UNACH

UNIVERSIDAD ADVENTISTA DE CHILE

ARTÍCULO TÉCNICO

Sistema de Cálculo de Carga Horaria

Arquitectura, Consolidación y Procesamiento Asincrónico

Versión 1.0 | mayo de 2026

UNACH
UNIVERSIDAD ADVENTISTA DE CHILE

Tabla de Contenidos

UNACH
UNIVERSIDAD ADVENTISTA DE CHILE

Resumen

El sistema implementa un motor de cálculo de carga horaria orientado a la administración académica y operativa de establecimientos educacionales.

La solución permite:

- Calcular horas pedagógicas y no pedagógicas
- Consolidar métricas institucionales
- Validar distribución horaria
- Detectar sobreasignaciones
- Automatizar recalculos
- Ejecutar procesamiento masivo concurrente
- Mantener sincronización institucional mediante eventos asincrónicos

El módulo fue desarrollado utilizando NestJS, PostgreSQL y TypeORM, bajo una arquitectura desacoplada basada en servicios, listeners y tareas ejecutables.

UNACH
UNIVERSIDAD ADVENTISTA DE CHILE

1. Introducción

El sistema implementa un módulo de cálculo de carga horaria orientado a la gestión académica y administrativa de establecimientos educacionales. Su propósito es centralizar el cálculo de horas asignadas a docentes y asistentes, controlar la distribución de horas pedagógicas y no pedagógicas, y mantener indicadores consolidados por período escolar.

La solución fue desarrollada utilizando:

- NestJS como framework backend
- TypeORM para acceso a datos
- PostgreSQL como motor de base de datos
- Arquitectura basada en eventos asincrónicos mediante listeners y tareas desacopladas

2. Objetivo del Módulo

El módulo de cálculo tiene como finalidad:

- Calcular la carga horaria total de funcionarios
- Distribuir horas pedagógicas y no pedagógicas
- Validar porcentajes de utilización
- Detectar sobreasignaciones horarias
- Consolidar métricas por período escolar
- Automatizar recalculos ante cambios en asignaciones
- Mantener sincronizada la información institucional

3. Arquitectura General del Cálculo

El sistema trabaja sobre múltiples entidades relacionadas entre sí:

Entidad	Responsabilidad
PersonSchool	Relación del funcionario con un establecimiento
PersonFunction	Información contractual
PersonSchoolNthTable	Configuración horaria aplicada
GradeSubjectTeacher	Horas asignadas en asignaturas
Activity	Actividades no lectivas
CalculatePersonSchool	Resultado consolidado individual
CalculateSchoolPeriod	Resultado consolidado institucional

UNACH
UNIVERSIDAD ADVENTISTA DE CHILE

4. Flujo General del Cálculo

El proceso completo se divide en tres niveles:

4.1 Cálculo Individual del Funcionario

Método principal:

```
calculateCompleteFuncionary(personSchoolId)
```

Responsable de calcular horas pedagógicas, horas no pedagógicas, horas no lectivas, recreos, porcentajes, excedentes y disponibilidad restante.

4.2 Consolidación Institucional

Método:

```
calculateFunctionaryHours(schoolPeriodId)
```

Responsable de consolidar métricas globales, calcular progreso institucional, totalizar horas docentes y asistentes, y medir avance del período escolar.

4.3 Recálculo Masivo

Método:

```
calculateFunctionaryAll(academicPeriodId)
```

Responsable de recalcular todos los establecimientos y funcionarios de un período académico completo.

UNACH
UNIVERSIDAD ADVENTISTA DE CHILE

5. Cálculo Individual del Funcionario

5.1 Obtención de Datos Base

El sistema obtiene: funcionario (PersonSchool), período escolar, período académico, configuración horaria, asignaturas asignadas y actividades registradas. Esto permite construir el contexto completo del cálculo.

5.2 Configuración Automática de Tablas Horarias

Cuando un docente no posee configuración horaria asociada, el sistema crea automáticamente una configuración inicial para garantizar que todo docente tenga: horas lectivas, horas no lectivas, recreos y planificación.

5.3 Uso de Tablas NTH

Las configuraciones horarias se obtienen desde tablas institucionales NTH. Cada configuración define:

Campo	Descripción
teachingHoursHa	Horas aula
teachingHoursHc	Horas cronológicas
recreation	Tiempo recreo
nonTeachingHours	Horas no lectivas

5.4 Conversión de Tiempos

El sistema utiliza intervalos PostgreSQL para almacenar tiempos.

```
intervalToSeconds()      → Conversión a segundos
secondsToIntervalString() → Conversión a intervalos
```

Esto permite cálculos precisos, compatibilidad con PostgreSQL y operaciones matemáticas eficientes.

5.5 Cálculo de Horas Pedagógicas

Las horas pedagógicas consideran horas lectivas, horas no lectivas y recreos. Fórmula general:

```
totalPedagogicalHours = totalTeachingHoursHc + totalNonTeachingHours + totalRecreation
```

5.6 Cálculo de Horas Utilizadas

Las horas utilizadas provienen de asignaturas (GradeSubjectTeacher) y actividades (Activity). Cada hora pedagógica se transforma a segundos para unificar cálculos y las horas de actividades se manejan cronológicamente:

```
totalUsedTeachingHoursHc = totalUsedTeachingHoursHa * 45 * 60
```

5.7 Cálculo de Disponibilidad Restante

El sistema calcula horas restantes mediante la fórmula rest = total - used, utilizando Math.max(total - used, 0) para evitar valores negativos.

