Códigos Usados en el Sistema Académico

Diseño_Sistema_Académico.md

Código utilizado:
Diseño y construcción: Sistema Académico (MongoDB + Redis)
Resumen
Proyecto: **Sistema académico** para una universidad/escuela que gestiona estudiantes, profesores, curso Requisitos implementados:
MongoDB: varias colecciones, documentos con campos anidados y arreglos, ejemplos CRUD.
Redis: uso de strings, hashes, lists, sets, sorted sets (al menos 3 tipos) y ejemplos con TTL.
Modelado Redis para complementar MongoDB: sesiones, caché de curso, cola de notificaciones, ranking de
---
1. Modelo conceptual Entidades principales:
`students` (estudiantes)
`professors` (profesores)
`courses` (materias/ofertas)
`enrollments` (matrículas: relación estudiante-curso por semestre)
`grades` (calificaciones por evaluación)
`academic_records` (historial resumido, opcional)
Decisiones: usar `enrollments` como colección separada (para consultas eficientes por estudiante y por
---
2. MongoDB — Esquemas de ejemplo (documentos) ### Base de datos: `academico`
Colección: `students`
```js
{
_id: ObjectId("651f1a2b3c4d5e6f7890abcd"), studentId: "S-2024001",
name: { first: "Laura", last: "Gómez" }, email: "laura.gomez@uni.edu",
dob: ISODate("2002-05-14"),
contacts: { phone: "+57-300-1112222", address: { street: "Calle 10", city: "Bogotá" } },
enrolledAt: ISODate("2024-02-01T08:00:00Z"),
metadata: { scholarship: true, creditsCompleted: 45 }
}
```
> Ejemplo de **campo anidado** (`name`, `contacts.address`) y **subdocumentos** (`metadata`). ### Colección: `professors`

```js
{
_id: ObjectId(), professorId: "P-1001",
name: { first: "Carlos", last: "Ramírez" }, departments: ["Matemáticas", "Estadística"], email:
"c.ramirez@uni.edu",
hireDate: ISODate(),
office: { building: "B", room: "204" }
}
```
Colección: `courses`

```js
{
_id: ObjectId(), courseCode: "MATH101", title: "Cálculo I",
description: "Introducción al cálculo diferencial e integral.", credits: 4,
department: "Matemáticas",
professors: [ ObjectId("..."), ObjectId("...") ], // referencias schedule: [
{ day: "Lunes", start: "08:00", end: "10:00", room: "A101" },
{ day: "Miércoles", start: "08:00", end: "10:00", room: "A101" }
],
createdAt: ISODate()
}
```
> `schedule` es un **arreglo de subdocumentos**. ### Colección: `enrollments`
```js
{
_id: ObjectId(),
studentId: ObjectId("651f1a2b3c4d5e6f7890abcd"), courseId: ObjectId("..."),
term: "2025-01", // semestre o cuatrimestre status: "active", // active, dropped, completed
grades: [
{ name: "Parcial 1", weight: 0.25, score: 85, date: ISODate() },
{ name: "Parcial 2", weight: 0.25, score: 78, date: ISODate() },
{ name: "Final", weight: 0.5, score: 90, date: ISODate() }
],
createdAt: ISODate(), updatedAt: ISODate()
}
```
> `grades` es un **arreglo de subdocumentos** que permite calcular promedios ponderados por matrícula.

---
3. MongoDB — Operaciones CRUD (mongosh) ### Conectar
```bash
mongosh "mongodb://localhost:27017/academico"
```

Insertar (Create)
```js db.students.insertOne({
studentId: "S-2024002",
name: { first: "Juan", last: "Pérez" }, email: "juan.perez@uni.edu",
dob: ISODate("2001-03-10"),
enrolledAt: new Date(),
metadata: { scholarship: false, creditsCompleted: 12 }
});
```
Consultar (Read)
```js
// Buscar estudiante por studentId db.students.findOne({ studentId: "S-2024002" });
// Cursos de un profesor
db.courses.find({ professors: ObjectId("...") });
// Matriculas activas de un estudiante
db.enrollments.find({ studentId: ObjectId("..."), term: "2025-01", status: "active" });
```
Actualizar (Update)

```js
// Añadir una nueva calificación a una matrícula db.enrollments.updateOne(
{ _id: ObjectId("...") },
{ $push: { grades: { name: "Tarea 3", weight: 0.1, score: 92, date: new Date() } }, $set: {
updatedAt:
);
// Actualizar créditos completados
db.students.updateOne({ studentId: "S-2024002" }, { $inc: { "metadata.creditsCompleted": 3
} });
```
Eliminar (Delete)
```js
// Eliminar matrícula de prueba
db.enrollments.deleteOne({ _id: ObjectId("..."), status: "test" });
```
---
4. MongoDB — Aggregation: GPA / promedio ponderado por estudiante
```js db.enrollments.aggregate([
{ $match: { studentId: ObjectId("..."), status: "completed" } },
{ $unwind: "$grades" },
{ $group: {
_id: "$_id",
weightedSum: { $sum: { $multiply: ["$grades.score", "$grades.weight"] } }
}
},
```

```
{ $group: {
_id: null,
gpa: { $avg: "$weightedSum" }
}
}
]);
```
---
5. Redis — Modelado y ejemplos
Usos sugeridos en Redis: sesiones, cache de detalles de curso, cola de notificaciones, ranking de estudi ### 5.1 Sesiones — String/Hash con TTL
```bash
# Guardar sesión (string JSON) con TTL 3600 segundos (1 hora)
SETEX session:sess123 3600 "{\"userId\":\"S-2024002\",\"role\":\"student\"}"
# O usar hash
HSET session:sess123 userId S-2024002 role student EXPIRE session:sess123 3600
```
5.2 Cache de curso — Hash + TTL
```bash
HSET course:cache:MATH101 courseCode MATH101 title "Cálculo I" credits 4 professors "P-1001,P-1002" EXPIRE course:cache:MATH101 3600
HGETALL course:cache:MATH101
```
5.3 Cola de notificaciones — List
```bash
LPUSH queue:notifications "{\"type\":\"grade_posted\",\"studentId\":\"S-2024002\",\"enrollmentId\":\"E-. BRPOP queue:notifications 0
```
5.4 Ranking de estudiantes — Sorted Set
```bash
# Score: promedio o puntos
ZINCRBY ranking:students 0 "S-2024002" # inicializa en 0 ZADD ranking:students 92 "S-2024002"   # establecer score

ZREVRANGE ranking:students 0 9 WITHSCORES
```
5.5 Usuarios en línea — Set
```bash
SADD online:users S-2024002 SISMEMBER online:users S-2024002 SREM online:users S-2024002
```
5.6 Tokens con TTL (reset password) — String + TTL
```bash
```

