Planificador de Materias – Programación 3 (UADE)

API reactiva en Spring Boot + Neo4j para planificar cursadas usando grafos y varios paradigmas algorítmicos: DFS/BFS, Topological Sort (Kahn), Dijkstra, Prim/Kruskal (MST), Greedy, Programación Dinámica (Knapsack), Backtracking y Branch & Bound.

## Stack

• Java 17+

• Spring Boot 3 (WebFlux, Data Neo4j Reactivo)

• Neo4j 5 (Bolt)

• Build: Maven

## Cómo correr

1) Levantar Neo4j Desktop → DBMS ACTIVE (bolt en localhost:7687).

2) Crear/confirmar usuario neo4j y contraseña.

3) Ajustar src/main/resources/application.properties:

spring.neo4j.uri=bolt://localhost:7687  
spring.neo4j.authentication.username=neo4j  
spring.neo4j.authentication.password=TU\_PASS  
spring.data.neo4j.database=neo4j  
spring.neo4j.security.encrypted=false  
server.port=8080  
spring.main.web-application-type=reactive

4) Cargar seed (ver abajo) en Neo4j Browser.

5) Correr la app:

mvn spring-boot:run

## Seed de datos (Cypher)

CREATE CONSTRAINT course\_code\_unique IF NOT EXISTS  
FOR (c:Course) REQUIRE c.code IS UNIQUE;  
  
MERGE (ayp:Course {code:'AYP', name:'Algoritmos y Programación', credits:6, hours:6, difficulty:2})  
MERGE (poo:Course {code:'POO', name:'Programación OO', credits:6, hours:6, difficulty:3})  
MERGE (bd: Course {code:'BD', name:'Bases de Datos', credits:6, hours:6, difficulty:3})  
MERGE (p3: Course {code:'PROG3', name:'Programación 3', credits:6, hours:6, difficulty:4})  
MERGE (ay2:Course {code:'AYED2', name:'Algoritmos y Estructuras 2',credits:6, hours:6, difficulty:4})  
MERGE (so: Course {code:'SO', name:'Sistemas Operativos', credits:6, hours:6, difficulty:3})  
MERGE (red:Course {code:'RED', name:'Redes', credits:6, hours:4, difficulty:3})  
MERGE (ia: Course {code:'IA', name:'Inteligencia Artificial', credits:6, hours:6, difficulty:5})  
  
MERGE (poo)-[:REQUIRES]->(ayp)  
MERGE (p3)-[:REQUIRES]->(poo)  
MERGE (p3)-[:REQUIRES]->(bd)  
MERGE (ay2)-[:REQUIRES]->(ayp)  
MERGE (so)-[:REQUIRES]->(poo)  
MERGE (ia)-[:REQUIRES]->(ay2)  
MERGE (ia)-[:REQUIRES]->(bd)  
MERGE (red)-[:REQUIRES]->(so);  
  
// Subgrafo no-dirigido RELATED (para MST)  
MERGE (poo)-[:RELATED {sim:1.0}]->(ayp)  
MERGE (p3)-[:RELATED {sim:0.9}]->(poo)  
MERGE (p3)-[:RELATED {sim:0.6}]->(bd)  
MERGE (ay2)-[:RELATED {sim:0.8}]->(ayp)  
MERGE (ia)-[:RELATED {sim:0.7}]->(ay2)  
MERGE (so)-[:RELATED {sim:0.5}]->(poo)  
MERGE (red)-[:RELATED {sim:0.4}]->(so);

## Dominio

• Nodo: Course { code, name, credits, hours, difficulty }

• Relación dirigida: (:Course {A})-[:REQUIRES]->(:Course {B}) ⇒ A requiere B.

• Relación no dirigida (solo para MST): (:Course)-[:RELATED {sim:double}]-(:Course) ⇒ similitud temática (peso = 1/sim).

## Estructura (resumen)

src/main/java/...  
 model/Course.java  
 repo/CourseRepository.java  
 service/GraphService.java # DFS/BFS, Kahn, ciclos, Dijkstra, MST  
 service/ScheduleService.java # Greedy, DP, Backtracking, Branch & Bound  
 controller/CourseController.java # CRUD básico  
 controller/GraphController.java # Endpoints de grafos  
 controller/ScheduleController.java # Endpoints de planificación  
 controller/dto/MstEdgeDTO.java

## Endpoints

### Cursos

• GET /courses → lista cursos

• GET /courses/{code} → un curso

• PUT /courses → upsert de curso (se puede incluir prereqs embebidos)

Ejemplo body (upsert):

{  
 "code": "PROG3",  
 "name": "Programación 3",  
 "credits": 6,  
 "hours": 6,  
 "difficulty": 4,  
 "prereqs": [  
 {"code":"POO"},  
 {"code":"BD"}  
 ]  
}

### Grafos

• GET /graph/dfs?from=PROG3

• GET /graph/bfs-layers?from=AYP

• GET /graph/toposort?approved=AYP,BD

• GET /graph/cycles

• GET /graph/shortest?from=PROG3&to=AYP&metric=difficulty

• GET /graph/mst?algo=prim|kruskal

### Planificación

• GET /schedule/available?approved=AYP,BD

• GET /schedule/greedy?approved=AYP,BD&value=credits&maxHours=12

• GET /schedule/dp?approved=AYP,BD&value=credits&maxHours=12

• GET /schedule/backtracking?from=PROG3&to=AYP&maxDepth=6

• GET /schedule/bnb?approved=AYP,BD&semesters=4&maxHours=24

## Pruebas rápidas (curl)

curl http://localhost:8080/courses  
curl "http://localhost:8080/schedule/available?approved=AYP,BD"  
curl "http://localhost:8080/graph/toposort?approved=AYP,BD"  
curl "http://localhost:8080/graph/shortest?from=PROG3&to=AYP&metric=difficulty"  
curl "http://localhost:8080/schedule/greedy?approved=AYP,BD&value=credits&maxHours=12"  
curl "http://localhost:8080/schedule/dp?approved=AYP,BD&value=credits&maxHours=12"  
curl http://localhost:8080/graph/cycles  
curl "http://localhost:8080/graph/mst?algo=prim"

## Complejidad

|  |  |
| --- | --- |
| Algoritmo / Endpoint | Complejidad |
| DFS / BFS (/graph/dfs, /graph/bfs-layers) | O(V+E) |
| Toposort Kahn (/graph/toposort) | O(V+E) |
| Detección de ciclos (/graph/cycles) | O(V+E) |
| Dijkstra (/graph/shortest) | O((V+E) log V) |
| Prim/Kruskal (/graph/mst) | O(E log V) |
| Greedy (/schedule/greedy) | O(n log n) |
| DP Knapsack (/schedule/dp) | O(n · capacidad) |
| Backtracking (/schedule/backtracking) | Exponencial (poda) |
| Branch & Bound (/schedule/bnb) | Exponencial (poda) |

Notas: V = #materias, E = #correlativas.