Clase 16

Almacenes de información

IIC2513 - Tecnologías y Aplicaciones Web

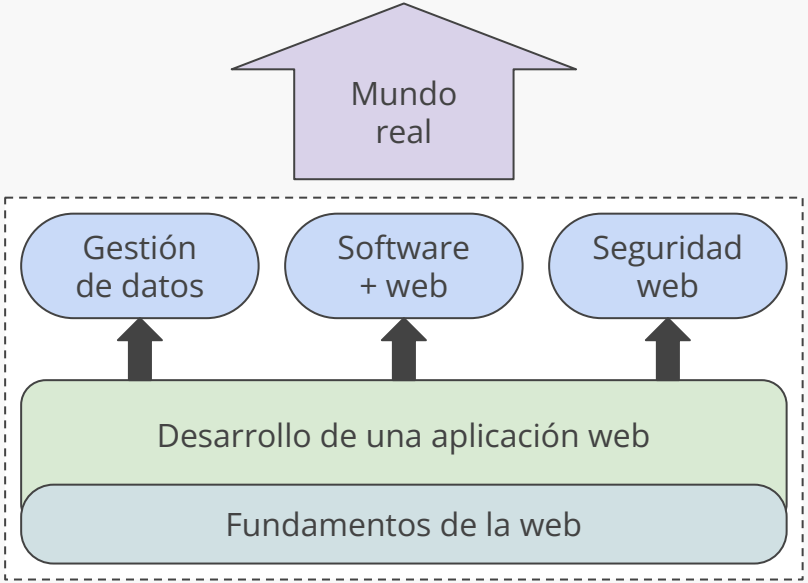
Antonio Ossa Guerra
aaossa@ing.puc.cl

1. Vamos en la Entrega 1
2. Sala de ayuda E1: prox. viernes

Anuncios

Contenidos del curso

(1) Fundamentos de la web
(2) Desarrollo de una app web
(3) Gestión de datos
(4) Software + web
(5) Seguridad web
(6) El mundo real

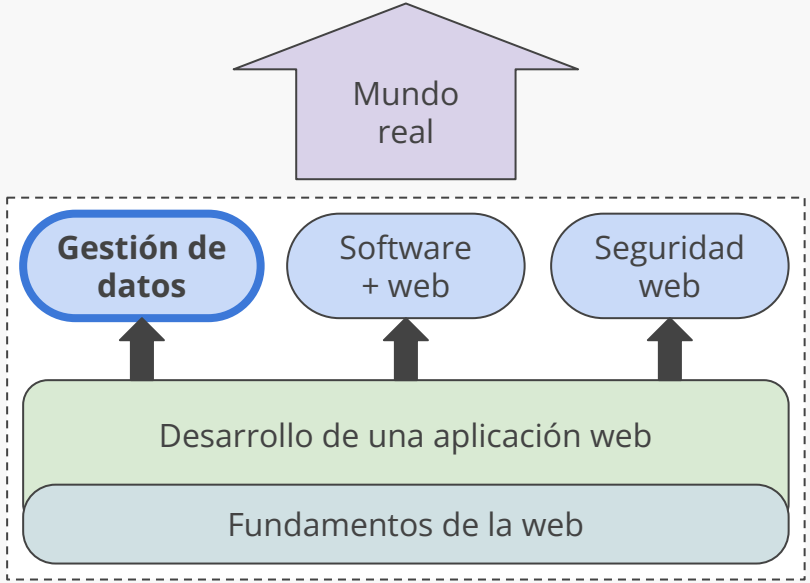


Contenidos del curso

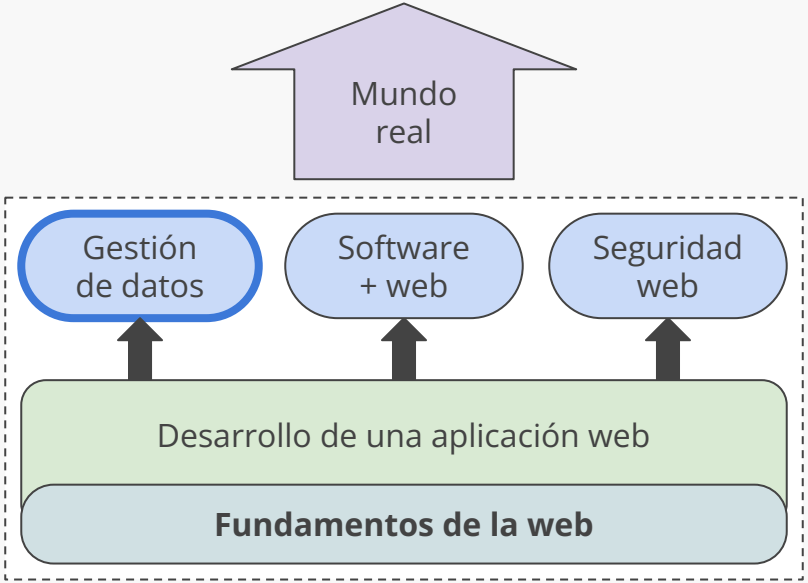
Gestión de datos

Veremos cómo diseñar y manejar ***cookies***, ***sesiones*** y **otros datos** en el navegador...

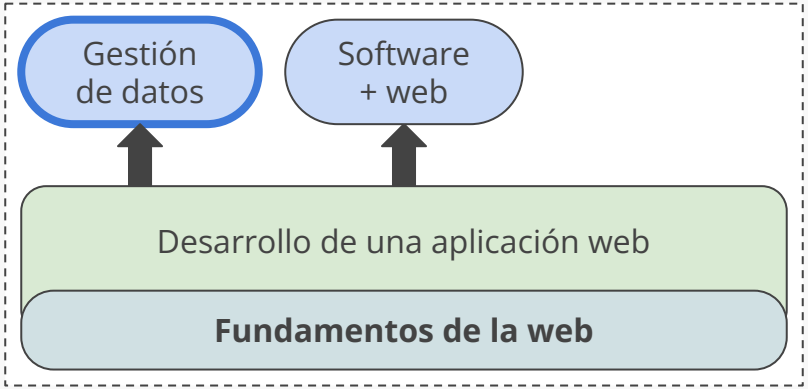
... para habilitar **funcionalidades avanzadas** de manera segura y con técnicas modernas



Contenidos del curso



Contenidos del curso



Menti*#Una pequeña encuesta*

Agenda

✨ **Almacenamiento *client-side***

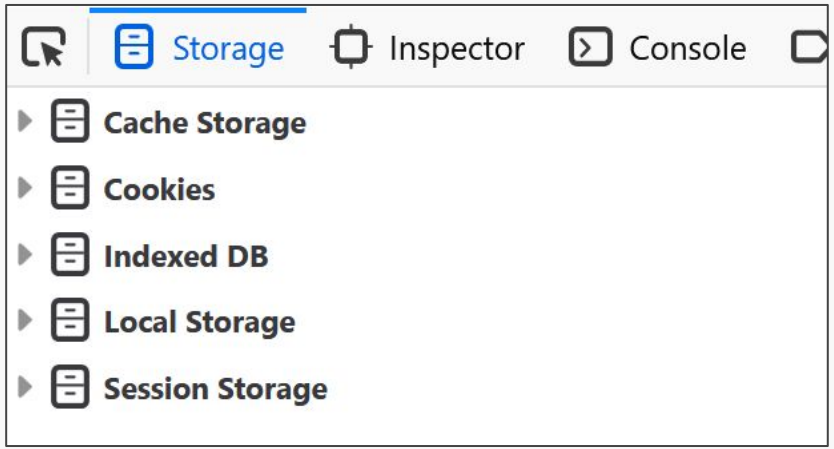
Almacenamiento *server-side*

Almacenamiento *client-side*

Cada vez existe más control sobre lo que podemos almacenar del lado del cliente, en almacenamientos de distinta duración y de distinto nivel de complejidad

Elementos que deben conocer:

- Cookies
- Web Storage API
- IndexedDB API
- Cache API



Fuente: Client-side storage - Learn web development | MDN

Tipos de almacenamiento



Figura: What is Web Storage - Types, Tips & Use Cases | Ramotion Agency

¿Por qué es almacenar del lado del cliente?

