Crearea aplicatiilor web pentru dispozitive mobile

Ispas Stefan

Damaschin Ionut

Frincu Andrei

Ispas Stefan:

Capitolul 1. Introducere (Design,interactiune cu utilizatorul)

Capitolul 2. Diferente intre aplicatiile Web si aplicatiile native pentru dispozitivele mobile

Damaschin Ionut:

Capitolul 3.Concepte si tehnici pentru constructia unor aplicatii web pentru dispozitivele mobile

3.1. Pastrarea fisierelor la dimensiuni mici

3.2. Confruntarea cu performanta imaginilor

Capitolul 4.Redirectionari.Avantaje CSS3

4.1. Evitarea redirectionarilor

4.2. Aplicarea datelor

4.3. Securitate si intimiate

4.4. Controlul si constientizarea utilizatorului

Frincu Andrei:

Capitolul 5.Securitate.Autentificare

5.1.1. SQL Injection

5.1.2. Cross-site Scripting(XSS)

5.1.3. Cross-site Request Forgery(CSRF)

Capitolul 6.Crearea unei aplicatii web folosind jQuery Mobile

Capitolul 7: Concluzii – Am adaugat toti concluziile temei.

**1. Introducere(Design,interactiune cu utilizatorul)**

Aplicatiile web pentru dispozitive mobile sunt alternative utile fata de aplicatiile native pentru dispozitive mobile. In zilele noastre, produsele iOS si Android bazate pe dispozitive, cum ar fi iPhone, iPad,Samsung toate vin standard cu browsere mobile (Mobile Chrome pentru Android și, respectiv, Mobile Safari pentru IOS).

De asemenea fanii Opera pot instala browser-ul lor preferat.

Aceste produse fac navigarea o experienta foarte plăcută Tranzițiile CSS3, HTML5 au facut ca utilizatorii de internet sa poata vizualiza pe mobil aceleasi pagini web ca de pe notebook sau descktop,avand avantajul mobilitati si portabilitatii ridicate(presupunând conținutul ce urmeaza a fi vizualizat conceput cu grijă și considerație).

Aplicatiile web pentru dispozitive mobile vor avea probleme de performanță. Un avantaj nativ al acestora este ca descărcați o mare parte din interfața atunci când descărcați aplicația inițial, astfel încât se incarca doar datele relevante atunci când este folosita. În plus, aplicații native pot profita de caracteristici integrate de dezvoltare ale platformei (cum ar fi standard de UIS), care pot în continuare performanța ajutor.

Aplicații mobile bazate pe Web funcționează în același mod ca și un site web ,de exemplu: vă încărcați conținutul unei pagini atunci când un utilizator cere vizualizarea paginii. Putem folosi cache-ul browser-ului pentru a face lucrurile mai rapide pentru utilizatorii cu cache amorsate, dar graficele, imaginile, script-urile și datele sunt încărcate de pe serverul de web, spre deosebire de a fi pe dispozitivul clientului (în cazul celor mai multe aplicații native).

Design-ul este un aspect deosbit de important in succesul unei aplicatii mobile, atat in modul in care utilizatorul interactioneaza cu aceasta (foarte usor, comod,intuitiv), cat si in modul in care este organizata informatia, viteza de acces la informatie si terminand cu partea pur vizuala.

In general, design-ul trebuie sa fie cat mai simplu si sa ofere orientare rapida, facila. Nu consider ca exista elemente obligatorii de design. Daca, spre exemplu, o aplicatie este destinata unui search, importanta este usurinta de cauta, poate chiar prin metode variate de cautare si afisare a rezutatelor. Iar daca, de exemplu, aplicatia ofera o forma de entertainment, designul este parte din respectiva forma de entertainment si poate fi oricat de complex posibil, atata timp cat aplicatia poate fi rulata de suporturile mobile.

Contrastul si rezolutia ridicata a elementelor grafice asigura calitatea aplicatiei. Butoanele din aplicatie trebuie sa fie suficient de mari si la fel si distantate intre ele, pentru a permite o interactiune fara erori. Pentru ca spatiul pe ecran este restrans, acesta trebuie folosit foarte judicios, mereu avand in vedere interactiunea cu utilizatorul

Pretul pentru dezvoltarea unei aplicatii variaza, de obicei, intre 2.000 de euro si 10.000 de euro, in functie de complexitate, numarul de platforme, numarul de utilizatori vizati, design, disponibilitate si calitate (viteza cu care se incarca, interactiunea, calitatea graficii si frecventa erorilor).

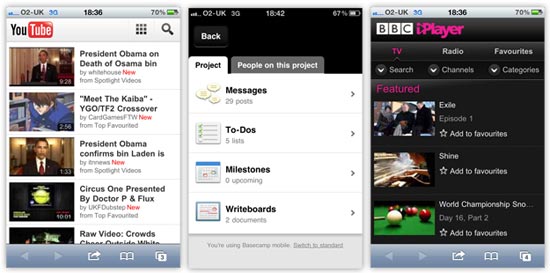
Cele mai importante criterii pe care o companie ar trebui sa stabileasca lansarea unei aplicatii pentru telefoanele mobile sunt:obiectivele si utilitatea sau beneficiile, atat pentru clienti, cat si pentru companie.

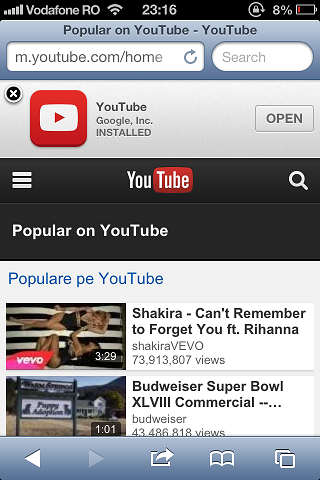
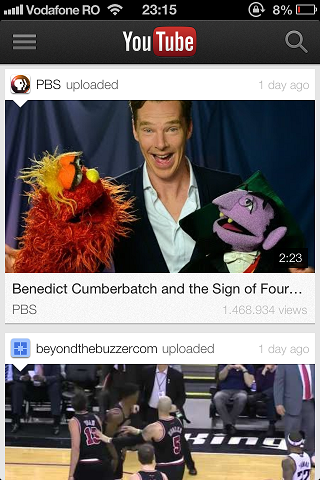
**2. Diferente intre aplicatiile Web si aplicatiile native pentru dispozitivele mobile**

In zilele noastre, din ce in ce mai multe site-uri web vin cu o versiune de mobil, care este, în general, o versiune ușor scalata in jos a unui site web principal optimizat pentru ecrane mici (a se vedea Amazon, Twitter, iar pagina de start BBC, de exemplu). O versiune mobilă a unui site are de multe ori conținutul unui site și iesiri astfel încât informațiile pot fi consumate mai ușor pe un dispozitiv mobil.

Aplicațiile web mobile sunt similare într-un anumit sens, pentru că se doreste ca acelasi continut sa fie disponibil cat mai ușor posibil.

O aplicatie web mobil diferă de o versiune mobilă a unui site web din cauza aceasta fiind adaptate pentru platforma mobilă. UI va fi, în general, mai personalizata, și va include mai multe dispozitive-interfețe utilizator centric-mobile (mobile-device-centric user interfaces), așa cum se poate vedea din exemplele de mai jos.





**Avantajele aplicatiilor native :**

* Posibilitatea de a accesa diferite componente hardware (bluetooth, senzori de mișcare, leduri de notificare, camera și altele);
* Timp mai scurt de răspuns pentru unele operații;
* Programatorul are acces la ultimile API-uri apărute pe platformă încă de la versiunile beta;
* Aplicația are o vizibilitate mai mare pe platforma unde este lansată datorită faptului că este distribuită prin piețele de aplicații folosite de majoritatea utilizatoriilor de dispozitive mobile;
* Oferă utilizatorului o experiență de utilizare superioară fapt ce poate duce la creșterea utilizării aplicației;
* Unelte de dezvoltare gratuite;
* Mediile de dezvoltare vin de obicei cu unelte bune de testare a codului dar si de analiză a aplicației din punct de vedere a consumului de memorie si timpilor de răspuns.

