Web Architecture

...

Announcements

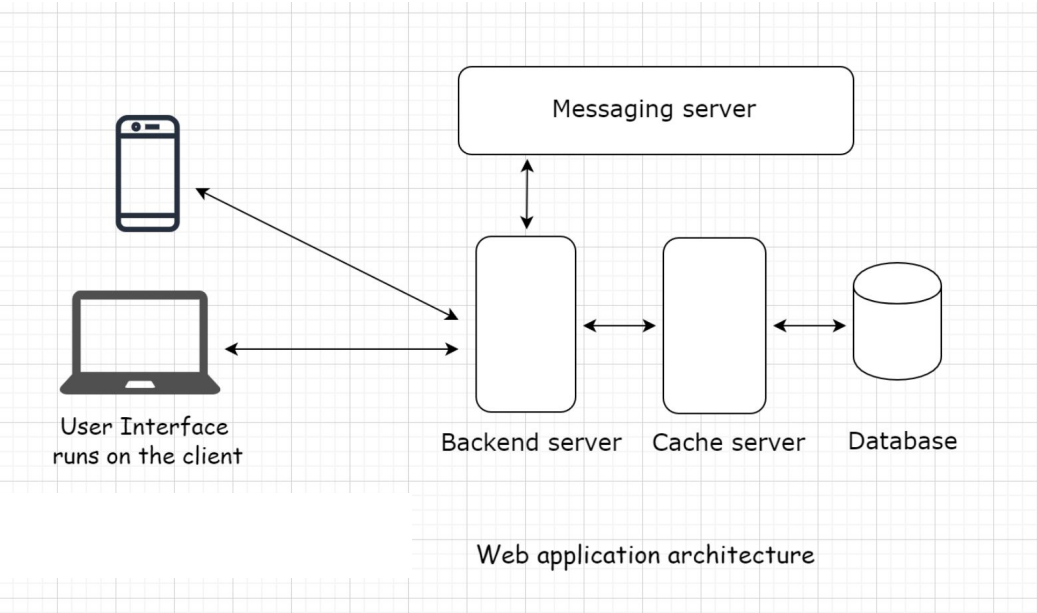
- Project 1: Due Monday, individual
- Project 2: implementing your designs for project 1 into a functional app! :)
- This lecture will be relevant for projects!

MVC Design Pattern

- We are starting our transition to higher-level architectural patterns!
- Congrats on surviving the Design Patterns phase
- MVC is closer to an architectural pattern than a design pattern
- For a small project, can still just be a way of organizing code

What is Web Architecture?

Web architecture involves multiple components like a database, message queue, cache, user interface, etc., all running in conjunction to form an online service.



