WLUG
COMMUNITY | KNOWLEDGE | SHARE

# API DEMYSTIFIED

## THE FOSS FILES SEASON 4 | EPISODE 1

API ALCHEMY


Shreyash Patil

Shruti Jadhav

Aditya Aparadh


15 June 2024

WLUG
COMMUNITY | KNOWLEDGE | SHARE

# Table of Contents

# What are APIs?

- **API : A**pplication **P**rogramming **I**nterface
- Enable two software components to communicate with each other using protocols
- Abstract the underlying implementation and expose required parts only
- Ex. Os Apis, Library Apis etc

# What are Web Apis?

- APIs accessed over **Web** using HTTP/HTTPS protcols
- Commonly used to interact with web services, databases
- Specifically designed for web-based communication and often provides data to web applications
- Ex. Weather Api

# Birth of the Internet

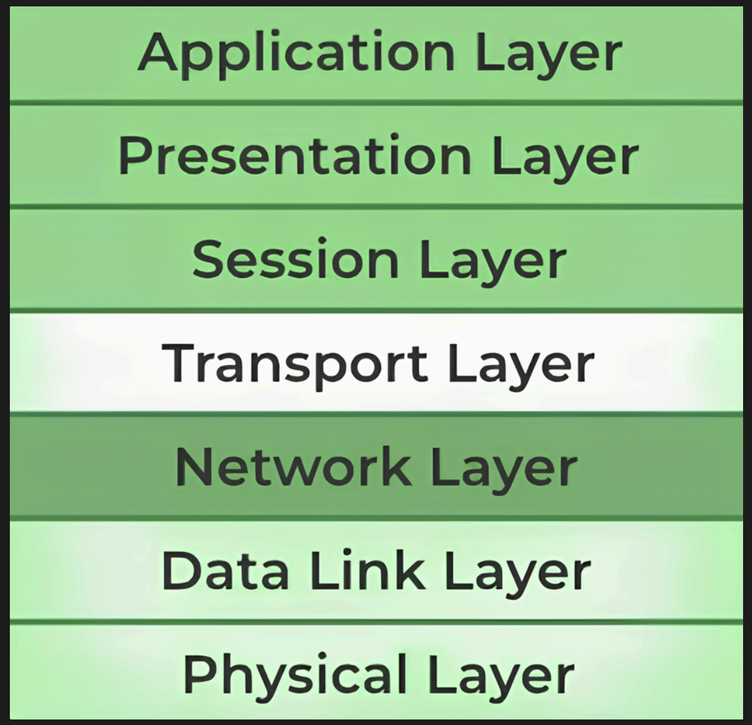
- First kind of interconnection in 1969 called the ARPANET by US DoD
- They invented Packet Switching
- Ray Tomlison invented email in 1971, he was the one who invented the '@'
- Tim Berners-Lee invented the World wide web in 1991

# Standardization of Communication protocol

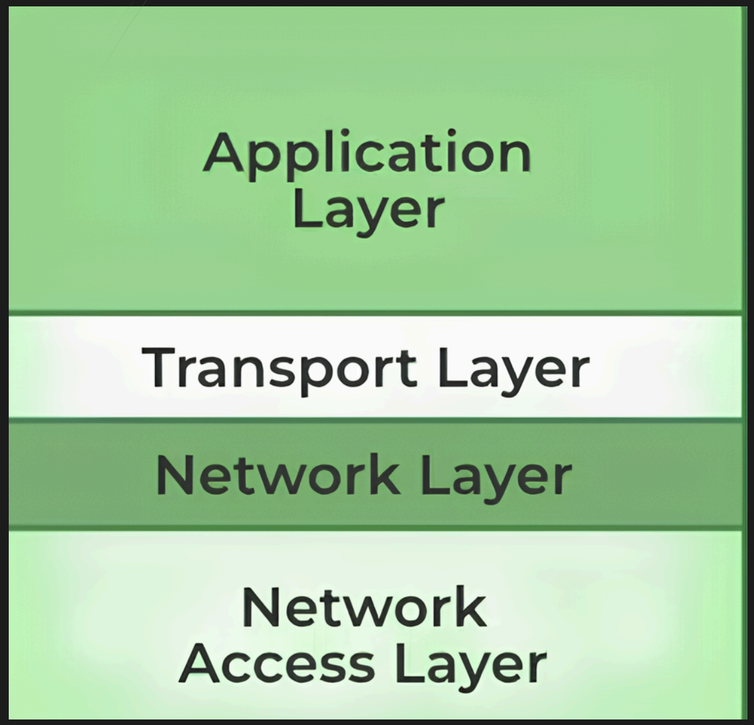
- To achieve communication between different networks
- Consistency between communication
- A reference model: OSI
- A practical model: TCP/IP