Aplicațiile native pot aduce utilizatori noi în timp ce-i păstrează pe cei care deja le folosesc. Majoritatea brand-urilor mari din lumea aplicațiilor sociale (facebook, twitter, linkedin, etc.) și-au dat seama de puterea aplicațiilor native de a atrage utilizatori și au ales să meargă pe această nișă.

**Dezavantajele aplicatiilor native :**

* Spre deosebire de aplicațiile web, distribuția aplicațiilor native este adesea limitată de forțarea acestora de a putea fi distribuite doar prin piețele de aplicații care pot impune diferite bariere (ex. Prin unele piețe de aplicații nu se pot vinde aplicații din anumite țări),
* Dacă se urmărește lansarea produsului pe mai multe platforme, costurile cresc semnificativ, fiind nevoie de rescrierea codului pentru fiecare platformă în parte,
* De regulă, specializarea programatorului pe o platformă poate dura ani de zile fapt ce duce și la o limitare a resurselor umane.

**Avantajele aplicatiilor web :**

* Timp de dezvoltare mic pentru o acoperire largă de sisteme de operare;
* Învățarea mai rapidă a tehnologiilor de dezvoltare, fiind vorba în principiu de tehnologi WEB;
* Aplicația are o vizibilitate mai mare pe platforma unde este lansată datorită faptului că este distribuită prin piețele de aplicații folosite de majoritatea utilizatoriilor de dispozitive mobile;
* Unelte de dezvoltare gratuite.

**Dezavantajele aplicatiilor web :**

* Dependență față de dezvoltatorii uneltelor de lucru, fapt ce poate întârzia lansarea aplicației care să funcționeze pe o versiune nouă a unui sistem de operare sau întârzieri în repararea problemelor tehnice ce pot apărea mai ales la versiuni noi ale sistemelor de operare pentru mediile mobile;
* Performanță mai scăzută în unele locuri
* Reputație scăzută în rândul utilizatorilor fideli a unei anumite platforme mobile;
* Pierdere de timp pe repararea problemelor găsite în urma faptului că nu pe toate platformele aplicația o să ruleze la fel de bine.

**3.Concepte si tenhnici pentru a construi aplicatii web pentru dispozitive mobile**

*3.1. Pastrarea fisierelor la dimensiuni mici*  
Cand nu aveti acoperire suficienta de WiFi, conexiunile vor vi de obicei incite sau inexistente. Pentru orice produs pentru dispozitivele mobile, care are probleme legate de conexiunea la internet, trebuie ca dimensiunea fisierelor pe care utilizatorul sa o descarce sa fie redusa cat de mult cu putinta. Experientele mai rapide sunt, in general, mai satisfacatoare. Putem accelera lucrurile prin mentinerea fisierelor la dimensiuni mici si prin reducerea numarului acestora.   
  
     Trebuie sa retinem:   
-viteze mici de descarcare = timp de incarcare mare   
-fisiere mici si putine = timp de incarcare mici   
  
     Este de datoria noastra sa punem in balanta egalitatiile de mai sus pentru a crea un produs ce arata si functioneaza bine.   
     Aplicatiile mai mici se vor descarca si executa mai rapid si mai fiabil pe un dispozitiv mobil, decat aplicatiile mari.   
     Procesati fisierele HTML, JavaScript si CSS pentru a elimina spatiul gol si a le minimiza inainte de livrare.   
  
  
  
*3.2. Confruntarea cu performanta imaginilor*   
     Vrem sa incercam sa scapam de cat mai multe imagini cu putinta. Imaginile pe care le pastram vrem sa fie de dimensiuni cat mai mici.   
     In cazul in care imaginile sunt o necesitate pentru anumite parti din designul aplicatiei web pentru mobile, atunci putem lua o serie de masuri pentru a scapa de "grasime" aceste fisiere.   
     De exemplu putem folosi Adobe Fireworks pentru PNGs transparente. Fireworks are un as in maneca fata de restul editoarelor de imagini. Daca vrem sa folosim PNGs cu transparenta alfa, atunci din pacate, Adobe Photoshop ne va permite sa exportam doar 24-bit PNGs, care sunt mai robuste decat cele de 8-bit. Folosing Fireworks, putem salva in 8-bit PNG, chiar si imaginile care necesita transparenta alfa. Astfel scapam de aproximativ 20% din marimea fisierului fara a-I afecta calitatea.

**4.Redirectionari.Avantaje CSS3**

Browserele de web pentru mobile sunt destul de avansate in aceste zile. Dispozitivele Android folosesc o versiune mobila a Google Chrome, in timp ce Apple face acelasi lucru cu Apple Safari. Altele folosesc Opera, sau iti permit sa instalezi ce browser doresti. Deci vorbim despre unele browsere care stau destul de bine in materie de CSS3 si caracteristici support de HTML5.   
     CSS3 ne permite sa redam prin cod ccea ce anterior ar fi necesitat in prealabil o imagine. Putem folosi gradienti de culoare, sa trasam colturi rotunjite, sa cream umbre, sa aplicam backgrounduri multiple elementelor HTML si multe altele, toate ajutand la imbunatatirea performantei si la reducerea timpului de developare al aplicatiei.   
     Si totusi trebuie sa fim atenti la performantele redate de CSS3. Oricat de brilliant ar fi CSS3, anumite proprietati pot sa incetineasca o pagina web. Browserele bazate pe WebKit, de exemplu, par sa aiba probleme cu umbrele in particular, deci trebuie sa fim atenti la numarul de astfel de elemente pe care le aplicam. 

*4.1. Evitarea redirectionarilor*   
     Regula 11 din "High Performance Web Sites" este aceea de a evita redirectionarile. O redirectionare fnctioneaza similar cu redirectionarile apelurilor telefonice. In loc de a se intoarce la continutul actual, serverul returneaza un raspuns cu un antet de locatie, indicand ca URL-ul pe care browserul incerca sa il contacteze este cel asteptat. Acest lucru poate lua ceva timp, deoarece o redirectinare duce la alta si tot asa. Fiecare redirectionare aduce cu ea cererea completa si tot latency-ul sau. Pe un desktop aceasta consecinta poate sa fie chiar neobservabila, dar pe un dispozitiv mobil o redirectionare poate fi dureros de lenta.   
     Redirectionarea cererilor (folosid HTTP 3xx status sau HTML meta refresh) este de obicei folosita pentru a schimba informatii intre servere (de exemplu autentificarea unui account). Intarzierea inregistrata de redirectionari este mult mai mare in retelele pentru dispozitivele mobile si astfel numarul redirectionarilor trebuie tinut la minimum.   
     In cazul in care sunt necesare mai mult de 2 redirectionari, luati in considerare utilizarea unei pagini interstitialepentru a comunica utilizatorului ca aplicatia este inca in lucru.   
     Multe site-uri si aplicatii web au adoptat conventia de a folosi "www.example.com" pentru site-urile accesate de pe desktop si "m.example.com" pentru site-urile accesate de pe dispozitive mobile. Greseala lor a fost ca au crezut ca utilizatorii ar introduce numele de domeniu complet pentru site-ul bazat pe versiunea pe care o doresc. In realitate, utilizatorii tind sa tasteze doar numele gazdei, precum "example.com", ceea ce inseamna ca serverul trebuie sa isi dea seama ce sa faca cu aceasta cerere. Frecvent, primul pas este de a redirectiona la versiunea "www"de domeniu, care este serverul pe care ruleaza aplicatia web. Apoi aplicatia se uita la sirul agent al utilizatorului si determina daca dispozitivul este unul mobil, fapt care conduce la o a doua redirectionare catre versiunea "m" a domeniului.   
     Bing face acest lucru, dar cu niste rezultate foarte proaste.   
  
