REST APIs

Davide Borsatti

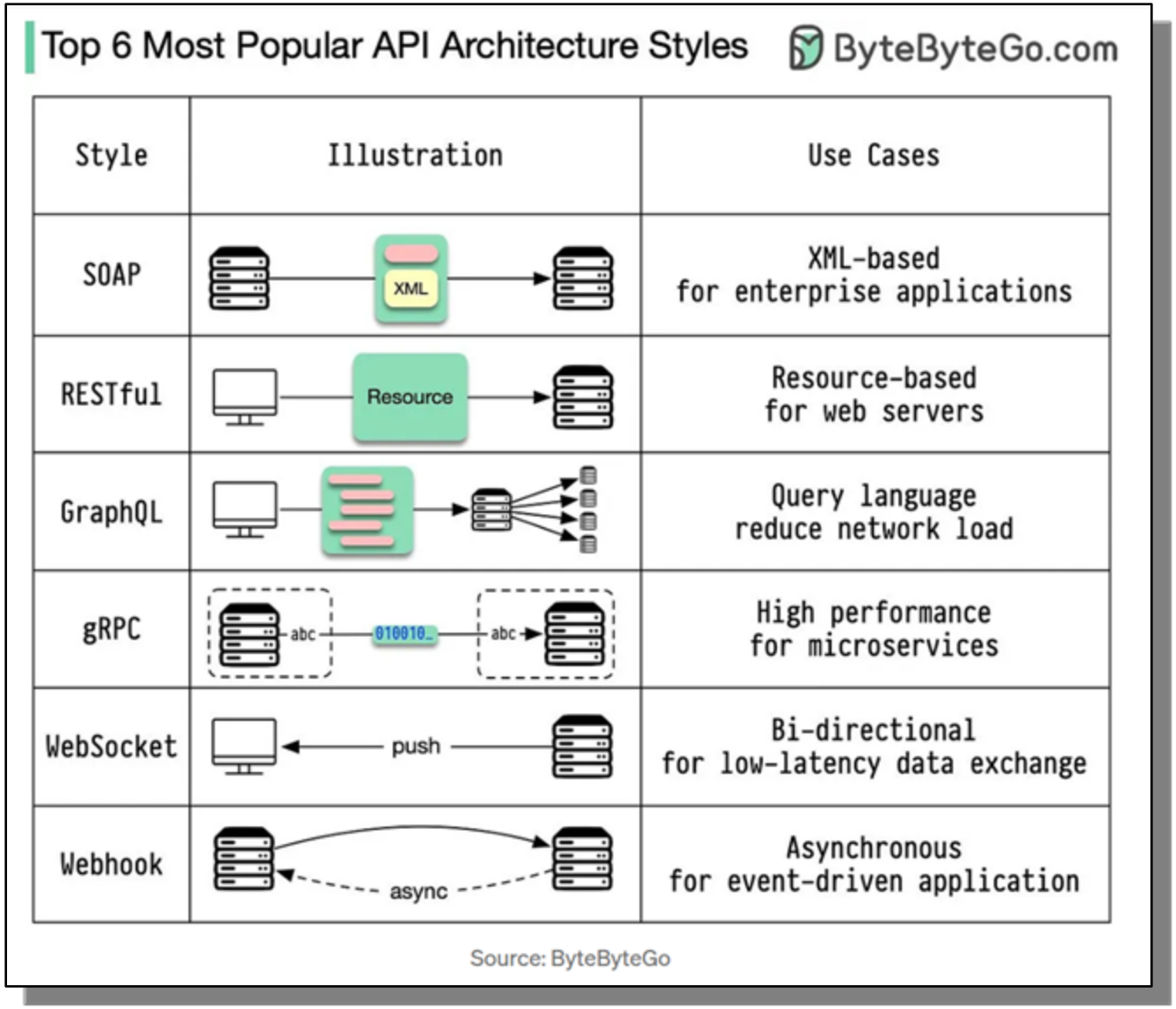
A.A. 2025-2026

Application Programming Interface (API)

- An **API** is a set of code-based instructions that enable separate software components to communicate and transfer data
- APIs are typically described through **API specifications** in terms of syntax and semantics
- API ≠ User Interface (UI)
  - APIs are not intended for interactions with generic human end-users
  - APIs are used by software programs (e.g.: web applications) or software programmers (e.g.: calls to libraries, OS system calls)

- APIs are widely used to enable communication and coordination between distributed system components

API architecture styles for distributed systems



Context: service-oriented software architectures

- **Resource-based** approach (typical for web applications):
  - A service is offered in the form of a **collection of resources**, individually managed by components
  - **RESTful** architecture (**Representational State Transfer** – REST) --> resources may be added, removed, retrieved, and modified by (remote) applications
  - Resources are identified through a single **naming scheme**
  - All services offer the same **interface**, typically consisting of the four **CRUD** operations (e.g. defined in terms of HTTP methods):
    - **C**reate a new resource (HTTP **POST**)
    - **R**etrieve the state of a resource in some representation format (e.g., JSON, XML) (HTTP **GET**)
    - **U**pdate/modify a resource by transferring a new state (HTTP **PUT**)
    - **D**elete a resource (HTTP **DELETE**)
  - **Stateless execution**: after executing an operation at a service, that component forgets everything about the caller

RESTful APIs

Characteristics of a true RESTful API service

1) Uniform interface: standardized methods and unique resource identification scheme
2) Client-server based: separation of user concerns from data storage concerns, enabling indepentent development and maintenance
3) Stateless operations: requests from client to server must contain all of the information necessary to set the resource state
4) Resource caching: data within a response to a request must be labeled as cacheable or non-cacheable
5) Layered system: REST allows for an architecture composed of hierarchical layers (e.g.: interface implementation, data storage facility, authentication facility), where each component cannot see beyond the immediate layer with which it is interacting

REST API endpoint

- A URL defines how to reach a given resource (**endpoint**)
- Example:
- **http://hostname:5000/people/v1/users/id**
  - **http** specifies the application-layer protocol to be used
  - **hostname:5000** specifies the name of the host where the application endpoint is stored and the TCP port number to access the API server
  - **people** identifies the application context, i.e. which application's APIs we want to access
  - **v1** specifies the application version we want to access, if multiple versions are available
  - **users** identifies the resource we want to access
  - **id** specifies an optional parameter, e.g. to pass a variable or value as input to the endpoint to modify its behavior, or to request specific data