Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

Mobile Crowdsensing da browser: uno studio sulla compatibilità

Relatore:
Dott.
FEDERICO MONTORI

Presentata da:
PIERO STABELLINI

Sessione I
Anno Accademico 2023/2024

Alla mia famiglia ...

Abstract

Sensorworker è un'applicazione web per il mobile crowdsensing integrata nella piattaforma di crowdsourcing Microworkers.

Questa applicazione utilizza una serie di API JavaScript per accedere ai sensori hardware dei dispositivi ed effettuare rilevamenti sensoriali. Tuttavia, a causa delle diverse politiche di adozione delle API da parte dei browser per tutelare la privacy degli utenti, Sensorworker era inizialmente compatibile solo con dispositivi Android che utilizzassero Google Chrome, limitando fortemente la portabilità e l'accessibilità a livello globale.

Questo documento affronta il problema di compatibilità tra queste API e i browser più diffusi. Per risolverlo, vengono prima chiariti i concetti di Crowdsourcing e Crowdsensing e i loro contesti di utilizzo. Successivamente, si analizzano le API Web utilizzate da Sensorworker e le compatibilità dichiarate dal W3C per diversi browser. Vengono poi condotti alcuni test sulle campagne della web app e analizzate le discrepanze tra la compatibilità dichiarata e quella testata, implementando funzioni nella nuova versione dell'applicazione per ottimizzare l'esperienza utente e la stabilità del sistema.

Introduzione

Il mondo dei dispositivi *mobile* è in costante fermento e crescita. Gli strumenti a nostra disposizione diventano sempre più raffinati ed accurati, contribuendo in maniera significativa allo sviluppo di diverse branche del panorama tecnologico odierno, in particolar modo grazie a diversi paradigmi di raccolta dati.

In questo contesto, emerge l'importanza dei sistemi di Consapevolezza Collettiva (CAP, *Collective Awareness Paradigms*), i quali sfruttano il potere collettivo di una moltitudine di persone per raccogliere, analizzare e migliorare dati a beneficio della comunità, ampliando così nuove frontiere di conoscenze e servizi, contribuendo significativamente al progresso della società [1].

I dispositivi mobili, essendo sempre più potenti e diffusi, diventano strumenti fondamentali per la partecipazione attiva dei cittadini nella raccolta di dati ambientali, infrastrutturali e sociali, attraverso lo sfruttamento di tecnologie avanzate e metodi innovativi dei sistemi CAP tra cui il *crowdsourcing* e il *mobile crowdsensing*.

Queste piattaforme si basano su diversi principi tra cui il concetto di intelligenza collettiva o saggezza della folla. L'idea consiste nel fatto che l'aggregazione di dati e osservazioni da parte di una folla di individui può fornire una visione più completa ed accurata del fenomeno oggetto dello studio, rispetto a ciò che un singolo individuo potrebbe ottenere.
La consapevolezza emergente dalla folla dunque può portare a comprendere meglio un particolare fenomeno e adottare politiche più efficaci per sfruttare

l'evento a nostro vantaggio.

Le applicazioni più note e diffuse del prodotto di questa intelligenza collettiva sono molteplici, tra cui:

- Internet of Things (IoT), un paradigma tecnologico che consente la connessione a internet di diversi dispositivi in grado di comunicare tra loro e scambiarsi informazioni.
  I dati raccolti vengono utilizzati ad esempio per il monitoraggio ambientale, migliorare il traffico e la sicurezza urbana;

- Intelligenza Artificiale, che sfrutta i dati raccolti come dataset per l'addestramento degli algoritmi di machine learning;

- Analisi dei social media per monitorare la valutazione di un brand o identificare i trends;

- Ricerca scientifica per la creazione di nuove conoscenze e l'innovazione di prodotti e servizi.

Nel corso di questo documento verrà analizzato in particolar modo l'approccio del *Mobile Crowd Sensing (MCS)*, presentando un'applicazione che adotta tale strategia, *Sensorworker*.

In seguito lo studio esaminerà in dettaglio le specifiche tecniche e le API utilizzate dall'applicazione web Sensorworker per interagire con i sensori responsabili della raccolta dei dati.

Il focus di questa ricerca si concentra sull'analisi della compatibilità delle principali API che consentono l'accesso ai sensori dei dispositivi mobili con i browser più diffusi attualmente disponibili.

Infine, saranno presentati i risultati di alcuni tests condotti su Sensorworker, finalizzati a verificare la compatibilità precedentemente discussa.

Indice

Elenco delle figure

Elenco delle sezioni di codice

Capitolo 1

Stato dell'arte

Nel contesto dell'evoluzione tecnologica odierna, l'integrazione tra dispositivi mobili e Internet delle Cose (IoT) ha dato nuovo impulso a tecniche innovative per la raccolta e l'analisi dei dati, tra cui il Mobile Crowdsensing (MCS). Il MCS sfrutta la partecipazione di una moltitudine di individui per raccogliere dati attraverso i loro dispositivi mobili, generando informazioni utili per vari ambiti, che spaziano dall'analisi ambientale alla gestione delle infrastrutture.

Nei prossimi paragrafi vengono introdotti alcuni concetti fondamentali.

1.1 Crowdsourcing e Crowdsensing

Crowdsourcing

Il termine *"Crowdsourcing"* compare per la prima volta in un articolo del giornalista Jeff Howe nella rivista *Wired* [2].
Il termine deriva dalla combinazione delle parole "crowd" (folla) e "outsourcing" (esternalizzazione) e descrive un modello di lavoro che sfrutta la partecipazione di una moltitudine di persone per eseguire dei compiti che in precedenza venivano svolti internamente all'organizzazione dai dipendenti stessi.

Dalla definizione di crowdsourcing si può dunque intuire che tale approccio si basa su una serie di caratteristiche fondamentali tra cui: la partecipazione volontaria di un gran numero di individui; la diversità di compiti; la chiamata aperta (le proposte infatti sono solitamente esposte su piattaforme online accessibili a tutti) ed il vantaggio reciproco da parte dell'organizzazione o istituto che beneficia del lavoro svolto, sia per i partecipanti che ricevono benefici quali compensi, riconoscimenti o sviluppo di competenze.

Howe, inoltre, ha individuato una serie di modelli di crowdsourcing che variano in base a diversi fattori, tra cui la natura del compito, l'obiettivo del progetto e le competenze richieste [3]:

- *Collective intelligence, o Crowd Wisdom*, la saggezza collettiva della folla è utilizzata per prendere decisioni o risolvere problemi;
- *Crowd creation*, la folla contribuisce con contenuti creativi come disegni, articoli, foto e video;
- *Crowd voting*, la folla ricopre un ruolo attivo nel votare o valutare idee, contenuti o prodotti;
- *Crowdfunding*, ai membri volontari viene richiesto di finanziare personalmente delle iniziative attraverso piccole donazioni. Questo modello viene usato in modo molto diffuso nelle realtà imprenditoriali.

Crowdsensing

Un altro modello che estende i sistemi di Consapevolezza Collettiva (CAP) e che richiama alcune delle caratteristiche riportate nel paragrafo precedente è il Mobile Crowdsensing (MCS).

Il Mobile Crowdsensing o Crowdsourcing sensoriale, è un approccio di raccolta dati che sfrutta la diffusione e la versatilità dei dispositivi mobili degli utenti, come smartphone e sensori indossabili, per raccogliere dati dall'ambiente circostante attraverso la partecipazione di un gran numero di individui.

Questa metodologia combina il crowdsourcing e il sensoring (sensori) per ottenere informazioni sottoforma di dati sensoriali, provenienti da dispositivi dotati di capacità di calcolo e rilevamento oltre alla connessione a Internet, come smartphones, sistemi di gioco o sensori di rilevamento posizionati sui veicoli.

Esistono due forme principali di crowdsensing [1]. Nel crowdsensing partecipativo (PCS) l'utente è coinvolto attivamente nella raccolta dei dati ed interagisce con un'applicazione che gli consente di effettuare le letture dei dati richieste (ad esempio cliccando un pulsante o scattando una foto).
Viceversa il crowdsensing opportunistico (OCS) sfrutta il monitoraggio dei dati tramite lavori autonomi dell'applicazione in background (come il rilevamento costante della posizione per registrare il percorso seguito da un utente durante un'attività sportiva).

1.2 Esempi di app che adottano il MCS

Le applicazioni che si basano sul mobile crowdsensing si possono classificare in tre categorie a seconda del fenomeno studiato: ambientali, infrastrutturali e sociali [4]. Di seguito vengono riportati alcuni esempi di applicazioni che utilizzano questo modello.

App ambientali

Questa categoria di applicazioni è volta a studiare fenomeni direttamente correlati all'ambiente naturale come l'inquinamento dell'aria, il monitoraggio del livello dell'acqua nei fiumi o della fauna selvatica.

Un esempio è CreekWatch, sviluppata dal centro di ricerca IBM Almaden, che ha come obiettivo il monitoraggio dei livelli dell'acqua nei torrenti. Gli utenti che prendono parte a questa campagna sono coinvolti attivamente dall'applicazione, il cui programma richiede di fotografare periodicamente il livello dell'acqua in punti diversi di uno stesso fiume. Gli utenti hanno inol-

tre la possibilità di segnalare tramite messaggi testuali eventuali situazioni critiche di inquinamento dell'acqua [5] [6].

App infrastrutturali

Questa seconda tipologia di applicazioni si rivolge allo studio delle misurazioni di dati su larga scala circa le infrastrutture pubbliche, come la misura della congestione del traffico, la disponibilità di parcheggi, le interruzioni di servizi pubblici per malfunzionamenti ed altri fenomeni di questo genere.

L'applicazione OpenSignal, ad esempio, mappa la copertura e le prestazioni delle reti mobili su scala globale [7].

Street Bump, sviluppata dalla città di Boston (USA), si schiera invece sul fronte degli interventi infrastrutturali incentivando la partecipazione attiva dei propri abitanti nel segnalare buche e generici problemi infrastrutturali alle strade della città, in modo da velocizzare ed ottimizzare la procedura di riparazione del pericolo segnalato. L'applicazione utilizza l'accelerometro e il GPS dello smartphone per rilevare vibrazioni evidenti durante la guida [8].

Infine, tra le più note applicazioni di questa categoria nel promuovere un sistema basato su MCS si colloca Waze, un'applicazione mobile di navigazione stradale, sviluppata da Waze Mobile, che utilizza i dati raccolti dalla sua community per fornire aggiornamenti in tempo reale sul traffico [9].
L'applicazione lato client in funzione sullo smartphone invia periodicamente dati relativi alla posizione attuale del dispositivo, acquisita tramite il ricevitore GPS dello smartphone o del dispositivo mobile, al server Waze che istantaneamente restituisce il piano di navigazione più conveniente ed ottimizzato per raggiungere la destinazione prescelta.
Oltre alla generazione dei percorsi, l'applicazione fornisce l'opportunità ai propri utenti di partecipare al costante monitoraggio del traffico, mettendo a disposizione una serie di features che permettono di segnalare ingorghi, blocchi stradali da parte delle forze dell'ordine, ostacoli temporanei, incidenti, autovelox e pericoli di varia natura ed, infine, l'aggiornamento dei prezzi dei carburanti e della cartografia stessa.

Esempi similari, tra i più noti, sono Google Maps [10] e Apple Maps, sviluppata dalle rispettive celebri aziende. Queste due si differenziano per un approccio di rilevamento meno partecipativo da parte dell'utente. Le informazioni sul traffico vengono infatti catturate ed elaborate in maniera più silenziosa prelevando i dati da altri sensori oltre al segnale GPS e calcolando l'accelerazione e la velocità lineare dei dispositivi che si trovano nella stessa area geografica per determinare l'intensità del traffico.

App sociali

Quest'ultima categoria accomuna le applicazioni che hanno come obiettivo la valorizzazione della partecipazione, offrendo un senso di appartenenza e gratificazione per il contributo fornito alla comunità. Altre caratteristiche che distinguono queste applicazioni includono il supporto di obiettivi personali (monitoraggio dell'attività fisica, delle abitudini alimentari o altri comportamenti quotidiani), la creazione di reti sociali e la condivisione delle informazioni tra utenti.

L'esempio per eccellenza di questa categoria è Strava, un'applicazione che permette di registrare e condividere gli allenamenti della propria community [11]. Oltre alla possibilià di registrare un'attività sportiva tramite GPS (tra le tante ciclismo, running e nuoto), l'applicazione si presenta anche come un social network per atleti e sportivi, mettendo a disposizione una serie di classifiche e sfide temporanee per coinvolgere i propri utenti a migliorarsi nel proprio allenamento. Strava utilizza i diversi sensori dello smartphone per generare ed elaborare una serie di statistiche sull'attività svolta dai propri utenti, come la velocità media e massima, il percorso seguito, il dislivello percorso e molto altro.

1.3 Applicazioni native e applicazioni web

Nel contesto del mobile crowdsensing, dove gli utenti contribuiscono alla raccolta di dati attraverso compiti minimi (microtasks), la scelta tra l'uso di

applicazioni web e applicazioni native gioca un ruolo cruciale nell'ottimizzazione dell'esperienza utente e dell'efficacia complessiva del sistema.

