

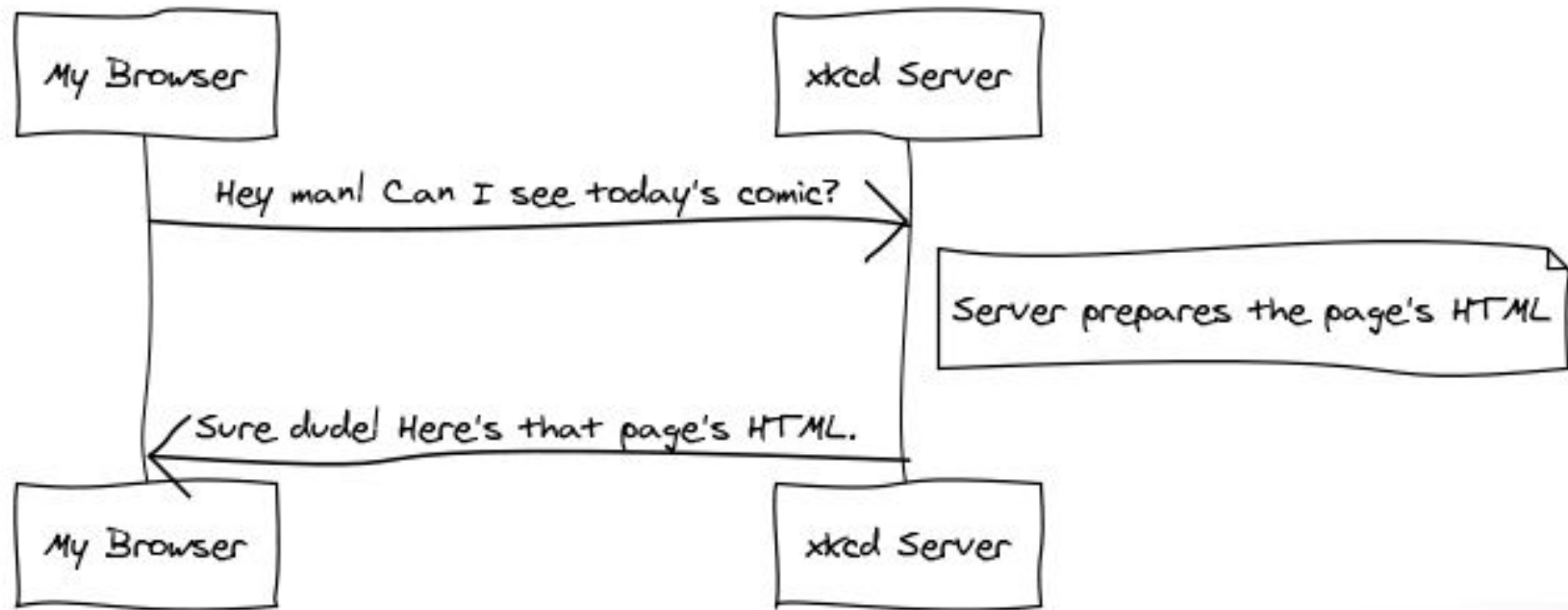
# Arquitectura cliente-Servidor

LAS BASES DE INTERNET

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev





*Hace peticiones, Recibe respuestas.*

**Cliente**

**DEV.F**

dev

El cliente pide los recursos al servidor, recibe la respuesta del servidor y se la muestra al usuario de manera adecuada.



*Recibe peticiones, envía respuestas.*

# Servidor

# Diferentes tipos de servidores.

- Servidores de base de datos.
- Servidores de correo electrónicos.
- Servidores de imágenes.
- Servidores WEB.



