INSTITUTIO SUPERIOR SUDAMERICANO

Autores:

Christian Rojas

ASIGNATURA:

Despliegue de

diagramas

CURSO:

Tercer ciclo de software

DOCENTE ENCARGADO:

Ing.Vacacela Jhostin

2025-2026

Introducción

En un sistema de gestión universitaria es muy importante que los datos se manejen correctamente, ya que se trabaja con información sensible como estudiantes, materias y matrículas.
Para garantizar que estos datos no se dañen o se pierdan, se aplican los principios **ACID**, los cuales permiten que las operaciones importantes se realicen de forma segura.

En este proyecto, los principios ACID se aplican principalmente en el proceso de **matriculación de estudiantes**.

Atomicidad

La **atomicidad** significa que una operación se ejecuta **completa o no se ejecuta nada**.

En la matriculación del estudiante, el sistema realiza varias acciones:

- Verifica que el estudiante esté activo
- Revisa si la materia tiene cupos
- Registra la matrícula
- Descuenta el cupo disponible

Si alguna de estas acciones falla, **todo se cancela automáticamente**.
Esto evita que existan matrículas incompletas o errores en los cupos.

Consistencia

La **consistencia** asegura que los datos siempre estén correctos y cumplan las reglas del sistema.

En el proyecto:

- Un estudiante inactivo no puede matricularse
- Una materia no puede tener cupos negativos
- No se pueden registrar datos que no existan

Gracias a estas reglas, la base de datos siempre pasa de un estado correcto a otro estado correcto.

Aislamiento

El **aislamiento** evita que dos procesos interfieran entre sí.

Por ejemplo, si varios estudiantes intentan matricularse al mismo tiempo en una materia, el sistema:

- Controla los cupos de forma segura
- Evita que dos estudiantes ocupen el mismo cupo
- Mantiene la información ordenada

Esto es muy importante cuando el sistema tiene muchos usuarios al mismo tiempo.

Durabilidad

La **durabilidad** significa que los datos se guardan de forma permanente.

Cuando una matrícula se registra correctamente:

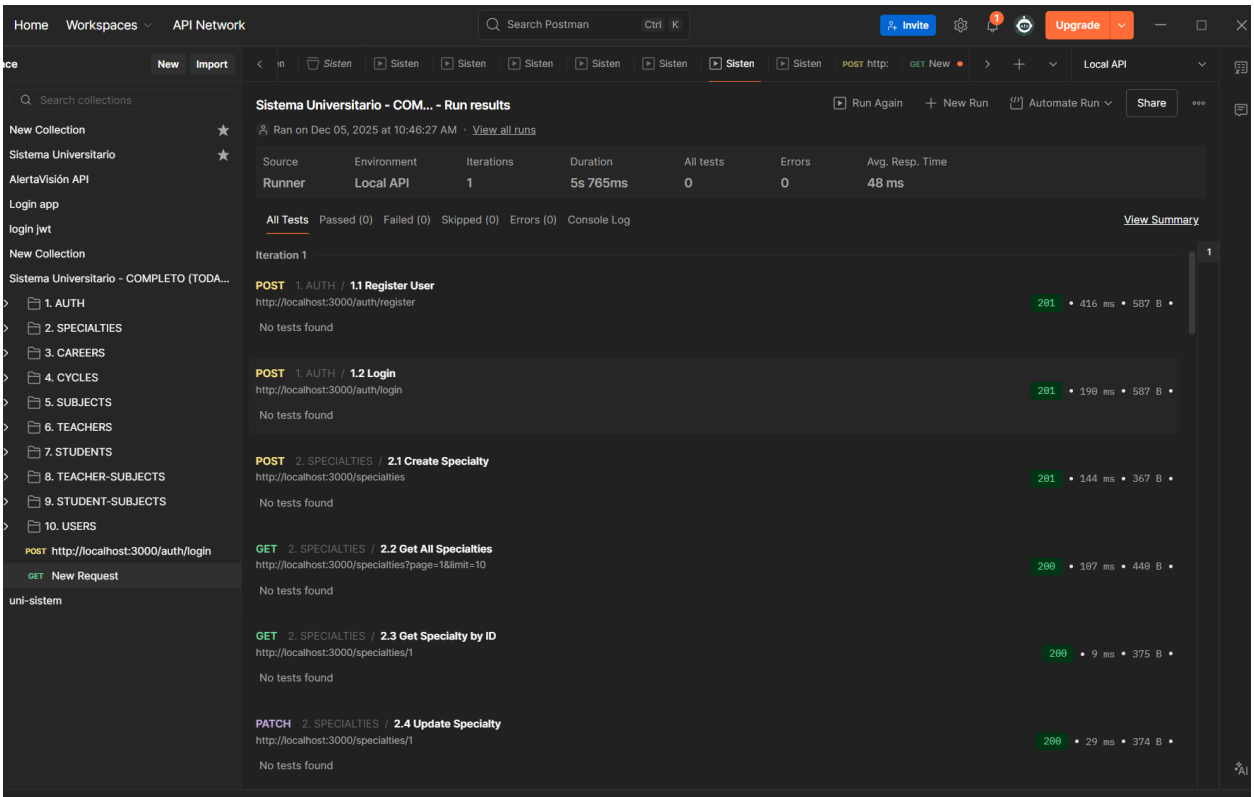
- La información queda guardada en la base de datos
- No se pierde aunque el servidor se apague
- El historial académico del estudiante se conserva