Una applicazione web può peccare sulla reattività e sul punto di vista estetico dell'interfaccia utente ma offre diversi vantaggi [12]:

1. Accessibilità: questo è uno dei principali vantaggi delle applicazioni web. Le web app possono essere utilizzate istantaneamente su qualsiasi dispositivo connesso a Internet, senza necessità di download da uno store. Questo elimina gli ostacoli all'adozione, essenziali in contesti dove la partecipazione è basata su microtasks con ricompense minime. Dunque si cerca di eliminare operazioni che possono risultare scomode, noiose e poco pratiche che possono rappresentare uno sforzo aggiuntivo non giustificato dalla ricompensa.

2. Accesso alle funzionalità native: l'approccio web non vieta l'accesso alle funzionalità e ai sensori nativi del dispositivo, che sono raggiungibili e garantite tramite opportune API Javascript o Typescript.

3. Spazio di archiviazione: le applicazioni web sono generalmente più leggere delle applicazioni native. Non richiedono spazio di archiviazione significativo sul dispositivo dell'utente, evitando così problemi di memoria insufficiente che possono sorgere soprattutto su dispositivi più datati o con capacità di storage limitate.

4. Compatibilità: le applicazioni web vantano una compatibilità universale con una vasta gamma di dispositivi e sistemi operativi. Questo elimina i problemi di compatibilità che spesso sono un ostacolo per le applicazioni native.

5. Consistenza della user experience e Portabilità: le web app offrono un'esperienza utente consistente su vari dispositivi e piattaforme. Gli utenti possono passare senza problemi dal proprio smartphone al tablet o al computer, continuando a lavorare sulle stesse microtasks sen-

za interruzioni, una notevole importanza in un mondo sempre più interconnesso.

6. Aggiornamenti: gli aggiornamenti del software avvengono in tempo reale senza richiedere alcuna azione da parte dell'utente. Al contrario, le applicazioni native richiedono che l'utente scarichi manualmente gli aggiornamenti

7. Sicurezza e Privacy: un ultimo aspetto lato utente riguarda la sicurezza e la privacy. Tali applicazioni limitano l'accesso diretto al sistema operativo del dispositivo, sono meno vulnerabili a certi tipi di attacchi informatici.

8. Sviluppo: il costo dello sviluppo e del mantenimento di un'applicazione di questo tipo è sicuramente più ridotto rispetto alle applicazioni native, poiché non è necessario creare versioni separate a seconda del sistema operativo utilizzato.

In sintesi, l'utilizzo di un'applicazione web, offre numerosi vantaggi significativi rispetto ad applicazioni native. Questi vantaggi si traducono in migliori prestazioni, accessibilità immediata, compatibilità e richiede meno impegno da parte dell'utente. Pertanto, la scelta di utilizzare una web app per il mobile crowdsensing non solo migliora l'esperienza utente, ma anche l'efficacia complessiva del sistema, rendendola una scelta strategica vincente in questo contesto.

Capitolo 2

Sensorworker

Dopo aver affrontato e discusso i vantaggi significativi delle applicazioni web rispetto alle applicazioni native nel contesto del mobile crowdsensing, è il momento di introdurre un'applicazione innovativa che sfrutta appieno queste potenzialità: *Sensorworker* [13].

Prima di passare ad un'analisi più tecnica e dettagliata dell'applicazione presentata, è utile comprendere il contesto in cui Sensorworker opera, ovvero la nota piattaforma di crowdsourcing Microworkers.

2.1 Microworkers.com

Microworkers [14] è una piattaforma ben nota nel campo del crowdsourcing, con un'ampia comunità di utenti attivi, i quali svolgono regolarmente microtasks in cambio di compensi monetari.

Nel panorama del crowdsourcing online, esistono diversi tipi di compiti richiesti ai lavoratori [15]. Alcuni di questi compiti richiedono competenze tecniche e specifiche, specialmente nella "Crowd Creation", dove possono essere necessarie abilità nel campo della produzione ed editing di video o audio, lavori di design o scrittura.
All'altro estremo si trovano le microtasks o microworks. Il termine stesso

suggerisce una tipologia di compiti più semplici, come classificare immagini, inserire dati basilari o interagire sui social media tramite like e commenti.

Microworkers si colloca proprio in quest'ultima categoria. Questa piattaforma online facilita il crowdsourcing, permettendo alle aziende di esternalizzare microtask a una vasta rete globale di lavoratori. La piattaforma mette a disposizione dei propri utenti un elenco di "campagne", ovvero progetti creati dai crowdsourcer, ciascuno dei quali include una serie di compiti specifici (microtask), semplici ma ripetitivi da completare. I lavoratori hanno accesso ad una varietà di microtasks, che possono svolgere in cambio di piccoli pagamenti. Questa piattaforma si distingue per la sua interfaccia user-friendly ed intuitiva, la possibilità di creare campagne di crowdsourcing altamente personalizzabili ed una robusta infrastruttura di supporto per i lavoratori.

2.2 Vantaggi offerti da Sensorworker

Sensorworker è una web app progettata per integrarsi con la piattaforma Microworkers, sfruttando la sua consolidata politica di completamento di microtasks e il sistema di remunerazione.
Grazie a questa sinergia, Sensorworker beneficia di una vasta base di utenti già attivi su Microworkers, ampliando il servizio di crowdsourcing con l'introduzione di un efficiente sistema di crowdsensing.

Le microtasks di Sensorworker sono progettate per sfruttare i diversi sensori presenti sui dispositivi mobili. Questi sensori includono videocamera, microfono, accelerometro, giroscopio, magnetometro, GPS, sensore di luce ambientale, sensore di accelerazione di gravità, sensore di accelerazione lineare, sensore di orientamento assoluto e relativo. Gli utenti possono essere coinvolti in una varietà di attività, come registrare video, audio, raccogliere dati di movimento o posizione, contribuendo così a progetti di ricerca o commerciali, raccogliendo dati ricchi e diversificati da una vasta gamma di fonti.

Sensorworker offre un portale dedicato ai crowdsourcer, ovvero i creatori delle campagne. Attraverso un'interfaccia utente semplice e intuitiva, i crowdsourcer possono creare campagne di Sensorworker in pochi secondi. La creazione di una campagna richiede l'inserimento di alcuni dettagli essenziali:

- Titolo e descrizione: identificano e descrivono la campagna;
- Limiti spaziali: specificano l'area geografica in cui le task possono essere svolte (definita da un array di coordinate che forma un poligono);
- Buffer di distanza: indica l'area entro cui gli utenti possono prendere parte alla campagna ma non eseguire le task;
- Lista di sensori: elenca i sensori da utilizzare, il numero di misurazioni, la distanza e il tempo che deve intercorrere tra una misurazione e la successiva;
- Domanda: un campo opzionale che permette al crowdsourcer di raccogliere messaggi di testo scritti direttamente dai workers (come opinioni, feedback sulla campagna e molto altro);
- Time to complete (TTC): tempo a disposizione del worker per completare il compito;
- Limite temporale: durata complessiva della campagna.

Il portale consente inoltre ai crowdsourcer di monitorare l'andamento delle task e di valutare i contributi degli utenti, accettando o rifiutando i dati raccolti.

Una campagna in Sensorworker richiama il medesimo concetto in Microworkers, con una differenza però sostanziale: al momento della registrazione di un nuovo crowdsourcer su Sensorworker, viene creata una campagna su Microworkers, associata univocamente a quest'ultimo, che funge da contenitore per quelle di Sensorworker create da quel determinato crowdsourcer. Gli

utenti di Microworkers possono accedervi, visualizzare la lista delle campagne di Sensorworker associate e scegliere a quale partecipare.

Una volta avviata, il lavoratore può effettuare le letture dei sensori richieste dalla campagna stessa utilizzando un'interfaccia semplice e intuitiva (figure 2.1 e 2.2). Questa interfaccia presenta una serie di bottoni associati ai rispettivi sensori, che consentono al lavoratore di raccogliere facilmente i dati richiesti.

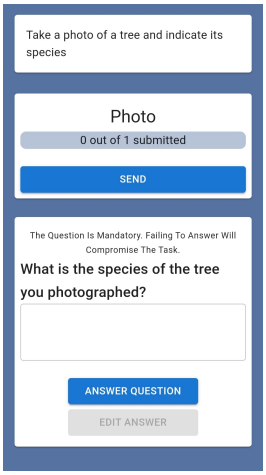

Figura 2.1: *Campagna1, raccolta dei dati multimediali*

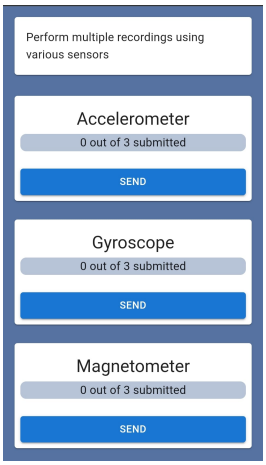

Figura 2.2: *Campagna 3, raccolta dei dati sensoriali*

In conclusione, Sensorworker rappresenta una soluzione all'avanguardia nel campo del mobile crowdsensing, unendo l'accessibilità e la portabilità delle applicazioni web alla potenza e alla flessibilità dei dispositivi mobili. Integrandosi con una piattaforma consolidata come Microworkers, Sensorworker è in grado di offrire una soluzione completa e integrata per la raccolta di dati tramite dispositivi mobili, rendendo il processo semplice e conveniente sia per i crowdsourcer che per i workers.

2.2.1 Il problema di Sensorworker

Prima dell'implementazione della nuova funzionalità, che sarà descritta nel capitolo successivo e costituisce il focus principale di questo studio, Sensorworker presentava un significativo problema di compatibilità tra i diversi browser. L'applicazione era infatti configurata per funzionare esclusivamente su Google Chrome, limitando così l'accesso a un'ampia gamma di dispositivi e in particolare agli utenti con smartphone iOS.

Questo problema si verifica perché, a differenza delle applicazioni native che interagiscono direttamente e in modo sicuro con l'hardware, inclusi i sensori, le applicazioni web devono affidarsi alle API JavaScript per accedere a determinate funzionalità hardware. Tuttavia, queste API non sono supportate in modo uniforme da tutti i browser. Alcuni browser, infatti, scelgono di non adottarle a causa delle loro politiche restrittive riguardo alla privacy degli utenti.

Nel capitolo successivo, questo problema di compatibilità viene analizzato in dettaglio, e viene presentata una soluzione specifica per affrontarlo.

Capitolo 3

Compatibilità tra browser e sensori

Prima di esaminare le diverse API utilizzate in Sensorworker per accedere ai sensori, è importante introdurre la fonte principale di riferimento per questo studio: *MDN Web Docs* [16]. Le MDN Web Docs costituiscono il repository principale di Mozilla, che fornisce una vasta documentazione sugli standard web alla comunità globale degli sviluppatori. Tra le risorse offerte, la sezione dedicata alle Web APIs è particolarmente rilevante, in quanto include una serie di interfacce per la creazione di applicazioni web, come la Sensor API [17].

Le MDN Web Docs forniscono implementazioni basate sulle specifiche delineate dal W3C (*World Wide Web Consortium*), un'organizzazione internazionale che promuove l'interoperabilità tra le diverse tecnologie web mantenendo standard aperti.

Grazie alla collaborazione avviata nel 2016 [18], le MDN Web Docs rappresentano una fonte affidabile e aggiornata per lo sviluppo di siti web e applicazioni. Tra le documentazioni disponibili, vi sono diverse API per interagire con i sensori dei dispositivi, basate sugli standard del W3C. Queste API forniscono agli sviluppatori i mezzi per interagire con i sensori dei dispositivi.

Prima di analizzare nel dettaglio le diverse API dei sensori è utile sottolineare che, secondo il W3C [19], le implementazioni delle APIs dei sensori possono essere suddivise in tre categorie principali basate su vari fattori come la fase di sviluppo, stabilità, sicurezza e supporto. Queste categorie sono:

- Tecnologie ben implementate: includono APIs dei sensori ampiamente implementate e testate nei browser web, considerate stabili e affidabili. Esempi di queste APIs sono Geolocation e Media Capture and Streams;
- Tecnologie in fase di sviluppo: comprendono APIs dei sensori attualmente in sviluppo, che potrebbero ancora presentare problemi di stabilità o compatibilità. Tra queste APIs troviamo Generic Sensor, Proximity Sensor, Ambient Light Sensor, Battery Status, Accelerometer, Gyroscope, Magnetometer, Orientation Sensor, DeviceOrientationEvent Specification e Screen Orientation;
- Tecnologie in fase di sperimentazione: includono APIs dei sensori ancora in fase di sperimentazione, non ampiamente implementate nei browser web. Questa categoria comprende le APIs NFC e Bluetooth.

Problema di Privacy Policy

In questo capitolo, verranno analizzate alcune delle APIs precedentemente menzionate ed utilizzate in Sensorworker per accedere ai dati dei sensori. Per ogni API, saranno presentate le specifiche tecniche, come le modalità di autorizzazione e gestione dei permessi, nonché la compatibilità con i vari browser web, al fine di fornire una panoramica completa e aggiornata sulla compatibilità delle APIs dei sensori.

