

# Introducción a las Aplicaciones Web

Programación Web

J.M. García



NETFLIX

HomeTV ShowsMoviesRecently AddedMy List

Award-Winning TV Shows

Trending Now

N

PEAKY BLINDERS

Br eaking Bad

N

TRAILER PARK BOYS

N

Critically Acclaimed TV Dramas

N

BLACK MIRROR BANDERSNATCH

GREY'S ANATOMY

TWO

N

RUSSIAN DOLL

Top Picks for bm25

Parks and Recreation

The Good Place

AMERICAN HORROR STORY

N

ARREST DEVELOPMENT

amazon

Entrega en Lebanon 66952

Actualizar ubicación

Off to College 2023

Buscar Amazon

Todo

Vuelta a Clases

Prepárate para la Uni

Clínica

Más Vendidos

Amazon Basics

Servicio al Cliente

Música

Nuevos lanzamientos

Prime

Ofertas del Día

Libros

Para la Universidad

Ofertas

Electrónicos

Favoritos

Habitación

Cocina

M

Hasta 45% en esenciales de tech universitaria

Compra ahora

Prime Student: prueba gratis de 6 meses

8:08

Buscar

+ ...

Todo

Recetas

Decoración

Insp

Inicio

Siguiendo

Notificaciones

Guardado

8:08

Buscar tus Pines

+ ...

Andrea

487 seguidores · Siguiendo a 224

Tableros

Pines

Probados

Sigui

49 tableros

Ideas para cenar

Hace 12 minutos

Sale

Inicio

Siguiendo

Notificaciones

Guardado

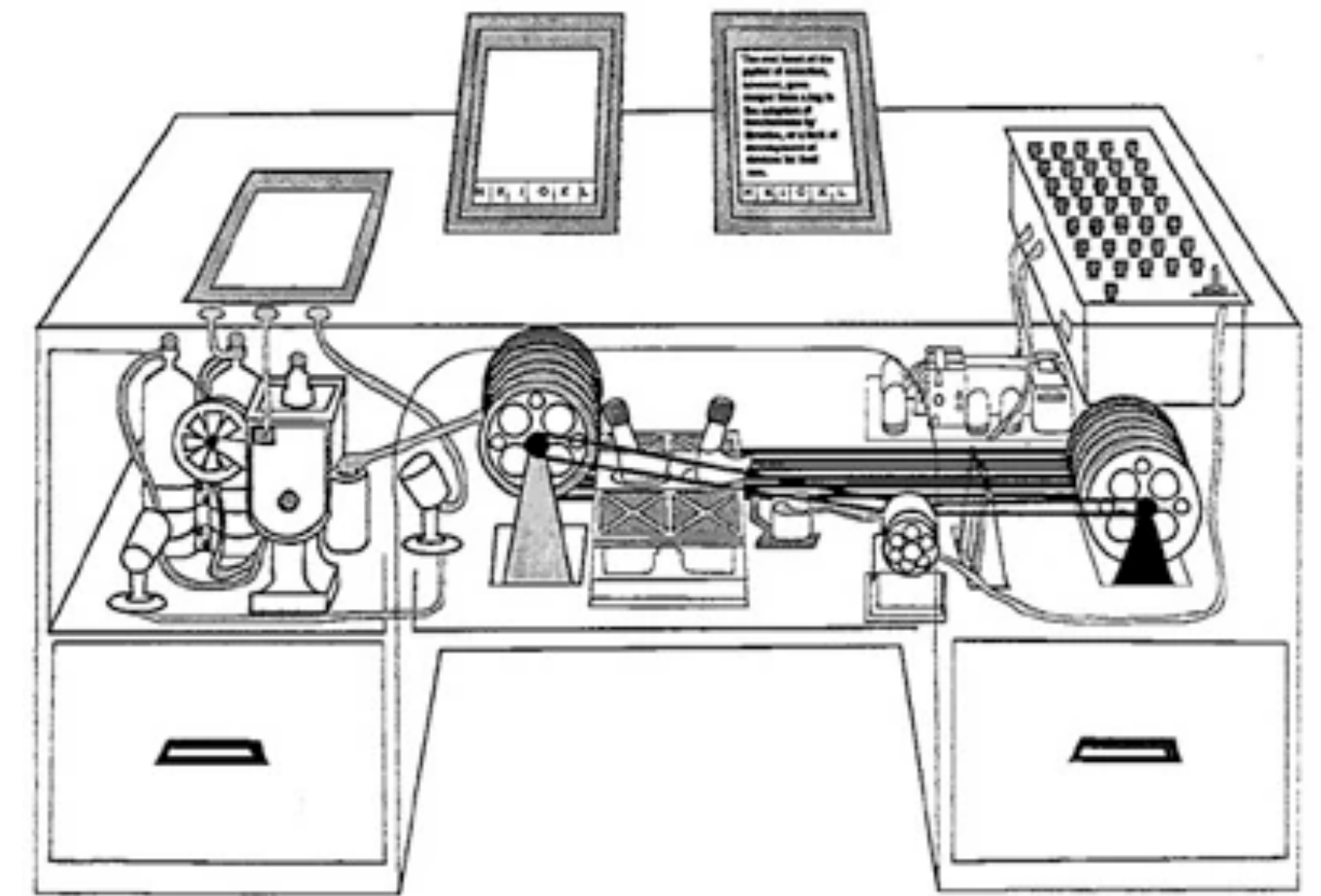


# Evolución de las aplicaciones Web

# As We May Think (Cómo podríamos pensar)

Vannvar Bush (Julio/Septiembre de 1945)

- Artículo de 10 páginas
- ¿Cómo podemos acceder más fácilmente al conocimiento?
- Memex (Memory Extender):
  - Recupera Microfilms y los despliega en múltiples pantallas.
  - Se pueden establecer secuencias asociativas entre los films, se registran los documentos visitados.
  - Se pueden escribir anotaciones en ellos.





# Proyecto Xanadú

Es el primer proyecto de hipertexto fundado en 1960 por Ted Nelson.

- Un documento puede tener hipervínculos a otros textos y sus distintas versiones.
- Pone a su alcance la gestión de enlaces entre textos relacionados a su obra, fragmentos, notas, comentarios, borradores, bosquejos, referencias, etc.
- En Xanadú se planteaba un solo documento global que contenga todo lo escrito en el mundo. Un docuverso.

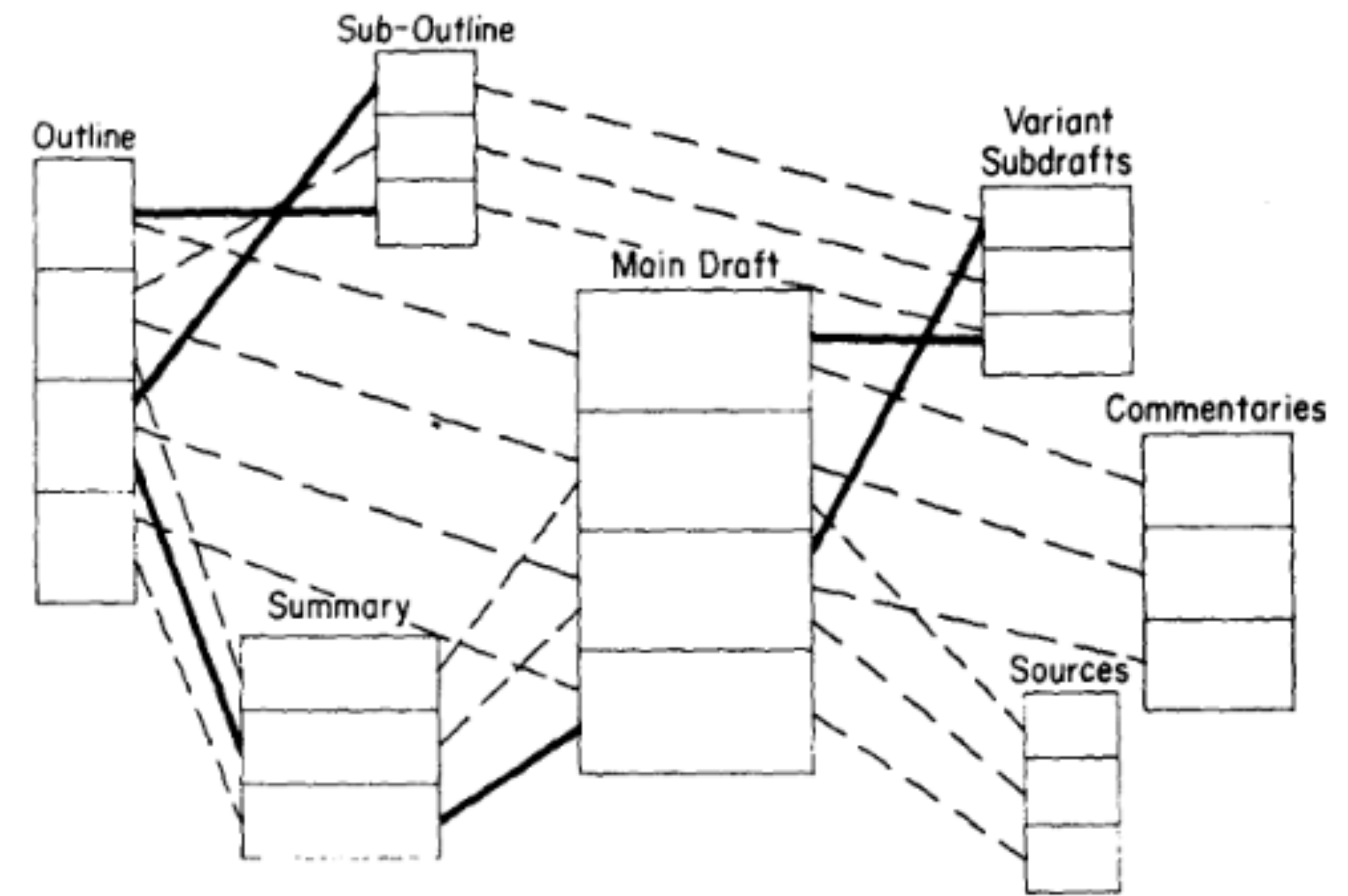
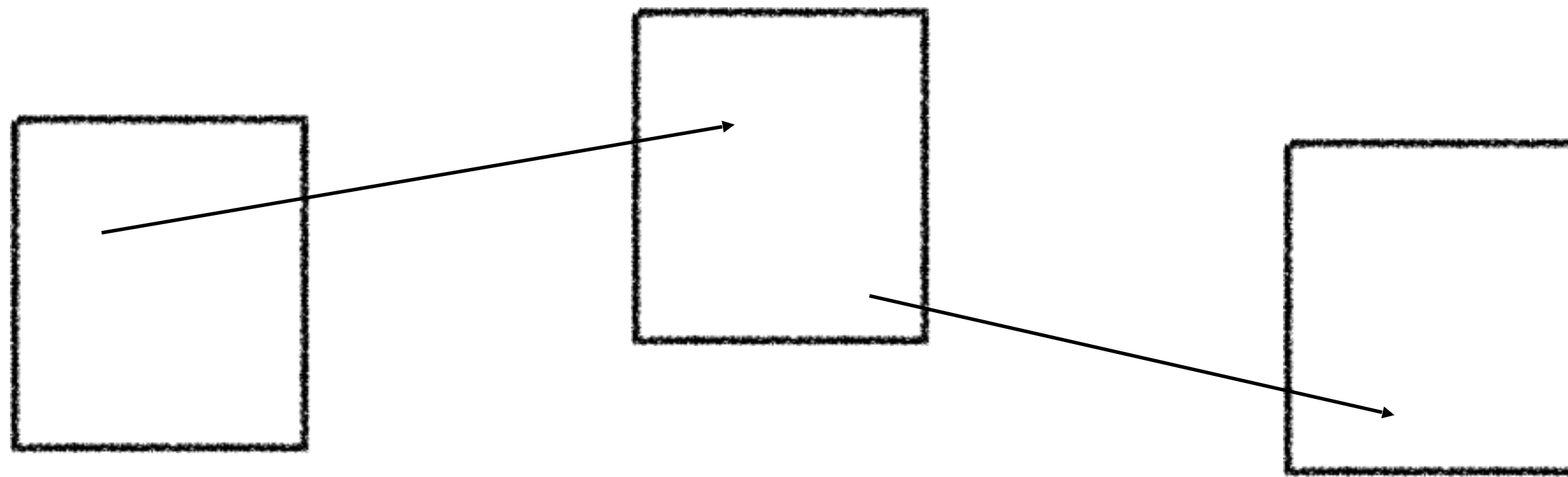


