
| Oficinas-PuntoAtencion | Embedido | Es una relación directa y de poca frecuencia, lo lógico es usar embedido para simplificar los datos. |
| Cuentas-Oficinas | Referenciado | Mantiene un tamaño manejable de documentos de cuentas, además evita la duplicación de datos y facilita la gestión independiente de cuentas y oficinas. |

d.

Cliente-Cuenta:

Embedido

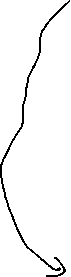




Cuenta-OperacionCuenta: Referenciado

A computer code with text

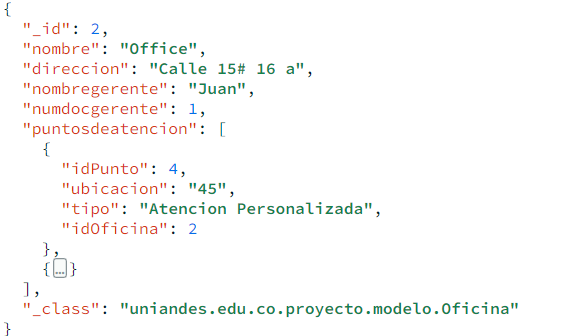
Description automatically generated



A computer code with text

Description automatically generated with medium confidence

Oficinas-PuntoAtencion: embedido





Cuentas-Oficina:

A computer code with text

Description automatically generated



A computer screen shot of text

Description automatically generated

5. **ESCENARIOS DE PRUEBA**:

2. Una inserción que no cumple con el esquema de validación para la colección de clientes (tanto cliente como cuenta se prueba acá):

db.Clientes.insertOne({

\_id: "incorrectType", // Debe ser int

tipodocumento: 12345, // Debe ser string

nombre: "Juan Pérez",

nacionalidad: "Colombiana",

direccionfisica: "123 Calle Falsa",

direccionelectronica: "juan.perez@example.com",

telefono: "555-5555",

ciudad: "Bogotá",

departamento: "Cundinamarca",

// codigopostal está ausente

password: "password123",

cuentas: [

{

numcuenta: "ABC123", // Debe ser int

tipocuenta: "Ahorros",

saldo: 5000,

estado: "Activo",

fechaultimatransaccion: "2024-04-18", // Debe ser date

fechacreacion: "2023-01-01", // Debe ser date

// id\_oficina está ausente

}

]

});

**Explicación de los errores:**

1. **Campo \_id**:
   * Debe ser de tipo int, pero se proporciona como string.
2. **Campo tipodocumento**:
   * Debe ser de tipo string, pero se proporciona como int.
3. **Campo codigopostal**:
   * Es requerido, pero está ausente.
4. **Campo cuentas.numcuenta**:
   * Debe ser de tipo int, pero se proporciona como string.
5. **Campo cuentas.fechaultimatransaccion y cuentas.fechacreacion**:
   * Deben ser de tipo date, pero se proporcionan como string.
6. **Campo cuentas.id\_oficina**:
   * Es requerido, pero está ausente.

Una inserción que no cumple con el esquema de validación para la colección de oficinas (tanto oficina como punto de atención se prueban acá):

db.Oficinas.insertOne({

nombre: "Banco Central",

direccion: 12345, // Debe ser string

\_id: "IDIncorrecto", // Debe ser int

// nombregerente está ausente

numdocgerente: "DocumentoIncorrecto", // Debe ser int

puntosdeatencion: [

{

idPunto: "Punto1", // Debe ser int

ubicacion: "Calle Principal 123",

// tipo está ausente

}

]

});

**Explicación de los errores:**

1. **Campo direccion**:
   * Debe ser de tipo string, pero se proporciona como int.
2. **Campo \_id**:
   * Debe ser de tipo int, pero se proporciona como string.
3. **Campo nombregerente**:
   * Es requerido pero está ausente.
4. **Campo numdocgerente**:
   * Debe ser de tipo int, pero se proporciona como string.
5. **Campo puntosdeatencion.idPunto**:
   * Debe ser de tipo int, pero se proporciona como string.
6. **Campo puntosdeatencion.tipo**:
   * Es requerido pero está ausente.

Una inserción que no cumple con el esquema de validación para la colección de operaciones:

db.OperacionesCuenta.insertOne({

tipo: 12345, // Debe ser string

idpuntoatencion: "IDIncorrecto", // Debe ser int

valor: "ValorIncorrecto", // Debe ser int

numcuentaprincipal: "CuentaPrincipalIncorrecta", // Debe ser int

\_id: "IDIncorrecto", // Debe ser int

// fecha está ausente

numcuentadestino: "CuentaDestinoIncorrecta" // Debe ser int o null

});

A continuación, se muestra el ejemplo de un extracto:

