Pràctica 3: Aplicacions per a mòbils

Interfícies d'usuari — Enginyeria de Sistemes TIC

Sebastià Vila-Marta Aleix Llusà Serra

17 de febrer de 2023

Índex

Resum

Llenguatges web: HTML, CSS, JS
Estàndards web: WebAPIs, PWA
Entorn d'aplicacions per a mòbils: Apache Cordova

1 Organització

En aquesta pràctica dissenyarem una interfície mòbil.

1.1 Lliurables

En acabar aquesta pràctica caldrà lliurar l'aplicació desenvolupada juntament amb un manual tècnic sobre el funcionament i el disseny de l'aplicació.

1.2 Material necessari

Per dur a terme la pràctica cal tenir instal · lat l'entorn d'aplicacions per a mòbils *Apache Cordova*, eines per a emular una interfície mòbil i eines per a empaquetar l'aplicació mòbil.

2 Introducció

Una interfície d'usuari per a dispositius mòbils és la interfície que permet els usuaris interactuar amb les funcionalitats i característiques dels dispositius mòbils, com per exemple els telèfons intel·ligents i les tauletes tàctils.

Principalment, hi ha tres maneres de desenvolupar interfícies d'usuari per a dispositius mòbils: aplicacions web per a mòbils, aplicacions natives i aplicacions híbrides.

2.1 Aplicacions web per a mòbils

Les *aplicacions web per a mòbils* construeixen les pàgines web *amigables* per a mòbils. Així doncs, consisteixen en reestructurar els continguts i els estils per tal que s'adaptin als requeriments dels dispositius mòbils. En aquest cas es poden utilitzar les tècniques de *responsive web design* i *CSS media queries* que hem treballat en pràctiques anteriors.

L'inconvenient principal de les aplicacions web per a mòbils és la dificultat d'accedir a les funcionalitats extres dels dispositius, com per exemple la càmera o els sensors. Actualment l'accés a aquestes funcionalitats des dels navegadors es restringeix per motius de seguretat. Això no obstant, s'estan desenvolupant interfícies de programació segures per tal de solucionar aquestes restriccions (`https://developer.mozilla.org/en-US/docs/WebAPI`).

2.1.1 Progressive Web Apps

Les Progressive Web Apps (PWA) són aplicacions web que aporten la mateixa experiència d'usuari que les aplicacions natives per a mòbils o les híbrides. De manera similar també aporten la mateixa experiència d'usuari que les aplicacions natives per a escriptoris o les híbrides.

Les PWA són un conjunt de tecnologies web. Inclouen les mencionades anteriorment de *responsive web design* i les WebAPI (p.ex. la Push i Notifications API que permet emular les notificacions offline). A més, un fitxer de Manifest W3C (que permet emular la instal·lació de la app amb una icona) i ServiceWorkers (que permeten consultar offline les dades de la app).

L'avantatge de les PWA és que per a l'usuari semblen una app normal i s'intał · len directament des del navegador, no necessiten ser instal·lades des de la botiga del fabricant. L'inconvenient

principal és que no estan suportades encara per tots els navegadors, així com tampoc totes les WebAPI tenen encara un ampli suport.

2.2 Aplicacions natives

Les *aplicacions natives* es dissenyen específicament per a cada dispositiu mòbil. És a dir, el disseny de l'aplicació depèn de la plataforma a la qual es vulgui executar. Així, habitualment, es dissenyen en Java per a dispositius Android, en Objective-C per a dispositius iOS, en C# per a dispositius Windows i en QML per a dispositius Ubuntu Touch. D'aquesta manera, aquestes aplicacions poden aprofitar tot el potencial de cada dispositiu.

L'inconvenient principal de les aplicacions natives és precisament la dificultat de dissenyar una mateixa aplicació en diferents entorns i els costos elevats de desenvolupament que això pot suposar.

Des d'una altra vessant cal destacar els desenvolupaments per a utilitzar qualsevol GNU/Linux en els ARM més el suport per a drivers específics de mòbils. Per exemple Manjaro ARM, Mobian, postmarketOS, PureOS o Ubuntu Touch. Actualment és plenament viable mitjançant els dispositius construïts específicament per a executar-los, com PinePhone i Librem. Per a més informació vegeu `https://wiki.debian.org/Mobile` i `https://wiki.debian.org/PinePhone`.