FIGURE 3—All levels of documentation in the ELF.

# Hipermedia

- Además de texto se pueden mostrar otros formatos digitales como imágenes, audio y video.
- Se pueden presentar diferentes fragmentos de información al usuario.
- Podemos recorrer múltiples recursos siguiendo las hiperligas, a esto le llamamos navegar.





# World Wide Web

Tim Berners Lee - Robert Cailliau 1990



- Robert Cailliau desarrolló el primer sistema hipertexto para el CERN en 1987.
- Tim Berners-Lee creó el sistema hipertexto para acceder a los múltiples documentos relacionados con el CERN.
- La WWW obtuvo el premio ACM Software System Award en 1995



# Componentes de la World Wide Web

- Primer versión del protocolo HTTP
- Formato HTML
- Sintáxis de URL
- Se esperaba que funcionara con distintos tipos de archivo.
- Descentralizado
- Los archivos son mutables
- Hiperlinks de Ted Nelson
- Protocolo liberado por CERN en 1993

## World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents. Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 access](#) , [Frequently Asked Questions](#) .

[What's out there?](#)  
Pointers to the world's online information, [subjects](#) , [W3 servers](#) , etc.

[Help](#)  
on the browser you are using

[Software Products](#)  
A list of W3 project components and their current state. (e.g. [Line Mode](#) , [X11 Viola](#) , [NeXTStep](#) , [Servers](#) , [Tools](#) , [Mail robot](#) , [Library](#) )

[Technical](#)  
Details of protocols, formats, program internals etc

[Bibliography](#)  
Paper documentation on W3 and references.

[People](#)  
A list of some people involved in the project.

[History](#)  
A summary of the history of the project.

[How can I help?](#)  
If you would like to support the web.

[Getting code](#)  
Getting the code by [anonymous FTP](#) , etc.

HTML

WorldWideWeb  
Navegador

HTTP Request  
GET /index.html

HTTP Response

Servidor HTTP

HTML

Internet  
TCP/IP

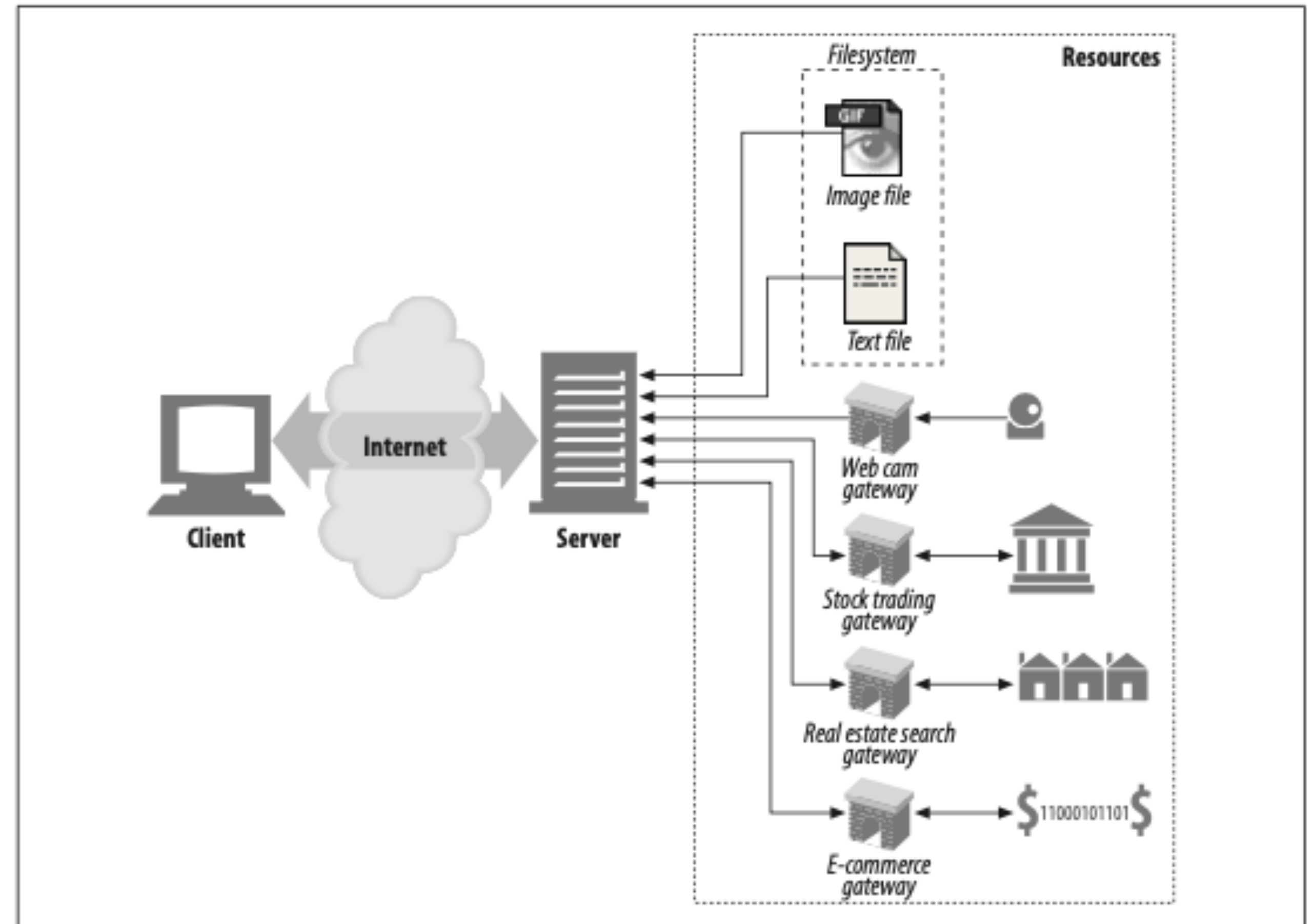
<https://www.w3.org/Protocols/HTTP/AsImplemented.html>