- Permite personalizar preferencias en un sitio (por ejemplo, orden de algunos elementos, colores de preferencia, tamaño de fuente preferido, etc.)
- La actividad en páginas anteriores se hace persistente (por ejemplo, contenidos de un carro de compras, recordar si ya inició sesión, etc.)
- Almacenar datos y *assets* localmente permiten que un sitio cargue más rápido (y consuma menos recursos) e incluso funcionar sin conexión a Internet
- Almacenar elementos generados por una aplicación web para ser usados *offline*

Almacenamiento *client-side*

- Cookies es la manera más tradicional y conocida de almacenar pequeñas piezas de información del lado del cliente (aunque hay novedades como Cookie Store API)
- La Web Storage API nos permite almacenar datos más pequeños (solo *strings*), con distintos niveles de persistencia, con absoluto control desde JavaScript
- La IndexedDB API permite que el navegador utilice un sistema de base de datos completo para almacenar datos más complejos (objetos de JS)
- La Cache API nos permite almacenar y recuperar solicitudes y respuestas con el fin de entregar respuestas rápidas a solicitudes, pero también puede usarse de almacén

Fuente: Client-side storage - Learn web development | MDN

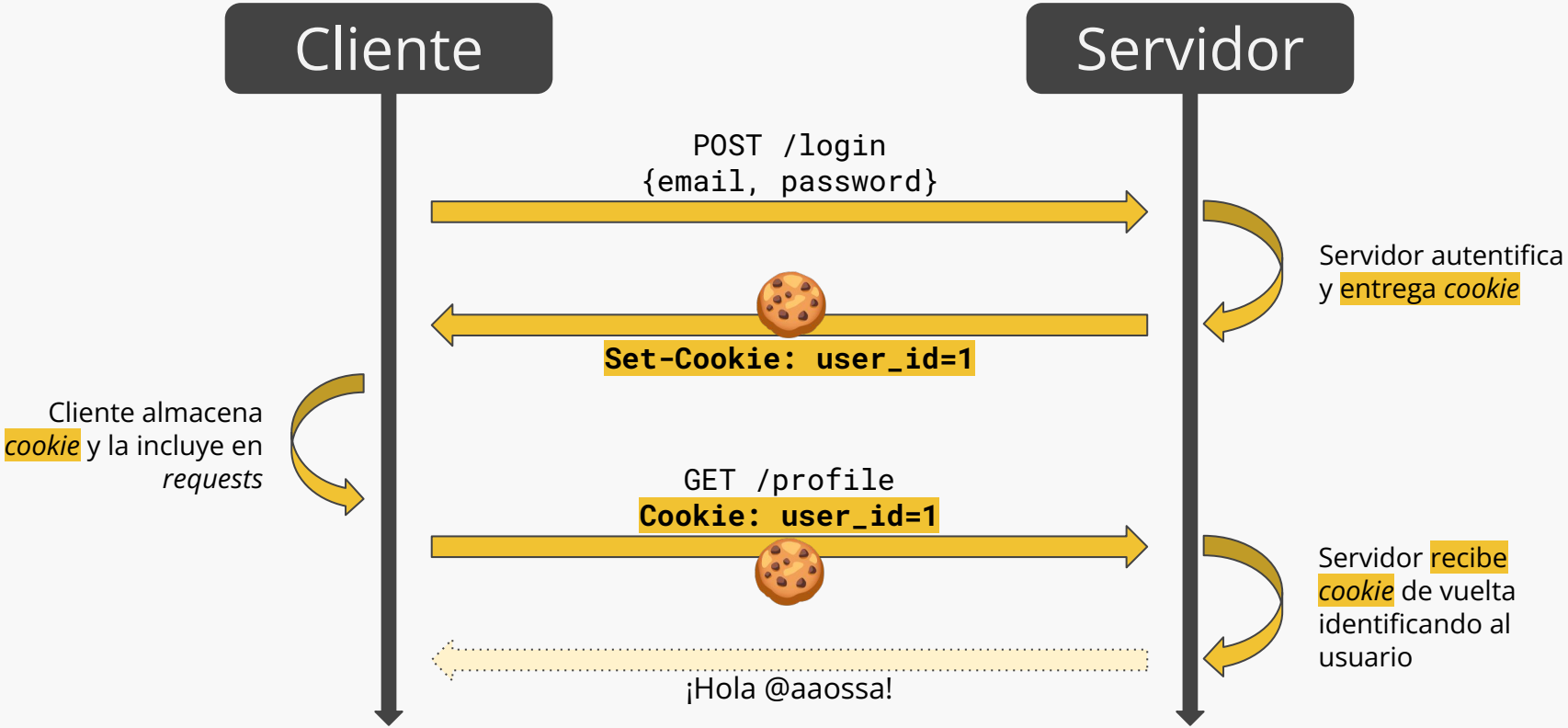
Agenda

Almacenamiento *client-side*

- ✨ ***Cookies***
- *Web Storage*
- *IndexedDB*
- *Cache API*

Almacenamiento server-side

Funcionamiento *headers* para *cookies*



Cookie: manejadas por HTTP *headers*

Response cookie puede incluir una serie de directivas adicionales, tales como:

1. `Expires`
   tiempo de vida máximo útil
2. `Domain`
   hosts donde se puede enviar la *cookie*
3. `Secure`
   solo enviar si se usa HTTPS
4. `HttpOnly`
   evita que se pueda acceder desde JS
5. `Path`
   solo incluye *cookie* si estamos en *path*

```
Set-Cookie: cookie=valor
Set-Cookie: cookie=valor; Expires=<date>
Set-Cookie: cookie=valor; Max-Age=<age>
Set-Cookie: cookie=valor; Domain=<domain>
Set-Cookie: cookie=valor; Path=<path>
Set-Cookie: cookie=valor; Secure
Set-Cookie: cookie=valor; HttpOnly
Set-Cookie: cookie=valor; SameSite=Strict

// Es posible usar múltiples directivas
Set-Cookie: cookie=valor;
Domain=<domain-value>; Secure; HttpOnly

// Ejemplo de cookie real
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct
2021 07:28:00 GMT; Secure; HttpOnly
```

Cookie: manejadas por HTTP *headers*

Las cookies también se pueden manejar desde JavaScript, por lo que se pueden usar para almacenar, recuperar y gestionar datos pequeños en el cliente web

Algunos usos comunes son:

- Recordar sesiones de usuario
- Guardar preferencias de usuario
- Realizar seguimiento a la actividad

```
// Establecer una cookie
document.cookie = 'theme=dark; path=/; max-age=' + 24 *
60 * 60; // Expira en 24 horas

// Obtener una cookie específica
const themeCookie = cookies['theme'];
if (themeCookie) {
  console.log('Valor de "theme":', themeCookie);
} else {
  console.log('¡La cookie "theme" no existe!');
}

// Eliminar una cookie
document.cookie = 'theme=; path=/; max-age=0';
// Configurar max-age a 0 elimina la cookie
console.log('Cookie "theme" eliminada')
```

Cookie Store API: manejo moderno de *cookies*

`CookieStore` y `CookieStoreManager*` son APIs diseñadas para gestionar las cookies de manera eficiente y flexible en una aplicación web, permitiendo leer, escribir y observar cambios en las cookies

Algunas aplicaciones:

- Gestión estructurada de cookies
- Sincronización en tiempo real

```
// Establecer una cookie
cookieStore.set('theme', 'dark', {
  path: '/',
  expires: Date.now() + 24 * 60 * 60 * 1000,
}); // Expira en 24 horas

// Obtener una cookie específica
cookieStore.get('theme').then(cookie => {
  if (cookie) {
    console.log('Valor de "theme":', cookie.value);
    // Muestra el valor de la cookie
  } else {
    console.log('¡La cookie "theme" no existe!');
  }
});

// Eliminar una cookie
cookieStore.delete('theme').then(() => {
  console.log('Cookie "theme" eliminada');
});
```

Fuente: [CookieStore - Web APIs | MDN]

Agenda

Almacenamiento *client-side*

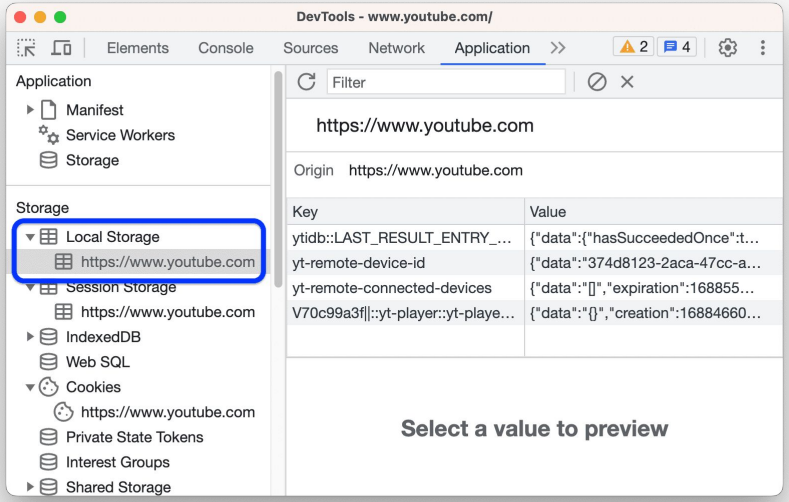
- *Cookies*
- ✨ ***Web Storage***
- *IndexedDB*
- *Cache API*

