UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Usabilità nel Web browsing su dispositivi mobili di ultima generazione

Candidato: *Daniele Duranti*

Relatore: *Fabio Paternò*

Correlatore: *Mirko Tavosanis*

Anno Accademico 2008-2009

Sommario

INTRODUZIONE

La presente tesi è il risultato dello studio e delle riflessioni personali sull'argomento del web browsing tramite dispositivi mobili di ultima generazione.

Negli ultimi anni stanno emergendo telefoni cellulari in grado di accedere al full Web, ovvero agli stessi siti web ai quali accediamo dai dispositivi desktop. A causa delle caratteristiche che rendono un dispositivo mobile diverso da un dispositivo desktop, la visualizzazione dei siti Web presenta alcuni problemi. Si tratta di problemi sia di natura tecnica (incompatibilità di formato, incompatibilità di dimensioni dei file) sia di usabilità (difficoltà di interazione da parte dell'utente).

Nel corso del tempo sono quindi state proposte varie soluzioni a questo problema: progettare applicazioni web appositamente per l'accesso mobile, oppure, adattare automaticamente i siti web già esistenti.

Le tecniche di adattamento automatico proposte, se da un lato sono riuscite nell'intento di rendere compatibili le pagine web con i browser mobili, dall'altro non hanno raggiunto risultati eccellenti per quanto riguarda l'usabilità dei siti web. La pecca più grande di questi sistemi è quella di modificare troppo profondamente il layout della pagina di partenza.

Adesso, stanno emergendo dispositivi sempre più potenti in grado di supportare quasi tutte le più moderne tecnologie, e che si pongono inoltre l'obiettivo di visualizzare la pagina web così com'è, senza modifiche del layout, ma che forniscono per mezzo dei loro browser tecniche di visualizzazione o interazione con la pagina web in grado di migliorarne l'usabilità.

In un contesto in cui i dispositivi stanno diventando sempre più potenti hanno ancora senso le tecniche di adattamento che cercano di risolvere i problemi di compatibilità tecnica? Le tecniche di visualizzazione e interazione fornite dai browser più recenti risolvono completamente i problemi di usabilità, o sono ancora necessarie tecniche di adattamento automatiche?

Il quesito da cui si è partiti è quindi se le tecniche di adattamento dei siti web ai dispositivi mobili studiate finora fossero ancora valide o se invece non fossero più necessarie o se ne occorressero di nuove. In particolare sono stati presi in esame l'iPhone e il Nokia N95, due modelli di punta al momento sul mercato.

La risposta alla nostra domanda prevedeva alcune conoscenze preliminari:

- Che cos'è l'accesso al web da dispositivi mobili?
- Quali erano le caratteristiche del mobile web browsing in passato?
- Che cosa sono le tecniche di adattamento e quali sono state adottate in passato?
- Cosa è cambiato oggi? Come si comportano i nuovi dispositivi mobili? Qual è la loro filosofia?

Non essendo un esperto di questo settore oltre alle prove dirette su questi due dispositivi, è stato necessario uno studio teorico della letteratura internazionale esistente in merito.

Nel corso della tesi si riportano le conoscenze teoriche apprese e si tenta di dare una risposta personale al quesito sull'adattamento. Successivamente si passa ad analizzare l'importanza di avere uno strumento automatico di supporto ai test di usabilità. In particolare, si è passati allo sviluppo del

prototipo di una nuova versione di uno strumento automatico per la valutazione dell'usabilità specifico per applicazioni mobili.

1. ACCESSO AL WEB DA DISPOSITIVO MOBILE

È quanto mai difficile sistematizzare in un discorso organico e lineare un argomento così complesso come quello dell'accesso a Internet tramite dispositivi mobili.

Si tratta infatti di un fenomeno recente che ha avuto inizio appena alla fine degli anni 90: il mondo dell'accesso al Web da dispositivo mobile si presenta quindi come frammentato, diviso in guerre tra browser, statistiche, opinioni spesso divergenti. Di punti fermi ce ne sono pochi.

Questa tesi, tra i vari obiettivi, si pone anche quello di cercare di fare un po' di chiarezza su questo argomento.

1.1 Ha senso accedere al web tramite dispositivo mobile?

L'accesso a Internet è ormai divenuto un fatto comune. Ormai, per necessità, per lavoro, per divertimento, quasi tutti ci colleghiamo a Internet per navigare tra le pagine Web, scaricare o inviare la posta, e altro.

Negli ultimi anni si assiste al tentativo di rendere Internet accessibile anche tramite il telefono cellulare. Quando si parla dell'accesso a Internet tramite telefoni mobili, molte persone sorridono e si chiedono a che cosa serva.

La prima domanda a cui si ritiene sia interessante dare una risposta è quindi se abbia senso e sia veramente utile accedere al web tramite il proprio telefono cellulare.

La risposta è sì. Come afferma NOKIA in un suo slogan "life goes mobile", per cui accedere al web on mobile diviene sempre più un'operazione necessaria e importante almeno quanto l'accesso al web tramite dispositivi tradizionali.
Un'altra ragione per cui l'accesso al Web tramite dispositivi mobili è da considerare importante è la possibilità di abbattimento o attenuazione del digital divide che esso potrebbe fornire[1].
Per esempio, in Africa, Internet è usata solo dal 4% della popolazione e solo l'1% ha accesso alla banda larga. Questo è causato sia dalla povertà endemica che non consente di far fronte ai costi ma anche all'effettiva mancanza di infrastrutture per la telefonia fissa. Tuttavia, dagli anni 2000 in Africa la telefonia mobile ha iniziato a crescere a livelli tripli rispetto a ogni parte del mondo. Alcune regioni del mondo conoscono il loro primo telefono attraverso il cellulare e conosceranno per la prima volta Internet per mezzo del cellulare.

1.2 Terminologia: Internet, Web, full Web, mobile Web

Prima di iniziare la trattazione, è importante metterci d'accordo sul significato di alcune parole chiave che ricorreranno spesso: Internet, Web, Full Web, Mobile Web.

Leggendo articoli e documentazioni sull'argomento, ci siamo accorti come spesso questi termini siano usati per riferirsi a concetti diversi, talvolta contrastanti.

[1] Per digital divide si intende la disomogeneità tecnologica tra le varie aree mondiali, problema importante, poiché una inferiorità tecnologica, in un mondo ormai pervaso dalla tecnologia, implica una condanna all'arretratezza sociale.

Si è deciso di proporre qui di seguito quelle che ci sembrano essere le definizioni più sensate o comunque quelle a cui ci riferiremo nel corso di questa tesi.

Partiamo da due termini base, spesso usati impropriamente come sinonimi: Internet e Web.
Internet è una rete di computer mondiale ad accesso pubblico attualmente rappresentante anche uno dei principali mezzi di comunicazione di massa. [Wikipedia]
Il Web, invece, è uno dei tanti servizi (il più utilizzato e conosciuto insieme alla posta elettronica) offerti da Internet. Consiste in una ragnatela di documenti collegati tra loro e presenti in server dislocati in tutto il mondo.

Quando erano solo i PC a collegarsi a Internet sarebbero bastate queste due definizioni. Ma verso la fine degli anni 90, anche i primi dispositivi mobili cominciano a collegarsi al Web. A causa delle loro capacità limitate che ne impedivano l'accesso al Web tradizionale, venne sviluppato un Web dedicato e specifico per la telefonia mobile[2].

Per evitare confusione parleremo di **Full web** per riferirci al web a cui siamo abituati ad accedere tramite PC, mentre ci riferiremo a **Mobile Web** per riferirci al Web creato su misura per l'accesso tramite dispositivi mobili. Utilizzeremo invece la parola **Web** come termine generale che comprende sia il Full web che il Mobile Web.

Nella seguente figura si tenta di schematizzare il concetto appena proposto:

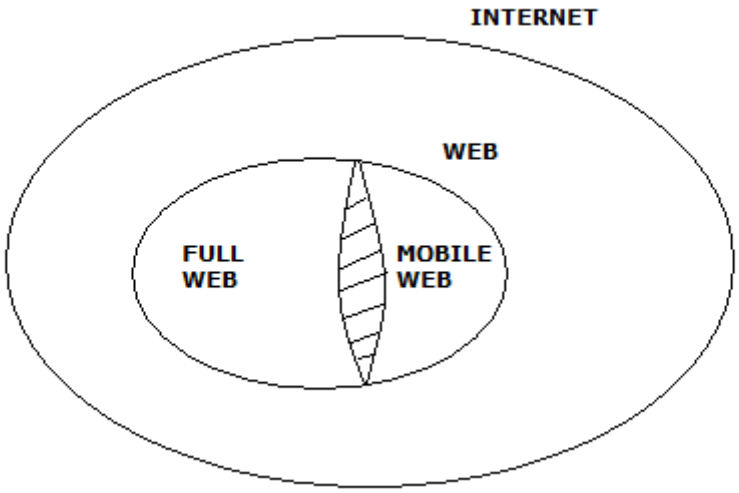

Figura 1 Rappresentazione grafica dei concetti di Internet, Web, Full Web e Mobile Web e loro relazione

Potremmo domandarci il perché dell'intersezione tra full web e mobile web. Questa **intersezione** è presente in quanto soprattutto nei tempi più recenti sono stati proposti siti web progettati (come soluzione unica) per essere acceduti sia tramite dispositivi mobili, sia tramite dispositivi tradizionali[3].

[2] La distinzione ha senso ancora oggi che i telefoni mobili hanno la capacità di accedere al full web. Mostreremo più avanti perché sia comunque tuttora importante anche la progettazione di siti web progettati appositamente per l'accesso da dispositivo mobile

[3] In generale, nel momento in cui ci troviamo a dover sviluppare un'applicazione che funzioni su dispositivi diversi tra loro, abbiamo varie possibilità:

- Sviluppare una versione specifica per ciascuna piattaforma
- Sviluppare una versione con differenti sottoversioni

Se in passato un telefono cellulare poteva accedere solo ad un web dedicato e indipendente da quello tradizionale, attualmente i dispositivi mobili sono capaci di accedere sia al Mobile Web che al Full Web.

1.3 Che cosa significa accedere al web tramite dispositivo mobile?

È difficile dare una definizione di Web browsing on mobile[4].

Il fatto che l'accesso al Web tramite dispositivo mobile sia un fenomeno relativamente recente, il fatto che sia cambiato nel tempo, le diverse caratteristiche dei vari device mobili, il tipo di pagine che sono capaci di visualizzare, il modo in cui lo fanno, la nascita di standard e la loro compresenza con soluzioni non standard, le diverse opinioni dei ricercatori e produttori, hanno reso difficile definirlo.

I modi in cui è stato definito sono profondamente diversi:

- Hinman et al (2008) quando parlano di mobile browsing si riferiscono alla visualizzazione dei siti web tradizionali su dispositivo mobile
- Cui e Roto (2008) descrivono il mobile web browsing come la visualizzazione di pagine web per mezzo di un browser mobile, dove per pagine web intendono sia le pagine ottimizzate per i cellulari sia il full web
- Taylor et al. (2008): sembrano percepire il mobile browsing principalmente come l'utilizzo di servizi su misura per i dispositivi mobili
- Kaikkonen (2008): definisce il mobile browsing come qualsiasi accesso a Internet con dispositivo mobile

Che cos'è dunque il Mobile browsing *oggi*?

Potremmo dire che è tutto questo.

Mettendo insieme tutte queste possibilità, potremmo riassumere l'accesso al web tramite dispositivo mobile con il seguente schema:

- Sviluppare una versione generale

Lo sviluppo di una versione generale o di una versione con differenti sottoversioni (desktop/mobile) non sempre è stato possibile, dato che inizialmente utilizzavano linguaggi totalmente diversi e incompatibili tra loro. Tale possibilità si è presentata dal momento che le tecnologie di base per la costruzione di siti web sia per cellulari che per dispositivi tradizionali sono divenute le stesse.

[4] Abbiamo preferito utilizzare l'espressione "Web browsing on mobile" piuttosto che "Mobile Web browsing" perché vogliamo qui riferisci all'accesso al Web (senza specificarne il tipo) tramite dispositivo mobile. La seconda espressione potrebbe essere soggetta a interpretazioni ambigue ovvero "Navigazione del web tramite dispositivo mobile" ma anche "navigazione del Mobile Web"

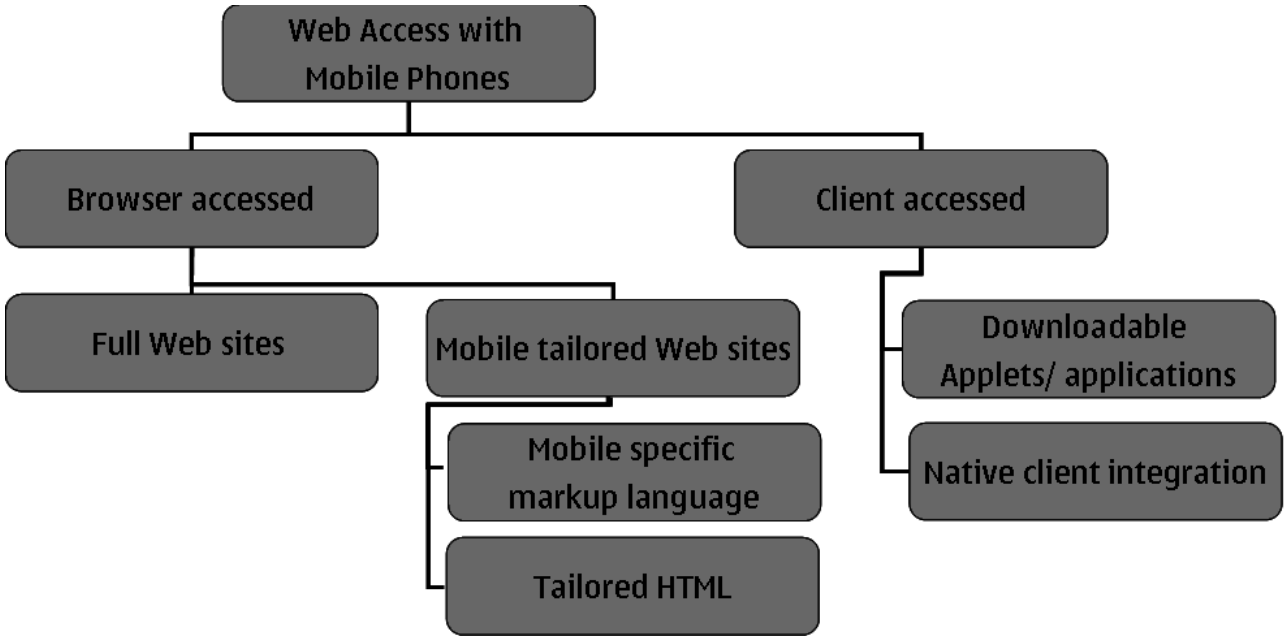

Figura 2 Differenti modalità di accesso al Web tramite dispositivo mobile (A.Kaikkonen, 2008)

Come mostra la figura l'accesso al web con dispositivo mobile può essere diviso prima di tutto in due categorie:

- Accesso per mezzo del browser
- Accesso per mezzo di un'altra applicazione (diversa dal browser)

Nell'accesso a Internet per mezzo del browser si possono individuare altre due alternative:

- Accesso agli stessi siti a cui si accede tramite PC (si parla in tal caso di Full web)
- Accesso a siti costruiti su misura per i dispositivi mobili (mobile tailored web sites)

1.3.1 Accesso al full web

I siti full web sono quelli sviluppati per l'uso su computer con lo standard HTML.

Il contenuto che si vede sul mobile browser è praticamente lo stesso che l'utente vede quando naviga il sito sul computer, fatta eccezione per alcune differenze dovute ai limiti tecnici dei dispositivi.
Molti mobile browser per esempio non supportano tutti i formati audio e video: l'utente potrebbe non essere capace di ascoltare la musica di sottofondo o vedere videoclips. Anche se c'è da dire che i browser e i dispositivi attuali si stanno evolvendo fino a poter visualizzare gli stessi siti senza limitazioni.

Le ridotte dimensioni del display e le modalità di interazione con il dispositivo determinano spesso tuttavia problemi di usabilità[5]. Nel corso del tempo sono stati quindi studiati dei metodi automatici che consentano di adattare i contenuti web allo schermo dei dispositivi mobili.

[5] Esistono varie definizioni di usabilità. La più riconosciuta è quella dello standard ISO 9241 (Requisiti ergonomici per lavoro di ufficio con terminali). Essa definisce l'usabilità come: la misura in cui un prodotto può essere usato da specifici utenti per raggiungere specifici obiettivi con:

- Efficienza: le risorse spese in relazione all'accuratezza e completezza degli obiettivi raggiunti (es: tempo richiesto)
- Efficacia: l'accuratezza e la completezza con cui gli utenti possono raggiungere i propri obiettivi
- Soddisfazione: il comfort e l'accettabilità del sistema per gli utenti e le altre persone influenzate dal suo uso.

1.3.2 Accesso al mobile web

Fanno parte del mobile web, tutte quelle pagine web che sono state pensate appositamente per un uso mobile. Possono essere costruite con un linguaggio di markup specifico dei telefoni mobili (come HDML, WML,cHTML, XHTML Mobile Profile) o, adesso, tramite XHTML standard.

1.3.3 Accesso tramite applicazione

Ci si riferisce alle applicazioni che permettono di scaricare o fare uploading di contenuto su e da Internet. L'applicazione può essere sia un'applicazione nativa (come un calendario, una photogallery, music player ecc.) oppure un'applicazione scaricata dal web e installata sul dispositivo.
Queste applicazioni si connettono ad un sito specifico per richiedere specifiche informazioni o svolgere specifici task.

1.4 Full web, mobile web: proprio necessari due web separati?

Inizialmente i dispositivi mobili non avevano le risorse sufficienti per accedere ai siti tradizionali: era fondamentale l'esistenza di un web dedicato per la telefonia mobile.

Abbiamo visto, però, che i più moderni dispositivi mobili sono ormai capaci di accedere agli stessi siti web a cui siamo abituati ad accedere tramite il computer, eventualmente riadattati automaticamente per sopperire ai limiti tecnici e migliorarne l'usabilità.

Una domanda che può essere lecita è se quindi abbia ancora senso l'esistenza del Mobile Web.

La risposta è sì. Proviamo a elencarne qui di seguito i motivi:

- *I siti web progettati appositamente per il mobile web sono più usabili*
  Se è vero che alcune tecniche di adattamento automatico possono migliorare l'usabilità dei siti full web quando sono fruiti da dispositivo mobile, è altrettanto vero che i risultati degli adattamenti automatici spesso sono deludenti o comunque presentano dei limiti.
  Il limite più grande è quello di non prendere abbastanza in considerazione un elemento fondamentale e cioè che il contesto d'uso è diverso per cui i task che l'utente ha bisogno di svolgere possono essere parzialmente o totalmente diversi da quelli presentati dall'interfaccia del sito full web.
  Quello che noi ci limitiamo a fare con l'adattamento automatico è invece semplicemente di adattare un'interfaccia che è stata progettata per desktop al dispositivo mobile: in questo approccio i task di partenza rimangono gli stessi anche se vengono concretizzati in un'interfaccia leggermente o totalmente diversa per ogni dispositivo.
- *Offre possibilità aggiuntive.*
  Una di queste possibilità è la localizzazione: si possono sviluppare siti web che si plasmano a seconda del posto in cui ci troviamo
- *Alcuni dispositivi non sono ancora in grado di accedere al full web e le traduzioni automatiche da un linguaggio all'altro portano a risultati molto deludenti.*

A questi motivi se ne possono aggiungere altri che confermano la volontà di esistenza di un Mobile Web:

- Esiste un consorzio che ne cura la standardizzazione (OMA: Open Mobile Alliance)
- Esiste un dominio dedicato .mobi
- Esiste un linguaggio ufficiale: xhtml-mp
- È effettivamente richiesto dagli utenti.

Secondo una recente statistica[6] :

- quasi il **70% dei partecipanti** al sondaggio naviga **sia siti full web che siti Mobile Web**
- solo il **14%** navigava solo **siti full web**
- il **32%** navigava solo **siti mobile web**
- solo il **7%** utilizzava **applicazioni** diverse dal browser per l'accesso al web.

Appurato che è utile l'esistenza di un mobile web, potremmo chiederci, a questo punto, se abbia senso accedere al full web da mobile.

Anche in questo caso la risposta è affermativa, per due motivi:

- Non sempre è possibile sviluppare e mantenere due versioni del proprio sito
- Talvolta gli utenti desiderano utilizzare gli stessi siti web a cui accedono tramite pc

Da quanto detto sembrerebbe quindi necessaria l'esistenza di entrambe le tipologie di Web, anche se non tutti sono d'accordo[7].

Per conoscere più in dettaglio le motivazioni per cui gli utenti scelgono di accedere al mobile web o al full web si legga il paragrafo successivo.

1.5 Identikit dell'utente mobile

Nel paragrafo precedente abbiamo accennato al fatto che l'utente del web on mobile è diverso da quello tradizionale. Abbiamo anche detto che l'utente desidera accedere sia al full web che al mobile web tramite dispositivo mobile.
Adesso, cercheremo di descrivere il comportamento degli utenti del web on mobile e le motivazioni che li spingono alla scelta del mobile web o del full web.

Quasi per definizione, gli utenti accedono al web **on mobile** (in macchina, in treno, seduti in un bar o su una panchina al parco, facendo una passeggiata, facendo shopping, in attesa di un appuntamento) , e sono quindi utenti distratti, poco concentrati sul sito che stanno navigando.

Generalmente non sono interessati a tutto ma solo a certi tipi di informazioni[8] che vogliono ottenere velocemente proprio per la loro mobilità, che impedisce loro di potersi concentrare per lungo tempo.

---

[6] Tale statistica è stata condotta nel 2007 da Kaikkonen su un insieme di 390 utenti del Web tramite dispositivi mobili provenienti da vari paesi (Hong Kong, Londra, New York), che hanno compilato un questionario. Di questi utenti, 23 sono stati scelti per ulteriori interviste più specifiche.

[7] Il W3C (World Wide Web Consortium) si è fatto promotore dell'iniziativa One Web che prevede che lo stesso contenuto sia accessibile da ogni dispositivo:

" […]One Web means making, as far as is reasonable, the same information and services available to users irrespective of the device they are using. However, it does not mean that exactly the same information is available in exactly the same representation across all devices. The context of mobile use, device capability variations, bandwith issues and mobile network capabilities all effect the representation. Furthermore, some services and information are more suitable for and targeted at particular user contexts[…] "

[8] Tra le varie contenuti a cui gli utenti potrebbero essere maggiormente interessati abbiamo:

1. vari tipi di informazioni "dell'ultimo momento" (notizie, previsioni del tempo, rapporti sul traffico e così via),
2. vari dizionari/enciclopedie, come Wikipedia, o alter pagine che spiegano il significato delle parole
3. informazioni sull'area locale,es: mappe e ricerca di ristoranti
4. varie tabelle, come orario dei treni, tabelle tariffarie, programmazione televisiva o del cinema, e così via
5. vari moduli, come quelli per la prenotazione dei biglietti

Ci sono comunque studi [Cui e Roto, 2008], che dimostrano come spesso l'utente del web tramite dispositivo mobile sia "stazionario", piuttosto che mobile.
Tra i motivi che spingono a questo abbiamo la mancata disponibilità di un computer oppure il fatto che il contesto sociale non ne permette l'uso.

È facile pensare, quindi che gli utenti in movimento preferiranno i siti web mobile, mentre quelli stazionari preferiranno quelli full web.
Generalmente è così ma non sempre: può capitare comunque che si decida di accedere al full web da dispositivo mobile proprio perché ci consente più mobilità (e quindi di cambiare la locazione) durante la navigazione.

1.5.1 Motivazioni della scelta del mobile Web o del full Web

Proviamo adesso a elencare i motivi che spingono gli utenti ad accedere al mobile web o al full web.

Riassumendo, accedono al mobile web gli utenti che:

- Vogliono accedere velocemente e con facilità a informazioni (come timetable e weather information)
- Gli utenti che desiderano passare un po' di tempo (time killing)

Accedono al full web gli utenti che:

- Cercano informazioni che sono disponibili solo nel full web
- Non sono a conoscenza di soluzioni mobili
- Non hanno a disposizione un computer e hanno bisogno di accedere al web tradizionale
- Il contesto sociale non permette loro l'uso del computer
- Sanno che cambieranno locazione durante lo svolgimento del task
- Time killing (ma più raro rispetto al mobile web)

Notare che nei primi due casi gli utenti hanno un bisogno informativo specifico, sanno cosa vogliono e dove cercarlo.

1.6 Differenza tra esperienza del web tramite computer e tramite dispositivo mobile

Accedere al Web dal telefono mobile è un'esperienza diversa rispetto ad accedervi dal PC, sia che si visitino siti mobile web che si usi il full web.

Sono interessanti, a questo proposito, gli studi di Hinman et al.(2008) che hanno portato alla definizione di una metafora per descrivere l'esperienza degli utenti che usano sia il telefono mobile che il computer per accedere a Internet:

- Il PC Web browsing è simile all'immersione subacquea: in quanto è immersivo, invita all'esplorazione e supporta multitasking
- L'accesso al web da dispositivo mobile è simile al nuoto in superficie (con uso del boccaglio): in quanto l'attenzione è divisa ed è difficile essere totalmente immersi

Questa metafora è stata successivamente estesa da A. Kaikkonen[9].
Come si è detto in precedenza, infatti, ci sono modi differenti di accedere al Web tramite Internet (accesso al full web o accesso al mobile web, accesso tramite applicazioni), per cui ci saranno diversi modi di nuotare in superficie:

- Il Full Web browsing on mobile è simile alle immersioni libere (senza scafandro)[10]

[9] MOBILE INTERNET- PAST, PRESENT USAGE AND THE FUTURE (2008), Anne Kaikkonen, Nokia Corporation, Finland

La navigazione on mobile di siti full web è quindi come le immersioni libere perché il bisogno informativo è generalmente specifico e l'utente sa dove trovare l'informazione, non importa quanto in profondità sia: va lì direttamente.

- L'accesso al mobile web è simile al nuoto in superficie con occasionali immersioni in profondità. Gli utenti accedono al mobile web per vari motivi (1.5.1) : nel corso di queste operazioni, gli utenti navigano in superficie ma soprattutto nel caso del time killing, se vedono informazioni interessanti, scendono in profondità, ma solitamente poi ritornano in superficie per visualizzare il prossimo argomento interessante.
- L'accesso al Web tramite applicazioni mobili è simile al nuoto in piscina poiché è possibile svolgere solo gli specifici tasks relativi all'applicazione, e non è solitamente possibile visualizzare informazioni che vanno al di fuori di certi confini.

È interessante notare inoltre come generalmente la sessione di browsing duri per un tempo piuttosto breve. La durata dipende dal tipo di accesso alla rete: il costo è la maggiore ragione di questa differenza.

[10] Secondo l'opinione di chi scrive questa metafora tiene conto soltanto degli utenti che hanno un bisogno informativo specifico escludendo tutti gli altri, la cui esperienza dovrà essere necessariamente più simile a quella tramite PC.

2. UN PO' DI STORIA

Nel capitolo precedente si è cercato di introdurre l'argomento dell'accesso a Internet tramite dispositivi mobili. Si è fornito uno schema che possiamo considerare una fotografia di che cosa sia il web browsing on mobile in questo momento.

È importante notare, tuttavia, come la situazione presentata dallo schema sia il risultato di una evoluzione storica in cui possiamo individuare varie fasi. In questo capitolo ripercorriamo quelle principali.

Prima, però, forniamo per completezza una seppur breve e schematica storia del telefono cellulare.

