REST Architectural Constraints

1. Client-Server

2. Stateless

3. Cacheable

4. Uniform Interface

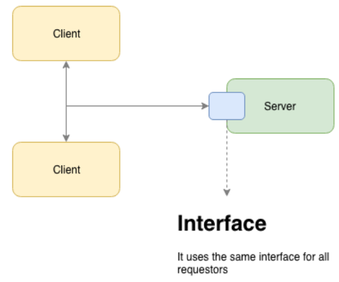
5. Layered System

6. Code On Demand

5 Constraints of the REST architectural style That You Should Know

Uniform interface

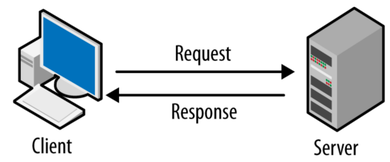
- Interface between client and server that allows for standardized client-server communication in a single language
- Necessary for the decoupling of client and server



- Uniform Interface
- Hypermedia as the engine of application state
- Self-descriptive messages: Messages contain all information necessary for recipient to interpret it, including metadata

Client-server

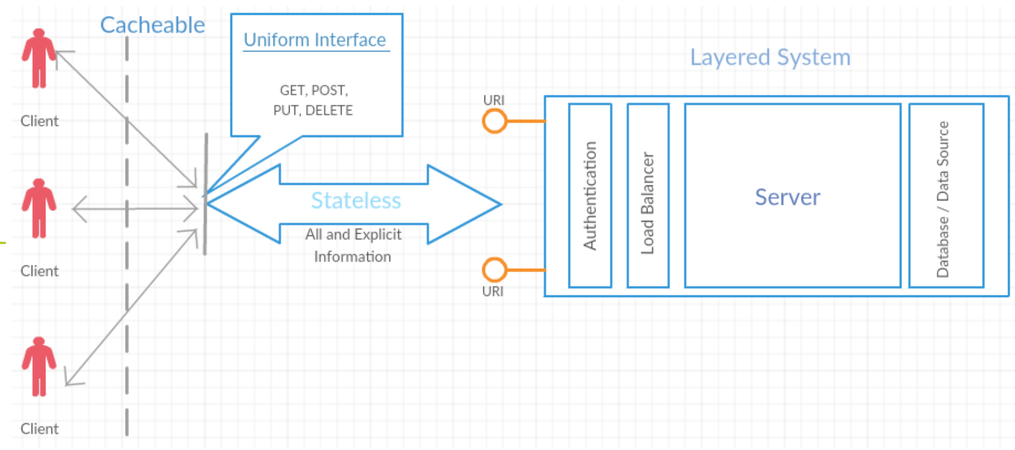
- Client-server model, for separation of concerns between client and server
- Supports portability and scalability



CS

Stateless

- Refers to stateless communication protocol, wherein the server stores no information about session states
- Improves performance by reducing server load



Server doesn't remember States

Cacheable

- Servers mark their responses as cacheable or non-cacheable.
- Clients and intermediaries are able to cache server responses.
- Reduces client-server interaction, supports scalability and performance.

Layered system

- Layers between client and server, can consist of intermediaries such as proxy servers or load balancers
- Layers have separate responsibilities but are able to interact with each other
- Supports system scalability and security

Optional

Code on demand

- optional constraint of RESTful architecture.
- client can download features after deployment
- Java applets and JavaScript

About Constraints

By applying this constraint:	We achieve the following system characteristics:
Client-Server	Simple, scalable and upgradeable
Stateless	Simple, visible, easy to maintain, reliable and upgradeable
Cache	Visible, scalable and efficient
Interface / Uniform contracts	Simple, usable, visible, accessible, upgradeable and reliable
Layered system	Flexible, scalable, reliable and efficient
Code-on-demand	Upgradable

6 Constraints