# HTTP

## Recursos

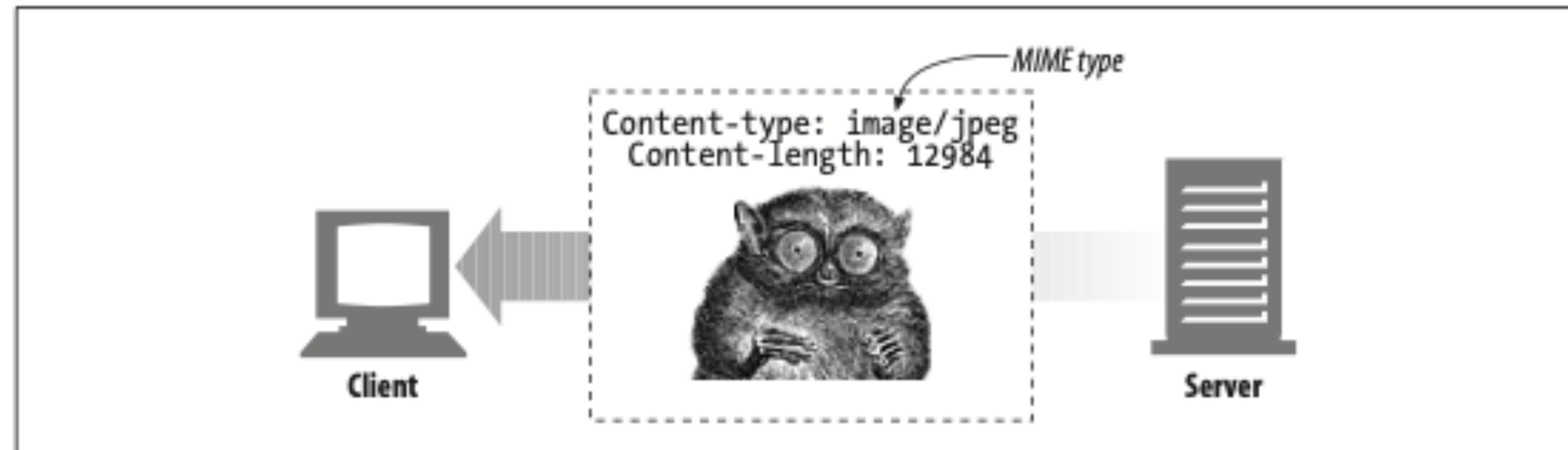
- Archivos Estáticos:  
Texto, pdf, Word, jpeg, avi, html, css.
- Recursos dinámicos  
Cámara web, intercambio de valores, compras en línea.  
Gateway, APIs.



# HTTP

## MIME

- text/html
- text/javascript
- image/jpeg
- application/json



[https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics\\_of\\_HTTP/MIME\\_types/Common\\_types](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types)

## URI/URL

- <https://developer.mozilla.org>
- <https://developer.mozilla.org/en-US/docs/Learn/>
- <https://developer.mozilla.org/en-US/search?q=URL>
- <http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument>

Protocolo

Dominio

Puerto

Ruta

Query String

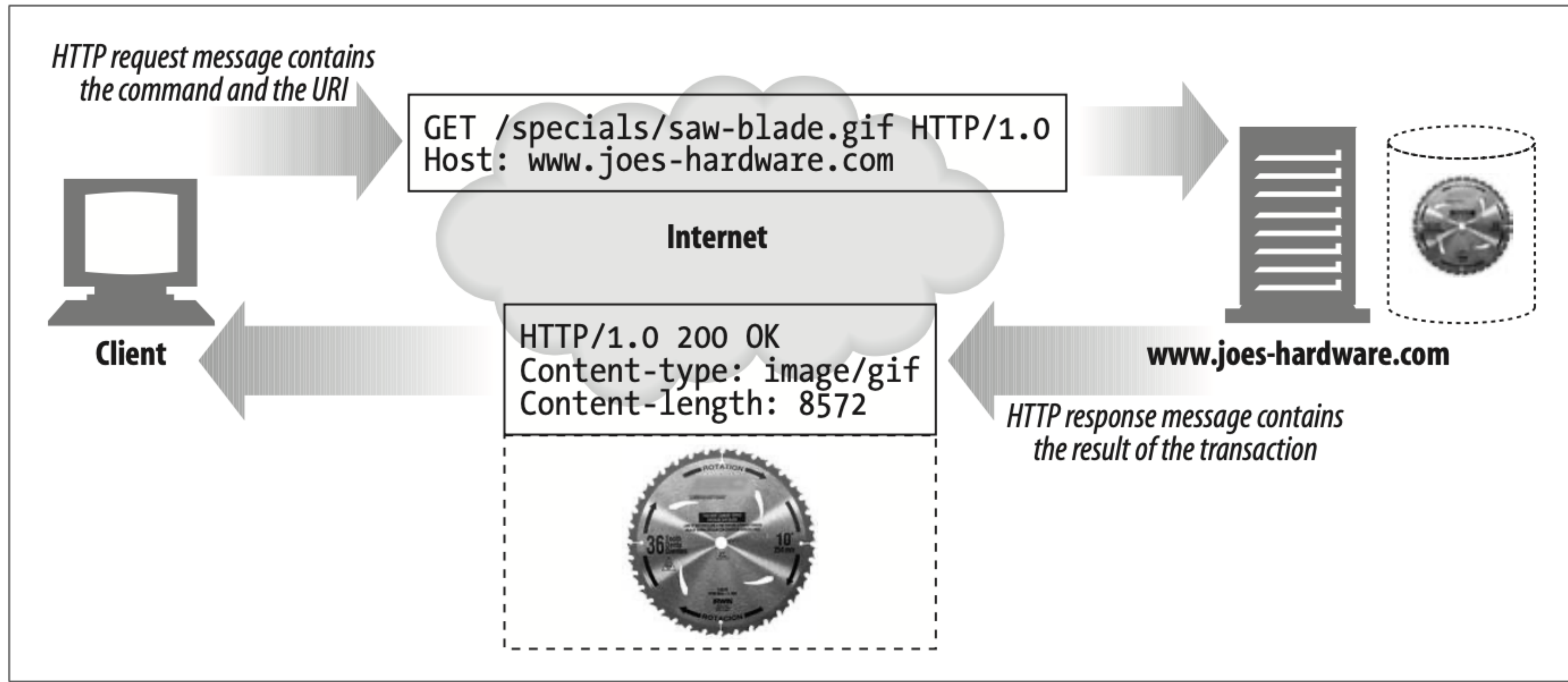
Ancla

file, ftp, ssh, urn, mailto, ws



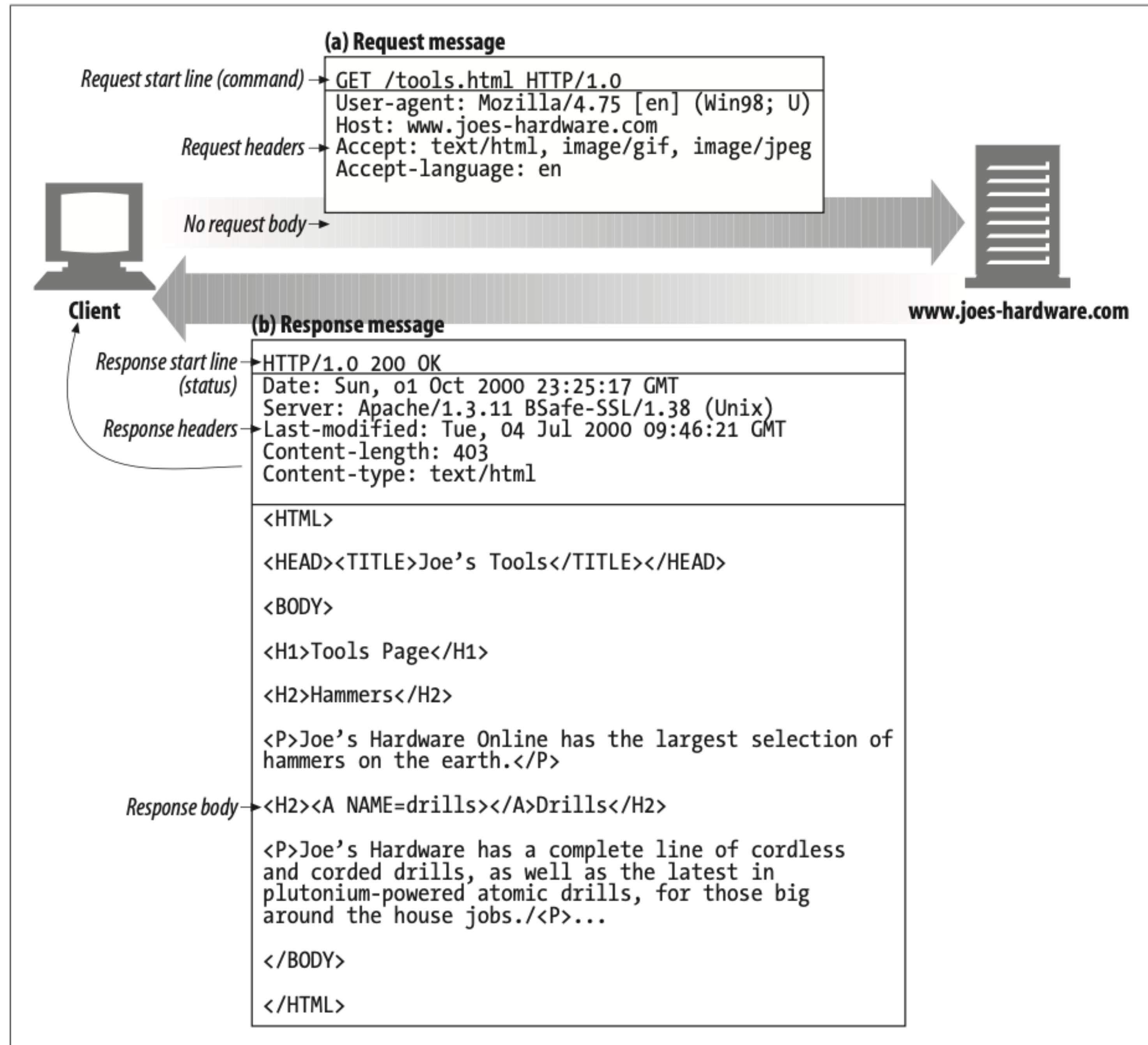
# HTTP

## Transacción HTTP



# HTTP

## Request-Response



# HTTP

Métodos más utilizados

Método	Descripción	Cuerpo
GET	Regresa el recurso solicitado (una representación), este método no hace modificaciones.	NO
PUT	Reemplaza las representaciones actuales del recurso solicitado. Con el payload del request.	SI
POST	Envía una entidad al recurso especificado, en muchos casos cambia el estado y puede causar efectos secundarios en el servidor.	SI
DELETE	Borra el recurso especificado.	NO

# HTTP

Métodos más utilizados

Código	Descripción
200	OK
302	Redirección, ve a otro lugar por el recurso.
404	No se encontró.
DELETE	Borra el recurso especificado.



