# 📊 Cálculo y Visualización de Retención Estudiantil

## Contexto

La Universidad Católica del Norte (UCN) dispone de cientos de miles de registros relacionados con la historia académica de cada estudiante.
Cada registro representa un **estado académico** asociado a un estudiante dentro de una carrera, en un año determinado.

### Campos de cada registro

- `rut`: identificador del estudiante
- `nombre`: nombre del estudiante
- `year_admision`: año de ingreso/admisión a la universidad
- `cod_programa`: código de carrera
- `nombre_estandar`: nombre de carrera
- `catalogo`: plan/catálogo de la carrera que estudia
- `year_estado`: año del estado
- `cod_estado` y `nombre_estado`: código y descripción del estado (ej. `"M" = Matriculado`)

> **Nota:** Los estudiantes se matriculan todos los años. Existen múltiples registros por estudiante y por carrera, representando distintos estados en el tiempo.

---

## Definición de Retención (Regla de Negocio)

Se define **Retención Año 1 (año+1)** para una carrera como:

1. **Primera matrícula válida**:
   Un estudiante es considerado "matriculado por primera vez en esa carrera" si existe al menos un registro con:
   - `cod_estado == "M"`
   - `year_admision == year_estado`

   Es decir, se matriculó en el mismo año de su admisión.

2. **Retenido**:
   Ese estudiante cuenta como retenido si además existe al menos un registro con:
   - `cod_estado == "M"`
   - `year_estado == (año de primera matrícula + 1)`
   - Para la misma carrera/catálogo

En otras palabras: entró y se matriculó en su año de admisión, y volvió a matricularse al año siguiente.

---

## 🎯 Objetivo General (Backend)

Construir un **backend** que:

- Lea el archivo `.json` como fuente de datos.
- Calcule indicadores de retención.
- Exponga una **API REST** que entregue resúmenes de retención por año, carrera, plan, etc.
- Esté diseñado con una arquitectura que permita reemplazar el origen JSON por una **base de datos relacional** en el futuro, sin reescribir la lógica principal.

---

## ⚙️ Requerimientos Funcionales

El sistema debe permitir obtener, como mínimo:

- **Retención global por año de cohorte**
  Para cada año `Y` (cohorte), calcular:
  - `matriculados_primera_vez`: número de estudiantes con primera matrícula en `Y`
  - `retenidos_Y+1`: número de esos estudiantes que se matriculan también en `Y+1`
  - `tasa_retencion`: `retenidos_Y+1 / matriculados_primera_vez`
- **Retención por carrera (y año de cohorte)**
  Igual que lo anterior, pero filtrado por `cod_programa`.
- **Retención con filtros** (al menos algunos de los siguientes):
  - Por `catalogo` (plan)
  - Por `cod_admision` / admisión
  - Por rango de años (ejemplo: `from=2010&to=2020`)

---

## 🔌 Endpoints REST

El backend debe exponer al menos los siguientes endpoints:

- `GET /api/retencion/resumen`
  Retención por año (todas las carreras).
- `GET /api/retencion/carreras`
  Listado de carreras disponibles (para poblar filtros en un frontend).
- `GET /api/retencion/por-carrera?cod_programa=8003`
  Retención por año para una carrera específica.

### Cada respuesta debe incluir:

- Año cohorte (`year`)
- Carrera (`cod_programa`, `nombre_estandar` cuando aplique)
- `matriculados_primera_vez`
- `retenidos_anio_siguiente`
- `tasa_retencion`

---

## Requerimientos de Arquitectura

El sistema debe implementarse con separación clara entre capas:

- **Repository / DAO (Acceso a datos)**
  Mock que lee desde el JSON (ejemplo: `JsonStudentStateRepository`).
- **Service (Lógica de negocio)**
  Implementa el cálculo de cohorte y retención. No debe depender directamente de JSON.
- **Controller / Routes (API)**
  Expone endpoints, valida parámetros y entrega respuestas.

> El diseño debe permitir reemplazar el repositorio JSON por uno basado en SQL (PostgreSQL/MySQL), manteniendo intacta la lógica de cálculo.

---

## Entregables

1. Código fuente del backend.
2. Breve documento o README que describa:
   - Cómo ejecutar el proyecto.
   - Cómo se define y calcula la retención.
   - Qué vistas incluye y cómo se usan.

---

## Observaciones

- Comprima todos los archivos de su trabajo y suba dicho archivo a **CampusVirtual**.
- El archivo es grande: se valorará manejo eficiente de los datos.
- Un estudiante puede tener múltiples registros por año: para retención basta con verificar existencia de matrícula `"M"` en el año correspondiente.
- La retención se calcula **por carrera**, no solo por estudiante global.