Servidor



Servidor de Correo



Servidor FTP



Servidor Web



Servidor Proxy



Servidor Base de Datos



Servidor Audio/Video



Servidor Chat



Servidor Groupware

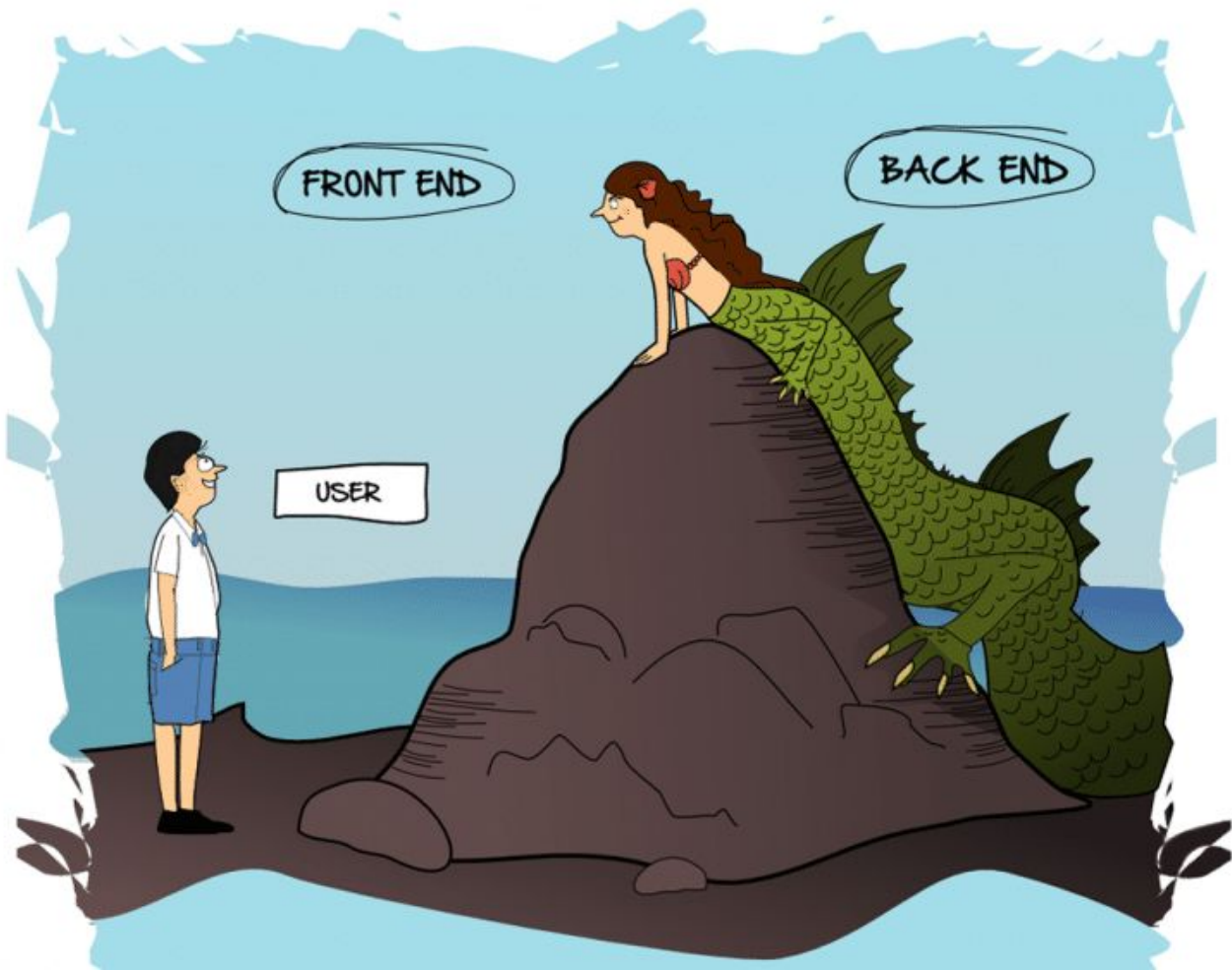


Cluster de Servidores







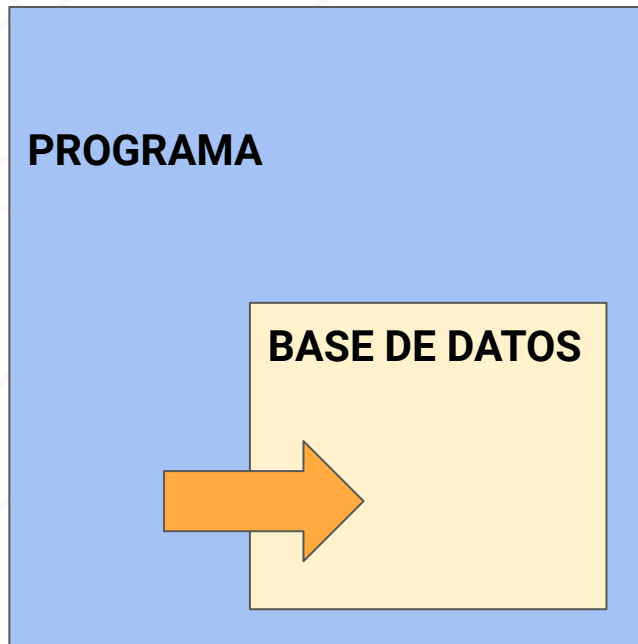


# Enviar datos al servidor

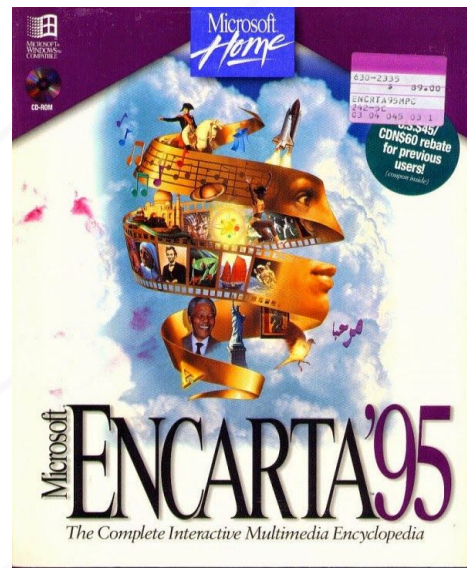
**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

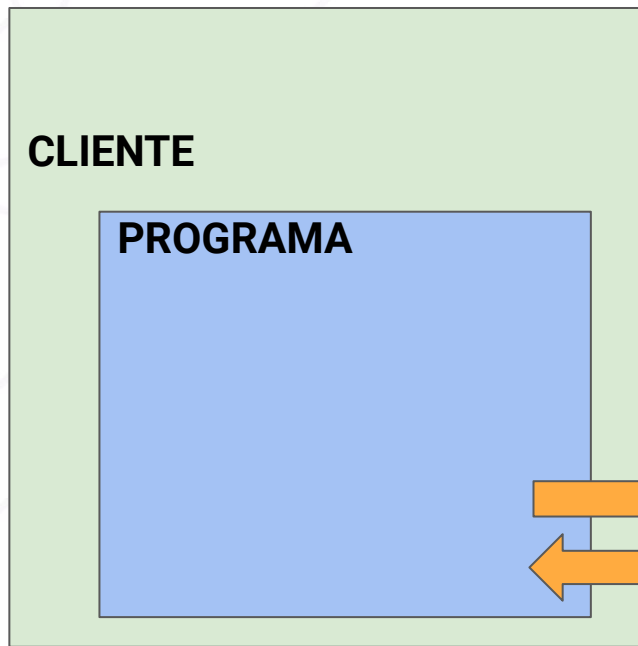
# Evolución de Acceso a la Información (1)



En los inicios, la aplicación convivía muy de cerca con su base de datos y ejecutándose en un mismo equipo.