# Aplicaciones Web

## De Hipermedia a Single Page Applications

- Los navegadores evolucionaron añadiendo distintas capacidades para ejecutar código en ellos. Applets, ActionScript, Flash, JavaScript.
- Tenemos ahora ***thick clients***. Buena parte de la aplicación se implementa en el cliente utilizando JavaScript, haciendo solo llamadas asíncronas a los servidores web intercambiando datos JSON.
- No es necesario navegar entre recursos. Esto contrasta con los sistemas hipermedia.
- Se utilizan APIs REST (basados en HTTP) para utilizar servicios internos y externos.
- Se despliegan las aplicaciones utilizando servicios cloud.

# Arquitectura de las aplicaciones Web

# Arquitectura de Aplicaciones Web

## Arquitectura multi capas



```
graph TD; A[Intefaz de Usuario] --> B[Logica de Negocio]; B --> C[Acceso a Datos];
```

The diagram illustrates a multi-layer web application architecture. It consists of three stacked rectangular boxes. The top box is blue and labeled 'Intefaz de Usuario'. The middle box is orange and labeled 'Logica de Negocio'. The bottom box is red and labeled 'Acceso a Datos'. Arrows indicate a flow from the top layer to the middle layer, and from the middle layer to the bottom layer.

Intefaz de Usuario

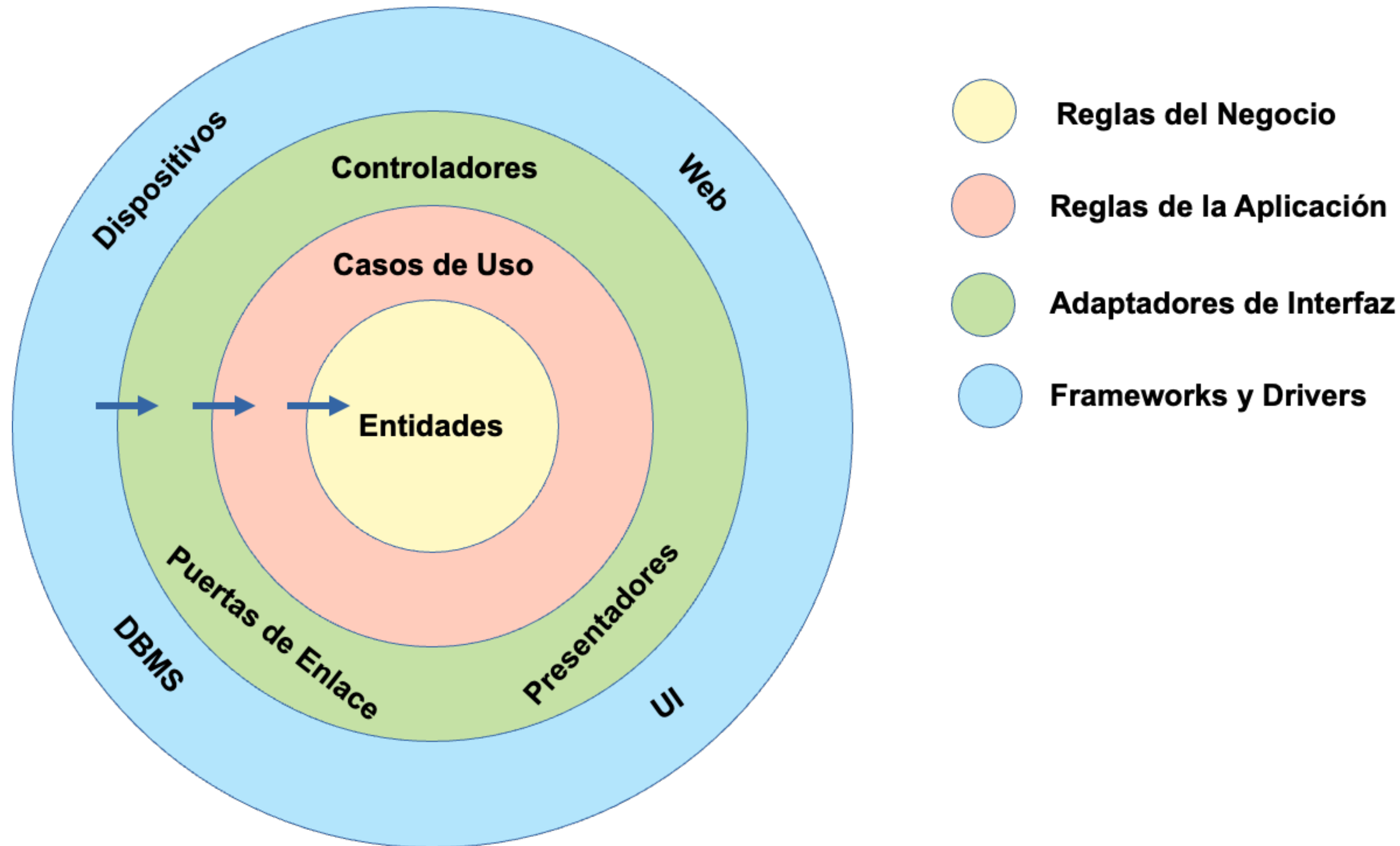
Logica de Negocio

Acceso a Datos



# Arquitectura de Aplicaciones Web

## Arquitectura limpia

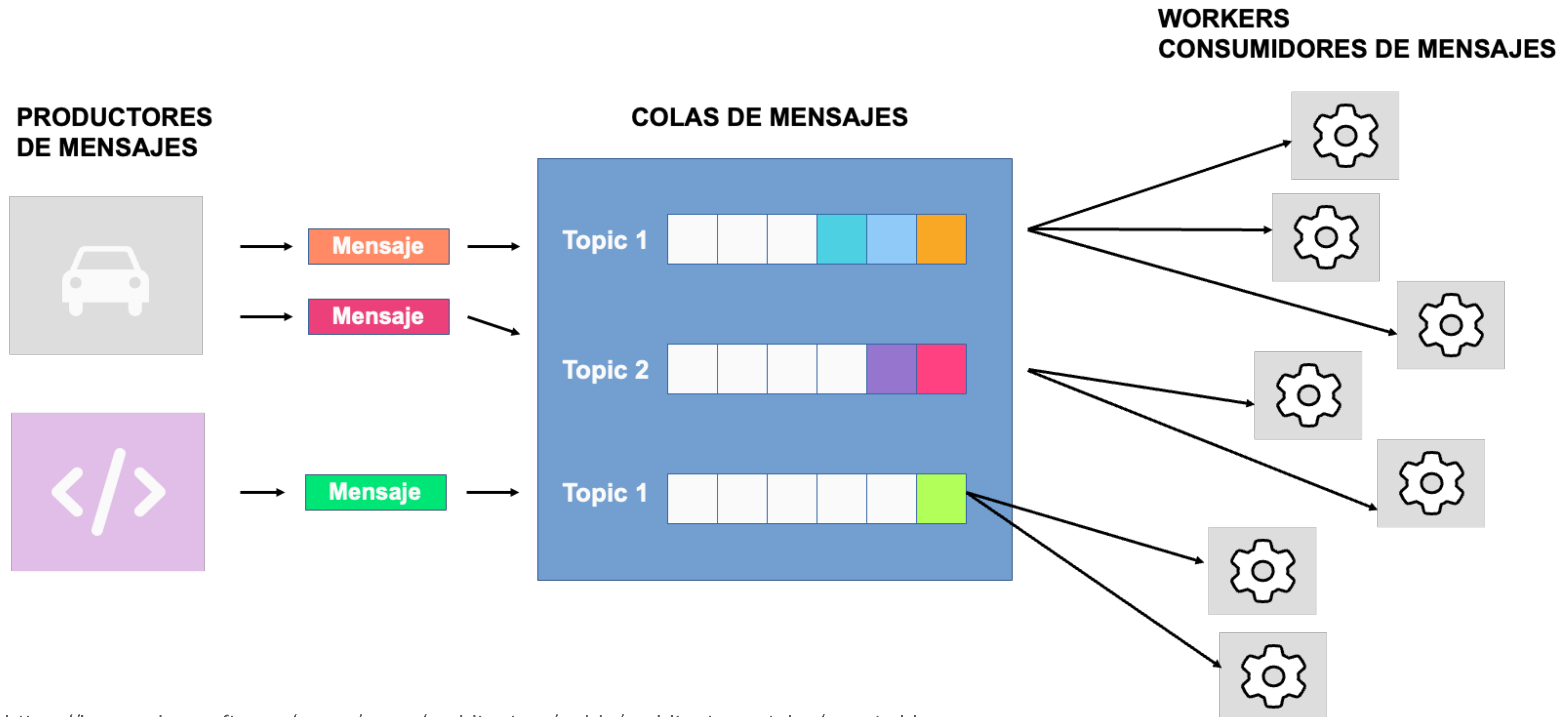


# Al seguir una arquitectura limpia se busca:

- Independencia de Frameworks. La arquitectura no depende del uso de un framework o librería en particular.
- La posibilidad de realizar pruebas a las Reglas de Negocio sin la necesidad de tener una UI, Base de Datos, Servidor Web, u otros elementos externos.
- Independiente del UI. La UI puede cambiar fácilmente, sin la necesidad de cambiar el resto del sistema. Por ejemplo, se podría cambiar la interfaz de web a la terminal sin cambiar las reglas de negocio.
- Independiente de la Base de Datos. Se debería poder cambiar de Oracle a SQLServer, a MongoDB o redis o cualquier otra tecnología.
- Independientes de sistemas externos. Las reglas de negocio no dependen del mundo externo.