L'accesso ai dati forniti dai sensori dei dispositivi da parte delle applicazioni web offre numerosi vantaggi e potrebbe portare a enormi potenzialità in futuro. Tuttavia, non tutti i browser supportano le APIs di accesso ai sensori, principalmente per motivi legati alla privacy degli utenti. L'accesso indiscriminato ai sensori da parte delle applicazioni web può comportare abusi, come il *browser fingerprinting*, una tecnica di tracciamento online che

sfrutta caratteristiche uniche del browser per identificare e monitorare l'attività degli utenti, con potenziali rischi di furto di identità e sorveglianza di massa.

Per questo motivo, alcuni browser come Apple Safari e Mozilla Firefox hanno adottato misure per proteggere la privacy degli utenti, limitando o bloccando l'accesso a diverse APIs dei sensori. Ad esempio, Apple Safari [20], nel giugno 2020, e Mozilla Firefox [21], dalla versione 72 (2020) con l'introduzione dell'Enhanced Tracking Protection, hanno deciso di non implementare o bloccare l'accesso a 16 APIs dei sensori che estendono l'interfaccia Sensor, sia nella versione per PC che mobile. Queste decisioni riflettono il loro impegno nella difesa della privacy degli utenti.

Al contrario, altri browser come Google Chrome, Microsoft Edge ed Opera non hanno adottato restrizioni simili, consentendo l'accesso ai sensori tramite le APIs disponibili. Di seguito, verrà fornita un'analisi delle APIs sopra citate e uno studio sulla compatibilità dei browser con i sensori utilizzati.

Interpretazione tabelle di compatibilità

Come accennato in precedenza di seguito viene proposta un'analisi dettagliata di ciascuna Web API utilizzata in Sensorworker per accedere alle letture dei sensori.

Per ogni libreria considerata, verrà presentata una tabella che riassume la compatibilità dei diversi browser con l'API in questione. Per facilitare la lettura e la comprensione delle tabelle, è importante tenere presente la legenda riportata in figura 3.1. Nelle tabelle, le colonne rappresentano i vari browser, mentre le righe indicano le rispettive versioni. È importante notare che l'ultima riga di ogni tabella mostra la compatibilità di ciascun browser con la sua ultima versione di rilascio, mentre le righe precedenti mostrano le versioni più vecchie.

Per quanto riguarda la codifica a colori, le celle verdi indicano che la versione del browser supporta l'API, quelle rosse indicano che non la supporta

e le celle grigie indicano che il supporto per l'API è sconosciuto per quella particolare versione del browser.

Infine per alcune API sarà di fondamentale importanza tenere in considerazione la presenza, in alcune celle di colore rosso, di piccole flag di colore verde nell'angolo in alto a destra della cella stessa. La presenza di questa flag indica che il supporto del browser per la relativa API è assente di default in quanto la tecnologia è in fase sperimentale. Tuttavia, l'utente può scegliere di abilitare il supporto direttamente dalle impostazioni del browser, attivando la preferenza corrispondente alle voci:

enable-generic-sensor-extra-classes per Magnetometer API;

enable-experimental-web-platform-features per Ambient Light Sensor API;

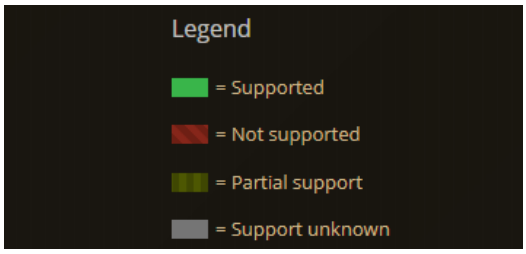

Figura 3.1: *Legenda tabelle compatibilità*

3.1 Geolocation API

L'API di geolocalizzazione [22] definisce un'interfaccia di alto livello per accedere alle informazioni geografiche del dispositivo, indipendentemente dalla tecnologia utilizzata (come GPS, Wi-Fi, reti cellulari). Quando l'utente concede l'autorizzazione, questa API fornisce vari dati sulla posizione del dispositivo, inclusi latitudine, longitudine, altitudine, velocità, direzione e l'istante di acquisizione.

Le applicazioni possono gestire separatamente la cache delle posizioni e utilizzarla insieme all'API di geolocalizzazione, a condizione che i dati memorizzati non siano più vecchi di un determinato intervallo di tempo.
Per utilizzare l'API di geolocalizzazione, si effettua una chiamata a *navigator.geolocation*. Il browser dell'utente richiederà quindi l'autorizzazione per accedere ai dati di posizione.

L'uso dell'API di geolocalizzazione può rappresentare un rischio per la privacy dell'utente, in quanto può svelarne la posizione. Pertanto, è necessario ottenere l'autorizzazione esplicita dell'utente prima che un'applicazione web possa accedere a questi dati. Questo requisito è imposto a livello normativo e si basa sui passaggi di verifica dell'autorizzazione presenti nei metodi *getCurrentPosition()* (che recupera la posizione corrente del dispositivo) e *watchPosition()* (che registra una funzione di gestione che verrà chiamata automaticamente ogni volta che cambia la posizione del dispositivo, restituendo una posizione aggiornata).
Gli utenti hanno generalmente diverse opzioni per determinare la durata dell'autorizzazione, che può essere basata sul tempo (ad esempio, 24 ore o una settimana) o sulla durata della sessione (consigliata).
La verifica dei permessi di accesso ai dati di geolocalizzazione è gestita tramite la Permission Policy, uno standard del W3C che permette ai siti web di controllare l'accesso alle funzionalità e alle API del browser attraverso una serie di policies.
Qualora vi sia una policy che blocca tale funzionalità, le callback dei metodi sopra citati, getCurrentPosition() e watchPosition(), ritorneranno un codice di errore, negando l'accesso.

Inoltre, gli sviluppatori devono richiedere informazioni sulla posizione solo quando necessario e utilizzarle solo per lo scopo specifico per cui sono state richieste. Le informazioni sulla posizione devono essere protette da accessi non autorizzati e, se memorizzate, gli utenti devono avere la possibilità di aggiornare o eliminare queste informazioni. La trasmissione delle informazioni sulla posizione deve essere fatta con il consenso esplicito dell'utente.

Questa API, come riportato da W3C e dalla relativa implementazione di MDN Web Docs [23], è supportata dalla maggior parte dei browser attuali. Dalla tabella seguente (figura 3.2) si evince che per molti browser minori il supporto non è ben documentato. Tuttavia, questi browser sono utilizzati molto meno a livello globale, come mostrato dalla seconda immagine (figura 3.3). D'altro canto, l'API è supportata da tutti i principali browser.

Questa documentazione è in continuo aggiornamento, dunque è consigliato agli sviluppatori di consultarla frequentemente, in modo da poter osservare costantemente gli aggiornamenti delle versioni dei browser.

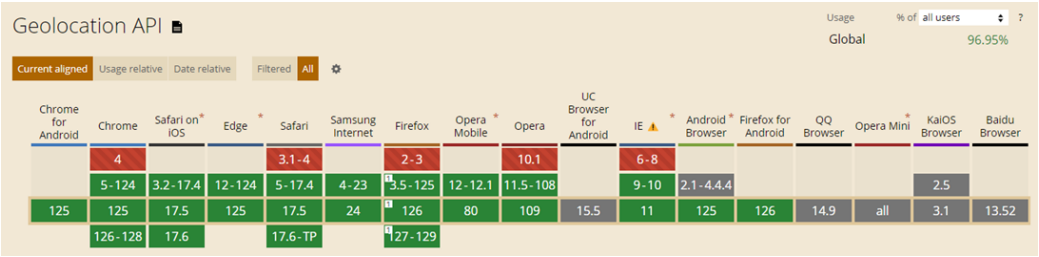

Figura 3.2: *Compatibilità browser con Geolocation API [24]*

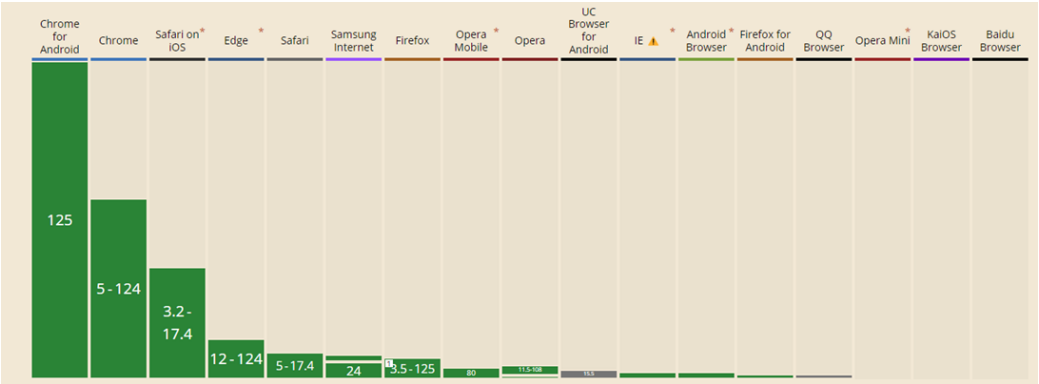

Figura 3.3: *Uso relativo di Geolocation API nei browser*

3.2 Media Capture and Streams API

Questa API [25] stabilisce le modalità per richiedere l'accesso ai dispositivi multimediali locali, come microfoni e videocamere. Essa include l'API MediaStream, che rappresenta un flusso di dati multimediali MediaStreamTrack sincronizzati e raggruppati. Un oggetto *MediaStreamTrack* rappresenta una singola traccia media all'interno di un MediaStream e fornisce controlli per la gestione delle tracce. Inoltre, l'interfaccia *MediaDevices* permette l'accesso ai dispositivi di input multimediale del client, come fotocamere, microfoni e la condivisione dello schermo.

L'API Media Capture and Streams richiede il consenso esplicito dell'utente per accedere a fotocamera e microfono, data la natura sensibile di queste informazioni e il loro potenziale per identificare l'utente. I permessi sono concessi per ogni sessione e possono essere revocati in qualsiasi momento.
Prima dell'acquisizione, un'applicazione può solo verificare la presenza di una fotocamera o di un microfono, ma non il loro numero, poiché i dispositivi potrebbero essere collegati ad altri terminali (come cuffie e autoparlanti).
L'autorizzazione per iniziare l'acquisizione può essere concessa caso per caso (ogni volta che si richiama un metodo di cattura di immagini, video o audio tramite i metodi "captureImage", "captureVideo", "captureAudio") oppure può essere persistente. In entrambi i casi, è essenziale che l'utente possa rifiutare la richiesta o revocare l'autorizzazione in modo semplice e accessibile.
Quando l'utente concede l'accesso, l'user agent deve garantire chiarezza e trasparenza, rendendo evidente che la pagina ha accesso ai dispositivi autorizzati e se un qualsiasi dispositivo sta registrando.
L'API non dovrebbe consentire l'invio automatico di flussi audio o video da dispositivi multimediali autorizzati a un punto finale selezionato da terzi, poiché ciò potrebbe esporre l'utente a potenziali abusi.

La tabella seguente (figura 3.4), basata sulle specifiche del W3C e utilizzata anche da MDN, mostra la compatibilità dei diversi browser con l'API Media Capture and Streams. Come si può notare dalla figura 3.5, la maggior parte dei browser supporta questa API, inclusi quelli più comune-

mente usati, ad eccezione di Opera Mini (Internet Explorer non viene preso in considerazione poiché è ufficialmente terminato nel 2022, rendendolo obsoleto).