2.1 Breve storia del telefono cellulare

2.1.1 Prima generazione – tecnologia analogica

Le prime tecnologie che permisero l'utilizzo di un telefono mobile furono sviluppate intorno agli anni 20 del 900 quando i primi pionieri nel campo proposero degli standard utilizzabili esclusivamente in auto.
Questi dispositivi comunicavano per mezzo di stazioni radio base (BSS) che erano connesse le une alle altre alla rete telefonica pubblica con dei sistemi di commutazione MSC che permettevano lo spostamento dell'utente senza chiudere la comunicazione.
Tuttavia, questi modelli primitivi, pesanti, non erano disponibili alla pubblica utilità ma erano destinate ai militari, alle forze dell'ordine ecc. Inoltre, nonostante garantissero una copertua di circa 100 Km, non consentivano la gestione di elevati numeri di utenze nella rete.

Nel 1943 si ebbe il primo avvicinamento all'odierna tecnologia cellulare, con il tentativo di distribuire una rete di celle in una vasta area, progetto che riuscì solo nel 1973 grazie all'invenzione di Martin Cooper di Motorola, il quale effettuò la prima chiamata da un telefono mobile, il motorola Dyna-Tac.

Da quel momento i cellulari erano detti da "valigia" in quanto erano pesanti 1 kg e dovevano essere portati all'interno di piccole valigie.
Purtroppo i Motorola Bag Phone resistevano per soli 55 minuti di comunicazione telefonica.

Negli anni 80 si diffusero i cellulari mattone (buck phone) che sfruttavano le reti analogiche come i cellulari da valigia ma erano di dimensioni più ridotte anche se la loro denominazione lascia intendere la somiglianza con un mattone.

2.1.2 Seconda generazione – GSM(Global System for Mobile Communication)

Nel 92 si ebbero sostanziali cambiamenti con l'avvento delle reti digitali (ad es. GSM) che consentì di aumentare il numero di utenze e consentì anche la spedizione di brevi messaggi di testo.
Tra le caratteristiche principali del GSM:

- Standardizzato in Europa
- Operativo in 110 paesi del mondo
- Area suddivisa in celle ciascuna coperta da un'antenna
- Sistema a commutazione di circuito
- Ogni utente ha un canale di 9600 bps
- Tariffazione a tempo

2.1.3 Generazione 2.5 – GPRS (General Packet Radio Service)

Il GPRS si basa su una rete a commutazione di pacchetto che consente di inviare MMS, effettuare chiamate di gruppo, connettersi a Internet (limitatamente a siti WAP).
Tra le caratteristiche più importanti:

- Utilizza in gran parte la struttura GSM esistente
- Banda fino ad un massimo teorico di 171.2 kbps
- Commutazione di pacchetto
- Tariffazione in base al volume dei dati trasferiti

2.1.4 Terza generazione – UMTS (Universal Telecommunication System)

Attivato nel 2003 il sistema UMTS si basa sulle specifiche date dalla International Telecom Union – IMT – 2000. Consente una capacità elevata di connessione alla rete Internet rispetto alla GPRS (fino a 2Mbps); è più orientata ai dati: consente la trasmissione di immagini, grafica, comunicazione video, voce e dati.

È interessante notare che andando avanti nella storia dei telefoni cellulari si vede come cresca via via sempre di più l'importanza dell'accesso ai dati.

2.2 Storia dell'evoluzione di Internet in direzione mobile

2.2.1 Nascita di Internet e del Web

Internet è stato sviluppato da ARPANET nel 1968 e da una rete di collaborazione tra università (1983) si è passati ad un sistema di reti di computer interconnessi tra loro a livello mondiale.

Fino al 1990 Internet era praticamente sconosciuta al di fuori del mondo accademico.
Essa era usata soprattutto da ricercatori, accademici e studenti universitari per il collegamento a host remoti, per trasferire dati, per spedire e ricevere posta elettronica.
È all'inizio degli anni '90 che arriva sulla scena l'applicazione killer di Internet che attirò lo sguardo del pubblico generico e che ha cambiato profondamente il modo di interagire delle persone all'interno e all'esterno del mondo del lavoro: il Web.
Il Web fu inventato al CERN da Tim Berners-Lee nel 1989-1991, basandosi sulle idee ispirate dai primi lavori sull'ipertesto degli anni 40 di Bush e degli anni 60 di Ted Nelson.
Berners-Lee e i suoi collaboratori svilupparono le versioni iniziali di HTML, HTTP, un server Web e un browser: i quattro componenti base del Web.

Il servizio Web si basa infatti su questi quattro elementi:

- Il **protocollo HTTP**:
  è il protocollo dello strato di applicazione del Web. Esso definisce la struttura e le modalità con cui un server e un client si scambiano i messaggi.
- **Linguaggio HTML**:
  linguaggio di marcatura con cui sono scritti i file base delle pagine Web. Le pagine Web consistono infatti di vari oggetti: un file base HTML e molti oggetti referenziati.
  Ogni pagina è caratterizzata da un indirizzo chiamato URL (Universal Resource Locator).

- **Browser** (interprete) è un agente dell'utente per il Web; esso mostra la pagina Web richiesta e fornisce molte funzionalità di navigazione e configurazione. I browser del Web implementano anche il lato client dell'HTTP.
- **Server web**: memorizza gli oggetti Web, ciascuno indirizzabile da un URL

A grandi linee il funzionamento generale del Web è il seguente: quando un utente richiede una pagina Web, il browser invia messaggi HTTP di richiesta al server per ottenere gli oggetti che costituiscono la pagina. Il server riceve la richiesta e risponde con messaggi di risposta HTTP contenenti gli oggetti.

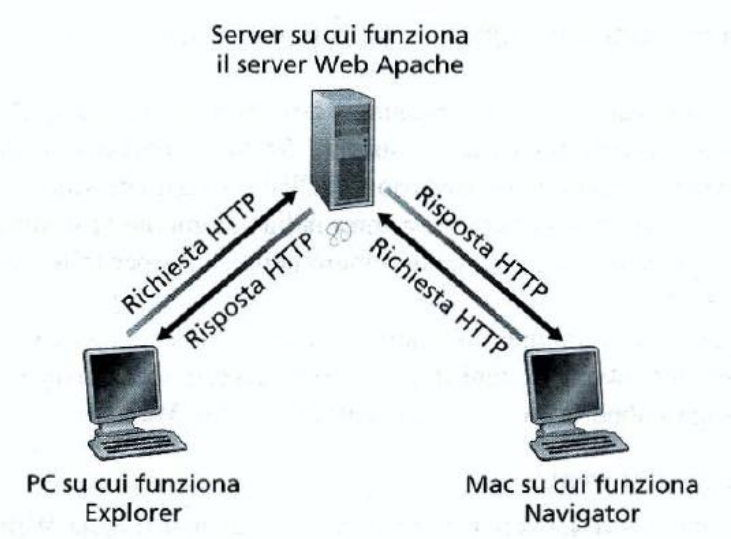

Figura 3 Comportamento richiesta-risposta dell'http (Kurose e Ross, 2003)

2.2.2 Nascita del primo WEB dedicato: il WAP

Nel 1998, Nokia rilasciò il primo telefono mobile capace di accedere ad informazioni su Internet.
Allora, i telefoni mobili non erano capaci di scaricare e mostrare le pagine appartenenti al full web, ma erano necessarie pagine web ottimizzate. Il linguaggio HTML era troppo complesso e barocco quindi inadeguato per i cellulari, soprattutto se consideriamo le ridotte capacità computazionali e di memoria del cellulare, lo schermo piccolo, la lentezza e il costo del collegamento.

Handheld Device Markup Language (HDML) fu il primo linguaggio ipertestuale progettato specificatamente per dispositivi mobili.
Da quel momento molti altri linguaggi di markup leggeri sono stati progettati per l'uso da parte di dispositivi mobili:

- C-HTML (Compact HTML)
- WML (Wireless Markup Language)
- XHTML Basic
- XHTML Mobile Profile

Questo primo servizio che permetteva l'accesso tramite dispositivi mobili ai contenuti scritti tramite i linguaggi di markup visti sopra prendeva il nome di WAP[11].

[11] Il WAP nasce nell'ambito del WAP-Forum, un'associazione di industrie comprendente Ericsson, Motorola, Nokia, Phone.com, nata nel 1997 con lo scopo principale di sviluppare nuovi standard e servizi per i terminali wireless.

Il modello WAP è simile al modello WWW:

- Utilizza un linguaggio di markup per i contenuti (WML): si tratta di un linguaggio basato su tag ma progettato per dispositivi con capacità limitate in grado di supportare testo, immagini, input da parte dell'utente e diversi meccanismi di navigazione.
  La pagina WML è detta deck cioè "mazzo di carte" e come questo è composta da più carte ovvero da unità di navigazione
- Utilizza il protocollo WAP che definisce la struttura e la modalità con cui un server e un client si scambiano i messaggi.
- Si basa su un'architettura client[12]- server.
  Nel WAP tuttavia è necessaria un'intermediazione tra client e server di un gateway (proxy). Il WAP gateway si occupa della traduzione dallo stack di protocolli WAP allo stack di protocolli Internet, permettendo così la comunicazione tra WAPClient e WebServer.
  Se quest'ultimo fornisce contenuti compatibili con WAP (es:WML) allora il proxy può recuperare direttamente queste informazioni e inviarle al client, altrimenti può essere configurato per agire da filtro che traduce il contenuto WWW (HTML) in contenuto WAP (WML) ma con risultati molto deludenti.

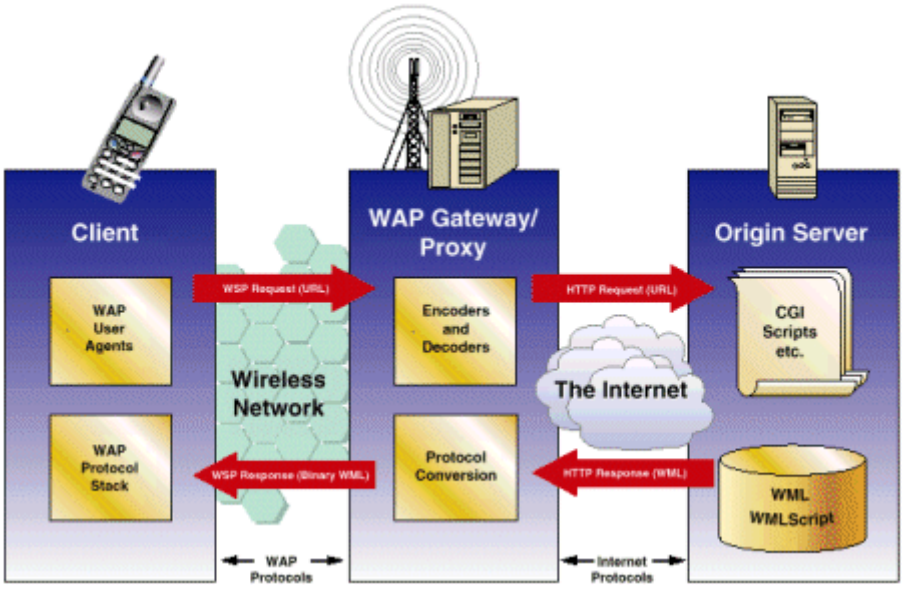

Figura 4 Architettura WAP (www.gsmworld.it)

Queste appena viste sono le caratteristiche del WAP 1.1.

Successivamente al WAP 1.1 fu introdotto il WAP 1.2 che non ha portato significative novità se si esclude l'introduzione del WAP PUSH che consente di ricevere, unendo la tecnologia SMS e WAP, contenuti multimediali o pagine Web direttamente sul proprio cellulare.

Queste prime due versioni del WAP si rivelarono un insuccesso clamoroso.
La causa di questo fallimento fu probabilmente dovuta al fatto che il WAP era stato pubblicizzato come "Internet disponibile nelle proprie tasche", così che le aspettative degli utenti erano relativamente alte. Tuttavia il WAP non rispondeva a queste aspettative:

Poiché lo scopo era lo sviluppo di nuovi standard, il WAP-Forum lavorò in stretta collaborazione con gli enti internazionali di standardizzazione (ETSI, W3C, TIA, IETF, CTIA)

[12] Il client è chiaramente sempre il browser, che nel caso dei dispositivi mobili prende il nome di microbrowser.

- I siti disponibili appositamente per il WAP apparivano così diversi (e poco accattivanti) rispetto ai siti web tradizionali
- Occorreva imparare nuovi indirizzi per raggiungere le pagine
- La scelta di siti era limitata rispetto al "full web"[13]

Se a questo aggiungiamo la lentezza della connessione, le spese piuttosto elevate, la difficoltà di configurazione, si capisce come questo insuccesso fosse prevedibile[14]: gli utenti non cominciarono mai ad utilizzare i siti mobili, gli sviluppatori di servizi mobili non videro un ritorno ai loro investimenti e dovettero togliere di mezzo i siti mobili.

Solo più tardi, dopo aver risolto molti dei problemi visti sopra, l'uso di servizi mobile è leggermente aumentato.

Che cosa ci ha insegnato l' insuccesso del WAP

Non è molto interessante chiederci se il WAP è stato uno sbaglio.
È molto più interessante chiederci cosa abbiamo imparato dal WAP e considerarlo come una fase essenziale nell'evoluzione del Web on mobile.
Prima di tutto, durante i primi anni del WAP molti ricercatori hanno pubblicato articoli riguardanti la progettazione di interfacce utente e l'usabilità per piccoli schermi e reti più o meno lente.
Dall'insuccesso del WAP si è inoltre compreso l'importanza di prendere in considerazione le aspettative e i modelli mentali dell'utente. In quel periodo infatti l'analisi dell'utente non era molto presa in considerazione, ci si concentrava molto di più sui messaggi di marketing. È fondamentale invece conoscere l'utente, le sue aspettative e valori e impostare il messaggio in base a queste caratteristiche.

2.2.3 WAP 2.0

Abbiamo visto che i siti WAP 1.x erano isolati dal World Wide Web: questi siti non erano visibili attraverso i browser desktop, e i siti www non potevano essere acceduti dai dispositivi WAP.
Un primo tentativo di integrazione tra WEB e WAP si ebbe con il WAP 2.0, il quale abbandona il WML mentre il linguaggio usato diventa l'XHTML Basic, un linguaggio che include molte delle possibilità di XHTML[15] eccetto quelle che non sono appropriate per dispositivi mobili[16].
Il vantaggio dell'uso di questo linguaggio, oltre al possibile e immediato utilizzo da parte degli sviluppatori, è la possibilità per i siti che lo utilizzano di essere visualizzati sia dai PC che dai dispositivi mobili. Adesso chi

[13] Nonostante, come abbiamo visto, fosse possibile richiedere al server pagine HTML, ricevendone una versione WML da parte del server, la conversione non era trouble-free: spesso essa limitava la quantità di informazioni disponibili o rendeva incomprensibili anche le informazioni ottenute. In pratica i dispositivi mobili potevano accedere non all'intero Web ma solo a quel sottoinsieme rappresentato dai siti WML. WEB e WAP erano due servizi separati E' per questo che molti Web publisher creavano due versioni delle loro pagine, una per il web e una per il wap

[14] In Giappone il servizio simile al WAP, il servizio i-mode, fornito dall'operatore NTT DoCoMo, ottenne un discreto successo forse proprio perché diverse erano le aspettative degli utenti

[15] XHTML consiste nella riformulazione di HTML come applicazione XML. Ne derivano alcuni vantaggi, in particolare: è possibile applicare strumenti (come validatori ecc) già esistenti per XML su XHML; inoltre, è possibile creare estensioni o sottoinsiemi di XHTML per diverse piattaforme.

[16] Tra gli elementi non supportati troviamo i frames, gli script di programmazione, l'uso di stili incorporati con <style>, diversi elementi di testo, oltre a vari attributi di presentazione. E ancora: sono supportati solo i moduli di base dei form, è consentito l'uso delle tabelle ma non nidificate. Fondamentale è la possibilità di definire la presentazione dei documenti con CSS esterni collegati.

vuole sviluppare un'unica versione del proprio sito accessibile dalle varie piattaforme può farlo utilizzando l'XHTML.

Accanto a questo avvio di integrazione tra web e wap, migliora l'appeal dei siti grazie anche ai display a colori, la difficoltà nel configurare i cellulari è superata con la possibilità offerta dagli operatori di ricevere sms auto-configuranti, l'alta velocità del GPRS e dell'UMTS garantisce prestazioni accettabili.

Tuttavia, i pure browser WAP 2.0 sono ancora incapaci di visualizzare direttamente le pagine web tradizionali a causa delle limitazioni del Mobile Profile di XHTML.

2.2.4 Accesso al full Web

Fino a questo momento, chi avesse voluto accedere direttamente al full web tramite il proprio telefono cellulare, non avrebbe potuto farlo.
Era comunque possibile un accesso indiretto tramite un proxy server che convertiva le pagine web in un formato compatibile con il dispositivo (es: WML, XHTML MP, ecc), ma con risultati spesso deludenti.

Arriviamo ad un punto in cui i dispositivi mobili sono capaci di accedere direttamente agli stessi siti web a cui accediamo dai browser desktop.

3 PROBLEMATICHE ACCESSO AL FULL WEB ON MOBILE

L'evoluzione dei dispositivi e delle tecnologie relativi alla telefonia mobile hanno portato ad un punto in cui un telefono cellulare può accedere agli stessi siti XHTML a cui siamo abituati ad accedere tramite PC.

Tuttavia, nel momento in cui ci si trova a voler visualizzare le normali pagine web che sono state progettate per i dispositivi tradizionali, ci si rende conto dell'esistenza di vari problemi da risolvere e di quanto questa operazione sia tutt'altro che banale soprattutto se consideriamo che negli ultimi anni le pagine web tradizionali stanno diventando sempre più complesse e pesanti.

3.1 Che cos'è una pagina web oggi?

Abbiamo provato a riflettere su che cosa siano le pagina web oggi, e abbiamo cercato di individuare quali siano le variabili che caratterizzano una pagina web:

- Tecnologie implementative
- Dimensioni
- Tipologia di contenuti

Sono stati poi individuati quali siano i possibili valori assunti da queste variabili:

- **Tecnologie implementative:**
    - XHTML
    - XHTML+js
    - XHTML +flash
    - XHTML +ajax
- **Dimensioni:**
  Per avere un'idea sulle dimensioni delle pagine web si è fatto riferimento ad uno studio disponibile sul sul web all'indirizzo http://www.websiteoptimization.com/speed/tweak/average che analizza come sia cresciuta la dimensione media delle pagine dal 2003 a oggi.
  Dato che la dimensione media delle pagine è di 312K, si è considerato un intorno di 312 (200K-424K) come pagine di dimensioni nella media, mentre alla destra e sinistra di questo intervallo sono state individuate rispettivamente le pagini grandi e quelle piccole.
- **Tipologie di contenuti (predominanti):**
    - Testo
    - Immagini
    - Forms
    - Mista
    - Tabelle
    - Links

Il tutto è stato riportato visivamente sul grafico che segue, in cui ogni tipologia di pagina è rappresentabile dai possibili incroci.

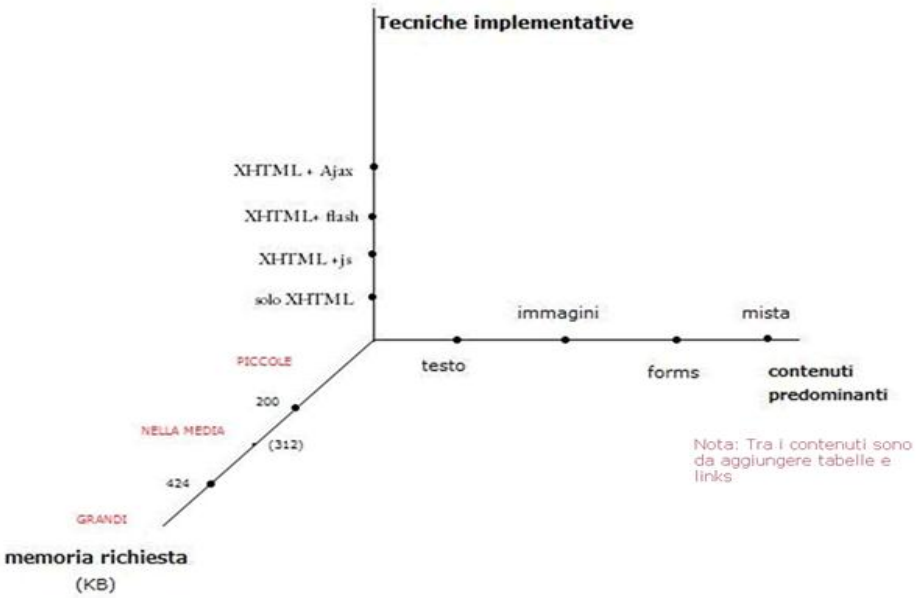

Figura 5 Tentativo di classificazione delle pagine web rispetto a 3 variabili

Benché questa classificazione necessiti di ulteriori riflessioni, si tratta comunque di un nostro primo tentativo di analisi. Ci sono infatti alcuni elementi poco chiari che ci fanno pensare che questa classificazione non sia del tutto corretta.

Per quanto riguarda le tecnologie implementative ogni pagina non è caratterizzata in modo mutuamente esclusivo da una sola delle tecnologie elencate. Ogni pagina si presenterà come un mix tra HTML e le varie possibili tecnologie (javascript, ajax, flash e altre).
Per quanto riguarda i contenuti non credo che abbia molto senso considerare i contenuti predominanti. Predominanti in che senso?- ci si potrebbe chiedere. Avrebbe più senso invece dire che una pagina è un mix tra i vari possibili contenuti.
Al momento non siamo riusciti a rappresentare graficamente il nostro pensiero o a migliorare il grafico sopra.

3.2 Caratteristiche dei dispositivi mobili e principali problematiche nella visualizzazione delle pagine web

I problemi principali che si incontrano nella visualizzazione delle pagine full web su dispositivo mobile derivano dal fatto che un dispositivo mobile non è un PC. Questa affermazione potrebbe sembrare ovvia, ma è importante partire da qui nella comprensione delle problematiche.

I dispositivi mobili differiscono profondamente dai sistemi fissi, in particolare:

- Potenza di calcolo ridotta
- Memoria ridotta
- Connessioni più lente
- Spesso l'utente deve pagare per accedere ai dati
- Diversa modalità di interazione (spesso manca il dispositivo di puntamento)
- Ampiezza del display piccola (e molto variabile)

Nei paragrafi che seguono cercheremo di analizzare quali sono le maggiori problematiche che si determinano.

3.2.1 Ridotta potenza di calcolo, memoria ridotta, connessioni più lente, costi di accesso

La ridotta potenza di calcolo e la memoria ridotta potrebbero in certi casi impedire il corretto funzionamento del browser o rallentarlo in modo significativo qualora si tenti di visualizzare pagine particolarmente grandi, complesse, o che fanno uso di tecnologie che richiedono una certa potenza di calcolo (come javascript, flash, ecc). Le stesse caratteristiche generano inoltre problemi di compatibilità tecnica: non tutte le tecnologie e formati possono essere supportate[17], il che determina l'impossibilità di accedere a determinati contenuti.

La maggiore lentezza delle connessioni rispetto a quelle tradizionali, talvolta, può comportare tempi di scaricamento lunghi per pagine di dimensioni particolarmente grandi, anche oltre i limiti dell'accettabilità da parte dell'utente. Senza contare che l'utente deve pagare per accedere ai dati e le tariffe sono solitamente a volume (in base alla quantità di byte scaricati), per cui sarebbe utile che le pagine fossero di piccole dimensioni.

Tutto questo ci porta ad affermare che uno dei requisiti che dovrebbero avere le pagine web siano **le dimensioni ridotte della pagina** e il fatto **che non richiedono eccessive risorse di calcolo.**

3.2.2 Diversa modalità di interazione

Per quanto riguarda la modalità di interazione tra l'utente e il browser, bisogna considerare che i PC sono normalmente dotati di tastiera e dispositivo di puntamento, come il mouse, e molte pagine web assumono un'interazione dell'utente tramite questi dispositivi di input.

Molti telefoni mobili non forniscono un dispositivo di puntamento (come mouse o stylus) e al suo posto forniscono un dispositivo di input che permette l'interazione tramite 5 pulsanti: due per il movimento orizzontale, due per quello verticale più un pulsante per la selezione.

Per selezionare un oggetto sullo schermo, per esempio un link ipertestuale, con questi 5 pulsanti occorre spostare il focus sull'oggetto desiderato con il movimento orizzontale e verticale e poi premere il pulsante di selezione. Solitamente, nelle implementazioni più intuitive lo stesso controllo viene usato per fare scrolling della vista e quindi focus movement e scrolling è fatto simultaneamente. Questa modalità di selezione, a differenza del mouse che consente una selezione abbastanza veloce dell'elemento desiderato, è inevitabilmente lenta paragonabile per certi versi alla navigazione tramite TAB da tastiera nel PC.

L'assenza del dispositivo di puntamento limita inoltre i possibili componenti dell'interfaccia con cui si può interagire: alcuni eventi potrebbero non funzionare.

Per quanto riguarda la tastiera, solitamente i cellulari presentano solamente un tastierino numerico, che viene usato anche per la digitazione dei caratteri. L'input testuale appare quindi estremamente lento, in parte dovuto alla maggiore possibilità di errore. Per capirlo si provi a digitare lo stesso testo con la tastiera del PC e con il tastierino numerico del proprio cellulare e si controlli il tempo impiegato.

Alcuni cellulari hanno cercato di ridurre questo svantaggio proponendo tastiere qwerty, che replicano in maniera se pur ridotta la tradizionale tastiera dei PC. Nonostante i vantaggi, la ridotta dimensione della tastiera rende la digitazione del testo comunque più lenta e meno libera da errori rispetto alla tastiera tradizionale.

In generale quindi si può affermare che l'input dell'utente nell'interazione con dispositivo mobile è lento. Per diminuire questa lentezza sarebbe utile **limitare l'input testuale** necessario da parte dell'utente nella

[17] Alle volte, alcune tecnologie (per es: flash) sono così pesanti da non poter essere implementate su alcuni modelli. Altre volte alcune tecnologie sono supportate ma fino ad un certo punto.

pagina. Anche l'uso di access-key permetterebbe agli utenti di attivare comandi con i tasti numerici del telefono (ma questo non sempre è possibile date le grandi differenze tra i vari dispositivi).

3.2.3 Ampiezza del display piccola

Infine, l'aspetto più importante: **le ridotte dimensioni dello schermo**.

È interessante notare come se alcune differenze come la potenza di calcolo, la quantità di memoria disponibile, la velocità della rete potranno crescere sempre di più fino a raggiungere quelle del PC, questo non vale per altre caratteristiche come la grandezza del display.
Il display, per quanto grande possa essere, sarà sempre più piccolo dei dispositivi fissi, in quanto se così non fosse non si potrebbe più parlare di dispositivo mobile.

In altre parole, ci sono delle inevitabili restrizioni fisiche dei dispositivi mobili, che sono essenziali per la loro portabilità. Il problema delle dimensioni dello schermo rimarrà quindi sempre un problema centrale.

L'interazione con la pagina web diviene molto più complicata e frustrante a causa delle dimensioni ridotte.

Immaginiamo di visualizzare la pagina così com'è sullo schermo di un telefono cellulare. Chiamiamo questa modalità di visualizzazione *original layout*.

Visualizzare una pagina full web su un dispositivo mobile è come visualizzare uno schermo desktop attraverso un rotolo di carta (quadrato anziché tondo) come in figura.