# Estilo arquitectónico basado en eventos

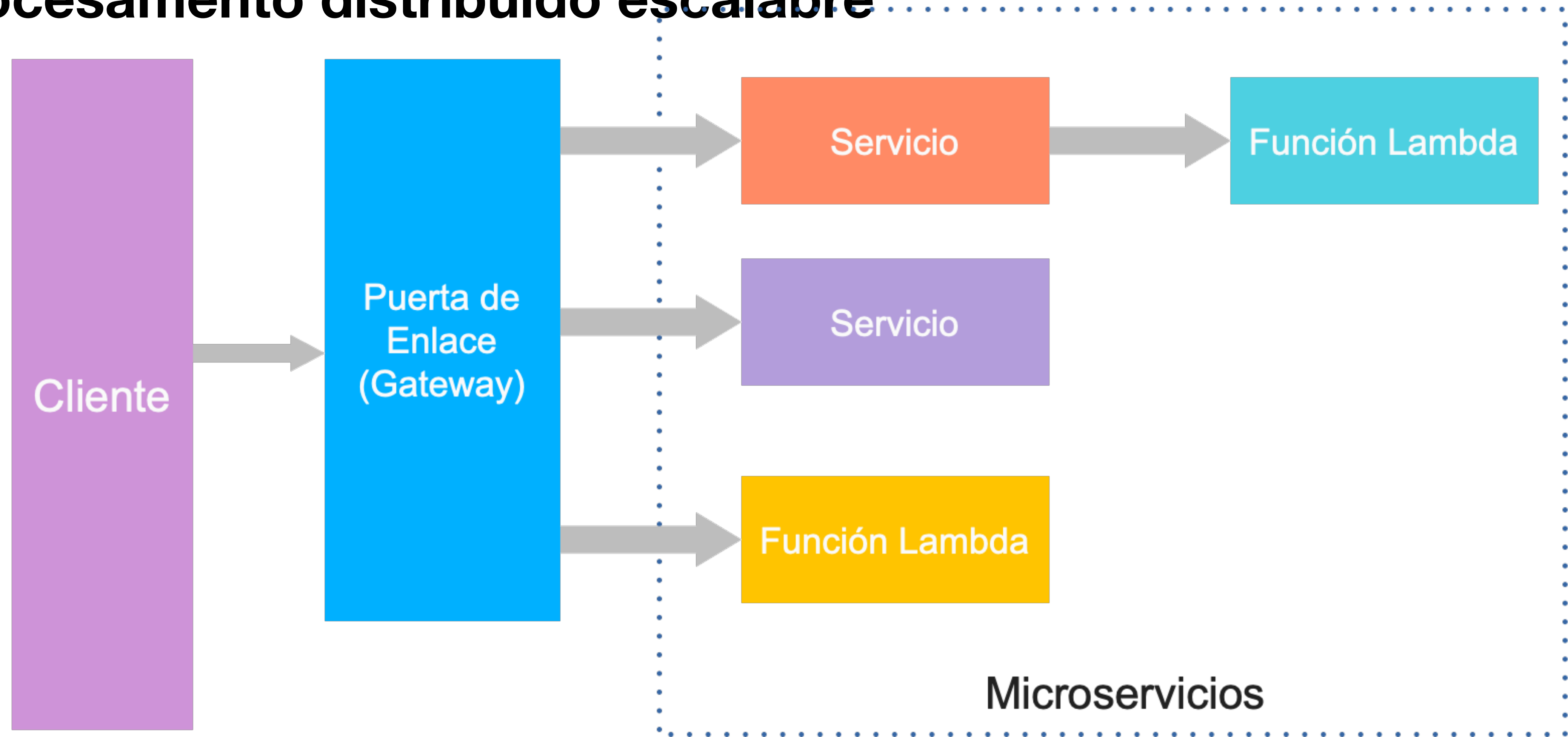
## Procesamiento distribuido escalable





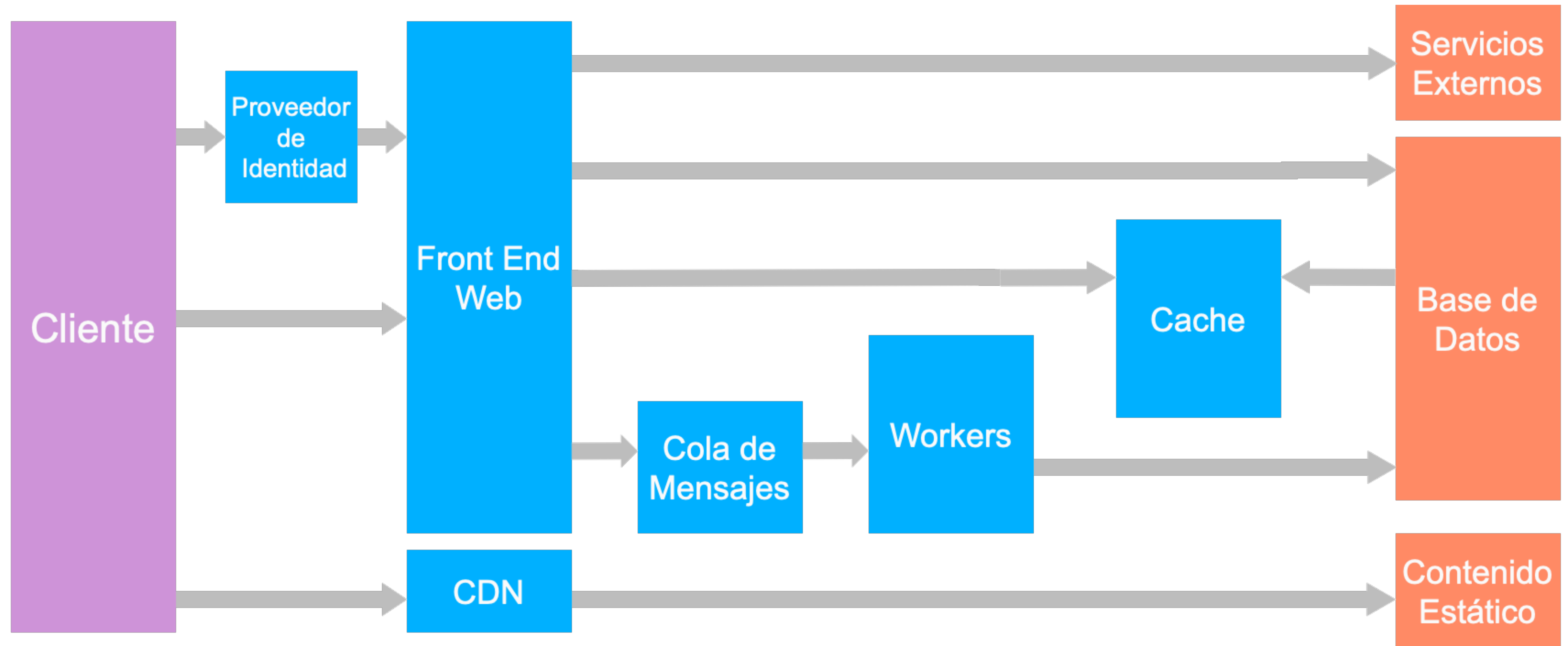
# Estilo arquitectónico basado en microservicios

## Procesamiento distribuido escalable



# Estilo arquitectónico web queue worker

## Procesamiento distribuido escalable



# Tecnologías para el desarrollo de aplicaciones Web



# Librerías para consumir HTTP

- En la mayoría de lenguajes se tienen librerías para implementar servidores HTTP básicos, que no deben usarse en producción, pero son buenos para aprender o proyectos sencillos.
- En Python se cuenta con la librería estándar `http.server`. La clase `HTTPServer` crea y escucha por el socket HTTP, enviando las peticiones a un handler:

```
def run(server_class=HTTPServer, handler_class=BaseHTTPRequestHandler):  
    server_address = ('', 8000)  
    httpd = server_class(server_address, handler_class)  
    httpd.serve_forever()
```

# Librerías estándar para consumir HTTP

- También se tienen librerías para procesar algunos elementos de HTTP
  - http
    - cookies
    - HTTP status codes
    - HTTP method
    - http.client
  - urllib con módulos para trabajar con URLs
    - response
    - parse
  - xmlrpc

# Librerías especializadas

## Requests: HTTP for Humans™

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"type":"User"...'
>>> r.json()
{'private_gists': 419, 'total_private_repos': 77, ...}
```

- La librería cubre con todas las necesidades para enviar peticiones HTTP/1.1 fácilmente
- Incluye añadir query strings, datos del cuerpo en una petición con POST, Keep-alive.
- Utiliza internamente urllib3.



# Librerías especializadas

## Beautiful Soup

- La librería cubre con todas las necesidades para extraer datos de documentos HTML y XML.

```
html_doc = """<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""
```

# Librerías especializadas

## Beautiful Soup

- La librería cubre con todas las necesidades para extraer datos de documentos HTML y XML.