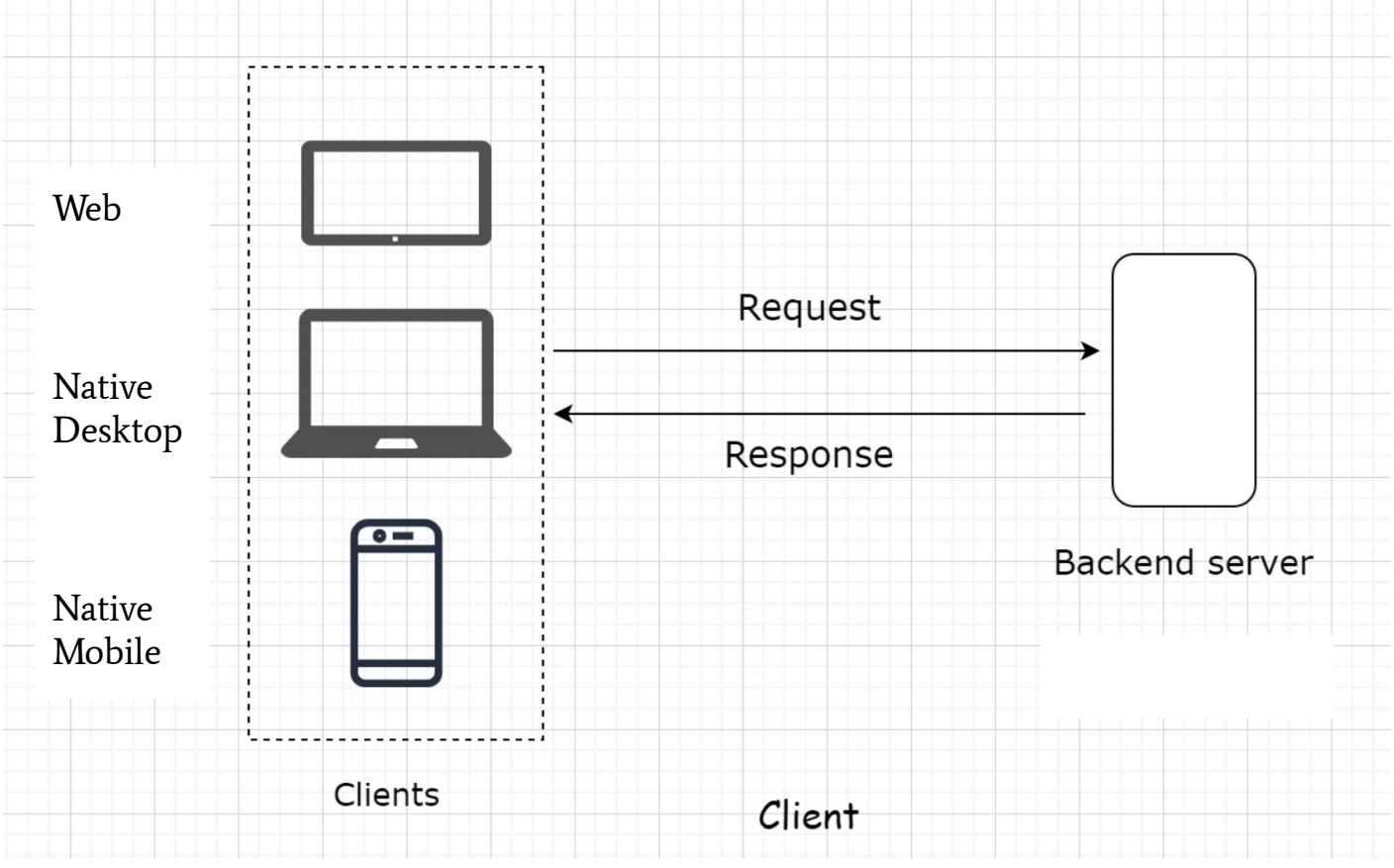
Client-Server

- Client-server architecture is the fundamental building block of the web.
- The architecture works on a request-response model.
  - The client sends the request to the server for information
  - The server responds with that information, or an error
- Nearly every website you browse is built on the client-server architecture.
  - A very small percentage of business websites and applications use the peer-to-peer architecture, which differs from the client-server.
    - We will see this later in the course, if there's time

Client Explained

- The client holds our user interface.
- The user interface is the presentation part of the application.
- In very simple terms, a client is a gateway to our application.
    - It can be a mobile app, a desktop or a tablet like an iPad.
    - It can also be a web-based console, running commands to interact with the backend server.

Types of Clients



Client Technologies

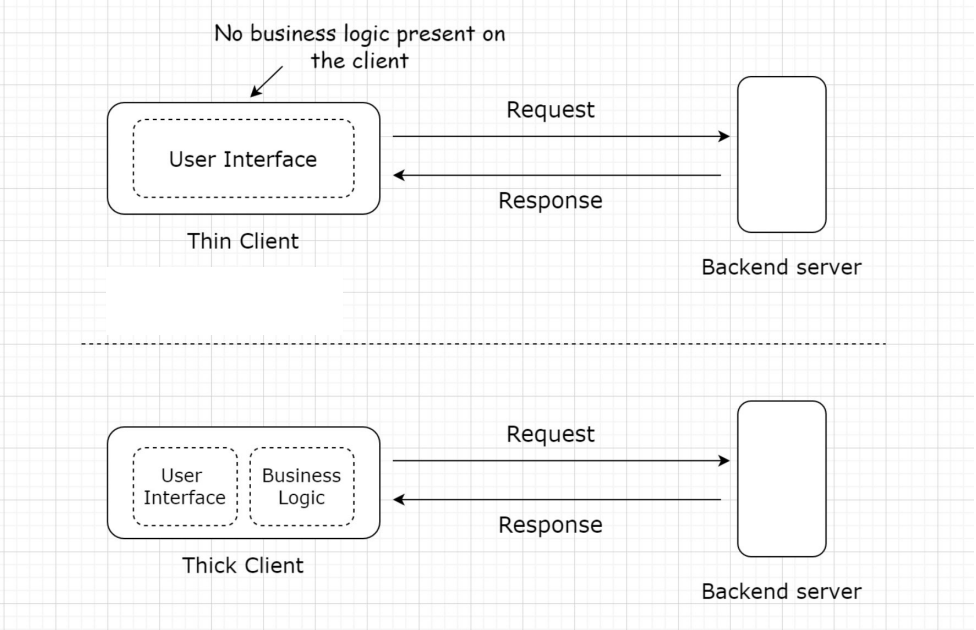
- Web:
  - The open-source technologies popular for writing the web-based user interface in industry:
    - jQuery, React, Angular, Vue, Svelte, etc.
    - All these libraries are written in JavaScript.
- Mobile Native:
  - Different platforms require different frameworks/libraries
    - Android
    - iOS
- Desktop Native:
  - Different platforms require different frameworks/libraries
    - MacOS
    - Windows
    - Linux