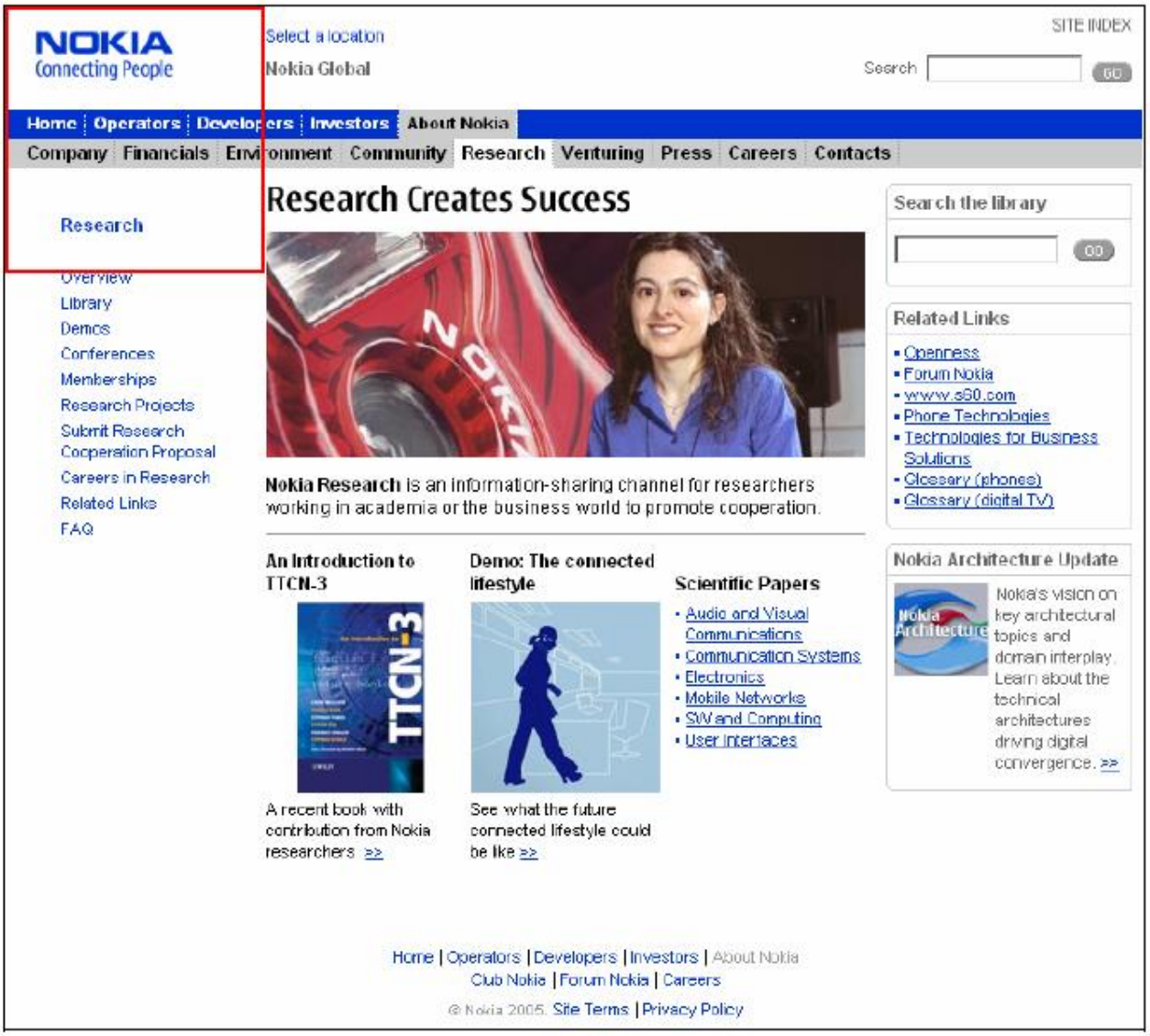

Figura 6 Layout originale di una pagina Web (www.nokia.com/research)

Le maggiori problematiche che si incontrano sono dovute al fatto che solo una porzione limitata della pagina è visibile in un determinato istante:

- è facile perdersi: è difficile sapere dove ci troviamo nella pagina e l'utente si sente facilmente perso. Se la pagina contiene una grande quantità di spazi bianchi, l'utente si trova a fare scrolling attraverso aree completamente vuote e la sensazione di essersi perso è perfino più forte.
- Le colonne di testo sono difficili da leggere se si estendono oltre lo schermo (necessità di scrolling orizzontale)
- Altre possibili difficoltà nello svolgimento di determinati task con i contenuti della pagina Web (es: visualizzazione di una immagine grande)

La domanda sorge spontanea: come facciamo a visualizzare grandi pagine Web su schermi così piccoli in una modalità maggiormente user-friendly?
Questo sarà l'argomento del prossimo capitolo.

4 ALLA RICERCA DI SOLUZIONI: L'ADATTAMENTO AUTOMATICO

Nel capitolo precedente abbiamo analizzato gli elementi che caratterizzano un dispositivo mobile e lo rendono diverso dai dispositivi tradizionali. Abbiamo inoltre analizzato quali problematiche essi potrebbero comportare quando si tenta di visualizzare una pagina web.

Ad una analisi più accurata, possiamo distinguere tra due categorie di problemi:

- *Problemi di compatibilità*
    - Tecnologie non supportate
    - Formati immagini, video ecc non supportati
    - Dimensioni delle risorse eccessive
- *Problemi di usabilità:*
    - Difficoltà nell'interazione con la pagina da parte dell'utente
        - Difficoltà di orientamento all'interno della pagina
        - Difficoltà di lettura di colonne di testo
        - Difficoltà a svolgere alcuni task con altri tipi di contenuto (tabelle, immagini)
    - Lentezza nell'interazione
        - Difficoltà di navigazione e interazione con la pagina
        - Connessioni più lente
        - Caricamento della pagina lento
        - Input testuale lento

La soluzione ad alcuni di questi problemi potrebbero essere cercate attraverso lo studio di **tecniche di adattamento** a dispositivo mobile, ovvero di tecniche che consentano di modificare in maniera automatica la pagina web in modo da ottenerne una più adatta al dispositivo mobile.

Per quanto riguarda la prima categoria di problemi – quelli di compatibilità – gli scopi dell'adattamento potrebbero essere:

- Riduzione del peso della pagina: tra le varie tecniche che si possono utilizzare abbiamo la compressione di immagini, eliminazione di contenuti ritenuti poco rilevanti, splitting della pagina
- Conversione di formati supportati in formati non supportati (*transducing*)

Si noti però che per quanto riguarda le tecnologie non supportate l'adattamento automatico può fare poco o niente. Se una pagina contiene, per esempio, un oggetto flash e il browser non supporta tale tecnologia, il contenuto veicolato tramite questa tecnologia non è accessibile. La soluzione a questo problema si trova nella progettazione di pagine che tengano conto dei problemi di accessibilità di alcuni browser e forniscano contenuti equivalenti alternativi.

Se l'adattamento automatico si rivela particolarmente adatto a risolvere questa prima categoria di problemi, più complesso e quindi più interessante da trattare è lo studio di tecniche di adattamento che consentono di migliorare l'usabilità della pagina.

4.1 Il concetto di usabilità e la sua importanza

Più volte, fino ad adesso abbiamo usato il termine *usabile* o *usabilità* senza mai definirlo precisamente.

Esistono molte definizioni di usabilità. La più importante e riconosciuta a livello internazionale è quella fornita dallo standard ISO 9241 che definisce i requisiti ergonomici per il lavoro di ufficio con i videoterminali.

Esso definisce l'usabilità come la misura in cui un prodotto può essere usato da specifici utenti per raggiungere specifici obiettivi con:

- *Efficienza*: sono le risorse spese in relazione all'accuratezza e completezza degli obiettivi raggiunti (per esempio, il tempo richiesto)
- *Efficacia*: è l'accuratezza e completezza con cui gli utenti possono raggiungere i loro obiettivi
- *Soddisfazione*: è il comfort e l'accettabilità del sistema per gli utenti e le altre persone influenzate dal suo uso.

Il concetto di usabilità è emerso soprattutto negli ultimi anni , quando l 'informatica, nata come strumento di calcolo elitario, in un mondo dove il progettista di un'applicazione coincideva con l'utente finale, è diventata uno strumento nelle mani di tutti[18].

In tale contesto è diventato importante capire come fare per ottenere sistemi informatici che siano facili da usare da parte degli utenti. Si intuisce cioè l'importanza di porre al centro dell'attenzione progettuale l'utente, e la relazione che esso instaura con il prodotto software. Nasce una disciplina specifica, la Human Computer Interaction, la quale si occupa di studiare le modalità con cui l'utente si relaziona e interagisce con quanto gli viene proposto attraverso lo schermo del computer. Il fine è quello di individuare metodi e tecniche di progettazione che consentano lo sviluppo di sistemi interattivi che siano usabili e che facilitino le attività umane.

L'usabilità di un prodotto software è un aspetto importante dato che:

- Può aumentare l'efficienza degli utenti e quindi la produzione nelle aziende e organizzazioni
- Si riducono le possibilità di errore e quindi aumenta la sicurezza
- Si riduce il bisogno di training degli utenti (riduzione dei costi)
- Aumenta l'accettazione degli utenti nell'uso di applicazioni informatiche
- Aumentano le vendite dato che l'utente sceglie il sistema che è più facile da usare.

Una volta progettato un prodotto è importante inoltre saper valutarne l'usabilità per individuare eventuali problematiche da risolvere successivamente. Vedremo meglio quest'ultimo aspetto nel capitolo 7.

4.2 Tecniche di adattamento per l'usabilità

4.2.1 Tecniche di adattamento per la visualizzazione su schermo piccolo.

Riassumendo, le principali problematiche di usabilità che si determinano dalla visualizzazione di un sito web così com'è (original layout) sul piccolo display del dispositivo mobile sono:

[18] La pervasività dei sistemi informatici in ogni attività umana e nella nostra vita quotidiana è tale che si potrebbe parlare di ubiquitous computing e di umanesimo dell'informatica.

- Il fatto che le colonne di testo siano difficili da leggere se si estendono oltre lo schermo
- Difficoltà di orientamento nella pagina

La prima soluzione che ci potrebbe venire in mente è quella di ridurre la dimensione del sito in modo che rientri totalmente nel display. Tuttavia, il semplice *scaling* non risolve niente: gli elementi della pagina appariranno quasi sicuramente troppo piccoli per potervi interagire.

Sono state proposte altre tecniche di adattamento che prevedono una trasformazione della struttura della pagina web (*trasforming*). Di seguito vedremo le più importanti.

4.2.1.1 Narrow Layout

La soluzione adottata da Opera Mini, meglio conosciuta come Single Coloumn o narrow layout, nasce con l'obiettivo di superare i problemi dell'original layout (difficoltà di lettura del testo, necessità di scrolling orizzontale e perdita di orientamento).

Essa prevede che il contenuto della pagina sia formattato in modo da adattarsi alla larghezza dello schermo:

- L'ordine del contenuto segue l'ordine con cui si presenta nel file di markup
- Il testo è impacchettato
- Grandi immagini sono scalate alla larghezza dello schermo

Così facendo, se da un lato il testo diviene più facile da leggere, e si elimina l'esigenza di fare scrolling orizzontale nella pagina, dall'altro si determinano altri svantaggi:

- L'utente non riesce a navigare conoscendo la locazione del contenuto, perché nessuno può sapere dove il contenuto del layout originale appare nel layout narrow.
- Contenuti che devono rimanere larghi come tabelle e mappe sono spesso impossibili da leggere in narrow layout.
- Forte aumento della lunghezza della pagina e necessità di molto scrolling verticale.

Come possiamo apprendere dalla tabella seguente *original layout* e *narrow layout* sono due metodi complementari: il secondo sembra risolvere gli svantaggi del primo, ma così facendo si perdono i vantaggi del layout originale.

Tabella 1 Vantaggi e svantaggi dell'original layout e del narrow layout a confronto (fonte: l'autore)

ORIGINAL LAYOUT	**NARROW LAYOUT**
mostra la pagina come è mostrata su PC, nella forma in cui è stata originariamente progettata dall'autore.	il contenuto è formattato per adattarsi alla larghezza dello schermo: - L'ordine del contenuto segue l'ordine con cui essa è presentata nel file di markup - Il testo è impacchettato - Grandi immagini sono scalate alla larghezza dello schermo

Vantaggi	Svantaggi	Vantaggi	svantaggi
La pagina appare familiare e la posizione del contenuto è quindi facile da localizzare ipotizzando la locazione del contenuto.	Le colonne di testo sono molto difficili da leggere se si estendono oltre lo schermo	Il testo è sempre facile da leggere	L'utente non riesce a navigare conoscendo la locazione del contenuto, perché nessuno può sapere dove il contenuto del layout originale appare nel layout narrow.
I contenuti non vengono riformattati (possibile leggere tabelle e mappe)	È facile perdersi: visualizzare full sized Web content su dispositivi mobili è come vedere uno schermo desktop attraverso un tubo di carta: è difficile sapere dove il contenuto che ci serve si trovi ed è facile perdersi. (> se tanti blank)	Contenuto compatto senza molto spazio bianco. Eliminazione bisogno scrolling orizzontale	Contenuti che devono rimanere larghi come tabelle e mappe sono spesso impossibili da leggere in narrow layout.

Proprio per i problemi di entrambe le modalità di layout, molti mobile browsers forniscono entrambe le modalità di visualizzazione.

4.2.1.2 Tecniche basate sull'individuazione di contenuti correlati e loro riorganizzazione

Esistono soluzioni che cercano di identificare contenuti nella pagina strettamente correlati, ognuno dei quali forma un "topic", e di riorganizzare questi topics in uno stile più adatto per il mobile browsing[19].

Gli approcci per l'inviduazione di contenuti correlati possono essere classificati in tre categorie:

a) Structure based approaches, che analizzano gli elementi HTML e la struttura della pagina Web
b) Layout-based approaches, che analizzano il layout della pagina Web
c) Approcci ibridi

Gli approcci structure-based possono fallire nel caso di pagine non strutturate correttamente, mentre gli approcci layout-based possono fallire in caso di layout inconsistenti.

Una volta che questi topics sono individuati, il layout originale può essere adattato a una differente presentazione.

[19] Efficient Web Browsing on Small Screen 2008

4.2.1.3 Tecniche basate sulla eliminazione o sommarizzazione dei contenuti

Esistono poi soluzioni che cercano di eliminare automaticamente dalla pagina contenuti non rilevanti[20]. Alcuni contenuti sono semplici da eliminare (script, applet, ecc), altri no.

Questo implica la ricerca di tecniche che consentano di individuare tali contenuti.

La selezione dei contenuti da mostrare può avvenire anche in base ai contenuti maggiormente guardati dall'utente; gli altri, considerati meno importanti per l'utente, sono eliminati oppure rimpiazzati da links o sommari.

Notare che questa tecnica, riducendo il numero di contenuti all'interno della pagina, ne consente anche una riduzione in dimensioni, rendendone più veloce il download e la visualizzazione da parte del browser.

4.2.1.4 Risolvere la lentezza nell'interazione

La lentezza nell'interazione, abbiamo detto, è determinata da vari fattori: le difficoltà degli utenti nello svolgere i task con un determinato dispositivo, le connessioni lente, le ridotte capacità computazionali.

La risoluzione dei problemi legati alla visualizzazione della pagina su schermo piccolo renderebbero gli utenti più efficienti nello svolgere i loro task, e di conseguenza più veloci.

I tempi di attesa lunghi dovuti in parte alle connessioni lente, alle ridotte capacità computazionali che rendono il caricamento della pagina lento, potranno trarre beneficio da una riduzione delle dimensioni della pagina.

Poco può fare l'adattamento automatico per rendere più veloce l'input testuale.

4.3 Come far fronte all'enorme varietà di dispositivi nell'adattamento?

È comunque importante notare che è impossibile identificare una strategia risolutiva unica di adattamento, dato che i dispositivi sono molto diversi tra loro. Anche modelli simili variano in alcuni parametri (es: dimensioni del display).

Data l'enorme varietà delle caratteristiche dei dispositivi, sarebbe quindi utile un adattamento sulla base delle caratteristiche del dispositivo.
Questo implica, per prima cosa, la necessità di individuare il dispositivo e le sue risorse prima di effettuare l'adattamento (soprattutto nel caso si adottino strategie server o proxy-side).

L'identificazione del dispositivo avviene sfruttando un'informazione contenuta all'interno del pacchetto http, detta User Agent. Esso consente di identificare il dispositivo che sta facendo la richiesta, il browser utilizzato, il sistema operativo e altre informazioni.

Una volta individuato il dispositivo è possibile identificarne le risorse su un server contenente la descrizione formale delle risorse di tutti i dispositivi.

Tali descrizioni si basano sul cosiddetto CC/PP (Composite Capabilities/Preferences Profile) del W3C. Si tratta di un'insieme di regole suggerite dal W3C per la creazione di schemi di descrizione dei dispositivi (preferibilmente usando il linguaggio RDF).
Due applicazioni del paradigma CC/PP sono:

[20] Collapse-to-Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content

UAPROF: consiste nella raccolta di descrizioni formali di vari dispositivi presenti nel mercato. Ogni descrizione presenta un URL. I dispositivi mobili inviano un header dentro all'http request contenente l'URL al suo UAPROF.

WURFL: progetto italiano che mira a costruire una libreria liberamente scaricabile di descrizioni formali. Consiste in un file XML che può essere memorizzato localmente e contiene una vasta gamma di informazioni riguardanti le caratteristiche di molti dispositivi.

4.4 Luoghi di adattamento

Da un punto di vista architetturale vi sono varie soluzioni per l'adattamento automatico, che si differenziano in base al luogo dove l'adattamento viene eseguito:

- sul server
- su un proxy-server
- sul dispositivo client

Nel primo caso l'adattamento avviene sul server dove risiede l'applicazione richiesta. Il dispositivo client invia una richiesta di accesso (contenente anche l'indicazione delle capacità del client) ed il server restituisce il contenuto opportunamente adattato. Il limite di questa soluzione è data dal fatto che i software per l'adattamento devono essere presenti su tutti i server applicativi.

Questo può essere evitato con l'utilizzo di un proxy server. Questi riceve la richiesta di accesso e l'indicazione delle capacità del dispositivo client, poi gira la richiesta al server applicativo che invia i contenuti. A questo punto, il proxy esegue l'adattamento e invia i contenuti adattati al client.

La terza possibilità è quella di eseguire l'adattamento direttamente sul dispositivo client. Un esempio è il narrow layout fornito dal browser Opera. Il problema è che molti dispositivi mobili hanno limitate capacità di calcolo, memoria ed ampiezza di banda. Bisogna però considerare che le capacità dei dispositivi mobili sono in continua crescita.

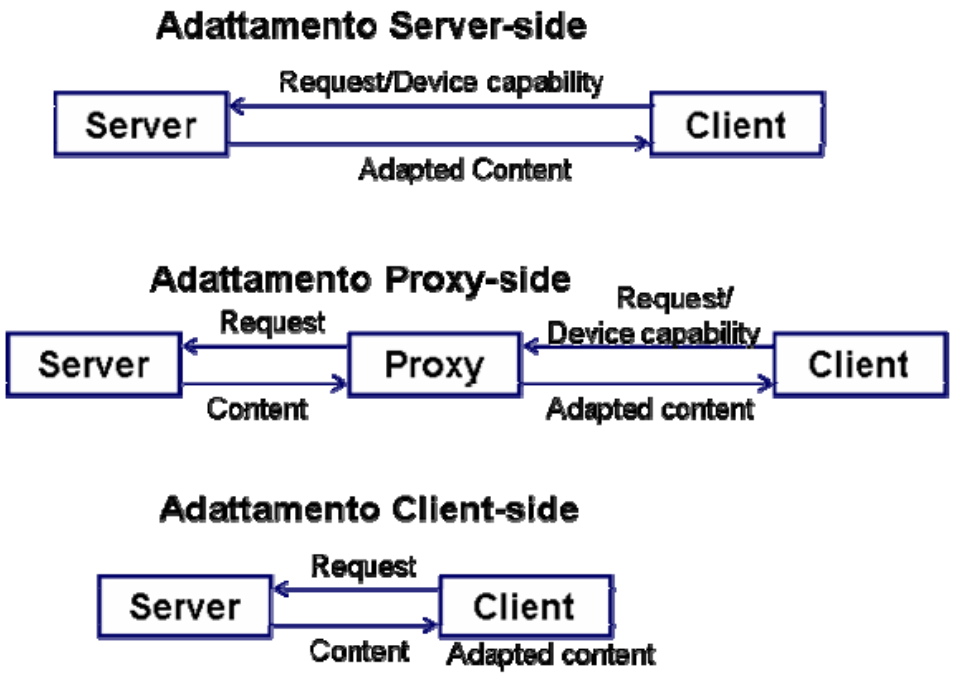

Figura 7 Possibili soluzioni architetturali per l'adattamento

4.5 Limiti dei sistemi di adattamento visti sopra[21]

I sistemi di adattamento basati sul transducing che abbiamo analizzato sopra presentano un significativo problema: essi modificano il layout della pagina web semplificando, cancellando o riorganizzandone i contenuti.

Chi decide di accedere al full web, se pur da dispositivo mobile, desidera che i contenuti siano gli stessi a cui è abituato quando usa il PC. Se il layout di una pagina Web viene cambiato, **gli utenti non possono riferirsi alle loro precedenti esperienze di browsing su PC desktop**.
Per esempio, gli utenti potrebbero essere abituati alla presenza di un menu list sul lato sinistro della pagina Web; tuttavia, se il layout viene modificato e questa funzionalità è rimossa o è diversa, gli utenti potrebbero non essere capaci di visualizzare in modo piacevole la pagina web.

Inoltre, basandosi su criteri sintattici, questi metodi non tengono conto della semantica dei contenuti e possono determinare **soluzioni poco usabili.**
Infine, la modifica del layout delle pagine web **potrebbe andare contro le intenzioni degli autori** delle stesse, che le hanno progettate in quel modo e desiderano che arrivino all'utente in quel modo. Questo aspetto è ancora più evidente se l'autore scrive "Guarda la figura a sinistra" nel testo di una pagina web: se il layout è cambiato, i lettori potrebbero non capire a quale figura l'autore si riferisce.

Crediamo quindi che siano da ricercare **approcci che mantengono il layout originale** e forniscono tecniche di visualizzazione per presentarlo in modo più confortevole.

Sembra essere questa la strada seguita da Nokia e da Apple.
Nel prossimo capitolo analizzeremo in dettaglio le soluzioni adottate nella visualizzazione delle pagine full web da parte di due dispositivi di ultima generazione:

- NOKIA N95
- Apple iPhone

Come vedremo la soluzione di Nokia si ispira a tecniche di information visualization[22], mentre iPhone adotta una diversa strategia di visualizzazione della pagina web (viewport, zoom, double tap per visualizzare contenuto allo schermo).

[21] OPA Browser: A Web Browser for Cellular Phone Users
[22] Si tratta di una disciplina che si occupa cerca di trovare strumenti, metodi, strutture, che permettano la visualizzazione di grandi quantità di dati in modo da poter essere compresi e interpretati dall'essere umano, che in quanto tale ha dei limiti. Oltre ai limiti umani, questa disciplina prende anche i considerazione i limiti tecnologici: per esempio cerca di trovare metodi per rappresentare efficacemente tutta l'informazione in caso di spazio limitato

5 LE SOLUZIONI DI NOKIA E IPHONE

Nokia N95 e Apple iPhone sono due dispositivi di punta nel mercato della telefonia mobile per quanto riguarda l'aspetto della navigazione di siti full Web. Ciò che accomuna questi due dispositivi è il tentativo di mostrare la pagina web così com'è, senza modifiche importanti[23] del layout, sfruttando qualche tecnica di visualizzazione.

5.1 La soluzione Nokia: Nokia S60 browser

Il Web Browser per S60 è un full mobile browser che riceve contenuti direttamente da un server Web e li visualizza sul display del dispositivo mobile.

5.1.1 Cenni sull'architettura

Il browser utilizza componenti open source che sono stati adattati per le piattaforme Symbian e S60.

I componenti open source includono:

- KHTML Rendering Engine from Apple Computer Inx.'s WebCore. Si tratta del software usato dal browser Safari di Apple Computer
- KWQ adaptation layer based on Apple's WebKit – KWQ è parte del WebCore
- KJS Javascript Engine from Apple's JavascriptCore – fornisce supporto Javascript
- Netscape Plug-in API from Netscape Communications Corporation

L'intera architettura del browser Nokia è mostrata in figura.

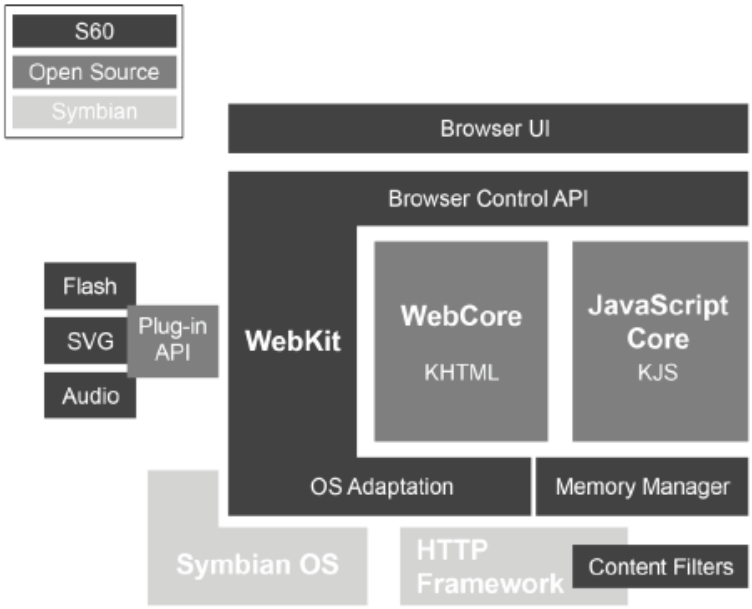

Figura 8 Architettura del browser di Nokia N95

[23] La soluzione NOKIA presenta comunque modifiche di piccola entità della pagina, che migliorano e non di poco l'interazione con essa. È il caso, per esempio, della riduzione della larghezza del testo a quella del viewport

5.1.2 Tipi di contenuti supportati

Il Browser supporta tutti i principali standard web, inclusi i seguenti:

- Web Markup and scripting
    - HTML 4.01 – incluse mappe immagine, frames, immagini di sfondo, <meta> tags, <object> tags
    - Suoni di sottofondo
    - CSS 1 e 2, inclusi fogli di stile esterni
    - Javascript 1.5
    - OMA markup and scripting – il browser rispetta lo standard OMA 2.1, che include il supporto per XHTML Mobile Profile 1.1, WAP CSS, WML 1.3, ECMAScriptMobileProfile support fot XHTML 1.1 tags.
- Images – JPEG, GIF, animated GIF, BMP, WBMP, OTA, PNG
- Localization – Il browser supporta 46 lingue. Gli utenti possono selezionare una codifica di caratteri di default tramite le preferenze, o possono selezionare l'opzione *automatic*. In questo caso il browser determina la corretta codifica leggendo nell'http header le informazioni sul set di caratteri.
- Plug-in support:
    - Scalable Vector Graphics – Tiny (SVG-T) – si tratta di una ottimizzazione da parte del corsorzio W3C per dispositivi mobili del SVG plugin. Visualizza immagini animate, grafica interattiva che può essere zoomata a varia risoluzioni e bassa memoria, bassa potenza di CPU, display limitati
    - FLASH-Lite 2.0 – Mostra .swf files, come cartoons e giochi su un dispositivo mobile
    - Browser Audio Plug-in- scarica e ascolta file audio e suoni di sottofondo in una pagina web. I formati audio includono: AMR-NB, AMR-WB, MP3,MPEG4,AAC, EAAC+, MIDI, WAV
    - Browser Video Pluging- download e visualizzazione di file video inline e stream nei seguenti formati: 3GPP, MP4, Real Video 7, 8, 9, 10
- AJAX (Asyncronous Javascript and XML) - è un insieme di linguaggi di sviluppo web e di tecniche che permettono agli utenti di interagire con le applicazioni web, senza dover ricaricare la pagina web ogni volta che una modifica viene fatta. Il Browser supporta l'oggetto XMLHttpRequest per scambiare dati asincronamente con il server web. Supporta inoltre Javascript, XML Parsing, e DOM level 2.x

5.1.3 Tecnica di visualizzazione proposta dal browser Nokia

Nel capitolo precedente abbiamo parlato delle problematiche nell'interazione con pagine web mostrate con *original layout* e di come si sia tentato di superarle per mezzo dei metodi di adattamento automatico, il più famoso dei quali –a livello commerciale- è il *narrow layout.*

Abbiamo visto come i metodi dell'original layout e del narrow layout possano essere visti come due metodi complementari: il secondo sembra risolvere gli svantaggi del primo, ma così facendo si perdono i vantaggi del layout originale. È per questo che solitamente i browser forniscono entrambe le modalità di visualizzazione.