WLUG
COMMUNITY | KNOWLEDGE | SHARE

# OSI Model

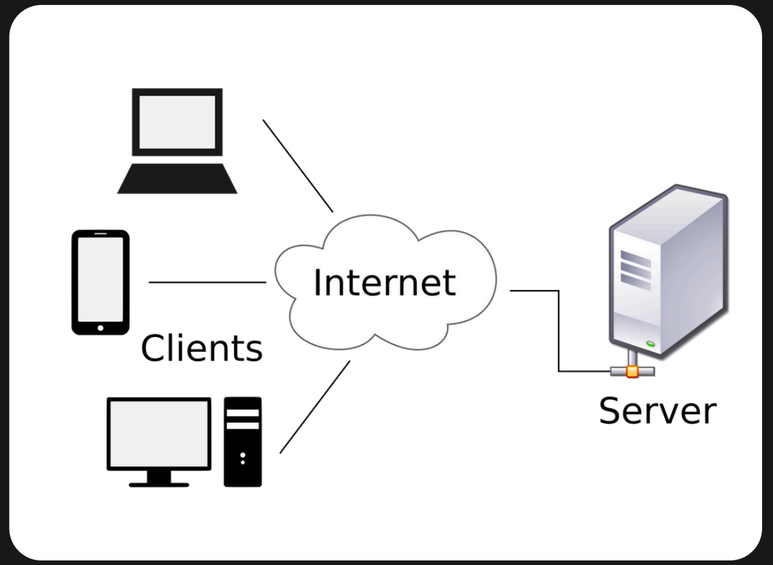



# TCP/IP Protocol



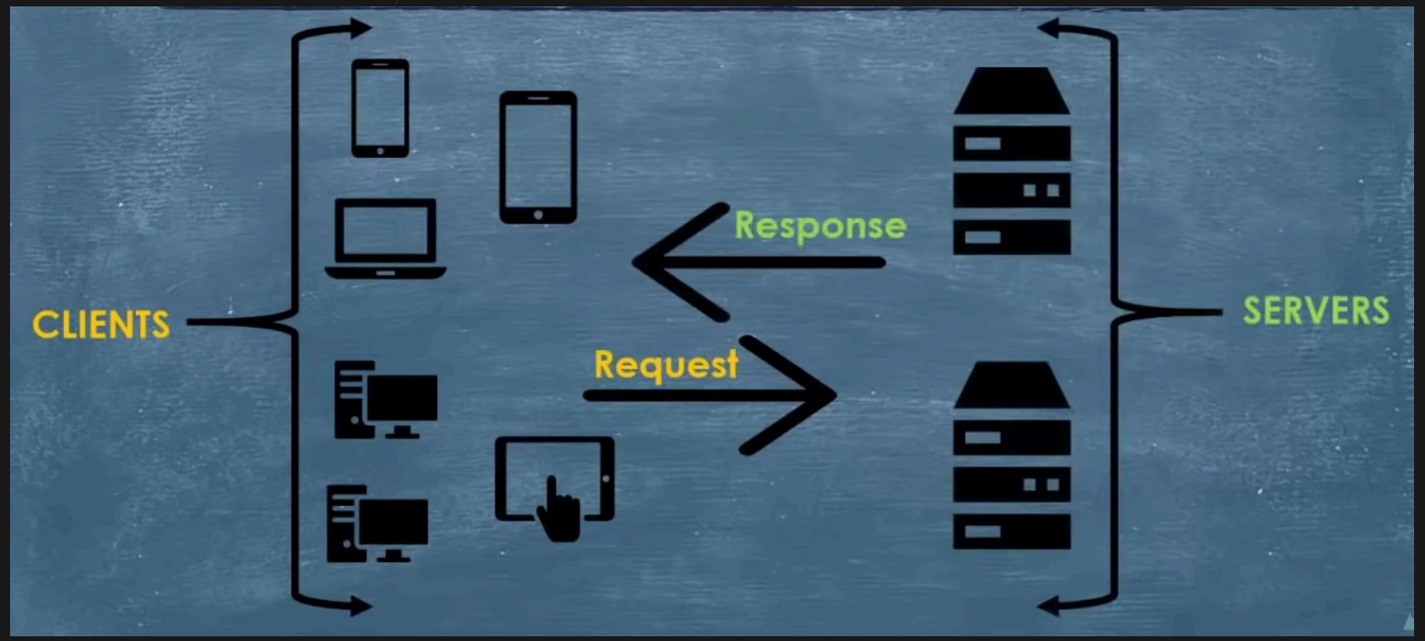
# Client-Server Architecture

- Client:
  - Machine or program that makes requests
  - Ex. PCs, Smartphones
- Server:
  - Program that responds to the requests
  - Ex. Http servers, Database Servers



