

# Persistencia de la INFORMACIÓN



Iñigo Ezcurdia  
[inigofermin.ezcurdia@unavarra.es](mailto:inigofermin.ezcurdia@unavarra.es)

upna

Universidad Pública de Navarra  
Nafarroako Unibertsitate Publikoa

# Persistencia de datos: ¿Qué, dónde, cómo, cuándo, quién?

- Al igual que con cualquier software, para entornos web también debemos preguntarnos **qué** información deseamos que persista entre diferentes ejecuciones.
- Al trabajar con una arquitectura cliente-servidor, además debemos preguntarnos **dónde** queremos que persistan.
- Según la cantidad, tipo, estructura y fin de la información, deberíamos preguntarnos también el **cómo** conviene almacenarla y servirla.
- Ciertos tipos de información deben persistir, pero no eternamente. ¿**Cuándo** debe estar disponible esta información?
- Una misma información puede ser requerida por varios servicios o por varios usuarios bajo ciertos privilegios o ciertas limitaciones. ¿**Quién** debe tener acceso a esta información?

# ¿Qué información queremos que persista?

- Si simplemente guardamos información en variables (cliente o servidor) esta información únicamente existe en memoria en tiempo de ejecución. Después, desaparece.
- ¿Qué información queremos que sobreviva entre ejecuciones?

## Texto

「T」

- Usuarios, contraseñas, e-mails
- Mensajes
- Contenido de plantillas
- Logs
- Contenido textual
- Tokens de autenticación/sesión
- .html .css .js ...

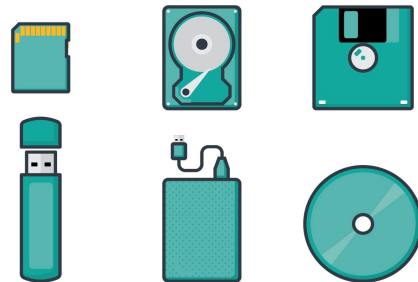
## Multimedia y ficheros



- Imágenes
- Video
- Audio
- Ejecutables
- Comprimidos
- Binarios
- ...

# ¿Cómo? → Opción A) En ficheros

- Guardamos esta información en ficheros en dispositivos de almacenamiento.
- Aprovechamos el **sistema de ficheros** del servidor que los aloja.
- Apropiado para multimedia y para funcionalidades con ficheros grandes (~5MB)



## Ventajas

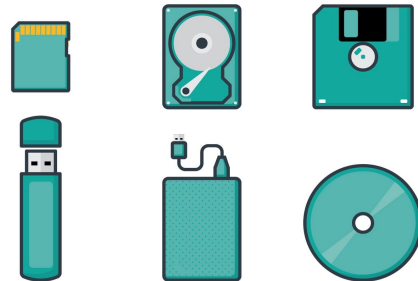
- Pagar más GB de almacenamiento bruto suele ser más **barato** que pagar más capacidad de base de datos.
- Sencilla **migración** de datos. Copiar en otro destino y fijarse en los permisos de la carpeta.
- Mejor **rendimiento** en ciertos escenarios. Ficheros grandes almacenados como blobs en bases de datos generan problemas de rendimiento.
- Añadir, borrar y descargar ficheros es una tarea trivial y **eficiente delegada** al sistema operativo.

## Desventajas

- **No** se garantiza una **estructura** o relación y consistencia entre ficheros. (sin software adicional)
- **No** se garantizan principios **ACID** (Atomicity, Consistency, Isolation, Durability)
- Mayores **riesgos de seguridad** y más propensos a subida de ficheros maliciosos al servidor.

# ¿Cómo? → Opción A) En ficheros

- Guardamos esta información en ficheros en dispositivos de almacenamiento.
- Aprovechamos el **sistema de ficheros** del servidor que los aloja.
- Apropiado para multimedia y para funcionalidades con ficheros grandes (~5MB)



## **Alternativas en la nube**

Podemos confiar nuestros ficheros a sistemas de almacenamientos en la nube para delegar así:

- Su almacenamiento.
- Su seguridad.
- Sus backups.
- Su distribución desde datacenters más cercanos al usuario final. (CDN - Content Delivery Network)
- Su escalabilidad.
- Su rendimiento.

Tienen un coste adicional. Se paga por cantidad de peticiones y volumen de tráfico.

Ofrece APIs rest para que su uso sea lo más transparente posible de cara a la aplicación.

**Lider actual:** [Amazon S3 - Amazon Simple Storage Service](#) Objetos almacenados en “buckets”.

# ¿Cómo? → Opción B) Bases de datos



- Información almacenada de manera **organizada y estructurada**.
- Apropiado para información textual e interconectada. Representación de relaciones más complejas.
- Algunos **Sistemas Gestores de Bases de Datos** (SGBD) pueden almacenar binarios como blobs. (Binary Large Objects)
- Históricamente, los SGBD más populares son **relacionales** y emplean SQL (Structured Query Language).
- También existen alternativas no-relacionales, noSQL.