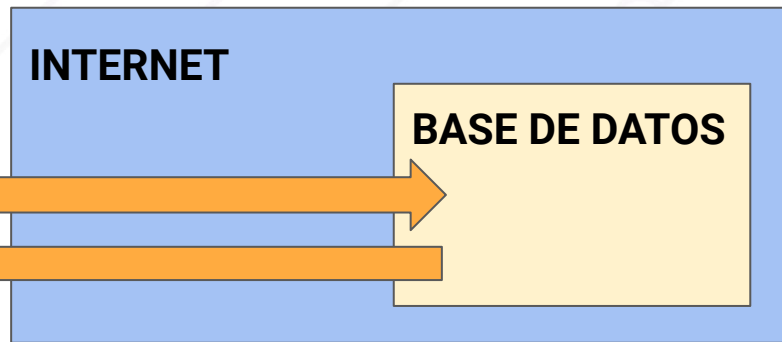


## Evolución de Acceso a la Información (2)



Los programas cada día comenzaban a necesitar cada vez más reglas de negocios.

Las bases de datos son primariamente para guardar y recuperar información, pero no para aplicar reglas de negocio.



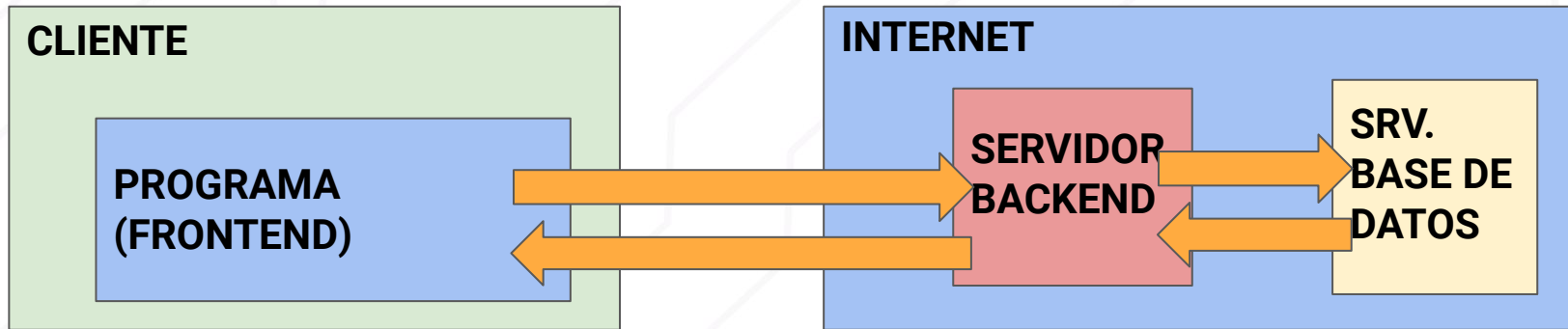
Sin embargo es poco seguro que el Cliente a través de internet acceda directamente a la base de datos.

# Evolución de Acceso a la Información (3)

Una solución fue escribir un programa **Backend** que acceda a la base de datos.

Normalmente **las bases de datos y el backend están servidores independientes**, y se configura la base de datos para que solo reciba instrucciones del backend (por ip).

De esta forma ahora el Backend es el programa encargado de validar la lógica de negocios y el acceso a la información de la base de datos.



# HTTP/HTTPS

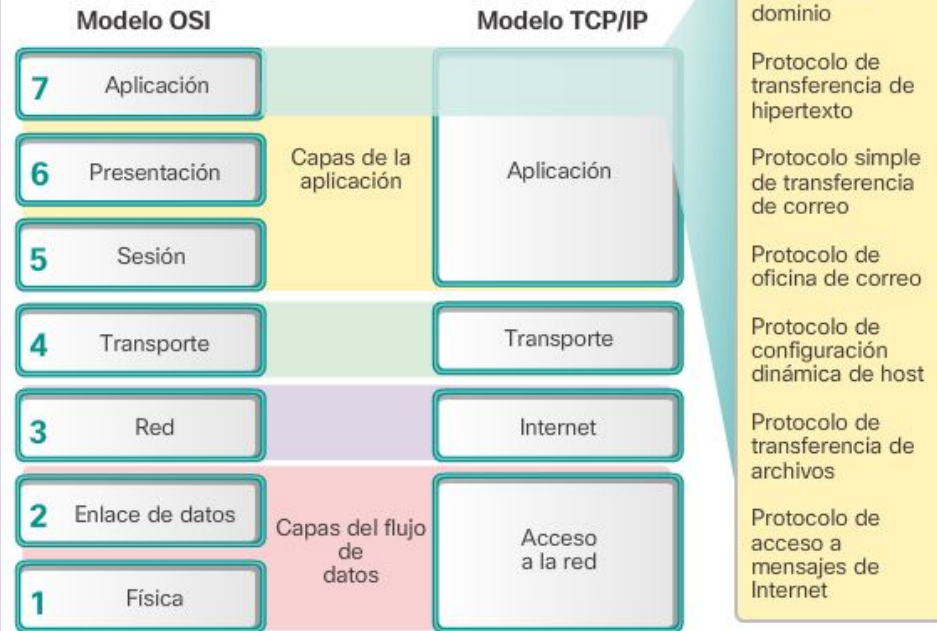
**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