  
  
*4.2. Aplicarea datelor*   
     Majoritatea aplicatiilor au necesitatea de a stoca date de diferite forme, atat cu continut intrinsec (de exemplu e-mail-urile unei aplicatii de e-mail, evenimentele calendaristice ale unei aplicatii de calendar etc.), cat si setari suplimentare de personalizare (de exemplu tema preferata, o vedere implicita etc.).   
     Cateva tehnici adecvate de utilizare pentru a gestiona datele aplicatiei web:   
  
  
  
     *4.2.1. Folosirea cumpatata de Cookies*  
     Cookie-urile sunt un mijloc comun si eficient pentru a stoca conditiile clientului. Ele sunt adecvate pentru date de personalizare simple si sunt cel mai adesea utilizate pentru a stoca un token care reprezinta identitatea utilizatorului, in scopul de a-i permite o autentificare automata.   
     Cu toate acestea, informatiile stocate in cookie-uri sunt trimise la server pentru fiecare cerere si deci utilizarea lor pentru cantitati excesive de date pot sa aiba un efect negativ asupra performantei, in special pentru un dispozitiv mobil.   
     De asemenea, in contextul dispozitivelor mobile, nu se poate baza pe suportul oferit de cookie-uri, deoarece acestea pot sa fie dezactivate fie de configuratia dispozitivului, fie de reteaua de telefonie mobila. Din acest motiv, aplicatiile trebuie sa incerce sa ramana functionale chiar daca cookie-urile nu sunt disponible.   
  
  
  
 *4.2.2. Reproducerea datelor locale catre un server daca este necesar*   
     Daca un API de stocare din partea clientului este utilizat, datele din el nu sunt vizibile pentru alte dispozitive ale utilizatorului. In timp ce acest lucru este necesar pentru unele tipuri de date (de exemplu preferintele si conditiile compatibile doar cu un anumit dispozitiv), este de multe ori necesar sa trimitem aceste date inapoi la un server, in scopul de a oferi o imagine consistenta pe toate dispozitivele (de exemplu, intre diversele aplicatii pentru mobile sau pentru desktop-uri) si face posibila recuperarea datelor in cazul ni care dispozitivul este pierdut sau deteriorate. Ca regula, datele care trebuie partajate cu alte dispozitive sau recuperate in cazul pierderii sau deteriorarii dispozitivului, trebuie replicate inapoi la server cat mai repede cu putinta.   
     Tehnologiile care ofera API de stocare din partea clientului ofera facilitati pentru a detecta conectivitatea la reteaua curenta. De exemplu, HTML5 ofera o proprietate asupra obiectului de navigare (navigator.onLine) pentru a indica daca clientul este online in momentul de fata si expediaza doua evenimente pe obiectul de fereastra pentru a indica o schimbare de stare a retelei (online sau offline).   
     Cu toate acestea, aceste API-uri trebuie folosite cu atentie. Chiar daca browserul reda o stare online, pe o retea intermitenta nu avem nici o garantie ca o conexiune ulterioara va reusi. Abordarea eficienta este de a esua gratios in cazul unei erori de conexiune, sa inmagazinam datele nesalvate intr-o coada care nu are inregistrate modificari si sa ne stabilim un timp pentru a incerca mai tarziu.   
  
 *4.3. Securitate si intimiate*  
     Acest lucru consta in utilizarea informatiilor de incredere si de a proteja toate informatiile personale de identificare. Aplicatiile web pentru dispozitive mobile sunt supuse acelorasi riscuri si considerente de securitate ca si aplicatiile web pentru desktop, si ca atare, majoritatea sfaturilor de securitate legate de desktop-uri se aplica si la dispozitivele mobile.   
  
  
  
*4.4. Controlul si constientizarea utilizatorului*     Permiteti utilizatorului sa controleze comportamentele aplicatiei care altfel nu ar fi evidente, cum ar fi accesul la retea si accesul la datele dispozitivului.   
  De exemplu:   
-imagini, muzica si videoclipuri;   
-contacte si calendar;   
-istoricul convorbirilor;   
-date de sistem (de exemplu baterie, locatie, semnal);   
-inregistrari media (de exemplu inregistrari audio/video, fotografierea)   
-contextul dispozitivului (de exemplu locatie, conectivitate, setari de profil) 

*4.4.1. Asigurarea ca utilizatorul este informat cu privire la utilizarea informatiilor personale si de dispozitiv*     Utilizatorul trebuie sa fie informat cu privire la tipurile de informatii care vor fi utilizate de catre aplicatie si daca/cum vor fi schimbate cu serverul.   
     Aceste notificari ar trebui sa fie furnizate cand utilizatorul acceseaza pentru prima data aplicatia sau la primul acces catre informatiile utilizatorului. Acest lucru ar trebui sa ii ofere utilizatorului suficiente informatii pentru a decide doresc sau nu sa permita accesul aplicatiei catre datele lor.   
     In multe cazurim utilizarea de API-uri care ofera acces la informatii personale sau de dispozitiv produce o casuta de dialog de confirmare nativa pentru a fi prezentata utilizatorului. In acest caz, cererea nu trebuie sa forteze utilizatorul pentru a confirma din nou nivelul aplicatiei, dar trebuie sa faca clar in UI ca datele afisate au fost accesate din dispozitiv.   
     In cazul in care utilizatorul refuza o solicitare pentru a permite accesul la informatii personale sau de dispozitiv, aplicatia trebuie sa isi revina gratios. De exemplu in cazul in care o cerere catre un dispozitiv API esueaza, nu incercati din nou in mod automat daca acest lucru va duce la a-i fi prezentat in mod repetitiv utilizatorului casute de dialog de confirmare native.   
  
  
  
   *4.4.2. Activarea autentificarii automate*     Daca o aplicatie necesita identificarea utilizatorului, este normal a solicita date de autentificare (nume de utilizator si parola) si sa ofere posibilitatea de a se autentifica automat la urmatoarea sesiune de utilizare. Acest lucru este in special important pentru un dispozitiv mobil, unde datele de intrare sunt mult mai greu de introdus decat de pe un desktop.   
     Retineti ca, in cazul in care avem activata otpiunea de autentificare automata, de asemnea trebuie sa avem si o optiune pentru delogare.   
     Astfel, datele de autentificare pot fi stocate intr-un cookie sau intr-un local storage. Cu toate acestea, este important sa nu stocam informatii necriptate despre parola din moment ce acest lucru este nesigur. De obicei, un hashed token sigur care, daca este necesar, poate fi revocat de la server, este stocat la nivel local pentru a permite autentificarea automata. 

*4.4.3. Folosirea conservativa a resurselor*  
     Cel mai eficient mod de a asigura ca aplicatiile ruleaza fara probleme si cu latency scazut este de a reduce la minim folosirea memoriei dispozitivului, puterii procesorului si latimii de banda a retelei, care sunt mult mai limitate pe un dispozitiv mobil decat pe un desktop. 

*4.4.3.1. Optimizarea cererilor de retea*     Stabilirea conexiunilor necesare pentru a completa o cerere HTTP poate dura in mod semnificativ mult mai mult pe o retea pentru dispozitive mobile decat pe o retea fixa. Chiar daca latimea de banda este de obicei mai restrictionata la o retea pentru dispozitive mobile este inca de preferat de a face cereri mai putine, dar mai mari.   
     Luati in considerare urmatoarele posibilitati la proiectarea unei aplicatii:   
-cereri de dozare: deoarece o singura cerere pentru mai multe date este mult mai probabil sa ofere o experienta placuta decat mai multe cereri mai mici, ori de cate ori este posibil, dozati cererile multiple la nivelul aplicatiei;   
-strangulati cererile de prioritate scazuta: in unele aplicatii anumite cereri pot sa fie mai putin importante decat altele (de exemplu cererile de logare); strangulati cererile de prioritate scazuta pentru a asigura ca ele nu blocheaza reteaua si pentru a preveni mai multe cereri critice de la a fi deservite rapid;   
-retrageti-va in perioadele de inactivitate: in cazul in care aplicatia intreaba pentru actualizari, aceasta ar trebui sa monitorizeze activitatea utilizatorului si sa intrebe mai rar in timpul unor perioade lungi de inactivitate;   
-contextul dispozitivului: daca dispozitivul permite, utilizati constientizarea conexiunii curente ( de exemplu WiFi) pentru a selecta un nivel adecvat de interactiune.   
  