## Ventajas

- Principios **ACID** (Atomicity, Consistency, Isolation, Durability)
- Relaciones, dependencias y **estructuras** más complejas.
- Uso de primary/foreign keys para crear **dependencias** y evitar información *huérfana*.
- Mayor control de la **seguridad** y privilegios de acceso y operaciones permitidas.
- Posibilidad de **indexación** para aumentar su rendimiento.
- En SGBD relacionales, **consultas** complejas mediante SQL

## Desventajas

- Necesidad de convertir a **BLOBS** para almacenar binarios.
- Mayor probabilidad de **duplicidad** de datos.
- Fuerte dependencia de **memoria RAM**, ya que la información de las queries se vuelca en memoria. ¿Qué ocurre con consultas que deben filtrar u ordenar MUCHA información?
- Mayor **coste** que simplemente almacenamiento.
- **Dependencia** de la herramienta escogida. Mayores dificultades para migrar entre tecnologías.

# ¿Cómo? → Opción B) Bases de datos



- Información almacenada de manera **organizada y estructurada**.
- Apropiado para información textual e interconectada. Representación de relaciones más complejas.
- Algunos **Sistemas Gestores de Bases de Datos** (SGBD) pueden almacenar binarios como blobs. (Binary Large Objects)
- Históricamente, los SGBD más populares son **relacionales** y emplean SQL (Structured Query Language).
- También existen alternativas no-relacionales, noSQL.

## **Alternativas en la nube**

- Mismos motivos que en el almacenamiento en la nube de ficheros.
- Tienen un coste adicional. Se paga por cantidad de peticiones y volumen de tráfico.

**Líderes actuales:**



[Amazon Relational Database Service \(Amazon RDS\)](#)

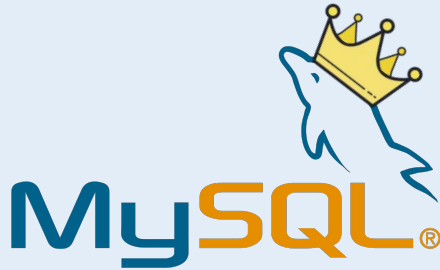


[Microsoft Azure SQL Database](#)



[Google Cloud SQL](#)

# SGBD populares



**Client-sided.** No server.  
Self-Contained  
Un solo fichero .sqlite  
No dependencias



## MariaDB

OpenSource  
Fork de MySQL  
Compatible con MySQL  
Corrección automática de  
tipos de entrada  
Soporta JSON para  
unstructured data.  
Para DBs ligeras



## PostgreSQL

OpenSource  
SQL+NoSQL  
(Relational + NoRelational)  
Soporta JSON  
Triggers, procedures...



## elasticsearch

Distribuido.  
Escalabilidad y replicación.  
Schema-less.  
Document-based  
Structured+Unstructured Data



## mongoDB

Document-based  
Agregaciones y filtros  
Transacciones ACID (limitadas)  
Escalabilidad



## redis

Clave-Valor, hashes, lists, bitmaps  
En memoria -> Velocidad  
Amazon Web Services  
Popular para sesiones, transacciones  
y chats.

Relacionales

NoRelacionales

teacher	
PK	teacher_id
FK	course_id
FK	dept_name
FK	dept_name
FK	language_1
FK	language_2
FK	title
FK	dept_name
FK	dept_name
FK	dept_name

course	
PK	course_id
FK	dept_name
FK	language_1
FK	language_2
FK	course_length_seconds
FK	dept_name
FK	dept_name
FK	dept_name
FK	dept_name
FK	dept_name

id	name	title	location
42	Inigo	Teacher	UPNA

{
"name": "Inigo",
"title": "Teacher",
"location": "UPNA",
}



# ¿Dónde? → Almacenamiento por parte del cliente

- En las anteriores diapositivas hemos hablado de persistencia en el lado del servidor, tanto autogestionado como en la nube.
- También podemos contemplar el **almacenamiento** de cierta información por parte del dispositivo **del cliente**.
- Útil para ciertas funcionalidades. (Sesiones, caché de cliente, configuraciones locales como tema oscuro o accesibilidad... )