# ¿Qué es capa de aplicación?

La capa de aplicación define las aplicaciones de red y los servicios de Internet estándar que puede utilizar un usuario. Estos servicios utilizan la capa de transporte para enviar y recibir datos. Existen varios protocolos de capa de aplicación. En la lista siguiente se incluyen ejemplos de protocolos de capa de aplicación:

- Servicios TCP/IP estándar como los comandos ftp, tftp y telnet.
- Servicios de nombres, como NIS o el sistema de nombre de dominio (DNS).
- Servicios de directorio (LDAP).
- Servicios de archivos, como el servicio NFS.
- Protocolo simple de administración de red (SNMP), que permite administrar la red.
- Protocolo RDISC (Router Discovery Server) y protocolos RIP (Routing Information Protocol).





# HTTP

Hypertext Transfer Protocol

Protocolo de Transferencia de Hipertexto en español es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML.

Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también.

Sigue el clásico modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta del mismo. Se trata de un protocolo sin estado, lo que significa que el servidor no guarda ningún dato (estado) entre dos peticiones.

# HTTPS

HyperText Transfer Protocol Secure

Protocolo seguro de transferencia de hipertexto en español es un protocolo de comunicación de Internet que protege la integridad y la confidencialidad de los datos de los usuarios entre sus ordenadores y el sitio web. Como los usuarios esperan que su experiencia online sea segura y privada, te recomendamos que adoptes HTTPS para proteger sus conexiones con tu sitio web, independientemente de lo que este contenga.

1)Cifrado

2)Integridad de los datos

3)Autenticación

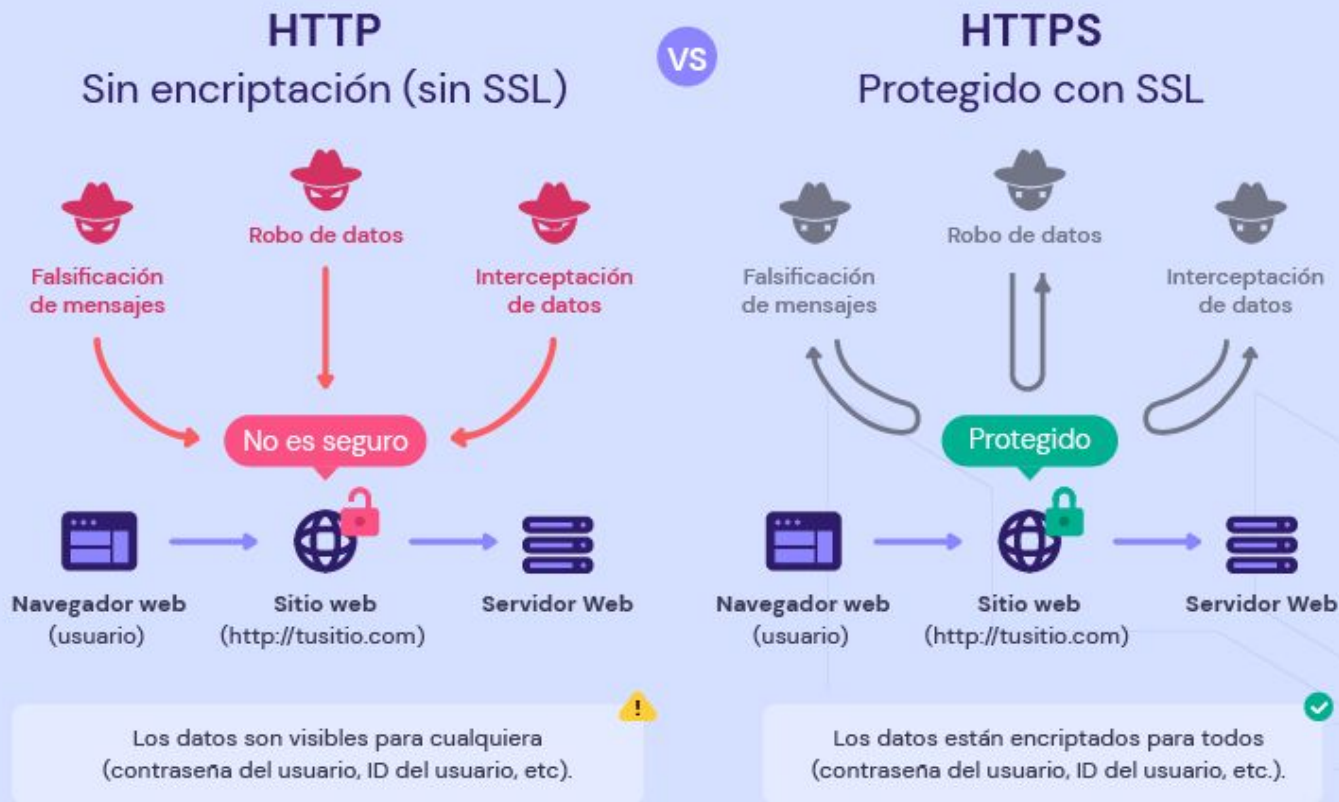
# ¿Qué es HTTP?