```
for link in soup.find_all('a'):
    print(link.get('href'))
# http://example.com/elsie
# http://example.com/lacie
# http://example.com/tillie
```

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

```
html_doc = """<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""

from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')

soup.title
# <title>The Dormouse's story</title>

soup.title.name
# u'title'

soup.title.string
# u'The Dormouse's story'

soup.title.parent.name
# u'head'

soup.p
# <p class="title"><b>The Dormouse's story</b></p>

soup.p['class']
# u'title'

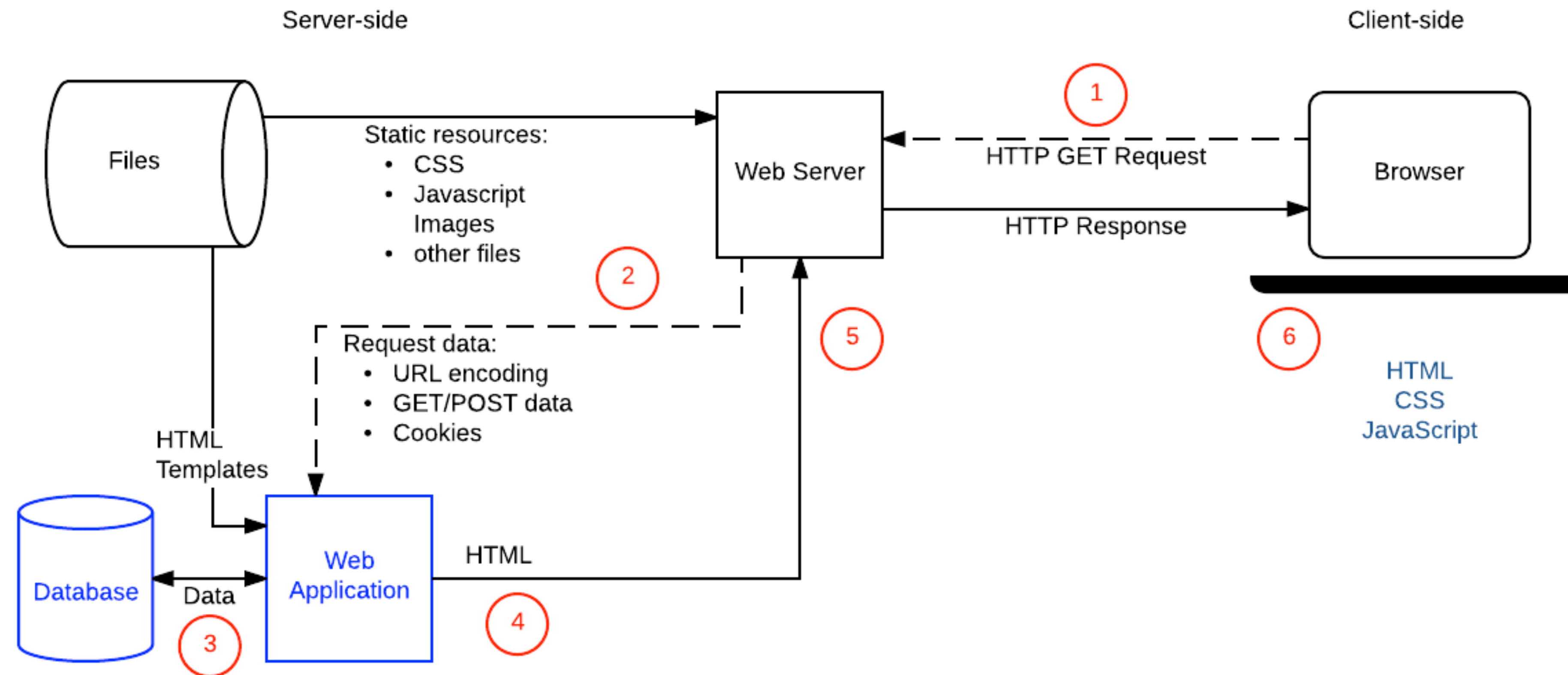
soup.a
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>

soup.find_all('a')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

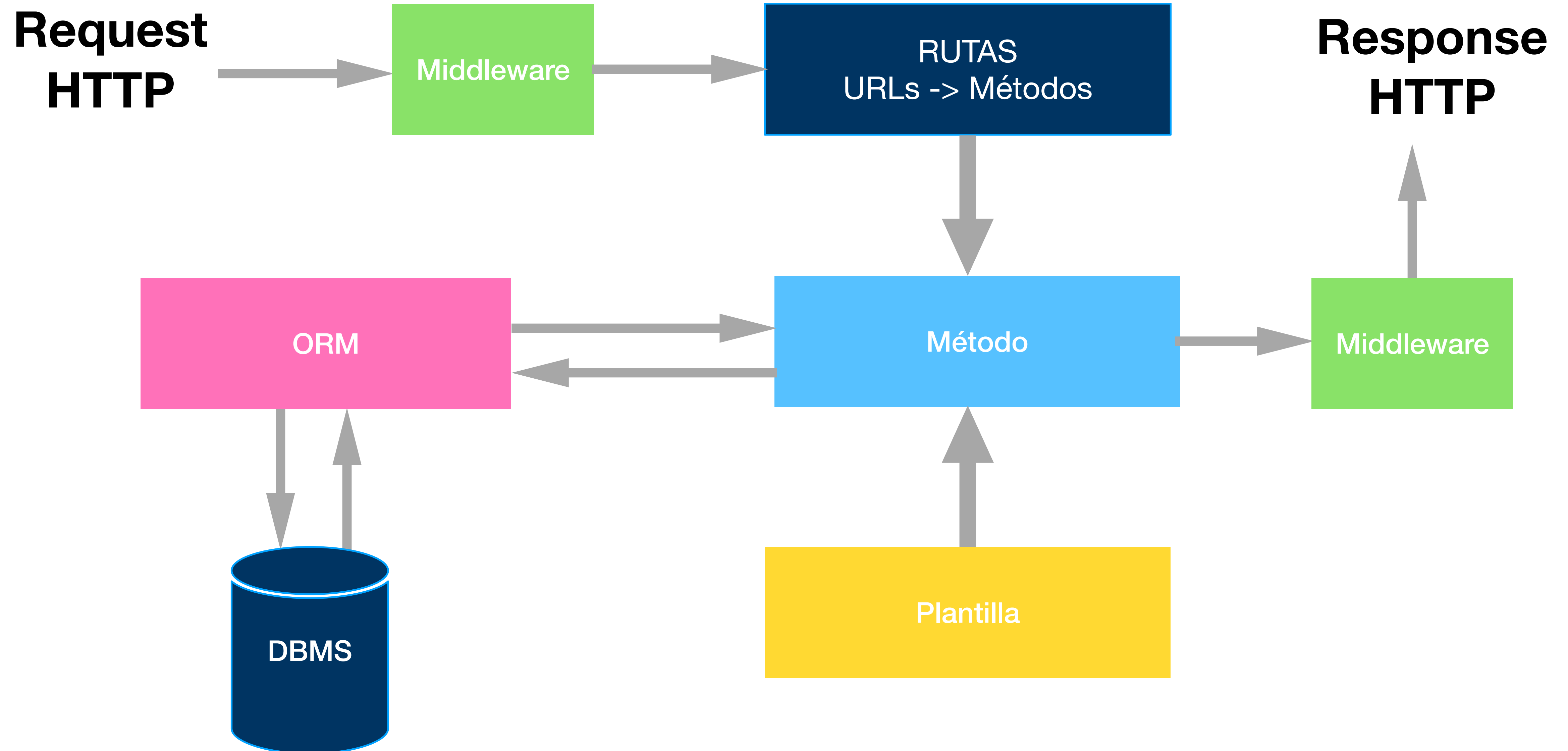
soup.find(id="link3")
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
```

# Web Frameworks Server-Side

## Componentes principales



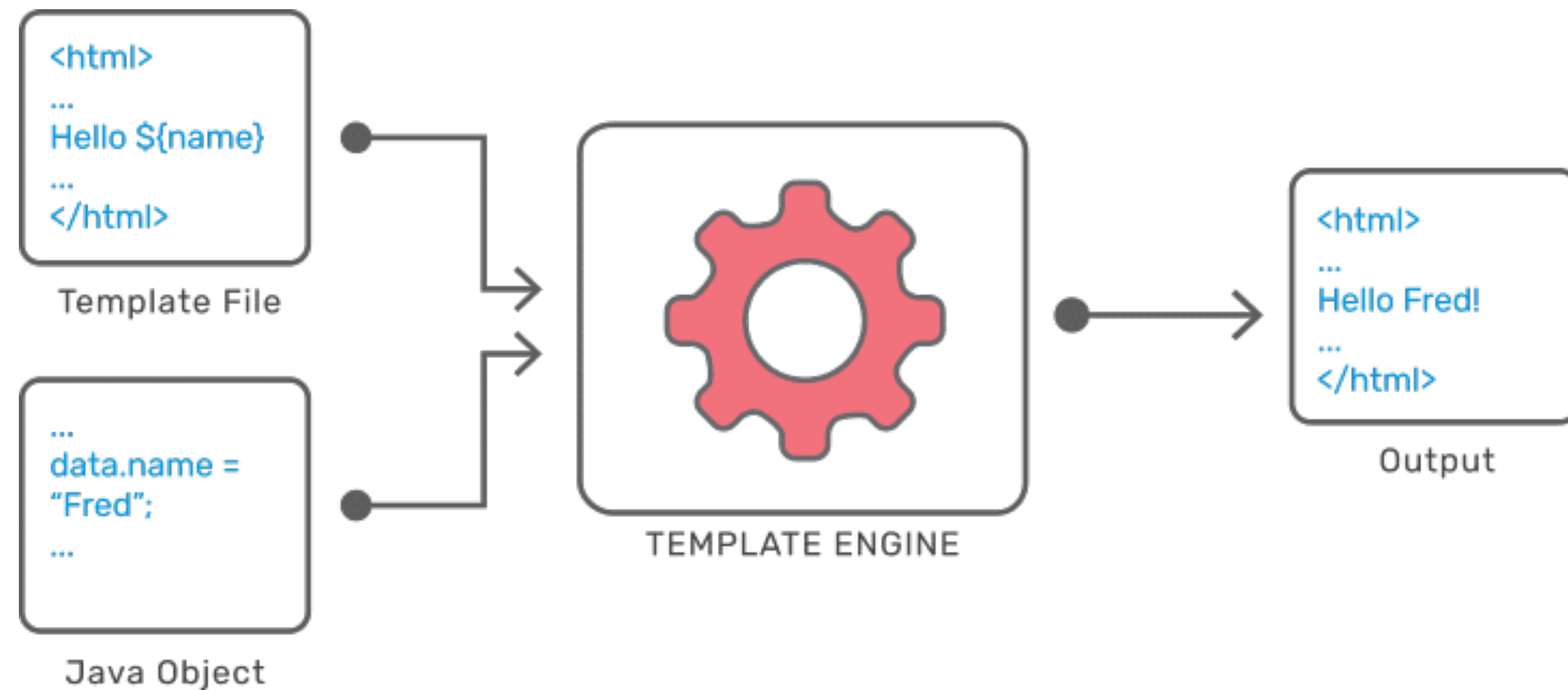
# URL Mapping y Middleware





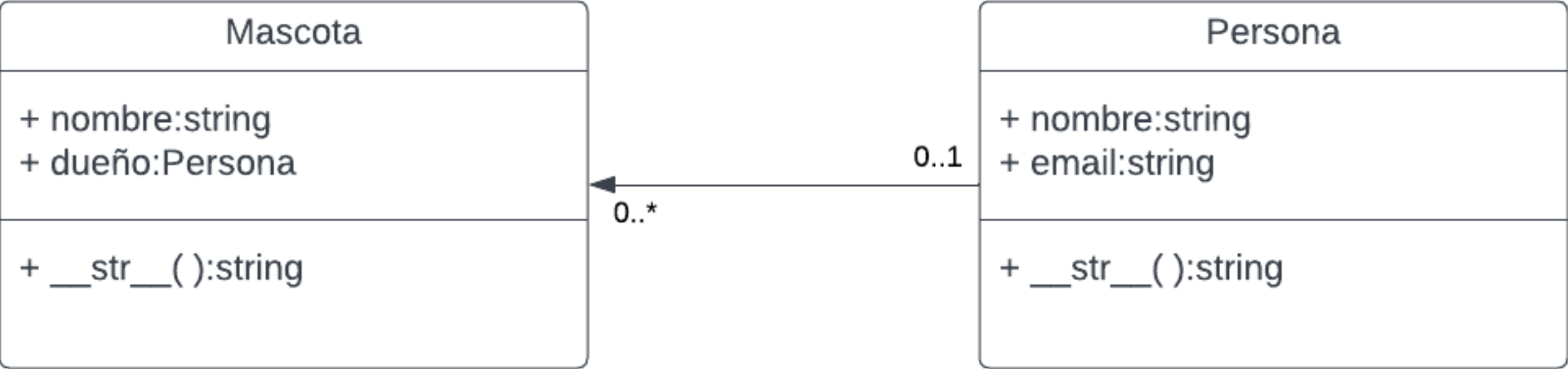
# Web Frameworks Server-Side

## Procesador de Plantillas



# ORM Object Relational Mapping

## Orientado a Objetos