Almacenamiento server-side

localStorage: parte de la Web Storage API

`localStorage` permite compartir valores entre todas las pestañas y ventanas del mismo origen

Los datos no expiran: se conservan a pesar de reinicios del navegador y del sistema operativo



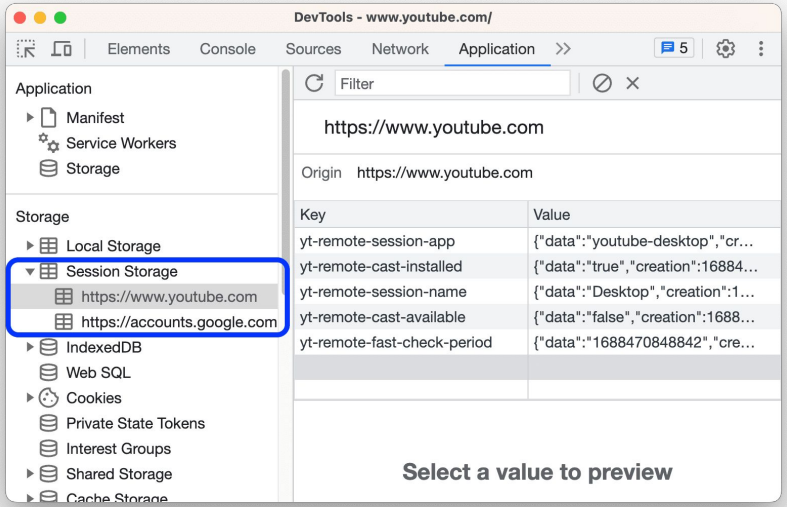

```
1 // Guardamos el valor 1 en la clave 'test'
2 localStorage.setItem('test', 1);
3
4 // después de cerrar el navegador o abrir otra pestaña o ventana
5 alert( localStorage.getItem('test') ); // 1
```

Fuente: LocalStorage, sessionStorage | Javascript.info Figura: View and edit local storage | Chrome DevTools

sessionStorage: parte de la Web Storage API

`sessionStorage` se utiliza menos que `localStorage` pues es más limitado:

- sus valores solo existen en la pestaña actual del navegador (salvo iframes)
- Los datos sobreviven a recargar la página pero no a abrir/cerrar la pestaña




```
1 // Guardamos el valor 1 en la clave 'test'
2 sessionStorage.setItem('test', 1);
3
4 // después de recargar la página
5 alert( sessionStorage.getItem('test') ); // 1
6
7 // después de cerrar el navegador o abrir otra pestaña o ventana
8 alert( sessionStorage.getItem('test') ); // undefined
```

Fuente: LocalStorage, sessionStorage | Javascript.info Figura: View and edit session storage | Chrome DevTools

Resumen de *Web Storage*

Tanto para `localStorage` como `sessionStorage`, el almacenamiento funciona en base a guardar pares de clave/valor en el navegador

- Tanto la clave como el valor deben ser *strings*
- El límite es de más de 5 MB (dependiendo del navegador)
- Los datos almacenados no expiran
- Los datos almacenados están vinculados al origen (dominio+puerto+protocolo)

IMPORTANTE
En clases dije que solo la clave debía ser *string*, lo que confirmé es incorrecto 😢

La gran diferencia entre ambos es que pueden sobrevivir apenas a una recarga de página (en el caso de `sessionStorage`) o a un reinicio completo del navegador (`localStorage`)

Fuente: LocalStorage, sessionStorage | JavaScript.info

¿Cuándo usar cada una?

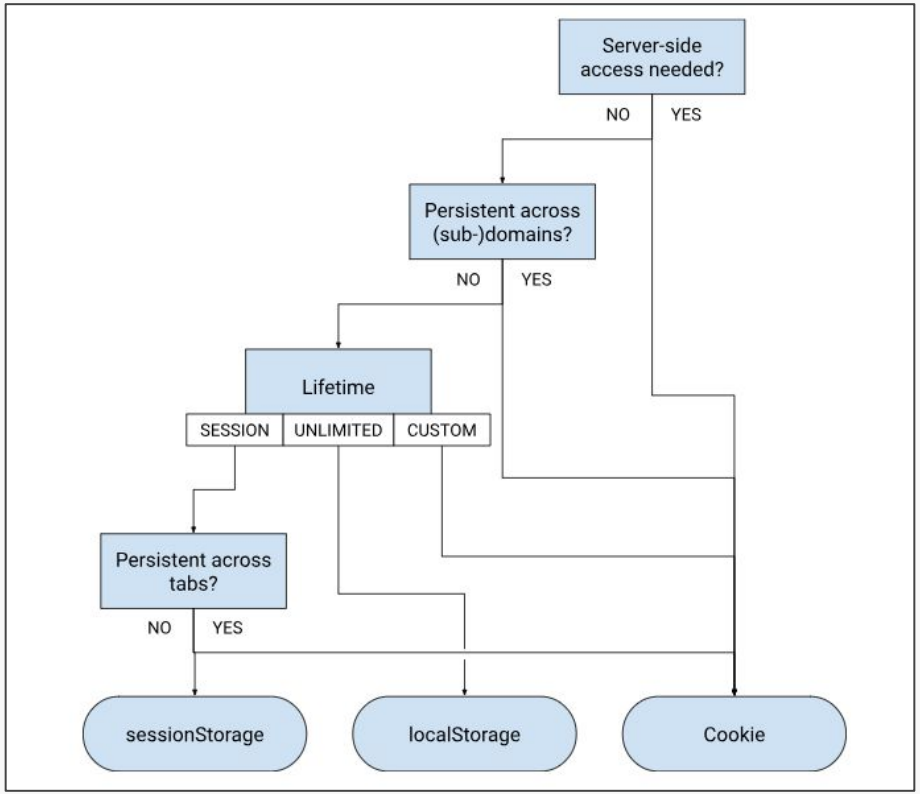


Figura: [Ultimate Guide to Data Storage in Google Tag Manager | TRKKN]

Agenda

- Almacenamiento *client-side*
  - *Cookies*
  - *Web Storage*
  - ✨ ***IndexedDB***
  - *Cache API*
- Almacenamiento server-side

IndexedDB: una API transaccional moderna

IndexedDB es un sistema de bases de datos transaccional, pero orientado a objetos de JavaScript

IndexedDB permite almacenar y recuperar objetos que están indexados con una clave o llave