**NO SUSTITUYE AL ALMACENAMIENTO POR PARTE DEL SERVIDOR**

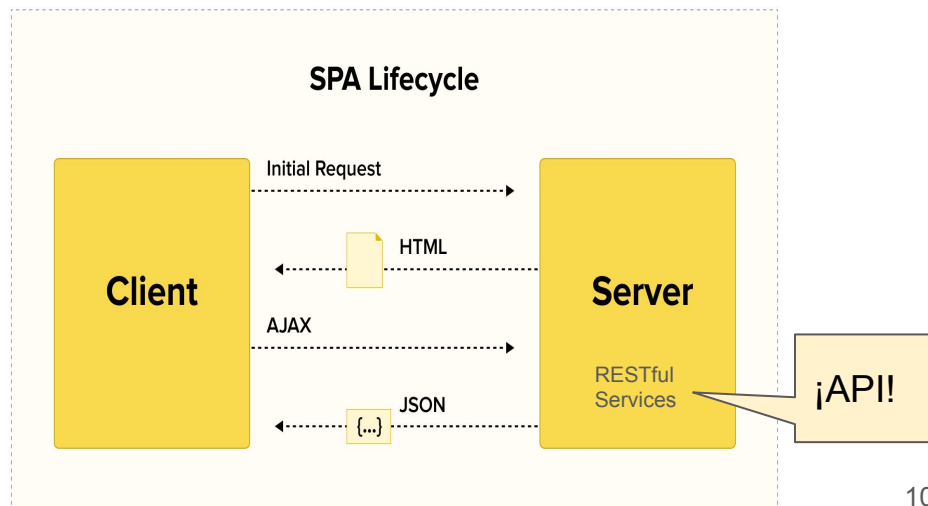
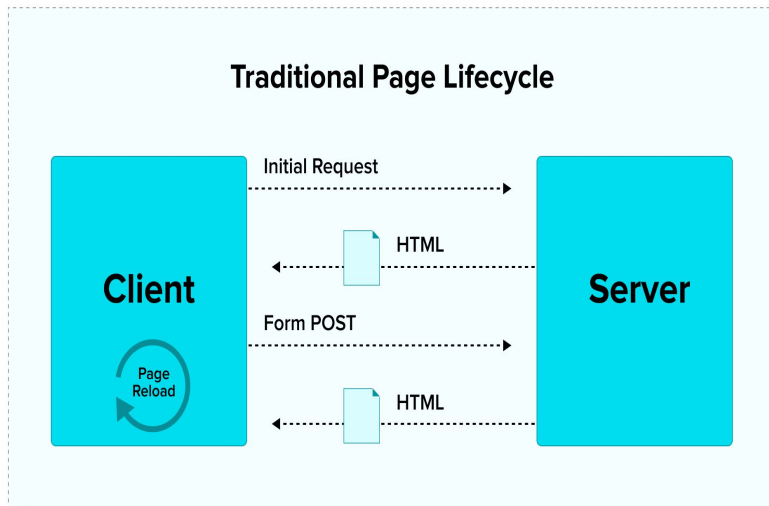
**El cliente tiene control total sobre sus contenidos → NO CONFIAMOS EN SU VERACIDAD**

- Indispensable para Single Page Applications (SPAs) y Progressive Web Applications (PWAs)

# ¿Dónde? → Almacenamiento por parte del cliente

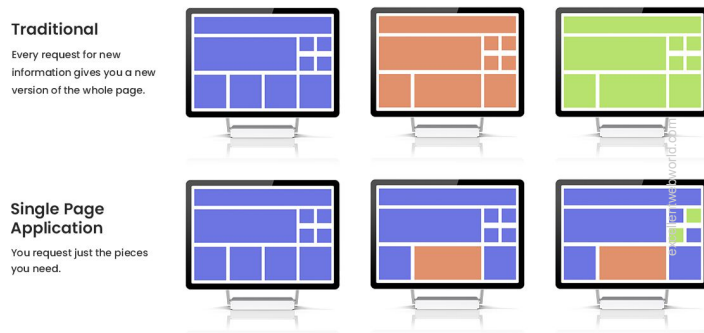
## Single Page Application (SPA)

- En la primera petición se carga todo el HTML/CSS/JS necesario. Toda navegación y **actualización** de la página se realiza **mediante AJAX/Fetch**.
- El usuario **NO recarga el navegador**. Se actualizan únicamente los elementos necesarios.



# ¿Dónde? → Almacenamiento por parte del cliente

## Single Page Application (SPA)



### VENTAJAS

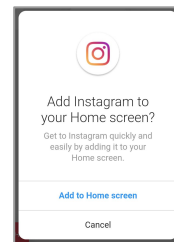
- Cargas más ligeras y rápidas. (Excepto la primera)
- Robustez offline, siempre y cuando no se requiera actualizar elementos nuevos.
- Experiencia de usuario más consistente.
- Desarrollo frontend muy independiente de backend.

### DESVENTAJAS

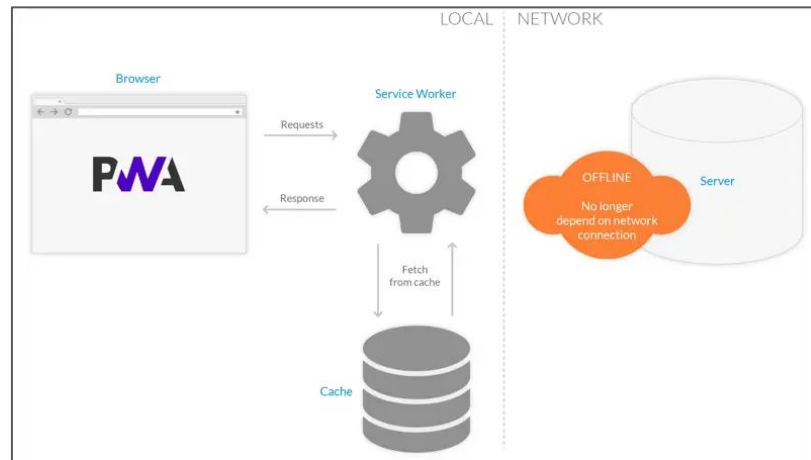
- Dificultades de cara al SEO, ya que operamos desde una única URL, una única página.
- Toma de analíticas más compleja.
- Pulsar “atrás” en el navegador no proporciona *por defecto* al usuario el comportamiento deseado.