  
 *4.4.3.2. Minimizarea resurselor externe*   
     O aplicatie web necesita de obicei o serie de resurse (stylesheets, script-uri, imagini etc.), fiecare din acestea avand nevoie de o cerere HTTP. Cererile HTTP sunt deosebit de scumpe pe o retea pentru dispozitive mobile si astfel cererile mai putine, dar mai mari, ar trebui favorizate fata de cererile multe, dar mici.   
     Combinati toate stylesheets-urile intr-o singura resursa si toata script-urile intr-o singura resursa. In cazul in care mai multe script-uri si stylesheets sunt necesare ca parte a procesului de creatie, incercati sa le aranjati astfel incat sa fie unite inaite ca pagina sa fie livrata. 

*4.4.3.3. Includerea imaginilor de fundal in linile codului stylesheets-urilor CSS*  
     Efectele vizuale (de exemplu imaginile de fundal si gradientii de culoare) sunt adesea folosite pentru a imbunatati aspectul unei aplicatii. Acestea pot fi incluse in CSS ca siruri codificate base64 pentru a evita o cerere suplimentara de HTTP.   
     Retineti ca codarea base64 adauga in jur de 10% la dimensiunea imaginii dupa comprimarea de tip gzip si acest cost suplimentar ar trebui pus in balanta cu beneficiile cererilor mai putine.   
     Imaginile de fundal pot fi codificate cu ajutorul schemelor de date URL: url(data:image/png;base64,[data]).   
  
  
  
*4.4.3.4. Informatii cookie inutile nu trebuiesc trimise*   
     Resursele statice nu au nevoie de informatii cookie si astfel performanta poate fi imbunatatita prin a le servi de la o cale sau sub-domeniu pentru care cookie-urile aplicatiei nu mai au rost.   
     Utilizati un alt domeniu, sub-domeniu sau nume de cale pentru resursele statice la aplicatia principala, si sa limiteze calea valida a cookie-urilor, astfel incat acestea nu vor fi schimbate cand nu vor mai fi necesare.   
  
  
  
*4.4.3.5. Pastrarea dimensiunilor DOM la un nivel rezonabil*   
     Dimensiunea in memorie a DOM(Document Objet Model) poate fi limitata pe dispozitivele mobile. Paginile mari sau complexe pot depasi aceasta limita si pot cauza erori imprevizibile.   
     Limitati cantitatea de informatii in DOM folosind paginarea sau alte tehnici adecvate.   
  
  
  
*4.4.4. Experienta utilizatorului*   
     Avand in vedere complexitatea de a interactiona cu o aplicatie pe un dispozitiv mobil, o atentie deosebita trebuie acordata pentru experienta utilizatorului. Experienta utilizatorului este influentata de un numar de factori, printre care: latency, modalitatea de interactiune, consistenta datelor etc.   
  
  
  
*4.4.4.1.Minimizarea latency-ul perceput*   
     Scaderea latency-ului perceput este un factor important in imbunatatirea gradului de utilizare in general a unei aplicatii web pentru dispozitivele mobile.   
     Cateva tehinici pentru a scadea latency-ul perceput:   
-permite Incremental Rendering: pozitionati JavaScript-ul in partea de jos a paginii si configurati pagina astfel incat orice informatie utila care ar putea fi disponibila este vizibila in timp ce continutul principal al aplicatiei inca se incarca;   
-pastrati utilizatorul informat asupra activitatii: folositi bare rotative progresiv pentru a mentine utilizatorul informat in timpul accesului la dispozitivul API si la retea, astfel incat acestea sa nu creada ca aplicatia este oprita;   
-evitati reincarcarile paginilor: pentru a reflecta schimbarile de stare sau pentru a arata diferite pareri/optiuni intr-o aplicatie, actualizati paginile in mod dinamic (prin manipularea DOM) mai degraba decat prin reincarcarea lor;   
-preincarcati urmatoarele posibile vizualizari: preincarcati date pentru caile traversate frecvent in aplicatie, astfel incat sa poate fi afisat mai rapid atunci cand utilizatorul cere acest lucru.   
  
  
  
   *4.4.4.2. Pastrarea atentiei asupra actualizarii dinamice a paginilor*  
     Metoda de focalizare JavaScript poate fi folosita pentru a muta focalizarea catre acea parte de pagina care s-a schimbat. Cu toate acestea, daca neasteptat, aceasta poate confunda sau irita utilizatorul, in special daca intoarcerea la focalizarea precedenta nu este usoara.   
     Folositi metoda de focalizare numai daca este esentiala pentru utilizarea aplicatiei si nu inhiba interactiunea/controlul utilizatorului.   
  
  
  
      *4.4.4.3. Utilizarea fragmentelor de ID-uri pentru a conduce la vizualizarea aplicatiei*   
     Aplicatiile web pot schimba vizualizarea paginii fara o reincarcare completa a paginii prin afisarea si ascunderea sectiunilor de continut. Oricum, asta inseamna ca butonul al browserului nu functioneaza in mod implicit si nu este posibil sa ne legam direct la vizualizarile specifice in cadrul unei aplicatii.   
     Gradul de utilizare este sporit prin aplicarea acestor caracteristici:   
-activarea deep link-urilor (de exemplu la continutul unui e-mal specific) inseamna ca utilizatorul poate marcheze aceasta vizualizare si sa se reintoarca la ea mai repede;   
-activarea istoricului browserului ofera o metoda naturala de a naviga vizualizarile aplicatiei care este sustinuta nativ de browser.   
     Acest lucru se poate realiza prin atribuirea unui URL cu un fragment identificator distinctiv. Utilizati JavaScript pentru a interoga locatia browserului in scopul de a determina vizualizarea care trebuie afisata.   
     Retineti ca afisarea si ascunderea continutului in acest mod poate avea efecte negative asupra accesibilitatii daca nu este manipulat cu grija.   
  
  
  
  *4.4.4.4. Folosirea Meta Viewport Element pentru a identifica dimensiunea dorita a ecranului*  
     Anumite categorii de brwosere incearca sa afiseze pagini de desktop pe un ecran mic prin zoom atuomat pe ecran. Acest lucru poate fi problematic pentru aplicatiile care au fost deja optimizate pentru un ecran mic. Viewport-ul meta tag spune aparatului la ce scala sa redea pagina.   
     O setare viewport tipica arata de genul:   
< meta name="viewport" content="width=device-width, initial-scale=1.0" />  
si este introdusa in elementul al documentului HTML. Aceasta setare informeaza browserul sa redea mereu pagina la 100% (de exemplu fara nici o scalare a browserului) si este adecvata pentru pagini special proiectate pentru ecrane cu o dimensiune tinta. 

*4.4.5. Manipularea variatiei in contextul de livrare*   
     Variatiile in contextul de livrare (cum ar fi diferite capacitati ale dispozitivului) sunt o caracteristica importanta a web-ului pe dispozitivele mobile. Aplicatiile web ar trebui sa se adapteze la proprietati cunoscute ale contextului de livrare prin ajustarea continutului, navigarii sau fluxului de pe pagina, cu scopul de a oferi experenta buna utilizatorului pe o gama cat mai larga de dispozitive.   
  
  
  
*4.4.5.1. Folosirea capacitatii de detectare din partea clientului unde este necesar*   
     In cazul in care nu este posibil sa se determine anumite proprietati ale contextului de livrare de la server, aceste informatii pot fi disponibile la client. Odata obtinut la client, informatia poate fi utilizata direct pentru a adapta prezentarea sau se poate folosi la cerere alternativa, adaptata la continutul de la server.   
     Exista cateva solutii pe partea clientului disponibile dezvoltatorului:   
-JavaScript: aceasta este cea mai comuna solutie; un script determina proprietatile dispozitivului/browserului si manipuleaza continutul si comportamentul aplicatiei in consecinta; aceasta se poate realiza in doua moduri:   
1. Prin incapsularea comportamentelor diferite in controlul logic al aplicatiei. De obicei informatiile contextului de livrare sunt adunate la inceputul sesiunii, cu toate acestea informatiile dinamice ar trebui sa fie actualizate si in timpul sesiunii.   
2. Prin cedarea informatiilor culese inapoi la server si cererea de continut alternativ ( de exemplu fie prin adaugarea dinamica a unui nou element de < script > la DOM sau cu o cerere XHR).   
-tipuri de CSS media: acestea ne permit ca stylesheets-uri diferite sa fie asociate cu diferite tipuri de media ( de exemplu de imprimare, de ecran, de telefonie mobila) si sunt folosite in mod traditional pentru a redirectiona continut catre aceste destinatii. Deoarece lista de tipuri de media recunoscute este limitata, cu toate acestea, si dispozitivele pot sa reactioneze negativ in interpretarea lor a tipurilor de media, aceasta nu este o tehnologie utila in acest context;   
-interogari CSS media: Interogarile media sunt o extensie a "tipurilor de media" paradigma care permite dezvoltatorilor sa aplice aceste reguli de stil specifice, bazate pe pe caracterisicile de afisare a dispozitivului (de exemplu de latimea ecranului, de orientare, de rezolutie). Aceasta specificatie ofera un mod util de a modifica aspectul paginii intr-un mod mai usor de intretinut decat cel posibil cu script. 