Es necesario especificar el esquema de la base de datos (crear un `objectStore`), abrir una conexión con la base de datos, y luego recuperar y actualizar datos dentro de una serie de transacciones

```
let transaction = db.transaction("books", "readwrite"); // (1)

// obtiene un almacén de objetos para operar con él
let books = transaction.objectStore("books"); // (2)

let book = {
  id: 'js',
  price: 10,
  created: new Date()
};

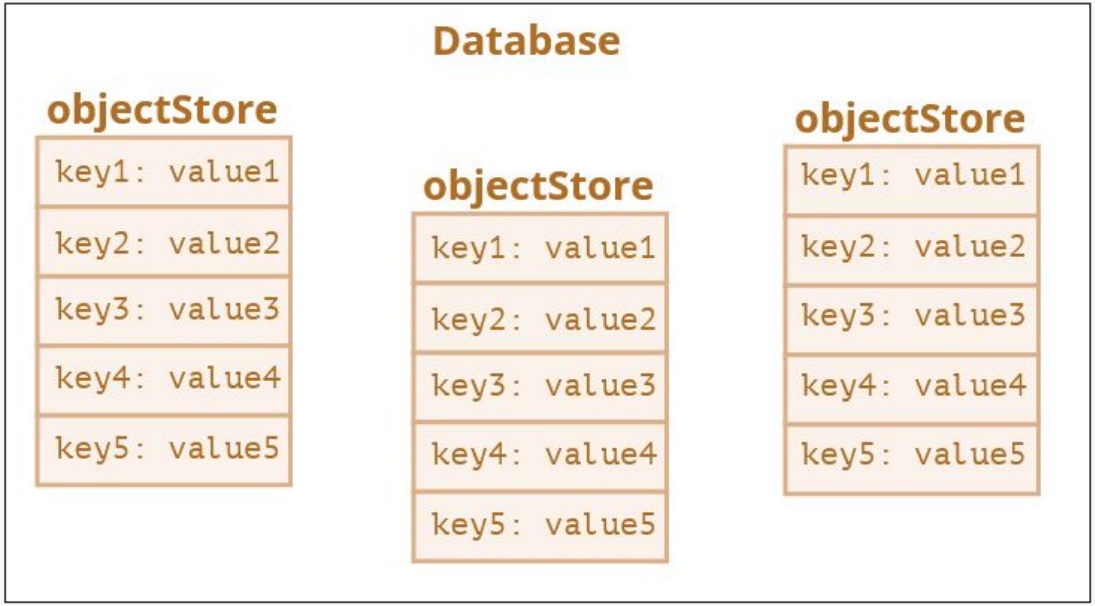
let request = books.add(book); // (3)

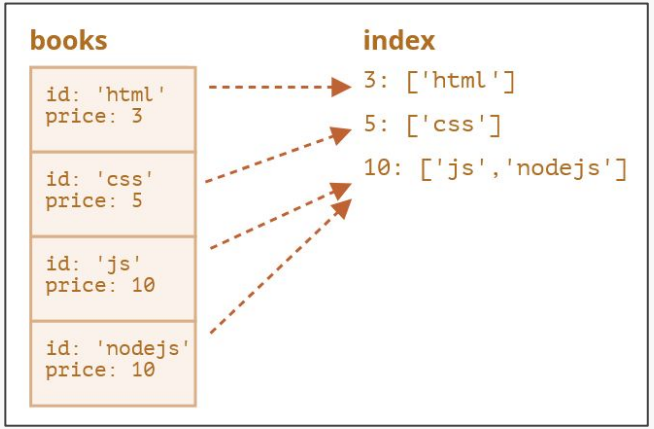
request.onsuccess = function() { // (4)
  console.log("Libro agregado al almacén", request.result);
};

request.onerror = function() {
  console.log("Error", request.error);
};
```

Fuente: [IndexedDB API | MDN]

IndexedDB: una API transaccional moderna

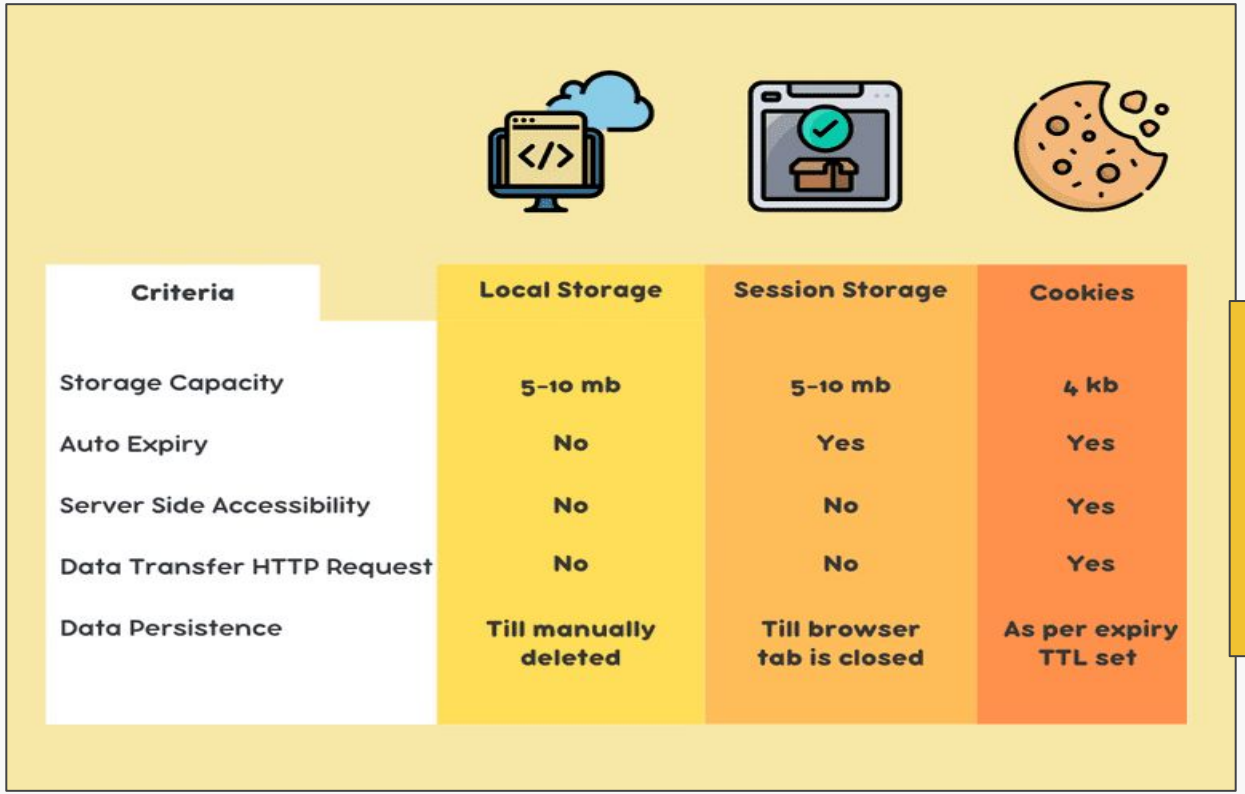





```
openRequest.onupgradeneeded = function() {
  // debemos crear el índice aquí, en la transacción versionchange
  let books = db.createObjectStore('books', {keyPath: 'id'});
  let index = books.createIndex('price_idx', 'price');
};
```

Figura: IndexedDB | JavaScript.info

¿Cuándo usar cada una?



Criteria	Local Storage	Session Storage	Cookies
Storage Capacity	5-10 mb	5-10 mb	4 kb
Auto Expiry	No	Yes	Yes
Server Side Accessibility	No	No	Yes
Data Transfer HTTP Request	No	No	Yes
Data Persistence	Till manually deleted	Till browser tab is closed	As per expiry TTL set

IndexedDB

Límite de capacidad depende del navegador y espacio en disco disponible (pueden ser GBs)

Figura: Local Storage vs. Session Storage vs. Cookies

Agenda

- Almacenamiento *client-side*
  - *Cookies*
  - *Web Storage*
  - *IndexedDB*
  - ✨ ***Cache API***
- Almacenamiento server-side

Tipos de caché

- **Caché privado**: caché vinculada a un cliente específico (en general, navegador)
- **Caché compartido**: respuestas compartidas entre varios usuarios y el servidor
  - Caché proxy (*proxy cache*): un servidor actúa como intermediario de contenido general para múltiples usuarios y aplicaciones
  - Caché administrada (*managed cache*): implementadas para evitar la sobrecarga en el servidor de origen (por ejemplo, proxies reversos, CDNs y *service workers*), específica y optimizada para aplicaciones y necesidades específicas

Figura: HTTP caching | MDN

Tipos de caché

- **Caché privado**: caché vinculada a un cliente específico (en general, navegador)
- **Caché compartido**: respuestas compartidas entre varios usuarios y el servidor
  - Caché proxy (*proxy cache*): un servidor actúa como intermediario de contenido general para múltiples usuarios y aplicaciones
  - Caché administrada (*managed cache*): implementadas para evitar la sobrecarga en el servidor de origen (por ejemplo, proxies reversos, CDNs y *service workers*), específica y optimizada para aplicaciones y necesidades específicas

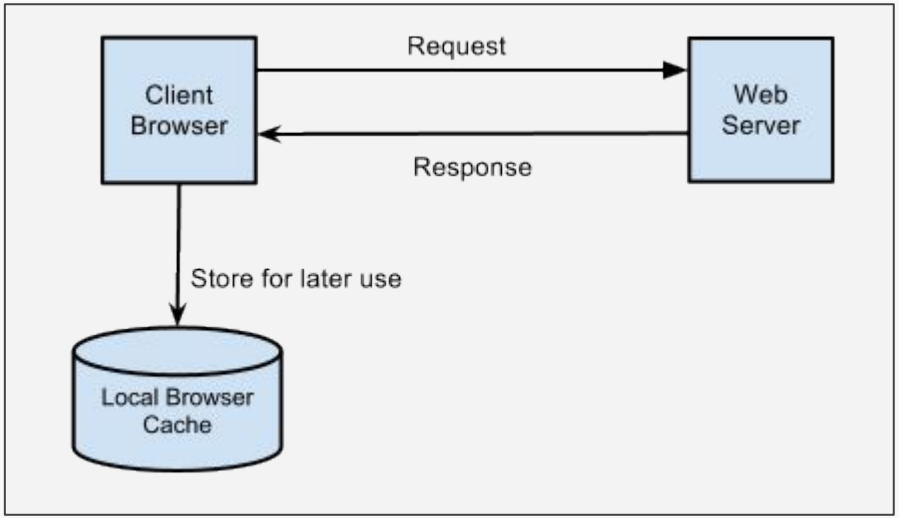
Figura: HTTP caching | MDN

Cache API: almacenando *requests*

Cache API permite almacenar y gestionar respuestas de manera eficiente, gestionando un almacenamiento en caché. Esto hace posible funcionar sin conexión, mejorar tiempos de carga y reducir el consumo de datos reutilizando respuestas

