Tecnologie Web: Progressive Web Apps

Prof. Raffaele Montella, PhD
raffaele.montella@uniparthenope.it

Sommario

- Introduzione
- Tecnologie Web e Terminali Mobili
- Progressive Web App
- Service Worker
- Manifest
- Hello World
- Conclusioni

Introduzione

- Web e mobile computing sono due tecnologie intimamente connessi.
- Attualmente l’85% degli utenti mobile utilizza applicazioni di tipo smart client, il 15% applicazioni microbrowser.
- Un utente di smartphone spende l’80% del proprio tempo interagendo con solo 3 delle applicazioni installate.
- La maggior parte delle applicazioni smart client non sono altro che dei browser altamente specializzati.

Introduzione

- Solo il 20% delle applicazioni presenti negli store è scritto con SDK nativi Android / iOS.
- L’80% delle applicazioni pubblicate sono sviluppate con metodologia ibrida.
- L’applicazione è scritta usando tecniche proprie delle tecnologie web. Poi è “embedded” in un runtime in codice nativo.
- Questa applicazioni si comportano come applicazioni native.

Introduzione

- Vantaggi delle applicazioni mobile native:
  - Possono sfruttare appieno le caratteristiche del device.
  - Possono funzionare anche in assenza di rete.
  - Sono pubblicabili (e vendibili sugli store)

- Svantaggi:
  - Lunghi tempi di pubblicazione ed aggiornamento.
  - È necessario sottostare alle policy degli store.
  - La gestione della codebase per applicazioni per differenti dispositivi può essere costosa.

Progressive Web App

- È progettata in modo da funzionare su qualsiasi browser la supporti incrementando progressivamente le funzionalità a seconda del dispositivo usato.

- È sviluppata in modo che si adatta a qualsiasi fattore di forma dei display utilizzati: desktop/laptop, tablet, smartphone, smart-tv e altro in futuro.

- È indipendente dalla connessione poiché deve essere sviluppata in modo da essere disponibile anche in assenza di rete o quando la rete è di cattiva qualità.

Progressive Web App

- L’interfaccia utente e la relativa user experience è tale da farla essere simile ad una app in tutto e per tutto applicando il principio della UX-Convergence.

- UX-Convergence:
  Interfaccia utente comune a differenti sistemi di fruizione in modo da aumentare il “riciclo di conoscenza” da parte degli utenti. È una delle leve per la limitazione del digital divide.

- Grazie a un componente che lavora in background ha un comportamento resposivo in termini di disponibilità dei dati (service worker).

Progressive Web App

- Sono sicure poichè devono essere servite tramite HTTPS 2.
- Hanno un manifest file che raccoglie le informazioni principali e la configurazione in modo da essere trovate e catalogate dai motori di ricerca.
- Uso intensivo delle notifiche di tipo push per garantire il coinvolgimento degli utenti che non devono visitare l’applicazione per essere aggiornati.
- Gli utenti possono installare un’applicazione di questo tipo sul proprio desktop al parti di una app nativa.

Interfaccia Utente e User Experience

- Le PWA devono poter essere scaricate immediatamente, anche nel caso in cui il server sul quale è pubblicata è down o la connessione di rete è di cattiva qualità.

- Deve rispondere in modo rapido, se presenti le animazioni devono essere armoniosi e veloci, soprattutto devono funzionare a tutto schermo senza dover richiedere lo scroll della pagina.

- Devono non essere dissimili da una app nativa e con queste condividere la U-X.

L’app store non è necessario.

- Le PWA sono installabili sullo home screen degli utenti.
- Non è necessario scaricarle da uno store.

Vantaggi:

- Rapido deployment, non bisogna sottostare alle policy.
- Sempre aggiornate, non è necessario cambiare software sul dispositivo.

Svantaggi:

- Non è possibile “vendere” le applicazioni.
- Le applicazioni non sono attualmente raccolte in “store” o cataloghi.