*4.4.5.2. Sustinerea unei variante non-JavaScript daca este cazul*  
     Sunt aplicatii care nu sunt suportabile de catre toate browserele, asadar se recomanda si folosirea unei alternative.

**5.Securitate si autentificare**

Securitatea este capacitatea de a conserva toate resursele la parametrii definiti de dezvoltator si de utilizator.

Acest lucru consta in utilizarea informatiilor de incredere si de a proteja toate informatiile personale de identificare. Aplicatiile web pentru dispozitive mobile sunt supuse acelorasi riscuri si considerente de securitate ca si aplicatiile web pentru desktop, si ca atare, majoritatea sfaturilor de securitate legate de desktop-uri se aplica si la dispozitivele mobile.

Aplicatiile in care utilizatorii trebuie sa se logheze folosind datele personale au un nivel de securitate ridicat. Datele furnizate de utilizator trebuie sa circule pe un canal securizat si daca este posibil ca acestea sa fie criptate pentru a nu fi obtinute si folosite de personae neautorizate. Aceste aplicatii sunt cele folosite pentru logarea unui cont bancar current sau furnizarea datelor de pe cardul bancar pentru cumpararea anumitor lucruri prin indermediul telefonului mobil.

*Activati autentificare automata:*  
Daca o aplicatie necesita identificarea utilizatorului, este normal a solicita date de autentificare (nume de utilizator si parola) si sa ofere posibilitatea de a se autentifica automat la urmatoarea sesiune de utilizare. Acest lucru este in special important pentru un dispozitiv mobil, unde datele de intrare sunt mult mai greu de introdus decat de pe un desktop.   
In cazul in care avem activata otpiunea de autentificare automata, de asemnea trebuie sa avem si o optiune pentru delogare.   
Astfel, datele de autentificare pot fi stocate intr-un cookie sau intr-un local storage. Cu toate acestea, este important sa nu stocam informatii necriptate despre parola din moment ce acest lucru este nesigur. De obicei, un hashed token sigur care, daca este necesar, poate fi revocat de la server, este stocat la nivel local pentru a permite autentificarea automata.

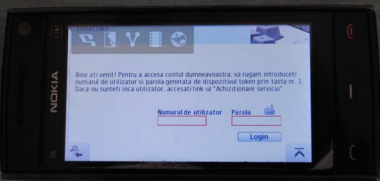
Pentru logare la un cont bancar este nevoie de numarul utilizatorului si parola acestuia. Aceste informatii trebuiesc criptate pe canalul de comunicare dintre dispozitivul mobil si baza de date a bancii. In cazul in care acest canal nu este asigurat impotriva atacurilor neautorizare, lucru care sigur nu va face bine pentru utilizatorul care s-a logat.

Securitatea este principala caracteristică de calitate a unui sistem bancar de colaborare. Pentru asigurarea unui nivel ridicat de securitate interioară sistem de informații, băncile au angajat

hackeri pentru a testa și de a descoperi vulnerabilitățile de fiecare noua aplicare care va fi lansata în producție, chiar dacă este o aplicație mobilă sau una normala.

Aplicațiile bancare sunt expuse la mai multe atacuri și este mai puțin costisitor de a plăti hackeri pentru a descoperi vulnerabilitățile decât la lansare într-un mediu real, o aplicație care nu este testat suficient. Pierderea de banca va fi mai mare în al doilea caz.

Tehnologiile și aplicațiile mobile oferă o mulțime de noi oportunități pentru întreprinderi, ele, de asemenea fiind intr-o continua dezvoltare și implementare noi provocări.



Inregistrarea online a unui cont bancar

Un alt exemplu este accesarea site-ului Amazon, care pentru aceasta aplicatie este necesara de asemenea criptarea datelor furnizate de catre utilizator la cumpararea de carti. In cadrul acestor aplicatii cea mai importanta caracteristica este securitatea datelor folosite.



Accesarea site-ului Amazon Accesarea site-ului Facebook

Aplicatiile de comunicare in retea sunt aplicatii cu logare folosind datele personale prin intermediul carora utilizatorii comunica cu personae aflate la distanta cu ajutorul telefonului mobil. Facebook este o astfel de aplicatie care face legatura dintre telefonul mobil si retele de socializare folosite sau server-ele de mail. Si in cadrul acestor aplicatii, securitatea datelor trebuie sa prezinte un nivel ridicat deoarece se folosesc informatii confidentiale la care nu trebuie sa aiba acces oricine.

Aplicatiile economice pentru dispozitivele mobile sunt realizate pentru a adduce aminte lucruri mai aproape de utilizatori. Astfel acestia pot face cumparaturile direct de pe telefonul mobil.

Noul nivel in securitatea organizatiilor este securitatea informatiilor de pe telefoanele mobile, deoarece utilizarea dispozitivelor mobile devine tot mai frecventa, cu date in mod regulat in masura sa faca trecerea de la sistemele desktop la dispozitibele de buzunar, cum ar fi telefoanele inteligente si PDA-uri. Dar utilizatorii prezinta si pericolul la pierdere al acestor despozitive mobile si, de asemenea al informatiilor stocate pe acestea.

**Protecţia împotriva ameninţărilor** 

Multe organizaţii cred că PDA-urile, smart phone-urile şi alte echipamente mobile, reprezintă un risc mic pentru business, însă dispozitivele mobile actuale au o putere, o capacitate de stocare şi o conectivitate ce le permit să rivalizeze cu laptopurile (şi chiar să le surclaseze).

Protejarea PDA-urilor şi smart phone-urilor este similară, în mod conceptual, cu protejarea laptop-urilor, însă este implementată mult mai diferit din cauza variaţiilor în OS-uri, implementarea securităţii, metodele I/O şi comportamentul utilizatorului. O securitate mobilă poate necesita parole şi poate implementa reguli complexe. Cu toate acestea, este dificil să introduci o parolă lungă şi complexă pe un PDA. Mai bine luaţi în considerare metode facile de intrare, cum ar fi autentificarea pe bază de imagine sau amprente digitale. Cum autentificarea în mod repetat este nepractică, gândiţi-vă şi la excepţiile pe care le-aţi putea face în privinţa utilizării. Encriptarea datelor stocate este importantă pentru dispozitivele mobile, însă opiniile în privinţa granularităţii şi a metodei variază. Unele companii au o politică corporativă de encriptare a tuturor dispozitivelor mobile, în timp ce altele encriptează doar dispozitivele ce stochează informaţii sensibile.   
Securizarea datelor în tranzit poate fi realizată cu aplicaţii sigure. Aceste metode nu sunt în mod reciproc exclusive. Ele servesc unor scopuri diferite. Alegeţi cu grijă; metodele specifice dispozitivelor sunt mai puţin extensibile, iar metodele specifice reţelei împiedică roaming-ul.   
Erupţia recentă de viermi şi Troiani pentru dispozitive mobile a sporit interesul pentru firewall-uri şi protecţii împotriva malware-ului. Firewall-urile PDA-urilor pot fi programe de tip “stand-alone” sau pot fi integrate cu alte aplicaţii. Mulţi vendori de anti-virus, printre care şi Kaspersky Labs, Symantec, McAfee, F-Secure, Trend Micro, vând scannere pentru PDA/smart phone, tocmai pentru a micşora ameninţările cu specific OS.   
Autorizarea este o zonă în care programele de securitate mobilă pavează calea. Listele negre pot opri utilizatorii să instaleze programe ce supun datele corporative la risc; listele albe pot verifica prezenţa unui program necesar, înainte ca dispozitivul respectiv să poată accesa o reţea sau un desktop. Multe produse pot impune reguli de utilizare pentru stocare, camere foto încorporate, telefoane şi interfeţe wireless.   
Blocarea utilizării neautorizate a reţelei poate fi realizată prin mai multe tehnici: agenţi Host-resident pot împiedica PDA-urile să se sincronizeze cu desktop-urile, gateway-urile şi portalurile VPN pot respinge accesul utilizatorilor neautorizaţi, iar gateway-urile care utilizează NAC ori TNC pot bloca “clienţii” ce nu trec de verificările de integritate sau îi pot băga în carantină pentru remediere.