2.3 Aplicacions híbrides

Les *aplicacions híbrides web* es dissenyen amb la mateixa tecnologia que les aplicacions web per a mòbils –HTML, CSS i JavaScript– però s'executen encastades en navegadors locals. Així, es dissenyen com a aplicacions multiplataforma però posteriorment s'empaqueten específicament per a cada dispositiu. Aquestes aplicacions tenen accés a les funcionalitats extres dels dispositius mitjançant interfícies estàndard que adapten les especificitats de cada dispositiu.

L'inconvenient principal de les aplicacions híbrides és que poden ser més lentes en explotar els recursos propis dels mòbils que les aplicacions natives. Tanmateix, les aplicacions híbrides poden permetre desenvolupar plugins natius per fer d'interfície amb aquestes parts. Un altre incovenient és que és més difícil obtenir l'aparença i el disseny natiu de cada plataforma mòbil.

En aquesta pràctica desenvoluparem una aplicació mòbil mitjançant un entorn d'aplicacions híbrides web: *Apache Cordova*. Podeu trobar altres entorns similars –com Ionic– o entorns de desenvolupament que utilitzen altres tècniques de programació per aconseguir que les aplicacions siguin *cross-platform* –com Xamarin, React Native o Flutter. Aquestes darreres desenvolupen *aplicacions híbrides natives* i se solen dissenyen amb JavaScript més elements d'interfície específics per a cada dispositiu.

En general, les aplicacions híbrides es poden executar tant en interfícies mòbils com en escriptori o com a Progressive Web App.

3 Progressive Web App

Referència principal: `https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps`

Les *Progressive Web App* (PWA) requereixen com a mínim dues tecnologies web: el fitxer *Manifest* i el *ServiceWorker*. A continuació es descriu com convertir una web a PWA implementant aquests mínims.

3.1 Web App Manifest

Referència principal: `https://developer.mozilla.org/en-US/docs/Web/Manifest`

El *Web Application Manifest* és un estàndard del W3C que defineix les metadades d'una app web en un fitxer JSON. Aquestes metadades són per exemple el nom de l'app, la icona, l'autor o si cal obrir l'app a pantalla completa.

Gràcies al Manifest i a la característica *Add to Home screen* (A2HS, `https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Add_to_home_screen`) es pot instal·lar l'aplicació en els mòbils des del navegador, a través d'una opció A2HS per a instal·lar-la que apareix a l'usuari. En els escriptoris de moment només Chrome (també Chromium i Edge basat en Chromium) ofereix la funcionalitat A2HS.

Per a declarar el Manifest només cal enllaçar el fitxer en el *head* del document web:

```
<link rel="manifest" href="web.manifest"/>
```

Un exemple de fitxer `web.manifest` és el següent:

```
{
  "short_name": "iu exemple",
  "name": "Interfícies d'usuari (exemple)",
  "start_url": ".",
  "background_color": "#fff",
  "display": "standalone",

"icons": [
    {
      "src": "/img/icona48.png",
      "sizes": "48x48",
      "type": "image/png"
  },
    {
      "src": "/img/icona192.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ]
}
```

3.2 Service Workers

Referència principal: `https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API` i `https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers`

Els *Service Workers* són un estàndard del W3C que defineixen un *Web Worker* (un procés que s'executa en paral·lel al navegador) que actua de proxy entre la navegació del navegador i les peticions que el navegador fa a la xarxa. El principal objectiu dels Service Workers en el context

de les PWA és la definició d'una memòria cau fora de línia per a les aplicacions web (*offline cache*, definida a la Cache Storage API dins del mateix W3C Service Worker).

El funcionament dels Service Workers és orientat a esdeveniments i en ser un procés a part no tenen accés directe al DOM. Cada Service Worker es registra en el context d'un origen i camí i només poden accedir als recursos continguts en aquell origin i subcamins, a més dels recursos externs que requereixi aquell origen.