Designing your client

- What should be in the client vs the server?
- Thick vs Thin Client

Designing your client

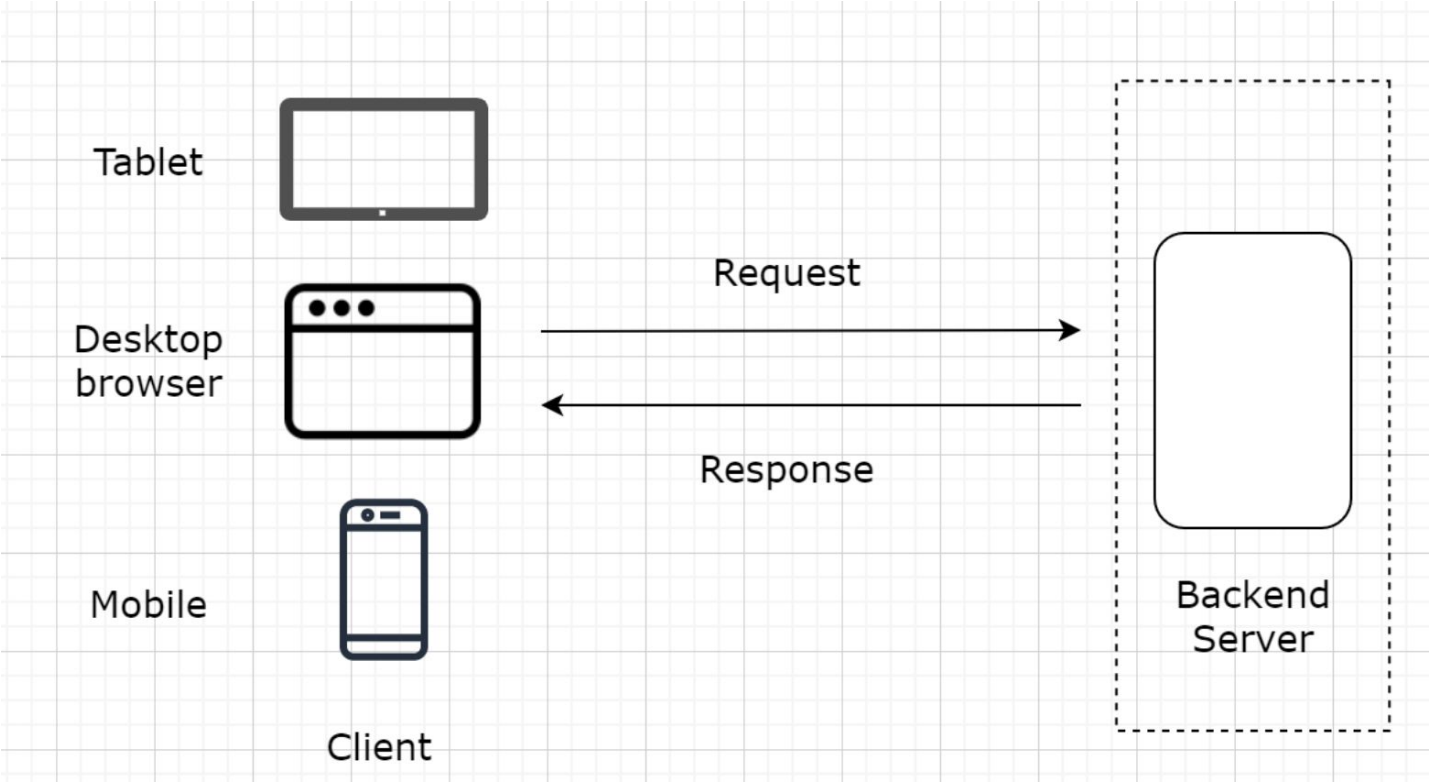
- Thin client
  - holds just the user interface of the application.
  - It contains no business logic of any sort.
  - For every action, the client sends a request to the backend server, just like in a three-tier application.
- Thick Client
  - Holds business logic



