Lab 5 - SDT

Student: **FINICHIU Eduard - Adelin**
FILS 1241 EB

In this document I wrote the phases of the Lab the required written explanations.

Phase 1: Architecture Design

The requirement is to have 4 services interacting via REST. To get the "bonus points" for using different databases, we will mix SQL and NoSQL.

1. The Database Strategy (including the strategy for the bonus points)

- Grading Service: H2 Database (In-memory, fast, good for demonstrating variety).
- Student Service: PostgreSQL
- Course Service: PostgreSQL
- Professor Service: PostgreSQL

2. Service APIs & Interactions

A. Student Service (Port 8081)

- Responsibility: Manage student PII (Personally Identifiable Information).
- Endpoints:
    - POST /students (Register a student)
    - GET /students/{id} (View personal data)
    - PUT /students/{id} (Update personal data)
    - GET /students/{id}/report-card -> Interaction: This endpoint calls the Grading Service to fetch grades for this student and aggregates them with student details.

B. Course Service (Port 8082)

- Responsibility: The "source of truth" for what courses exist.
- Endpoints:
    - POST /courses (Admin adds a course)
    - GET /courses/{id} (Get course description)
    - PUT /courses/{id} (Update description)

C. Professor Service (Port 8083)

- Responsibility: Manage professors and their assignments.
- Endpoints:
    - POST /professors (Add professor)
    - PUT /professors/{id}/assign-course/{courseId} -> Interaction: Before assigning, this service calls Course Service GET /courses/{courseId} to ensure the course actually exists.

D. Grading Service (Port 8084)

- Responsibility: Store grades.
- Endpoints:
    - POST /grades (Add a grade for a studentId and courseId).
    - GET /grades/student/{studentId} (Get all grades for a specific student).

    - Interaction: When adding a grade, it could call Student Service to verify the student exists (optional validation, but good practice).

3. Summary of Interactions (Synchronous REST)

1. Student Service calls Grading Service (to get the full report card).
2. Professor Service calls Course Service (to validate course assignment).

Phase 2: Architecture Evaluation & Patterns

1. Evaluation of the Architecture

Advantages:

- Polyglot Persistence: Since we are using different databases (each service uses the best tool for the job.
- Independent Scaling: If the "Grading" service gets heavy traffic during exam season, we can spin up 5 instances of just the Grading Service without touching the Professor Service.
- Fault Isolation: If the "Course Management" service crashes, students can still log in and view their personal details via the "Student Service" (though they might not see course names).

Disadvantages & Pitfalls:

- Latency: A function call is nanoseconds; a REST call is milliseconds. The Student -> Grading call introduces network lag.
- Distributed Transactions: We lost ACID transactions. If we delete a student in the Student Service, but the HTTP call to delete their grades in the Grading Service fails, we have "zombie data" (consistency issues).
- Complexity: We now have 4 apps to deploy and monitor instead of 1.