Virpi Roto (Nokia Research), e il suo gruppo di ricerca, cercano di fare un passo in più rispetto al fornire entrambe le modalità di visualizzazione: cercano un modo per combinare in una sola modalità di visualizzazione i vantaggi di uno dell'altro.

In particolare, basandosi su studi di usabilità dei browser mobili, sono stati definiti i requisiti che un buon metodo di visualizzazione di pagine web su telefono mobile dovrebbe avere.

1) Far sì che lo schermo contenga la maggior quantità di contenuto possibile
2) Eliminare il bisogno di scrolling orizzontale mentre si legge una colonna di testo
3) Fornire abbastanza informazione contestuale per dare un'idea della struttura della pagina e comunicare la posizione corrente
4) Fornire tutte le funzionalità di base (come selezione, e selezione di links) usando 5-way control key
5) Fare tutto questo senza distruggere il layout originale della pagina
6) Fare tutto questo senza introdurre modalità

Per soddisfare questi requisiti V.Roto propone una doppia soluzione che migliora la visualizzazione di contenuto Web su un piccolo schermo.

Primo passo: attraverso un processo chiamato LAYOUT SCALING, vengono applicati alcuni cambiamenti al normale modello di formattazione CSS e al processo di "painting" del browser:

- Modificando le dimensioni degli elementi non testuali presenti nella pagina[24]
- Limitando la massima grandezza dei paragrafi di testo alla larghezza della viewport del browser.

Secondo passo: renderizzare una versione scalata (overview) della pagina Web con un'indicazione del viewport corrente e mostrarla in modo trasparente in cima al browser viewport.
Si tratta principalmente di un aiuto navigazionale: dando all'utente più contesto gli consentiamo di visualizzare la posizione corrente della viewport all'interno del documento; aiuta inoltre l'utente a localizzare informazione nella pagina.

Il layout scaling method

Il layout scaling method ha lo scopo di ridurre di un fattore di scala gli elementi non testuali presenti nella pagina e di limitare la massima larghezza dei paragrafi di testo a quella del viewport.

Esso si basa su un algoritmo che prende in input un valore indicante la grandezza corrente del viewport, il valore del fattore di scala di cui vogliamo ridurre la pagina, e fornisce in output una rappresentazione bitmap del documento.

Come risultato otteniamo che:

- Le immagini e generalmente tutti gli elementi con dimensioni statiche (immagini,HTML, tabelle…) vengono ridotti del fattore di scala specificato
- La font size rimane costante per assicurare leggibilità
- La massima larghezza del paragrafo di testo è uguale alla larghezza del viewport (elimina il bisogno di scrolling orizzontale)

[24] Ho avuto modo di scambiare alcune mail con Virpi Roto a proposito del ridimensionamento degli elementi della pagina, che mi ha precisato che "The current phones do not use the layout scaling as the screen ppi (pixels per inch) is very high."

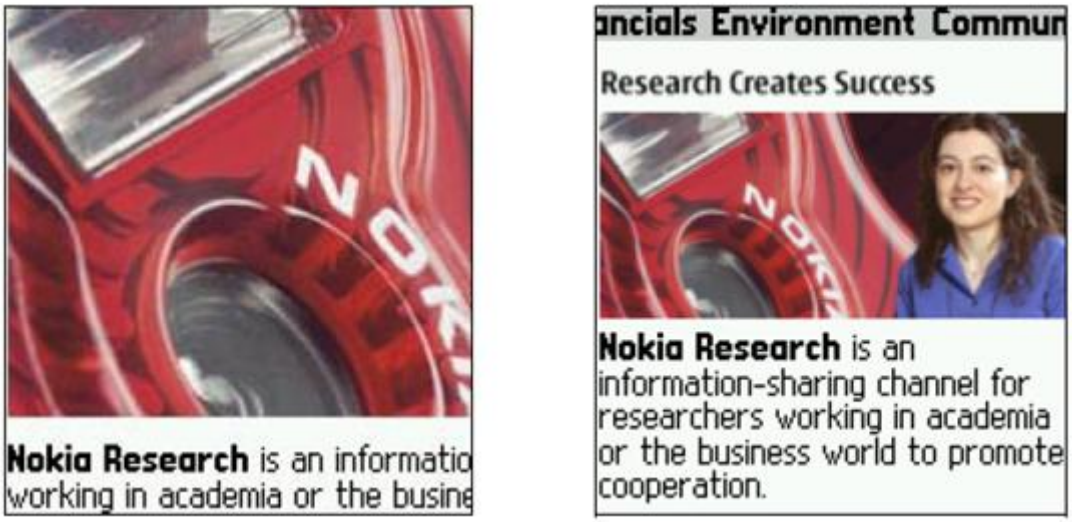

Figura 9 Visualizzazione in original layout di una pagina su un microbrowser (a sinistra), visualizzazione della visualizzazione della pagina dopo che è stato applicato il layout scaling method (a destra)

È stato osservato che la grandezza verticale dei documenti renderizzati potrebbe leggermente aumentare come risultato del layout scaling, dal momento che meno testo entra su una singola linea.
Tuttavia, spesso, l'area totale del documento spesso diminuisce a causa della riduzione delle dimensioni del contenuto non testuale.

Minimap

Per aiutare l'utente a navigare all'interno di una pagina web, gli forniamo un overview in scala della pagina. Questa nuova funzionalità, Mini Map, ha ispirato il nome dell'intero metodo.

La Mini map view è scelta in modo da contenere tutto il contenuto correntemente visibile nel view più il contesto intorno. La Mini Map è mostrata in trasparenza in alto a sinistra della pagina principale, ogniqualvolta l'utente fa scrolling orizzontale o verticale della pagina in modo continuo.
Essa appare con un leggero ritardo durante lo scrolling e scompare immediatamente quando l'utente cessa lo scrolling.

Figura 10 Minimap view visualizzata in trasparenza sulla pagina

Altre funzionalità

Altre funzionalità utili fornite dal browser sono:

- **Page overview:**
  Fornisce un overview a tutto schermo della pagina Web.
  Per certi versi è simile alla MiniMap, ma mentre la MiniMap diventa visibile automaticamente, la page overview deve essere attivato o disattivato dall'utente attraverso la pressione di un tasto del telefono.
  Inoltre, grazie alle maggiori dimensioni disponibili, Page Overview fornisce maggiore informazione di contesto.
  Muovendo il rettangolo rosso sull'overview tramite i tasti cursore, l'utente può riposizionare il viewport del browser sulla regione desiderata della pagina Web.
  Alcuni utenti hanno affermato che mentre la Page Overview funziona meglio per la visualizzazione di pagine web familiari, la Mini Map è più adatta per trovare contenuti su pagine non familiari.
- **Text search:**
  A causa delle ridotte dimensioni dello schermo, la ricerca su una pagina web è ancora più importante su schermo piccolo che sui browser desktop. Per questo viene fornita all'utente la possibilità di effettuare ricerche di stringhe testuali sulla pagina. Mentre l'utente digita, il viewport va automaticamente a visualizzare la prima occorrenza della frase ricercata. Premendo il left key verrà offerto un menu che consente agli utenti di spostare il viewport alla prossima occorrenza (se c'è).
- **Virtual Pointer:**
  Il browser Nokia fornisce un sofisticato puntatore virtuale che l'utente controlla con il joystick a 5 modalità. Vengono combinati i due metodi allo stato dell'arte: FOCUS NAVIGATION e MOUSE POINTER. In questo modo otteniamo i benefici di entrambi i metodi:
  - la pagina web effettua automaticamente lo srolling quando il puntatore si muove sulla pagina. È inoltre presente un adattamento intelligente della distanza percorsa dal puntatore ogni volta che l'utente preme i cusori in base alla distanza del più vicino elemento della pagina nella direzione del movimento . Questo permette di selezionare facilmente e velocemente un elemento focusable con una quantità minima di tasti premuti (focus navigation)
  - Abilità di posizionare liberamente il puntatore sopra qualsiasi elemento (per esempio per abilitare un menu contestuale sensibile. Questo rende possibile il supporto di contenuto dinamico come menu che usano DHTML e CSS
- **Visual history:**
  Si tratta di un miglioramento della classica funzione "indietro" e "avanti" del browser. La visual history mostra anteprime ridotte delle pagine web precedentemente visitate.. Questo permette all'utente di ritornare direttamente e quindi rapidamente a una vecchia pagina visitata senza dover aspettare che vengano ricaricate le pagine intermedie
- **Multiple document support:**
  Sebbene siano tutti d'accordo sul fatto che le pagine web non dovrebbero usare popups, ci sono siti che non funzionano senza il supporto per i popups.
  È dunque presente un supporto del multiple document, il che consente:
  - di aprire una pagina in una nuova finestra

- di aprire un link in un'altra finestra
- di navigare tra i vari documenti aperti (switching).

5.2 La soluzione di iPhone: Safari on iPhone

iPhone utilizza il web browser Safari on-Iphone: si tratta di un "full web browser" per dispositivo mobile con uno schermo ad alta risoluzione. Esso risponde alle dita come dispositivo di input e mostra pagine web sia con orientamento portrait che landscape.

5.2.1 Cenni sull'architettura

Dal punto di vista architetturale utilizza lo stesso web engine di Safari per desktop. Tutti i browser web Safari utilizzano lo stesso motore WebKit. Si tratta di un progetto OpenSource oltre che un frame work in Mac OS X che permette agli sviluppatori di includere un browser Web nelle loro applicazioni.

5.2.2 Tecnologie supportate

Webkit supporta tutti i più moderni standard web come:

- HTML 4.01
- XHTML 1.0
- CSS 2.1 e parzialmente CSS3
- ECMAScript3 (Javascript)
- DOM Level 2:
- AJAX technologies, incluso XMLHTTPRequest

In aggiunta, WebKit implementa alcune estensioni di HTML, CSS e JAVASCRIPT incluse alcune che sono specifiche per Safari on i-Phone, come:

- Proprietà CSS di trasformazione e animazione
- Supporto database javascript
- Supporto eventi multitouch

Analizziamo più nel dettaglio il supporto degli standard HTML, CSS, DOM, JAVASCRIPT da parte di questo browser.

HTML

Safari e WebKit implementano un vasto sottoinsieme delle specifiche HTML 4.01 del W3C. Notare che alcuni elementi disponibili su Safari per MAC OS X e Windows non sono automaticamente disponibili per Iphone e viceversa. Apple definisce alcune estensioni (che non fanno parte dello standard W3C). Notare che alcune di queste estensioni sono specifiche di Safari on iPhone.

CSS

Webkit supporta diverse proprietà, alcune definite dagli standard W3C ed altre aggiuntive:

- Supporto completo del CSS 2.1 (e precedenti)

  Si tratta di quelle proprietà pienamente supportate dalla maggior parte dei maggiori browser, e quindi anche da Safari
- Supporto parziale del CSS 3

  Le specifiche CSS 3 non sono state ancora rilasciate, sebbene il W3C pubblichi costantemente informazioni sulle novità in fase di sviluppo. I CSS 3 dovrebbero presentare soluzioni per la correzione di alcuni bug di interpretazione di Internet Explorer, migliorie

nella gestione degli sfondi e una soluzione per realizzare i *bordi arrotondati* la cui realizzazione affligge i webdesigner da tempo e altro ancora.
Tra le proprietà CSS3 supportate, Apple distingue tra:

- Stable CSS 3: proprietà nuove presenti in CSS versione 3 ma ritenute essere stabili. Apple si impegna a supportare queste proprietà andando avanti e non si aspetta che la loro sintassi o nome cambino. Diversamente dalle proprietà CSS 3 più sperimentali, queste non hanno il prefisso –webkit-, e molte sono supportate da altri browser come Internet Explorer e Firefox.
- Experimental CSS 3: proprietà nuove contenute in CSS3. La sintassi per queste proprietà potrebbe cambiare, ma dato che esse sono precedute dal prefisso –webkit-, Apple ritiene che la sintassi corrente potrà essere supportata in avanti.

Occorre comunque aggiornare queste proprietà alla sintassi CSS3 finale e rimuovere il prefisso webkit- se si ha bisogno di usarle su altri browser.

- Apple extension: proprietà definite da Apple. Pienamente supportate da WebKit e Safari. Notare che alcune di queste estensioni sono specifiche dei Dashboard widgets o di Safari per Iphone. Alcune di queste estensioni sono state sottomesse al W3C working group per la standardizzazione ma per il momento non sono ancora parte dell'ultima stesura dello standard.
- Sotto sviluppo: nuove proprietà CSS3, o estensioni di Apple che probabilmente cambieranno sintassi. Sebbene va bene usarle, il supporto per queste proprietà potrebbe cambiare in modo incompatibile nel futuro.

DOM

è una interfaccia software standardizzata che permette al codice scritto in Javascript e altri linguaggi di interagire con i contenuti del documento HTML. Consiste in una serie di classi che rappresentano elementi HTML, eventi e così via, ciascuno dei quali contiene metodi per operare su quegli elementi o eventi.
Il DOM, in Safari on Iphone presenta alcune estensioni. Queste estensioni includono **DOM touch events** che servono per poter gestire i gesti su dispositivi che supportano il touchscreen, e **effetti visuali** (trasformazioni 2D e 3D, animazioni, transizioni). La maggior parte di queste classi aggiuntive sono estensioni Apple, che potranno essere proposte come standard W3C.

JAVASCRIPT

Il framework WebKit di Apple, e il browser Safari che su esso si basa, supportano l'ultima versione di Javascript. Ci sono però alcune limitazioni nel tempo massimo di esecuzione di script (che vedremo dettagliatamente in seguito).Ci sono inoltre alcuni metodi Javascript non supportati da Safari onIphone, per cui è consigliabile usare finestre e dialoghi Javascript supportati (come Window.open(), Alert(), Confirm(), Prompt()) ed evitare altri come showModalDialog() e print(), che non sono supportati.

In generale, è consigliabile evitare l'uso di tecnologie non supportate:

Tabella 2 Tecnologie non supportate da iPhone e iPod touch

Area	Technologies not supported
Web technologies	Flash media, Java applets, SOAP, XSLT, SVG, and Plug-in installation
Mobile technologies	WML
File access	Local file system access
Text interaction	Text selection, Cut/Copy/Paste
Embedded video	In-place video (tapping an embedded element will put iPhone/iPod touch into video playback mode)
Security	Diffie-Hellman protocol, DSA keys, self-signed certificates, and custom x.509 certificates
JavaScript events	Several mouse-related events (see Chapter 5)
JavaScript commands	`showModalDialog()`, `print()`
Bookmark icons	`.ico` files
HTML	`input type="file"`, tool tips
CSS	Hover styles, `position:fixed`

5.2.3 Limiti di Safari on iPhone

Poiché la memoria disponibile su iPhone non è molta, ci sono limiti al numero di risorse che può processare:

Tabella 3 Limiti per le risorse

Resource	Limitation
Downloaded text resource (HTML, CSS, JavaScript files)	10MB
JPEG images	128MB (all JPEG images over 2MB are subsampled—decoding the image to 16x fewer pixels)
PNG, GIF, and TIFF images	8MB (in other words, `width*height*4<8MB`)
Animated GIFs	Less than 2MB ensures that frame rate is maintained (over 2MB, only first frame is displayed)
Non-streamed media files	10MB
PDF, Word, Excel documents	30MB and up (very slow)
JavaScript stack and object allocation	10MB
JavaScript execution limit	5 seconds for each top-level entry point (`catch` is called after 5 seconds in a `try/catch` block)
Open pages in Mobile Safari	8 pages

5.2.4 L'interfaccia grafica di Safari on-Iphone: aspetto e misure

Per default, Safari on iPhone mostra:

- una barra di stato: mostra la carica della batteria, stato della rete, l'orologio
- una URL text field: mostra un bottone bookmark e un bottone di refresh, in aggiunta al campo di testo in cui gli utenti possono digitare un URL
- una button bar: mostra i bottoni che agiscono sulla funzionalità o built-in application corrente, è la toolbar
- visible area: tra il lato più basso del campo di testo URL e il lato più alto della button bar, in cui viene mostrato il contenuto.

Notare che il campo URL se non è in uso, è ancorato sopra la pagina web e si muove con essa quando l'utente effettua lo scrolling; questo non vale per la button-bar, la cui posizione non può essere cambiata.

Figura 11 Interfaccia di Safari on iPhone e misure (portrait orientation)

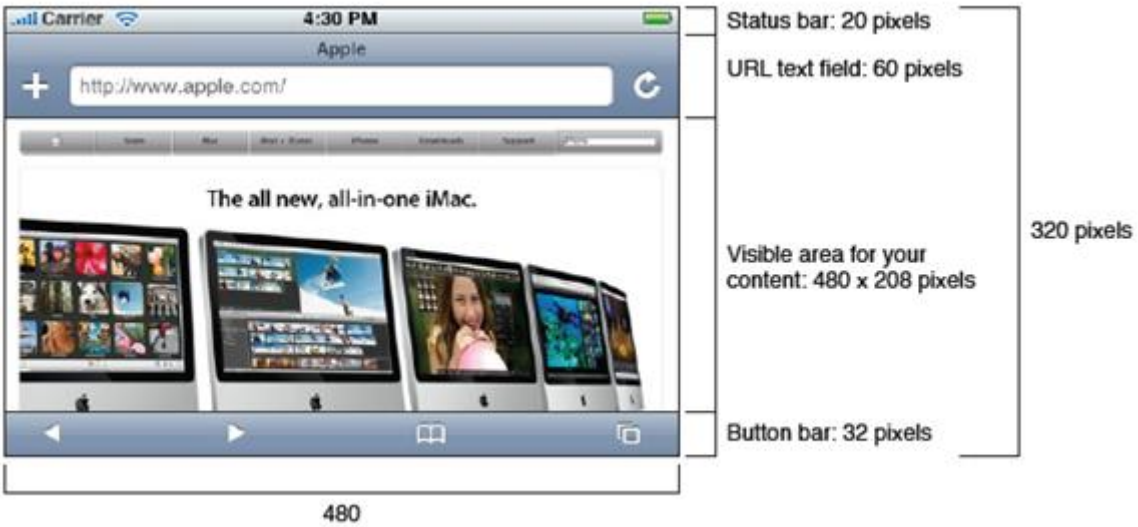

Figura 12 Interfaccia di Safari on iPhone e misure (Landscape orientation)

Si noti inoltre che quando si passa alla landscape orientation le misure cambiano, sebbene la disposizione delle varie barre e campi rimanga la stessa.

5.2.5 La soluzione di iPhone

Andiamo ora a vedere la soluzione proposta in Safari on iPhone per la visualizzazione e l'interazione con pagine web.

Per capire come si comporta Safari on iPhone nella visualizzazione dei siti Web occorre definire il concetto di viewport.

In generale potremmo definire il **viewport** come l'area visibile di una pagina web all'interno del browser.

Mentre solitamente il viewport, corrisponde alla grandezza della finestra del browser, Safari on iPhone ridefinisce il concetto di viewport.

Figura 13 Viewport in Safari per desktop

Esso non ha più a che fare con le dimensioni effettive della finestra del browser (che è fissa), ma è un generico riquadro avente certe dimensioni.

Di default la larghezza del viewport è di 980px[25]. La pagina web viene renderizzata all'interno di questo riquadro virtuale, e successivamente scalata automaticamente così che l'intera pagina si adatti allo schermo di iPhone.

Tuttavia, questi valori di default potrebbero non funzionare bene per alcune pagine web.
Per esempio se la pagina del sito web ha una larghezza molto inferiore a 980 px la pagina risultante potrebbe apparire troppo stretta come nell'esempio.

[25] Considerata la larghezza media delle pagine web

Figura 14

Viceversa, se la pagina web è più grande di 980px la pagina non sarà visibile per intero all'interno della finestra del browser ma sarà comunque navigabile per mezzo di scrolling orizzontale.

Per fortuna, Safari on iPhone fornisce un viewport metatag che permette di modificare le impostazioni di default. Tipicamente, la larghezza del viewport viene settata a quella della pagina web, in modo che tutta la pagina web sia visualizzata per intero all'interno della finestra del browser.
Per comprendere meglio il concetto di viewport si consiglia la lettura del capitolo 10.2.2.

Come si può vedere, in generale la volontà di chi ha progettato questa soluzione per la visualizzazione delle pagine web, è quella di visualizzare la pagina completa sullo schermo.
Così facendo, tuttavia, può capitare che alcuni elementi testuali o grafici possano divenire illeggibili a causa della riduzione delle loro dimensioni. Questo non è tuttavia un problema: l'interfaccia touchscreen consente di effettuare facilmente uno zoom sulla zona che ci interessa per poi tornare alla visualizzazione di partenza facendo zoom out. È possibile ripetere più volte lo zoom.

Per eliminare il bisogno di scrolling orizzontale nella visualizzazione di testo e immagini (e non solo) è possibile effettuare un double-tap sulla zona e il testo o l'immagine verranno mostrati ridimensionati alla larghezza dello schermo.

Si noti come la soluzione di iPhone non preveda nessun adattamento che modifichi il contenuto (né lato client, né lato proxy) ma solo alcune funzionalità che migliorano l'interazione con la pagina web.

5.2.6 Interazione touchscreen con il browser e con il contenuto della pagina

iPhone presenta una modalità di interazione per mezzo di un touschscreen, uno schermo in grado di recepire i tocchi che avvengono sullo schermo.

Gli utenti di Safari on iPhone svolgono sullo schermo movimenti specifici chiamati **gesti** per raggiungere particolari risultati.

Safari on Iphone è programmato per riconoscere determinati gesti all'interno di una sequenza di tocchi e "reagire" di conseguenza, con un determinato comportamento.
Per alcuni di questi gesti, (e purché l'elemento interessato rispetti certe condizioni), Safari on-Iphone invia un **evento** direttamente al contenuto della pagina web.
Tali eventi corrispondono ai tipici eventi solitamente generati dal mouse in Safari per desktop (come *mousemove, mouse down, mouseup, mouseover, mouseout, click*),) e che per questo motivo ho definito **standard** per distinguerli dagli eventi specifici di iPhone che vedremo in seguito.
Quando le dita si muovono sullo schermo, oltre agli eventi (che io ho definito standard) che emulano quelli generati dai movimenti del mouse, i DOM objects interessati ricevono degli eventi specifici di Iphone (che corrispondono a quelli solitamente ricevuti e gestiti al livello del browser).Tali eventi possono essere gestiti all'interno della pagina web, ottenendo la possibilità di personalizzare la gestione degli eventi in modo simile a quanto avviene per le applicazioni native.

Per una trattazione più completa dell'argomento si legga il capitolo 11.

5.3 Tabelle di confronto tra il browser di Nokia e il browser di iPhone

Per aver modo di confrontare in modo abbastanza veloce le caratteristiche del browser di Nokia e di iPhone, abbiamo provato a costruire le seguenti tabelle comparative. In futuro si cercherà di completarle con i dati mancanti, che al momento non siamo riusciti a reperire.

Tabella 4 Tabella di confronto tra le tecnologie supportate dal browser di Nokia e il browser di iPhone

Tecnologie supportate	Nokia Browser	Safari on iPhone
WEB MARKUP	HTML 4.01, XHTML 1.0, XHTML Mobile Profile 1.1, WML 1.3	HTML 4.01, XHTML 1.1 (con estensioni)
STYLESHEETS	CSS 1, CSS 2, WAP CSS	CSS 1, CSS 2, CSS3 (parziale)
SCRIPTING	JAVASCRIPT 1.5, ECMAScript Mobile Profile	ECMAScript 3 (Javascript)
DOM	LEVEL 2	LEVEL 2 (con estensioni)
PLUGIN	SVG-T, FLASH-Lite 2.0, AUDIO plugin (AMR-NB, AMR-WB,MP3,MPEG4,AAC,EAAC+,MIDI,WAV), Video Plug-in (3GPP, MP4, Real video 7,8,9,10)	non supportati
AJAX	Supportato[26]	supportato
IMMAGINI	JPEG, GIF, animated GIF, BMP, WBMP, OTA, PNG	JPEG, PNG, GIF, TIFF

[26] Da alcune prove fatte sembrerebbe che le richieste al server tramite metodo POST non funzionino in quanto non funziona l'impostazione degli headers, fondamentale in caso di richieste di questo tipo.

Tabella 5 Tabella di confronto tra i vincoli per le risorse sul browser di Nokia e sul browser di iPhone

Vincoli per le risorse:	Nokia Browser	Safari on iPhone
Textual file (HTML, CSS, JS)	non conosciuto	10MB
JPEG images	non conosciuto	128MB (tutte le immagini al di sopra dei 2 MB sono soggette al *subsampling*)
PNG, GIF, TIFF	non conosciuto	8MB
Animated GIFs	non conosciuto	<2MB. Se >2MB viene mostrato solo il primo frame
PDF, Word, Excel documents	non conosciuto	30MB and up (molto lento)
JAVASCRIPT stack and object allocation	non conosciuto	10 MB
JAVASCRIPT execution limit	non conosciuto	5 secondi
Open pages in browser	non conosciuto	8

Tabella 6 Altre caratteristiche utili di iPhone e N95 a confronto

Altre caratteristiche	Nokia N95	iPhone
dimensione schermo	240*356px	320*480px
area di schermo disponibile per la pagina web	dimensioni schermo (240*320px)-qualche px per alcuni elementi grafici	320*356px (portrait orientation) 480*208px(landscape orientation)
larghezza del viewport considerata in fase di presentazione	240 px - i px occupati dalla scrolling bar	980px (modificabile dal progettista)
modalità di interazione	5 pulsanti (left, right, up, down, più pulsante di selezione) che consentono di controllare un virtual pointer. La pagina effettua automaticamente lo scrolling quando il puntatore si muove nella pagina	touchscreen

6 ADATTAMENTI AUTOMATICI PER NOKIA e IPHONE?

Nella visualizzazione delle pagine web su dispositivi mobili, la ricerca e il mercato è giunta a proporre **approcci che mantengono il layout originale**, fornendo però tecniche di visualizzazione e interazione per presentarlo e interagirvi in modo più confortevole. È questo l'approccio seguito da Nokia con il suo browser Minimap e da Apple con il browser Safari on iPhone che non prevedono modifiche sostanziali del layout delle pagine.

