DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

TITOLO

Indice

Sommario

Introduzione

Capitolo 1

Strumenti Web utilizzati nelle PWA

Questa sezione copre le tecnologie web usate nelle Web App che non sono vincolate a uno specifico framework di sviluppo: di esse si fornirà una descrizione delle relative funzionalità e di una loro possibile implementazione in JavaScript.

Alcuni screenshot di questa sezione sono stati catturati dai Developer Tools (DevTools) del browser: per visualizzarli su PC è sufficiente premere F12 (valido per Firefox e per qualunque browser Chromium-based come Google Chrome o Microsoft Edge). Le immagini fanno in particolare riferimento ai DevTools di Google Chrome.

1.1 Cookie

Un *Cookie* è una stringa di testo che viene inviata dal server web al client, il quale avrà poi il compito di memorizzarla e reinviarla al server, senza modifiche, ogni volta che accede alla stessa porzione di uno stesso dominio web [1].

I Cookie hanno una dimensione ridotta: infatti, dato che il browser può dover inoltrare anche centinaia di Cookie durante la navigazione, delle dimensioni eccessive provocherebbero danni alle prestazioni. Essi sono inviati attraverso specifici header del protocollo *HTTP*: nel caso di *HTTP response* viene usato l'header `Set-Cookie` mentre per la *HTTP request* si usa `Cookie` [1]. A un Cookie viene associata inoltre una data di scadenza, oltre la quale non viene considerato più valido [1].

I Cookie possono avere diversi scopi: possono coprire funzionalità necessarie al corretto funzionamento del sito (in tal caso si parla di *cookie tecnici*), possono raccogliere dati in forma anonima a fini statistici (*cookie statistici*) oppure possono tracciare la navigazione dell'utente, con lo scopo di costruire un profilo personalizzato del cliente utile a fornire annunci mirati (*cookie pubblicitari*) [1]. Questi ultimi sono stati oggetto di diverse discussioni, in quanto un loro abuso potrebbe costituire una minaccia alla privacy degli utenti; per questo motivo il loro utilizzo è normato da svariate leggi in diversi stati: in Europa il *GDPR* (*General Data Protectioni Regulation*) ha stabilito che tutti i siti internet che fanno uso di Cookie pubblicitari di terze parti sono obbligati a comunicarlo all'utente all'inizio della sua navigazione nella pagina web [2]. Tale comunicazione deve essere effettuata attraverso

un banner che ostacoli la visualizzazione della pagina: in esso devono essere elencati tutti i Cookie pubblicitari presenti nel sito, specificando anche chi elaborerà i dati raccolti e con quali finalità [2]; è inoltre obbligatorio fornire la possibilità di negare il consenso dei singoli Cookie, non è sufficiente permettere all'utente di negare/consentire tutti i Cookie in blocco [2]. Un Cookie non potrà essere attivato se non dopo il consenso esplicito da parte dell'utente, il quale sarà registrato in opportuna documantazione a testimoniare alle autorità che l'autorizzazione è stata concessa [2]. Il consenso a un Cookie deve essere fornito mediante un'azione non equivoca (come ad esempio cliccare sul pulsante "Acconsento"); azioni come continuare la navigazione sul sito non devono essere considerate come permesso per installare i Cookie [2]. Si deve inoltre fornire la possibilità di modificare le proprie scelte in un secondo momento. All'inizio della navigazione tutti i Cookie di terze parti e non strettamente necessari al corretto funzionamento del servizio devono essere preventivamente disattivati in attesa della scelta dell'utente. Una volta confermata la scelta il *Cookie di consenso* si occuperà di attivare tutti i soli Cookie inserzionistici per il quale l'utente ha dato esplicito consenso [2].
Diversi siti internet fanno uso dei cookie come identificatore di sessione univoco dell'utente (utile ad esempio per permettere al cliente di mantenere l'accesso senza dover fare continuamente login): questo può costituire un rischio per la sicurezza, in quanto un utente malintenzionato può rubare il cookie di qualcun altro e sfruttarlo per impersonare la vittima. Questo problema può essere risolto sfruttando soli Cookie con la flag `"Secure"` (che consente l'invio del Cookie solo attraverso protocollo criptato HTTPS) [1].

1.1.1 implementazione

I Cookie hanno un'interfaccia molto primitiva: non sono definite, infatti, delle funzioni per l'aggiunta, rimozione o la modifica di essi. L'unico modo per inserire, modificare, leggere ed eliminare Cookie è mediante l'attributo `document.cookie` [3].