A grandes rasgos, puede decirse que la base de la world wide web es el protocolo HTTP, que es un protocolo de comunicación que define la manera en que se comunican un dispositivo cliente y un servidor conectados a través de la web. En cierta forma, puede decirse que HTTP es el idioma que hablan tus dispositivos informáticos para entenderse con las páginas web.

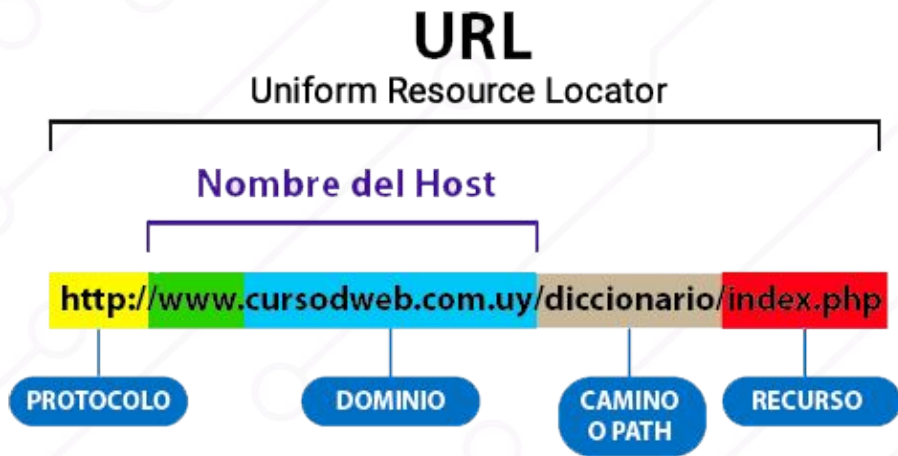
# ¿Qué es HTTPS?

El protocolo HTTPS es una evolución del protocolo HTTP diseñada para garantizar la seguridad de las comunicaciones en HTTP. Lo que hace este protocolo es combinar el protocolo HTTP con el protocolo SSL/TLS (Secure Socket Layer / Transport Layer Security) para otorgarle al protocolo HTTP una capa de protección que no tiene en su versión básica.

HTTPS surgió en 1995, cuando internet comenzó a transformarse en un espacio más complejo donde los usuarios podían intercambiar información sensible que era necesario proteger, como sus datos bancarios o sus claves de acceso personales a diferentes plataformas.



# URL



- Protocolo:** http, https, ftp, file...
- Subdominio**
- Dominio:** .com, .org, .edu... ccTLD, .com.uy, .com.ar, .es
- Path o camino al directorio del curso**
- Nombre de archivo:** .php, .htm, .html, .asp, .jsp...

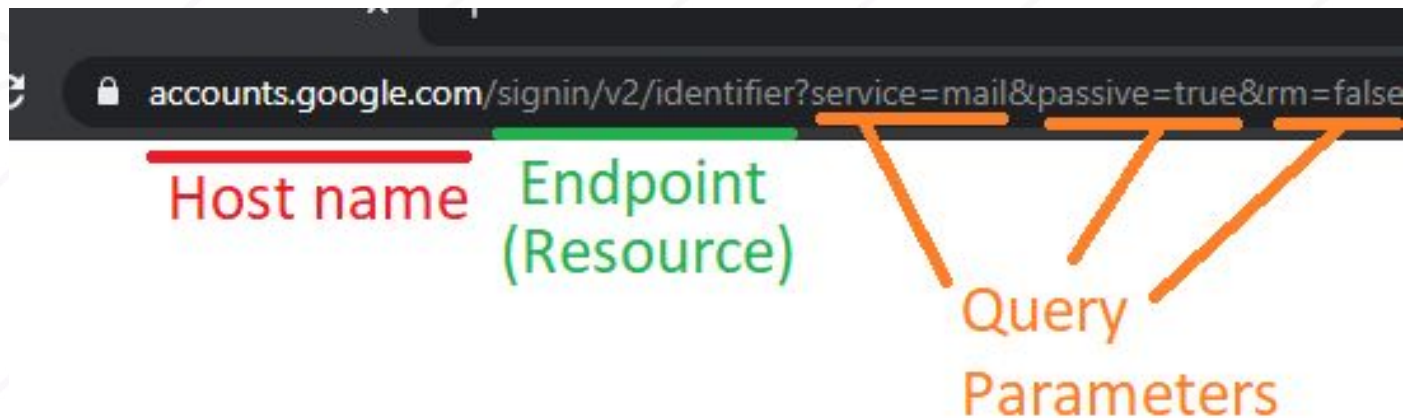
Las URL son términos para el frontend. Si lo que devuelve es información y es para el Backend nos referimos a estos como Endpoints.

Endpoints: URI que hacen referencia a una API.



# Query Parameters

Un query param es la clave valor (ejemplo: service=mail) que vemos al final de la URL, y como regla, siempre deberán estar después del símbolo de interrogación. Además, una URL puede tener N query params, separadas por &, cómo el siguiente ejemplo:

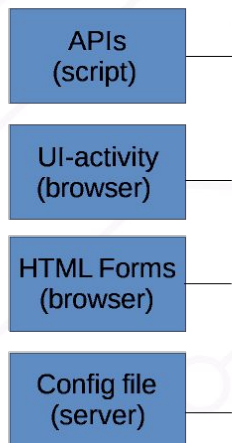


# Mensajes HTTP

Los mensajes HTTP, son los medios por los cuales se intercambian datos entre servidores y clientes.