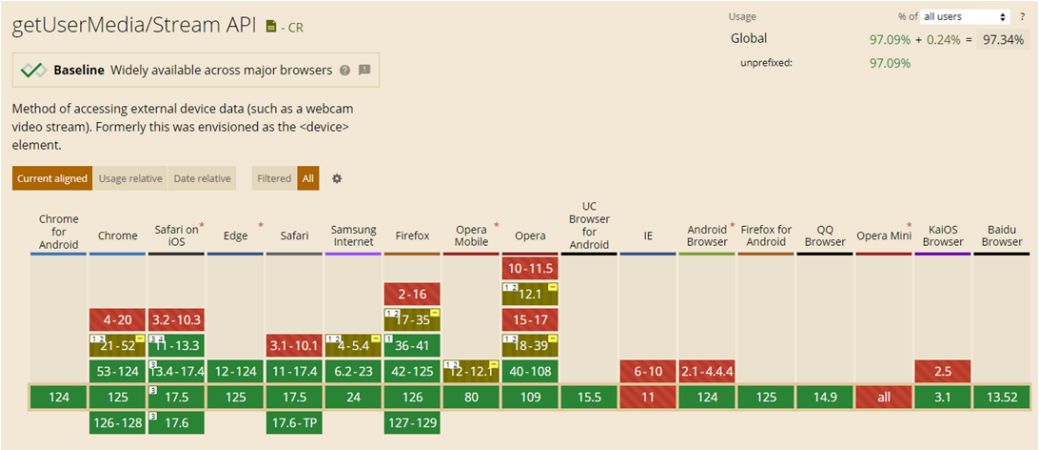

Figura 3.4: *Compatibilità browser con MediaCapture and Streams API [26]*

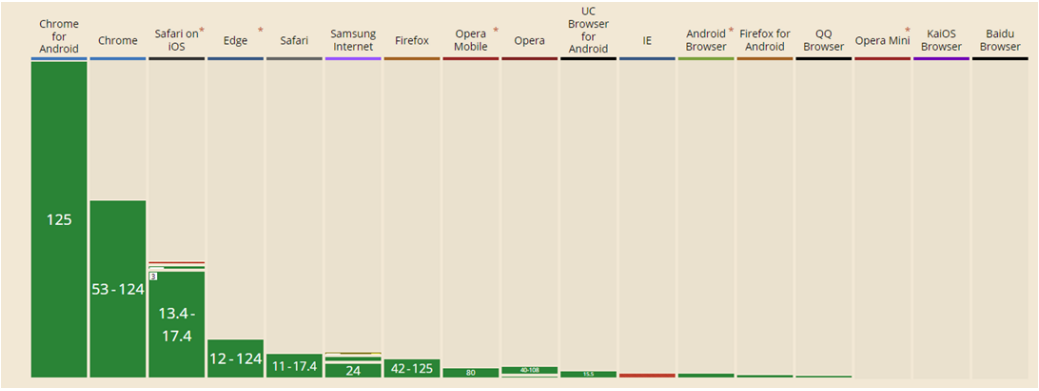

Figura 3.5: *Uso relativo di Media Capture and Streams API nei browser*

3.3 Generic Sensor API

Questa API [27] definisce un framework che espone i dati dei sensori alla piattaforma web in modo coerente, creando una base uniforme per le API dei sensori specifici. L'API Generic Sensor stabilisce una serie di interfacce che estendono l'interfaccia astratta Sensor, permettendo di accedere ai sensori del dispositivo in modo consistente. È importante sottolineare che tale libreria, insieme alle sue estensioni, è attualmente una bozza di lavoro del W3C, in fase di sviluppo ma oltre la fase esplorativa, con l'obiettivo di diventare una raccomandazione ufficiale di questo ente, soddisfacendo standard di qualità, interoperabilità e sicurezza.

L'interfaccia Sensor non espone direttamente un'API di rilevamento dei sensori, ma piuttosto un'API di rilevamento delle funzionalità hardware. Utilizzando questa libreria, le interfacce dei sensori fungono da collegamento (proxy) ai sensori reali presenti sul dispositivo. Pertanto, la presenza dell'API Sensor non garantisce necessariamente l'esistenza di un sensore hardware reale, il suo corretto funzionamento, la sua connessione continua o l'autorizzazione dell'utente per accedervi. Per gestire queste problematiche, l'API Generic Sensor utilizza la programmazione difensiva, rilevando i cambiamenti nello stato del sensore e gestendo gli errori, al fine di ottimizzare l'esperienza utente e di evitare degradazioni causate dall'assenza del sensore stesso.

Poiché le letture dei sensori sono dati sensibili e potrebbero diventare oggetto di attacchi da parte di pagine web dannose (come il rilevamento della posizione senza l'utilizzo del GPS, il monitoraggio della pressione dei tasti, l'identificazione dell'utente, ecc.), l'API Sensor mette a disposizione una serie di strategie per mitigare questi rischi. Tra queste strategie ci sono la lettura dei soli documenti attivi, la riduzione della precisione delle letture per minimizzare il rischio di browser fingerprinting e l'applicazione della Permission Policy su ciascun sensore.

Come per le altre API, l'accesso ai dati dei sensori richiede il permesso esplicito dell'utente. Specifici permessi saranno trattati nelle relative API. Inoltre, le specifiche del W3C richiedono che l'accesso ai dati avvenga solo in

contesti sicuri (HTTPS).

La tabella seguente (figura 3.6) mostra una panoramica della compatibilità dei diversi browser con l'API Generic Sensor. È importante notare che tale libreria rappresenta la base per tutte le APIs dei sensori più specifiche, che verranno analizzate in dettaglio nei prossimi paragrafi. Osservando le colonne che rappresentano i browser principali, si nota che Safari e Firefox, sia nella versione mobile che desktop, non supportano l'API Generic Sensor, in linea con la loro politica di tutela della privacy degli utenti, menzionata ad inizio capitolo.
Numerosi altri browser hanno un supporto indefinito, indicato dalle celle grigie nella tabella, ma sono generalmente meno utilizzati (figura 3.7).

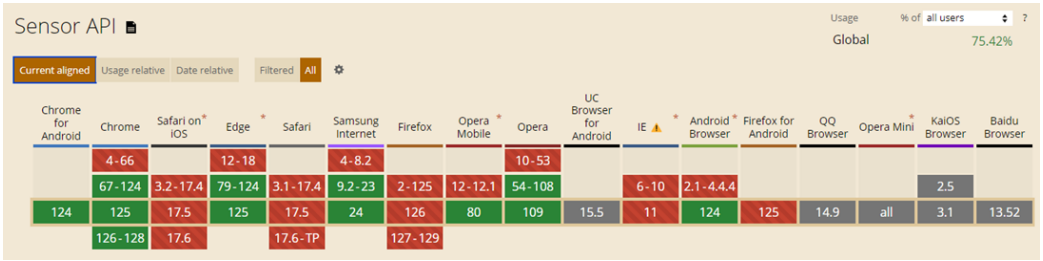

Figura 3.6: *Compatibilità browser con Generic Sensor API [28]*

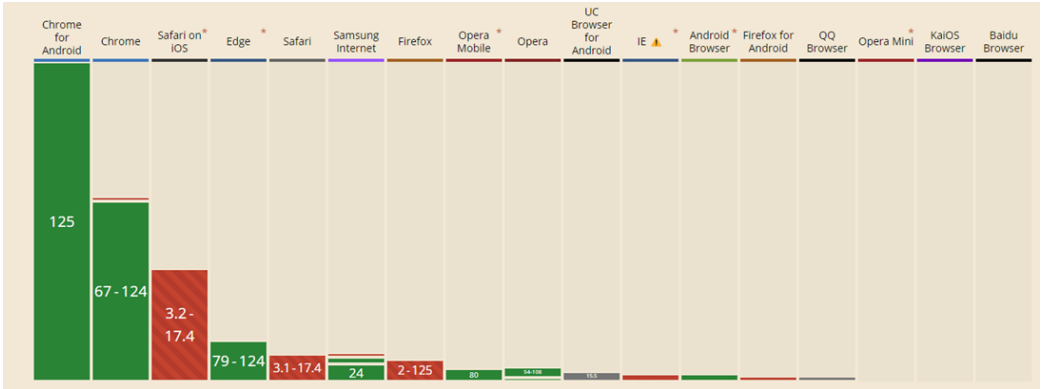

Figura 3.7: *Uso relativo di Generic Sensor API nei browser*

3.3.1 Ambient Light Sensor API

Questa API [29] monitora il livello di luce ambientale o illuminamento nell'ambiente del dispositivo, ed è utilizzata principalmente per regolare la luminosità dello schermo, ma può essere impiegata anche per altre funzionalità responsive.
Per funzionare, richiede i permessi utente per accedere al sensore *ambient-light-sensor*, che è anche il nome dell'autorizzazione associata.

La libreria mette a disposizione una proprietà: *illuminance* che restituisce il livello di luce ambientale rilevata dal dispositivo in lux (unità di misura dell'intensità della luce).

Oltre ai rischi già presentati dalla Generic Sensor API, ci sono rischi specifici legati all'Ambient Light Sensor:

- Profilazione degli utenti: i sensori di luce ambientale possono essere utilizzati per raccogliere informazioni sulle abitudini e sull'ambiente dell'utente, migliorando così le tecniche di profilazione e analisi comportamentale;

- Collegamento tra dispositivi tramite script condivisi: due dispositivi che accedono agli stessi siti web con script di terze parti possono avere i loro livelli di illuminazione correlati nel tempo, facilitando il collegamento tra di essi;

- Comunicazione tra dispositivi tramite segnali luminosi: utilizzando il flash dello schermo o il LED della fotocamera, i dispositivi possono trasmettere messaggi che possono essere letti dai sensori di luce ambientale di dispositivi vicini;

- Fughe di dati attraverso riflessioni luminose: la luce dello schermo riflessa su superfici vicine può essere captata dai sensori, permettendo a siti malevoli di incorporare contenuti da diverse origini e distinguere i contenuti in modo dettagliato;

- Compromissione della cronologia di navigazione: modificando lo stile dei link visitati e non visitati, è possibile distinguere i livelli di luce associati a ciascuno, permettendo di inferire la cronologia di navigazione dell'utente.

Per mitigare queste minacce, oltre alle soluzioni presentate dalla precedente API, gli *UserAgent* dovrebbero ridurre la precisione delle letture del sensore (usando un multiplo di arrotondamento per eccesso pari a 50 lux) e limitare la frequenza di campionamento massima.

Le tabelle seguenti (figure 3.8 e 3.9) mostrano la compatibilità dei diversi browser con l'API del sensore di luce ambientale.

Attualmente, l'unico browser che supporta pienamente questa API sembrerebbe essere Firefox per Android. Tuttavia, è importante notare che questo browser non supporta la Permission Policy necessaria per l'accesso al sensore. Ciò significa che, anche se l'API è supportata, non è possibile accedere effettivamente al sensore di luce ambientale su questo browser, per il fatto che il browser non supporta la politica di accesso all'autorizzazione dello stesso.

Inoltre, il sensore è in fase sperimentale, come indicato dalle piccole flags verdi in alcune celle. Questo suggerisce che la tecnologia è ancora emergente e non completamente sviluppata, attualmente in fase di integrazione nelle piattaforme web di alcuni browser come Chrome, Edge e Opera. Tuttavia, come verrà illustrato più avanti in questo capitolo, Opera non supporta queste funzionalità nella sua versione mobile, rendendo il sensore accessibile solo attraverso i primi due browser appena menzionati.

In sintesi, la compatibilità dell'API del sensore di luce ambientale è limitata e le funzionalità complete non sono ancora disponibili sulla maggior parte dei browser. La fase sperimentale su Chrome, Edge e Opera suggerisce che la tecnologia è ancora in sviluppo e potrebbe diventare più ampiamente disponibile in futuro.

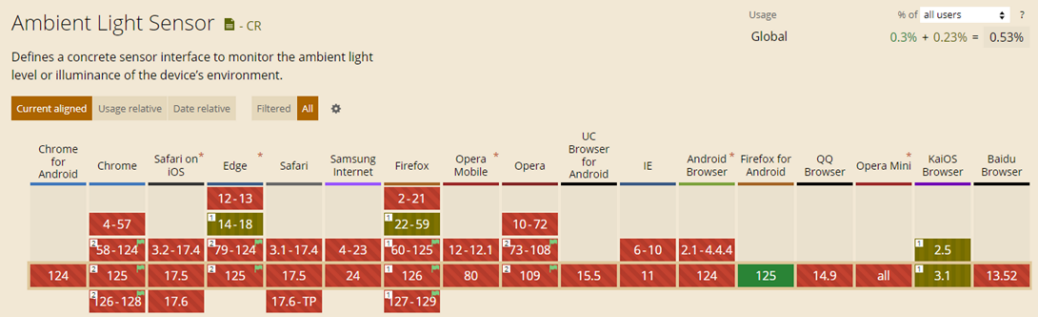

Figura 3.8: *Compatibilità browser con AmbientLightSensor API [30]*

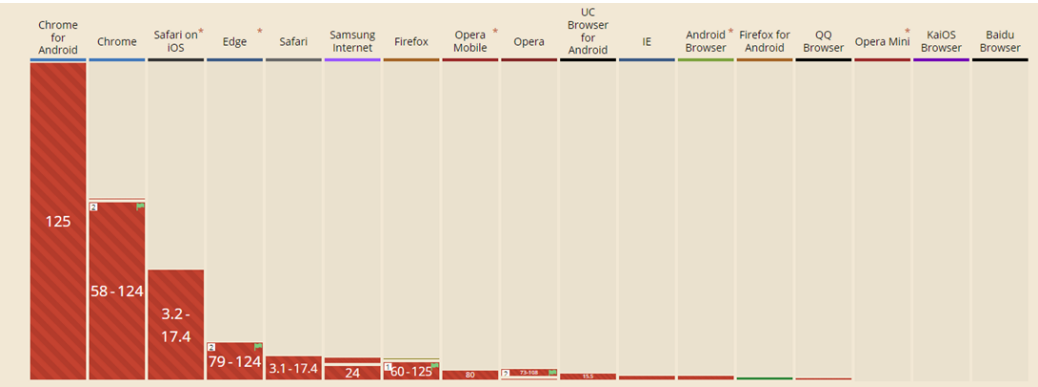

Figura 3.9: *Uso relativo di AmbientLightSensor API nei browser*

3.3.2 Accelerometer API

L'API Accelerometer [31] definisce le interfacce per tre tipi di sensori: Accelerometer, LinearAccelerationSensor e GravitySensor. Questa API permette di ottenere dati sull'accelerazione del dispositivo lungo i tre assi del suo sistema di coordinate locale.

Per utilizzare l'API Accelerometer, è necessario ottenere il permesso dell'utente per accedere all'accelerometro del dispositivo. Una volta concessa l'autorizzazione, si possono utilizzare le proprietà dell'oggetto accelerometer: *accelerometer.x, accelerometer.y e accelerometer.z*, che forniscono i valori dell'accelerazione lungo gli assi x, y e z, espressi in m/s^2.