Aplicațiile web au cunoscut o dezvoltare uluitoare de-a lungul ultimilor ani. Odată cu intrarea în era WEB 2.0 și dezvoltarea cloud computing-ului o mare parte din activitatea internauților s-a mutat în mediul web. Dezvoltarea spectaculoasă a aplicațiilor web a fost posibilă datorită inovațiilor tehnologice în domeniu, specifice WEB 2.0, care au transformat simplele pagini web în care erau afișate informații statice, în pagini dinamice, interactive ce permit o interacțiune ridicată a utilizatorului cu aplicația. Dezvoltarea uluitoare a web-ului a creat premisa apariției vulnerabilităților specifice oricărui produs tehnologic. Numărul acestora este în continuă creștere deși cele mai populare atacuri se bazează pe vulnerabilități identificate pentru prima dată acum câțiva ani buni.

Securitatea Web trebuia sa ia in consideratie:

1. clientul · interactiunea cu utilizatorul

· date personale stocate: cookie-uri, date off-line, cache,…

· transferurile asincrone – Ajax/Comet ori WebSocket-uri

· existenta plugin-urilor/extensiilor suspecte

2. datele in transit · securitatea retelei (cu/fara fir)

· schimbul sigur de mesaje intre diverse entitati

· ne-repudierea datelor

3. serverul · securitatea serverului/serverelor Web

· securitatea aplicatiilor, framework-urilor, bibliotecilor

· disponibilitatea serviciilor.

5.1. Atacuri

Toate aplicațiile web aplică o anumită logică pentru a furniza o funcționalitate oarecare. A scrie cod într-un limbaj de programare presupune exprimarea și separarea unui proces complex în niște pași logici foarte simpli. “Traducerea” corectă a unui element funcțional important pentru utilizator într-o secvență de operațiuni care pot fi executate de un calculator, presupune deținerea unor anumite abilități. Același lucru efectuat într-un mod securizat este încă mai dificil. În cazul în care asupra unui proiect lucrează un număr mare de oameni, probabilitatea de apariție a greșelilor logice este considerabilă.

În majoritatea aplicațiilor web, în orice etapă funcțională totul se execută conform unei logici. Toate acestea la rîndul lor împreună formează o suprafață potențială de atac, care e prezentă întotdeauna, însă în majoritatea cazurilor e trecută cu vederea. Majoritatea instrumentelor de analiză a codului sursă sau de penetration testing se axează pe vulnerabilități “populare” ca CSRF, RFI, SQL Injection, XSS, deoarece exploatarea acestor tipuri de vulnerabilități pot fi șablonate și au fost studiate foarte mult.

5.1.1. SQL Injection

Majoritatea proiectelor web folosesc baze de date pentru a-şi stoca datele. Limbajul SQL este un standard internațional ce stabilește regulile și sintaxa comenzilor ce pot fi transmise unui sistem de gestiune a bazelor de date (SGBD), pentru manipularea datelor stocate. Implementarea acestuia poate să difere de la caz la caz în funcție de producător. În prezent cele mai populare SGBD-uri sunt MySQL, PostgreSQL, Microsoft SQL Server și Oracle.

O vulnerabilitate de tip SQL injection(injecţie cu cod sursă SQL) apare atunci când un atacator poate introduce orice date într-o interogare SQL transmisă unei baze de date sau când, prin injectarea sintaxei, logica declaraţiei este modificată în asemenea fel încât să execute o acţiune diferită. Injecţia SQL poate fi crucială pentru sistem dar, în ciuda pericolului pe care îl prezintă, este cea mai frecvent întâlnită vulnerabilitate.

Cea mai bună strategie de apărare împotriva injecţiilor SQL se bazează pe implementarea unor rutine puternice de validare a datelor de intrare, astfel încât atacatorul să nu poată introduce alte date decât cele necesare aplicației.

Între măsurile specifice care pot fi implementate la nivelul bazelor de date şi aplicaţiilor se regăsesc următoarele:

a) *Utilizarea de variabile bine definite şi de definiţii ale coloanelor din baza de date*

Stocarea şi manipularea numerelor (ID-uri de sesiune, coduri etc.) ca şi numere întregi sau ca alte tipuri numerice potrivite. String-urile (varchars) ar trebui să conţină doar caractere alfanumerice şi să respingă semnele de punctuaţie şi caracterele specifice sintaxei SQL.

b) *Atribuirea rezultatelor interogării unei variabile bine definite*

Dacă aplicaţia caută valori numerice, atunci atribuiţi rezultatul unui număr întreg, acest lucru împiedicându-i pe atacatori să extragă informaţii din baza de date. De exemplu nu ar trebui să fie posibilă obţinerea şi afişarea numelui unei coloane, dacă variabila ce urmează să fie afişată în browser nu acceptă decât numere întregi. Această tehnică restricţionează sever anumite atacuri.

c) *Limitarea lungimii datelor*

Toate şirurile de caractere ar trebui să se limiteze la o lungime potrivită scopului lor. Un nume de utilizator, de exemplu, nu este necesar să fie stocat şi manipulat într-o variabilă care utilizează 256 de caractere. Limitarea numărului de caractere, care poate fi introdus într-un câmp, poate împiedica în mod eficient succesul unei injecţii SQL, reducând lungimea şirului de caractere pe care atacatorul îl poate introduce în cod.

d) *Evitarea creării de interogări prin concatenarea de şiruri de caractere*

Creaţi o funcţie view sau o procedură care operează asupra variabilelor furnizate de aplicaţie. Concatenarea şirurilor de caractere, unde interogarea este formată direct din datele furnizate de utilizatori (de genul: “SELECT something FROM table WHERE” + variable a), este cea mai vulnerabilă la atacurile SQL Injection. Pe de altă parte, o funcţie view sau o procedură particularizată, generează de obicei o eroare dacă primeşte date de intrare incorecte, însă nu îi va permite unui atacator să manipuleze întreaga interogare.

*e)**Aplicarea separării datelor şi accesul pe baza de rol în interiorul bazei de date*

Aplicaţia ar trebui să folosească un cont care are privilegii de acces doar pentru tabelele necesare respectivei funcţii. Tabelele interne ale bazei de date, în special cele legate de managementul conturilor şi variabilele sistemului, nu ar trebui să fie accesibile.

5.1.2. Cross-site Scripting(XSS)

XSS este o tehnică de atac, folosită pentru a forţa o pagină web să afişeze un cod maliţios (scris de obicei în HTML, JavaScript, ActiveX sau Flash), pe care îl execută ulterior în browser-ul unui utilizator. Acest tip de atac nu are ca ţintă serverul site-ului web, acesta fiind doar o gazdă pentru, codul malware fiind executat direct în browser, deoarece adevărata ţintă a atacului este utilizatorul.

Hackerul va folosi site-ul doar pentru a efectua atacul şi, odată ce are control asupra browser-ului utilizatorului, îl va putea folosi pentru a-i fura diferite date: conturi bancare, conturi de utilizator, parole, furtul înregistrărilor din istoricul browser-ului etc.