# ¿Dónde? → Almacenamiento por parte del cliente

## Progressive Web App (PWA)



- Suponen un paso más para acercar un servicio web a la experiencia **App Móvil** o **Software de Escritorio**.
- Un concepto parecido a SPA, pero va más lejos, emplean service workers y no evita que el backend envíe vistas ya renderizadas.
- No son incompatibles, muchas PWA también son SPA.
- Basadas en:
  - **Service Workers:** [Scripts](#) encargados de interceptar y controlar las peticiones, caché y notificaciones push. Se ejecutan en sus propios hilos de ejecución ajenos al principal de javascript.
  - **App Manifest:** Un fichero JSON con [metadatos](#) sobre la app, como si de una store de apps se tratase.
  - **App Shell:** (Opcional). El esqueleto HTML/CSS/JS que utilizábamos con las SPA.



# ¿Dónde? → Almacenamiento por parte del cliente

## Progressive Web App (PWA)

### VENTAJAS

- Rapided, fluided, experiencia de usuario mejorada.
- Posible alternativa para el desarrollo de apps multiplataforma.
- Usuarios actualizados automáticamente. Incluso en background.
- Sin costes de distribución (Google Play o Apple Store)
- Resiliencia a uso offline.
- Notificaciones PUSH.
- Pueden ejecutarse a pantalla completa.
- Afectan a métricas que pueden potenciar el SEO.
- Acceso a USB, bluetooth, sistema de ficheros, cámara, giroscopio y otras características del dispositivo. ([Ver más](#))

### DESVENTAJAS

- Más complejas y costosas de desarrollar.
- Problemas SEO... si las utilizamos para implementar SPA.

#### **¿Quieres saber más?**

##### Bibliografía recomendada sobre PWAs

- [MDN Web Docs - Aplicaciones Web Progresivas](#)
- [Google Progressive Web Apps Training](#)

# ¿Dónde? → Almacenamiento por parte del cliente

## Singe Web Applications (SPA) & Progressive Web Application (PWA)

### EJEMPLOS DE SPA



### EJEMPLOS DE PWA



- Frameworks, bibliotecas, plataformas y librerías JS comúnmente utilizados para implementar SPAs y PWAs:



# ¿Dónde? → Almacenamiento por parte del cliente

Entonces... ¿De qué herramientas disponemos para almacenar información en el cliente?

## ● WebStorage API:

- Bloquean el hilo principal
- Tamaño muy limitado
- Solo strings
- No accesibles desde Service Workers

### ➤ LocalStorage

Persistente

### ➤ SessionStorage

Caduca al cerrar la pestaña

## ● Cache API

- Cacheo de peticiones/respuestas de red.
- Cacheo también de objetos que no son respuestas de red.
- Trabaja con Service Workers
- No bloquean el hilo principal
- Tamaño virtualmente infinito
- Estandar



## ● IndexedDB

- Databases
- Información estructurada noSQL
- Indexado
- Mas eficiente que WebStorage para grandes cantidades de información.
- Asíncronía, sin bloquear el hilo principal.
- Transacciones: Grupos de operaciones
- Primary keys



## ● Cookies

- Solo strings
- Se anexan a cada petición HTTP!
- No accesibles desde Service Workers
- Utilizadas sobretodo para mantener sesiones.

## ● File and Directory Entries API

- Asíncronía, sin bloquear el hilo principal
- Ficheros binarios y grandes.
- Sandboxed: No tiene acceso directo al sistema de ficheros del usuario.

## ● File System Access API

- Ficheros binarios y grandes.
- Acceso al sistema de ficheros del usuario.
- Requiere permiso explícito por parte del usuario.

# ¿Dónde? → Almacenamiento por parte del cliente

## ● WebStorage API:

- Bloquean el hilo principal
- Tamaño muy limitado
- Solo strings
- No accesibles desde Service Workers

### ➤ LocalStorage

Persistente

### ➤ SessionStorage

Caduca al cerrar la pestaña

```
// set key
localStorage.test = 2;
// Or
localStorage.setItem("test", 2);

// get key
alert( localStorage.test ); // 2
// Or
alert( localStorage.getItem("test") );

// remove key
delete localStorage.test;
// Or
localStorage.removeItem("test");
```

```
sessionStorage.user = JSON.stringify({name: "John"});

// Más tarde...
let user = JSON.parse( sessionStorage.user );
alert( user.name ); // John
```



# ¿Dónde? → Almacenamiento por parte del cliente

## Cache API

- Cacheo de peticiones/respuestas de red.
- Cacheo también de objetos que no son respuestas de red.
- Trabaja con Service Workers
- No bloquean el hilo principal
- Tamaño virtualmente infinito
- Estandar



```
if ('caches' in window) {  
  const cacheName = 'my-website-cache'  
  const url1 = '/resource1'  
  const url2 = '/resource2'  
  caches.open(cacheName).then(cache => {  
    cache.addAll([url1, url2]).then(() => {  
      //all requests were cached  
    })  
  })  
}
```



```
// Retrieve data.json from the server and store the response.  
cacheName.put('/data.json');
```

```
// Create a new entry for test.json and store the newly created  
response.  
cacheName.put('/test.json', new Response('{"foo": "bar"}'));
```

```
// Retrieve data.json from the 3rd party site and store the response.  
cacheName.put('https://example.com/data.json');
```