Per inserire un nuovo Cookie basta assegnare una nuova stringa a `document.cookie` [3]: tale stringa dovrà presentarsi nel formato `"name=value; optionalField1=optionalValue1; optionalField2= optionalValue2;..."` [3]. La coppia `"name=value"` deve essere specificata, altrimenti l'inserimento fallirà silenziosamente, tutte le coppie successive sono invece opzionali. I campi opzionali sono i seguenti:

- `"expires="`: definisce la data di scadenza del Cookie come stringa UTC; per esprimere una data in tale formato è possibile usare il metodo `toUTCString()` della classe `Date` definita in JavaScript [3]. Se non sono specificati né `expires` né `max-age` allora il Cookie scadrà al termine della sessione [3].

- `"max-age="`: specifica la durata del Cookie in secondi [3].

- `"secure"`: indica che il Cookie può essere trasmesso solo attraverso il protocollo criptato HTTPS [3].

- `"partitioned"`: indica che il Cookie deve essere memorizzato in memoria partizionata [3]. Un Cookie partizionato impedisce il tracciamento cross-site dell'utente, meccanismo che viene ad esempio usato dagli inserzionisti per costruire un profilo personalizzato dell'utente utile a

fornire pubblicità mirata [4]. Si Supponga di accedere a un sito `A` che carica un annuncio da un sito di terze parti. Al momento del caricamento, quest'ultimo imposta un Cookie sul dispositivo dell'utente[1]. Si supponga ora di spostarsi a un sito `B`, che carica lo stesso annuncio di prima. Se il Cookie non è partizionato, allora il sito di terze parti sarà in grado di accedere al Cookie definito precedentemente, difatti, in questo caso, la chiave del Cookie è definita solo dall'host che lo ha impostato; il dominio da cui proviene l'inserzione è pertanto in grado di capire che l'utente ha visitato sia `A` che `B`. Se, invece, il Cookie è partizionato, allora il sito di terze parti non sarà in grado di accedere al Cookie definito durante la navigazione in `A`, in quanto, in questo caso, la sua chiave è definita dalla coppia host + sito in cui è caricato il contenuto [4].

- `"domain="`: specifica il dominio a cui il Cookie potrà essere inviato; se non inserito allora assume un valore di default che coincide con l'host della pagina. Si ha inoltre che il Cookie è visibile ai sottodomini solo quando questo parametro è esplicitato [3].

- `"path="`: specifica il percorso del dominio in cui il Cookie è visibile; il Cookie potrà essere inviato solo alla porzione indicata da `path` all'interno del dominio specificato; sono incluse anche eventuali sottodirectory [5]. Se non inserito allora assume il valore di default `"/"` (la *root directory*). `path` e `domain` definiscono assieme l'ambito di visibilità del Coookie [1].

[1] supponendo che l'utente abbia acconsentito a ciò

- `"samesite="`: definisce quando inviare il Cookie al server [3]. Esso può assumere valore `lax` se il Cookie può essere inviato solo in occasione di *same-site requests* e *top-level navigation requests*[2] [3] (in questo secondo caso, però, il Cookie può essere inviato solo attraverso *safe requests*, come `GET` o `HEAD` ma non `POST` [6]), `strict` se si vuole impedire l'invio del Cookie attraverso cross-site requests [3], `none` se non si applica alcun vincolo [3] (in quest'ultimo caso è però richiesto che sia esplicitato il parametro `secure` [5]).

In DevTools è possibile mostrare una visuale dettagliata di tutti i Cookie installati, con anche la possibilità di eliminare quelli indesiderati: per browser Chromium-based basta cliccare su `"Application"` e poi `"Cookies"` nel menù a sinistra, per Firefox invece `"Storage"` e poi `"Cookies"`.
Una volta inserito un Cookie esso non può essere modificato direttamente; è possibile solo sostituire questo con un altro: per farlo basta assegnare a `document.cookie` un nuovo Cookie con lo stesso nome di quello da sovrascrivere [7].
Per quanto riguarda l'eliminazione dei Cookie, l'unica strategia disponibile è quella di sovrascrivere il Cookie con un altro avente scadenza già passata [7].
L'attributo `document.cookie` può anche essere acceduto in lettura: in tal caso si ottiene una lista delle sole coppie nome-valore di tutti i Cookie salvati in quel momento [3]. Tutti gli altri parametri possono essere visualizzati solo da DevTools.