Server

- Server:
  - receive the requests from the client
  - provide the response after executing the business logic based on the request parameters received from the client.
- Every online service needs a server to run.
- Servers running web applications are commonly known as application servers.

Server


Tablet
Desktop browser
Mobile
Client
Request
Response
Backend Server

Server

- Besides the application servers, there are also other kinds of servers with specific tasks assigned. These include:
    - Proxy server
    - Mail server
    - File server
    - Virtual server
    - Data storage server
    - Batch job server and so on

Communication Between Client and Server

- Request-response model
  - The client sends the request and the server responds with the data.
  - If there is no request, there is no response.
- The entire communication happens over the HTTP protocol.
  - HTTP protocol is a request-response protocol that defines how information is transmitted across the web.
- It’s a stateless protocol, and every process over HTTP is executed independently and has no knowledge of previous processes.

Communication Between Client and Server

- When a client wants to communicate with a server, either the final server or an intermediate proxy, it performs the following steps:
  - Open a TCP connection
    - used to send a request, or several, and receive an answer.
  - The client may open a new connection, reuse an existing connection, or open several TCP connections to the servers.
- Send an HTTP message:
  - HTTP messages (before HTTP/2) are human-readable.
  - With HTTP/2, these simple messages are impossible to read directly, but the principle remains the same.
- Close or reuse the connection for further requests.

Example: Communication Between Client and Server

- Let's say we want to write an application to keep track of the birthdays of all our Facebook friends.
    - The app would then send us a reminder a couple of days before the birthday of a certain friend.
- To implement this, the first step would be to get the data of the birthdays of all our Facebook friends.
- We would write a client to hit the Facebook Social Graph API, which is a REST-API, to get the data and then run our business logic on the data.

Example: Communication Between Client and Server

- Mini-Quiz:
- When should we implement a **thick** client for our application? Which of the following option(s) are correct?
    - When we do not want anyone to have access to the code/business logic of our application.
    - When we need to reduce the bandwidth consumption in the client server communication, for a smooth user experience.