Hay dos tipos de mensajes: **peticiones**, enviadas por el cliente al servidor, para pedir el inicio de una acción; y **respuestas**, que son la respuesta del servidor.

Activity initiation



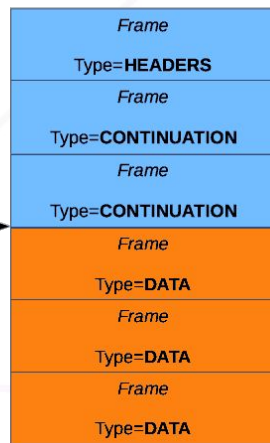
HTTP/1.x message

```
PUT /create_page HTTP/1.1
Host: localhost:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: text/html
Content-Length: 345
```

```
Body line 1
Body line 2
...
```

Binary framing

HTTP/2 stream  
(composed of frames)



# Mensajes HTTP

La línea de inicio y las cabeceras HTTP, del mensaje, son conocidas como la cabeza de la peticiones, mientras que su contenido en datos se conoce como el cuerpo del mensaje.

## Petición

Verbo      Recurso      Versión

↑      ↑      ↑

```
GET /index.html HTTP/1.1
Host: wikipedia.org
Accept: text/html
```

Primera línea

Encabezados

Cuerpo

## Respuesta

Versión      Código respuesta

↑      ↑

```
HTTP/1.1 200 OK
Server: wikipedia.org
Content-Type: text/html
Content-Lenght: 2026

<html>
...
</html>
```



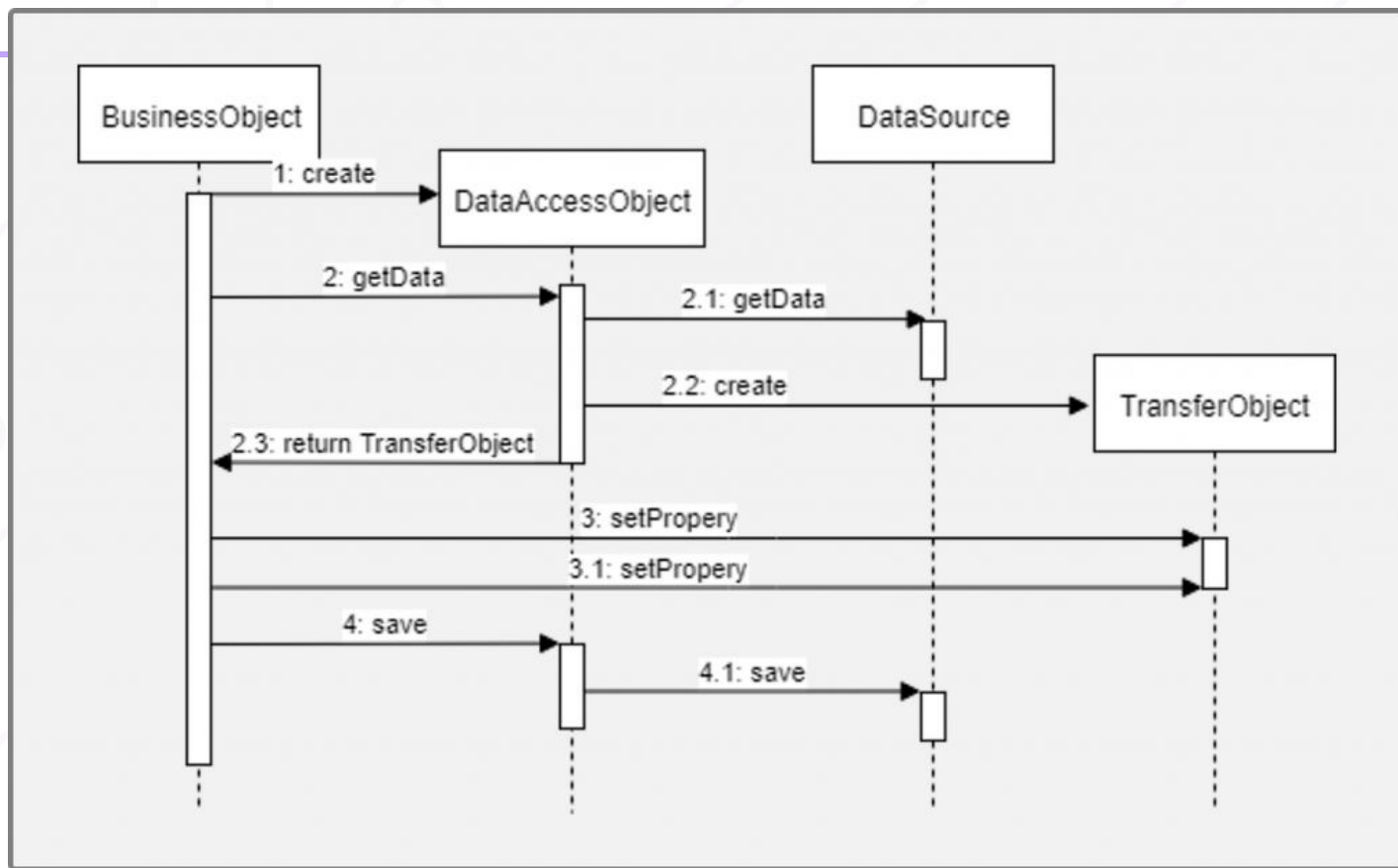
# DAO

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

# Data Access Object (DAO) Pattern

El patrón DAO propone separar por completo la lógica de negocio de la lógica para acceder a los datos, de esta forma, el DAO proporcionará los métodos necesarios para insertar, actualizar, borrar y consultar la información; por otra parte, la capa de negocio solo se preocupa por lógica de negocio y utiliza el DAO para interactuar con la fuente de datos.






