ESCUELA DE INGENIERÍA UCN COQUIMBO

Examen Individual

Proyecto Integrador Software – 19 de Diciembre 2025

Nombre		RUT	

Antecedentes generales:

Puntaje total de la prueba/Puntos para nota aprobatoria(4.0)	100 puntos 60 puntos	**Puntaje Obtenido**	
Duración de la prueba	5 horas	**Nota final**	
Resultados de Aprendizaje a evaluar	1. Integrar las buenas prácticas para el desarrollo de software seguro en los procesos de producción de software. 3. Gestionar un proyecto de desarrollo de software, incorporando actividades de gestión de calidad 4. Ejecutar un proyecto de desarrollo de software utilizando un modelo de proceso iterativo e incremental. 5. Identificar los objetivos y requerimientos de las soluciones TIC 6. Seleccionar los procesos, técnicas y herramientas adecuados de acuerdo a los requerimientos. 7. Desarrollar la solución tecnológica más adecuada en base a las características del problema y los recursos disponibles. 8. Aplicar herramientas TIC para el almacenamiento, la comunicación, la transmisión e intercambio de información de manera efectiva.		
Fecha de entrega de resultados	22 diciembre 2025		

Instrucciones:

1. Esta evaluación tiene 5 páginas (incluyendo la portada). Compruebe que dispone de todas las páginas.
2. Lea la prueba completamente **DOS** veces antes de hacer cualquier pregunta
3. Una prueba respondida correctamente en un 60% corresponde a una nota 4,0.
4. Solamente se pueden realizar preguntas durante los primeros 30 minutos de la prueba. Solo se responderán preguntas respecto a los enunciados usando el canal de comunicación informado.
5. La prueba es individual, cualquier sospecha de copia será calificada con la nota mínima y el caso será remitido a las autoridades de la Escuela de Ingeniería.
6. Al entregar su solución, DEBE incluir una copia de este documento, con sus datos y su firma en los lugares que corresponda.
7. Después de leer el contexto general de la problemática, debe elegir si implementa el frontend o el backend de acuerdo a las especificaciones respectivas.

Acepto las condiciones firmando: _____

Cálculo y visualización de retención estudiantil

Contexto

La UCN dispone de cientos de miles de registros relacionados con la historia estudiantil de cada estudiante. Cada registro representa un estado académico asociado a un estudiante dentro de una carrera, en un año determinado. Cada registro contiene los siguientes campos:

- 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)

Entre los estados, uno de los más relevantes es: cod_estado = "M", que indica que el/la estudiante se matriculó ese año (recuerda que las/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. Un estudiante es considerado “matriculado por primera vez en esa carrera” si existe al menos un registro con:
    i. cod_estado == "M" y
    ii. year_admision == year_estado

    Es decir: se matriculó en el mismo año de su admisión (primera matrícula en esa carrera).

2. Ese estudiante cuenta como **retenido** si además existe al menos un registro con:
    i. cod_estado == "M" y
    ii. 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 (FrontEnd)

Construir una aplicación frontend que lea los datos desde su backend, y que calcule y visualice la retención de estudiantes por año y por carrera, con resúmenes y filtros que permitan explorar los resultados.

Requerimientos funcionales

La aplicación debe calcular métricas de retención: para cada combinación relevante (por ejemplo: año de admisión + carrera), calcular:

- Cohorte: cantidad de estudiantes con primera matrícula válida (regla 1).
- Retenidos: cantidad que se matricula en año+1 (regla 2).
- Porcentaje de retención = retenidos / cohorte * 100.

Mostrar resultados en la interfaz:

- Tabla resumen por año (cohortes y retención).
- Tabla por carrera (filtrable por año).

Vistas que permitan agrupar o filtrar por:

- año de admisión (cohorte)
- carrera (cod_programa / nombre)

Como el backend real no está disponible, su frontend debe cargar los datos simulando el backend. En otras palabras, debe simular un backend mediante un módulo/servicio que lea el JSON entregado y provea resultados como si fueran respuestas de API.

Debe diseñar el frontend como si en el futuro consumiera un backend real (capas: services/api, DTOs, etc.). En otras palabras, al conectar el frontend a un backend real, no debe ser necesario cambiar nada excepto lo mínimo necesario en las clases de servicios/api.

Requerimientos técnicos mínimos

1. La solución debe ser un frontend web (framework libre: React/Vue/Angular u otro).
2. Debe existir una capa de “API”, por ejemplo:
    i. getRetentionSummaryByYear()
    ii. getRetentionByProgram(year)
    iii. getPrograms()
3. Se debe implementar una versión de la API, pero como mock, usando los datos del archivo JSON entregado.
4. El diseño debe permitir reemplazar el mock por un backend real sin reescribir toda la UI. No debe escribir el mock “real”.

Entregables

1. Código fuente del frontend.
2. Breve documento o README que describa:
    a. cómo ejecutar el proyecto,
    b. cómo se define y calcula la retención,
    c. qué vistas incluye y cómo se usan.

Observaciones

1. Comprima todos los archivos de su trabajo y suba dicho archivo a CampusVirtual.

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). Lo mismo anterior, pero filtrado por cod_programa.

Retención con filtros (al menos algunos de los siguientes)

- por catalogo (plan)
- por cod_admision / admision
- por rango de años (por ejemplo, from=2010&to=2020)

Requerimientos técnicos mínimos

Tu backend debe exponer endpoints REST (puedes ajustar nombres, pero deben cubrir la funcionalidad):

- 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, al menos, campos como:

- año cohorte (year)
- carrera (cod_programa, nombre_estandar cuando aplique)
- matriculados_primera_vez
- retenidos_anio_siguiente
- tasa_retencion

Requerimientos de arquitectura (pensando en BD futura)

El sistema debe implementarse con separación clara entre:

- Capa de acceso a datos (Repository / DAO): En esta entrega, esta capa será un mock que lee desde el JSON (por ejemplo, JsonStudentStateRepository).
- Capa de lógica de negocio (Service): Aquí se implementa el cálculo de cohorte y retención. Esta capa no debe depender de JSON directamente.
- Capa de API (Controller / Routes): Expone endpoints, valida parámetros, entrega respuestas.

En la evaluación se considerará que el diseño permita en el futuro 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:
    a. cómo ejecutar el proyecto,
    b. cómo se define y calcula la retención,
    c. qué vistas incluye y cómo se usan.

Observaciones

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