Esto garantiza la seguridad de la información académica.

Conclusión

Los principios ACID permiten que el sistema universitario funcione de manera confiable y segura.
Gracias a su implementación, el proceso de matriculación evita errores, mantiene la información correcta y protege los datos del sistema.

El uso de NestJS, Prisma y PostgreSQL facilita la aplicación de estos principios y permite desarrollar un sistema sólido y bien estructurado.



All Tests Passed (0) Failed (0) Skipped (0) Errors (0) Console Log View Summar

POST 3. CAREERS / **3.1 Create Career**
http://localhost:3000/careers
201 • 30 ms • 563 B •
No tests found

GET 3. CAREERS / **3.2 Get All Careers**
http://localhost:3000/careers?page=1&limit=10
200 • 7 ms • 623 B •
No tests found

GET 3. CAREERS / **3.3 Get Career by ID**
http://localhost:3000/careers/1
200 • 112 ms • 586 B •
No tests found

PATCH 3. CAREERS / **3.4 Update Career**
http://localhost:3000/careers/1
200 • 39 ms • 570 B •
No tests found

POST 4. CYCLES / **4.1 Create Cycle**
http://localhost:3000/cycles
201 • 9 ms • 360 B •
No tests found

GET 4. CYCLES / **4.2 Get All Cycles**
http://localhost:3000/cycles?page=1&limit=10
200 • 10 ms • 434 B •
No tests found

GET 4. CYCLES / **4.3 Get Cycle by ID**
http://localhost:3000/cycles/1
200 • 6 ms • 369 B •

No tests found

PATCH 4. CYCLES / **4.4 Update Cycle**
http://localhost:3000/cycles/1
200 • 18 ms • 367 B •

No tests found

POST 5. SUBJECTS / **5.1 Create Subject**
http://localhost:3000/subjects
201 • 32 ms • 722 B •

No tests found

GET 5. SUBJECTS / **5.2 Get All Subjects**
http://localhost:3000/subjects?page=1&limit=10
200 • 9 ms • 782 B •

No tests found

GET 5. SUBJECTS / **5.3 Get Subject by ID**
http://localhost:3000/subjects/1
200 • 126 ms • 745 B •

No tests found

PATCH 5. SUBJECTS / **5.4 Update Subject**
http://localhost:3000/subjects/1
200 • 45 ms • 729 B •

No tests found

http://localhost:3000/teachers/1
200 • 30 ms • 453 B •

No tests found

POST 7. STUDENTS / **7.1 Create Student**
http://localhost:3000/students
201 • 25 ms • 800 B •

No tests found

GET 7. STUDENTS / **7.2 Get All Students**
http://localhost:3000/students?page=1&limit=10
200 • 10 ms • 860 B •

No tests found

GET 7. STUDENTS / **7.3 Get Student by ID**
http://localhost:3000/students/1
200 • 10 ms • 809 B •