**DEV.F.**  
DESARROLLAMOS(PERSONAS);

# CRUD

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev



CRUD (Create, Read, Update, Delete) es un acrónimo para las maneras en las que se puede operar sobre información almacenada.

CRUD usualmente se refiere a operaciones llevadas a cabo en una base de datos o un almacén de datos, pero también puede aplicar a funciones de un nivel superior de una aplicación como soft deletes donde la información no es realmente eliminada, sino es marcada como eliminada a través de un estatus.

La mayoría de las aplicaciones que utilizamos nos permiten añadir o crear nuevas entradas, buscar las existentes, realizar cambios en ellas o eliminarlas. Para ello, el CRUD ofrece muchas ventajas, entre ellas:

- Facilita el control de la seguridad para los distintos requisitos de acceso.
- Simplifica y facilita el diseño de la aplicación haciéndola más escalable.
- Tiene un mejor rendimiento en comparación con las sentencias SQL ad-hoc.



CREATE

C



READ

R



UPDATE

U



DELETE

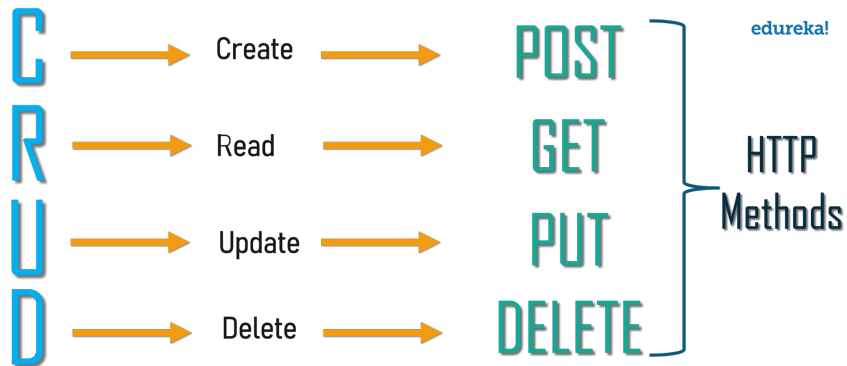
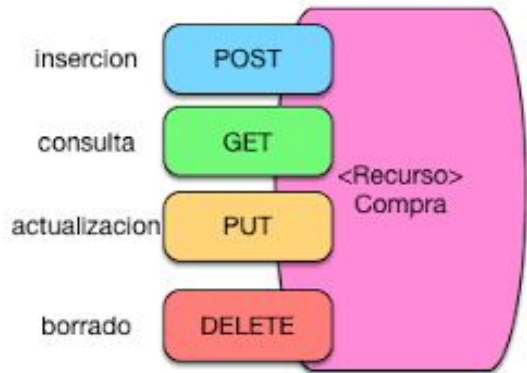
D

# Verbos HTTP

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev





# VERBOS HTTP

La primera línea de un mensaje de petición empieza con un verbo (también se le conoce como método). **Los verbos definen la acción que se quiere realizar sobre el recurso.**

Los verbos más comunes son:

- **GET:** Solicitar un recurso.
- **POST:** Publicar un recurso.
- **PUT:** Reemplazar un recurso.
- **DELETE:** Eliminar un recurso.
- **PATCH:** Actualizar un recurso

Nota: Cuando ingresas a una página desde un navegador, por debajo el navegador envía un mensaje GET, lo mismo cuando oprimas un vínculo a otra página.

