Arquitetura Cliente/Servidor

Cliente

- Programa ou dispositivo que inicia uma requisição por dados ou serviços.

- Ex: navegador, aplicativo de email, app de banco.

Servidor

- Programa (ou máquina) que escuta as requisições e responde com informações ou funcionalidades.

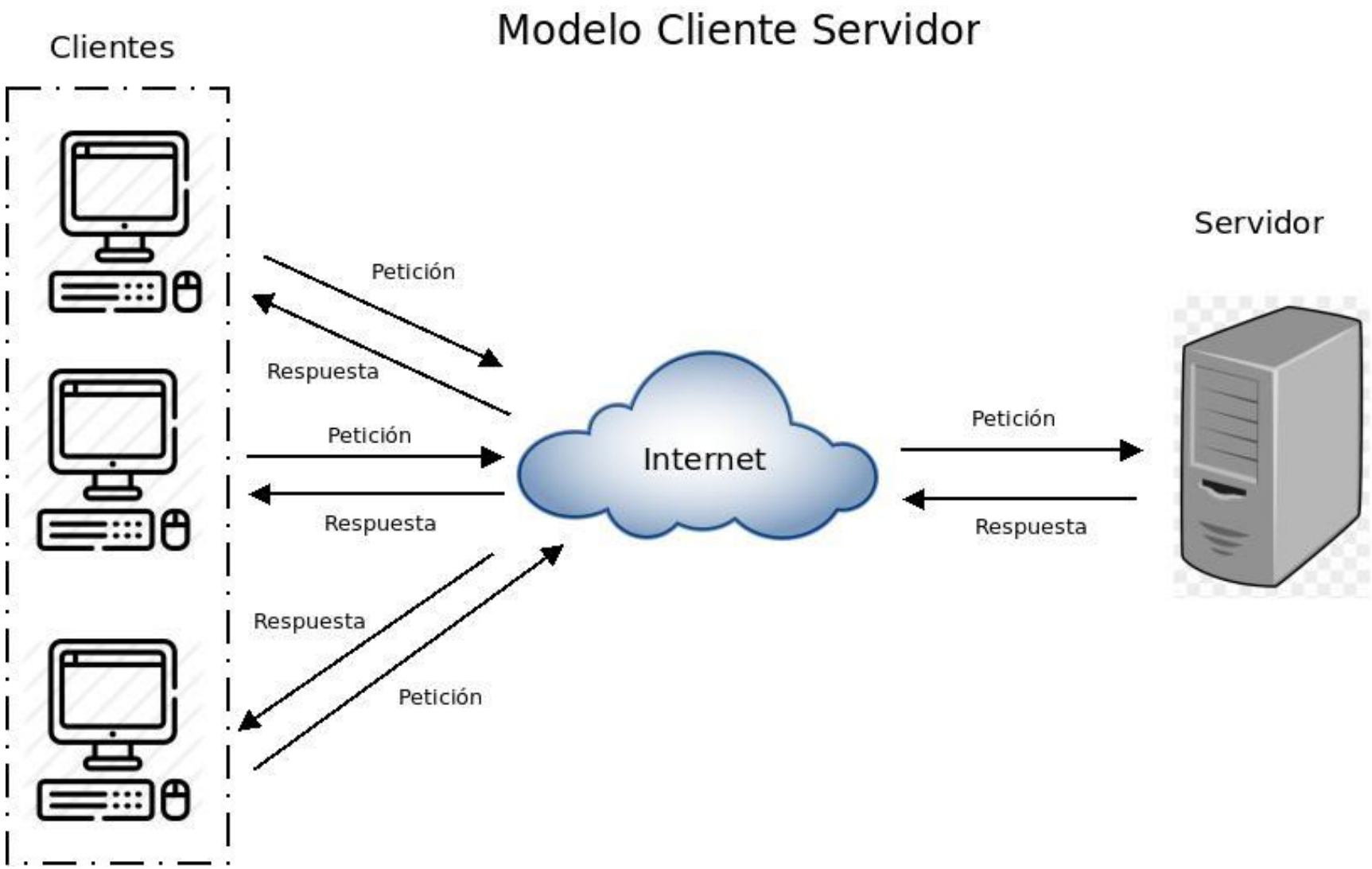
- Pode atender vários clientes ao mesmo tempo.

Serviço

- A funcionalidade oferecida pelo servidor.
- Ex: serviço de web, serviço de *email*, serviço de banco de dados, serviço de autenticação.
- O cliente consome o serviço. O servidor oferece o serviço.

Arquitetura Cliente/Servidor

- Modelo de comunicação onde:
  - Cliente faz requisição.
  - Servidor envia resposta.
- Responsabilidades separadas:
  - Cliente: interface e interação.
  - Servidor: dados, lógica e processamento.
- **Exemplo:** Navegador acessando um site (cliente requisita, servidor responde com HTML).


Modelo Cliente Servidor
Clientes
Servidor
Petición
Respuesta
Petición
Respuesta
Internet
Petición
Respuesta
Respuesta
Petición

Servidor Web

- Programa que fornece páginas web.
- Recebe requisições do navegador e devolve HTML, CSS, JS.
- Exemplo: Apache, Nginx, Flask, Express, Django

Navegadores e Portas

- Navegadores assumem **porta 80 para HTTP** e **porta 443 para HTTPS.**
- http://site.com → tenta site.com:80
- https://site.com → tenta site.com:443
- http://localhost:3000 → porta 3000 explícita

Endereço:porta

- Forma de acessar um serviço específico em um
  - servidor:127.0.0.1:5000
  - meusite.com:8080
- Evita conflitos e direciona para o serviço correto.

Node.js

- É um ambiente de execução JavaScript fora do navegador.
- Permite que o .js possa ser executado no computador, no servidor ou em scripts de linha de comando.
- É construído sobre o motor V8 do Google Chrome.

Vantagem

- Antes, o .js só era usado no **lado cliente**.
- Com node.js, podemos usá-lo no **lado servidor**.
- É possível:
  - Criar servidores web;
  - Construir scripts de automação;
  - Escrever testes automáticos;
  - Criar API e;
  - Desenvolver ferramentas e pacotes reutilizáveis.

Por que usar?

- É possível criar pacotes (módulos) com funções úteis.
- Testar e rodar código JS diretamente do terminal.
- Simular um ambiente de desenvolvimento real.

PACOTES E IMPORTAÇÃO

Conceito de pacote

- É um arquivo ou conjunto de arquivos com código reutilizável (funções, objetos e classes).
- Pode ser criado ou instalado via npm (usando pacotes já existentes).
- Importar um pacote significa usar o require() para trazer seu conteúdo para outro arquivo.