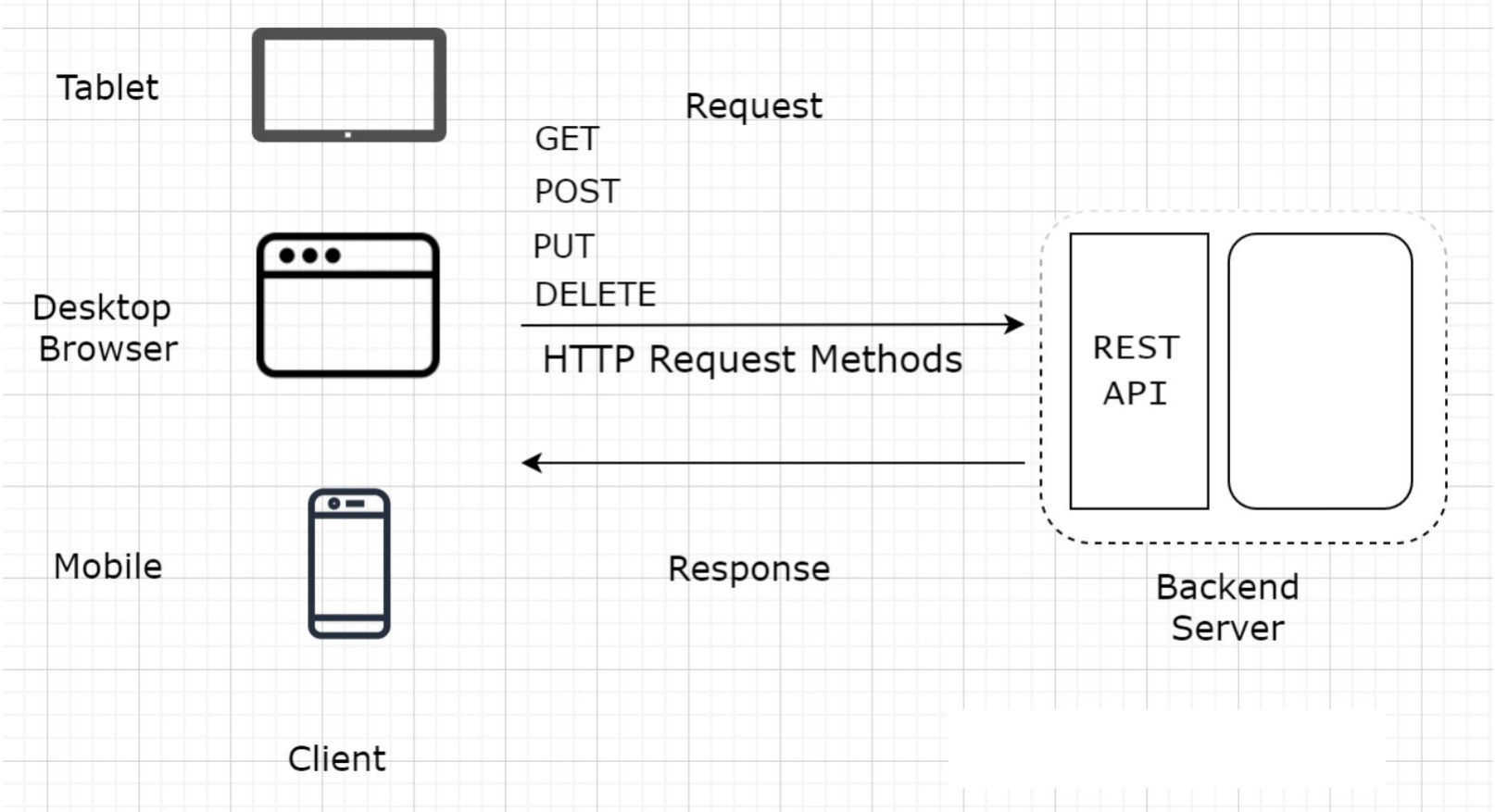
Example: Communication Between Client and Server

- Mini-Quiz:
- Who usually initiates communication in a client-server, request-response model?
    - Client
    - Server

REST API

- REST stands for Representational State Transfer.
- It’s a software architectural style for implementing web services.
- Web services implemented using the REST architectural style are known as the RESTful web services.
    - The communication between the client and the server happens over HTTP.
    - REST API takes advantage of the HTTP methodologies to establish communication between the client and the server.
    - REST also enables servers to cache the response that improves the application’s performance.

REST API


Tablet
Desktop Browser
Mobile
Client
Request
GET
POST
PUT
DELETE
HTTP Request Methods
Response
REST API
Backend Server

REST API: Methods

Method	Description
GET	Retrieve information about the REST API resource
POST	Create a REST API resource
PUT	Update a REST API resource
DELETE	Delete a REST API resource or related component

REST API: Statelessness

- The communication between the client and the server is a **stateless** process.
  - every communication between the client and the server is like a new one.
- There is no information or memory carried over from the previous communications.
  - Every time a client interacts with the backend, the client has to send the authentication information to it as well.
  - This enables the backend to figure out whether the client is authorized to access the data or not.