Algunas aplicaciones:

- Almacenar recursos para uso offline
- Optimizar tiempos de carga
- Reducir solicitudes redundantes



Fuente: Cache - Web APIs | MDN Figura: Increasing Application Performance with HTTP Cache Headers | Heroku Dev Center

Cache API: almacenando *requests*

Cache API permite almacenar y gestionar respuestas de manera eficiente, gestionando un almacenamiento en caché. Esto hace posible funcionar sin conexión, mejorar tiempos de carga y reducir el consumo de datos reutilizando respuestas

Algunas aplicaciones:

- Almacenar recursos para uso offline
- Optimizar tiempos de carga
- Reducir solicitudes redundantes

```
// Open a cache and store a resource
caches.open('my-cache').then(cache => {
  cache.add('/example-resource'); // Add a resource to
the cache
});

// Retrieve the cached resource
caches.match('/example-resource').then(response => {
  if (response) {
    response.text().then(content => {
      console.log('Cached content:', content);
      // Log the cached content
    });
  } else {
    console.log('Resource not found in cache!');
  }
});
```

Fuente: Cache - Web APIs | MDN

Cache API: almacenando *requests*

Métodos :

- `add(request), addAll(request)`
- `put(request, response)`
- `delete(request, options)`
- `keys(request?, options?)`
- `match(request, options?)`
- `matchAll(request, options?)`

Puntos de entrada:

- `self.caches` (service worker)
- `window.caches` (la página)

```
// Open a cache and store a resource
caches.open('my-cache').then(cache => {
  cache.add('/example-resource'); // Add a resource to
the cache
});

// Retrieve the cached resource
caches.match('/example-resource').then(response => {
  if (response) {
    response.text().then(content => {
      console.log('Cached content:', content);
      // Log the cached content
    });
  } else {
    console.log('Resource not found in cache!');
  }
});
```

Fuente: Cache - Web APIs | MDN

[The Cache API: A quick guide | Articles | web.dev](): Guía rápida en web.dev

The Cache API: A quick guide

Learn how to use the Cache API to make your application data available offline.



Pete LePage

The Cache API is a system for storing and retrieving network requests and their corresponding responses. These might be regular requests and responses created in the course of running your application, or they could be created solely for the purpose of storing data for later use.

The Cache API was created to enable service workers to cache network requests so that they can provide fast responses, regardless of network speed or availability. However, the API can also be used as a general storage mechanism.

Where is it available?

The Cache API is available in all modern browsers. It is exposed via the global `caches` property, so you can test for the presence of the API with a simple feature detection:

```
const cacheAvailable = 'caches' in self;
```

On this page

Where is it available?

What can be stored

How much can be stored?

Creating and opening a cache

Adding to a cache

cache.add

cache.addAll

cache.put

Retrieving from a cache

Searching

Filtering

Creating an index

Deleting an item

Deleting a cache

Cache - Web APIs | MDN: Documentación en MDN sobre la Cache API

Filter

Service Worker API

Cache

▾ Instance methods

add()

addAll()

delete()

keys()

match()

matchAll()

put()

▾ Related pages for Service Workers API

CacheStorage

Client

Cache

Baseline Widely available

Secure context: This feature is available only in secure contexts (HTTPS), in some or all supporting browsers.

Note: This feature is available in Web Workers.

The `Cache` interface provides a persistent storage mechanism for `Request` / `Response` object pairs that are cached in long lived memory. How long a `Cache` object lives is browser dependent, but a single origin's scripts can typically rely on the presence of a previously populated `Cache` object. Note that the `Cache` interface is exposed to windowed scopes as well as workers. You don't have to use it in conjunction with service workers, even though it is defined in the service worker spec.

An origin can have multiple, named `Cache` objects. You are responsible for implementing how your

In this article

Instance methods

Examples

Specifications

Browser compatibility

See also

Almacenamiento *client-side*: Recordatorio

- Cookies es la manera más tradicional y conocida de almacenar pequeñas piezas de información del lado del cliente (aunque con alternativas nuevas como `CookieStore`)
- La Web Storage API nos permite almacenar datos más pequeños (solo *strings*), con distintos niveles de persistencia, con absoluto control desde JavaScript
- La IndexedDB API permite que el navegador utilice un sistema de base de datos completo para almacenar datos más complejos (objetos de JS)
- La Cache API nos permite almacenar y recuperar solicitudes y respuestas con el fin de entregar respuestas rápidas a solicitudes, pero también puede usarse de almacén

Fuente: Client-side storage - Learn web development | MDN

Almacenamiento *client-side*: Recordatorio

- Cookies es la manera más tradicional y conocida de almacenar pequeñas piezas de información del lado del cliente (aunque con alternativas nuevas como `CookieStore`)
- La Web Storage API nos permite almacenar datos más pequeños (solo *strings*), con distintos niveles de persistencia, con absoluto control desde JavaScript
- La IndexedDB API permite usar bases de datos tipo NoSQL
- La Cache API está diseñada para almacenar recursos de red, y es útil para escenarios como entregar respuestas rápidas a solicitudes, pero también puede usarse de almacén

Desafío práctico

Fuente: Client-side storage - Learn web development | MDN

Desafío práctico: IndexedDB

1. Creen un repositorio especial para este desafío. Lo podrán crear con la siguiente invitación de GitHub Classroom:

   https://classroom.github.com/a/Bq5Fu5QB

   (Las instrucciones detalladas se encuentran en el README del repositorio creado)

2. En esta actividad, se deberán implementar funciones para **guardar** y **cargar** datos del lado del cliente, utilizando **IndexedDB**, y también almacenar una imagen utilizando la **Cache API**. Además, deberán responder las preguntas del README

Agenda

Almacenamiento *client-side*

✨ **Almacenamiento *server-side***

Agenda

Almacenamiento *client-side*

Almacenamiento *server-side*

✨ ***Cache***

Bases de datos SQL

Bases de datos NoSQL

Tipos de caché

- **Caché privado**: caché vinculada a un cliente específico (en general, navegador)
- **Caché compartido**: respuestas compartidas entre varios usuarios y el servidor
  - Caché proxy (*proxy cache*): un servidor actúa como intermediario de contenido general para múltiples usuarios y aplicaciones
  - Caché administrada (*managed cache*): implementadas para evitar la sobrecarga en el servidor de origen (por ejemplo, proxies reversos, CDNs y *service workers*), específica y optimizada para aplicaciones y necesidades específicas

Figura: HTTP caching | MDN

CDN (caché administrada)