Cada Service Worker segueix un pas inicial de registre més un flux bàsic de quatre esdeveniments, on els objectius genèrics de les PWA poden ser els següents (cada PWA pot adaptar l'emmagatzematge cau a les seves necessitats, vegeu-ne més a `https://serviceworke.rs/`):

1. Registrar: inicialment, indicar quin és el fitxer JS que defineix el Service Worker
2. Instal·lar: emmagatzemar a la memòria cau els recursos estàtics inicials
3. Activar: eliminar memòries caus anteriors
4. Fetch: atendre a les peticions de xarxa com a proxy. Retornar els resultats de la memòria cau o fer les peticions HTTP (tot emmagatzemant el nous resultats a la memòria cau)
5. Push: rebre notificacions del sistema operatiu, fins i tot quan l'usuari no utilitza l'aplicació.

3.2.1 Exemple

`index.html`

```
<!DOCTYPE html>
<html lang="ca">
<head>
  ...
  <link rel="manifest" href="pwa.webmanifest">
  <script src="serviceworker_register.js" defer></script>
  ...
</head>
<body>
...
```

`serviceworker_register.js`

```
if('serviceWorker' in navigator) {
  // Registering Service Worker
  navigator.serviceWorker.register('serviceworker.js');
};
```

`serviceworker.js`

```
// version
var cacheVersion = 'pwa-v1';

// pre-cache
initialCache = [
    '/',
    'index.html',
    'serviceworker_register.js',
    'img/icona48.png',
    'img/icona192.png'
];
```

```
// always update
updateCache = [
    '/',
    'index.html',
    '/8000/',
];

// without cache
noCache = [
    '/pagina.html'
]



// Installation
self.addEventListener('install', function(e) {
    console.log('[Service Worker] Install');
    e.waitUntil(
    caches.open(cacheVersion).then(function(cache) {
      console.log('[Service Worker] Caching all content');
      return cache.addAll(initialCache);
    })
  );
    self.skipWaiting();
});


// Activate
self.addEventListener('activate', (e) => {
    console.log('[Service Worker] Activating');
    e.waitUntil(
    caches.keys().then((keyList) => {
          return Promise.all(keyList.map((key) => {
        if(key !== cacheVersion) {
          return caches.delete(key);
        }
      }));
    })
  );
  self.clients.claim();
});

// Active fetch
self.addEventListener('fetch', function(e) {
  e.respondWith(fetch_cache_fallback_network(e.request));
  // always update from network?
  var requestURL = new URL(e.request.url);
  if (updateCache.indexOf(requestURL.pathname) != -1 ){
    console.log('[Service Worker] Always update: '+e.request.url);
      e.waitUntil(fetch_network_and_cache(e.request));
  }
});


// auxiliars
function fetch_network_and_cache(request){
  return fetch(request).then(function(response) {
    // no cache?
```

```
    var requestURL = new URL(request.url);
    if (noCache.indexOf(requestURL.pathname) != -1 ){
        console.log('[Service Worker] No cache: ' + request.url);
        return response;
    }
    return caches.open(cacheVersion).then(function(cache) {
      console.log('[Service Worker] Caching new resource: ' + request.url);
      cache.put(request, response.clone());
      return response;
    });
  });
}

function fetch_cache_fallback_network(request){
  return caches.match(request).then(function(r) {
    console.log('[Service Worker] Fetching resource: '+request.url);
    // return from cache, fallback to network
    return r || fetch_network_and_cache(request);
  });
}
```

El fitxer serviceworker.js és el Service Worker i sempre s'óbté de la xarxa. El cicle d'actualitzar un Service Worker (instal·lar/activar) es duu a terme cada cop que hi ha un canvi en aquest fitxer JS. Quan es canvia la *cacheVersion* es genera un nou repositori d'emmagatzematge cau i s'eliminen les versions anteriors.