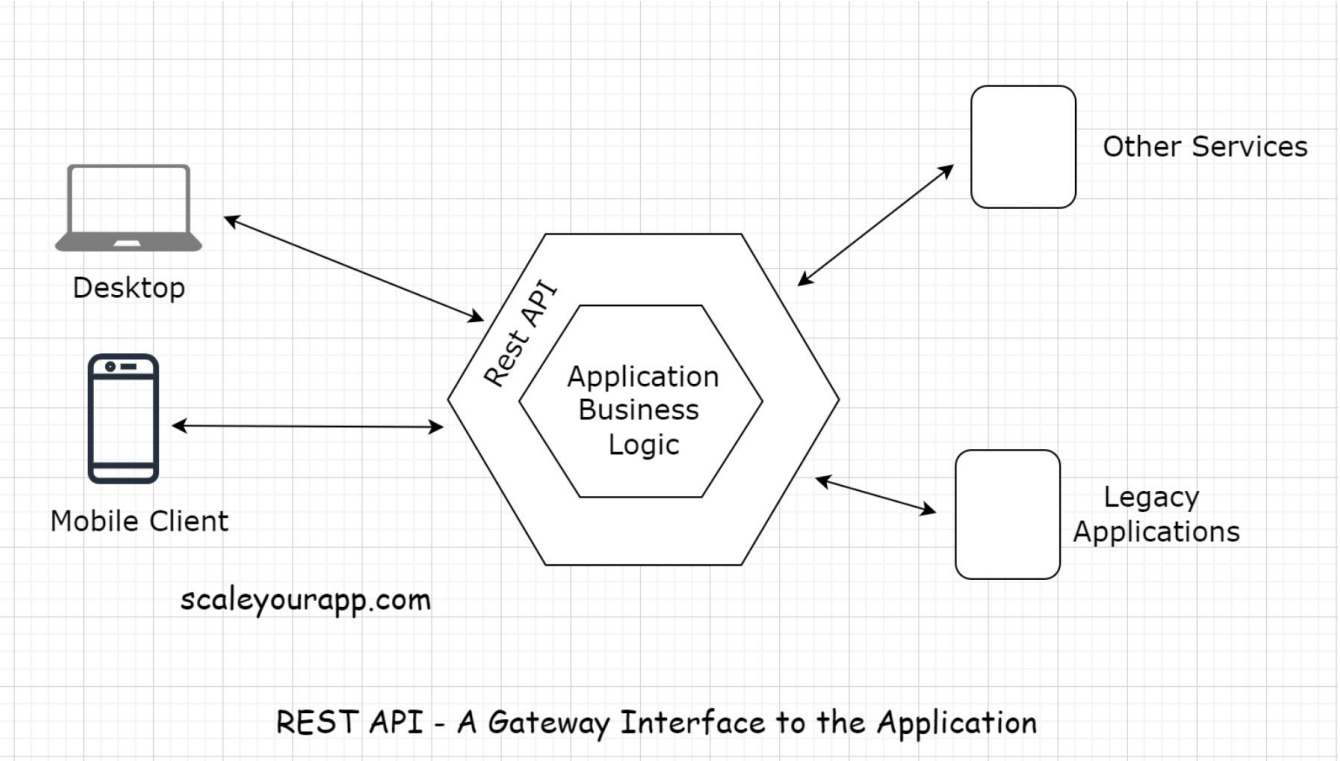
REST API: Definitions

- An API/REST/Backend endpoint means the URL of the service that the client could hit.
    - https://myservice.com/users/{username} is a backend endpoint for fetching the user details of a particular user from the service.
- The REST-based service will expose this URL to all its clients to fetch the user details using the above stated URL.

REST API: Decoupling

- When implementing a REST API, the client communicates with the backend endpoints. This entirely decouples the backend and the client code.
- With the REST implementation, developers can have different implementations for different clients, leveraging different technologies with separate codebases.
    - Different clients accessing a common REST API could be:
        - a mobile browser
        - a desktop browser, a tablet or an API testing tool.
- Introducing new types of clients or modifying the client code does not affect the functionality of the backend service.
- This means the clients and the backend service are **decoupled**.

REST API: Standardization == less work



REST API: Standardization == less work

- Before REST, client and server code could be tightly coupled
  - On the backend, we had to write separate code/classes for handling requests from different types of clients.
  - We needed a separate servlet for handling requests from a mobile client and a separate one for a web-based client.
  - Messy and hard to change
- In today's application development landscape, there is hardly any online service implemented without a REST API.
  - Want to access the public data of any social network? Just use their REST API.