SETEX token:reset:abc123 600 "S-2024002" # token válido 10 minutos GET token:reset:abc123
```
---
## 6. Flujo típico
Un estudiante inicia sesión: crear `session:{id}` en Redis con TTL (rápido) y registrar en `online:use
Lectura de curso en la web: intentar `HGETALL course:cache:{code}` -> si no existe, leer de Mongo y gu
Publicación de calificaciones: el servicio que añade la nota en Mongo `db.enrollments.updateOne(...)`
Recuperación de contraseña: generar token, `SETEX token:reset:{token} {studentId} 600` y enviar por em
---
## 7. Ejemplos en Node.js (breve) Dependencias: `npm i mongodb ioredis`
```js
const { MongoClient } = require('mongodb'); const Redis = require('ioredis');
const redis = new Redis();
const client = new MongoClient('mongodb://localhost:27017'); await client.connect();
const db = client.db('academico');
async function getCourse(code) { const key = `course:cache:${code}`;
const cached = await redis.hgetall(key);
if (Object.keys(cached).length) return cached;
const course = await db.collection('courses').findOne({ courseCode: code }); if (!course) return null;
await redis.hset(key, 'courseCode', course.courseCode, 'title', course.title, 'credits', String(course await redis.expire(key, 3600);
return course;
}
```
---
8. Docker-compose y seed (incluidos en el `.zip`)\n `docker-compose.yml` para Mongo + Redis\n- `seed_academico.js` script que inserta datos de ejemplo en

docker-compose.yml
Código utilizado:
version: '3.8' services:
mongo:
image: mongo:6.0 restart: unless-stopped ports:
- "27017:27017"
volumes:
- mongo_data:/data/db
```

redis:
image: redis:7
restart: unless-stopped ports:
- "6379:6379"
volumes: mongo_data:

seed_academico.js
Código utilizado:

```
/*
Seed script: sistema academico Requires: npm i mongodb ioredis Run: node
seed_academico.js
*/
const { MongoClient } = require('mongodb'); const Redis = require('ioredis');
(async () => {
const client = new MongoClient('mongodb://localhost:27017'); await client.connect();
const db = client.db('academico');
// insert sample students const students = [
{ studentId: 'S-2024001', name: { first: 'Laura', last: 'Gomez' }, email: 'laura.gomez@uni.edu',
enr
{ studentId: 'S-2024002', name: { first: 'Juan', last: 'Perez' }, email: 'juan.perez@uni.edu',
enrol
];
await db.collection('students').insertMany(students);
// insert courses const courses = [
{ courseCode: 'MATH101', title: 'Cálculo I', credits: 4, department: 'Matemáticas', schedule:
[{ day
{ courseCode: 'CS102', title: 'Programación I', credits: 3, department: 'Ciencias de la
Computación'
];
await db.collection('courses').insertMany(courses);
// enrollments
const enrollments = [
{ studentId: (await db.collection('students').findOne({ studentId: 'S-2024001' }))._id,
courseId: (a
{ studentId: (await db.collection('students').findOne({ studentId: 'S-2024002' }))._id,
courseId: (a
];
await db.collection('enrollments').insertMany(enrollments);
// seed redis cache and ranking const redis = new Redis();
// cache course
await redis.hset('course:cache:MATH101', 'courseCode', 'MATH101', 'title', 'Cálculo I',
'credits', '4' await redis.expire('course:cache:MATH101', 3600);
// ranking
```

```
await redis.zadd('ranking:students', 0, 'S-2024001'); await redis.zadd('ranking:students', 0,
'S-2024002');
console.log('Seed académico completo'); process.exit(0);
})();
```