Understanding APIs and RESTful Architecture

1.2 REST Principles and Design

Learning objectives

- REST architecture basics
- Core principles
- Resource naming
- Idempotency and safety
- Design Lab at the end

What Is REST?

- Representational State Transfer
- Architectural style for web APIs
- Uses HTTP for CRUD operations
- Each resource identified by a URI
- Communicates using representations (JSON/XML)

Core REST Concepts

- Resources: /users, /orders
- URIs: Unique identifiers
- Uniform Interface: Standard HTTP verbs
- Statelessness: No session stored
- Cacheable: Responses can be cached
- Representation: Same resource, multiple formats

Good Resource Naming

- Use nouns, not verbs → /books, not /getBooks
- Use plural names for collections
- Keep URLs short, lowercase, and consistent
- Nest for relationships → /authors/{id}/books

Demo: What Is a “Resource”?

- In REST, everything revolves around resources
- A resource represents a thing in your system
- Example: a book, user, or order
- Each resource has attributes (data fields)
- REST APIs expose these resources in a consistent way
- Let’s explore a few examples conceptually

Demo Example: Library Domain

Resource	Attributes	Description
Book	id, title, author, available	Represents one book
Member	id, name, email	Library user
Loan	id, bookId, memberId, date	Borrowing record

Lab: Identify Your Resources

- Goal: Practice identifying key resources and their operations
  - 1. Choose a simple domain you're interested in, for example:
    - Tasks, Library, Orders, HR / Employees, Movies / Reviews
  - 2. Identify your main resources (nouns, not actions).
  - 3. For each resource, list main attributes (e.g., title, price, date).
  - 4. Then, for each resource, decide what basic operations might be needed:
    - Create/Read/Update/Delete (CRUD)
- (You don't need to write any code or URLs yet.)

Lab - Example Solution

Resource	Attributes	CRUD Operations
Task	id, title, description, completed	Create, Read, Update, Delete
User	id, name, email	Create, Read, Update, Delete
Project	id, name, deadline	Create, Read, Update, Delete

Lab - Reflection

- What resources did you identify?
- How similar were your CRUD operations?
- Did you notice patterns across different domains?
- Why do you think REST is centered around resources instead of actions?

Key Takeaways

- REST focuses on resources, not actions
- Each resource represents a thing in the system (e.g., book, user, order)
- Clients interact with resources using standard operations
- REST promotes clarity and consistency in API structure
- Thinking in nouns (resources) prepares you for designing endpoints
- Next: We'll see how HTTP methods implement these operations