El patró de funcionament d'aquest Service Worker és el següent. Els recursos estàtics *initialCache* s'emmagatzemen a la cau i no es tornen a obtenir mai més de la web (només quan es canvia la *cacheVersion*). Els recursos definits a *updateCache* són emmagatzemats a la cau per a usar-los fora de línia però se segueixen actualitzant de la web. La resta de recursos s'emmagatzemen a la cau la primera vegada que se sol·liciten i segueixen el mateix patró que els recursos estàtics, excepte que es defineixin a *noCache*.

3.3 Eines

Podeu provar la vostra PWA tant al mòbil com a l'escriptori. El suport a escriptori de l'A2HS de moment només es troba a Chrome/Chromium. Això no obstant, el Service Worker té un suport més ampli en els diversos navegadors i per tant podeu utilitzar la funcionalitat de fora de línia sense problemes.

Els Service Worker només funcionen per localhost o per HTTPS. En el cas de Chrome es poden inserir altres dominis HTTP a `chrome://flags/#unsafely-treat-insecure-origin-as-secure` per a testejar. Per a Firefox comproveu a `about:config` que està activat el `dom.serviceWorkers.enabled`.

En el cas de Firefox escriptori podríeu provar d'activar `dom.manifest.enabled` i `browser.ssb.enabled` per a utilitzar l'opció *Use This App in Site Mode*.

4 Apache Cordova

Apache Cordova és un entorn de desenvolupament d'aplicacions híbrides per a mòbils. Permet dissenyar aplicacions amb tecnologia web –HTML5, CSS3, i JavaScript– i empaquetar-les per

a múltiples dispositius mòbils –Apple iOS, Firefox OS, Google Android, Microsoft Windows Phone, Ubuntu Touch– i també per a aplicacions d'escriptori de qualsevol plataforma mitjançant Electron (`https://www.electronjs.org/`).

Per a utilitzar Apache Cordova primer cal instal·lar l'entorn de desenvolupament i després cal instal·lar les eines particulars de cada plataforma per a la qual es vulgui empaquetar l'aplicació. També es poden instal·lar eines per a emular les plataformes i poder provar prèviament l'aplicació. A més, Apache Cordova per defecte té un emulador de les aplicacions preparat per a ser utilitzat a través del navegador d'escriptori.

4.1 Instal·lació de l'entorn

Podeu seguir la documentació ràpida `http://cordova.apache.org/#getstarted` per a configurar l'entorn de treball bàsic.

Per a Debian o Ubuntu primer us caldrà instal·lar els paquets de sistema: `nodejs npm gradle`.

En segon lloc, heu d'instal·lar les eines de *cordova*. Podeu:

- Instal·lar-les en un directori (`node_modules`) del vostre usuari executant `npm install cordova`
- Instal·lar-les en el sistema (com a superusuari) executant `npm install -g cordova`
- A Ubuntu, instal·lar el paquet `cordova-cli`

Un cop instal·lades tindreu disponible l'ordre `cordova`, excepte en el primer cas que la tindreu disponible a `node_modules/cordova/bin/cordova`.

En tercer lloc, podeu crear una nova aplicacio `cordova create nom-aplicacio`.

Finalment, podeu afegir la plataforma de navegador

```
cd nom-aplicacio
cordova platform add browser
```

I comprovar el funcionament de l'aplicació a través del navegador:

```
cordova run browser
```

En el cas que no tingueu Chrome o el tingueu en un altre camí, podeu usar un altre navegador amb `cordova run browser -target=firefox`.

4.2 Eines per a Android

Referència: `https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html`

Per a empaquetar l'aplicació per a dispositius Google Android, cal instal·lar l'entorn de desenvolupament d'aquests dispositius. Per Debian o Ubuntu podeu seguir les instruccions següents; heu de tenir instal · lat `java` al sistema.

En primer lloc, heu de descarregar les eines. Amb les Command Line Tools ja en tindreu prou `https://developer.android.com/studio/index.html#downloads`. Descomprimiu-ho per exemple a `~/android-sdk`.

En segon lloc, instal·leu les *build-tools*: `~/android-sdk/tools/bin/sdkmanager "build-tools; XX.X.X"`. Amb `~/android-sdk/tools/bin/sdkmanager --list` podeu comprovar quina és la darrera versió. *Nota: En cas que no funcioni l'ordre, hauríeu de tenir una estructura similar a* `~/android-sdk/cmdline-tools/tools/bin/sdkmanager`.