```
ana = new Persona('Ana', 'ana@gmail.com')
solovino = new Mascota('Solovino', ana)
print(solovino.dueño.nombre) // 'Ana'
```

## Object Relational Mapping

Mascota

id	nombre	dueño
1	Fifí	1
2	Firulais	1
4	Solovino	<null>

Persona

id	nombre	email
1	Ana	ana@gmail.com
2	Tom	tom@gmail.com

```
SELECT p.nombre
FROM Persona p
JOIN Mascota m ON m.dueño = p.id
WHERE m.id = 4;
```

## Relacional

# Frameworks lado del Servidor

## Python

### Web

- Django
- Flask
- CherryPy
- Masonite
- FastAPI
- web2py
- TurboGears
- aiohttp

### Plantillas

- Genshi
- Jinja
- Mako
- Django templates
- Jinja2

### ORM

- SQLAlchemy
- Django ORM
- SQLAlchemy
- Peewee
- Masonite ORM

# Frameworks lado del Servidor

## Otros Lenguajes

### Web

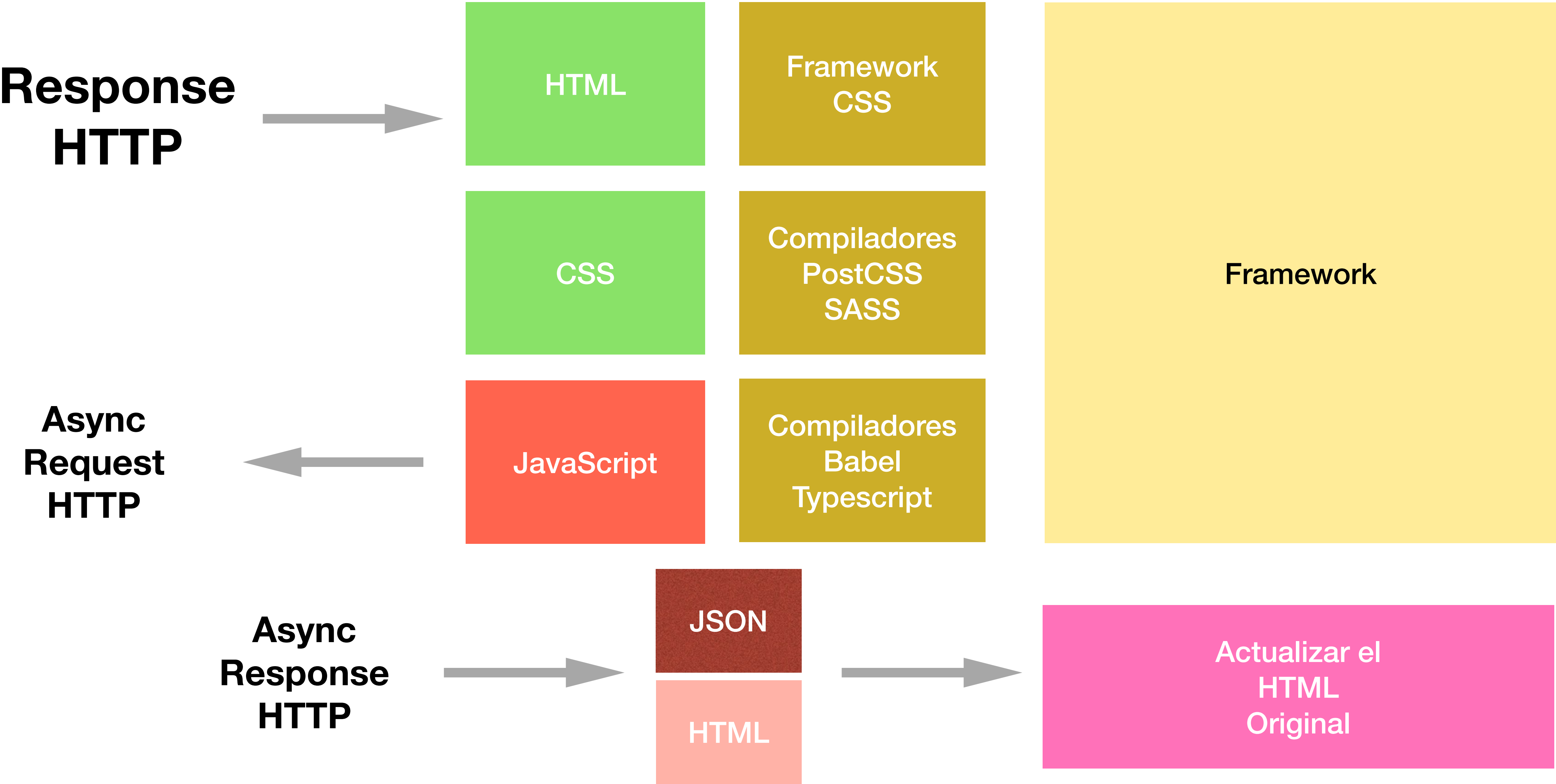
- Ruby on Rails (Ruby)
- [ASP.NET](#) (.NET)
- Vapor (Swift)
- Next.js (JS)
- Deno (JS)
- Laravel (PHP)
- Mojolicious (Perl)
- Phoenix (Elixir)
- Spring Boot (Java)

### Otras Herramientas

- Linters
- Sistemas de Control de Versiones
- Empaquetadores
- Contenedores
- Orquestadores
- Frameworks de Pruebas
- Automatización (Build Automation)
- Herramientas para despliegue (CD)
- Bitácoras



# Frameworks del lado del Cliente



# Actualizar el DOM (HTML) con JSON

## Ejemplo en *Vanilla* JavaScript

```
const state = [
  {
    id: "todo-0",
    name: "Learn some frameworks!",
  },
];
```

```
function buildTodoItemEl(id, name) {
  const item = document.createElement("li");
  const span = document.createElement("span");
  const textContent = document.createTextNode(name);

  span.appendChild(textContent);

  item.id = id;
  item.appendChild(span);
  item.appendChild(buildDeleteButtonEl(id));

  return item;
}
```

```
<ul>
  <li id="todo-0">
    <span>"Learn some frameworks!"</span>
    <button type="button">Delete</button>
  </li>
  ...
</ul>
```

```
function renderTodoList() {
  const frag = document.createDocumentFragment();
  state.tasks.forEach((task) => {
    const item = buildTodoItemEl(task.id, task.name);
    frag.appendChild(item);
  });