## Breve inciso: array.find() y Arrow functions


- `array.find(funcionBusqueda)` devuelve el primer elemento del array que satisfaga la función de búsqueda
- Se recorre el array, elemento a elemento, y para cada uno de ellos que se ejecuta la función de búsqueda, hasta que se devuelva true para un elemento.

```
var info = "apellido=EzcurdiaInigo;ciudad=Pamplona";
info = info.split(";")
info.find(function(elemento) {
    return elemento.startsWith('apellido=');
});
```

## Breve inciso: array.find() y Arrow functions

- Sintaxis simplificada. Prescinden de 'function' 'return' y paréntesis de parámetros (si solo hay uno).

```
var info = "apellido=EzcurdiaInigo;ciudad=Pamplona";  
info = info.split(";")  
info.find(function(elemento) {  
    return elemento.startsWith('apellido=');  
});
```



```
var info = "apellido=EzcurdiaInigo;ciudad=Pamplona";  
info = info.split(";")  
info.find(elemento => elemento.startsWith('apellido='));
```

## Breve inciso: array.find() y Arrow functions

- Sintaxis simplificada. Prescinden de 'function' 'return' y paréntesis de parámetros (si solo hay uno).
- No tienen su propio "this" o "super". (Por lo que no pueden ser utilizadas como métodos en OOP)
  - Lo cual puede ser útil en algunos casos en los que el "this" hace referencia a la entidad contenedora.

```
function Timer() {  
  this.seconds = 0;  
  setInterval(() => {  
    this.seconds++;  
    console.log(this.seconds);  
  }, 1000);  
}  
new Timer();
```

# MiniReto: Visor de imágenes optimizado mediante caché

1. Simulamos una conexión de red precaria -> Activamos el las dev tools el Network Throttling.
2. Abrimos y observamos <https://jsfiddle.net/afrog7xm/>
3. **Visor estandar:** Crea un visor de imágenes mediante botones de “anterior” y “siguiente”

⚠ Fíjate en cómo las URL de las imágenes identifican la imagen mediante un ID entero positivo. Explota esta característica para navegar entre las imágenes.



## MiniReto: Visor de imágenes optimizado mediante caché

1. Simulamos una conexión de red precaria -> Activamos el las dev tools el Network Throttling.
2. Abrimos y observamos <https://jsfiddle.net/afrog7xm/>
3. **Visor estandar:** Crea un visor de imágenes mediante botones de “anterior” y “siguiente”
4. **Visor con caché:** Añade otros dos botones que también permitan retroceder y avanzar  
Pero esta vez, cada vez que se carga una imagen, también se utiliza la cache API para precargar la imagen anterior y siguiente.
5. Una vez implementado:
  - a. Observa las diferencias en cuanto velocidades de carga y experiencia de usuario
  - b. Analiza el comportamiento desde la pestaña Network
  - c. Analiza los elementos cacheados desde la pestaña Application

# ¿Dónde? → Almacenamiento por parte del cliente

## IndexedDB

- Databases
- Información estructurada noSQL
- Indexado
- Mas eficiente que WebStorage para grandes cantidades de información.
- Asíncronía, sin bloquear el hilo principal.
- Transacciones: Grupos de operaciones
- Primary keys



```
// This works on all devices/browsers, and uses IndexedDBShim as a final fallback
var indexedDB = window.indexedDB || window.mozIndexedDB || window.webkitIndexedDB || window.msIndexedDB ||
window.shimIndexedDB;
// Open (or create) the database
var open = indexedDB.open("MyDatabase", 1);
// Create the schema
open.onupgradeneeded = function() {
    var db = open.result;
    var store = db.createObjectStore("MyObjectStore", {keyPath: "id"});
    var index = store.createIndex("NameIndex", ["name.last", "name.first"], { unique: false } );
};
open.onsuccess = function() {
    // Start a new transaction
    var db = open.result;
    var tx = db.transaction("MyObjectStore", "readwrite");
    var store = tx.objectStore("MyObjectStore");
    var index = store.index("NameIndex");
    // Add some data
    store.put({id: 12345, name: {first: "John", last: "Doe"}, age: 42});
    store.put({id: 67890, name: {first: "Bob", last: "Smith"}, age: 35});
    // Query the data
    var getJohn = store.get(12345);
    var getBob = index.get(["Smith", "Bob"]);
    getJohn.onsuccess = function() {
        console.log(getJohn.result.name.first); // => "John"
    };
    getBob.onsuccess = function() {
        console.log(getBob.result.name.first); // => "Bob"
    };
    // Close the db when the transaction is done
    tx.oncomplete = function() {
        db.close();
    };
}
```

## MiniReto: App de notas mediante indexedDB

1. Abre el siguiente enlace y examina la plantilla inicial.  
<https://jsfiddle.net/y457dpc0/>
2. Completa los huecos ??? para lograr un uso correcto de indexedDB
  - a. El usuario puede añadir notas que seguirán ahí cuando regrese posteriormente a la página.