2. Microservices Patterns to Discuss

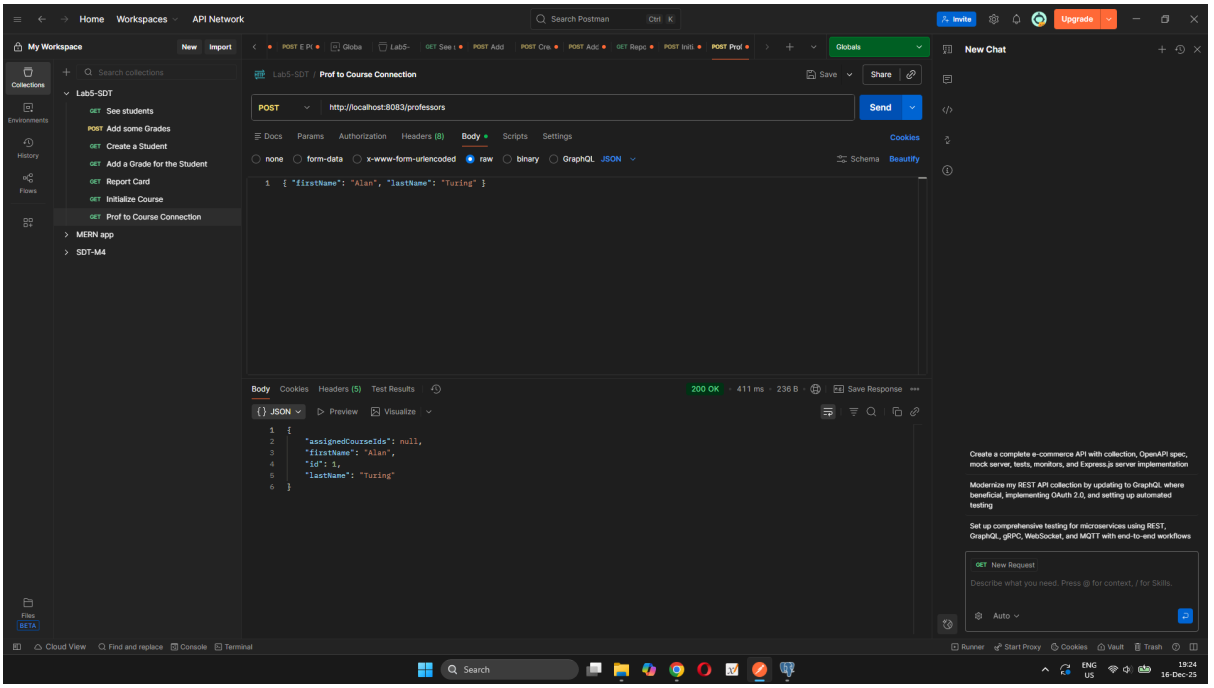
- Database per Service Pattern: (We are implementing this). Each service has its own DB. Other services cannot touch that DB directly; they must go through the API.
- API Composition Pattern: The Student Service acts as a composer. When you ask for a "Report Card", it fetches its own data AND makes a call to the Grading Service, combines the results in memory, and returns the aggregate.
- Circuit Breaker (Future): Since we rely on synchronous REST calls, if the Grading Service hangs, the Student Service might hang waiting for it. A Circuit Breaker (like Resilience4j) would prevent this cascade of failure.