Sunt mai multe modalităţi prin care un malware scris în JavaScript poate deveni rezident pe o pagină web:

- Proprietarul paginii web îl poate încărca intenţionat;

- Un atacator îl poate injecta în secțiunea publică a unui site profitând de anumite vulnerabilități ale acesteia (vulnerabilitate permanentă).

- Pagina web poate primi un deface folosind o vulnerabilitate a reţelei sau a straturilor sistemului de operare, iar parte din codul introdus să fie malware JavaScript:

- Victima poate accesa un link special pregătit (transmis prin mail sau alte metode) în spatele căruia se ascunde un XSS non-persistent sau bazat pe Document Object Model (DOM).

5.1.2.1. Metode de prevenire a atacurilor de tip Cross-site scripting (XSS)

Codarea datelor de intrare şi de ieşire au fiecare argumentele lor pozitive şi negative. Partea pozitivă a codificării datelor de intrare oferă un singur punct de acces, în timp ce codarea datelor de ieşire oferă posibilitatea de a face faţă tuturor utilizărilor textului şi poziţionarea acestuia în pagina. Părţile negative sunt că nici codarea datelor de intrare nu poate opri un atac XSS persistent odată ce a fost stocat, iar codarea datelor de ieşire nu poate opri alte forme de atac, cum ar fi injecţia cu cod SQL, deoarece intervine prea târziu.

Există un număr de soluţii de a vă proteja în calitate de client. Nişte idei simple sunt:

- alegerea unui browser securizat;

- folosirea unei maşini virtuale;

- accesarea doar a link-urilor cunoscute;

- grijă la ce informaţii divulgă utilizatorul.

5.1.3. Cross-site Request Forgery(CSRF)

Cross-site Request Forgery (CSRF sau XSRF) este o formă de atac asupra aplicaţiilor web care se foloseşte de relaţiile de încredere existente între aplicaţiile web şi utilizatorii autentificaţi prin a forţa acei utilizatori să facă tranzacţii sensibile în numele atacatorului. Această vulnerabilitate, deşi mai puţin cunoscută ca XSS, este mult mai periculoasă decât cross-site scripting, deoarece îşi are rădăcinile în natura lipsită de stare (stateless) ale specificaţiilor HTTP-ului, care cer ca un token de autentificare să fie trimis cu fiecare cerere a utilizatorului.

În mod obişnuit, vulnerabilităţile web apar ca urmare a unor greşeli făcute de dezvoltatorii paginilor web în timpul proiectării şi dezvoltării acestora sau de către administratori, în timpul utilizării acestora. Spre deosebire de restul, vulnerabilităţile de tip XSRF apar atunci când dezvoltatorii omit un mecanism de prevenire a XSRF din aplicaţia lor. Pentru atac un exemplu clasic este cel al unei aplicaţii bancare care le permite utilizatorilor să transfere fonduri dintr-un cont în altul folosind o cerere simplă GET prin HTTP.

5.1.3.1. Metode de prevenire a vulnerabilităţilor de tip Cross-site Request Forgery

*a)* Cookie-uri postate de două or**i**

Această metodă de apărare constă în introducerea unui câmp de introducere a datelor secrete care să conţină valoarea actuală a ID-ului de sesiune a utilizatorului sau o altă valoare securizată generată aleator într-un cookie al clientului pentru orice formular folosit la transmiterea datelor sensibile. Când formularul este postat, serverul aplicaţiei va verifica dacă valoarea cookie-ului din formular coincide cu cea din antetul HTTP al cererii, în caz contrar cererea va fi ignorată ca şi invalidă şi va fi înregistrată în fișierele log ca potenţial atac. Această metodă se bazează pe faptul că atacatorul nu ştie valoarea cookie-ului de sesiune al utilizatorului, însă dacă prin altă metodă acesta reuşeşte să afle valoarea, această strategie de apărare nu va avea succes.

*b)* Nonce unic pentru formular

Este probabil cea mai folosită metodă de apărare împotrivă CSRF şi constă în construirea fiecărui formular folosind un câmp ascuns care conţine un nonce (number used once) obţinut folosind un generator pseudo-aleator de numere securizate prin criptare, pentru a nu fi vulnerabil la atac. Când serverul aplicaţiei primeşte valorile parametrilor formularului ca făcând parte dintr-o cerere HTTP POST, va compara valoarea nonce-ului cu valoarea stocată în memorie şi va ignora cererea dacă valorile acestora diferă sau dacă valoarea nonce-ului a expirat.

*c)* Cererea credenţialelor de autentificare

Această metodă le cere utilizatorilor autentificaţi să reintroducă parola corespunzătoare sesiunii în care sunt autentificaţi ori de câte ori fac o tranzacţie sensibilă. Această strategie este des întâlnită în aplicaţiile web în cadrul cărora tranzacţiile de o natură sensibilă se întâmplă rar (cel mai adesea fiind schimbări ale informaţiilor de pe profilul utilizatorului).

**6.Despre crearea unei aplicatii web folosind jQuery Mobile:**

Integrarea Dreamweave-ului cu jQuery Mobile ajuta la proiectarea rapida a unei aplicatii web care functioneaza pe majoritatea dispozitivelor mobile, in timp ce se adapteaza la dimensiunile dispozitivului.   
Deschideți o pagină de pornire jQuery Mobile, sau creati o pagina de HTML5.

Utilizați paginile de pornire jQuery mobile în Dreamweaver pentru a crea aplicatia dumneavoastră. Alternativ, puteți începe crearea aplicatiei dvs. web cu o pagină nouă HTML5.

Paginile de pornire jQuery mobile includ HTML, CSS, JavaScript, și fișiere de imagine care vă ajută să începeți cu proiectarea aplicatiei dumneavoastră. Puteți utiliza fișiere CSS și JavaScript găzduite pe un CDN, propriul server, sau fișierele instalate împreună cu Dreamweaver.

Introduceți componente jQuery mobile de la panoul de Insert.

Introduceți componente jQuery mobile de la panoul Insert în pagina HTML. Fișierele jQuery Mobile CSS și JavaScript definesc stilul și comportamentul componentelor.

*Despre CDN și fișierele locale jQuery mobile:*

CDN (Content Delivery Network) este o rețea de calculatoare care conține copii ale datelor plasate în diferite puncte din rețea. Când creați o aplicație web folosind URL-ul pentru fișiere CDN, CSS și JavaScript specificate în URL, sunt folosite pentru aplicația dumneavoastră. În mod implicit, Dreamweaver utilizează jQuery Mobile CDN.

Alternativ, puteți utiliza URL-uri de CDN de la alte site-uri, cum ar fi Microsoft și Google. În code view, editați locația serverului de CSS și fișierelor JavaScript specificate în <link> și tag-uri <src script>.

Fișierele descărcate de pe un CDN sunt de tip read-only.

*Fișierele locale jQuery mobile:*

Când instalați Dreamweaver, o copie a fișierelor jQuery Mobile este copiata pe computerul dumneavoastra. Pagina HTML care se deschide atunci când vă alegeți jQuery Mobile pagina de pornire (locala) este legată de CSS locale, JavaScript, și fișiere imagine.

*Pagini de pornire pentru jQuery Mobile:*

Dreamweaver vă oferă următoarele pagini de pornire pentru a crea aplicația web:

jQuery Mobile (CDN) (CS5.5 și versiunile ulterioare)

-Utilizați această pagină de pornire, dacă aveți de gând a găzdui biblioteca jQuery Mobile pe un CDN.

jQuery Mobile (locală) (CS5.5 și versiunile ulterioare)

-Utilizați această pagină de pornire în cazul în care cererea dumneavoastră nu se bazează pe o conexiune la Internet.

jQuery Mobile (PhoneGap) (CS5.5 și versiunile ulterioare)

-Utilizați această pagină de pornire în cazul în care aplicatia dumneavoastra web, atunci când este instalata ca o aplicație mobilă, accesează caracteristici native pentru dispozitive mobile.

Utilizarea paginilor de pornire pentru a crea o aplicație pentru dispozitive mobile:

1. Select File > New.
2. Selectați una din următoarele in funcție de necesitățile dvs.:

-Din pagina Exemplu> Mobile Starters> jQuery Mobile (CDN).