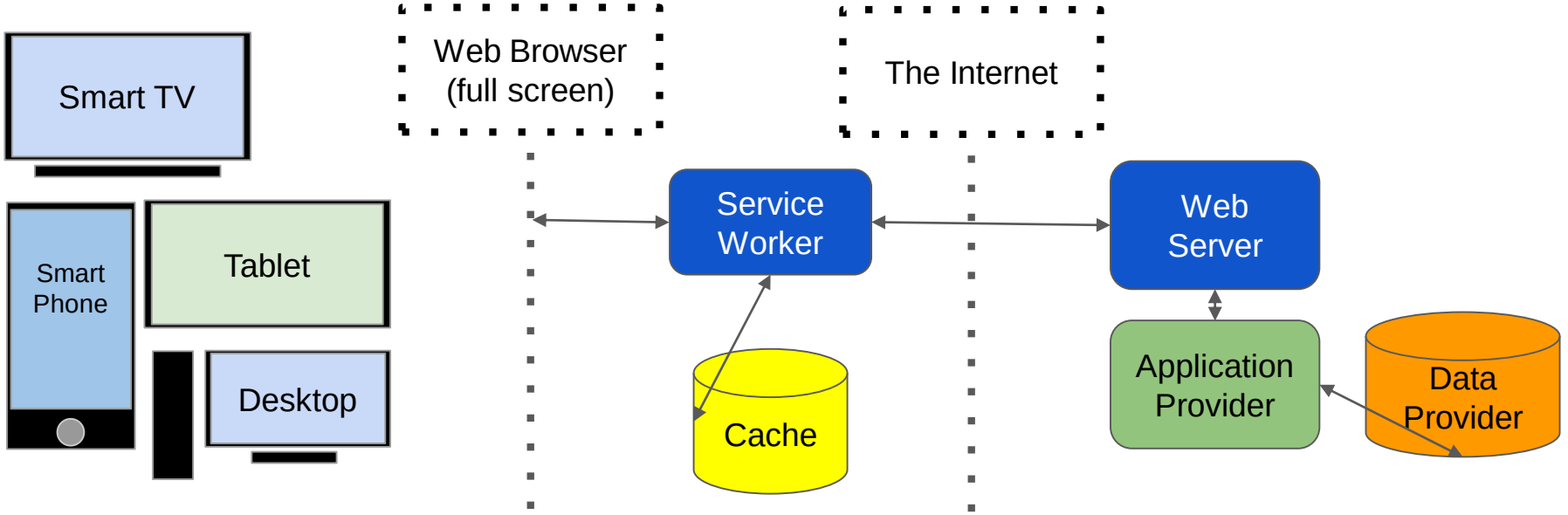
Sempre disponibile: il service worker

- È un proxy client side che consente di intercettare tutte le HTTP request effettuate dall'applicazione.
- Consente di avere un controllo completo sulla cache.
- Le risorse sono identificate secondo uno schema chiave/valore.
- La cache è gestita automaticamente dal browser, ma è possibile personalizzare la risposta alle richieste.
- Il pre-caching delle risorse consente di mitigare l'assenza di rete.

Sempre disponibile: il service worker

- L’uso personalizzato della cache può consentire elaborazioni avanzate.
- Esempio:
  - Un’applicazione meteo deve mostrare le previsioni in relazione alla posizione geografica dell’utente.
  - L’applicazione fa una HTTP Request per controllare se ci sono previsioni aggiornate.
  - Se sono disponibile le scarica.
  - Se è disponibile la rete, l’applicazione scarica i dati per 72 ore di previsione.
  - Alla successiva richiesta, anche se la rete non è presente le previsioni saranno disponibili.

Sempre disponibile: il service worker



- Quando l'applicazione è lanciata tramite icona, il service worker consente alla progressive web app di caricare istantaneamente indipendentemente dallo stato della rete.

Sempre disponibile: il service worker

- In questo modo si evita che l’utente abbandoni l’applicazione se il caricamento è troppo lento.

- Il 53% degli utenti abbandona un’applicazione o un portale web se non è responsivo entro 3 secondi.