Il sensore LinearAccelerationSensor fornisce informazioni sull'accelerazione del dispositivo lungo i tre assi, escludendo l'influenza della gravità. Questo sensore è particolarmente utile per rilevare i movimenti del dispositivo nello spazio, come oscillazioni e vibrazioni. Il sensore GravitySensor, invece, misura la gravità che agisce sul dispositivo lungo i tre assi, ed è utile per determinare l'orientamento del dispositivo, ad esempio se è in posizione verticale o orizzontale.

Le letture provenienti dai sensori inerziali, come l'accelerometro, potrebbero essere sfruttate per minare la sicurezza, facilitando attività come keylogging, tracciamento della posizione, fingerprinting e identificazione dell'utente. Alcuni studi del W3C suggeriscono che limitare la frequenza e la precisione delle letture potrebbe diminuire tali rischi, sebbene ciò possa penalizzare le applicazioni web che utilizzano legittimamente i sensori. Per mitigare questi rischi, oltre alle soluzioni proposte dall'API Generic Sensors, alcuni ricercatori propongono di fornire un'indicazione visiva all'utente quando i sensori sono in uso e di richiedere il consenso esplicito dell'utente prima di accedere ai dati dei sensori. Questa API necessita della Permission Policy associata all'omonimo sensore.

La tabella seguente (figura 3.10) fornisce una panoramica della compatibilità tra vari browser e l'API Accelerometer. È importante notare che la compatibilità delle versioni dei browser riportata riguarda le interfacce che gestiscono l'accelerazione (Accelerometer), l'accelerazione lineare (LinearAccelerationSensor) e l'accelerazione gravitazionale (GravitySensor).

In particolare, le proprietà x, y e z dell'accelerometro e la Permission Policy di questo sensore sono supportate dal browser Samsung Internet dalla versione 9.2 alla 24. Tuttavia, il supporto per queste proprietà e la Permission Policy è indefinito per altri browser come UC Browser for Android, QQ, Opera Mini, KaiOS e Baidu, il che significa che l'utilizzo di queste funzionalità su tali browser potrebbe non essere possibile o potrebbe dare risultati imprevedibili. Tuttavia, l'uso di questi browser è minimo rispetto ai principali (figura 3.11).

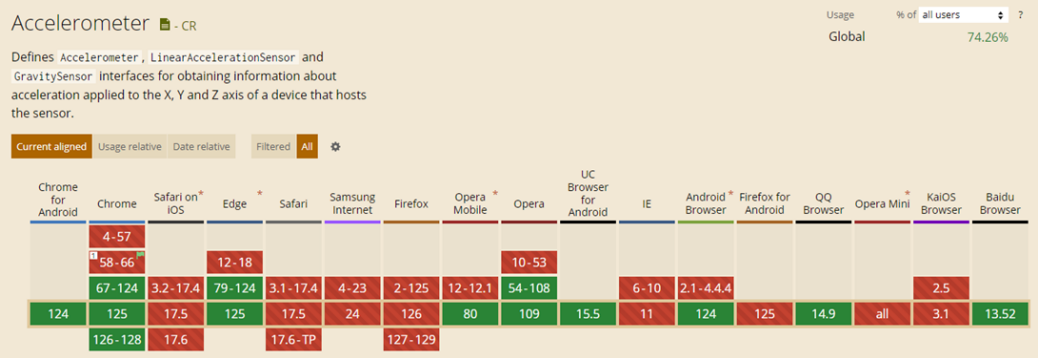

Figura 3.10: *Compatibilità browser con Accelerometer API [32]*

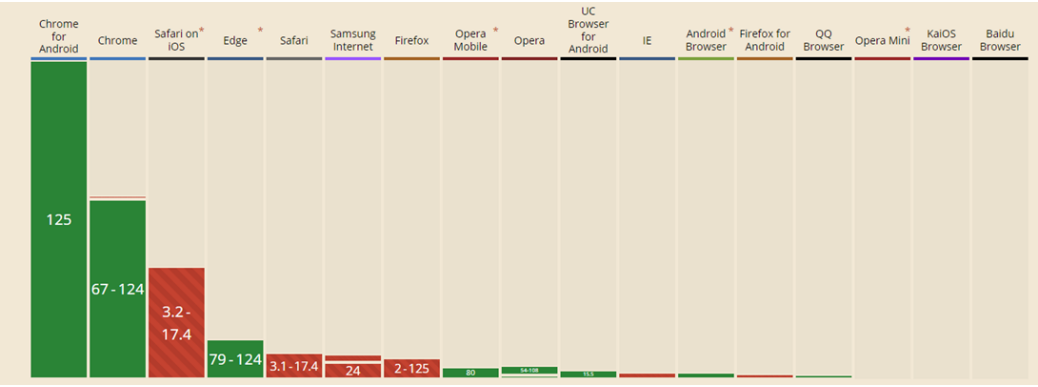

Figura 3.11: *Uso relativo di Accelerometer API nei browser*

3.3.3 Gyroscope API

L'API Gyroscope [33] consente di monitorare la velocità angolare attorno ai tre assi principali del dispositivo, espressa in radianti al secondo (rad/s).

Esistono tre proprietà: *gyroscope.x, gyroscope.y e gyroscope.z*, che rappresentano rispettivamente la velocità angolare attorno ai tre assi. La direzione della rotazione, indicata dal segno della velocità angolare, segue la convenzione della mano destra nel sistema di coordinate locali del dispositivo.

Come l'accelerometro, anche il giroscopio è un sensore inerziale e le sue letture da parte di software malevoli possono essere sfruttate per minacciare

la sicurezza degli utenti, per mezzo delle stesse tecniche presentate per l'API precedente. Per mitigare questi rischi, il W3C prevede le stesse raccomandazioni. Anche in questo caso, i ricercatori suggeriscono agli sviluppatori di notificare l'utente con un'indicazione visiva al momento della lettura e richiedere un consenso esplicito. Anche per questa libreria, è necessaria la Permission Policy associata al sensore corrispondente.

Come per l'API Accelerometer, anche per l'API Gyroscope il supporto da parte di molti browser è sconosciuto (figura 3.12). Tuttavia, questi browser sono utilizzati da una piccola minoranza di utenti a livello globale, come illustrato nel grafico 3.13 che mostra l'utilizzo dell'API suddiviso per browser.

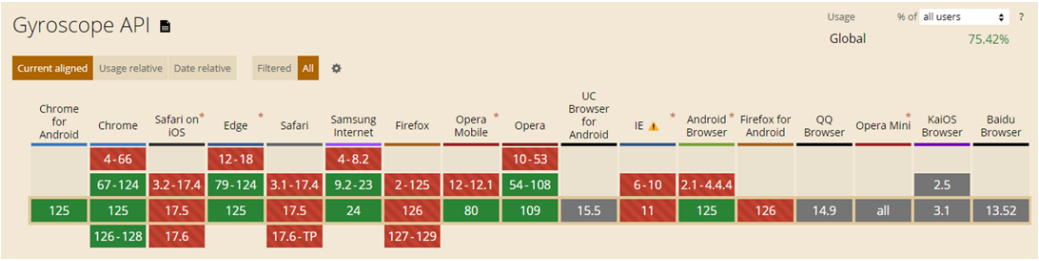

Figura 3.12: *Compatibilità browser con Gyroscope API [34]*

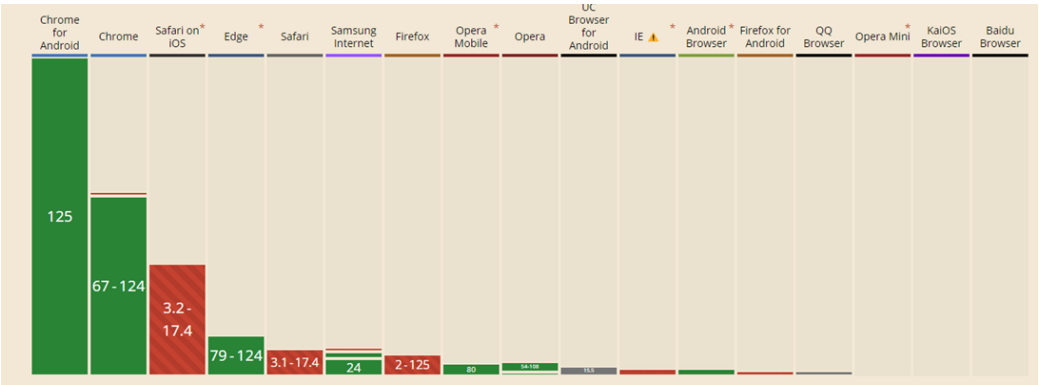

Figura 3.13: *Uso relativo di Gyroscope API nei browser*

3.3.4 Magnetometer API

Questa API [35] definisce due interfacce per rilevare il campo magnetico attorno al dispositivo lungo i tre assi principali (x, y e z), misurato in micro Tesla (μT). Le interfacce disponibili sono:

- Magnetometer: riporta valori del campo magnetico corretti in modo tale da eliminare le distorsioni causate da oggetti ferromagnetici (hard iron distortion) e da materiali che influenzano il campo magnetico in modo non uniforme (soft iron distortion);
- UncalibratedMagnetometer: fornisce valori di campo magnetico grezzi, senza correzioni per distorsioni da oggetti ferromagnetici, ma con correzioni per distorsioni da materiali non uniformi.

Questa API mette a disposizione tre proprietà: *magnetometer.x, magnetometer.y e magnetometer.z*, che contengono il valore del campo magnetico lungo i rispettivi assi. È importante considerare che l'intensità e la direzione del campo magnetico possono essere influenzate dalla latitudine e da dispositivi elettronici o magneti permanenti vicini al dispositivo.

I dati raccolti da questo sensore possono comportare seri rischi per la privacy dell'utente. Come l'API di geolocalizzazione, il magnetometro può essere utilizzato per esporre e verificare la posizione di un utente. Ad esempio, un attacco potrebbe mappare la posizione in base alle alterazioni magnetiche costanti causate da un edificio. Inoltre, se l'utente utilizza una VPN, il campo magnetico associato alle informazioni geo-IP potrebbe essere confrontato con le letture del magnetometro per rilevare la posizione reale dell'utente, rivelando così l'uso della VPN.

Le letture del magnetometro possono essere influenzate da oggetti magnetizzati vicini, come gioielli, che potrebbero essere utilizzate per monitorare la pressione dei tasti. Gli sviluppatori devono essere consapevoli del rischio di perdita di informazioni attraverso canali secondari legati alla forza del campo magnetico e ad altri fattori come l'uso della CPU, che potrebbero rivelare informazioni sulle applicazioni o i siti web visitati.

I browser dovrebbero adottare strategie di mitigazione, come limitare la frequenza e la precisione di campionamento, oltre alle soluzioni proposte dalla Generic Sensor API, per ridurre l'efficacia di questi attacchi.

Le tabelle 3.14 e 3.15 mostrano la compatibilità dei diversi browser con l'API Magnetometer. L'API è teoricamente indisponibile poichè i rischi per la privacy hanno portato i browser a limitarne l'applicabilità. Tuttavia, come per l'Ambient Light Sensor API, il magnetometro è effettivamente utilizzabile solo in un numero molto limitato di browser sottoforma di versione sperimentale, tra cui Chrome, Edge ed Opera. In sintesi, anche se il sensore è disponibile, le preoccupazioni per la privacy ne limitano fortemente l'uso nello sviluppo di applicazioni web.

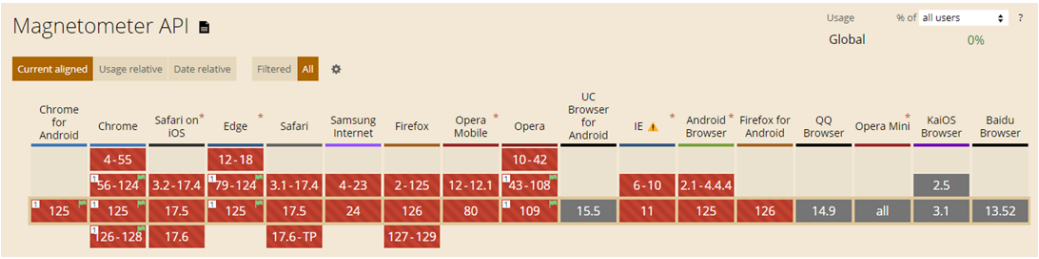

Figura 3.14: *Compatibilità browser con Magnetometer API [36]*

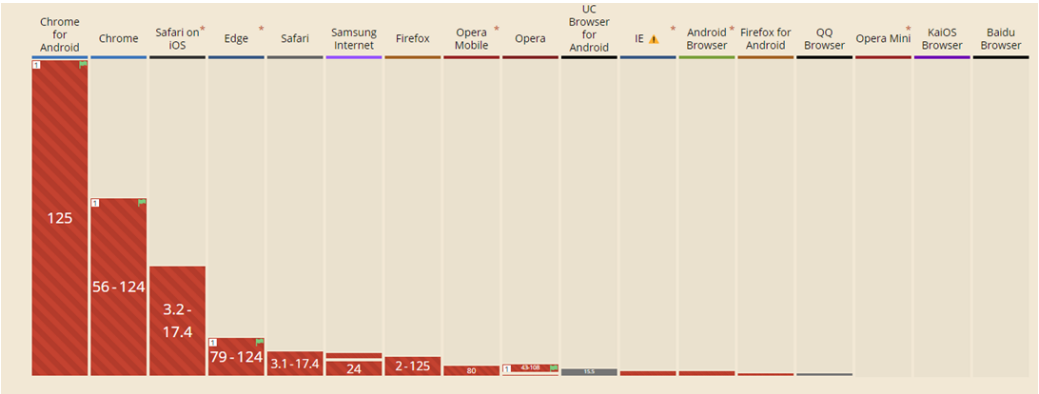

Figura 3.15: *Uso relativo di Magnetometer API nei browser*

3.3.5 Orientation Sensor API

Questa specifica [37] definisce un'interfaccia di base per monitorare l'orientamento fisico del dispositivo in relazione ad un sistema di coordinate cartesiane tridimensionale. La libreria Orientation Sensor non può essere utilizzata direttamente, ma fornisce proprietà e metodi accessibili attraverso le interfacce dei sensori che la estendono:

- AbsoluteOrientationSensor: descrive l'orientamento fisico del dispositivo in relazione al sistema di coordinate di riferimento della Terra e fornisce un sensore virtuale chiamato *absolute-orientation*;
- RelativeOrientationSensor: descrive l'orientamento fisico del dispositivo indipendentemente dal sistema di coordinate di riferimento della Terra e fornisce un sensore virtuale chiamato *relative-orientation*.