# ¿Dónde? → Almacenamiento por parte del cliente

- Cookies

- Solo strings
- Se anexan a cada petición HTTP!
- No accesibles desde Service Workers
- Utilizadas sobretodo para mantener sesiones.



```
// Javascript (Client-Side)
```

```
document.cookie = "yummy_cookie=choco";
```

```
/* PHP (Server side) */
```

```
setcookie("yummy_cookie", "choco", time()+3600); /* expire in 1 hour */
```

```
// NodeJS (Server-Side)
```

```
response.setHeader('Set-Cookie', ['yummy_cookie=choco']);
```

```
// Python (Server-Side)
```

```
from http import import cookies
```

```
myCookies = cookies.SimpleCookie()
```

```
myCookies["yummy_cookie"] = "choco"
```

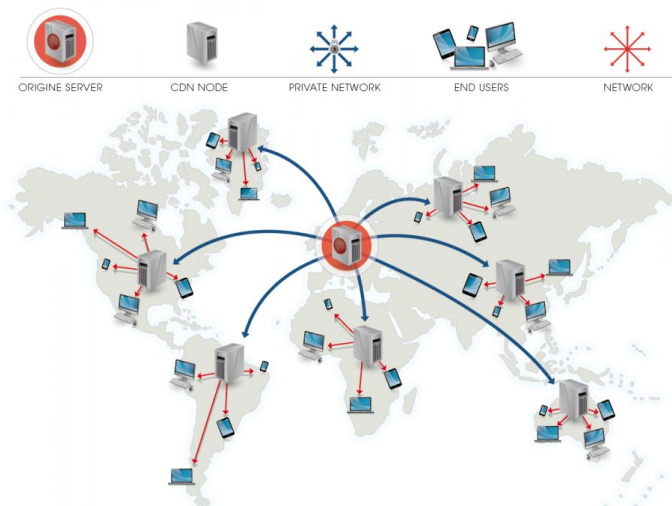
## MiniReto: Preferencia de color mediante cookies

1. Copia la siguiente web en visual studio y abrela con live server
  - a. <https://jsfiddle.net/58qLpt16/>  
(Este ejemplo no funciona en jsfiddle o codepen por motivos de seguridad)
2. Modifícalo para que, **mediante el uso de cookies**, el usuario pueda guardar su nombre y color favorito.
3. Haz que al cargar la página, si hay cookies disponibles, se rellenen automáticamente los inputs.
4. Utiliza las pestañas application y network de las dev tools para analizar cómo se almacena y viaja la información de las cookies con CADA PETICIÓN del path definido.

# ¿Dónde? → Almacenamiento de frontera

- El cliente puede almacenar y cachear ficheros para reutilizarlos ...
- El servidor puede cachear respuestas a peticiones populares ...
- ¿Algún lugar más en el que almacenar y cachear? ¡En la frontera!

## Content Delivery Network (CDN)



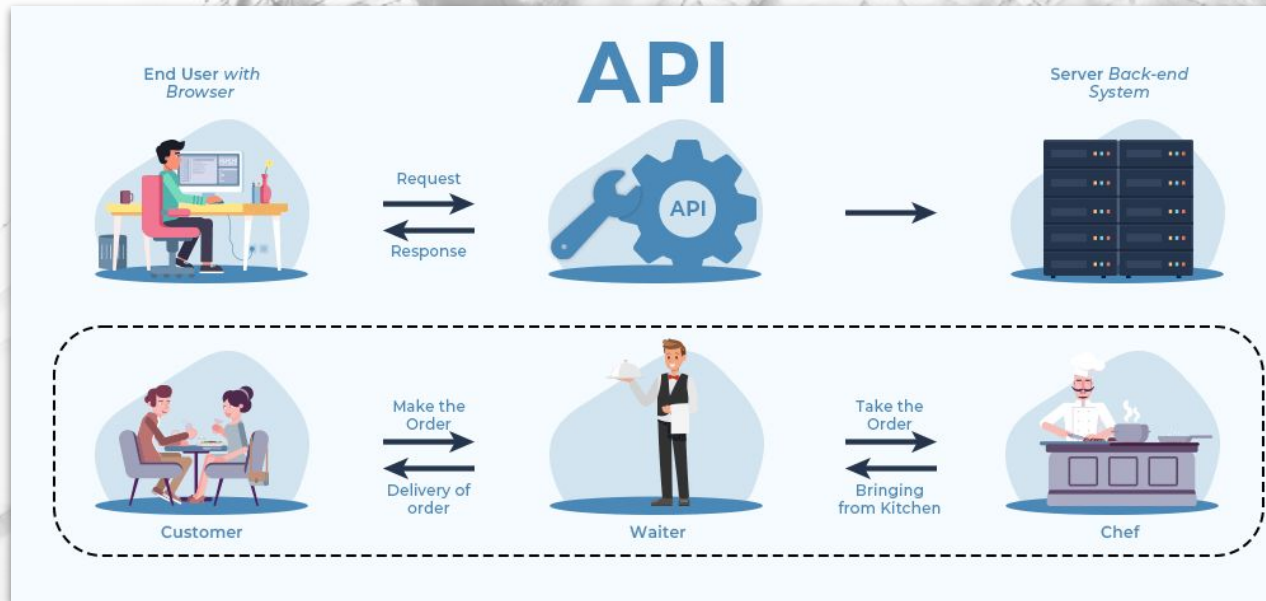
- Se replica el contenido de tu servidor original (o copias cacheadas de las respuestas) en distintos nodos.
- El nodo disponible más cercano al cliente será quien atienda la petición.
- Seguridad y protección ante [ataques DDoS](#).
- Balanceo de carga