La domanda che ci poniamo in questo capitolo è se alcune modifiche automatiche delle pagine web potrebbero migliorare l'usabilità della pagina quando acceduta da questi dispositivi.
Sicuramente per quanto abbiamo detto finora sono da evitare gli adattamenti che modificano in modo forte il layout, proprio perché si andrebbe contro la filosofia di questi browser.
Attraverso una serie di prove e riflessioni siamo arrivati alla conclusione che piccoli adattamenti alle singole tipologie di contenuto della pagina potrebbero migliorare e non di poco la qualità dell'interazione dell'utente con la pagina.

Qui di seguito esponiamo i passi dello studio che ci ha portato ad affermare quanto appena detto.

6.1 Individuazione di task

Ci siamo per prima cosa domandati quali fossero i task generici che gli utenti potrebbero voler compiere quando navigano sul Web tramite dispositivo mobile.
Partendo dalla constatazione che i task svolti con una pagina web dipendono in larga misura dal tipo di elemento della pagina web con cui si interagisce, ci siamo posti l'obiettivo di individuare i vari possibili task per le diverse tipologie di contenuto.
I diversi contenuti presi in esame sono:

- Testo
- Tabelle
- Immagini
- Forms
- Links

Nell'individuazione dei task ci siamo avvalsi, oltre alle osservazioni personali, della lettura di alcuni articoli[27]. I task individuati sono:

- **Task sulle tabelle:**
    - leggere una cella all'intersezione di una data riga e colonna
    - leggere le celle in una data riga (o colonna)
    - leggere una tabella riga per riga (o colonna per colonna)
    - confrontare un insieme di specifiche righe (o colonne),oppure
    - determinare dove un particolare valore appare sulla tabella
- **Task sul testo**

[27] **Browsing Large HTML Tables on Small Screen**, Keishi Tajima, Kaori Ohnishi
Comparing Table Views for Small Devices, Carolyn Watters, Rul Zhang, Jack Duffy
Image Classification for Mobile Web browsing, Takuya Maekawa, Takahiro Hara, Shojiro Nishio
Automatic Browsing of Large Picture on Mobile Device, Hao Liu, Xing Xie, Wei-Ying Ma, Hong-Jiang Zhang

- Lettura sequenziale di un testo
- Ricerca di una parola in un testo
- Scorrere un testo alla ricerca delle informazioni che ci interessano
- Andare velocemente alla fine del testo
- Ritornare velocemente all'inizio del testo

- **Task sulle immagini:**

  in una pagina contenente molte immagini (per es: una galleria fotografica):

  - Costruirsi una visione di insieme di tutte le immagini
  - Guardare una per volta le immagini
  - Cercare una particolare immagine
  - Visualizzare una particolare immagine "a tutto schermo"
  - Confrontare un insieme di immagini

  In una pagina contenente una mappa immagine:
  - Costruirsi una visione di insieme della mappa immagine
  - Selezionare l'area di interesse

- **Task sui form:**

  - Avere una visione di insieme della form da compilare
  - Compilare la form (e inviare dati al server)
    - Compilare un campo (il modo dipende dal tipo di controllo specifico)
    - andare al successivo
    - ripetere i primi due passi finché non si raggiunge il submit button
    - click su submit button per inviare i dati al server
  - Ricontrollare i valori inseriti
  - Correggere il valore di un campo precedentemente inserito

- **Task sui link:**

  - Ricercare un link interessante
  - Selezionare il link

6.2 Prove pratiche con i dispositivi mobili

Una volta individuati questi task abbiamo provato a svolgerli, per individuare quali fossero le problematiche di usabilità più importanti a cui va incontro l'utente nello svolgimento degli stessi.
Le prove sono state fatte su Nokia N95.

Poiché uno studio completo avrebbe richiesto troppo tempo, abbiamo per il momento focalizzato la nostra attenzione sulle tabelle e sul testo. Abbiamo individuato le maggiori problematiche e proposto delle soluzioni. Quest'ultime sono in parte personali, in parte prendono spunto dalle letture fatte.
La lettura di tali articoli ci ha infatti permesso oltre che di individuare i task, anche di approfondire quali fossero le problematiche che si incontrano quanto si desidera interagire con un determinato tipo di contenuto tramite dispositivo mobile, e di farsi un'idea di quali soluzioni sono state proposte nel corso degli anni in risposta a questi problemi.

6.2.1 Prove con le tabelle

Il browser di Nokia N95, utilizza quello che da alcuni ricercatori è stato chiamato default method. È il metodo più diretto di presentare dati tabulari su piccolo schermo. Esso consiste semplicemente nel

mostrare della pagina tutta la porzione che entra nello schermo, dando alla possibilità all'utente di effettuare operazioni di scrolling per navigare la parte restante dei dati come in figura :

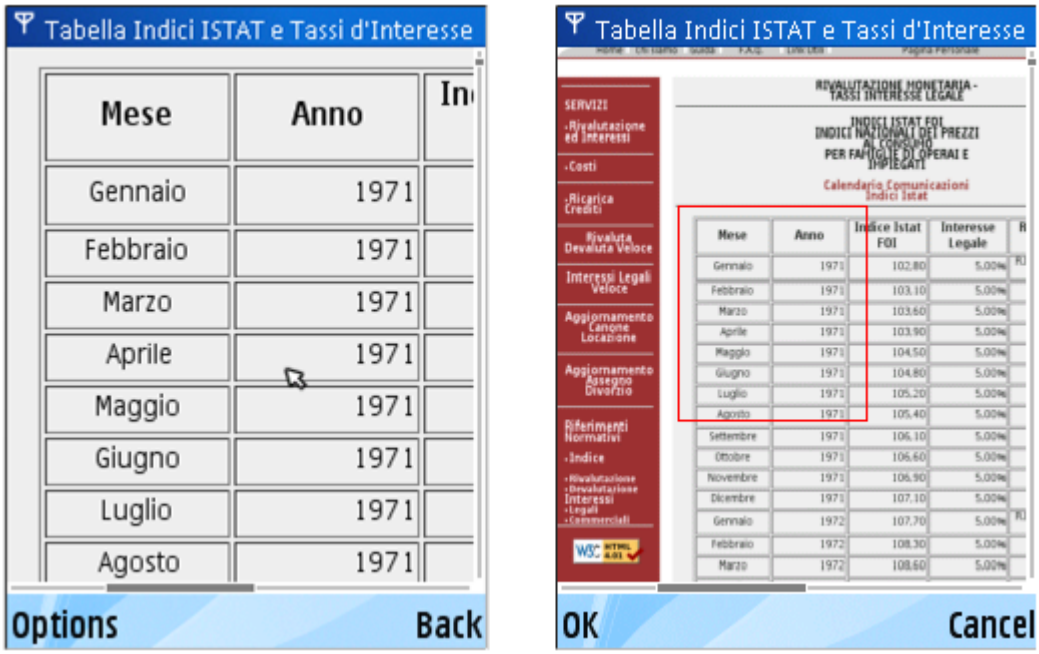

Figura 15

Per effettuare le prove è stata utilizzata la seguente tabella.

COMPOSIZIONE CHIMICA PER 100g DI PARTE EDIBILE															
Numero Codice	ALIMENTI	Parte edibile	Acqua	Proteine	Lipidi	Carboidrati	Amido	Zuccheri solubili	Fibra totale	Energia		Sodio	Potassio	Ferro	Calcio
		%	g	g	g	g	g	g	g	kcal	kJ	mg	mg	mg	mg
401010	Aranciata	100	86,0	0,1	0	10,0	0	10,0	0	38	159	tr	18		
401100	Bevanda tipo cola	100	89,0	tr	0	10,5	0	10,5	0	39	165	8	1		5
508010	Cacao amaro in polvere	100	2,5	20,4	25,6	11,5	10,5	tr		355	1486			14,3	51
508020	Cacao magro in polvere	100			8,0										
409010	Cacao, bevanda al latte	100							0						
508200	Caffe' tostato	100	4,1	10,4	15,4	28,5				287	1201	74	2020	4,1	130
503010	Dadi da brodo	100	4,6	15,7	18,7	5,0				250	1045				
504010	Farina di castagne	100	11,4	6,1	3,7	76,2				343	1437	11	847	3,2	50
504100	Fecola di patate	100	16,1	1,4	0	90,7	74,2	9,1		346	1447			0,3	10
503100	Lievito di birra, compresso	100	71,0	12,1	0,4	1,1	0	1,1	6,9	56	235	16	610	4,9	13
502760	Maionese	100	15,0	4,3	70,0	2,1	0	2,1	0	655	2741			0,5	18
900010	Minestre in scatola, crema di asparagi	100	92,9	1,0	0,7	4,2				26	109	410	50	0,3	11

Figura 16

La tabella presentata mostra la composizione chimica di alcuni prodotti.
Proviamo a visualizzare la tabella con il browser del Nokia N95:

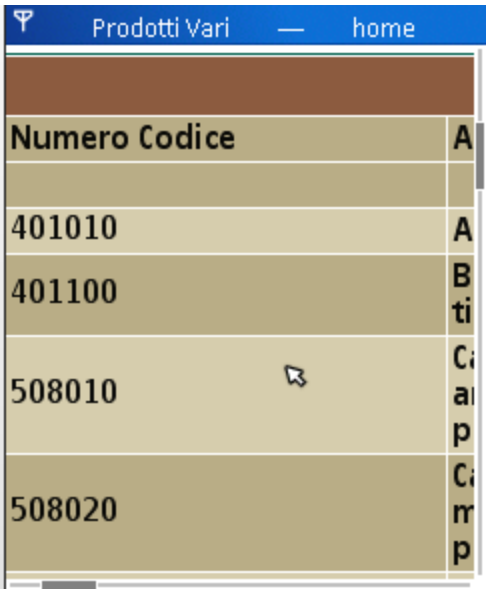

Figura 17

Come si può vedere, con il viewport posizionato nell'angolo in alto a sinistra della tabella si riesce a vedere solamente la prima colonna della tabella e una piccola parte della seconda. È possibile comunque navigare la tabella facendo scrolling a destra e a sinistra per visualizzarne altre parti.

Per cercare di migliorare la visualizzazione della tabella su dispositivo mobile, una prima operazione potrebbe essere quella di cercare di aumentare la porzione di tabella visibile. Questo si potrebbe ottenere riducendo le dimensioni delle colonne al minimo:

- La larghezza della colonna deve essere quella del testo massimo che contiene
- Prendere in considerazione la possibilità di mettere un "a capo" tra le varie parole scritte in una cella

Se ad esempio applichiamo tale operazione alla tabella presa come esempio, potremmo avere come risultato il seguente:

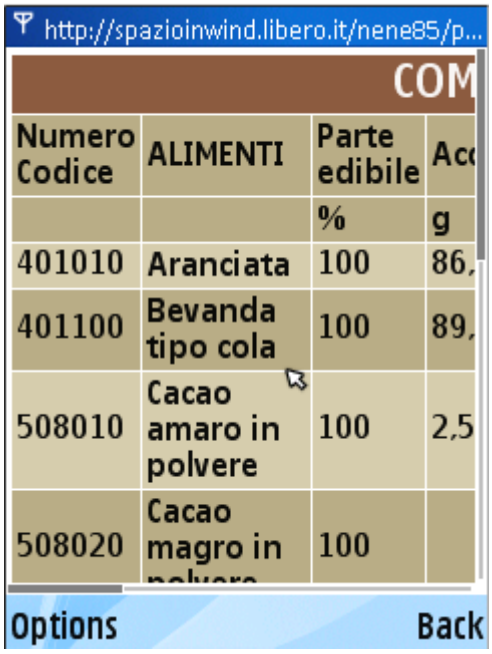

Figura 18

A questo punto abbiamo provato a svolgere i vari task individuati:

- Leggere una cella all'intersezione di una data riga e colonna
- Leggere le celle in una data riga (o colonna)
- Leggere una tabella riga per riga (o colonna per colonna)
- Confrontare un insieme di righe specifiche (o colonne) o
- Determinare dove un particolare valore appare nella tabella

TASK 1.

Individuare la quantità di ferro nella maionese.

Il task è relativamente facile da raggiungere su dispositivo desktop, dato che sia l'intestazione di colonna "ferro" sia quella di riga "maionese" sono contemporaneamente visibili:

si cerca la riga contenente l'alimento "maionese", si cerca la colonna avente come intestazione "ferro" e si va a vedere il valore presente all'intersezione.

Proviamo ad eseguire lo stesso task con il Nokia N95[28]:

1) Ci si sposta (scrolling a destra) sulla colonna avente per intestazione "alimenti"
2) Si fa scrolling verticale fino al raggiungimento della cella contenente "maionese"
   Notare che una volta eseguita questa operazione, le intestazioni di colonna che mostrano il significato di ogni colonna non sono più visibili quindi dobbiamo seguire una procedura complicata:
3) Si torna alla prima riga, si fa scrolling a destra e si conta il nuemro di colonne per arrivare a quella intitolata "Ferro" (sono 13)
4) Si rifà scrolling a sinistra per raggiungere la colonna alimenti, scroll down alla riga della maionese
5) Si fa scrolling di 13 colonne a destra per raggiungere quella del ferro e si legge il numero.

Notare che ciò che rende complicato il raggiungimento di questo task è soprattutto il fatto che **le intestazioni di colonna non sono più visibili.**

Ritorniamo al passo 2 e immaginiamo che l'intestazione di colonna si sposti con noi mentre ci spostiamo.

[28] Quello proposto è un possibile metodo. Ne esistono altri, ma il livello di difficoltà è simile.

Il task si eseguirebbe in modo molto più veloce in quanto basterebbe fare scrolling orizzontale a destra fino ad individuare la colonna con intestazione interessata quindi leggere il numero.

Questo task potrebbe essere raggiunto partendo dalla localizzazione dell'intestazione di colonna "Ferro".
Non si riporta la procedura, dato che è sostanzialmente simile a quella precedente. In questo caso il problema più grosso è il fatto che le intestazioni di riga non sono più visibili.

TASK 2.

Leggere tutte le caratteristiche nutrizionali della "maionese"(interpretandone il contenuto)
Anche in questo caso, il task è piuttosto semplice e veloce da eseguire su PC desktop, dato che sia le intestazioni di riga (a noi interessa "maionese") che di colonna (le quali ci servono per poter dare un significato ad i numeri che leggiamo nelle celle) sono contemporaneamente visibili.

Proviamo ad eseguire lo stesso task su N95:

1) Ci si sposta (scrolling a destra) sulla colonna avente per intestazione "alimenti"
2) Si fa scrolling verticale fino al raggiungimento della cella contenente "maionese"
3) Se il valore della cella successiva non è visibile, si fa scrolling a destra, altrimenti lo si legge
4) Scrolling verticale su per andare a leggere l'intestazione di riga (parte edibile)
5) Torniamo giu alla riga maionese (sempre visibile in questo caso)

   A questo punto:

6) Se il valore della cella successiva non è visibile, si fa scrolling a destra, altrimenti lo si legge
7) Scrolling verticale su per andare a leggere l'intestazione di riga (acqua)
8) Adesso torniamo giù: ma di quanto? L'intestazione di riga (se così possiamo chiamarla) non è più disponibile.
   A questo punto una possibile soluzione è questa:
   a) Si torna alla prima riga, si fa scrolling a sinistra e si conta il numero di righe per arrivare a quella intitolata "maionese" (12)
   b) Dato che ci si ricorda che questo è il secondo valore in questo caso si può abbreviare la procedura contando direttamente 2 e leggendo il valore)

A questo punto:

9) Se il valore della cella successiva non è visibile, si fa scrolling a destra, altrimenti lo si legge
10) Scrolling verticale su per andare a leggere l'intestazione di riga (acqua)
11) Adesso torniamo giù di 12 e leggiamo il valore

Iteriamo i passi 9-10-11 fino a che non arriviamo alla fine della riga.

Come si può vedere un'operazione di per sé semplicissima, diventa complicata o comunque laboriosa su dispositivo mobile, senza contare che tutto questo movimento da una parte all'altra della tabella può generare confusione e perdita di orientamento, fino a suscitare all'utente una sensazione di rifiuto.

Notare che ciò che rende complicato il raggiungimento di questo task è ancora soprattutto il fatto che **le intestazioni di colonna non sono più visibili.**
Proviamo a vedere come una soluzione simile a quella vista per il task 1, possa giovare anche allo svolgimento di questo task.
Ritorniamo al passo 2 e immaginiamo che l'intestazione di colonna si sposti con noi mentre ci spostiamo.

L'utente, una volta raggiunta la riga "maionese", dovrà solo scorrere la riga leggendo i valori e le intestazioni (che sono direttamente visibili).

Se il task richiedeva, anziché la lettura sequenziale delle righe, quella delle colonne, la procedura sarebbe stata simmetrica. In questo caso il problema più grosso è il fatto che le intestazioni di riga non sono più visibili.

TASK 3.

Leggere tutte le celle riga per riga[29] (o colonna per colonna).

Durante lo svolgimento di questo task, si incontrano **le stesse difficoltà** e si applica la stessa procedura vista per il task 2.
Solamente, a rendere ancora più pesante la situazione per l'utente, il fatto che una volta letta una riga (o colonna), si trova a dover ripetere il tormento per la riga (o colonna) successiva.
Non so quanto possa durare.

TASK 4.

Confrontare un insieme di righe.
Es: Dire quale prodotto, tra tè e caffè tostato ha la maggior quantità di lipidi.
Notare che ciò rende complicato il raggiungimento di questo task è da una parte, ancora il fatto che **le intestazioni di colonna non sono più visibili**; ma dall'altro si aggiunge un altro grosso problema: l'eventuale **distanza tra le righe della tabella** che intendiamo confrontare.

Oltre a fornire un'intestazione di colonna visibile, un aiuto allo svolgimento di questo task si otterrebbe se fosse possibile restringere le righe che non ci interessano ad esempio facendo doppio click su esse.

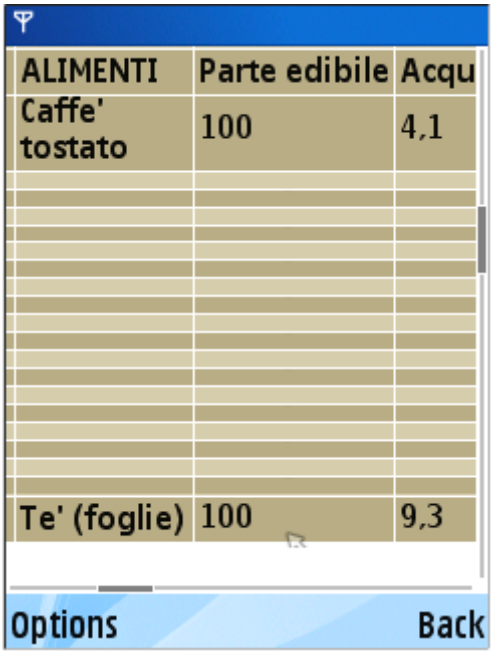

Figura 19

[29] Si tratta di un task che spesso non ha senso di per sé, ma contribuisce a svolgere task più importanti, ad esempio: dire quali sono i prodotti che hanno una quantità di ferro superiore a 10 mg e molti altri.

Certo, se le righe distano molto nemmeno questo sistema può supportare l'utente nel compiere il task, perché per quanto possiamo restringere le colonne che separano i due record essi non sono contemporaneamente visibili sullo schermo.
È quello che succede per esempio se vogliamo confrontare la riga riguardante l'aranciata e quella riguardante il tè (prima e ultima riga) come in figura:

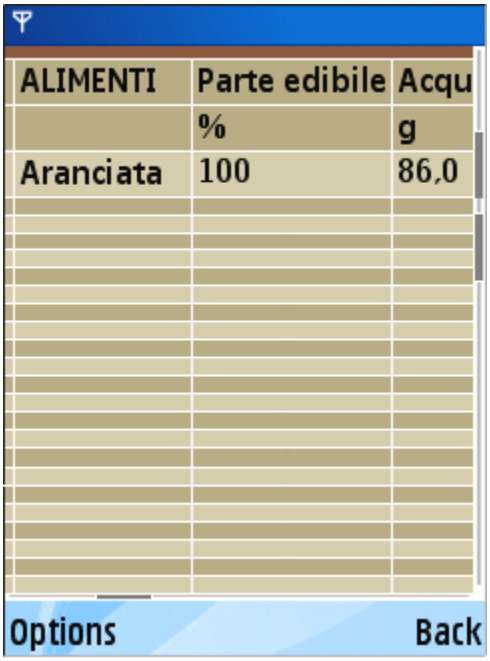

Figura 20

Una possibile soluzione sarebbe quella di fornire all'utente la possibilità di compattare le righe già ristrette in un unico blocco più piccolo (magari avente particolari stili di formattazione che lo rendono riconoscibile all'utente), ottenendo una visualizzazione del tipo

Figura 21

Nel caso il task richieda il confronto tra un insieme di righe la procedura è del tutto simmetrica alla precedente, e così anche la soluzione: intestazione di riga e colonna sempre visibili, e possibilità di restringere le colonne che non ci interessano.

TASK 5.

Determinare dove un particolare valore appare nella tabella.

Può capitare ad esempio che un utente osservi la tabella e casualmente rimanga colpito da un valore, per esempio 0.

Come fa l'utente a stabilire a quale prodotto e a quale sua componente si riferisce?

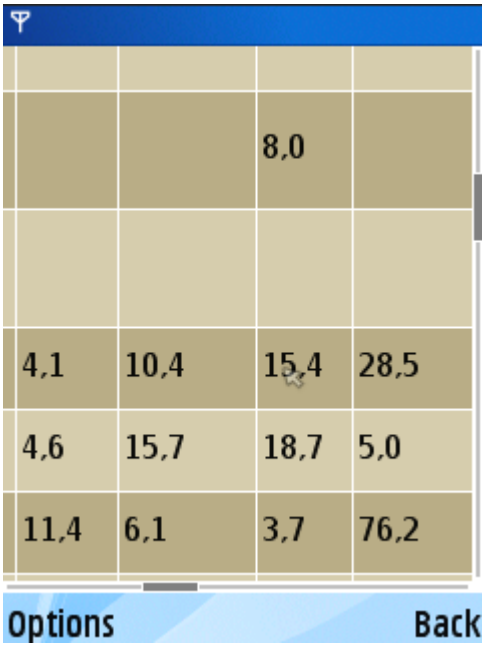

Figura 22

L'operazione è molto difficile da compiere: l'utente facilmente si sentirà perso, o ammesso che riesca a completare con successo il task, avrà sicuramente dovuto perdere molto tempo per un'operazione all'apparenza molto semplice e banale.

Anche qui il problema è che non sono visibili le intestazioni (in questo caso sia di riga che di colonna). La loro presenza agevolerebbe molto il raggiungimento del task.

Cenni sulle problematiche di implementazione

Provando a svolgere i vari task ci si rende conto che i problemi più grossi da risolvere sono:

- Intestazioni di colonna e riga non visibili
- Distanza tra le righe o le colonne

A chi voglia provare ad implementare le soluzioni proposte sopra a questi problemi, si aprono nuovi problemi:

1) Come fare a localizzare le intestazioni di colonna?

Nello standard HTML 4.0, vengono definiti due tag diversi, uno per le celle di intestazione (<th>) e uno per le celle contenenti dati (<td>).
In teoria quindi sarebbe semplice localizzare le intestazioni. Tuttavia, in molte tabelle che troviamo sul Web tutte le celle sono descritte dallo stesso tag "<td>", oppure i tag <th> sono usati non correttamente. Quindi, nella localizzazione delle intestazioni, ci si può basare sulla presenza dei tag <th> solo se questi sono presenti e se formano una struttura di intestazione appropriata.

Altre informazioni ausiliare per la distinzione tra intestazione e corpo della tabella potrebbero essere: il colore di foreground o di back down, fonts, grid lines, e i contenuti della cella nell'angolo in alto a sinistra.
Secondo alcune ricerche, tuttavia, queste informazioni sono alle volte utili e alle volte no, ma è difficile distinguere tra i due casi

2) Come localizzare le intestazioni di riga?

La situazione è ancora più complessa per le intestazioni di riga, che di solito non sono presenti esplicitamente ma è l'utente che, ragionando sul contenuto della tabella che ha di fronte, costruisce delle intestazioni di riga "virtuali".
Per esempio nella tabella che abbiamo visto non sono presenti vere e proprie intestazioni di riga, anche se noi consideriamo tali le celle contenenti il nome degli alimenti.

Per trovare un metodo che funzioni anche quando le intestazioni non sono presenti, Keishi Tajima & Kaori Ohnishi, nell'articolo "Browsing large HTML Tables on Small Screen" , propongono di basarsi sul concetto di chiave (quello delle basi di dati relazionali) e sul fatto che le chiavi hanno come caratteristica l'unicità dei valori.
In effetti, gli attributi (o le chiavi) di una tabella hanno generalmente valori distinti.
Quindi, si identificano le righe e colonne che consistono di celle con valori diversi e si trattano come intestazioni indipendentemente dal fatto che essi siano nomi di attributi o chiavi.

La procedura di "header detection" per le colonne è spiegata più in dettaglio nell'articolo sopra citato nel paragrafo "Header Detection Based on Value Uniqueness"

CONCLUSIONI

Facendo delle prove di visualizzazione delle tabelle sul browser di N95[30] abbiamo visto come sia complicato o comunque frustrante svolgere task con questo tipo di contenuti.
Si sono identificati quali sono le possibili cause di questo, e si è cercato di fornire una possibile soluzione.
Non è detto che queste soluzioni siano applicabili concretamente direttamente al browser del dispositivo mobile data la capacità di calcolo limitata di questi dispositivi; forse è meglio effettuare questi adattamente via proxy.
Non è detto inoltre che queste soluzioni siano di gradimento degli utenti: occorrerebbero dei test.
Si vuole far notare, che questa soluzione non funziona con tutte le possibili tabelle presenti sul web, per esempio quelle che presentano celle composte. Una soluzione a questo problema si trova comunque nel già citato articolo "Browsing Large HTML Tables on Small Screen".

[30] Crediamo comunque che le stesse considerazioni possano valere anche per l'iPhone.

Infine, secondo il parere di chi scrive, questi metodi potrebbero essere utili anche se applicati al browsing di tabelle su PC, dato che spesso, e inevitabilmente, queste superano le dimensioni dello schermo, determinando problemi simili a quelli visti sopra nello svolgimento dei task.

6.2.2 Prove con il testo

Sul browser NOKIA la larghezza delle righe di testo viene ridotta alla larghezza del viewport, eliminando il bisogno di scrolling orizzontale.

A differenza degli altri elementi non testuali che potrebbero essere leggermente scalati, il font mantiene la stessa dimensione, in modo da assicurarne la leggibilità. Ovviamente le dimensioni dei caratteri appariranno più piccoli se confrontati con quelli del monitor a causa di una maggiore risoluzione dello schermo del cellulare irraggiungibile dagli attuali monitor.