La proprietà principale è *quaternion* che restituisce un array di quattro elementi, ognuno dei quali rappresenta una componente del quaternione unitario che descrive l'orientamento del dispositivo, ovvero le rotazioni nello spazio tridimensionale.

I dati forniti dall'Orientation Sensor derivano dalla combinazione delle letture di accelerometro, giroscopio e magnetometro, che forniscono informazioni sul movimento e sulla posizione del dispositivo, elaborate insieme per determinarne l'orientamento rispetto a un sistema di coordinate.

La libreria Orientation Sensor non fornisce considerazioni specifiche sulla sicurezza e sulla privacy oltre a quelle descritte nella Generic Sensor API. Inoltre, la lettura dei dati dall'interfaccia RelativeOrientationSensor richiede i permessi per i sensori *accelerometer* e *gyroscope*, mentre AbsoluteOrientationSensor richiede anche il permesso del sensore *magnetometer*.

L'interfaccia Orientation Sensor è supportata dalla maggior parte dei browser più popolari, ad eccezione di Safari e Firefox, così come di Samsung Internet (figura 3.16). Tuttavia, le API delle sottoclassi che estendono l'interfaccia Orientation Sensor sono supportate da Samsung Internet, mentre il supporto su browser meno utilizzati è incerto (figure 3.17, 3.18, 3.19, 3.20).

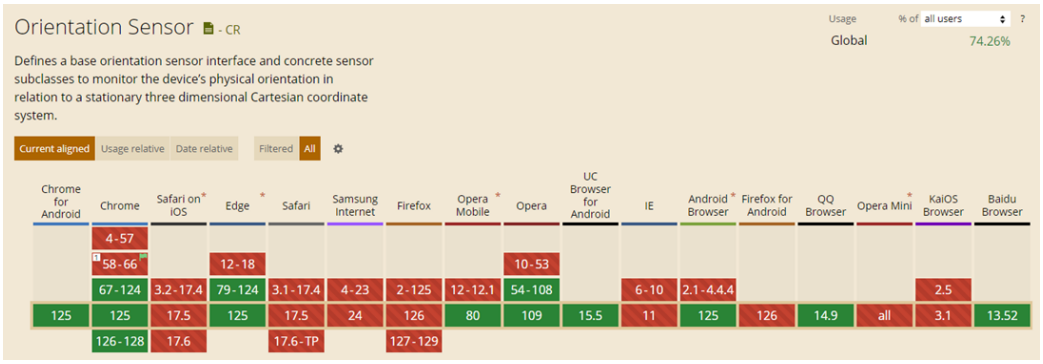

Figura 3.16: *Compatibilità browser con Orientation Sensor [38]*

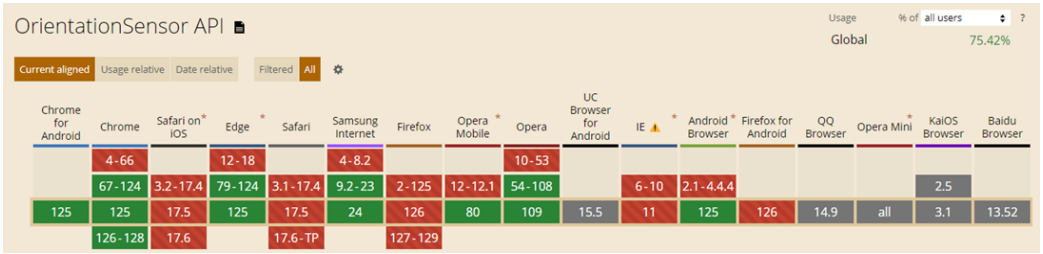

Figura 3.17: *Compatibilità browser con Orientation Sensor API*

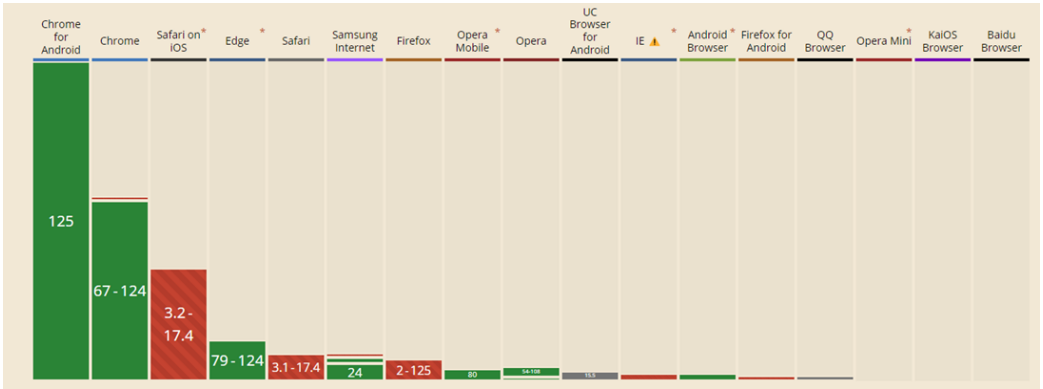

Figura 3.18: *Uso relativo di Orientation Sensor API nei browser*

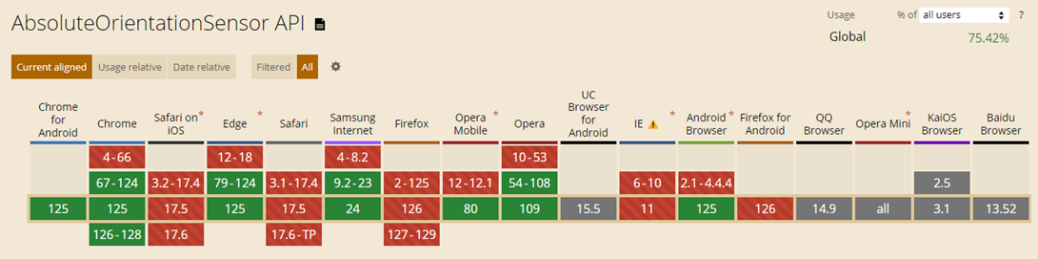

Figura 3.19: *Compatibilità browser con Absolute Orientation Sensor API*

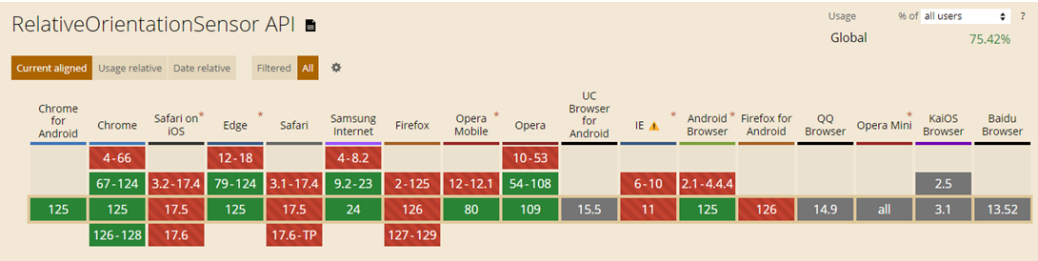

Figura 3.20: *Compatibilità browser con Relative Orientation Sensor API*

3.4 Compatibilità dei principali browser

La tabella rappresentata in figura 3.22 riassume le compatibilità tra i browser più diffusi e le APIs per l'accesso ai sensori dei dispositivi esaminate in precedenza.

Le informazioni riportate si basano sui dati dichiarati dal repository MDN Web Docs [16].

La tabella è organizzata in modo da presentare nelle colonne le diverse APIs, mentre nelle righe sono elencati i browser, suddivisi per sistemi operativi Android e iOS. Ogni browser include una riga che indica la compatibilità dichiarata dal W3C, con le versioni minime di rilascio del browser, separate per versione mobile (M) e desktop (D).

L'analisi condotta verifica se la compatibilità dichiarata corrisponde a quella effettivamente testata, che verrà trattata nel paragrafo successivo.

Le celle della tabella utilizzano diverse icone e colori per rappresentare i risultati:

- Icona verde: indica che la compatibilità testata corrisponde a quella dichiarata quando era presente;
- Croce rossa: segnala che sia la compatibilità dichiarata che quella testata sono assenti;
- Sfondo rosso: indica che la compatibilità dichiarata era presente, ma quella testata è assente;
- Sfondo giallo: segnala che la compatibilità dichiarata era assente, ma quella testata è presente.

Inoltre, la prima riga della tabella include un'icona per ogni API che indica se il sensore è sperimentale o stabile come riportato nella legenda (figura 3.21).

La tabella fornisce, dunque, un quadro dettagliato delle compatibilità reali e testate su Sensorworker, rispetto a quelle dichiarate per ciascun browser e sistema operativo, evidenziando eventuali discrepanze tra i dati dichiarati e quelli testati.

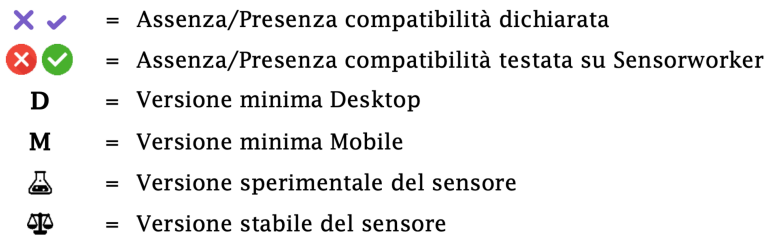

Figura 3.21: *Legenda tabella riassuntiva compatibilità*

	Browser	Geo Location	Media Capture and Streams	Generic Sensor API	AmbientLight Magnetometer	Accelerometer	GravitySensor	Gyroscope LinearAcceleration	Orientation AbsoluteOrientation RelativeOrientation
		⚖	⚖	⚖	🧪	🧪	⚖	⚖	⚖
CHROME	**Chrome**	✓ D: 5 M: 18	✓ D: 55 M: 55	✓ D: 67 M: 67	✗ D: 56 M: 56	✓ D: 67 M: 67	✓ D: 91 M: 91	✓ D: 67 M: 67	✓ D: 67 M: 67
	Chrome Android	✓	✓	✓	✓	✓	✓	✓	✓
	Chrome IOS	✓	✓	✗	✗	✗	✗	✗	✗
SAFARI	**Safari**	✓ D: 5 M: 3	✓ D: 11 M: 11	✗	✗	✗	✗	✗	✗
	Safari IOS	✓	✓	✗	✗	✗	✗	✗	✗
MOZILLA FIREFOX	**Firefox**	✓ D: 3.5 M: 4	✓ D: 15 M: 15	✗	✗	✗	✗	✗	✗
	Firefox Android	✓	✓	✗	✗	✗	✗	✗	✗
	Firefox IOS	✓	✓	✗	✗	✗	✗	✗	✗
SAMSUNG IT	**Samsung Internet**	✓ M: 1	✓ M: 6	✓ M: 9	✗	✓ M: 9	✓ M: 16	✓ M: 9	✓ M: 9
	Samsung Internet Android	✓	✓	✓	✗	✓	✓	✓	✓
OPERA	**Opera Mobile**	✓ D: 10.6 M: 11	✓ D: 42 M: 42	✓ D: 54 M: 48	✗ D: 43 M: no	✓ D: 54 M: 48	✓ D: 77 M: 64	✓ D: 54 M: 48	✓ D: 54 M: 48
	Opera Mobile Android	✓	✓	✓	✗	✓	✓	✓	✓
	Opera Mobile IOS	✓	✓	✗	✗	✗	✗	✗	✗
EDGE	**Edge**	✓ D: 12	✓ D: 12	✓ D: 79	✗ D: 79	✓ D: 79	✓ D: 91	✓ D: 79	✓ D: 79
	Edge Android	✓	✓	✓	✓	✓	✓	✓	✓
	Edge IOS	✓	✓	✗	✗	✗	✗	✗	✗

Figura 3.22: *Tabella riassuntiva compatibilità browser e API*

3.5 Testing

Il lavoro di testing è stato incentrato sul tentativo di effettuare le rilevazioni con i sensori del dispositivo su browser differenti utilizzando le campagne di Sensorworker.
L'obiettivo era adattare i suggerimenti e i warning dei browser consigliati in relazione ai sensori richiesti dalle campagne, in modo da massimizzare la user experience dell'applicazione ed evitare fallimenti del sistema.