[2] cioè una navigazione a un altro sito che porta alla modifica del contenuto della barra degli indirizzi [6]

Utilizzo nella PWA

Nell'app `PWA-Meteo` viene utilizzato il Cookie `"last-searched"` per salvare il nome dell'ultima città cercata (Figura 1.1), cosicché l'utente possa ottenere nuovamente le relative informazioni meteo al prossimo avvio dell'app stessa. Il Cookie ha una durata di 24 ore e il suo valore viene sovrascritto ogni volta che la chiamata all'API `Open-meteo` termina con successo.
A ogni caricamento della pagina, la funzione `getLastSearchedCity()` (Figura 1.2) si occupa di leggere il valore di `last-searched`.

```
if(city2 != "località sconosciuta") {
    document.cookie = "last-searched=" + city + "; max-age=" + 24*3600 + ";"
}
```

Figura 1.1: Code snippet per la creazione del Cookie `last-searched` (file: `/src/TabellaMeteo.jsx`)

```
function getLastSearchedCity() {
    if(!document.cookie.includes("last-searched")) {
        return null;
    }

    let begin = document.cookie.indexOf('last-searched=') + 'last-searched='.length;
    return document.cookie.substring(begin);
}
```

Figura 1.2: Code snippet della funzione `getLastSearchedCity()` (file: `/src/input.jsx`)

1.2 Local Storage e Session Storage

window.localStorage e *window.sessionStorage* sono entrambi oggetti di tipo `Storage`: in quanto tali, essi permettono di salvare dati in memoria

locale sottoforma di coppie chiave-valore (entrambe stringhe) [8].
I due oggetti differscono per visibilità e durata: `localStorage` serve a contenere dati permanenti (cioè che rimangono in memoria indefinitamente, salvo esplicita rimozione da parte dell'utente o del programmatore) e condivisi fra le varie schede, per cui schede distinte del medesimo browser connesse allo stesso sito condividono lo stesso `localStorage` e, pertanto, modifiche apportate da una pagina sono visibili anche nell'altra pagina [9]. D'altro canto `sessionStorage` è progettato per contenere dati relativi alla singola sessione di navigazione: esso, quindi, non è condiviso fra schede e viene rimosso automaticamente alla chiusura della pagina (viene però conservato quando la pagina è ricaricata) [10].
DevTools fornisce la possibilità di visualizzare `localStorage` e `sessionStorage` di una pagina: per browser Chromium-based è sufficiente cliccare su "`Application`" e poi "`Local Storage`" o "`Session Storage`" mentre per Firefox si deve cliccare su "`Storage`", quindi "`Local Storage`" o "`Session Storage`".

1.2.1 implementazione

Dato che `localStorage` e `sessionStorage` sono entrambi istanze di `Storage`, essi condividono la stessa API, composta dai metodi

- `setItem(key, value)`, che inserisce una nuova coppia chiave-valore in `Storage` se la chiave non è presente, altrimenti sostituisce il precedente valore a essa associato con il nuovo [9].
- `getItem(key)`, che restituisce il valore associato a `key` [9].
- `removeItem(key)`, che rimuove la coppia con chiave `key` [9].

- `clear()`, che svuota l'intero `Storage` [9].
- `key(n)`, che restituisce il nome della `n`-esima chiave [9].

È definito anche l'evento `"storage"`, che viene lanciato quando il contenuto dello `Storage` subisce delle modifiche: questo evento può essere catturato da tutte le schede connesse allo `Storage` diverse dalla pagina che ha modificato il contenuto. Per ascoltare questo evento basta aggiungere un opportuno `event listener` all'oggetto `window` [8].
`localStorage` e `sessionStorage` non sono disegnati per contenere una grande quantità di dati: un oggetto di tipo `Storage`, infatti, ha una capienza massima di 10MB. Se è necessario memorizzare grandi moli di dati bisogna ricorrere a `IndexedDB` [11].

Utilizzo nella PWA

Nell'app `PWA Meteo` si fa utilizzo del session storage per capire se è la prima volta che si sta caricando la pagina all'interno della sessione di utilizzo (e quindi c'è bisogno di inserire i dati meteo relativi alla città salvata nel Cookie `last-searched`) oppure no: questo perché si vuole che siano mostrate le informazioni legate all'ultima località cercata solamente all'avvio dell'app, e non ogni volta che viene aggiornata la pagina. Per fare ciò è stata implementata la funzione `isAtStartup()` (Figura 1.3), la quale può assumere due comportamenti: se il `sessionStorage` è vuoto allora vi inserisce una coppia e poi restituisce `true`, altrimenti restituisce `false`. Quando si avvia l'app, il session storage è sicuramente vuoto e pertanto `isAtStartup()` comunica che c'è bisogno di caricare l'ultima città salvata, poi, a tutti i successivi ag-

giornamenti della pagina, session storage continuerà a contenere la coppia inserita e dunque la funzione restituirà sempre `false`.

```
function isAtStartup() {
    if(!window.sessionStorage.getItem("session")) {
        window.sessionStorage.setItem("session", "active");
        return true;
    }

    return false;
}
```

Figura 1.3: Code snippet della funzione `isAtStartup()` (file: `/src/input.jsx`)

1.3 IndexedDB

Gli *`IndexedDB`* (*Indexed database*) costituiscono un'altra strategia con cui le Web App possono immagazzinare dati nel Browser dell'utente. Un `IndexedDB` implementa un *Object Oriented Database Management System* (*OODBMS*) e, in quanto tale, memorizza coppie chiave-valore.