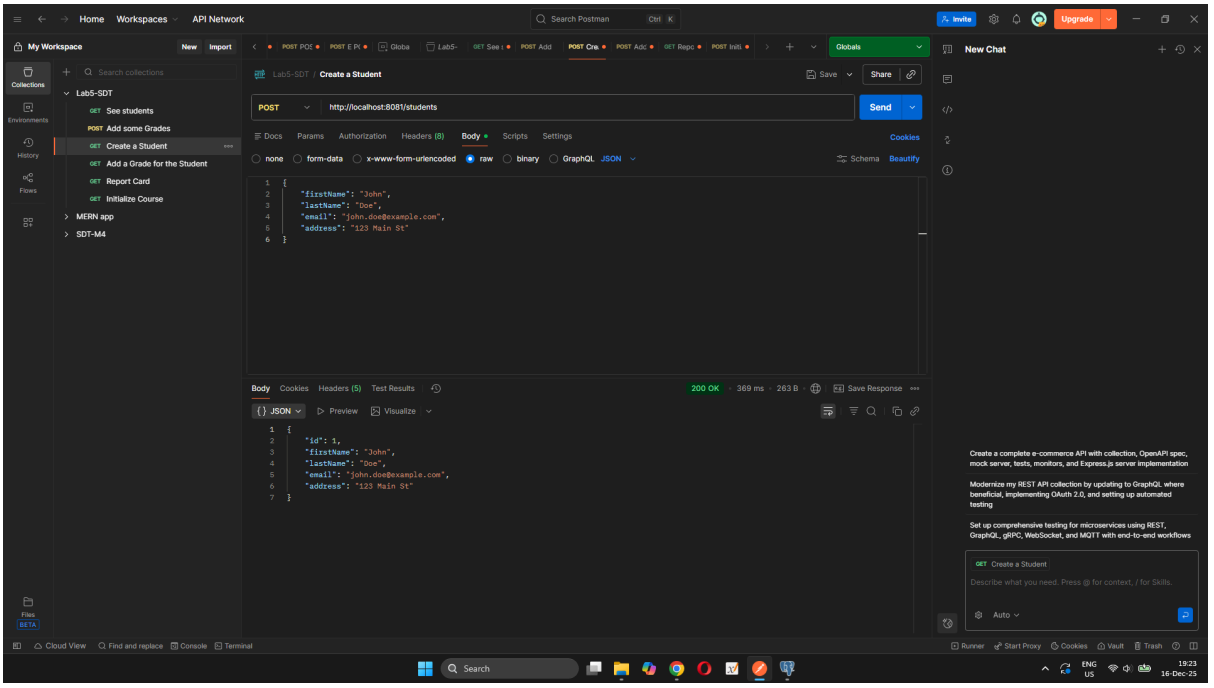
Phase 3: Implementation Strategy

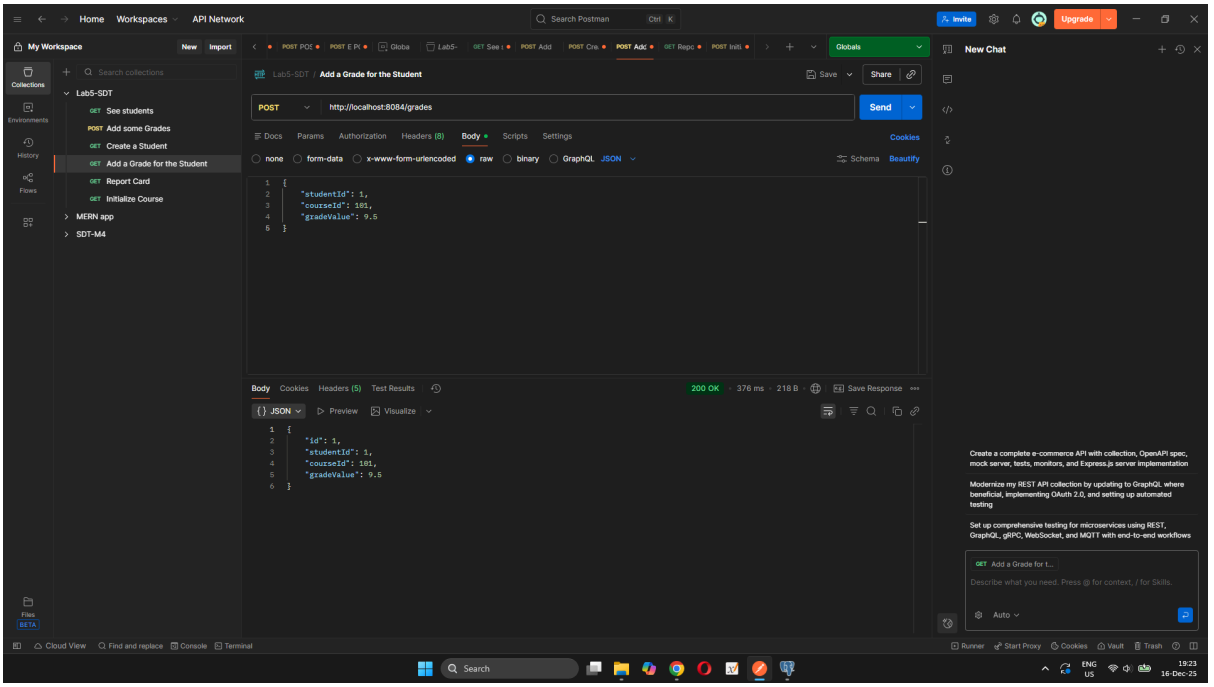
I will start with the Student Service. It is the most central one, uses a standard SQL database (PostgreSQL), and allows us to demonstrate the "API Composition" pattern later by calling the Grading service. We will use:

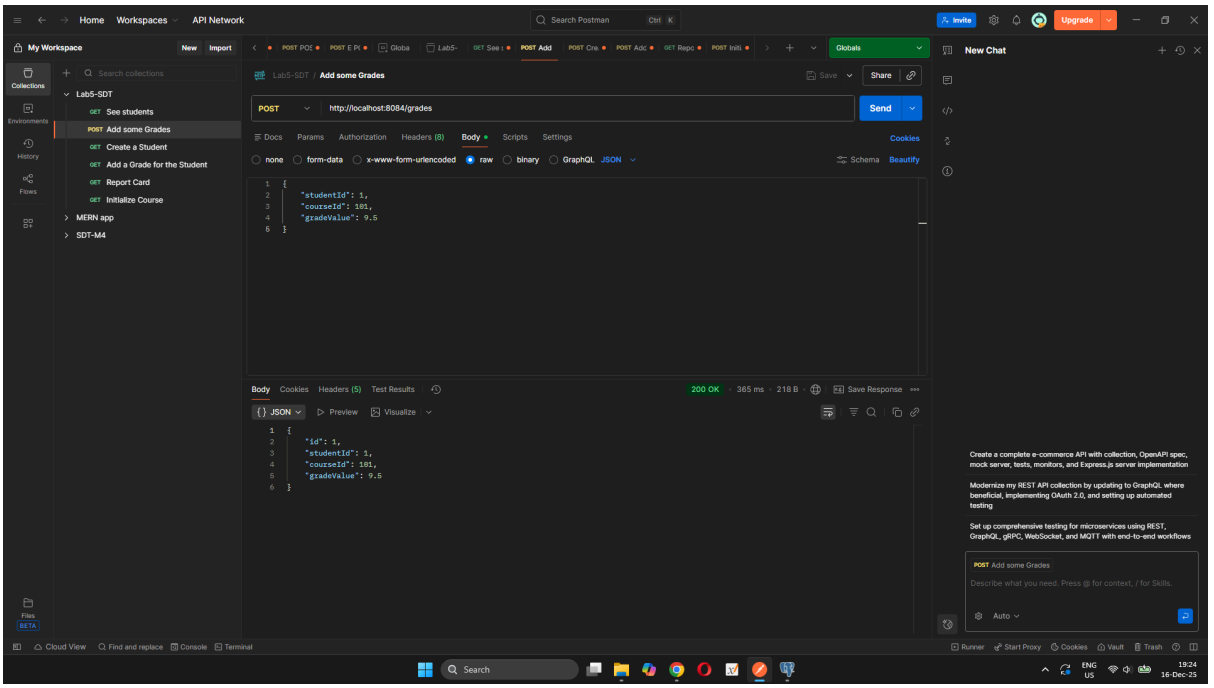
- Language: Java 17+
- Framework: Spring Boot 3.x
- Dependencies: Spring Web, Spring Data JPA, PostgreSQL Driver, Lombok.