Sempre disponibile: il service worker

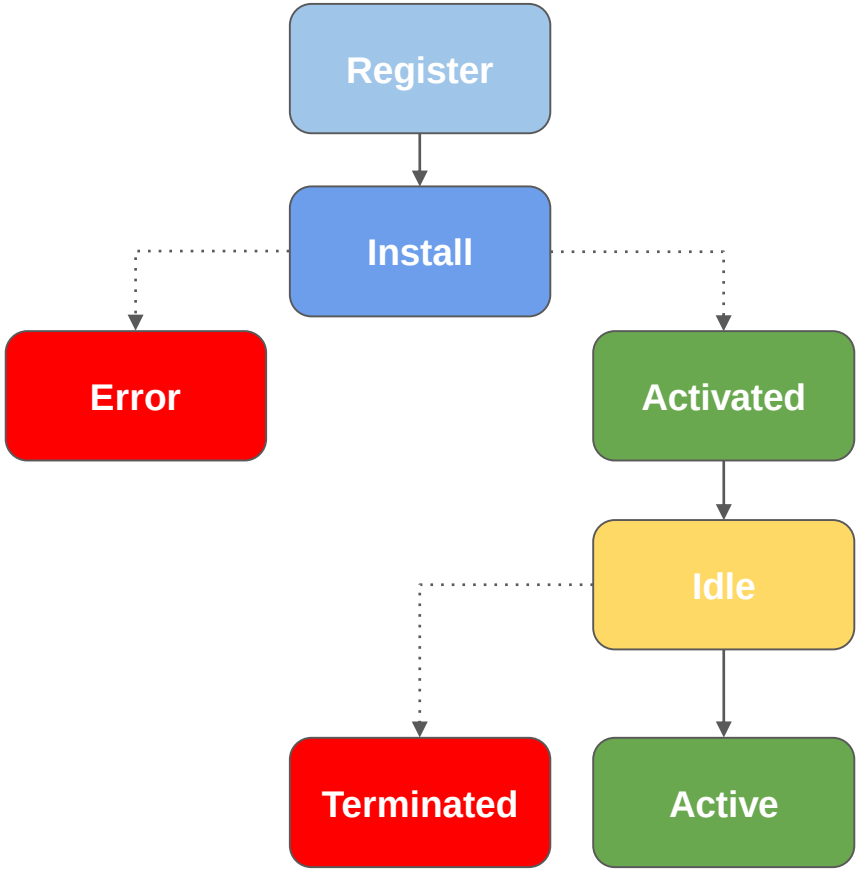
- Il service worker è eseguito in background dal browser indipendentemente dall’applicazione.

- È un file javascript che NON ha accesso diretto al Document Object Model.

- Può comunicare con le pagine che controlla rispondendo a messaggi inviati attraverso postMessage.

Sempre disponibile: il service worker

- Le pagine, se necessario possono intervenire sul DOM.
- Il service worker è un network proxy programmabile che consente di controllare come le pagine vengono caricate.
- Lifecycle.



L’application manifest

- Le PWA offrono una U-X di tipo full screen.
- Possono usare la push notification.
- Il manifest è un file json che raccoglie le impostazioni dell’applicazione.
- Il manifest consente di specificare come l’applicazione deve essere lanciata.
- Consente di specificare l’icona, l’orientazione dello schermo, nascondere o visualizzare la finestra del browser

Lighthouse

- È un tool automatico open source per migliorare la qualità delle pagine web.

- È possibile controllare qualsiasi pagina, anche pubblica, anche se richiede autenticazione.

- Ha la possibilità di fare testare su:
    - Performance
    - Accessibilità
    - Progressive Web App
    - Altro...

PWA Hello World

- Assicurarsi di avere Google Chrome installato.
- Utilizzare un qualsiasi IDE provvisto di un web server locale per la prova delle applicazioni web.
- Si suggerisce di usare WebStorm (https://www.jetbrains.com/webstorm/)
- Creare un nuovo progetto vuoto chiamandolo pwa01

PWA Hello World

- Creare le seguenti directory:
  - css
  - js
  - Images
- Creare il file index.html
- Creare il file css/style.css

PWA Hello World: index.html

```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Hello World</title>
        <link rel="stylesheet" href="css/style.css">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body class="fullscreen">
        <div class="container">
            <h1 class="title">Hello World!</h1>
        </div>
    </body>
</html>
```