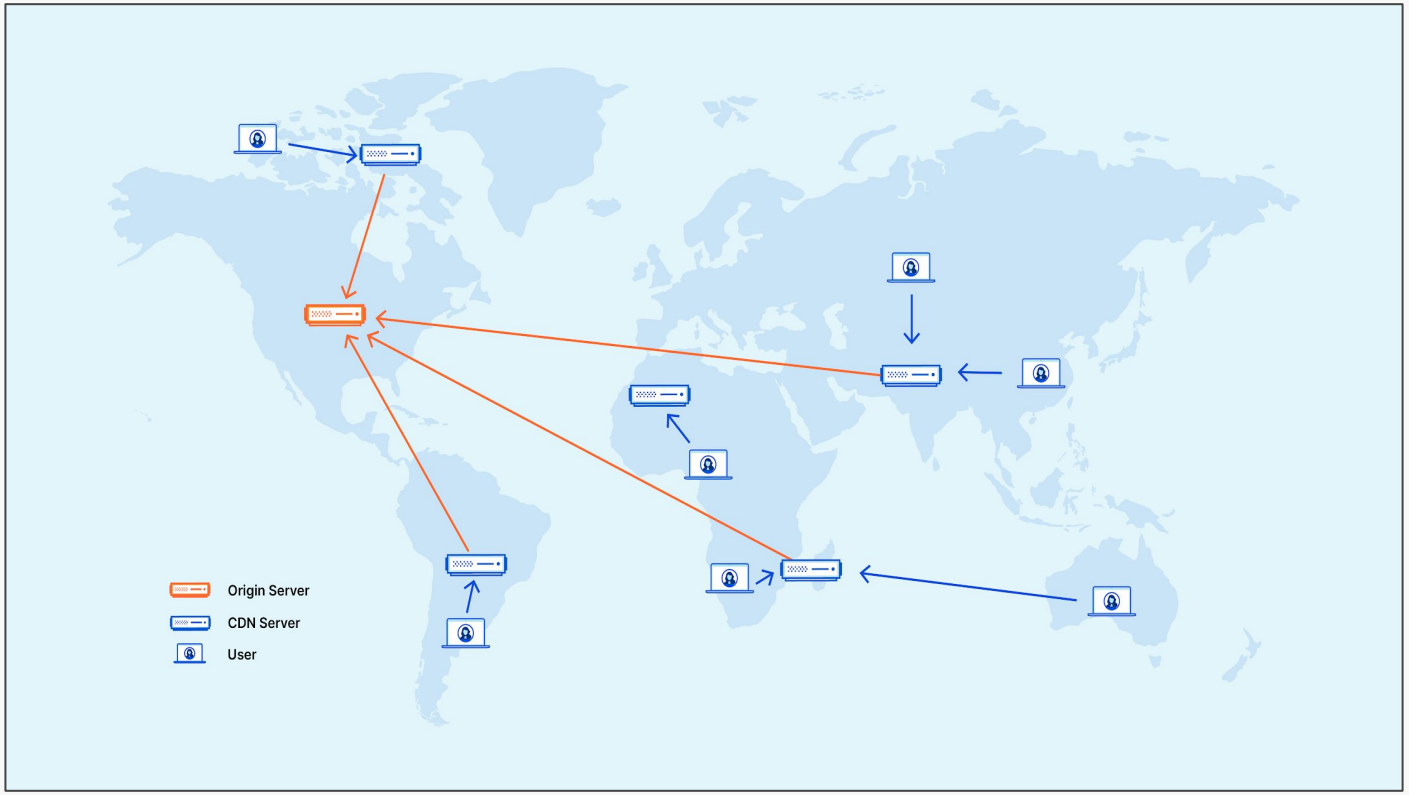


Figura: What is a content delivery network (CDN)? | How do CDNs work? | Cloudflare

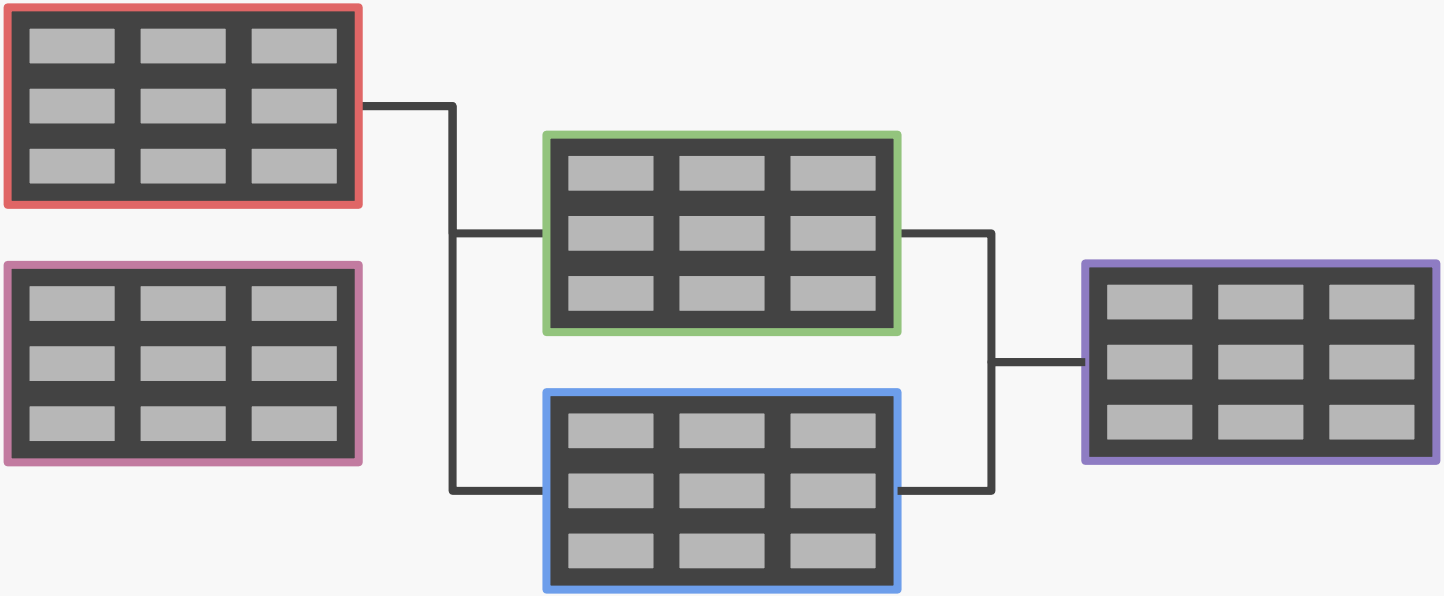
Agenda

Almacenamiento *client-side*

Almacenamiento *server-side*

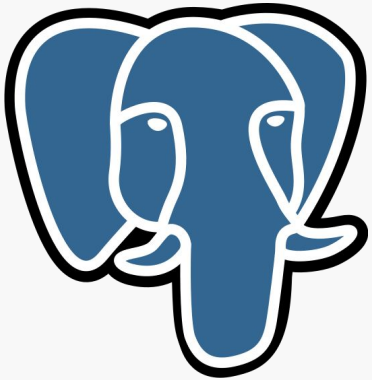
- *Cache*
- ✨ **Bases de datos SQL**
- Bases de datos NoSQL