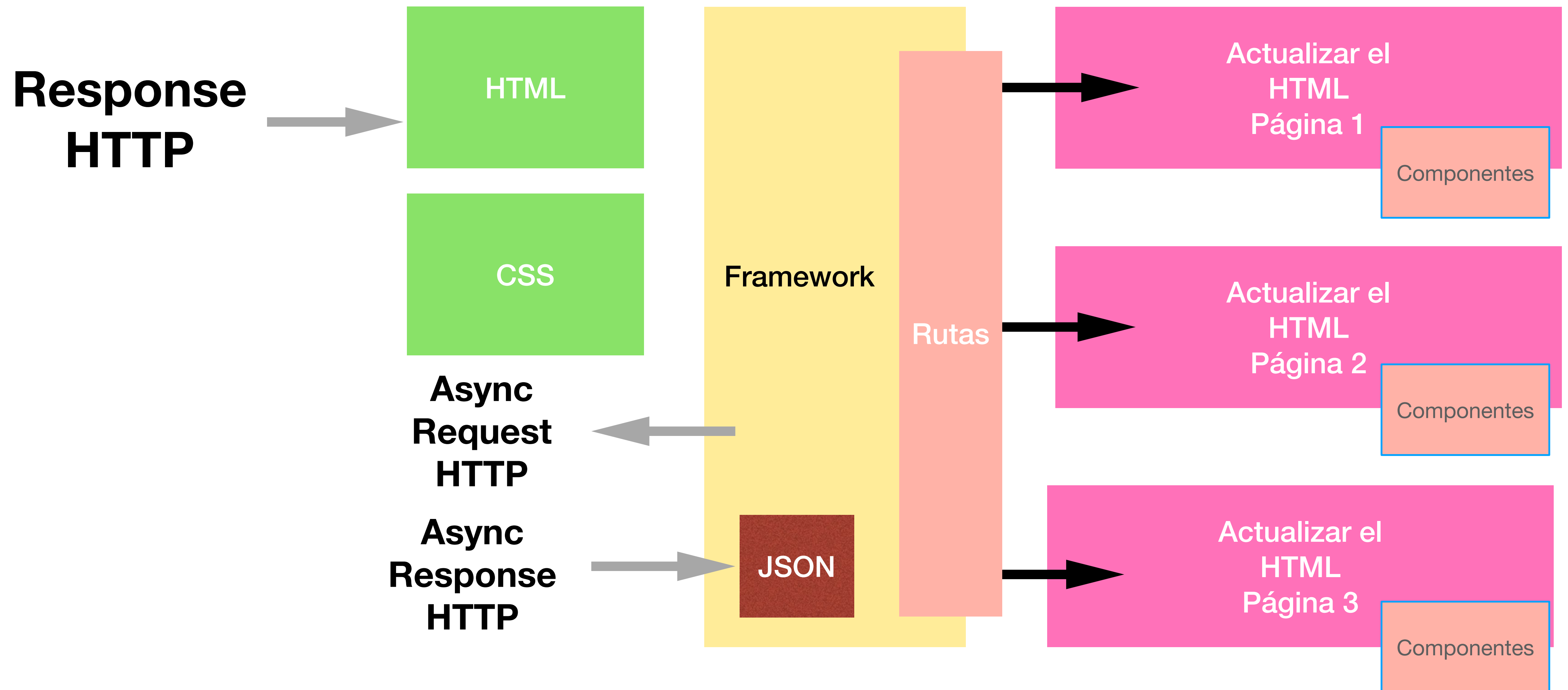
  while (todoListEl.firstChild) {
    todoListEl.removeChild(todoListEl.firstChild);
  }
  todoListEl.appendChild(frag);
}
```

# Actualizar el DOM (HTML) con JSON

## Ejemplo en *Vue*

```
<ul>
  <li v-for="task in tasks" v-bind:key="task.id">
    <span>{{task.name}}</span>
    <button type="button">Delete</button>
  </li>
</ul>
```

# Frameworks del lado del Cliente





# Frameworks del lado del Cliente

Framework	Browser support	Preferred DSL	Supported DSLs	Citation
Angular	Modern	TypeScript	HTML-based; TypeScript	<a href="#">official docs</a>
React	Modern	JSX	JSX; TypeScript	<a href="#">official docs</a>
Vue	Modern (IE9+ in Vue 2)	HTML-based	HTML-based, JSX, Pug	<a href="#">official docs</a>
Ember	Modern (IE9+ in Ember version 2.18)	Handlebars	Handlebars, TypeScript	<a href="#">official docs</a>

# Planificación de aplicaciones Web

# Manifiesto Ágil

Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

- **Individuos e interacciones** sobre procesos y herramientas
- **Software funcionando** sobre documentación exhaustiva
- **Colaboración con el cliente** sobre negociación contractual
- **Respuesta ante el cambio** sobre seguir un plan

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

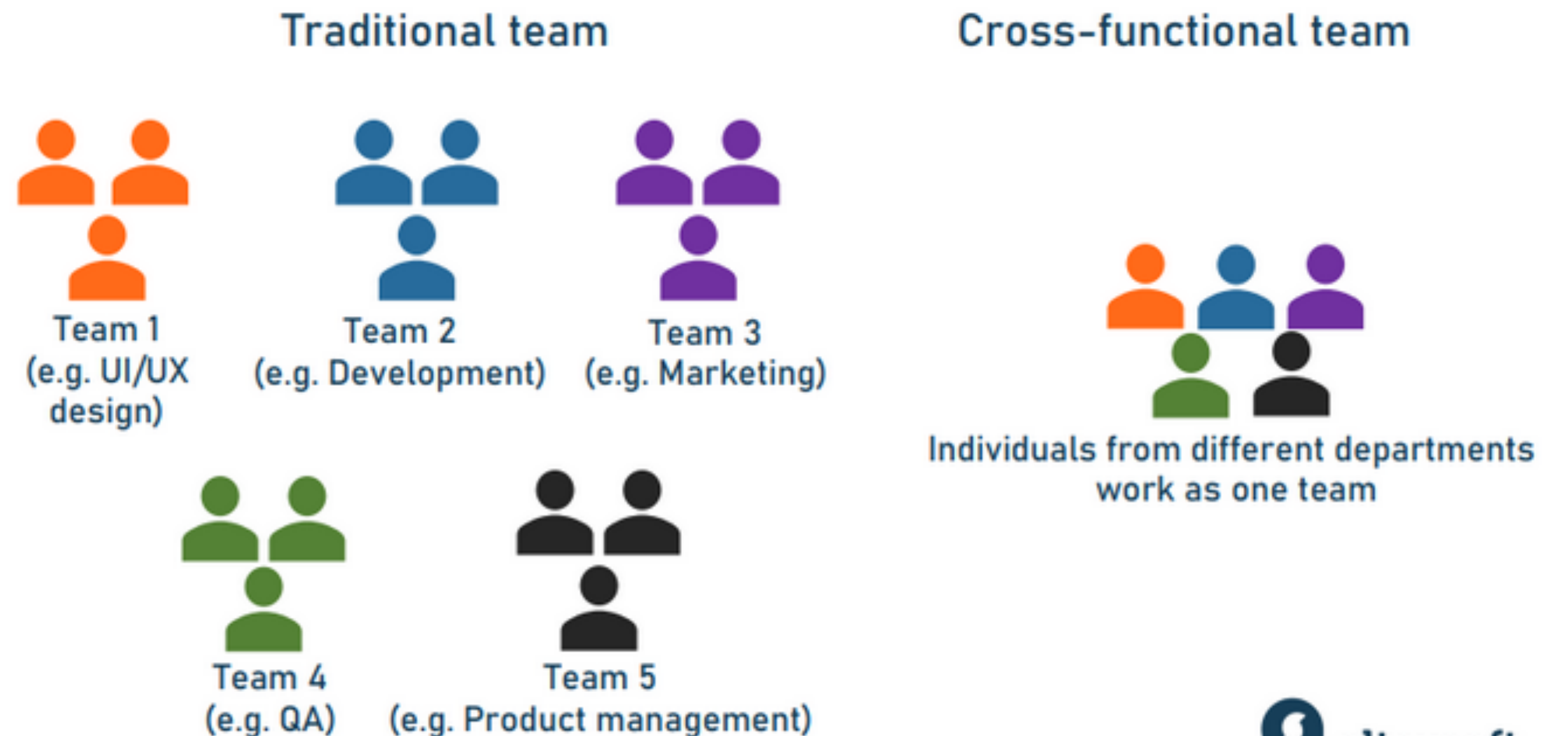
firmado por [Kent Beck](#), [Mike Beedle](#), [Arie van Bennekum](#), [Alistair Cockburn](#), [Ward Cunningham](#), [Martin Fowler](#), [James Grenning](#), [Jim Highsmith](#), [Andrew Hunt](#), [Ron Jeffries](#), [Jon Kern](#), [Brian Marick](#), [Robert C. Martin](#), [Steve Mellor](#), [Ken Schwaber](#), [Jeff Sutherland](#) y [Dave Thomas](#),<sup>1</sup>

# El equipo ágil

## Características

- Multidisciplinario
- Dedicados al equipo
- Esfuerzo colaborativo
- Con tiempo trabajando juntos

### TRADITIONAL TEAM & CROSS-FUNCTIONAL TEAM





# El equipo ágil

## Desde la organización

- El trabajo se asigna al equipo, no a individuos.
  - El equipo divide el trabajo en tareas y decide a que miembro asigna cada tarea.
  - Esto puede cambiar la manera en que se evalúa el desempeño individual.
- El equipo decide sus propios procesos. La administración puede establecer restricciones a los procesos, pero de manera justificada.
- Los equipos deciden sus procesos de:
  - Desarrollo
  - Construcción (Build Automation)
  - Prueba
  - Liberación

# Desarrollo Ágil

