Understanding APIs and RESTful Architecture

1.1 Introduction to APIs

Learning objectives

- What is an API
- Why APIs matter
- Client–server communication
- Overview of API types

What Is an API?

- API = Application Programming Interface
- Defines how software components communicate
- Acts as a contract between client and server
- Enables interoperability between systems
- Foundation of all modern web applications

Why APIs Matter

- Power mobile, web, and cloud applications
- Connect different services and systems
- Enable automation and integrations
- Drive microservices architectures
- Make data and functionality reusable

Client–Server Model

- Client sends requests, server returns responses
- Communication uses the HTTP protocol
- Each request is independent (stateless)
- Clients: browser, app, backend service
- Servers: host business logic & data
- Same idea across all modern web APIs

API Landscape Overview

- APIs come in different architectural styles
- Each with its own way of structuring communication
- Common styles:
  - REST
  - SOAP
  - GraphQL
- We'll explore these next

REST (Basic Overview)

- REpresentational State Transfer
- Resource-based approach using HTTP
- Uses verbs like GET, POST, PUT, DELETE
- Returns representations (often JSON)
- Most common style for modern web APIs
- (We'll go deeper into REST design in Module 1.2)

SOAP (Basic Overview)

- Simple Object Access Protocol
- Older, XML-based protocol
- Uses strict contracts (WSDL files)
- Heavyweight, verbose structure
- Still used in enterprise or legacy systems

GraphQL (Basic Overview)

- Query-based API developed by Facebook
- Client defines exactly what data to retrieve
- Uses a single endpoint for all queries
- Reduces over-fetching and under-fetching
- Growing in popularity, but adds complexity

API Styles Comparison

Feature	REST	SOAP	GraphQL
Data Format	JSON, XML	XML only	Custom query format
Flexibility	Medium	Low	High
Learning Curve	Easy	Moderate	Advanced
Typical Use	Web/Mobile APIs	Enterprise	Data-heavy frontends

- Focus for this course: REST APIs

Demo: What Does an API Response Look Like?

- APIs return data, not web pages
- You can view some APIs directly in your browser
- Each URL identifies a resource
- The browser shows the raw JSON returned
- Let’s look at a few examples together

Demo Example 1: JSONPlaceholder

- URL: https://jsonplaceholder.typicode.com/posts/1
- Returns data for a fake blog post
- Shows fields like userId, id, title, body
- Try changing the number → /posts/2, /posts/3
- Notice same structure, different content

Demo Example 2: Cat Facts API

- URL: https://catfact.ninja/fact
- Returns a random cat fact each time
- Example output: {"fact": "Cats sleep 70% of their lives", "length": 32}
- Refresh the page — new fact, same format
- Demonstrates dynamic API responses

Demo Example 3: Agify API

- URL: https://api.agify.io/?name=michael
- Predicts age based on a given name
- Uses query parameters (?name=michael)
- Try other names → ?name=emma, ?name=oliver
- Returns: {"name":"emma","age":32,"count":12904}

Lab 1: Explore Public APIs in Your Browser

- Goal: Understand how APIs expose resources and parameters.
  - 1. Open your web browser.
  - 2. Visit these URLs (one by one):
    - https://jsonplaceholder.typicode.com/posts/1
    - https://catfact.ninja/fact
    - https://api.agify.io/?name=michael
  - 3. Modify the URLs:
    - Change IDs (/posts/2, /posts/3)
    - Change parameters (?name=emma)
  - 4. Observe how the data changes.

Lab 2: Reflection

- Which part of the URL controls what data you get?
- How can a client (browser) “ask” for different information?
- What is the response format?
- Why doesn’t the API return an HTML page?
- What did you notice about consistency in the responses?

Key Takeaways

- APIs expose data in a structured format (JSON)
- The browser can make simple GET requests
- Changing the URL changes the resource or parameters
- API design affects how easily data can be accessed
- Next: we'll learn how REST defines these rules