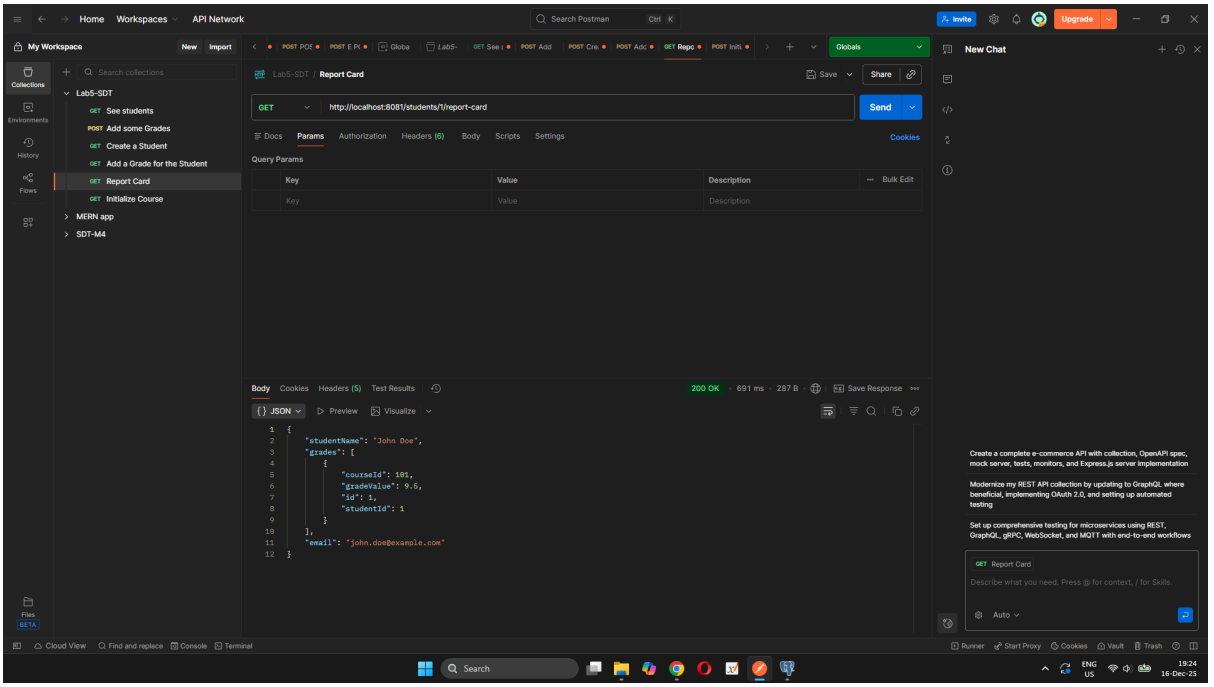
Screenshots

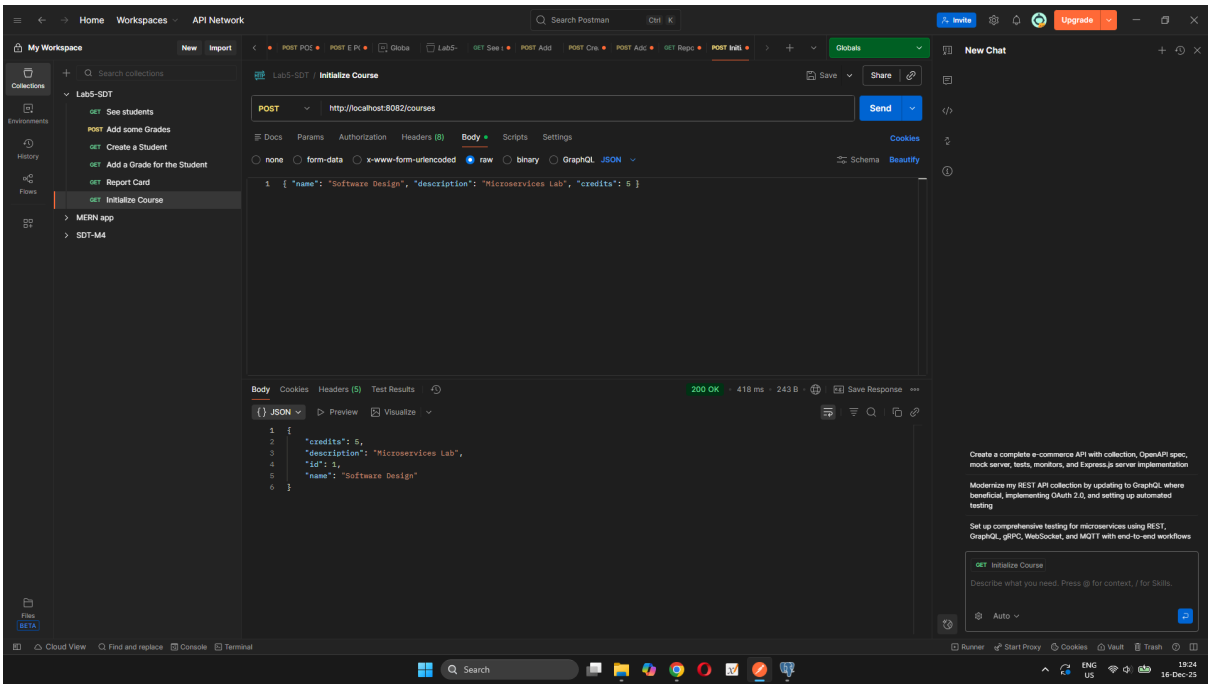





Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
History
Flows
Files
BETA
Lab5-SDT
See students
Add some Grades
Create a Student
Add a Grade for the Student
Report Card
Initialize Course
MERN app
SDT-M4
Lab5-SDT / Add a Grade for the Student
Save
Share
POST
http://localhost:8084/grades
Send
Docs
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"studentId": 1,
"courseId": 101,
"gradeValue": 9.5
}
Body
Cookies
Headers (5)
Test Results
200 OK
376 ms
218 B
Save Response
JSON
Preview
Visualize
{
"id": 1,
"studentId": 1,
"courseId": 101,
"gradeValue": 9.5
}
Globals
New Chat
Create a complete e-commerce API with collection, OpenAPI spec, mock server, tests, monitors, and Express.js server implementation
Modernize my REST API collection by updating to GraphQL where beneficial, implementing OAuth 2.0, and setting up automated testing
Set up comprehensive testing for microservices using REST, GraphQL, gRPC, WebSocket, and MQTT with end-to-end workflows
GET Add a Grade for t...
Describe what you need. Press @ for context, / for Skills.
Auto
Cloud View
Find and replace
Console
Terminal
Runner
Start Proxy
Cookies
Vault
Trash
Search
ENG
US
19:23
16-Dec-25


Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
History
Flows
Files
BETA
Lab5-SDT
See students
Add some Grades
Create a Student
Add a Grade for the Student
Report Card
Initialize Course
MERN app
SDT-M4
Lab5-SDT / Add some Grades
Save
Share
POST
http://localhost:8084/grades
Send
Docs
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"studentId": 1,
"courseId": 101,
"gradeValue": 9.5
}
Body
Cookies
Headers (5)
Test Results
200 OK
365 ms
218 B
Save Response
JSON
Preview
Visualize
{
"id": 1,
"studentId": 1,
"courseId": 101,
"gradeValue": 9.5
}
Globals
New Chat
Create a complete e-commerce API with collection, OpenAPI spec, mock server, tests, monitors, and Express.js server implementation
Modernize my REST API collection by updating to GraphQL where beneficial, implementing OAuth 2.0, and setting up automated testing
Set up comprehensive testing for microservices using REST, GraphQL, gRPC, WebSocket, and MQTT with end-to-end workflows
POST Add some Grades
Describe what you need. Press @ for context, / for Skills.
Auto
Cloud View
Find and replace
Console
Terminal
Runner
Start Proxy
Cookies
Vault
Trash
Search
ENG
US
19:24
16-Dec-25


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
History
Flows
Files
BETA
Lab5-SDT
See students
Add some Grades
Create a Student
Add a Grade for the Student
Report Card
Initialize Course
MERN app
SDT-M4
Lab5-SDT / Report Card
Save
Share
GET
http://localhost:8081/students/1/report-card
Send
Docs
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (5)
Test Results
200 OK
691 ms
287 B
Save Response
JSON
Preview
Visualize
{
"studentName": "John Doe",
"grades": [
{
"courseId": 101,
"gradeValue": 9.6,
"id": 1,
"studentId": 1
}
],
"email": "john.doe@example.com"
}
New Chat
Create a complete e-commerce API with collection, OpenAPI spec, mock server, tests, monitors, and Express.js server implementation
Modernize my REST API collection by updating to GraphQL where beneficial, implementing OAuth 2.0, and setting up automated testing
Set up comprehensive testing for microservices using REST, GraphQL, gRPC, WebSocket, and MQTT with end-to-end workflows
GET Report Card
Describe what you need. Press @ for context, / for Skills.
Auto
Cloud View
Find and replace
Console
Terminal
Runner
Start Proxy
Cookies
Vault
Trash
Search
ENG
US
19:24
16-Dec-25


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
History
Flows
Files
BETA
Lab5-SDT
See students
Add some Grades
Create a Student
Add a Grade for the Student
Report Card
Initialize Course
MERN app
SDT-M4
Lab5-SDT / Initialize Course
Save
Share
POST
http://localhost:8082/courses
Send
Docs
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{ "name": "Software Design", "description": "Microservices Lab", "credits": 5 }
Body
Cookies
Headers (5)
Test Results
200 OK
418 ms
243 B
Save Response
JSON
Preview
Visualize
{
"credits": 5,
"description": "Microservices Lab",
"id": 1,
"name": "Software Design"
}
New Chat
Create a complete e-commerce API with collection, OpenAPI spec, mock server, tests, monitors, and Express.js server implementation
Modernize my REST API collection by updating to GraphQL where beneficial, implementing OAuth 2.0, and setting up automated testing
Set up comprehensive testing for microservices using REST, GraphQL, gRPC, WebSocket, and MQTT with end-to-end workflows
GET Initialize Course
Describe what you need. Press @ for context, / for Skills.
Auto
Cloud View
Find and replace
Console
Terminal
Runner
Start Proxy
Cookies
Vault
Trash
Search
ENG
US
19:24
16-Dec-25