Per eseguire il lavoro di testing è stato necessario creare un account su Microworkers. Questo ha permesso di ottenere l'API key dell'account Microworkers, utilizzabile nella fase di registrazione di un utente con il ruolo di crowdsourcer su Sensorworker, consentendo così la creazione delle campagne (dettagli nel capitolo 2.2).

Su Sensorworker sono state create tre campagne che accedono a sensori di differenti tipologie:

1. Campagna 1: utilizzo della fotocamera, test sulla compatibilità con Media Capture and Streams API;

2. Campagna 2: utilizzo della videocamera e microfono, test sulla compatibilità con Media Capture and Streams API;

3. Campagna 3: utilizzo dei diversi sensori, test sulla compatibilità con Generic Sensor API.

Di seguito vengono riportati i test effettuati sulle campagne descritte, con i risultati della compatibilità dei browser utilizzati con le APIs dei sensori.

3.5.1 Campagne 1 e 2: Test sensori multimediali

La prima campagna richiedeva al worker di effettuare una foto a un albero e dichiararne la specie nel campo di testo (figura 2.1).

Nella seconda campagna, gli utenti dovevano realizzare un video e una registrazione audio di un paesaggio, specificandone il tipo.

Entrambe le campagne avevano l'obiettivo di testare la compatibilità dei browser con la Media Capture and Streams API, progettata per consentire alle applicazioni web di accedere ai dispositivi di input multimediale [39], e con la GeoLocation API, utilizzata per determinare il punto di rilevamento.

Le rilevazioni effettuate tramite fotocamera, microfono e GPS sono risultate efficaci sui principali browser e per entrambi i sistemi operativi, Android e iOS.

3.5.2 Campagna 3: Test sensori generici

Infine, la terza campagna mirava a testare la compatibilità con la Generic Sensor API, e, dunque, con le APIs dei sensori che la estendono.

Agli utenti è stato richiesto di effettuare rilevazioni multiple con i basic sensors (accelerometro, sensore di accelerazione gravitazionale, giroscopio, sensore di accelerazione lineare, sensore di orientamento assoluto e relativo) e sperimentali (magnetometro e sensore di luce ambientale) con un buffer temporale (1-3 secondi) o di distanza (1-3 metri) tra una misurazione e la successiva (figura 2.2).

Dai risultati del test è emerso che, in generale, la compatibilità dichiarata dal W3C rispecchia quella effettiva. Tuttavia, sono state riscontrate delle eccezioni significative sui dispositivi iOS, che verranno analizzate nel dettaglio nel capitolo finale: *Conclusioni*.

Capitolo 4

Nuove Feature di Sensorworker

In questo capitolo verranno esaminate le modifiche apportate al codice di Sensorworker per implementare lo studio sulla compatibilità tra browser e le APIs javascript per l'accesso ai sensori, discussa nel capitolo precedente.

Le nuove funzionalità introdotte hanno come obiettivo principale il miglioramento dell'esperienza utente nell'utilizzo dell'applicazione, fornendo messaggi di errore, avvertimenti e suggerimenti per guidare l'utente ad un utilizzo corretto di Sensorworker e prevenire eventuali malfunzionamenti del sistema.

Un obiettivo significativo raggiunto è stato quello di estendere l'utilizzo di una particolare funzionalità dell'applicazione anche agli utenti dotati di dispositivi iOS, precedentemente limitati.

Prima di analizzare il codice che gestisce il rendering della pagina, verranno presentate le tre principali funzioni che saranno richiamate frequentemente. Va sottolineato che tali metodi necessitano di una manutenzione costante in modo da verificare le associazioni di compatibilità tra i diversi browser e le APIs precedentemente analizzate, al variare delle specifiche del W3C. Gli sviluppatori che si occuperanno della manutenzione di Sensorworker possono verificare il supporto direttamente dal repository Mozilla, MDN Web Docs [17], o consultare il sito suggerito dal W3C: `https://caniuse.com`.

Il listato 4.1 presenta la funzione che permette di individuare, a partire dalla proprietà navigator.userAgent dell'oggetto window, il browser corrente

in uso da parte dell'utente.

```
export const getCurrentBrowserName = (userAgent) => {
  if (userAgent.includes("Firefox")) {
    return "Mozilla Firefox";
  } else if (userAgent.includes("SamsungBrowser")) {
    return "Samsung Internet";
  } else if (userAgent.includes("Opera") || userAgent.includes("OPR")) {
    return "Opera";
  } else if (userAgent.includes("Edge") || userAgent.includes("Edg")) {
    return "Microsoft Edge";
  } else if (userAgent.includes("Chrome")) {
    return "Google Chrome";
  } else if (userAgent.includes("Safari")) {
    return "Apple Safari";
  } else
    return "Unknown";
};
```

Listing 4.1: *Get current browser name*

Il listato 4.2 mostra la funzione utilizzata quando un utente avvia una campagna che richiede l'utilizzo di sensori non supportati dal suo browser attuale, ignorando i warning preventivi. La funzione restituisce quindi un elenco di browser alternativi che l'utente può utilizzare per completare la campagna.

```
export const checkBrowserSensorSupport = (sensorName) => {
  const currentBrowser = getCurrentBrowserName(navigator.userAgent);

  if (sensorBrowserAvailability.hasOwnProperty(sensorName)) {
    let supportedBrowsers = sensorBrowserAvailability[sensorName];

    if (supportedBrowsers.includes(currentBrowser)) {
      supportedBrowsers = supportedBrowsers.replace(currentBrowser + ",",
          "");
      if (supportedBrowsers.slice(-1) === ",")
        supportedBrowsers = supportedBrowsers.slice(0, -1);
    }

    return supportedBrowsers;
  } else
    return ["Not supported by any browser"];
};
```

Listing 4.2: *Check browser support*

Infine, la funzione 4.3 accetta in input l'array di sensori richiesti dalla campagna selezionata dal worker e filtra i browser che soddisfano le compatibilità. In questo modo vengono suggeriti all'utente solo quelli mediante i quali la campagna può essere completata senza incorrere in limitazioni.

```
export const checkBrowserSensorSupport2 = (sensors) => {
  let allBrowsers = browserNamesArray;
  for (let sensor of sensors) {
    if (
      sensor === "Accelerometer" ||
      sensor === "GravitySensor" ||
      sensor === "Gyroscope" ||
      sensor === "LinearAccelerationSensor" ||
      sensor === "AbsoluteOrientationSensor"||
      sensor === "RelativeOrientationSensor"
    ) {
      allBrowsers = allBrowsers.filter(
        (browser) =>
          browser !== "Mozilla Firefox" &&
          browser !== "Apple Safari"
      );
    } else if (sensor === "AmbientLightSensor" || sensor === "Magnetometer")
        {
      allBrowsers = allBrowsers.filter(
        (browser) =>
          browser !== "Mozilla Firefox" &&
          browser !== "Opera" &&
          browser !== "Samsung Internet" &&
          browser !== "Apple Safari"
      );
    }
  }
  return allBrowsers.join(", ");
};
```

Listing 4.3: *Check sensors support*

4.1 Multiselect browsers

Passando ora ai listati che gestiscono il rendering delle pagine, una delle feature più significative aggiunte è sicuramente il selettore multiplo presente nella pagina *CampaignSelect*. Questa pagina consente al worker di scegliere

la campagna Sensorworker a cui partecipare. Il selettore multiplo è composto da coppie di checkbox e browser (figura 4.1), in modo da consentire all'utente di filtrare le campagne in base ai browser che desidera utilizzare per svolgerle. Il filtro utilizza la funzione 4.3 per restituire solo le campagne i cui sensori sono supportati dai browser selezionati.

Se il client accede alla pagina da un dispositivo mobile, verrà selezionata automaticamente solo la checkbox relativa al browser corrente. Se invece il client accede alla pagina da un dispositivo desktop, nessun browser verrà selezionato di default, ma saranno visibili tutte le campagne disponibili.

```
<FormControl sx={{ m: 1, width: "50vw" }}>
    <Select
        labelId="multiple-browsers-checkbox-label"
        id="multiple-browsers-checkbox"
        multiple
        displayEmpty
        value={selectedBrowsers}
        onChange={handleChange}
        renderValue={(selected) => {
            if (selected.length === 0) {
                return "Please select the browser...";
              }
              return selected.join(", ");
        }}
    >
        {browserNamesArray.map((browser) => (
            <MenuItem key={browser} value={browser}>
                <Checkbox
                    checked={selectedBrowsers.indexOf(browser) > -1}
                />
                {browser}
            </MenuItem>
        ))}
    </Select>
</FormControl>
```

Listing 4.4: *Multiselect browser*

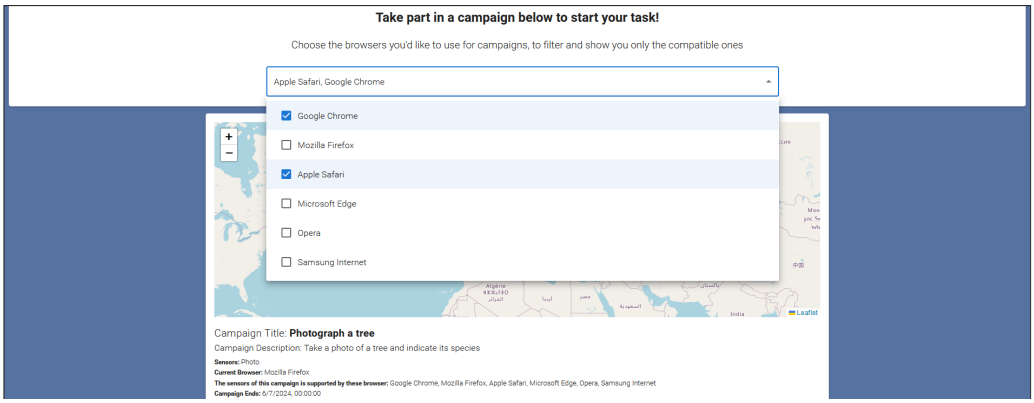

Figura 4.1: *Multiselect dei browser nella pagina di selezione delle campagne*

4.2 Messaggi di warning

Warning popup

Un'altra piccola feature aggiunta consiste in un popup di avviso che viene visualizzato nel caso in cui il worker voglia partecipare a una campagna i cui sensori non sono supportati dal browser attuale. In questo caso, il popup invita il worker a rinunciare alla campagna ed a proporgli i browser supportati.

Se l'accesso alla pagina avviene tramite un dispositivo mobile, il popup di avviso viene mostrato solo se il browser corrente non supporta i sensori richiesti dalla campagna selezionata. Se invece l'accesso avviene da un dispositivo desktop, viene sempre visualizzato un popup di sicurezza. Ciò è dovuto al fatto che il lavoratore dovrà utilizzare il proprio smartphone per inquadrare un QR code nella fase successiva, che lo reindirizzerà alla pagina di raccolta dati. Pertanto, potrebbe essere indirizzato a un browser diverso da quello utilizzato sul proprio computer.

Warning mancata rilevazione del sensore

Nella pagina *SensorCollect* (figure 2.1 e 2.2) vengono mostrati i sensori richiesti dalla campagna su cui effettuare le letture, tramite appositi pulsanti. Al caricamento della pagina, viene verificata l'effettiva esistenza del sensore e, in caso di esito positivo, viene controllata l'esistenza dell'API responsabile dell'accesso a tale sensore tramite un approccio di *programmazione difensiva*. Se uno di questi controlli non viene superato, viene visualizzato un messaggio di errore che indica i browser compatibili con il sensore. In questo modo, il pulsante tramite cui è possibile effettuare le letture viene disabilitato, in modo da evitare eventuali errori.

4.3 QrCode di verifica delle compatibilità

Sensorworker consente all'utente di effettuare un test preventivo della compatibilità con i diversi sensori, browser e sistema operativo. Una volta selezionata la campagna di Microworkers, all'utente viene mostrato un QR code nella pagina *CampaignSelect* che lo reindirizza a una pagina di controlli e verifiche.

Nella versione precedente del software, il test era disponibile solo per i dispositivi Android che utilizzavano Google Chrome come browser. Tuttavia, a seguito dello studio presentato in questo testo, la funzione è stata estesa anche ai dispositivi iOS ed a qualsiasi browser.

Inoltre, è stato aggiunto un ulteriore passaggio di verifica sul browser utilizzato in relazione alla sezione che mostra la compatibilità o meno con i sensori sperimentali (magnetometro e sensore di luce ambientale). Se l'utente clicca sul link di aiuto per abilitarli, qualora fossero disponibili per il browser, vengono fornite delle istruzioni su come abilitarli in base al browser utilizzato.