En tercer lloc, configureu la variable d'entorn `export ANDROID_HOME=/home/<usuari>/android-sdk`.

Nota: per a Ubuntu hi ha disponible el paquet snap `https://snapcraft.io/android-studio`: *snap install android-studio*.

Finalment, ja podeu usar les eines a Cordova. Primer afegiu la plataforma:

```
cd nom-aplicacio
cordova platform add android
```

I tot seguit ja podeu empaquetar l'aplicació:

```
cordova build android
```

El fitxer `.apk` resultant ja està llest per a ser instal·lat en un dispositiu Android.

Si connecteu el dispositiu mòbil mitjançant USB, podeu instal · lar el paquet `adb` i utilitzar una drecera a empaquetar, copiar i instal·lar l'apk a través de:

```
cordova run android
```

Per a la connexió mitjançant `adb` pot caldre activar les *Opcions per a desenvolupadors*. Per activar-les s'ha d'anar a *Configuració*, *Sobre el Telèfon* i clicar 7 cops a *Número de compilació* (vegeu `https://developer.android.com/studio/command-line/adb#Enabling`)

4.2.1 Debugar

Referència: `https://cordova.apache.org/docs/en/latest/guide/next/#chrome-remote-debugging`

Per a tenir eines per a poder debugar l'execució de l'app a android haureu de fer dos passos:

1. `cordova run android` Executa l'aplicació a android en mode debug
2. `chrome://inspect` Visitar la URL per a debugar des de `chrome`

Llavors podreu obrir un inspector igual que l'inspector web però per a l'aplicació mòbil.

4.2.2 CSP i CORS

Referència: `https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-whitelist`

Per a poder utilitzar AJAX a l'aplicació haureu de canviar al fitxer `app/www/index.html`:

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self' ...

a

<meta http-equiv="Content-Security-Policy"
      content="default-src 'self' http://servidor.test:port ...
```

Per a poder provar AJAX en el mode *browser*, a més a més del canvi en el fitxer `app/www/index.html`, també haureu d'afegir en el vostre servidor la configuració del CORS:

```
Access-Control-Allow-Origin: '*'
```

Per a poder establir Cookies a través d'AJAX/XHR, pot fer falta activar el paràmetre *withCredentials* (`https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/withCredentials`):

```
xhr.withCredentials = true;
```

4.2.3 Emulador

Si voleu emular l'aplicació, sense haver d'instal·lar-la en un dispositiu, podeu seguir els passos següents.

En primer lloc, instal·leu una imatge de sistema, per exemple seleccioneu tota la darrera API, mitjançant `~/android-sdk/tools/androidsdk`.

A continuació, creeu un nou AVD a partir de la imatge instal·lada mitjançant `~/android-sdk/tools/androidavd`.

Finalment, emuleu l'aplicació en el dispositiu virtual:

```
cordova emulate android
```

Nota: Les emulacions solen ser procediments bastant lents. És preferible l'execució de l'aplicació directament al dispositiu.

5 Feina a fer

Heu de dissenyar i implementar la vostra aplicació mòbil. Seguiu els passos següents:

1. Dissenyeu els wireframes de la vostra aplicació. Això vol dir dibuixar: en paper o amb un programa gràfic.
2. Implementeu l'aplicació segons les instruccions d'aquesta pràctica i consultant la documentació completa d'Apache Cordova a `http://cordova.apache.org/docs/en/latest/`. Feu ús de la documentació extra que us calgui, per exemple si heu d'utilitzar plugins d'Apache Cordova per a accedir a maquinari del dispositiu o per exemple si necessiteu accedir a una API externa d'un backend web.
3. Simuleu la vostra aplicació en el navegador local fins que esteu segurs que funciona correctament.
4. Empaqueteu l'aplicació i proveu-la en un dispositiu real.
5. Testegeu la usabilitat de la vostra aplicació.
6. Elaboreu un manual d'usuari i un manual tècnic del disseny de la vostra aplicació. Incloeu els wireframes en aquest manual.