# Client Server Model

- Centralized Structure
- One way to communicate via the web



# Application Layer

- HTTP: HyperText Transfer Protocol
- SMTP: Simple Mail Transfer Protocol
- DNS: Domain Name System
- FTP: File Transfer Protocol
- NFS: Network File System

# HyperText Transfer Protocol

- Allows web based applications to exchange data
- TCP/IP based protocol
- Used to deliver content in a client-server model
  Ex. images, videos, audio etc.
- Connectionless, stateless protocol

# HTTP 1.0

- Standardized in 1996
- Various methods : GET
  POST
  HEAD
- Just to fetch html documents
- Separate status code for request and response states
- Lack of persistent connection

# HTTP 1.1

- Standardized in 1997
- Overcame many limitations of HTTP 1.0
- Addition methods: DELETE
  CONNECT
  PUT
  TRACE
- Allowed pipelining
- Lack of persistent connection

# HTTP 2.0

- Standardized in 2015
- Implemented multiplexing
- Stream prioritization
- Header compression
- Used single long-lived connection
- Server push

# Status Codes

- Give the current state of HTTP request/response
  - 100: Continue
  - 200: OK
  - 404: Not found
  - 301: Moved permanently
  - 500: Internal server error

# Principles of Web API Design

## 1. Abstraction of Concerns

- **Seperation of Concerns:** API modules should focus on a single functionality
- **Encapsulation:** Hiding internal workings for a clear and user-friendly API

# Principles of Web API Design

## 2. Standardized Protocols / Interfaces

- **HTTP/HTTPS:** It defines a set of request methods (GET, POST, PUT, DELETE, etc.)

# Principles of Web API Design

## 3. Error Response Mechanism

- **HTTP Status Codes:** Use standard HTTP status codes to indicate the result of an API request

# Principles of Web API Design

## 4. Proper Documentation

- It serves as the primary interface between the API provider and the developer using the API

# Principles of Web API Design

## 5. Security Implementations

- Encryption
- Authentication
- Rate limiting
- Input validation
- API gateways

# Needs of Web API

## 1. Interoperability

- **Different Systems Communication:**

  APIs let different apps chat, even if they're built differently

- **Standardization:**

  Protocols ensure APIs speak a common language for effortless system integration

# Needs of Web API

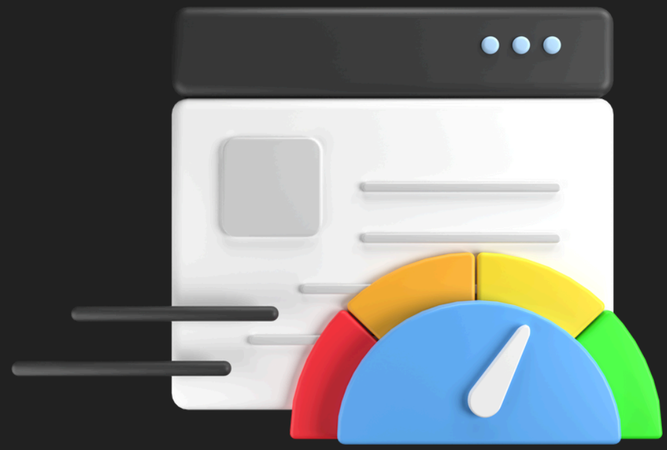
## 2. Scalability

- **Microservices Architecture:**

  Enable building applications as self-contained services

- **Load Distribution:**

  Enable workload distribution across servers



# Needs of Web API

## 3. Reusability

- **Modular Development:**

  Enable development with modular, reusable components

- **Cost Efficiency**

  Leverage pre-built functionalities, reducing development effort and cost

# Needs of Web API

## 4. Flexibility

- **Multi-Platform Support:**

  Provide functionalities accessible across platforms  application reach

- **Agility:**

  Enable independent backend updates, fostering agile development

WLUG
COMMUNITY | KNOWLEDGE | SHARE

# THANK YOU

Community | Knowledge | Share

API ALCHEMY


Shreyash Patil

Shruti Jadhav

Aditya Aparadh


15 June 2024