4.4 Altre funzioni

Oltre alle nuove funzionalità appena discusse, l'ultimo rilascio di Sensorworker include diversi altri piccoli interventi di manutenzione volti a migliorare l'esperienza utente e la stabilità del sistema.
In particolare, è stata effettuata una pulizia approfondita del codice per renderlo più leggibile e user-friendly per gli sviluppatori. Inoltre, è stato corretto un piccolo problema CORS (Cross-Origin Resource Sharing) lato sviluppo. Infine, è stata risolta una importante problematica che causava il malfunzionamento del sistema in determinate condizioni, di seguito descritta.

Question bug

La prima versione di Sensorworker consentiva ai crowdsourcer di creare campagne che includevano, oltre all'elenco dei sensori richiesti, un campo di testo facoltativo, correlato a una domanda o ad una richiesta posta dal crowdsourcer (come discusso nella sezione 2.2 del capitolo 2), in modo da raccogliere dei messaggi di testo dai workers, come feedback, opinioni o descrizione dei contesti in cui sono state eseguite le rilevazioni sensoriali.

Tuttavia, se il crowdsoucer non includeva il campo facoltativo al momento della creazione della campagna, il job (ovvero il compito di crowdsensing assegnato al worker) falliva al raggiungimento della scadenza prevista per il completamento della campagna.
Ciò accadeva perché, anche se il worker eseguiva le letture dei sensori richieste in modo corretto, il programma rimaneva in attesa della compilazione del campo relativo alla domanda, che non essendo presente, non poteva essere compilato. Di conseguenza, il job era inevitabilmente destinato a fallire, rimanendo dunque in stato active e senza passare in stato ready-to-rate, fino alla scadenza del tempo previsto per il completamento della campagna (ulteriori dettagli tecnici sui passaggi di stato del job di un worker possono essere consultati qui [13]).

Per risolvere questo problema, è stato implementato un controllo a livello di backend per verificare la presenza del campo relativo alla campagna, in modo da consentire il completamento del job senza inconvenienti anche in caso di assenza del campo.

```
static async checkJobCompleted(
    req: Request,
    res: Response,
    next: NextFunction
  ) {
    try {
      const jobId = req.body.jobId ? req.body.jobId : req.query.jobId;
      const data = await SensorData.find({ jobId: jobId }).exec();
      const jobDoc = await Job.findOne({ _id: jobId }).exec();
      const campaign = await Campaign.findOne({ _id: jobDoc?.campaignID })
        .populate("sensors")
        .exec();

      const result = campaign?.sensors.every((sensor) => {
        const sensorData = data.find((d) => d.name == sensor.name);
        if (sensorData)
          return sensorData.data.length == sensor.maxReadings;
        return false;
      });

      if (campaign && jobDoc && campaign.question == null) {
        jobDoc.answer = "Default Answer";
        await jobDoc.save();
      }

      if (result && jobDoc && jobDoc.answer != null) {
        jobDoc.active = false;
        jobDoc.readyToRate = true;
        await jobDoc.save();
        await SensorDataController.submitProof(req, res, next);
      } else
        super.ok(res, {});
    } catch (err: any)
      super.fail(res, err);
  }
```

Listing 4.5: *Check job completed*

Conclusioni

Nel panorama attuale, l'accesso ai sensori dei dispositivi dalle applicazioni web è diventato sempre più facile e compatibile con i diversi browser.

Molti sensori hanno raggiunto una stabilità tale da permettere un accesso sicuro ed efficiente al relativo hardware, mentre altri, come il magnetometro, il sensore di luce ambientale e l'accelerometro, sono ancora in fase sperimentale. Tuttavia, ciò non preclude la possibilità di accedervi.

Come riportato nella tabella 3.22, due API hanno una compatibilità completa con tutti i browser, indipendentemente dal sistema operativo del dispositivo: Geolocation e Media Capture and Streams. Molti browser hanno infatti deciso di adottare ed implementare queste librerie sin dalle loro prime versioni. Tra gli esempi più rilevanti troviamo Google Chrome che ha adottato Geolocation API sin dalla quinta versione, oggi alla versione 125, e Firefox, adottata dalla versione 4 fino alla 126 odierna.

Una considerazione differente deve essere fatta circa la Generic Sensor API. Il browser più utilizzato a livello globale, Chrome, consiglia e supporta l'utilizzo dell'API a partire dalla versione 67 [40] (inizialmente solo accelerometer, gyroscope, orientation sensor e motion sensor).
A seguire i browser Samsung Internet, Opera (a partire dalla versione 54 [41] e poi estesa ad Opera Mobile a partire dalla versione 48) e Microsoft Edge, hanno deciso di adottarla anche loro.
Tuttavia, dai test effettuati su Sensorworker, emerge una discrepanza con le compatibilità dichiarate dal W3C per i dispositivi iOS. Infatti, a causa delle politiche in favore della privacy dei propri utenti adottate dalla Big Tech di

Cupertino, nonostante il browser sia compatibile con le APIs, interviene il sistema operativo in modo bloccante, negando così l'accesso ai sensori dal web.
Infatti, come riportato nella terza colonna della tabella, guardando i browser che hanno adottato l'API tra cui Chrome, Opera ed Edge, questa è supportata solo nel mondo Android.

Le medesime considerazioni valgono per le APIs dei basic sensor che estendono la Generic Sensor (accelerometro, sensore di accelerazione gravitazionale, giroscopio, sensore di accelerazione lineare, sensore di orientamento assoluto e relativo). Le versioni dei browser per le rispettive APIs sono infatti le stesse delle versioni di adozione di Generic Sensor.

Particolare attenzione va prestata alle API per l'accesso a due dei tre odierni sensori sperimentali: AmbientLightSensor API e Magnetometer API. Dalle informazioni messe a disposizione dal W3C sembra che tali sensori non siano accessibili ad alcun browser. Questa condizione è effettivamente rispettata da molti browser, ad eccezione di Chrome ed Edge. Questi sensori infatti sono accessibili da questi ultimi due browser. Bisogna però evidenziare che sono una versione sperimentale, quindi non si ha la sicurezza che i dati raccolti siano affidabili indiscutibilmente.

In conclusione, l'accesso ai sensori dei dispositivi dalle applicazioni web è diventato sempre più facile e compatibile con i diversi browser. Tuttavia, ci sono ancora alcune limitazioni e discrepanze nelle compatibilità, soprattutto per quanto riguarda i sensori sperimentali e i dispositivi iOS. Pertanto, è importante e consigliato prestare attenzione alle specifiche dei diversi sensori e delle APIs corrispondenti per garantire un accesso sicuro ed efficiente al relativo hardware.

Bibliografia

[1] Montori F., Jayaraman P. P., Yavari A., Hassani A., Georgakopoulos D. (2018). The Curse of Sensing: Survey of techniques and challenges to cope with sparse and dense data in mobile crowd sensing for Internet of Things. Pervasive and Mobile Computing, 49, 111-125.

[2] `https://www.wired.com/2006/06/crowds` Jeff Howe, 2006

[3] Sloane P. (2011). A guide to open innovation and crowdsourcing: Advice from leading experts in the field. Kogan Page Publishers.

[4] Ganti R. K., Ye F., Lei H. (2011). Mobile crowdsensing: current state and future challenges. IEEE communications Magazine, 49(11), 32-39.

[5] `https://creekwatch.ca/about/` (consultato in data 28.06.2024)

[6] `https://creekwatch.ca/wp-content/uploads/2017/03/CW-Monitoring-Plan-May-2015.pdf` (consultato in data 28.06.2024)

[7] `https://www.rvmobileinternet.com/gear/opensignal-app/` (consultato in data 28.06.2024)

[8] `https://appadvice.com/app/street-bump/528964742` (consultato in data 28.06.2024)

[9] `https://blog.route4me.com/how-to-use-waze/` (consultato in data 28.06.2024)

[10] https://electronics.howstuffworks.com/how-does-google-maps-predict-traffic.htm (consultato in data 28.06.2024)

[11] https://www.neuralword.com/en/article/connecting-a-heart-rate-monitor-to-strava-a-step-by-step-guide (consultato in data 28.06.2024)

[12] https://aws.amazon.com/it/compare/the-difference-between-web-apps-native-apps-and-hybrid-apps Amazon Web Services (consultato in data 14/06/2024)

[13] Fabio Mirza. Sensorworker: An Integrated Crowdsensing Platform. 2022

[14] Hirth M., Hoßfeld T., Tran-Gia P. (2011, June). Anatomy of a crowdsourcing platform-using the example of microworkers. com. In 2011 Fifth international conference on innovative mobile and internet services in ubiquitous computing (pp. 322-329). IEEE.

[15] Webster J. (2016, September). Microworkers of the gig economy: Separate and precarious. In New labor forum (Vol. 25, No. 3, pp. 56-64). Sage CA: Los Angeles, CA: SAGE Publications.

[16] https://developer.mozilla.org/en-US/docs/Web/API (consultato in data 25.05.2024)

[17] https://developer.mozilla.org/en-US/docs/Web/API/Sensor_APIs (consultato in data 25.05.2024)

[18] https://www.w3.org/blog/2017/w3c-to-work-with-mdn-on-web-platform-documentation (consultato in data 25.05.2024)

[19] https://www.w3.org/2020/09/web-roadmaps/mobile/sensors.html (consultato in data 29.05.2024)

[20] https://www.zdnet.com/article/apple-declined-to-implement-16-web-apis-in-safari-due-to-privacy-concerns (consultato in data 26.05.2024)

[21] https://blog.mozilla.org/security/2020/01/07/firefox-72-fingerprinting (consultato in data 26.05.2024)

[22] https://www.w3.org/TR/geolocation (consultato in data 25.05.2024)

[23] https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API (consultato in data 25.05.2024)

[24] https://caniuse.com/?search=geolocation%20api (consultato in data 25.05.2024)

[25] https://www.w3.org/TR/mediacapture-streams (consultato in data 26.05.2024)

[26] https://caniuse.com/stream (consultato in data 26.05.2024)

[27] https://www.w3.org/TR/generic-sensor (consultato in data 27.05.2024)

[28] https://caniuse.com/?search=sensor%20api (consultato in data 27.05.2024)

[29] https://www.w3.org/TR/ambient-light (consultato in data 25.05.2024)

[30] https://caniuse.com/?search=Ambient%20Light%20Sensor (consultato in data 27.05.2024)

[31] https://www.w3.org/TR/accelerometer (consultato in data 25.05.2024)

[32] https://caniuse.com/?search=Accelerometer (consultato in data 27.05.2024)

[33] https://www.w3.org/TR/gyroscope (consultato in data 29.05.2024)

[34] https://caniuse.com/?search=gyroscope (consultato in data 29.05.2024)

[35] https://www.w3.org/TR/magnetometer (consultato in data 29.05.2024)

[36] https://caniuse.com/?search=magnetometer (consultato in data 29.05.2024)

[37] https://www.w3.org/TR/orientation-sensor (consultato in data 29.05.2024)

[38] https://caniuse.com/?search=Orientation%20sensor (consultato in data 29.05.2024)

[39] https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices (consultato in data 25.05.2024)

[40] https://developer.chrome.com/blog/new-in-chrome-67 (consultato in data 01.06.2024)

[41] https://dev.opera.com/blog/opera-54 (consultato in data 01.06.2024)

Ringraziamenti

Desidero esprimere la mia profonda gratitudine a tutte le persone che mi hanno sostenuto ed accompagnato durante questo percorso accademico.

In primo luogo, voglio ringraziare con tutto il cuore mio padre e mia madre. La vostra presenza costante, il sostegno emotivo, l'amore e la spinta che mi avete dato nel corso del mio intero percorso formativo sono stati fondamentali. Grazie per la pazienza e la comprensione che mi avete sempre dimostrato, anche nei momenti di maggiore stress e per l'aiuto economico e i sacrifici fatti nell'arco di questi anni. Grazie per avermi permesso di diventare chi sono senza alcuna limitazione, con i vostri consigli e valori trasmessi.

Un grazie speciale a mio fratello, che con un'empatia che solo noi tacitamente conosciamo, mi ha permesso di superare ostacoli e confortato (anche se anche dopo quattro anni ancora non conosce il titolo del mio corso di studi).

Un grazie ai miei nonni, agli zii, a mio cugino e a Laura per il continuo affetto, l'interesse e il supporto che mi avete dimostrato.

Un grazie di cuore ai miei amici più stretti. Anche se iscritti a facoltà diverse, abbiamo condiviso emozioni simili che hanno arricchito questo percorso.

Ringrazio i miei compagni di corso, i nuovi amici e i conoscenti, con cui ho intrapreso ed affrontato questo viaggio. La loro presenza e compagnia mi ha arricchito e divertito profondamente.

Ovviamente, desidero esprimere la mia gratitudine al mio relatore, il professor Federico Montori, per la sua guida, il supporto e la costante disponi-

bilità. Grazie per l'energia e l'attitudine alla ricerca e al lavoro che mi ha trasmesso in questo breve ma intenso periodo. La sua esperienza e i suoi consigli mi hanno permesso di affrontare con maggiore sicurezza e consapevolezza questo atto finale del mio percorso accademico.

Ad maiora!