Los CDNs más populares



# Application Programming Interfaces

## APIs



# ¿Qué son las APIs?

- Application Programming Interface
- Conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre sí.
- Servicio intermediario entre distintas aplicaciones.
- Ej:



# APIs en Frontend

- **APIs Públicas:** Accesibles públicamente. (clima, noticias, finanzas, wikis...)
- **APIs REST:**
  - Funcionan utilizando el estándar de los verbos HTTP. Las respuestas también.
  - Utilizan identificadores uniformes de recursos.
  - Sin estados.

	CRUD	Ejemplo	Idempotente (siempre mismo resultado)
GET	Obtener recurso	GET /user GET /user/1	Sí
POST	Crear un recurso	POST /user?id=3	No
PUT	Actualizar un recurso	PUT /user/3?name=newname	Sí
DELETE	Eliminar un recurso	DELETE /user/3	Sí


# Consumir una API para front

- Ejemplo javascript:

```
fetch('https://api.ejemplo.com/data')  
  .then(response => response.json())  
  .then(data => console.log(data));  
// Y trabaja con el JSON devuelto como puedas/quieras!
```

- Ejemplo completo con pokeapi: <https://jsfiddle.net/ye1p2dcn/>
  - Con REACT consumiremos las APIs con un método similar, pero transformando los resultados a componentes de interfaz!

# ¿Y si la API no es pública?

- Hay muchas APIs que son privadas...
  - Cobrando por su servicio. O limitando el nº y tipo de peticiones.
  - Son únicamente de uso interno.
-  **API tokens:** Claves únicas utilizadas en las cabeceras para autenticar solicitudes.

```
fetch('https://api.ejemplo.com/data', {  
  headers: { 'Authorization': 'Bearer YOUR_API_TOKEN' }  
})  
.then(response => response.json())  
.then(data => console.log(data));
```

- **Cuidado!** Estas exponiendo tu API\_TOKEN y cualquiera podría robarla...
  - Servidor intermediario.
  - Servidores API con políticas de CORS (Cross-Origin Resource Sharing)
  - Tokens de corta duración



# ¿Quieres crear tu propia API?

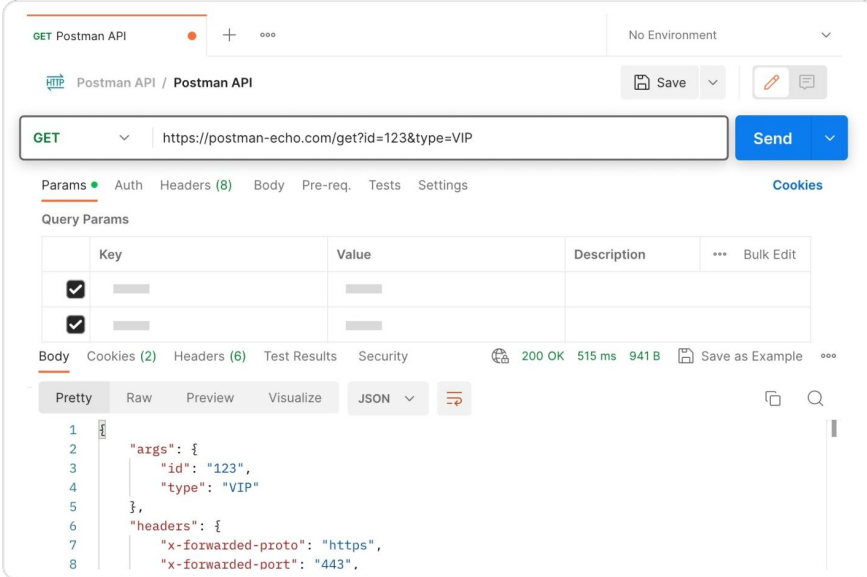
## Desarrollo

Python -> [Flask](#)

```
2 import pymysql
3 from app import app
4 from db_config import mysql
5 from flask import jsonify
6 from flask import flash, request
7 from werkzeug import generate_password_hash, check_password_hash
8
9 @app.route('/user')
10 def user():
11     try:
12         conn=mysql.connect()
13         cur = conn.cursor(pymysql.cursors.DictCursor)
14         cur.execute("SELECT* from user;")
15         rows=cur.fetchall()
16         resp=jsonify(rows)
17         resp.status_code=200
18         return resp
19     except Exception as e:
20         print(e)
21     finally:
22         cur.close()
23         conn.close()
24 @app.errorhandler(404)
25 def not_found(error=None):
26     message = {
27         'status': 404,
28         'message': 'Not Found: ' + request.url,
29     }
30     resp = jsonify(message)
31     resp.status_code = 404
32
33     return resp
34
35 if __name__ == "__main__":
36     app.run()
```

## Testeo

[Postman](#)



The screenshot shows the Postman API client interface. At the top, there's a tab labeled "GET Postman API" and a dropdown menu showing "No Environment". Below this, the URL bar contains "https://postman-echo.com/get?id=123&type=VIP" and a "Send" button. The interface is divided into several sections: "Params", "Auth", "Headers (8)", "Body", "Pre-req", "Tests", "Settings", and "Cookies". The "Query Params" section is expanded, showing a table with two rows of query parameters. The "Body" section is also expanded, showing the response in JSON format.