# GET

*Pedir datos al servidor*

# POST

*Mandar datos al servidor*

# PUT

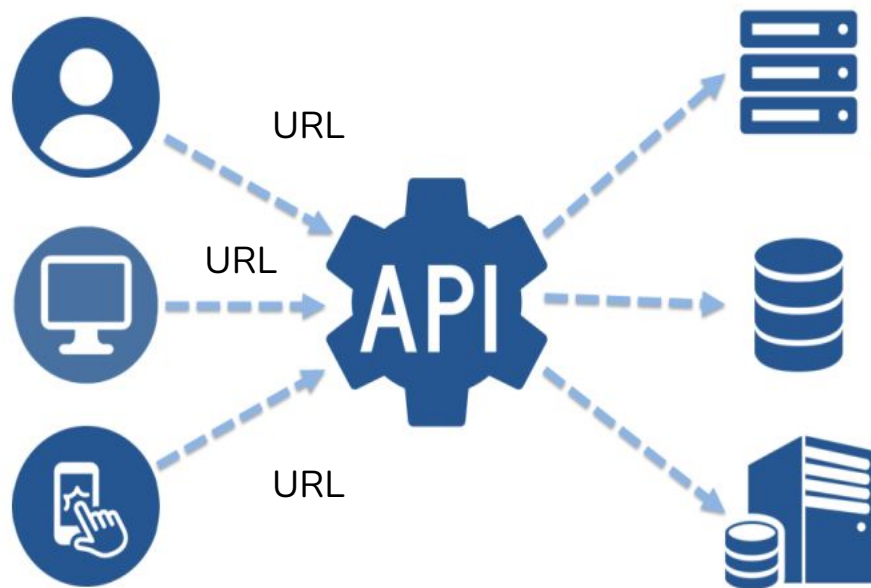
*Actualizar datos en el servidor*

# PATCH

*Actualizar datos en el servidor*

# DELETE

*Eliminar datos del servidor*

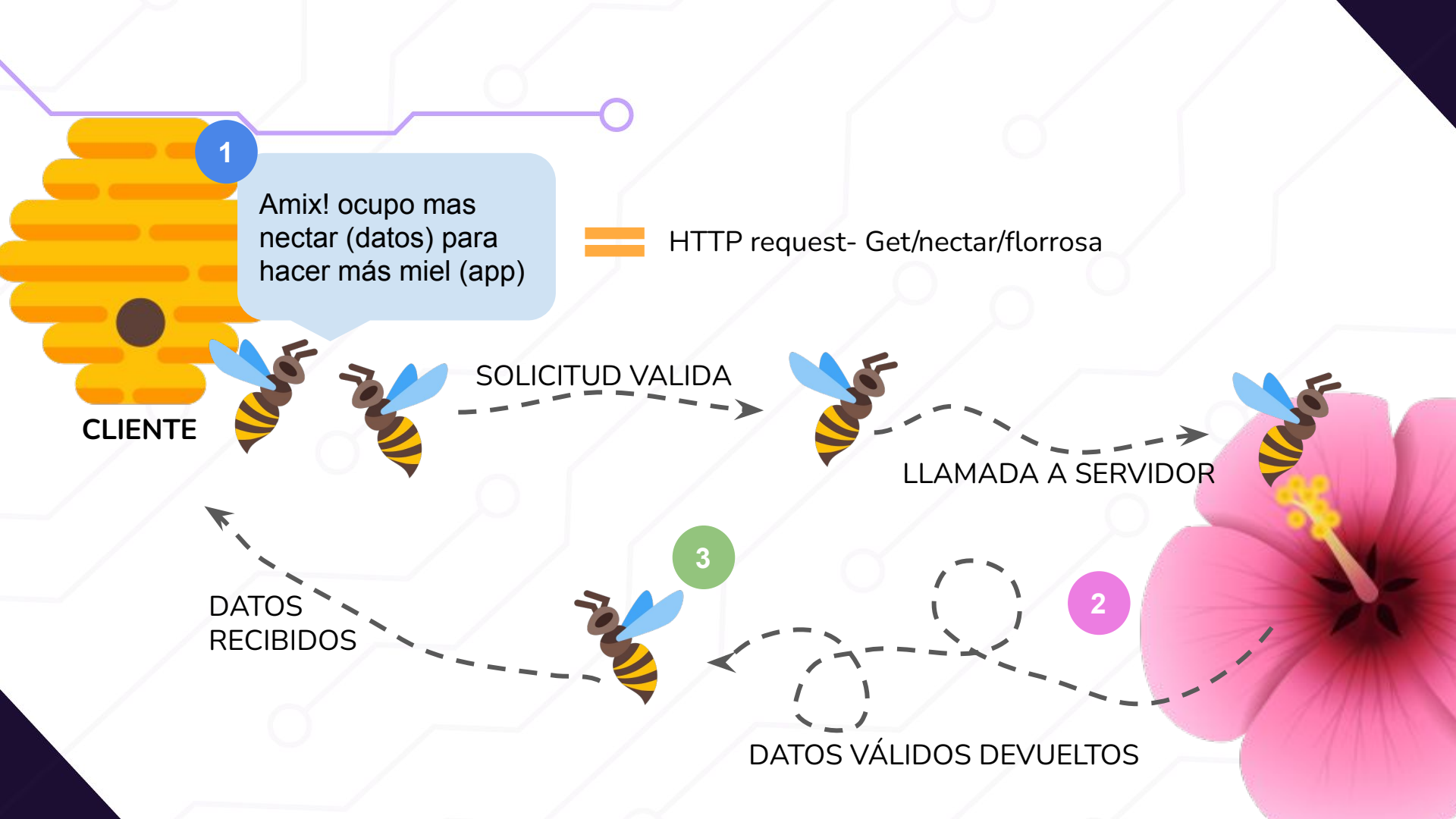


# API

## (Application Programming Interface)

Una interfaz de programación de aplicaciones (API) es un conjunto de herramientas, definiciones y protocolos que se utiliza para integrar los servicios y el software de aplicaciones.

Es lo que permite que sus productos y servicios se comuniquen con otros, sin tener que diseñar permanentemente una infraestructura de conectividad nueva.





The diagram illustrates the API request-response cycle using a bee analogy. A purple line at the top represents the flow of the process. Below it, three numbered steps are shown: 1. REQUEST (blue circle), 2. RECEIVE (pink circle), and 3. RESPONSE (green circle). Each step is followed by a description of the action. The background features a light purple circuit-like pattern with diagonal lines and circles.

1

### REQUEST

La llamada a API es iniciada por la aplicación cliente a través de una solicitud HTTP

2

### RECEIVE

La abeja trabajadora actúa como una API, que irá a la flor (servidor) para obtener néctar (datos)

3

### RESPONSE

La API transfiere la información solicitada de vuelta a la solicitud de la app, usualmente en formato JSON