A differenza di `localStorage` e `sessionStorage`, `IndexedDB` non presenta limiti di spazio (in questo caso la capienza massima è definita dal singolo browser), inoltre chiavi e valori possono essere un qualunque oggetto, non per forza stringhe.

Mentre in un *RDBMS* (*Relational Database Management System*) i dati sono memorizzati come righe all'interno di una tabella, in un OODBMS essi sono salvati come oggetti all'interno di *Object Store*: all'interno di un Object Store ciascun record sarà identificato da una *chiave* univoca (come nei RDBMS),

che può essere sfruttata per ottenere il riferiìmento al valore associato.
Oltre che con le chiavi, gli OODBMS possono essere interrogati anche attraverso *indici*: un indice è un Object Store ausiliario che si riferisce a un altro Object Store, e che serve a effettuare ricerche su quest'ultimo. A titolo di esempio, si supponga di implementare un semplice database che registri i voti di un esame: esso è composto da un solo Object Store "*exam*", in cui ogni record ha per chiave la matricola dello studente e per valore il suo voto. È ora possibile costruire un indice "*grades*" che permette di effettuare ricerche in base ai voti (ad esempio cercare tutti gli studenti che hanno preso trenta e lode).
All'interno di un indice cisacun record ha valore che coincide con la chiave di un altro record all'interno dell'Object Store riferito; inoltre, ogni volta che il contenuto di quest'ultimo subisce una modifica, l'indice viene aggiornato automaticamente.
Quando si effettua una ricerca tramite indice, il risultato che viene restituito non è una lista di record bensì un *cursore*, oggetto ausiliario negli OODBMS che permette di iterare fra record di un Object Store.
Gli indexed database applicano un modello basato su *transazioni*, cioè una sequenza di operazioni che va a costituire un'unità logica di lavoro: qualunque modifica apportata al database deve essere eseguita entro una transazione. Le transazioni sono atomiche, ciò vuol dire che esse possono o essere completate nella loro interezza o non avere alcun effetto, dunque, se una delle operazioni della sequenza fallisce, è necessario annullare tutte le modifiche effettuate dalle precedenti operazioni. Inoltre, il fatto che le transazioni siano atomiche le rendono sicure nel caso in cui più finestre del browser tentino di

accedere concorrentemente al database.
La indexedDB API presenta delle limitazioni, fra cui il fatto che è prpogettata per l'amministrazione di un server solamente Client-Side, non sono forniti strumenti per la gestione Server-Side.

1.3.1 Implementazione

La IndexedDB API è principalmente asincrona, ciò vuol dire che la maggioranza delle operazioni sulla base di dati è eseguita per mezzo di *richieste*, rappresentate mediante oggetti di tipo `IDBRequest`: esse presentano i metodi `onSuccess()` e `onError()` che permettono di definire delle callback functions da eseguire rispettivamente in caso di successo o errore.
È possibile inviare una richiesta di apertura della connessione con il database per mezzo della funzione `window.indexedDB.open(name, version)`, dove `name` è il nome del database (che deve essere unico all'interno del dominio) e `version` è la sua versione. Aprire un database con lo stesso nome di uno già esistente ma numero di versione maggiore viene interpretato come la richiesta di effettuare un aggiornamento sul database stesso e causa quindi il lancio dell'evento `upgradeneeded`: per gli oggetti di tipo `IDBOpenDBRequest` (la richiesta restituita da `open()`), è presente anche il metodo `onupgradeneeded()` che definisce che funzione eseguire in questa situazione.

1.4 Service Worker

Bibliografia

[1] "Cookie." https://it.wikipedia.org/wiki/Cookie.

[2] "GDPR Cookies, Consent and Compliance." https://www.cookiebot.com/en/gdpr-cookies/.

[3] "Document: cookie property." https://developer.mozilla.org/en-US/docs/Web/API/Document/cookie.

[4] "Cookies Having independent Partitioned State (CHIPS)." https://developer.mozilla.org/en-US/docs/Web/Privacy/Guides/Privacy_sandbox/Partitioned_cookies.

[5] "Using HTTP cookies." https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Cookies.

[6] Prima risposta al thread *"What is the difference between SameSite="Lax" and SameSite="Strict""* di Stack Overflow https://stackoverflow.com/questions/59990864/what-is-the-difference-between-samesite-lax-and-samesite-strict.

[7] "Javascript cookies." https://www.w3schools.com/js/js_cookies.asp.

[8] "Using the Web Storage API." https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API.

[9] "Window: localStorage property." https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage.

[10] "Window: sessionStorage proprerty." https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage.

[11] "IndexedDB key characteristics and basic terminology." https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Basic_Terminology.