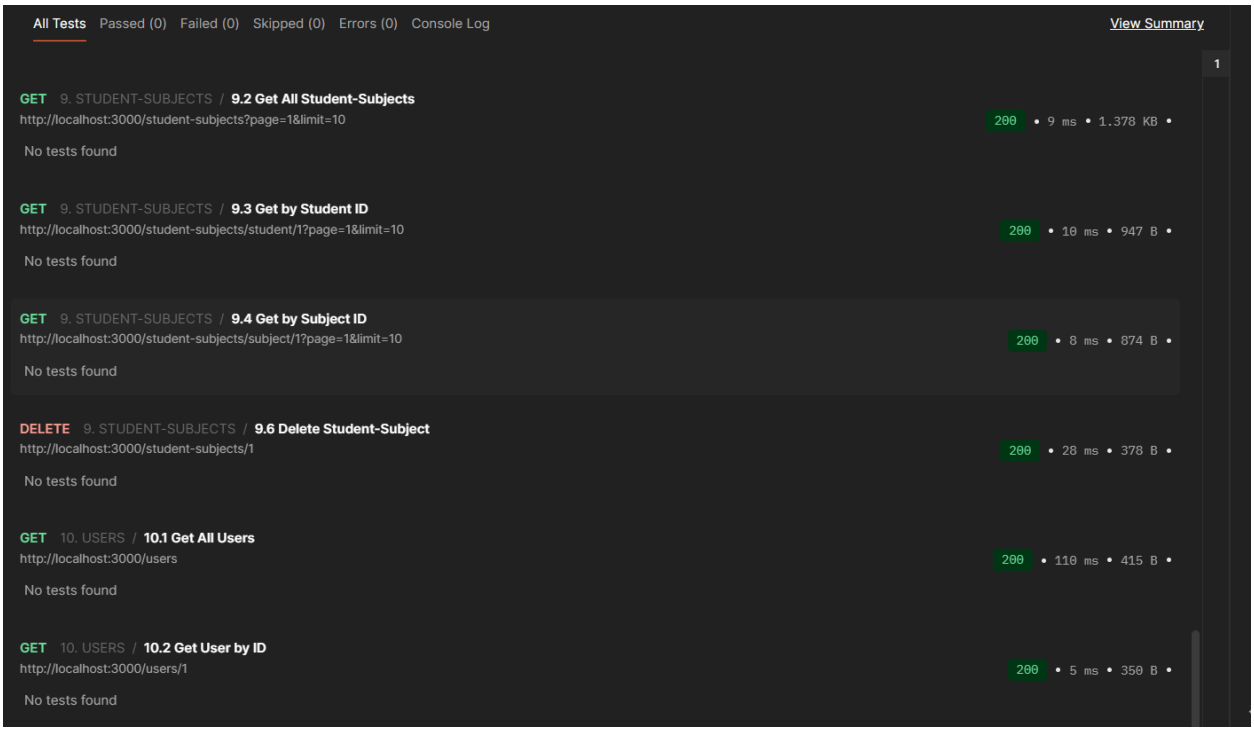
No tests found

PATCH 7. STUDENTS / **7.4 Update Student**
http://localhost:3000/students/1
200 • 39 ms • 807 B •

No tests found

POST 8. TEACHER-SUBJECTS / **8.1 Assign Teacher to Subject**
http://localhost:3000/teacher-subjects
201 • 38 ms • 1.199 KB •

No tests found


All Tests Passed (0) Failed (0) Skipped (0) Errors (0) Console Log
View Summary
1
GET 9. STUDENT-SUBJECTS / 9.2 Get All Student-Subjects
http://localhost:3000/student-subjects?page=1&limit=10
200 • 9 ms • 1.378 KB •
No tests found
GET 9. STUDENT-SUBJECTS / 9.3 Get by Student ID
http://localhost:3000/student-subjects/student/1?page=1&limit=10
200 • 10 ms • 947 B •
No tests found
GET 9. STUDENT-SUBJECTS / 9.4 Get by Subject ID
http://localhost:3000/student-subjects/subject/1?page=1&limit=10
200 • 8 ms • 874 B •
No tests found
DELETE 9. STUDENT-SUBJECTS / 9.6 Delete Student-Subject
http://localhost:3000/student-subjects/1
200 • 28 ms • 378 B •
No tests found
GET 10. USERS / 10.1 Get All Users
http://localhost:3000/users
200 • 110 ms • 415 B •
No tests found
GET 10. USERS / 10.2 Get User by ID
http://localhost:3000/users/1
200 • 5 ms • 350 B •
No tests found