Bases de datos



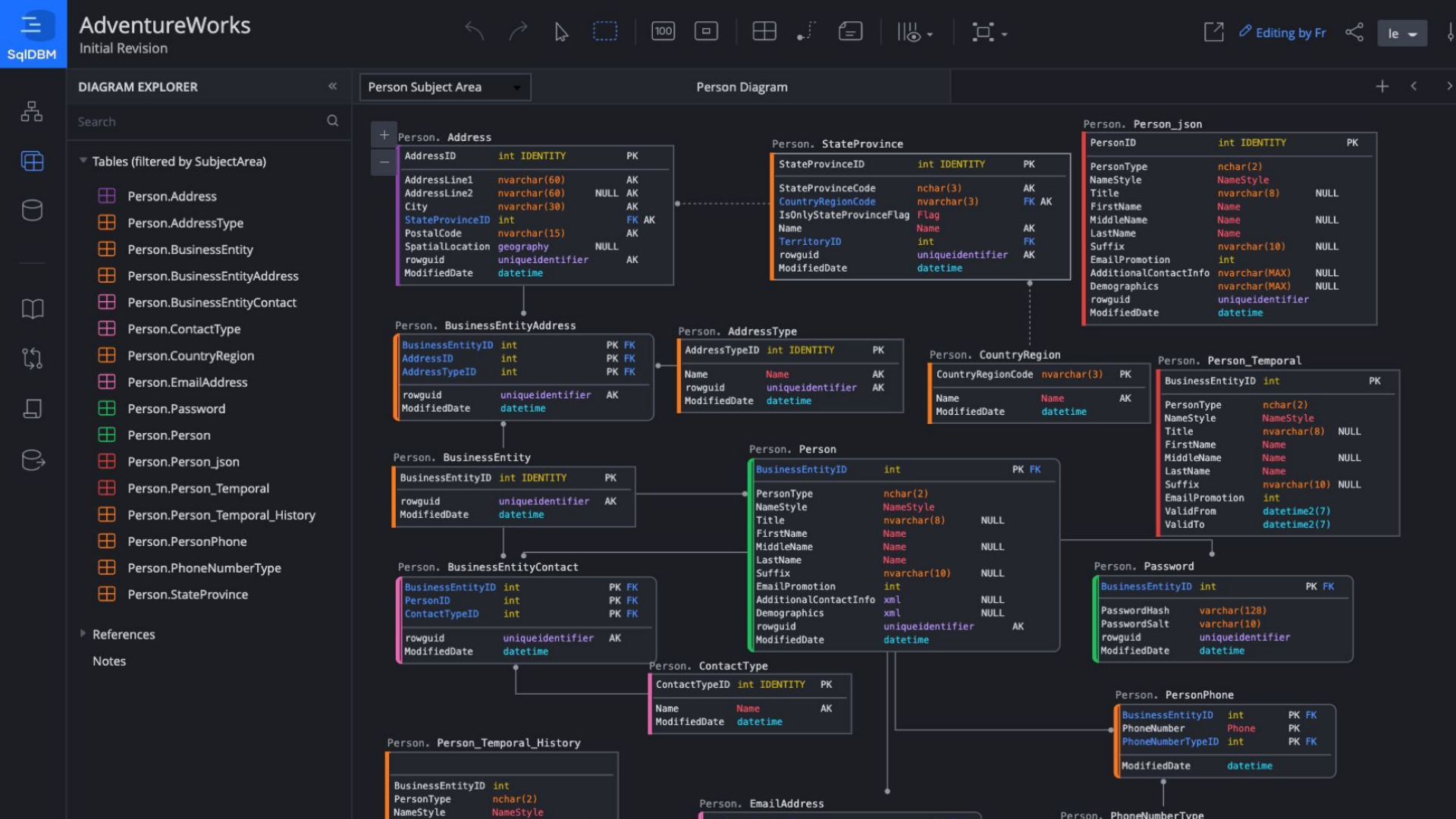
Bases de datos

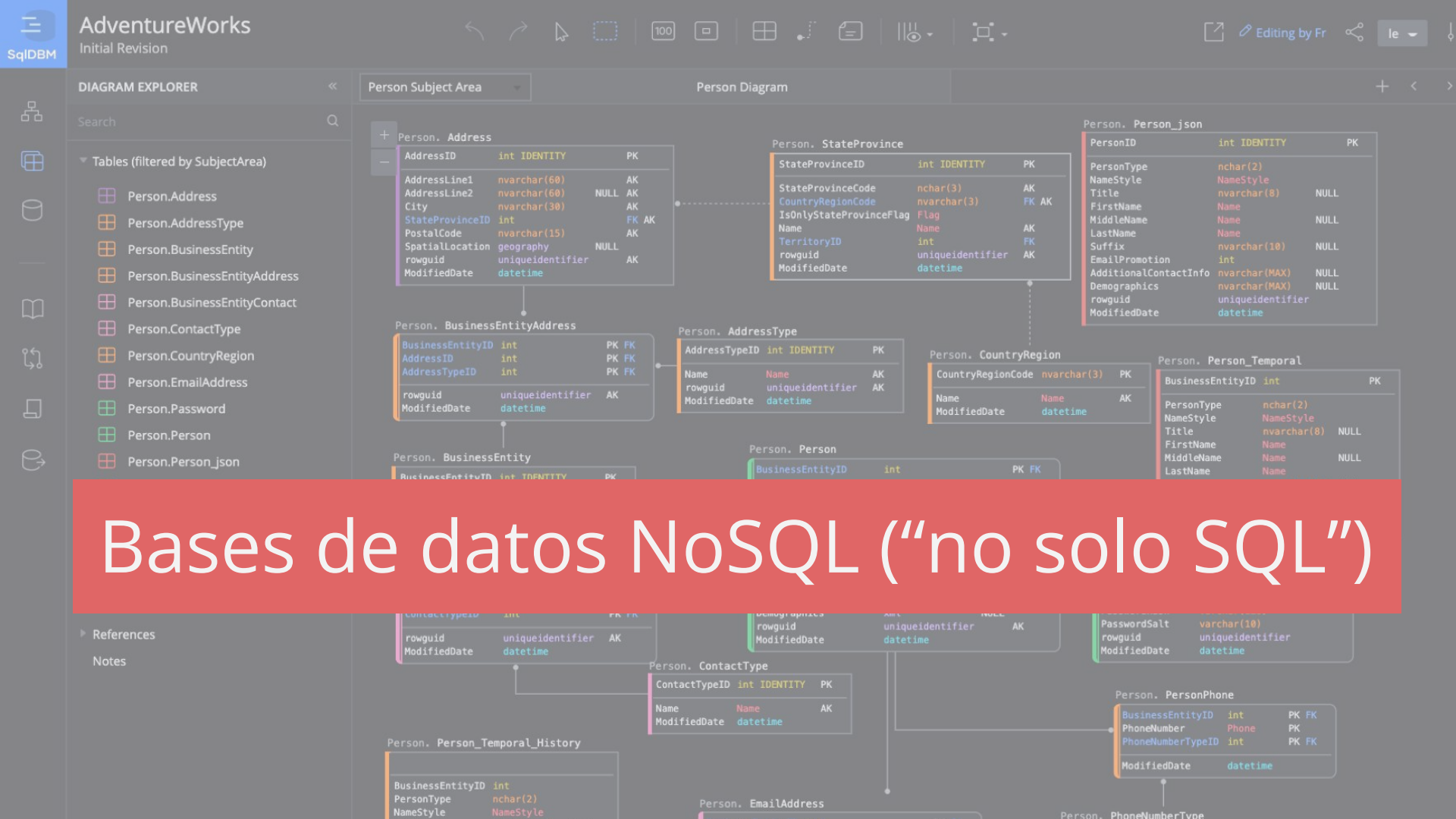









SqlDBM
AdventureWorks
Initial Revision
Editing by Fr
DIAGRAM EXPLORER
Search
Tables (filtered by SubjectArea)
Person.Address
Person.AddressType
Person.BusinessEntity
Person.BusinessEntityAddress
Person.BusinessEntityContact
Person.ContactType
Person.CountryRegion
Person.EmailAddress
Person.Password
Person.Person
Person.Person_json
Person.Person_Temporal
Person.Person_Temporal_History
Person.PersonPhone
Person.PhoneNumberType
Person.StateProvince
References
Notes
Person Subject Area
Person Diagram
Person. Address
AddressID int IDENTITY PK
AddressLine1 nvarchar(60) AK
AddressLine2 nvarchar(60) NULL AK
City nvarchar(30) AK
StateProvinceID int FK AK
PostalCode nvarchar(15) AK
SpatialLocation geography NULL
rowguid uniqueidentifier AK
ModifiedDate datetime
Person. StateProvince
StateProvinceID int IDENTITY PK
StateProvinceCode nchar(3) AK
CountryRegionCode nvarchar(3) FK AK
IsOnlyStateProvinceFlag Flag
Name Name AK
TerritoryID int FK
rowguid uniqueidentifier AK
ModifiedDate datetime
Person. Person_json
PersonID int IDENTITY PK
PersonType nchar(2)
NameStyle NameStyle
Title nvarchar(8) NULL
FirstName Name
MiddleName Name NULL
LastName Name
Suffix nvarchar(10) NULL
EmailPromotion int
AdditionalContactInfo nvarchar(MAX) NULL
Demographics nvarchar(MAX) NULL
rowguid uniqueidentifier
ModifiedDate datetime
Person. BusinessEntityAddress
BusinessEntityID int PK FK
AddressID int PK FK
AddressTypeID int PK FK
rowguid uniqueidentifier AK
ModifiedDate datetime
Person. AddressType
AddressTypeID int IDENTITY PK
Name Name AK
rowguid uniqueidentifier AK
ModifiedDate datetime
Person. CountryRegion
CountryRegionCode nvarchar(3) PK
Name Name AK
ModifiedDate datetime
Person. Person_Temporal
BusinessEntityID int PK
PersonType nchar(2)
NameStyle NameStyle
Title nvarchar(8) NULL
FirstName Name
MiddleName Name NULL
LastName Name
Suffix nvarchar(10) NULL
EmailPromotion int
ValidFrom datetime2(7)
ValidTo datetime2(7)
Person. BusinessEntity
BusinessEntityID int IDENTITY PK
rowguid uniqueidentifier AK
ModifiedDate datetime
Person. Person
BusinessEntityID int PK FK
PersonType nchar(2)
NameStyle NameStyle
Title nvarchar(8) NULL
FirstName Name
MiddleName Name NULL
LastName Name
Suffix nvarchar(10) NULL
EmailPromotion int
AdditionalContactInfo xml NULL
Demographics xml NULL
rowguid uniqueidentifier AK
ModifiedDate datetime
Person. Password
BusinessEntityID int PK FK
PasswordHash varchar(128)
PasswordSalt varchar(10)
rowguid uniqueidentifier
ModifiedDate datetime
Person. BusinessEntityContact
BusinessEntityID int PK FK
PersonID int PK FK
ContactTypeID int PK FK
rowguid uniqueidentifier AK
ModifiedDate datetime
Person. ContactType
ContactTypeID int IDENTITY PK
Name Name AK
ModifiedDate datetime
Person. PersonPhone
BusinessEntityID int PK FK
PhoneNumber Phone PK
PhoneNumberTypeID int PK FK
ModifiedDate datetime
Person. Person_Temporal_History
BusinessEntityID int
PersonType nchar(2)
NameStyle NameStyle
Person. EmailAddress
Person. PhoneNumberType