PWA Hello World: css/style.css (1/2)

```
body {
  font-family: sans-serif;
}

/* Make content area fill the entire browser window */
html,
.fullscreen {
  display: flex;
  height: 100%;
  margin: 0;
  padding: 0;
  width: 100%;
}
```

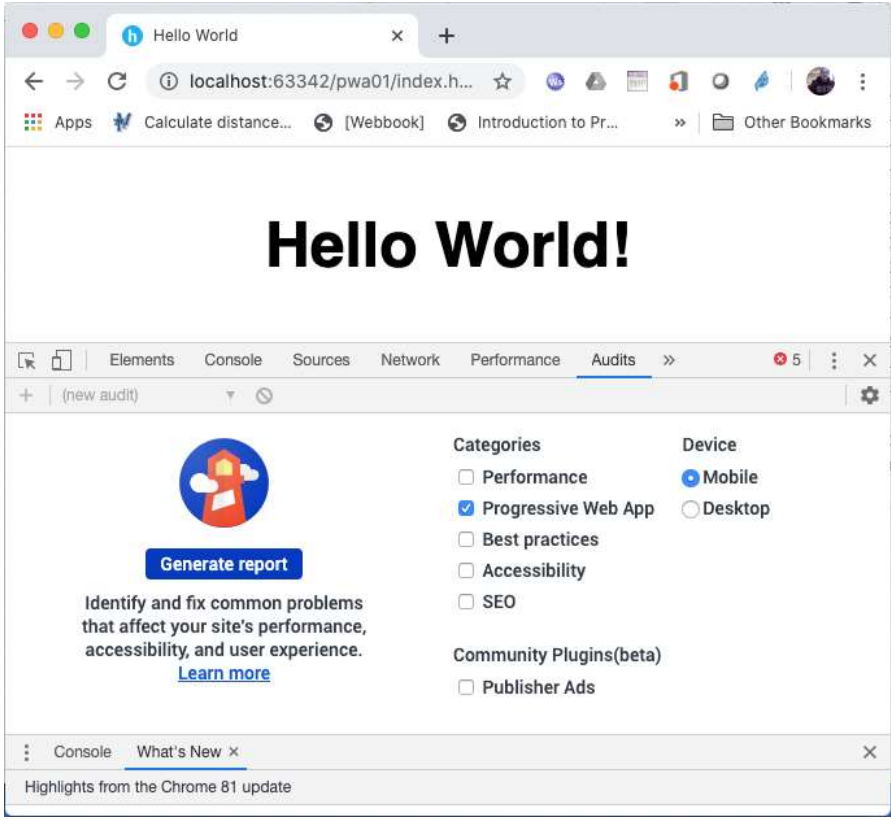
PWA Hello World: css/style.css (2/2)

```
/* Center the content in the browser window */
.container {
  margin: auto;
  text-align: center;
}

.title {
  font-size: 3rem;
}
```