5.8 Cálculo de Excedentes

Cuando las horas utilizadas superan el límite disponible, el sistema registra sobreasignación horaria:

```
exceeded = used - total
```

5.9 Cálculo Porcentual

Se utiliza una función segura safeDivide(num, den) que evita divisiones por cero:

```
percentage = den > 0 ? (num / den) * 100 : 0
```

5.10 Subsidios Institucionales

El cálculo separa horas según subsidio. Cada bloque horario se consolida individualmente:

Subsidio	Descripción
SG	Subvención General
PIE	Programa de Integración Escolar
SEP	Subvención Escolar Preferencial

UNACH
UNIVERSIDAD ADVENTISTA DE CHILE

6. Consolidación Institucional

El método calculateFunctionaryHours() realiza la consolidación global del establecimiento.

6.1 Métricas Calculadas

El sistema contabiliza funcionarios con percentageTotal === 100 y calcula:

- Progreso docentes
- Progreso asistentes
- Progreso cursos

```
totalProgress = (progressAssistants + progressTeachers + progressGrades) / 3
```

6.2 Consolidación de Horas

El sistema totaliza: horas docentes, horas asistentes, horas utilizadas, horas restantes y horas de cursos.

6.3 Persistencia de Resultados

Los resultados son almacenados para evitar recalcular información constantemente:

Tabla	Propósito
CalculatePersonSchool	Resultado individual
CalculateSchoolPeriod	Resultado institucional

UNACH
UNIVERSIDAD ADVENTISTA DE CHILE

7. Procesamiento Masivo Concurrente

El sistema incorpora procesamiento concurrente controlado mediante el método runWithConcurrency(), que permite recalcular múltiples funcionarios simultáneamente sin saturar la base de datos.

7.1 Arquitectura Asincrónica Basada en Eventos

El sistema utiliza listeners desacoplados mediante @OnEvent(), implementado con EventEmitter, tareas asincrónicas y procesamiento diferido.

7.2 Eventos Soportados

Recálculo institucional:

```
school-period.recalculate → Ejecuta calculateFunctionaryHours()
```

Recálculo individual:

```
school-period.recalculate-person-school → Ejecuta calculateCompleteFuncionary()
```

7.3 Debounce de Eventos

El listener implementa un mecanismo debounce utilizando Map<number, NodeJS.Timeout> para evitar múltiples recalculos consecutivos innecesarios. Cuando ocurren múltiples cambios: se cancela el temporizador anterior, se espera un intervalo y se ejecuta un único recalculo consolidado.

Beneficios:

- Menor carga de base de datos
- Menos recalculos duplicados
- Mejor rendimiento
- Mayor estabilidad

UNACH
UNIVERSIDAD ADVENTISTA DE CHILE

8. Ejecución de Cálculos Mediante Scripts CLI

El sistema incorpora scripts ejecutables mediante npm para permitir la ejecución manual o automatizada de recalculos masivos de carga horaria. Estos scripts permiten: recalcular períodos académicos completos, ejecutar procesos de mantenimiento, sincronizar información institucional, reprocesar datos históricos y automatizar tareas programadas.

8.1 Definición del Script

```
"calculate-all-values": "npx ts-node -r tsconfig-paths/register
./src/scripts/calculate-all-values.ts"
```

8.2 Implementación

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from 'src/app.module';
import { CalculateSchoolPeriodTaskService } from 'src/school-general/...';

async function bootstrap() {
  const app = await NestFactory.createApplicationContext(AppModule);
  const taskService = app.get(CalculateSchoolPeriodTaskService);
  const academicPeriodId = parseInt(process.argv[2], 10);
  if (!academicPeriodId) { process.exit(1); }
  await taskService.calculateFunctionaryAll(academicPeriodId);
  console.log('✅ Ejecución completada');
}
```

8.3 Funcionamiento del Script

1. Inicialización del contexto NestJS: el script inicializa el contenedor completo sin levantar servidor HTTP, reutilizando servicios, repositorios, inyección de dependencias y lógica de negocio.

2. Obtención del servicio principal: se obtiene el servicio encargado de ejecutar los cálculos institucionales.

3. Recepción y validación de parámetros: el identificador del período académico es recibido desde la línea de comandos y validado antes de ejecutar el proceso.

4. Ejecución del cálculo masivo: el sistema ejecuta recalculo individual de funcionarios, consolidación institucional, actualización de métricas y persistencia de resultados.

8.4 Ejemplo de Ejecución

```
npm run calculate-all-values 1
```

Parámetro	Descripción
1	Identificador del período académico

8.5 Beneficios del Enfoque CLI

La ejecución mediante scripts desacoplados permite:

- Automatización mediante cron jobs
- Integración con pipelines CI/CD
- Mantenimiento institucional
- Reprocesamiento controlado
- Independencia del servidor HTTP
- Menor consumo de recursos

9. Estrategias de Rendimiento

El módulo incorpora las siguientes optimizaciones:

Estrategia	Beneficio
Promise.all	Paralelismo
Procesamiento concurrente	Mejor throughput
Consolidación persistida	Menos recalculo
Debounce	Reducción de eventos
Conversión en memoria	Menos operaciones SQL
Agrupación previa	Menos consultas repetidas

10. Beneficios de la Arquitectura

La solución implementada proporciona:

- Alta escalabilidad
- Desacoplamiento de procesos
- Trazabilidad de cálculos
- Consolidación institucional
- Automatización de recalculos
- Control de sobrecarga horaria
- Flexibilidad para distintas configuraciones académicas

11. Conclusión

El sistema de cálculo de carga horaria implementa una arquitectura orientada a procesos asincrónicos y consolidación progresiva de información académica. Sumado a su manejo de las horas a nivel de segundos para tener el control completo de las asignaciones de las horas en los funcionarios.

Mediante el uso de eventos, cálculos desacoplados y persistencia de resultados, el sistema logra administrar eficientemente la distribución horaria institucional, manteniendo consistencia y rendimiento incluso en escenarios de alta carga operativa.