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>				
<input checked="" type="checkbox"/>				

```
1 {
2   "args": {
3     "id": "123",
4     "type": "VIP"
5   },
6   "headers": {
7     "x-forwarded-proto": "https",
8     "x-forwarded-port": "443",
```

# Sensores de dispositivo



Accelerometer



Gyroscope



Compass



GPS



Light sensor



Barometer

# SENSORES DEL DISPOSITIVO

Hay MUCHA información adicional que podemos obtener del dispositivo del cliente!  
Para completar las funcionalidades de nuestro servicio web...

...o para hacer el mal (fingerprinting)...

... por eso la mayoría requiere de permisos adicionales por parte del usuario.

- [Geolocation API](#)
- [Microphone](#)
- [Camera](#) (mediante [Media Capture and Streams API](#))
- [Vibration API](#)
- [Battery](#)
- [Sensor API](#)
  - [AbsoluteOrientationSensor](#) (Orientación respecto al norte magnético tierra)
  - [Accelerometer](#) (Aceleración lineal en 3 ejes)
  - [AmbientLightSensor](#) (Luz ambiental)
  - [GravitySensor](#) ()
  - [Gyroscope](#) (Aceleración angular en 3 ejes)
  - [LinearAccelerationSensor](#)
  - [Magnetometer](#)
  - [RelativeOrientationSensor](#) (Orientación respecto a otra orientación inicial)

⚠ La posibilidad de utilizar información de los sensores depende de:

- Que el dispositivo tenga dicho sensor.
- Que el navegador soporte dicho sensor.
- Que el navegador tenga habilitada la configuración de seguridad que permita utilizar dicho sensor.
- Que la persona usuaria haya dado permiso para utilizar dicho sensor.

# Geolocation

```
function success(pos) {  
    var crd = pos.coords;  
    console.log('Tu posición actual es:');  
    console.log(`Latitud : ${crd.latitude}`);  
    console.log(`Longitud: ${crd.longitude}`);  
}  
function error(err) {  
    console.warn(`ERROR(${err.code}): ${err.message}`);  
}  
navigator.geolocation.getCurrentPosition(success, error);
```

- Podemos “simular geolocalizaciones desde las dev tools.”
  - DevTools -> Ctrl+Shift+P -> Show Sensors -> Location

# Vibration API

```
function startPattern() {  
    // Values at even indices (0, 2, 4, ...) specify vibrations, while the odd  
    // indices specify pauses.  
    // Vibrate for 500ms 6 times, pausing for 250ms in between each one.  
    navigator.vibrate([500, 250, 500, 250, 500, 250, 500, 250, 500, 250, 500]);  
}
```

- Ejemplos: <https://googlechrome.github.io/samples/vibration/>

# Battery

```
navigator.getBattery().then(function(battery) {  
  console.log(`Nivel de Batería: ${battery.level * 100}%`);  
  battery.addEventListener('levelchange', function() {  
    console.log(`Nuevo Nivel de Batería: ${this.level * 100}%`);  
  });  
});
```

- Ejemplos: <https://googlechrome.github.io/samples/vibration/>

# Absolute/Relative Orientation Sensor

```
if ('AbsoluteOrientationSensor' in window) {  
  const sensor = new AbsoluteOrientationSensor({frequency: 60});  
  
  sensor.addEventListener('reading', () => {  
    console.log(`Orientación absoluta:  
      Cuaternión - [${sensor.quaternion}]`);  
  });  
  
  sensor.addEventListener('error', error => {  
    console.error('Error en el sensor de orientación absoluta:', error);  
  });  
  
  sensor.start();  
} else {  
  console.log('El sensor de orientación absoluta no está disponible.');
```

```
if ('RelativeOrientationSensor' in window) {  
  const sensor = new RelativeOrientationSensor({frequency: 60});  
  
  sensor.addEventListener('reading', () => {  
    console.log(`Orientación relativa:  
      Cuaternión - [${sensor.quaternion}]`);  
  });  
  
  sensor.addEventListener('error', error => {  
    console.error('Error en el sensor de orientación relativa:', error);  
  });  
  
  sensor.start();  
} else {  
  console.log('El sensor de orientación relativa no está disponible.');
```

- Ejemplo de Absolute Orientation Sensor en 3D:  
<https://intel.github.io/generic-sensor-demos/orientation-phone/>
- Ejemplo de lecturas en crudo de distintos sensores:  
<https://intel.github.io/generic-sensor-demos/sensor-info/build/bundled/>