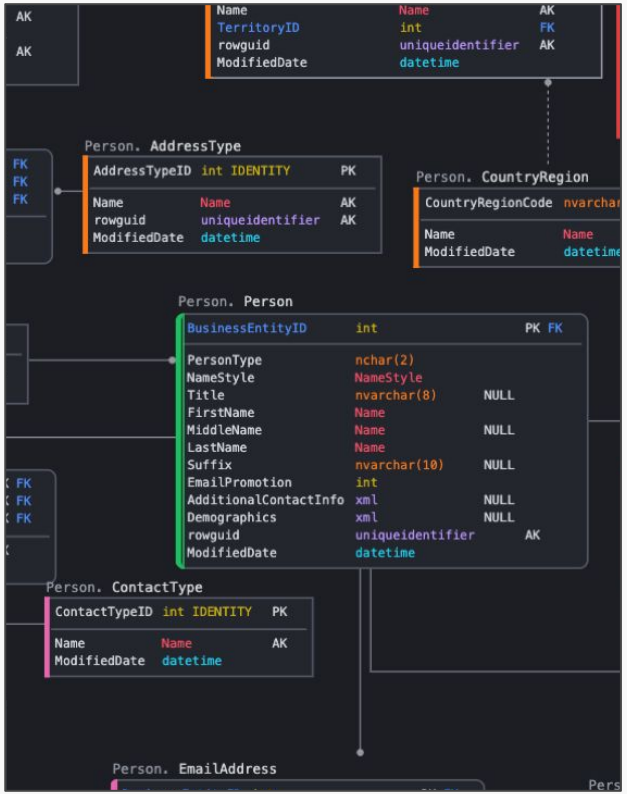
Bases de datos NoSQL (“no solo SQL”)

Agenda

- Almacenamiento *client-side*
- Almacenamiento *server-side*
  - *Cache*
  - Bases de datos SQL
  - ✨ **Bases de datos NoSQL**

Bases de datos NoSQL

- Menos restricciones que un esquema relacional
- “Schemaless”
- Facilidad para distribuir



Tipos de bases de datos NoSQL

- **Bases de datos de documentos**
- Bases de datos key - value
- Bases de datos vectoriales
- Bases de datos de grafos

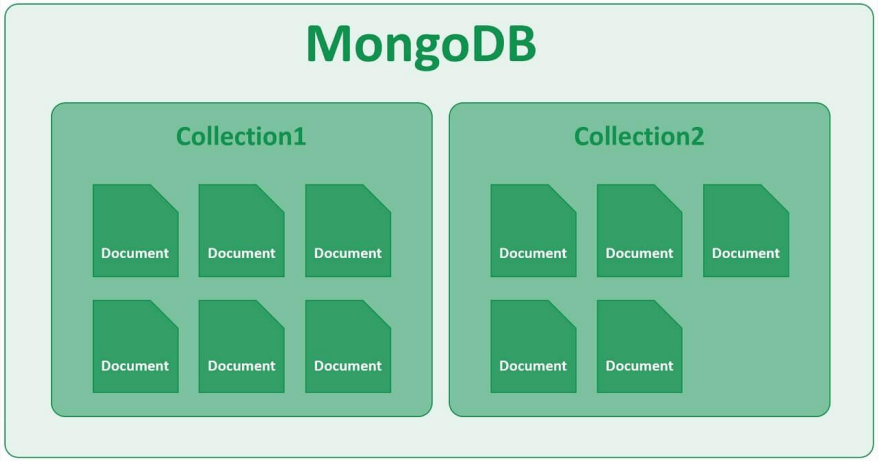


Figura: Relational and non relational databases | Pragim Technologies

Tipos de bases de datos NoSQL



- Bases de datos de documentos
- **Bases de datos key - value**
- Bases de datos vectoriales
- Bases de datos de grafos

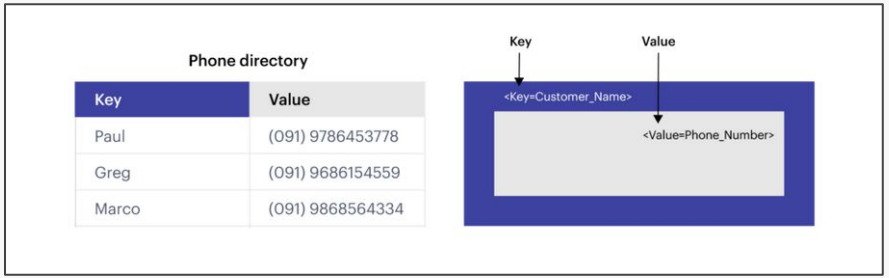


Figura: What is a Key-Value Database? - Redis

Tipos de bases de datos NoSQL

- Bases de datos de documentos
- Bases de datos key - value
- **Bases de datos vectoriales**
- Bases de datos de grafos

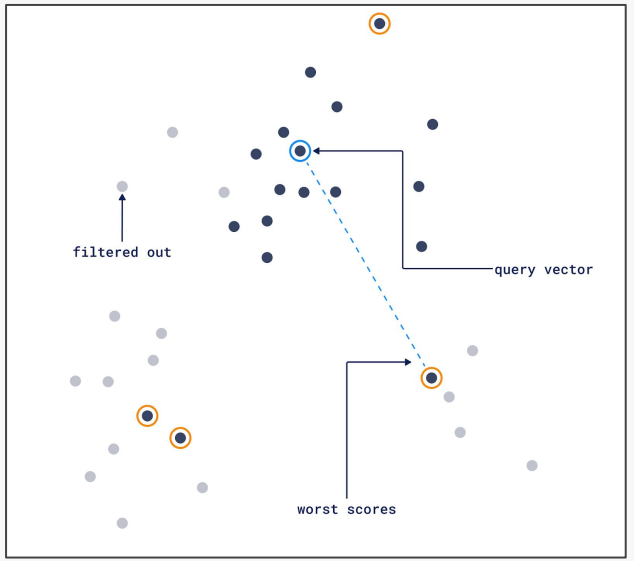


Figura: What is a Vector Database? - Qdrant

Tipos de bases de datos NoSQL

- Bases de datos de documentos
- Bases de datos key - value
- Bases de datos vectoriales
- **Bases de datos de grafos**

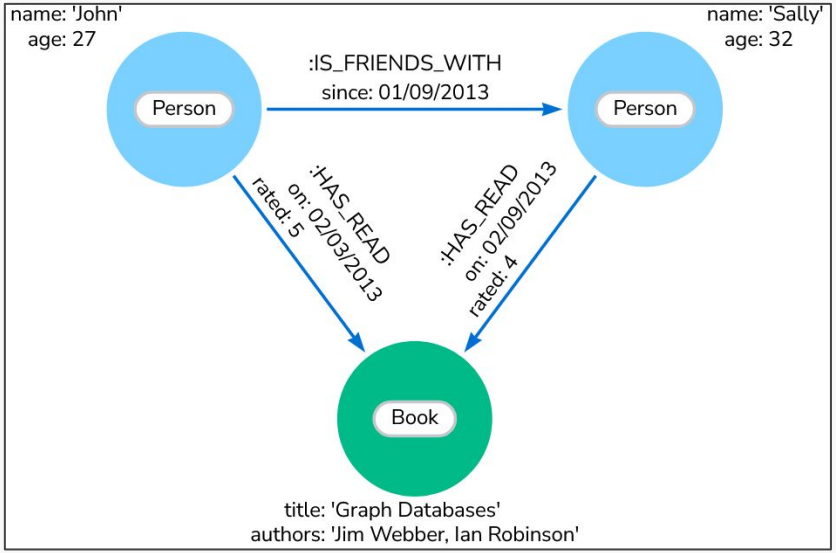


Figura: Graph modeling guidelines - Getting Started

Agenda

Almacenamiento *client-side*

Almacenamiento *server-side*

✨ **¿Qué se viene pronto?**

Clase 16

Almacenes de información

IIC2513 - Tecnologías y Aplicaciones Web

Antonio Ossa Guerra
aaossa@ing.puc.cl