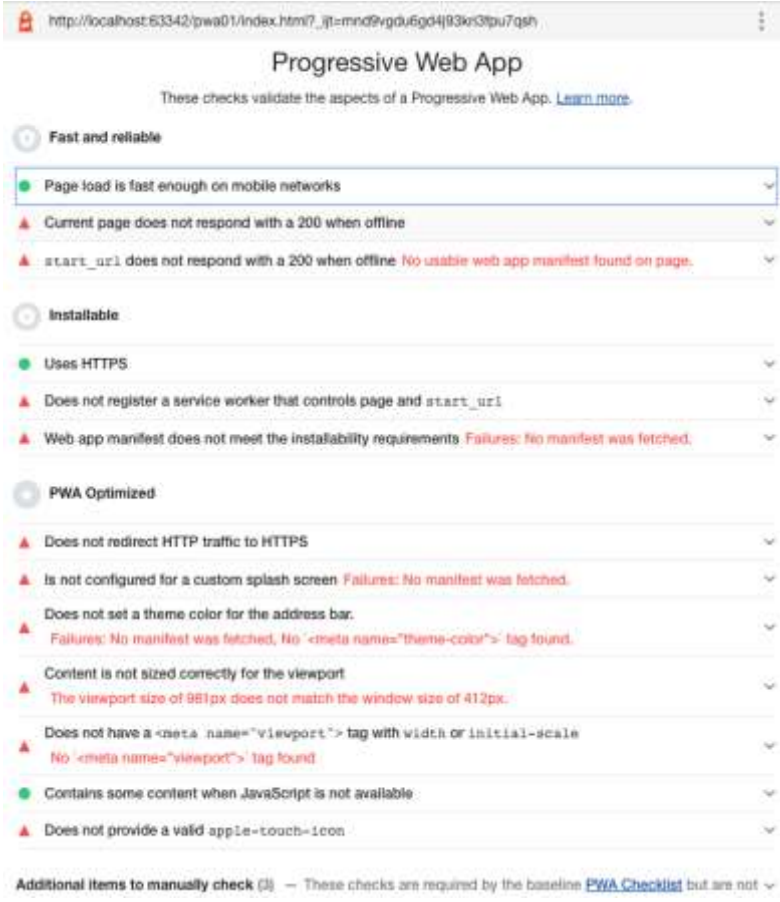
PWA Hello World

- Dalla finestra dell'editor del file index.html, lanciare il browser Chrome.
- Attivare Lighthouse premendo il tasto F12.
- Selezionare l'opzione Audits.
- Selezionare solo "Progressive Web App" e fare click su **Generate Report**.



PWA Hello World

- L’applicazione non risulta ottimizzata come PWA.
- Creiamo il file che implementa il service worker e il programma chiamante dell’applicazione.
- File:
  - sw.js
  - js/main.js



PWA Hello World: sw.js (1/2)

```
let cacheName = 'pwa01';
let filesToCache = [
    'index.html',
    'css/style.css',
    'js/main.js'
];
/* Start the service worker and cache all of the app's content */
self.addEventListener('install', function(e) {
    e.waitUntil(
        caches.open(cacheName).then(function(cache) {
            return cache.addAll(filesToCache);
        })
    );
});
```

PWA Hello World: sw.js (2/2)

```
/* Serve cached content when offline */
self.addEventListener('fetch', function(e) {
    e.respondWith(
        caches.match(e.request).then(function(response) {
            return response || fetch(e.request);
        })
    );
});
```

PWA Hello World: js/main.js (1/2)

```
window.onload = () => {
    'use strict';

    if ('serviceWorker' in navigator) {
        navigator.serviceWorker
            .register('./sw.js').then(function (registration) {

            // Service worker registered correctly.
            console.log('ServiceWorker registration successful with scope: ',
registration.scope);
        },
```

PWA Hello World: js/main.js (2/2)

```
        function (err) {

            // Troubles in registering the service worker. :(
            console.log('ServiceWorker registration failed: ', err);
        });
    }
}
```