-Din pagina Exemplu> Mobile Starters> jQuery Mobile (Local).

-Din pagina Exemplu> Mobile Starters> jQuery Mobile cu tema (Local).

3. Click Create.

În pagina care apare, permite Follow Links Continuously (View> Live View Options), și comutati la Live View. Utilizați componente de navigare pentru a testa modul în care funcționează aplicația.

Utilizați opțiunile din meniul Multiscreen pentru a vedea cum proiectarea este afișata în dispozitive cu diferite dimensiuni. Dezactivati Live View, și reveniti la Design View.

4.În panoul Insert (Window> Inserare), selectați jQuery Mobile. Componentele pe care le puteți adăuga la aplicația web sunt afișate.

5. În Design View, plasați cursorul în locul unde doriți să inserați componenta, și faceți clic pe

componenta în Insert Panel. În caseta de dialog care apare, personalizati componentele folosind opțiunile.

6. (jQuery Mobile (Local), jQuery Mobile cu tema (Local)) După ce salvați fișierul HTML, fișierele

jQuery mobile, inclusiv imagini, sunt copiate într-un folder în locația fișierului HTML.

Previzualizați pagina în Live View. Unele dintre clasele CSS se aplică numai în Live View.

*Crearea unei aplicații web pentru dispozitive mobile dintr-o pagină nouă:*

Componenta Page acționează ca recipient pentru toate celelalte componente jQuery mobile. Adauga componenta Page înainte de a introduce alte componente.

1. Select File > New.
2. Select Blank Page > HTML.

Unele dintre componentele jQuery mobile folosesc atribute specific HTML5. Pentru a asigura respectarea HTML5 în timpul validării, asigurați-vă că selectați HTML5 ca DocType-ul dumneavoastra.

1. În panoul Insert (Window> Inserare), selectați jQuery Mobile din meniu. Componentele jQuery

mobile apar în panou.

1. De la panoul de Insert, glisați componenta page la Design View.
2. În caseta de dialog jQuery Mobile Files, selectați una dintre următoarele:

Remote (CDN)

- Dacă doriți să vă conectați la un server CDN la distanță care găzduiește fișierele jQuery Mobile. Utilizați opțiunea implicită pentru site-ul jQuery dacă nu ați configurat un site care conține fișiere jQuery Mobile. Puteți alege, de asemenea, să utilizeze alte servere CDN.

Local

- Fișierele care sunt disponibile în Dreamweaver sunt afișate. Pentru a specifica un alt folder, faceți clic pe Browse, și navigați la folderul care conține fișierele jQuery Mobile.

Fișierele CSS și JavaScript sunt copiate într-un director temporar local, până când salvați fișierul HTML pe computer. După ce salvați fișierul HTML, toate asociate jQuery Mobile și fișiere imagine sunt copiate dintr-un folder în folderul rădăcină al site-ului.

1. Introduceți proprietățile pentru componenta Page.
2. În vizualizarea Proiectare, plasați cursorul în locul unde doriți să inserați componenta, și faceți

clic pe componenta în panoul Inserare. În caseta de dialog care apare, personalizati componentele folosind opțiunile.

Previzualizați pagina în Live View. Unele dintre clasele CSS se aplică numai în Live View.

*Utilizarea fișierelor particularizate și foldere:*

Puteți alege pentru a crea CSS personalizat și fișiere JS pentru aplicația dumneavoastră. Asigurați-vă că fișierele sunt denumite jquery.mobile.js, jquery.mobile.css, și jquery.js

***Tema centrala a limbajului***

jQuery Mobile oferă un cadru tematic puternic, care permite dezvoltatorilor de a personaliza scheme de culori și anumite aspecte CSS de caracteristici UI . Dezvoltatorii pot utiliza aplicația jQuery Mobile ThemeRoller pentru a personaliza aceste aparențe și de a crea experiențe extrem de marcă. După ce în curs de dezvoltare o temă la cererea ThemeRoller , programatorii pot descărca un fișier CSS personalizat și să o includă în proiectul lor de a folosi tema lor personalizat .

Fiecare temă poate conține până la 26 de unic " specimene ", de culoare fiecare dintre care constă dintr- un bar antet , corp de conținut , și statele buton . Combinarea diferitelor specimene permite dezvoltatorilor să creeze o gamă mai largă de efecte vizuale decât ar fi în măsură să cu doar un singur specimen pe temă . Comutarea între diferitele specimene într-o temă este la fel de simplu ca adăugarea unui atribut numit " date temă " pentru elemente HTML.

Implicit tema jQuery Mobile vine cu cinci specimene de culori diferite , numite "a" , " b " , " c " , " d " , și " e " . Aici este un exemplu de cum să creați o bară de instrumente cu " b " specimen :

<**div** data-role="header" data-theme="b">

<**h1**>Page Title</**h1**>

</**div**>

Există deja o mână de teme de stil open source, care sunt dezvoltate și susținute de organizații terțe. Un astfel de temă stil open source este tema de stil Metro, care a fost dezvoltat și lansat de Microsoft Open Technologies, Inc Tema stil Metro este menit pentru a imita UI de Metro (limbajul de design) pe care Microsoft utilizează în exploatare sale mobile sisteme.

**7.Concluzii**

Tehnologia si aplicatiile telefoanelor mobile ofera foarte multe oportunitati pentru intreprinderi, ele prezentand de asemenea o continua dezvoltare si provocari noi.

Toate aplicațiile mobile introduse în piață au anumite efecte asupra mediului pe durata ciclului lor de viață.

Prin urmare, este important ca designerii de aplicatii au acces la informați ale mediului, astfel încât acestia să poată lua decizii adecvate indeplinind totodata cerințele de proiectare.

cel mai mare avantaj al aplicatiilor web pentru dispozitive mobile este acela ca suntem capabili sa le constrium pentru a fi independente de platforma, astfel putem mentine independenta fata de tipul dispozitivului si de sistemele de operare.   
            Pe scurt, dispozitivele mobile forteaza dezvoltatorii web sa se gandeasca la lucruri la care nu au fost nevoiti sa se gandeasca inainte. Aplicatiile web trebuie acum sa ia in considerare tipul dispozitivului utilizat pentru a determina cea mai buna experienta pentru utilizator. Dispozitivele mobile care au conexiuni cu latency mare, procesoare mai lente si o memorie mai mica trebuie luate in considerare la fel de mult ca si desktop-urile cu conexiuni cablate, procesoare rapide si memorie imensa. Dezvoltatorii web trebuie , acum mai mult ca niciodata, sa acorde o atentie sporita modului cum creaza interfetele, avand in vedere aceste constrangeri. Byte-ul conteaza, cererea conteaza, utilizarea memoriei si timpul de executie, toate trebuie sa fie luate in considerare, deoarece dezvoltarea aplicatilor pentru dispozitivele mobile este intr-o continua evolutie.

În cazul aplicațiilor web securitatea joacă un rol destul de important si totodată vast, deoarece nivelul informatic la care a ajuns web-ul este din ce mai mare. Utilizatorii cât și producătorii trebuie sa își mărească securitatea cât mai mult și în același timp să adauge mai multe constrângeri în sistemele pe care le gestionează. Adaptabilitatea sistemelor web este o proprietate care va deveni din ce în ce mai prețuită de către utilizatori.

Cadrul(framework-ul) JQuery este la început de dezvoltare deoarece produsele cu touch-screen sunt mai des întâlnite acum pe piață și beneficiile pe care le aduce programatorilor sunt considerabile. O observație importantă ar fi că acest framework ar putea fi înlocuit o tehnologie mai buna și de aceea nu putem afirma ca are un statut stabil în dezvoltare.

Bibliografie:

[1]Ciurea Cristian. The development of a mobile application in a collaborative banking system. Economy Informatics, vol.14,nr. 3,2010.

[2] http://www.marketwatch.ro/articol/3383/Mobile\_Information\_Security\_-\_o\_noua\_etapa\_in\_securitatea\_organizatiilor/

[3] http://helpx.adobe.com/dreamweaver/using/mobile-multiscreen-2.html