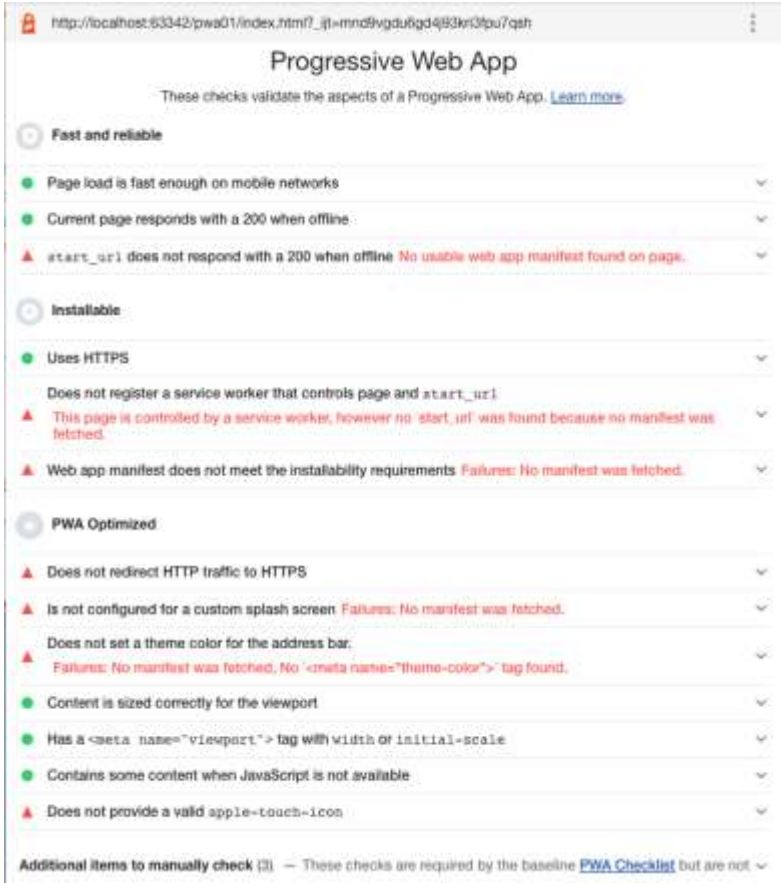
PWA Hello World

- Aggiungere la seguente linea nel file index.html subito prima della chiusura del tag </body>:

```
<script src="js/main.js"></script>
```

- Eseguire nuovamente il test tramite Lighthouse.

- Ora il service worker consente di caricare la pagina quando si è offline.



PWA Hello World: manifest.json

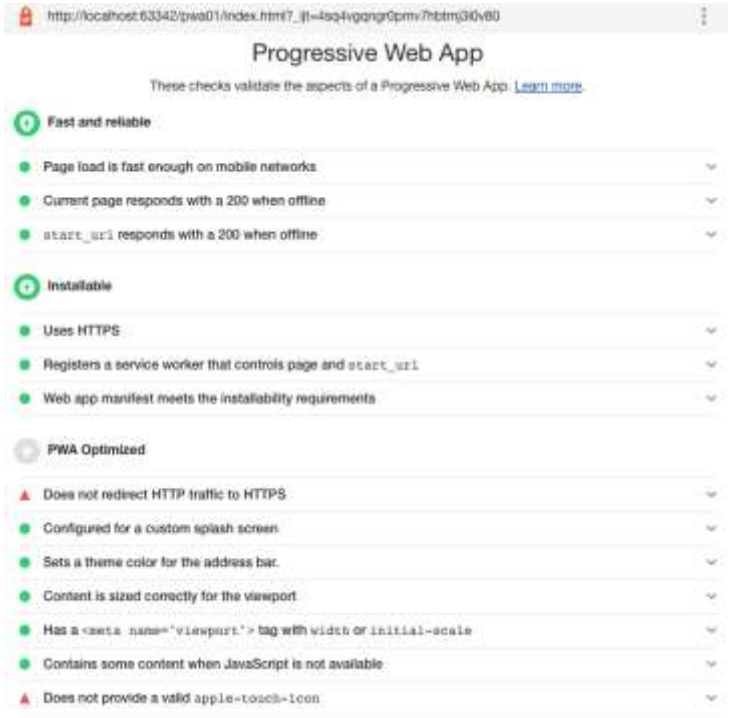
- È necessario creare il file manifest.json nella root del progetto:

```
{
  "name": "Hello World",
  "short_name": "Hello",
  "lang": "en-US",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "white",
  "theme_color": "white"
}
```

- E aggiungere le seguenti righe nello header del file index.html:

```
<link rel="manifest" href="manifest.json">
<meta name="theme-color" content="white"/>
```

PWA Hello World



- Eseguire nuovamente il test tramite Lighthouse.
- Gran parte dei requisiti perchè un’applicazione web sia considerata una PWA sono soddisfatti.
- Non rimane che aggiungere le icone.
- Poiché device differenti hanno risoluzioni differenti è necessario creare un set di icone appropriato.

Conclusioni

- Le Progressive Web App sono una valida alternativa alle smart client app.
- Oggi costituiscono lo standard a meno di casi particolari.
- Non hanno bisogno di essere pubblicate su di uno store.
- Non è possibile venderle.
- È possibile mantenere una unica codebase per differenti client.