Si riportano qui di seguito casi in cui l'algoritmo appena citato fallisce: alcune righe di testo presentano una larghezza superiore a quella del viewport.

http://spazioinwind.libero.it/nene85/

Figura 23

http://it.wikipedia.org/wiki/Nokia_N95

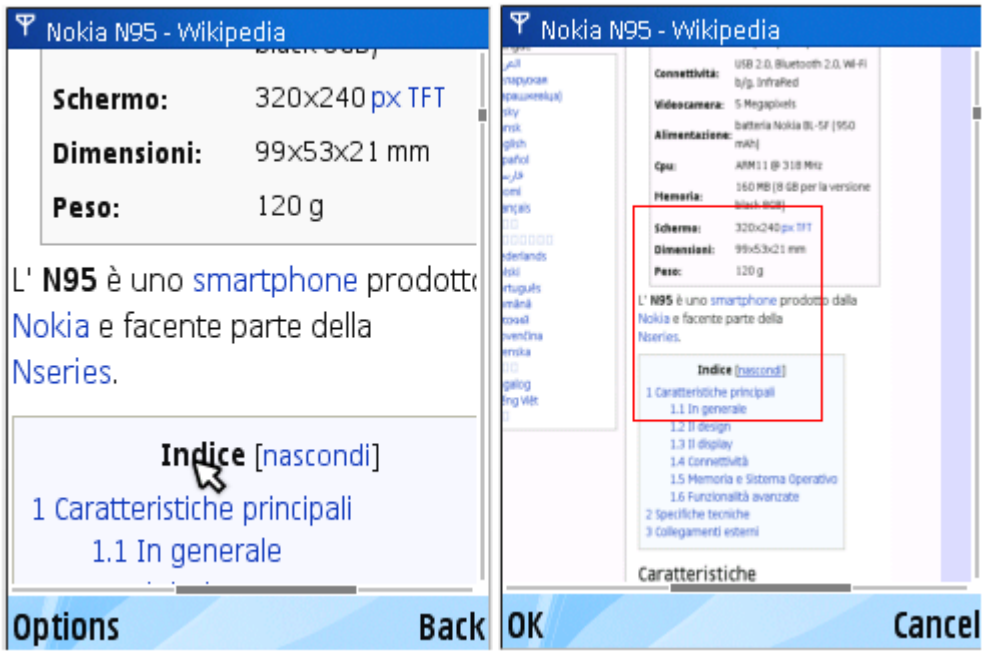

Figura 24

http://it.wikipedia.org/wiki/Italia

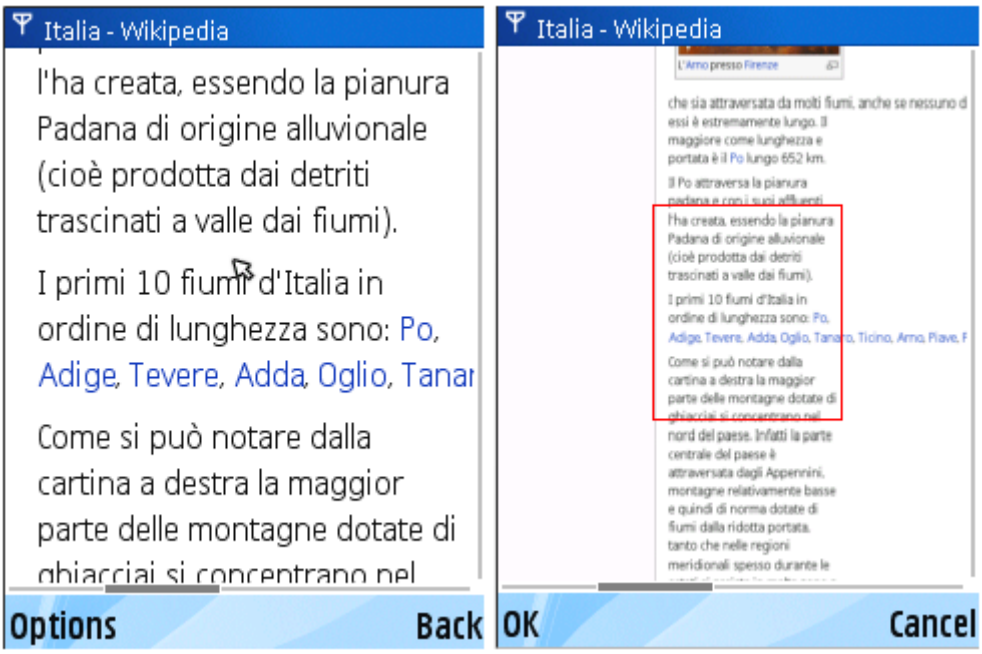

Figura 25

Figura 26

Non siamo però ancora riusciti a determinarne la causa.

Sono stati inoltre rilevati i seguenti problemi:

- Nel caso il testo contenga link particolarmente lunghi.
  http://infouma.di.unipi.it/specialistica/index.asp

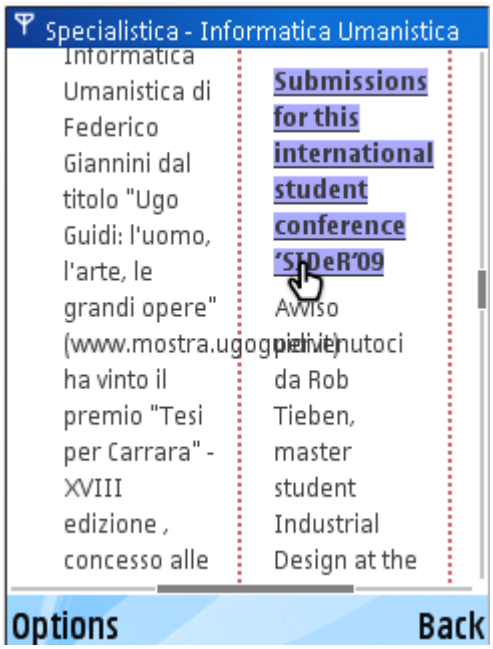

Figura 27

- Nel caso si abbiamo blocchi di testo non allineati tra loro
  http://www.fotodlflivorno.it/frontend/page.php?p=Programma

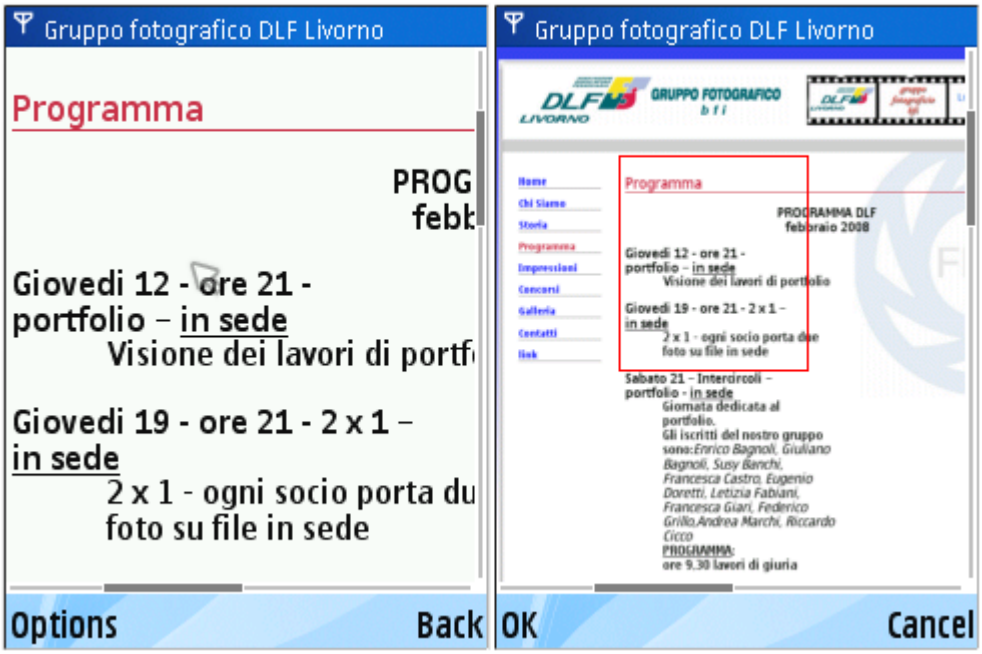

Figura 28

- Nel caso il testo contenga degli elenchi puntati
  http://spazioinwind.libero.it/nene85/freeopen.html

Figura 29

Lettura sequenziale di un testo

Nei casi in cui l'algoritmo funziona correttamente, la lettura del testo risulta essere gradevole grazie soprattutto al fatto che non è necessario lo scrolling orizzontale.
Tuttavia quando si verificano altri casi sopra menzionati la lettura potrebbe risultare più complicata o comunque richiedere maggiore interazione all'utente.

Per capire meglio abbiamo costruito una pagina ad hoc per fare delle prove.

Figura 30

Questa pagina ha la seguente struttura:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it" lang="it">
<head>
        <title>Prova</title>
        <link rel="stylesheet" media="screen, projection" href="css.css" type="text/css" />
</head>

<body>

<div id='contenitore'>
        <div id="contenuto" >
                <h1>Esempio di titolo di prova</h1>
                testo
                </div>
         </div>
</body>
</html>
```

e il seguente foglio di stile:

```
@charset "utf-8";

body{
background:url(../Grafica/bluebkg.jpg) top repeat-x;
padding:0;
}

#contenitore{
width:761px;
min-height:600px;
padding:15px;
margin-top:2%;
margin-left:auto;
margin-right:auto;
background-color:white;
border:1px solid red;
}

#contenuto{
width:500px;
padding:15px;
font-family:Arial, Helvetica, sans-serif;
font-size:12px;
border:1px solid orange;
background-color:white;
overflow:hidden;

}

#contenuto h1{
margin-top:3%;
font-size:15px;
color:#c7273f;
}
```

Proviamo a visualizzarla così com'è nel browser di N95 e vediamo cosa succede.

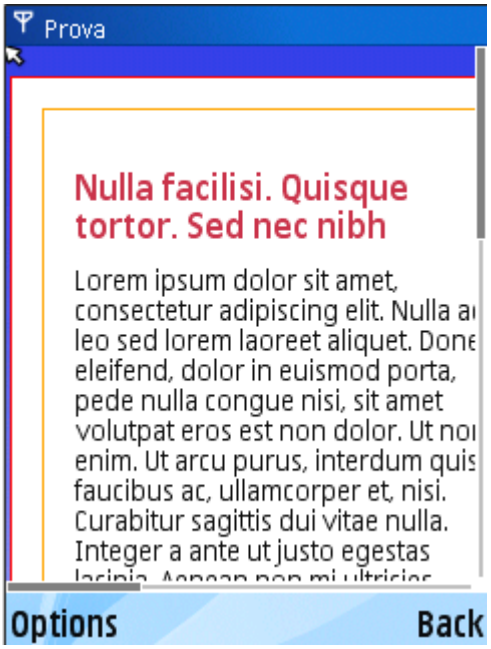

Figura 31

Una volta caricata la pagina basta spostarsi con le freccia verso destra (come nella seconda figura) per vedere che tutto il testo è perfettamente adattato alle dimensioni del viewport:

Figura 32

In base alle problematiche che abbiamo elencato all'inizio, proviamo a fare delle modifiche alla pagina:

1) **Inserisco un elenco puntato:**

Figura 33

Il problema che si viene a generare per l'utente è che una volta letta la parte iniziale del testo si trova a dover fare scrolling orizzontale a destra per la lettura dell'elenco puntato per poi ritornare a sinistra per proseguire la lettura del blocco di testo principale.

Il problema potrebbe essere risolto togliendo il padding (o comunque settandolo a quello del blocco di testo principale) tramite CSS:

```
#contenuto ul{
padding:0px;
margin:0px;
}
```

Figura 34

2) **blocchi di testo non allineati tra loro**:

Figura 35

Il problema è lo stesso visto sopra e potrebbe essere risolto allo stesso modo

3) Titolo centrato e testo allineato a sinistra

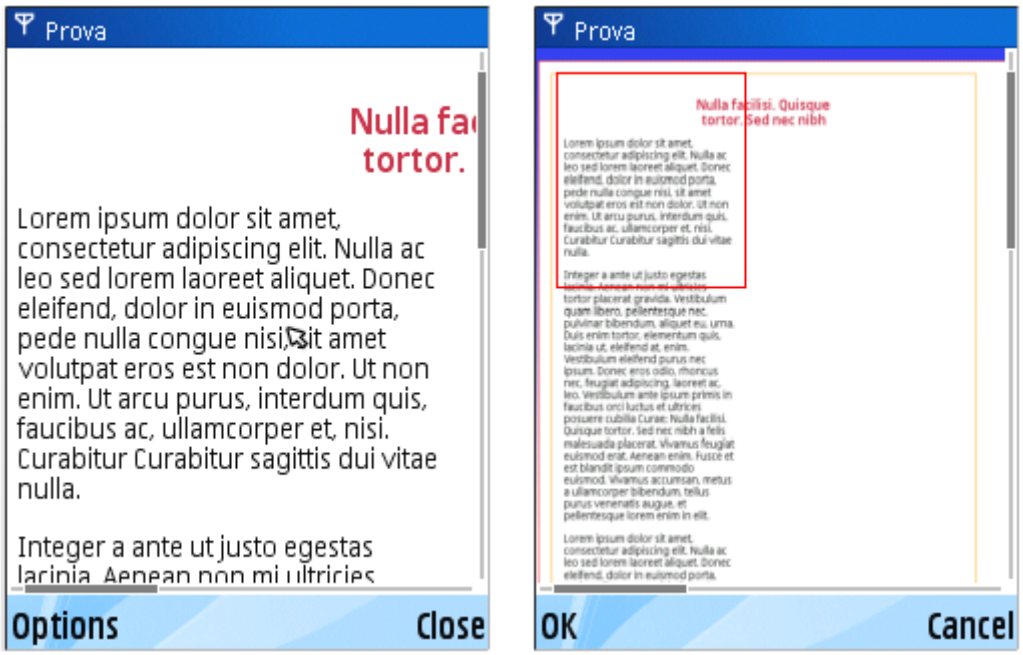

Figura 36

Ho individuato vari modi per ottenere questo effetto.

a) utilizzare l'attributo align='center' all'interno del tag h1 del titolo (anche se questo attrbuto è deprecato, è molto presente nelle pagine web).
b) Inserire il tag h1 dentro il tag center (deprecato)
c) Nel CSS applicare al tag h1 la proprietà **text-align='center'** e nella pagina HTML inserire il tag h1 all'interno di un div.

È possibile risolvere il problema semplicemente eliminando l'attributo align nel caso **a**, eliminando il tag center nel caso b, e eliminando la proprietà text-align='center' dagli stili.

Sarebbe utile trovare metodi automatici per individuare queste situazioni e correggerle prima che le pagine siano inviate al dispositivo (si potrebbe pensare a un proxy server che fa da tramite tra il dispotivo client e il server).

4) Parole o Link più lunghi della larghezza del viewport

Può capitare, anche se di rado, che una parola sia più lunga della larghezza del viewport, o più frequentemente che il testo di un link superi la larghezza del viewport dato che questi spesso riportano un testo tipo http://spazioinwind.libero.it/nene85/freeopen.html. In tal caso avremo questa situazione:

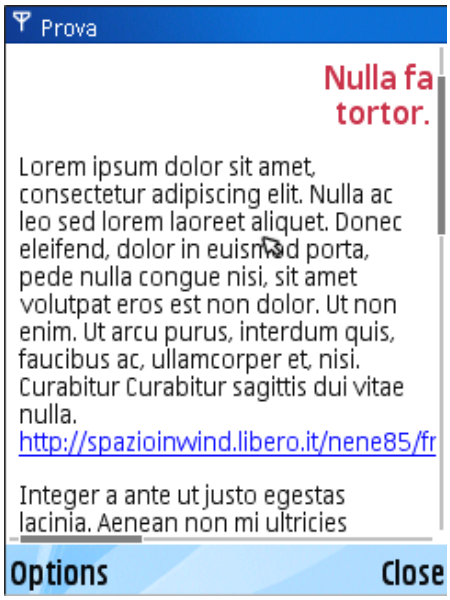

Figura 37

Una possibile soluzione che ho pensato potrebbe essere quella di determinare la porzione massima del link visualizzabile e a quel punto inserire un tag

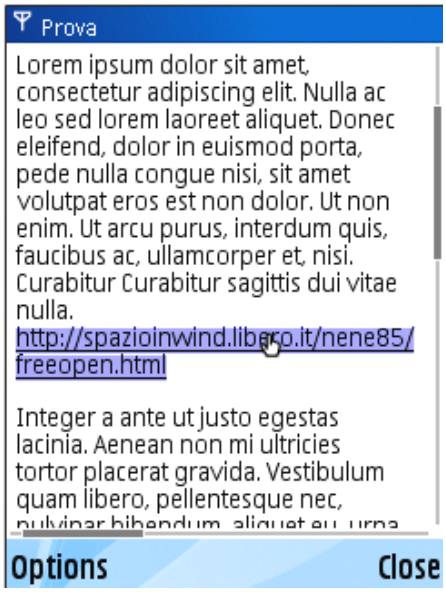

Figura 38

Come in figura, il testo del link mantiene la sua unità (per esempio se lo selezioniamo si seleziona tutto e non solo la riga su cui passiamo il cursore) pur apparendo scomposto su più righe.

Ricerca di una parola in un testo

A causa delle ridotte dimensioni del viewport del browser, la ricerca all'interno di una pagina Web è ancora più importante che su un browser per desktop.

Il browser di N95 fornisce agli utenti un modo facile per cercare velocemente stringhe di testo all'interno della pagina corrente. Mentre l'utente digita, il viewport si sposta automaticamente alla prima occorrenza della stringa di input. Premendo il "left soft key" l'utente può spostare il viewport alla occorrenza successiva (se c'è).

Proviamo a valutare l'usabilità di questa soluzione considerando due diversi **tasks**:

- La pagina mostra un elenco di nomi (incluso quello dell'utente) e l'utente vuole raggiungere velocemente il proprio.
  Questo task è così raggiungibile:
  - Si preme il pulsante opzioni
  - Si preme più volte il bottone giù fino al raggiungimento della voce "trova"
  - Si preme la freccia destra
  - Si seleziona testo
  - Si preme seleziona
  - Si digita il nome cercato:
  - Si preme chiudi
  - Si fa scrolling verticale per leggere le informazioni che ci interessano
- L'utente, dato un testo molto lungo vuole trovare velocemente tutti i luoghi in cui occorre una certa parola. Per esempio, un utente potrebbe essere interessato, data una pagina di wikipedia a proposito dell'Italia, a ricercare informazioni interessanti (per lui) su Pisa.

  Questo task è così raggiungibile:
  - Si preme il pusante opzioni
  - Si preme più volte il bottone giù fino al raggiungimento della voce "trova"
  - Si preme la freccia destra
  - Si seleziona testo
  - Si preme seleziona
  - Si digita il nome cercato

  A questo punto il browser mostra la prima occorrenza di Pisa; l'utente vuole subito farsi un'idea del contesto in cui si cita Pisa. (http://it.wikipedia.org/wiki/Italia)

  Per fare questo l'utente è obbligato a:
  - premere chiudi
  - Si fa scrolling verticale per leggere le informazioni che lo interessano

  Per andare a verificare le occorrenze successive di Pisa, l'utente deve compiere nuovamente tutti i passi visti sopra (bassa efficienza)

La mia personale opinione è che mentre questo strumento è utilizzabile per compiere il primo task, non lo è per niente nel secondo. Questa funzionalità del browser meriterebbe di essere migliorata da Nokia.

Andare alla fine (o ritornare all'inizio) del testo

Ammettiamo di avere una pagina molto lunga e che l'utente sappia che l'informazione di cui ha bisogno si trova proprio alla fine: occorre trovare un modo per raggiungere la fine velocemente senza fare lo scrolling verticale (troppo lento). Non mi sembra che venga fornita nessuna funzionalità da parte del browser per svolgere rapidamente questo task.

Una soluzione potrebbe essere quella di inserire automaticamente all'inizio della pagina un link che porti alla fine del documento. La stessa cosa si potrebbe fare se siamo alla fine e vogliamo poter ritornare velocemente all'inizio della pagina.

6.3 Conclusione finale

Riassumendo, nei capitoli precedenti abbiamo mostrato come la visualizzazione di siti web tradizionali per mezzo di dispositivi mobili di piccole dimensioni presenti alcune problematiche di compatibilità e di usabilità.

Se le problematiche di compatibilità stanno scomparendo a poco a poco con il superamento dei limiti tecnologici, restano comunque le problematiche di usabilità dipese soprattutto dalle dimensioni dello schermo.

Varie tecniche di adattamento automatico delle pagine web sono state proposte. Esse però presentano un significativo problema: modificano il layout della pagina, riorganizzandone i contenuti. Questo porta a risultati poco usabili o comunque contrari alla volontà dell'autore o del fruitore del sito web.

La ricerca e il mercato è giunta quindi a proporre approcci che mantengono il layout originale, dotando i browser della capacità di applicare tecniche di visualizzazione e/o di interazione nuove per presentarlo e interagirvi in modo più confortevole.

Benché la navigazione del full web tramite i cellulari sia sicuramente più piacevole rispetto al passato, le prove che abbiamo fatto, ci hanno consentito di individuare alcune problematiche di usabilità. Si è mostrato come alcune di queste potrebbero essere risolte effettuando piccole modifiche alla pagina ma che non ne alterano profondamente il layout. Riteniamo, cioè, che alcune piccole modifiche dei singoli elementi della pagina (es: tabelle, testo ecc) possano contribuire a migliorare ulteriormente l'esperienza di navigazione. Tali modifiche possono essere effettuate per mezzo di un proxy server. Certo, è solo un ipotesi dato che per esserne certi occorrerebbe provare a implementare questi adattamenti e fare test con gli utenti, ma ci crediamo molto.

In futuro, se sarà possibile, cercheremo di completare questo studio sull'usabilità nell'interazione con i vari tipi di contenuto da dispositivo mobile. Cercheremo inoltre di sviluppare un software lato proxy che si occupi di effettuare l'adattamento. Al momento riteniamo che i linguaggi su cui baseremo l'eventuale implementazione saranno PHP, XSL-T e Javascript e HTML.

Per concludere, riteniamo quindi che da una parte la ricerca debba puntare sullo studio di tecniche sempre migliori, di cui dotare i browser, che consentono una visualizzazione e un'interazione migliore con le pagine web senza alterarne significativamente il layout; dall'altra occorre studiare meglio le problematiche di usabilità nell'interazione con i singoli contenuti e fornire soluzioni che ne effettuino piccole modifiche.

7. STRUMENTI AUTOMATICI PER LA VALUTAZIONE DELL'USABILITA' CON DISPOSITIVI MOBILI

Nel paragrafo 4.1 abbiamo definito il concetto di usabilità e spiegato perché è importante che un sito web sia usabile.

Oltre a saper progettare applicazioni usabili, è importante essere in grado di valutarne l'usabilità in modo da poter individuare le maggiori problematiche e tentare di risolverle.

Esistono varie misure quantitative di usabilità, per esempio:

- Il tempo che l'utente impiega per svolgere un compito
- Il numero dei compiti che l'utente ha completato in un intervallo di tempo
- Rapporto tra interazioni corrette ed errori
- Numero di errori
- Quantità di tempi morti
- Ecc..

Esistono poi vari approcci alla valutazione dell'usabilità:

- *Valutazione per mezzo di test agli utenti*: i test consistono in una serie di compiti (task) che gli utenti devono cercare di svolgere. La sessione di interazione dell'utente viene memorizzata tramite telecamere o appositi software. Può essere eseguito in laboratorio, sul campo o da remoto. Oltre allo svolgimento dei task all'utente può essere richiesta la compilazione di questionari o interviste alla fine del test (feedback dell'utente)
- *Metodi di valutazione basati su modelli*: si crea un modello dell'interazione di un essere umano con un computer, e si utilizza per predire quale sarà la performance di un utente (per esempio i tempi richiesti a portare a termine il compito)
- *Valutazione basata su ispezione da parte di esperti*: l'esperto valuta il prototipo o l'implementazione finale dell'interfaccia utente secondo dei criteri predefiniti (valutazione euristica) oppure simula l'interazione di un certo tipo di utente nello svolgimento dei vari possibili task cercando di individuare le possibili problematiche che potrebbe incontrare (cognitive walkthrough)

Nel caso di metodi di valutazione che utilizzano gli utenti, può essere utile la presenza di strumenti automatici che memorizzano la sessione di test su file di log. Essi rappresentano un'alternativa o un supporto alle registrazioni video.

Tra i vantaggi degli strumenti automatici:

- possibilità di memorizzare tutte le azioni svolte dall'utente durante il test
- possibilità di eventuali analisi automatiche dei i file di log
- non sono intrusivi
- particolarmente utili in caso si vogliano effettuare valutazioni remote o sul campo

Molti tool automatici sono stati sviluppati per valutare l'usabilità di siti web su browser desktop, mentre ce ne sono meno che sono stati pensati appositamente per i dispositivi mobili.

L'esigenza di avere un tool di logging è emersa nel corso del tirocinio svolto presso il laboratorio di interfacce utenti dell'ISTI-CNR, nel momento in cui ci era stato proposto di effettuare un test di confronto tra l'usabilità delle soluzioni proposte da iPhone e Nokia N95 per quanto riguarda il Web browsing.

È stato quindi redatto un test plan (consultabile in appendice), in cui si discutono gli obiettivi del test, e i criteri utilizzati per la valutazione dell'usabilità.

Questi ultimi prevedevano:

- La raccolta di feedback dell'utente durante il test
- Memorizzazione delle azioni svolte dall'utente durante il test

Mentre il feedback poteva essere facilmente ottenuto mediante l'uso di questionari somministrati in seguito a ciascun test svolto, per la memorizzazione del file di log si rendeva necessario l'uso di un tool apposito.
Si è posto quindi il problema della scelta di quale strumento utilizzare a questo scopo.

Gli strumenti che abbiamo preso in considerazione sono due:

- USERFLY
- Vecchio tool sviluppato nel laboratorio HIIS all'ISTI-CNR intorno al 2000

Analizziamo brevemente il funzionamento di entrambi.

7.1 Il tool userfly

Il tool userfly è disponibile sul sito www.userfly.com. Si tratta di un software in grado di catturare tutti i movimenti e click del mouse nell'interazione con i vari elementi del sito (comprese le form) che gli utenti compiono nel sito, e di ricostruire in seguito un video dell'interazione.
È possibile registrarsi come utente *free* o come utente a pagamento. Nel primo caso il software presenterà dei limiti come, ad esempio, il numero massimo di test che possono essere effettuati in un mese.
Per utilizzare il software sul proprio sito è sufficiente inserire all'interno del tag <head> poche righe di codice javascript.

7.2 Il nostro tool

L'architettura del logging tool può essere schematizzata attraverso la seguente figura.

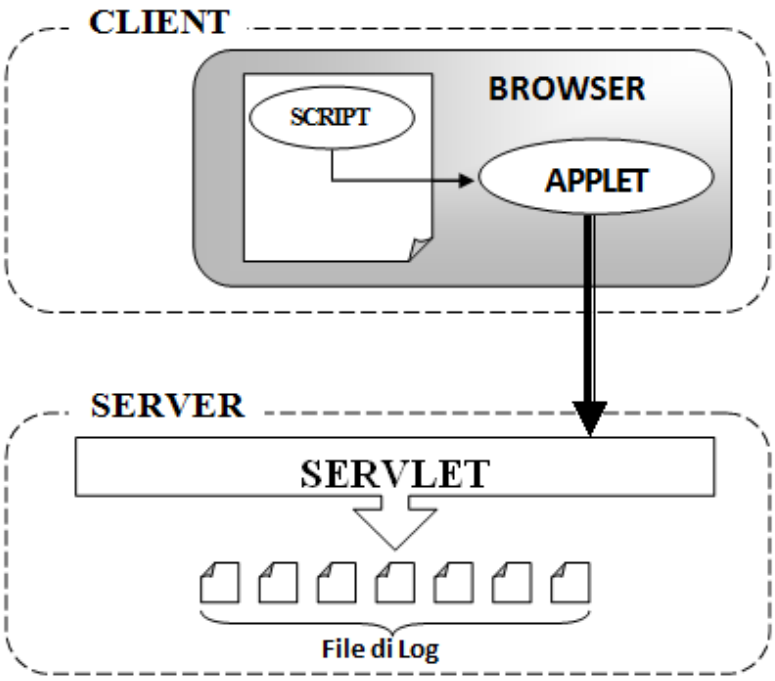

Figura 39 Architettura originaria del tool di logging (laboratorio HIIS, ISTI-CNR)

Ogni pagina del sito da testare incorpora uno script predisposto per la raccolta di tutte le azioni effettuate dall'utente sulla pagina a cui si riferisce. Per ogni evento generato, lo script comunica le informazioni all'applet, che le concatena con quelle delle pagine precedentemente accedute dall'utente.
Al termine di una sessione di test, l'applet comunica al server di origine tutte le informazioni raccolte. Sul server è stata implementata una servlet che riceve i dati dai vari client e li salva su file.

Analizziamo un po' più nel dettaglio i vari componenti:

- **SCRIPT**: inserito in ogni pagina del sito da valutare, definisce le operazioni necessarie per collezionare le informazioni relative all'evento che si è verificato.
  Oltre alle caratteristiche strettamente legate all'evento (tipo di evento, oggetto che lo ha generato, ecc.) viene associata anche una marca temporale ad ogni evento, che permette di determinare per esempio quanto tempo intercorre tra due eventi.

  Per semplicità lo script trascura gli eventi relativi al movimento del mouse e all'input da tastiera e ridefinisce i gestori solo per gli eventi:

  - Abort e error sulle immagini
  - Change sugli elementi di un form
  - Click e dbclick sui link, sulle immagini e sugli elementi di un form
  - Load e upload di una pagina
  - Submit e reset di un form
  - Resize della finestra del browser
  - Scroll (nel casi di IE)

Tutti i gestori di eventi vengono associati alla stessa funzione handler che ha il compito di estrapolare dall'oggetto Event le informazioni per il log:

- Il tipo di evento(click, change, load, ecc.)
- L'oggetto su cui si è verificato l'evento (target) e se definiti
  - Il nome associato all'oggetto (name)
  - Il valore associato all'oggetto (value)
  - L'indirizzo della pagina riferita dall'oggetto (href)
  - Ed altri dati dipendenti dal tipo di oggetto (per esempio se un radio button è selezionato oppure no)

ed associare all'evento una marca temporale calcolata tramite la funzione time. Infine invia tutte queste informazioni ad un applet java che si occupa di registrarle concatenandole con quelle relative agli eventi precedenti.

- **APPLET:** per fare in modo che l'applet una volta caricata rimanga attiva per tutta la durata del test, essa è stata inserita in un frame indipendente da quello contenente il sito da testare.
  Lo scopo dell'applet è quella di ricevere dallo script le informazioni relative alle azioni compiute dall'utente, concatenarle con quelle precedentemente ricevute, quindi inviarle al server alla fine della sessione.
  La sessione termina quando l'utente ha completato il test oppure quando l'utente clicca sul link stop. A quel punto l'applet comunica con la servlet effettuando una richiesta (POST request) attraverso una connessione http all'URL relativo alla servlet.

- **SERVLET[31]:** Prima di salvare i dati ricevuti dall'applet, la servlet inserisce in ogni file una intestazione definita dalle seguenti informazioni:
  - data e ora di ricezione del log
  - indirizzo IP assegnato alla macchian client
  - il nome dell'host della macchina client

  L'ultima operazione effettuata dalla servlet consiste nel salvare il file in una directory predefinita della macchina server.

7.3 Scelta di quale soluzione utilizzare

Il tool *userfly* è stato scartato per vari motivi:

- il fatto che è un software molto recente, in fase beta e quindi suscettibile a malfunzionamenti sui quali non abbiamo controllo

[31] Può essere definita genericamente come un'estensione del server. È rappresentata da una classe Java che può essere caricata dinamicamente ed espandere le funzionalità del server. Le servlet sono comunemente usate con i server web e possono facilmente sostituire gli script CGI e al contrario di questi ultimi, che utilizzano processi multipli per trattare programmi e/o richieste differenti, le servlet sono tutte trattate attraverso thread separati, dal processo server web. Questo significa che le servlet risultano più efficienti rispetto ai CGI. Inoltre, le servlet vengono elaborate all'interno della macchina virtuale Java (JVM) presente nel server ed operano soltanto con il dominio del server, per questo sono sicure e portabili.

- il fatto che è stato progettato per i browser desktop.
  Questo rende impossibile la registrazione di alcune operazioni che non esistono su desktop, come lo zoom effettuato tramite l'apposito gesto (*pinch)* su iPhone
- il fatto che è stato progettato per una interazione per mezzo del mouse
  Su dispositivi dove non è presente un puntatore sullo schermo, la registrazione delle azioni del mouse non sarà possibile.

Abbiamo quindi scelto di concentrare la nostra attenzione sul secondo tool. Studiandone il funzionamento e facendo delle prove ci siamo tuttavia accorti che, così come esso si presentava, non poteva essere utilizzato efficacemente per svolgere test tramite dispositivi mobili.

La prima problematica che salta agli occhi è il fatto che esso faccia uso di un' applet java . Le applet non sono supportate dai browser mobili.
Altre possibili problematiche riguardano:

- Il fatto che fosse stato progettato per browser vecchi come Explorer 4 e Netscape 4
- Il fatto che utilizzi i frame
- Possibili problemi riguardanti la visualizzazione dell'interfaccia grafica del tool su schermo piccolo
- Il fatto che non vengano rilevati alcuni gesti specifici di iPhone (come ZOOM, DOUBLE-TAP ecc), interessanti invece in un test.

Si è deciso quindi di tentare di risolvere queste problematiche per rendere utilizzabile questo tool per i nostri scopi, data anche la disponibilità dei codici sorgenti che ci permettevano di effettuare le modifiche. L'obiettivo a cui si è teso era quello di ottenere un tool che funzionasse con tutti i maggiori browser desktop del momento (Explorer, Internet, Safari, Opera) e sui browser mobili di iPhone e di Nokia N95.

Nei capitoli seguenti prenderemo in esame ciascuno dei problemi citati sopra, per ognuno dei quali si esporrà il materiale teorico che ci è stato necessario per studiare il problema e cercare una possibile soluzione, e il modo in cui è stato risolto a livello teorico e implementativo. Nonostante i numerosi tentativi, non siamo riusciti a risolvere completamente alcune problematiche. Per queste forniremo comunque una descrizione dei tentativi effettuati, così che possano servire come punto di partenza nella proposizione di eventuali soluzioni future.

8. PROBLEMA 1: COMPATIBILITA' DELLO SCRIPT DI RILEVAMENTO CON I BROWSER ATTUALI

Il tool è stato sviluppato nel 1997. All'epoca i browser più importanti erano quelli cosiddetti della quarta generazione: Internet Explorer 4 e Netscape 4. Sono dunque questi gli user-agents che sono stati presi in considerazione nella progettazione e implementazione del logging tool.

In questo capitolo ci concentreremo sulla comprensione dello script programmato all'epoca per il rilevamento degli eventi e sulle modifiche necessarie per renderlo funzionante con i browser moderni sia desktop che mobili.

Tuttavia, per essere in grado di capire uno script del genere, è necessario conoscere quali siano i *modelli di eventi* dei browser.
È su questo che focalizzeremo la nostra attenzione in questa prima parte del capitolo. Successivamente si passerà ad analizzare la soluzione adottata dal tool all'interno dello script, e mostreremo le modifiche adottate.

8.1 Modelli a eventi

In generale, tutte le applicazioni che presentano un'interfaccia grafica (GUI) utilizzano un modello di programmazione guidato dagli eventi (event-driven). Esse attendono che l'utente compia un'azione interessante (cioè che un evento si verifichi), quindi rispondono in qualche modo.

In quanto applicazione grafica, anche il browser utilizza un modello ad eventi: un evento viene generato ogniqualvolta accade qualcosa di interessante al documento o a qualche elemento in esso contenuto. Tramite Javascript è possibile programmare una risposta a un determinato evento che si verifica su un determinato elemento. Più precisamente, un'applicazione Javascript che sia interessata a gestire un particolare tipo di evento per un determinato elemento, può registrare un gestore di evento (event handler) per quel tipo di evento sull'elemento di interesse.
Un event handler non è altro che una funzione o porzione di codice javascript che viene eseguito al momento in cui quel particolare evento si verifica.

Fin qui è tutto abbastanza semplice, ma a complicare la situazione è il fatto che non esiste un solo modello per la gestione degli eventi, ma ne esistono almeno quattro, tra loro distinti e incompatibili:

- Il modello a eventi originario
- Il modello a eventi standard
- Il modello a eventi di Internet Explorer
- Il modello a eventi di Netscape 4

Analizziamo tali modelli.

8.1.1 Modello a eventi originario

Il modello a eventi originario è considerato parte del DOM Level 0.

Sebbene presenti funzionalità limitate, esso risulta essere supportato da tutti i browser abilitati ad eseguire codice javascript.

In questo modello, quando l'utente interagisce con l'applicazione web, vengono generati degli eventi. Tali eventi vengono inviati dal browser agli elementi del documento su cui essi sono avvenuti. Se l'oggetto possiede un gestore di evento, questo viene eseguito. Una volta che questo è stato eseguito, vengono compiute automaticamente alcune azioni di default[32]. Per esempio se è stato fatto un click su un link ipertestuale, l'azione di default eseguita dal browser sarà quella di seguire il link.

Un gestore di evento può essere registrato su un particolare elemento usando degli attributi HTML.
La tabella seguente illustra i diversi tipi di attributi che si possono usare, gli elementi che li supportano, e le situazioni in cui il gestore di eventi associato a questi attributi viene invocato.

Tabella 7 Attributi HTML per la registrazione dei gestori di eventi e elementi che li supportano

attributes	Triggered when	Supported by
onabort	Image loading interrupted	<img>
onblur	Element loses input focus	<button>,<input>,<label>,<select><textarea>, <body>
onchange	Selection in <select> element or other form element loses focus, and its value has changed since it gained focus.	<input>,<select>,<textarea>
onclick	Mouse press and release; follows mouseup event. Return false to cancel default action (i.e., follow link, reset, submit).	Most Elements
ondblclick	Double-click	Most elements
onerror	Error when loading image	<img>
onfocus	Element gains input focus	<button>,<input>,<label>,<select><textarea>, <body>
onkeydown	Key pressed down. Return false to cancel	Form elements and <body>
onkeypress	Key pressed; follows keydown. Return false to cancel	Form elements and <body>
onkeyup	key released; follows keypress	Form elements and <body>
onload	Document load complete	<body>,<frameset>,<img>
onmousedown	Mouse button pressed	Most elements
onmousemove	Mouse moved	Most Elements
onmouseout	Mouse moves off elements	Most Elements
onmouseover	mouse moves over element	Most Elements
onmouseup	Mouse button released	Most Elements

[32] È comunque possibile impedire lo svolgimento delle azioni di default.

onreset	Form reset requested. Return false to prevent reset	<form>
onresize	Window size changes	<body>,<frameset>
onselect	Text selected	<input>, <textarea>
onsubmit	Form submission requested. Return false to prevent submission	<form>
onunload	Document or frameset unloaded	<body>, <frameset>

Un modo alternativo di registrare un gestore è quello di impostarlo come proprietà di un oggetto del documento tramite codice Javascript, ottenendo come vantaggio la modularità e la pulizia, nonché la dinamicità (ovvero la possibilità di cambiamento) dei gestori di eventi.
Nel DOM Level 0 Event model, quando viene registrata una funzione come gestore di un determinato evento su un elemento, essa diviene un metodo di quell'elemento. Per questo motivo il parametro implicito *this* fa riferimento all'elemento su cui l'evento si è verificato. Sempre per il solito motivo, è possibile invocare un gestore di evento esplicitamente.

Un gestore di evento può impedire che il browser svolga le azioni di default associate a quell'evento, semplicemente restituendo il valore false.

8.1.2 Modello a eventi avanzato

Il modello avanzato fa parte del DOM Level 2.

Esso è supportato da tutti i browser tranne Internet Explorer (almeno dalla versione 4 alla versione 6 ma da alcune prove che abbiamo fatto pare che non sia supportato neanche nelle versioni successive, la 7 e la 8).

Esso è abbastanza diverso dal modello originario e certamente più potente e complesso.

In questo modello, quando si verifica un evento su un elemento del documento, il gestore (o i gestori) dell'evento registrati sull'oggetto target sono invocati ma, in aggiunta, ciascuno degli elementi antenati del target hanno la possibilità di gestire l'evento[33]. Si parla cioè di propagazione degli eventi.

Nel Level 2 Model, un gestore di eventi può essere registrato per un particolare elemento invocando il metodo addEventListener() di quell'oggetto.
Questo metodo riceve tre argomenti:

- Il nome del tipo di evento per cui si vuole registrare il gestore: tale nome è una stringa che contiene in lettere minuscole il nome dell'attributo HTML senza il prefisso "on"
- La funzione che deve essere invocata quando l'evento si verifica.
  È necessario passare come argomento della funzione un oggetto Event[34]
- Un valore booleano: se true l'evento viene catturato e gestito dall'event handler specificato in fase di capturing, altrimenti viene gestito in fase di bubbling

È possibile rimuovere un gestore da un oggetto usando il metodo removeEventListener() che aspetta gli stessi argomenti della addEventListener()[35].

[33] È possibile scegliere se tale gestione deve avvenire prima o dopo che l'evento è stato gestito dall'elemento target.
[34] Vedremo in seguito a cosa serve

Le tre caratteristiche che rendono questo modello "avanzato" rispetto al primo sono quindi:

- La propagazione degli eventi
- La possibilità di avere più gestori per lo stesso evento sullo stesso elemento
- L'esistenza dell'oggetto e

8.1.2.1 Propagazione degli eventi

Propagazione degli eventi significa che un evento non avviene semplicemente su un oggetto e subito muore; piuttosto esso invece si propaga e può essere gestito su altri oggetti presenti nella gerarchia del documento.

La propagazione degli eventi consiste in tre fasi:

1) **Fase di cattura**: durante questa fase gli eventi si propagano dall'oggetto Document attraverso l'albero del documento fino al nodo target. Se qualche elemento antenato del target ha un gestore per la cattura dell'evento, questo viene eseguito
2) **Fase del target node:** questa fase avviene direttamente sul nodo target; se disponibili, vengono eseguiti tutti gli appropriati gestori di eventi
3) **Fase di bubbling:** durante questa fase, l'evento risale all'indietro dall'elemento target fino all'oggetto Document.

È importante notare che, sebbene tutti gli eventi sono soggetti alla fase di cattura, non tutti possono risalire la gerarchia[36].

Un gestore di evento può decidere di bloccare l'ulteriore propagazione dell'evento richiamando il metodo stopPropagation() dell'oggetto che rappresenta l'evento.

Nel caso, in risposta ad un evento, esistano azioni di default, queste sono eseguite dopo che tutte le fasi di propagazione degli eventi sono state completate e ciascuno dei gestori è stato eseguito. Ciascuno dei gestori invocati durante la propagazione possono prevenire l'azione di default del browser invocando il metodo preventDefault() dell'oggetto evento.

8.1.2.2 Gestori multipli

Dato che, in questo modello esteso, i gestori di eventi sono registrati invocando un metodo piuttosto che settando un attributo o proprietà, si possono registrare più gestori per lo stesso tipo di evento su un certo oggetto.
Quando si hanno più gestori, tutti verranno invocati. Non è possibile però conoscere l'ordine in cui le varie funzioni sono invocate, che non è detto che sia lo stesso in cui sono state registrate.
Registrare più funzioni di gestione per un certo evento su un certo oggetto ha il vantaggio di permettere la modularizzazione del codice, il quale può essere separato in più funzioni anziché essere raggruppato in un'unica funzione.

---

[35] Si noti che i metodi addEventListener() e removeEventListener() sono definiti nell'interfaccia EventTarget. Teoricamente, nei browser web che supportano il DOM level 2 event module tutti e soli i nodi presenti in un documento (compresi quelli di testo) implementano l'interfaccia EventTarget; in pratica i browser supportano la registrazione di gestori di eventi solo sui nodi Element e Document, e sull'oggetto Window, anche se è al di fuori dei confini del DOM.

[36] in generale, tornano su i "raw input events" ma non i "higher level semantic event"

Se si registra più volte la stessa funzione sullo stesso elemento, tutte le registrazioni dopo la prima vengono ignorate.

8.1.2.3 L'oggetto e

Una delle mancanze del modello originario è quello di non permettere di ottenere dettagli sull'evento che si è verificato. Per esempio quando si verifica l'evento "MouseDown" potremmo essere interessati a conoscere varie informazioni come: il tasto premuto, la posizione del puntatore, e così via.

La soluzione è l'invenzione dell'oggetto che rappresenta un evento e le cui proprietà forniscono specifici dettagli.

Quando un evento si verifica, al suo gestore viene passato come parametro un oggetto che rappresenta l'evento. Il gestore potrà accedere alle proprietà di quest'oggetto per conoscerne i dettagli.

Tale oggetto ha la caratteristica di implementare l'interfaccia Event o delle sue sottointerfacce a seconda del tipo di evento. L'interfaccia principale è Event che supporta i principali tipi di eventi (abort, blur, change, error, focus, load, reset, resize, scroll, select, submit, upload). Per tipi specifici di eventi il supporto è delegato a delle sottointerfacce: MouseEvent (click, mouse down, mousemove, mouse out, mouseover, mouseup), UIEvent (DOMActivate, DOMFocusIn, DOMFocusOut), MutationEvents (quasi mai utilizzata).

Queste tre interfacce formano una gerarchia: l'interfaccia Event è la radice. UIEvent è una sottointerfaccia di Event, MouseEvent è una sottointerfaccia di UIEvent.
Questo significa che tutti gli oggetti che rappresentano eventi implementano l'interfaccia Event, per cui hanno a disposizione certe proprietà e metodi in comune. A questi si aggiungono le proprietà e i metodi delle specifiche sottointerfacce.

L'interfaccia Event

L'interfaccia Event definisce le seguenti proprietà:

- type: il tipo di evento che si è verificato. Il valore di questa proprietà è il nome dell'event handler senza "on".
- target: l'elemento del documento su cui l'evento è avvenuto.
- currentTarget: il nodo su cui l'evento viene gestito in questo momento. Il valore di questa proprietà è diversa dal valore della proprietà target, quando l'evento è gestito in fase di capturing o di bubbling
- eventPhase: è un numero che specifica in quale fase del processo di propagazione ci troviamo.
- timeStamp: un oggetto di tipo Date che specifica quando un evento si è verificato
- bubbles: un valore booleano che specifica se questo tipo di evento risale (bubbles up) l'albero del documento
- cancelable: un valore booleano che specifica se l'evento ha una default action associata.

In aggiunta a queste proprietà, l'interfaccia definisce i seguenti metodi:

- stopPropagation(): serve per impedire l'ulteriore propagazione dell'evento
- preventDefault(): serve per impedire che il browser svolga l'azione di default associata con l'evento

L'interfaccia UIEvent

L'interfaccia UIEvent definisce altre due proprietà:

- view: l'oggetto Window all'interno del quale l'evento si è verificato
- detail: un numero che fornisce informazioni aggiuntive sull'evento

L'interfaccia MouseEvent

L'interfaccia MouseEvent definisce le seguenti ulteriori proprietà:

- button: è un numero che specifica quale bottone ha cambiato stato durante un evento mouse down, mouseup, click.
- altkey, ctrlKey, meta Key, shiftKey: indicano se un un tasto (Alt, Ctrl, Meta, Shift) era premuto quando si è verificato l'evento del mouse
- clientX, clientY: specificano le coordinate del puntatore del mouse relative alla finestra del browser
- screenX, screenY: le coordinate del puntatore del mouse relative al lato in alto a sinistra del monitor
- relatedTarget: contiene il riferimento ad un nodo che è correlato al nodo target dell'evento.

8.1.3 Modello a eventi di Internet Explorer

Il modello ad eventi di Internet Explorer è un modello intermedio, una via di mezzo tra l'original Level 0 event model e lo standard level 2 model.

Esso presenta molte delle caratteristiche dello standard level 2 model, ma con alcune differenze:

- utilizzo di un oggetto per la rappresentazione dell'evento. Tale oggetto, anziché essere disponibile come parametro delle funzioni di gestione, è accessibile come proprietà dell'oggetto Window. Inoltre le proprietà di questo oggetto sono leggermente diverse
- Supporto della propagazione degli eventi, ma solo tramite bubbling. Il capturing non è supportato anche se IE5 e successivi forniscono funzionalità alternative per catturare gli eventi del mouse
- Possibilità di registrare gestori multipli (ma solo dalla versione 5 di IE), con funzioni di registrazione non standard.

8.1.3.1 Registrazione gestore di eventi

In IE 4 i gestori di eventi sono specificati come attributi HTML o come proprietà degli oggetti del documento, allo stesso modo dell'original Level 0 event model.

In IE5 e successivi sono stati introdotti i metodi attachEvent() e detachEvent().
Questi eventi funzionano come addEventListener() e removeEventListener() con le seguenti eccezioni:

- Dato che l'event capturing non è supportato, queste funzioni si aspettano solo due parametri (tipo di evento e funzione di gestione);
- Il tipo di evento viene passato con il prefisso "on"
- Le funzioni registrate con attachEvent() sono invocate come funzioni globali, piuttosto che come metodi dell'elemento su cui l'evento si è verificato. La parola chiave this farà quindi riferimento all'oggetto Window piuttosto che all'oggetto target dell'evento.
- attachEvent() permette di registrare più volte lo stesso gestore. Quando un certo tipo di evento si verifica, la funzione sarò invocata tante volte quante è stata registrata

8.1.3.2 Propagazione degli eventi

Nel modello ad eventi di IE non esiste il concetto di event capturing. Gli eventi possono solo risalire la gerarchia. Come nel modello avanzato, tuttavia, non tutti gli eventi possiedono questa possibilità[37].

A partire da IE 5, sono disponibili dei metodi che permettono di effettuare il capturing ma solo per gli eventi del mouse.
Si tratta dei metodi setCapture() e releaseCapture(). Invocando setCapture() su un elemento, tutti i successivi mouse events sono diretti a quell'elemento, finché non viene invocato il metodo releaseCapture o finché la cattura non viene interrotta[38]. Se questo accade l'elemento su cui setCapture è stato chiamato riceverà l'evento onlosecapture.

8.1.3.3 L'oggetto IE Event

Le proprietà più importanti dell'oggetto IE Event sono:

- Type: una stringa che specifica il tipo di evento che è avvenuto. Il valore di questa proprietà è il nome dell'event handler senza "on".
  È compatibile con la type property del DOM Event object.
- srcElement: l'elemento del document su cui l'evento è avvenuto.
  Compatibile con la proprietà target del DOM Event object.
- Button: un intero che specifica il bottone del mouse che è stato premuto. Il valore 1 indica il bottone sinistro, 2 indica il bottone destro, 4 indica il bottone centrale. Se più bottoni sono premuti contemporaneamente, questi valori sono sommati tra loro
  Anche se il nome della proprietà è uguale nel DOM Event model, cambia l'interpretazione che viene data ai valori
- clientX, clientY: proprietà di tipo intero che specificano le coordinate del mouse nel momento di un evento. Le coordiante sono relative all'angolo in alto a sinistra della finestra.
  Compatibili con DOM Level 2 MouseEvent properties aventi lo stesso nome.
- offsetX, offsetY: queste proprietà di tipo intero specificano la posizione del puntatore del mouse relativamente all'elemento source (questo permette per esempio di determinare quale pixel dell'immmagine è stato premuto)
  Non esistono proprietà equivalenti nel DOM event model
- altkey, ctrlKey, shiftKey: queste proprietà booleane specificano se Alt, Ctrl e Shift erano premuti quando un evento è avvenuto.
  Queste proprietà sono compatibili con le proprietà del DOM Level 2 che hanno lo stesso nome.
  (notare, tuttavia, che l'oggetto IE Event non ha una proprietà metaKey)
- keyCode: proprietà di tipo integer che specifica il codice del carattere per gli eventi keydown e keyup, e codice Unicode per gli eventi keypress.
- fromElement: specifica l'elemento del documento da cui il mouse proviene per eventi mouseover
- toElement: specifica l'elemento del documento verso cui il mouse si è mosso per mouse out events. Simile alla proprietà relatedTarget del DOM level 2.
- cancelBubble: proprietà booleana che, quando settata a True, impedisce all'evento corrente di risalire ulteriormente la gerarchia.

[37] in generale, tornano su i "raw input events" ma non i "higher level semantic event"

[38] La cattura può essere interrotta se il browser web perde il focus, compare una finestra di dialogo (alert()), è visualizzato un menu di sistema.

Simile a stopPropagation del DOM Level 2.

- returnValue: una proprietà booleana che può essere impostata a false per impedire al browser di svolgere l'azione di default associata con l'evento. Si tratta di un'alternativa alla tecnica tradizionale di ritornare false da un event handler.
  Simile al metodo preventDefault del DOM level 2.

Nonostante alcune somiglianze tra queste proprietà e quelle del modello ad eventi avanzato del DOM Level 2[39], ci sono alcune differenze.

Diverso è anche il modo in cui l'oggetto Event è reso disponibile. Esso non è mai passato come argomento ai gestori di evento, ma è accessibile come proprietà dell'oggetto globale Window[40]: un gestore può riferirsi ad un evento tramite window.event o semplicemente event.

8.1.4 Modello a eventi di Netscape 4

Il modello a eventi di Netscape 4, come quello di IE, contiene dei concetti (come quello di oggetto evento, o di propagazione degli eventi) che ritroviamo anche nel modello tradizionale ma con alcune differenze:

- l'oggetto che rappresenta l'evento presenta proprietà diverse
- solamente il capturing e non il bubbling è supportato nella propagazione degli eventi
- solo gli oggetti Window, Document e Layer hanno la possibilità di gestire certi tipi di eventi prima che essi vengano processati dagli elementi che li hanno generati.

8.1.4.1 L'oggetto Event

L'oggetto Event possiede le seguenti proprietà:

- type: una stringa che contiene il nome del tipo di evento
- target: elemento sorgente dell'evento
- which: per conoscere quale tasto del mouse è stato premuto
- which: contiene il codice unicode del carattere che è stato premuto
- modifiers: fornisce una bitmask contenente i flag: Event.ALT_MASK, Event.CONTROL_MASK, Event.META_MASK, Event.SHIFT_MASK
- pageX, pageY: specificano le coordinate relative alla pagina web
- screenX, screenY: specificano le proprietà relative allo schermo.

8.1.4.2 Registrazione dei gestori

La registrazione dei gestori di eventi avviene nelle stesse modalità del modello originario.

8.1.4.3 Propagazione

In Netscape 4 la propagazione degli eventi avviene solamente per mezzo del capturing.

[39] Le somiglianze tra le proprietà dell'IE Event object e le proprietà degli oggetti Event, UIEvent, MouseEvent del modello avanzato sono dovute al fatto che gli oggetti Event definiti nel modello standard sono stati modellati sull'IE Event object.

[40] Sebbene potrebbe apparire strano il fatto di usare una variabile globale al posto di un argomento di funzione, il modello di IE funziona perché assume che solo un evento alla volta venga processato: due eventi non vengono mai gestiti contemporaneamente.

Gli oggetti Window, Document e Layer possono richiedere la possibilità di vedere in anteprima certi tipi di eventi che si verificano. Tale richiesta può essere fatta tramite il metodo captureEvent(). Esso richiede come argomento una bitmask composta di costanti definite come proprietà statiche dell'Event constructor.

Per esempio, scrivendo window.captureEvents(Event.MOUSEDOWN|Event.MOUSEUP), l'oggetto Window sarà in grado di vedere gli eventi mouse down e mouse up.
A questo punto, registriamo i gestori per questi eventi:
window.onmousedown=function(event){…};
window.onmouseup=function(event){…};
Quando uno di questi capturing event handlers riceve un evento, può decidere cosa deve succedere successivamente:

- l'evento deve essere propagato al prossimo oggetto che ha usato captureEvent per specificare l'interesse per quell'evento (metodo routeEvent() o handleEvent() se invece si desidera specificare noi l'oggetto a cui passare l'evento.
- L'evento deve andare direttamente al source element.

8.2 La soluzione adottata dal tool

Come è già stato detto, il rilevamento degli eventi è stato realizzato per mezzo di un javascript inserito all'interno di ciascuna pagina web da testare.
Il tool doveva poter essere utilizzabile sia tramite IE4 che Netscape 4.
Tuttavia, abbiamo visto, i modelli ad eventi dei due browser non sono tra loro compatibili.
Di seguito espliciteremo quali sono le principali incompatibilità tra i due modelli e come sono state risolte da chii ha progettato la versione originaria del tool

8.2.1 Risoluzione dei problemi di compatibilità

Se confrontiamo tra loro i due modelli notiamo che essi sono abbastanza diversi.
Elenchiamo le differenze più significative:

- Diversa modalità di propagazione degli eventi
- Diversità dell'oggetto che rappresenta l'evento
- Diversità riguardanti i gestori di eventi (modalità di registrazione, caratteristiche, ecc)

Proviamo ad analizzare meglio questi problemi, spiegando il modo in cui sono stati risolti.

Propagazione degli eventi

Il modello di Netscape è detto"event capturing": gli eventi percorrono la gerarchia di oggetti del documento dall'alto verso il basso, dall'oggetto Window all'oggetto sorgente dell'evento.
Il modello di Microsoft è detto "event bubbling": gli eventi risalgono la gerarchia dal basso verso l'alto, dal source element fino all'oggetto Window[41].

Tuttavia possiamo notare che entrambi i modelli hanno in comune la possibilità per certi eventi di essere propagati fino agli oggetti Window e Document (a seconda del tipo di evento), centralizzandone il processo di gestione.
Questa proprietà è stata sfruttata dallo script, nel quale sono stati registrati dei gestori per i vari oggetti su questi oggetti.

[41] È possibile per un gestore specificare che l'evento non deve risalire ulteriormente.

Per gli eventi che non si propagano fino agli oggetti Window e Document, è stato ridefinito il gestore di evento relativo all'oggetto sorgente.

Oggetti su cui possono essere registrati i gestori
In Explorer 4 possiamo registrare un gestore di evento su ogni elemento del documento. In Netscape 4 solo gli oggetti Window, Document e Layer possono richiedere la possibilità di gestire gli eventi prima che vengano gestiti dal source element.
Dato che abbiamo deciso di gestire i vari eventi al livello dell'oggetto Window e Document, non sussiste nessun problema.

Diversità dell'oggetto che rappresenta l'evento
Sebbene entrambi i modelli di eventi definiscano un oggetto Event, il modo in cui l'oggetto è reso disponibile differisce drasticamente nei due browser: in Netscape 4 l'oggetto è passato come argomento al gestore di evento, in IE esso viene memorizzato nella variabile globale denominata event.

Questo problema può essere risolto scrivendo il gestore in questo modo:

```
function handler(e){
if(navigator.appName.indexOf("Microsoft")!=-1)
                                        e=window.event;
}
```

In questo modo, nel caso in cui si usi IE settiamo noi il parametro e al valore della variabile globale window.event.

Una seconda diversità relativa all'oggetto Event, riguarda il fatto che le proprietà dell'oggetto event sono quasi completamente differenti nei due browser (vedi tabella)

Tabella 8 Le proprietà dell'oggetto e in Netscape 4 e in Explorer 4 a confronto.

Descrizione	Netscape 4.0	Explorer 4.0
Il nome del tipo di evento (ad es. click, load, submit)	`type`	`type`
L'oggetto che riceve o che genera l'evento	`target`	`srcElement`
Indica quale bottone del mouse è stato premuto oppure il codice ASCII associati ai tasti della tastiera	`which`	`Button` `keyCode`

Specifica se al verificarsi dell'evento i tasti funzione ALT, CONTROL, SHIFT erano premuti	`modifiers`	`AltKey`
		`CtrlKey`
		`ShiftKey`

Coordinata orizzontale dell'evento relativa alla finestra del browser	`pageX`	`clientX`
Coordinata verticale dell'evento relativa alla	`pageY`	`clientY`

finestra del browser		
Coordinata orizzontale dell'evento relativa all'intero schermo	`screenX`	`screenX`
Coordinata verticale dell'evento relativa all'intero schermo	`screenY`	`screenY`

Tuttavia questo non è un grande problema dato che consente, se pur utilizzando nomi diversi, di recuperare le stesse identiche informazioni.

8.2.2 Lo script di rilevamento originario: analisi

```
var isNav = (navigator.appName.indexOf("Netscape") != -1);
var isIE = (navigator.appName.indexOf("Microsoft") != -1);
if (isNav)
{
        document.captureEvents(Event.CLICK | Event.MOUSEOVER|Event.KEYPRESS);
        window.captureEvents(Event.LOAD|Event.UNLOAD);
}
var applet=null;
for(var i = 0; i < parent.frames.length; i++)
  if (top.frames[i].name == "hidden")
  {
        var doc = top.frames[i].document
        applet = doc.applets[0]
  }

/FUNZIONI
function time()
{       …//restituisce la data
}

//gestori di eventi
function handlerImage(e)
{
 if (applet != null)
  {
    var src;
    if (isNav) src = e.target;
    if (isIE)
    {
     e = window.event;   // Cattura evento.
     src = e.srcElement;
    }
    var linea = e.type;
    if (e.type == "abort")
```

```
                applet.anEvent(time(),e.type,"",src.name);
    if (e.type == "click")
                applet.imageEvent(time(),e.type,src.name,src.src);
    if (e.type == "dblclick")
                applet.imageEvent(time(),e.type,src.name,src.src)

        //linea = e.type+"|image| "+src.name+"| "+src.src+"| "+ src.href;
   if (isIE) e.cancelBubble = true;
  }
}
function handler(e)
{
 var url = unescape(document.location);
 var src;
        if (isNav) src = e.target;
        if (isIE)
        { e = window.event;
          src = e.srcElement;
        }

 if (e.type == "change")
        applet.changeEvent(time(),src.type,src.name,src.value,src.selectedIndex);
 if (e.type == "click")   applet.clickEvent(time(),src.type,src.name,src.value,src.href);
 if (e.type == "dblclick")   applet.anEvent(time(),e.type,src.type,"");
 if (e.type == "error")   applet.anEvent(time(),e.type,src.type,"");
 if (e.type == "keypress")   applet.keyEvent(time(), src.type, e.keyCode, e.which);
 if (e.type == "mouseover")   applet.anEvent(time(),e.type,src.type,"");
 if (e.type == "reset")   applet.anEvent(time(),e.type,"", src.name);
 if (e.type == "resize")   applet.anEvent(time(),e.type,"","");
 if (e.type == "select")   applet.anEvent(time(),e.type,src.type,"");
 if (e.type == "submit")  applet.anEvent(time(),e.type,"","");
 if (e.type == "unload")   applet.anEvent(time(),e.type,"","");
 if (e.type == "scroll")   applet.anEvent(time(),e.type,"","");

//registrazione not bubbling events sui singoli elementi sorgenti
 if (e.type == "load" && applet != null)
 {
        for(var i=0;i<document.forms.length;i++)
        {       document.forms[i].onsubmit = handler;
                document.forms[i].onreset = handler;
                for(var j=0;j<document.forms[i].elements.length;j++)
                {
                document.forms[i].elements[j].onchange = handler;
                }
        }

        for(var i =0; i<document.images.length;i++)
```

```
        {
                document.images[i].onabort=handler;
                document.images[i].onclick = handlerImage;
                document.images[i].ondblclick = handlerImage;
        }

        applet.loadEvent(time(),url);

  }

   if (isIE)  e.cancelBubble = true;

 return true;
}

document.onclick = handler;
document.ondblclick = handler;
document.onselect = handler;
document.onerror = handler;
window.onload = handler;
window.onunload = handler;
window.onresize= handler;
if (isIE)  window.onscroll=handler;
}
```

Nella prima parte dello script avviene l'inizializzazione di alcune variabili:

- isNav: se vale *true* significa che l'utente sta usando Netscape come user agent
- isIE: se vale *true* significa che l'utente sta usando Internet Explorer come user agent
- applet: viene inizializzata con il riferimento all'applet che si trova all'interno del frame hidden (il ciclo poteva essere evitato sfruttando direttamente il nome del frame: top.hidden.applet[0];

Sempre nella prima parte si richiama il metodo captureEvents in modo che gli oggetti window e document possano gestire preliminarmente alcuni eventi.
Nella parte finale avviene la registrazione dei gestori dei metodi per gli eventi che possono essere gestiti direttamente sull'oggetto window o document, o perché sono questi ultimi gli elementi sorgenti o perché questi elementi vengono raggiunti durante la fase di bubbling.
Tutti gli eventi vengono gestiti per mezzo del medesimo gestore *handler* definito nella parte centrale del codice.
Notare che sia nel caso di Explorer che di Netscape i gestori vengono registrati settando le proprietà degli oggetti javascript, data che è l'unica modalità supportata.

Nella parte centrale abbiamo le funzioni che gestiscono gli eventi.
Il gestore *handler* per prima cosa, nel caso si utilizzi Netscape setta il parametro *e* con il riferimento all'oggetto globale window.event. La variabile src conterrà il riferimento all'oggetto sorgente dell'evento (accedibile tramite la proprietà target in Netscape e srcElement in Explorer).
A seconda del tipo di evento vogliamo inviare certi tipi di informazioni all'applet e non certe altre, per questo vengono richiamati metodo diversi residenti nell'applet a seconda del tipo di evento.

Nel caso in cui il tipo di evento è load, cioè la pagina è stata caricata, andiamo a registrare alcuni gestori su certi elementi come le form e le immagini per certi eventi non rilevabili all'altezza dell'oggetto window o document perché non hanno la capacità di risalire la gerarchia. (Non si capisce perché nel caso delle immagini sia stato definito il gestore per l'evento onabort e non per quello onerror). Nel caso delle immagini è stato definito un gestore diverso per gli eventi click e dblclick forse perché si voleva differenziare questo evento dal click su altri elementi.

8.3 La nostra versione

Lo script è stato da noi modificato in modo da:

- Inviare le informazioni ad un altro script presente nel frame hidden anziché all'applet (che come vedremo verrà rimossa)
- funzionare con tutti i maggiori browser moderni desktop e mobili. Tutti i browser moderni utilizzano il modello avanzato tranne Explorer che non si è adeguato agli standard mantenendo il suo modello
- Rendere il codice più pulito eliminando le parti inutili

In seguito a queste operazioni abbiamo ottenuto il seguente script:

```
//Riconosce se il browser è explorer
var isIE = (navigator.appName.indexOf("Microsoft") != -1);
var script=null;
script= top.hidden;

 //funzioni di gestione
function time(){          var data= new Date();
          //Rappresentazione in millisecondi dell'oggetto data.
          //e'  il numero di millisecondi tra la mezzanotte(GMT) del 1 gen 1970 e
          //la data specificata.
          var millesimi = data.getTime();
          return "time:" + millesimi //+"$"+ data.toUTCString(); //+"$"+ data.toLocaleString();
}

function handlerImage(e)
{
 if (script != null)
   {
     var src;
     if (isIE)
     {
       e = window.event;   // Cattura evento.
          src = e.srcElement;
     }
     else
          src=e.target;

     if (e.type == "abort")
```

```
                script.anEvent(time(),e.type,"",src.name);
    if (e.type == "click")
                script.imageEvent(time(),e.type,src.name,src.src);
    if (e.type == "dblclick")
                script.imageEvent(time(),e.type,src.name,src.src)
    if (isIE) e.cancelBubble = true;
   }
}

function handler(e)
{
 var url = unescape(document.location);
 var src;
 if (isIE)
 {
     e = window.event;   // Cattura evento.
        src = e.srcElement;
 }
 else
  src=e.target;

 if (e.type == "change")
                script.changeEvent(time(),src.type,src.name,src.value,src.selectedIndex);
 if (e.type == "click")
                script.clickEvent(time(),src.type,src.name,src.value,src.href);
 if (e.type == "error")
                script.anEvent(time(),e.type,src.type,"");
 if (e.type == "reset")
                script.anEvent(time(),e.type,"", src.name);
 if (e.type == "resize")
                script.anEvent(time(),e.type,"","");
 if (e.type == "select")
                script.anEvent(time(),e.type,src.type,"");
 if (e.type == "submit")
                script.anEvent(time(),e.type,"","");
 if (e.type == "unload")
                script.anEvent(time(),e.type,"","");
 //solo IE
        if (e.type == "scroll")
                script.anEvent(time(),e.type,"","");

 if (e.type == "load" && script != null)
 {
        for(var i=0;i<document.forms.length;i++)
        {       document.forms[i].onsubmit = handler;
                document.forms[i].onreset = handler;
                for(var j=0;j<document.forms[i].elements.length;j++)
```

```
                {
                document.forms[i].elements[j].onchange = handler;
                document.forms[i].elements[j].onselect = handler;
                }
        }
        for(var i =0; i<document.images.length;i++)
        {
                document.images[i].onabort=handler;
                document.images[i].onerror=handler;
        }
        script.loadEvent(time(),url);
  }
  if (isIE)  e.cancelBubble = true;
 return true;
}

//Handler degli eventi
if (script != null)
{
        document.onclick = handler;
        window.onload = handler;
        window.onunload = handler;
        window.onresize= handler;
        window.onscroll=handler;
}
```

Non abbiamo apportato cambiamenti per quanto riguarda la scelta, a nostro parere poco chiara, di richiamare funzioni diverse (dello script) a seconda del tipo di evento. Riteniamo comunque che sia utile in futuro rivalutare questa scelta ed eventualemente sostituirla con una diversa e più chiara.

A seconda dell'applicazione che si vorrà fare di questo tool sarà importante rivalutare quali eventi è necessario catturare e quali tipi di informazioni memorizzare a seconda del tipo di evento.

Un'altra cosa da notare è che, nonostante il modello di eventi avanzato preveda l'utilizzo del metodo addEventHandler() per la registrazione dei gestori di eventi, noi abbiamo utilizzato la modalità semplice di registrazione degli eventi attraverso il settaggio di una proprietà javascript. Questo perché tale modello di registrazione è compatibile sia con Internet Explorer sia con gli altri browser, e Javascript ci consente di mescolare il modello di eventi avanzato con quello originario, ancora supportato dagli attuali browser per compatibilità all'indietro.

9. PROBLEMA 2: ALTERNATIVA ALL' APPLET

Un componente fondamentale dell'architettura del tool è l'applet. Lo scopo dell'applet è quella di ricevere dal javascript le informazioni relative alle azioni compiute dall'utente, concatenarle con quelle precedentemente ricevute, quindi inviarle al server alla fine della sessione.
La sessione termina quando l'utente ha completato il test oppure quando l'utente clicca sul link stop. L'applet comunica con la servlet effettuando una richiesta (POST request) attraverso una connessione http all'URL relativo alla servlet.
Per fare in modo che una volta caricata rimanga attiva per tutta la durata del test, l'applet è stata inserita in un frame indipendente da quella contenente il sito da testare.

9.1 Che cosa si intende per Applet e Java Applet

Il termine applet deriva dalla combinazione di due termini: *application* e *gadget.*
Si tratta di un programma che viene eseguito all'interno di un altro programma, detto *contenitore* che si trova su un client.
Nonostante il termine sia stato introdotto nel 1993 con gli Applescript, oggi, quando parliamo di applet ci riferiamo comunemente alle applet Java. Si tratta di programmi scritti in linguaggio Java e che per essere eseguiti necessitano di essere inseriti all'interno di un browser web su cui è installata la Java Virtual Machine[42]. Esse vengono solitamente utilizzate per creare funzioni interattive che non possono essere realizzate con altre tecnologie web.
A differenza dei programmi scritti con i linguaggi di scripting, generalmente non considerati applet, le applet presentano in genere un qualche tipo di interfaccia utente.

9.2 Incompatibilità dell'applet con i dispositivi mobili

Come abbiamo detto, un componente essenziale nell'architettura del logging tool, è l'applet java.
Il problema è che le applet non funzionano sui browser mobili. Infatti, per poter essere eseguita un'applet richiede che sul client sia installata la Java Virtual Machine nella versione J2SE (Standard edition).
Purtroppo, la maggior parte dei dispositivi mobili non supportano la versione J2SE ma la J2ME (Mobile Edition), una versione specificamente progettata per dispositivi mobili con risorse limitate.
Ora, la piattaforma J2ME è in grado di leggere solo le applicazioni Java pensate ad hoc per dispositivi mobili (le cosiddette midlet, formate da un file jar e uno jad). Non è invece in grado di leggere le applicazioni java per desktop e le applet per cui è necessaria l'edizione standard.

Nel paragrafo successivo riflettiamo su quali potrebbero essere le soluzioni al problema.

9.3 Risolvere le incompatibilità

Per risolvere le incompatibilità del tool dovute all'utilizzo dell'applet, siamo partiti dall'analisi di quali siano le operazioni da essa svolte. Abbiamo poi riflettuto su quali potessero essere delle soluzioni alternative che, pur facendo uso di altre tecnologie supportate dai dispositivi mobili, consentissero di ottenere i medesimi risultati.

Dunque, le operazioni svolte dall'applet sono:

[42] In alternativa le applet possono essere eseguite all'interno di un programma prodotto da Sun allo scopo di testare le applet: Sun Apple Viewer.

- ricevere dal javascript le informazioni relative alle azioni compiute dall'utente e concatenarle con quelle precedentemente ricevute[43]
- inviarle al server alla fine della sessione.

La prima operazione può essere svolta da un normale javascript, anch'esso contenuto in un frame diverso per garantire la persistenza dei dati.
L'altra operazione – l'invio dei dati al server alla fine della sessione- può essere eseguita da un AJAXscript. Almeno in teoria, i moderni browser mobili sono in grado infatti di supportare AJAX.

9.3.1 La nostra soluzione per ricevere le informazioni e concatenarle con le precedenti

Dall'analisi del sorgente dell'applet abbiamo visto che la parte relativa alla ricezione delle informazioni e alla loro concatenazione viene ottenuta attraverso una serie di funzioni:

- Funzioni (diverse a seconda del tipo di evento), che ricevono le informazioni, le salvano in una variabile e richiamano una funzione per il concatenamento con quelle precedenti
- Funzione di concatenamento, che ricevute le informazioni le salva nella variabile globale log

Si tratta di semplici operazioni che potrebbero essere eseguite anche attraverso un semplice javascript. Si è quindi passati ad una "traduzione" del codice dal linguaggio Java al linguaggio Javascript.

9.3.2 La nostra soluzione per l'invio delle informazioni al server

Per l'invio delle informazioni al server abbiamo pensato di utilizzare uno script AJAX. Poiché non si aveva un'esperienza pregressa di uso di questa tecnologia è stato necessario uno studio preliminare per poter effettuare le modifiche necessarie, che riportiamo sotto.

9.3.2.1 Cenni su AJAX

Che cos'è AJAX

AJAX è l'acronimo di Asynchronous Javascript and XML.

Non si tratta di una nuova tecnologia, ma di un insieme di tecnologie già esistenti che vengono utilizzate insieme:

- XHTML per il markup
- CSS per lo stile
- DOM (Document Object Model) manipolato attraverso un linguaggio ECMAScript come JavaScript o JScript, per modificare dinamicamente il documento
- XML come formato di scambio dei dati, anche se di fatto qualunque formato può essere utilizzato.

L'unica invenzione è rappresentata dall'oggetto XMLHttpRequest per l'interscambio asincrono dei dati tra il browser dell'utente e il web server.

La tecnologia AJAX si basa appunto su uno scambio di dati in background fra web browser e server tramite javascript, il che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente.

[43] Le informazioni vengono concatenate all'interno di una variabile. Poiché vogliamo che tale variabile contenga alla fine tutte le informazioni relative all'intera sessione del test, è necessario che essa rimanga attiva per tutta la durata del test. Il ricaricamento determinerebbe la perdita del precedente contenuto della variabile. È per questo che l'applet è stata inserita all'interno di un frame indipendente, che viene caricato una sola volta e rimane fisso per tutta la durata della sessione.

Questo modo di procedere, differisce molto da quello che succede nelle applicazioni tradizionali.

Nelle applicazioni web tradizionali, l'utente esegue delle azioni sull'interfaccia del sito (pressione di un link, sottomissione di una form ecc). Alcune di queste azioni determinano delle richieste HTTP al server. Il server esegue alcuni processi e poi restituisce una nuova pagina HTML al cliente.

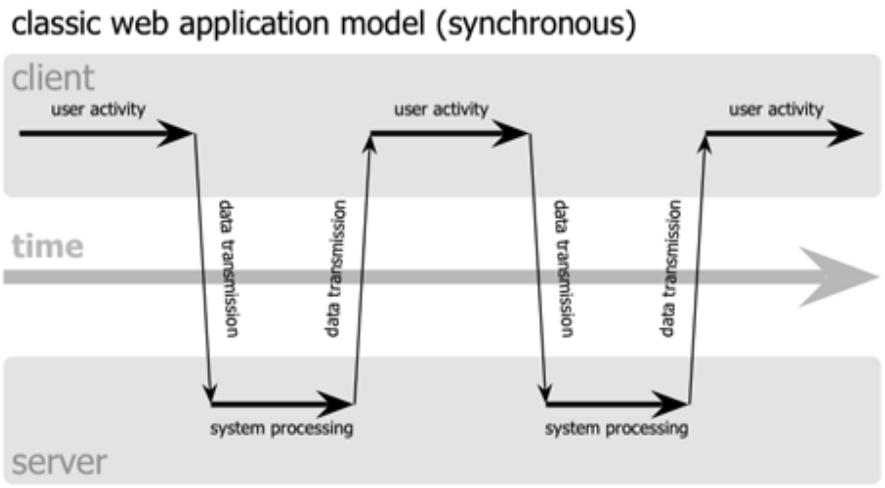

Figura 40 scambio di dati tra client e server nelle applicazioni web classiche (Jesse James Garrett)

In questo modello classico:

- Si può avere un grande spreco di tempo e di banda dato che molto spesso la nuova pagina è molto simile a quella precedente,
- l'utente si trova a dover attendere che il server abbia completato le sue operazioni e che la pagina venga inviata al browser, prima di poter continuare ad interagire con la pagina

Le applicazioni Ajax, invece, possono inviare richieste al web server per ottenere solo i dati che sono necessari. Come risultato si ottengono applicazioni più veloci (dato che la quantità di dati interscambiati fra il browser ed il server si riduce) e che utilizzano meno la banda a disposizione. Inoltre, l'utente non si troverà più davanti ad una barra di caricamento o una clessidra nell'attesa che la nuova pagina venga ricaricata e possa continuare l'interazione[44].

[44] Una barra di caricamento potrà talvolta apparire ma essa riguarderà solamente la piccola porzione della pagina richiesta, non tutta. Nell'attesa l'utente potrà continuare a interagire con altre parti della pagina.

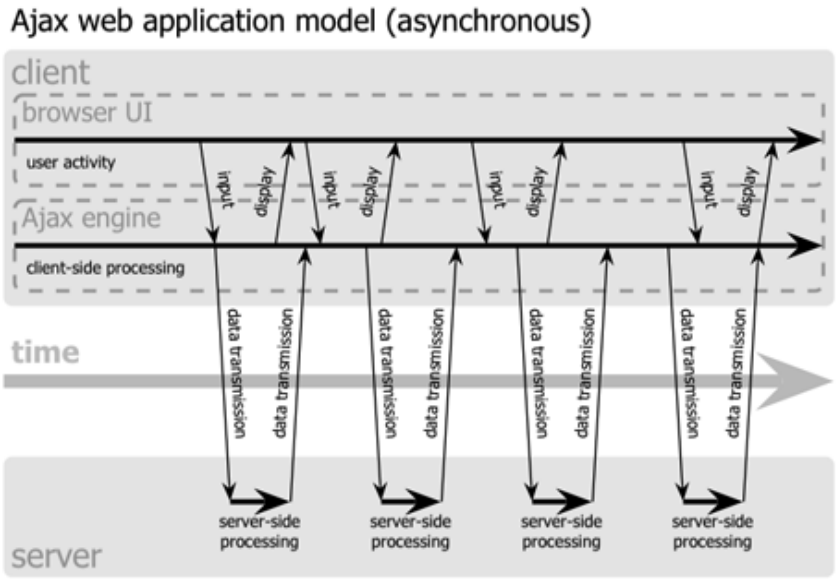

Figura 41 Scambio di dati tra client e server nelle applicazioni Web che utilizzano Ajax (Jesse James Garrett)

Da un punto di vista tecnico, AJAX consiste essenzialmente nell'utilizzo dell'oggetto XMLHttpRequest per inviare richieste http al web server e nell'uso di Javascript per valutare e visualizzare sul client i dati ottenuti dal server.

L'oggetto XMLHttpRequest

La base di tutte le applicazioni AJAX è l'oggetto XMLHttpRequest. Non tutti i moderni browser lo supportano nativamente: in IE fino alla versione 6 è necessario utilizzare l'oggetto ActiveX[45].
La creazione dell'oggetto deve quindi avvenire in modo differente a seconda del browser utilizzato.
Per fare questo può essere utilizzata la funzione seguente:

```
// funzione per assegnare un oggetto XMLHttpRequest
function assegnaXMLHttpRequest()
{
var
        //variabile di ritorno, di default=null
        XHR = null,
        //informazioni sul nome del browser
        browserUtente = navigator.userAgent.toUpperCase();
        //se l'oggetto XMLHttpRequest è supportato nativamente dal browser
        if(typeof(XMLHttpRequest) === "function" || typeof(XMLHttpRequest) === "object")
                        XHR = new XMLHttpRequest();
                else
                  //filtriamo la versione 4 di IE
                   if(window.ActiveXObject && browserUtente.indexOf("MSIE 4") < 0) {
                        //se il browser è IE6
                        if(browserUtente.indexOf("MSIE 5") < 0)
                                XHR = new ActiveXObject("Msxml2.XMLHTTP");
                        Else
                                //il browser è IE 5 o IE 5.5
```

[45] Notare che IE 4 supporta malamente anche l'oggetto ActiveX.

```
                                XHR = new ActiveXObject("Microsoft.XMLHTTP");
                        }
        return XHR;
};
```

La funzione ritorna l'oggetto XMLHttpRequest (creato differentemente a seconda dei browser) nel caso di browser riconosciuti come validi per l'utilizzo di AJAX, restituisce null in caso contrario.

Una volta definito l'oggetto, è possibile utilizzare metodi e proprietà per poterlo sfruttare a seconda delle necessità.
La lista dei metodi è diversa da browser a browser, per questo in genere si usano solo quelli generalmente supportati da tutti. Questi sono:

- Open
- setRequestHeader
- send
- abort
- getAllResponseHeaders
- getResponseHeader

Invece, la lista delle proprietà comunemente supportate dai vari browser è:

- onreadystatechange
- readyState
- responseText
- responseXML
- status
- status Text

Qui di seguito vedremo come questi metodi e parametri vengano utilizzati per effettuare le richieste al server, ricevere e gestire le informazioni ricevute in risposta da esso.

Effettuare la richiesta AJAX al server

Prima di tutto occorre ricordare che esistono due metodi per l'invio dei dati: GET e POST.
Scegliendo GET le variabili vengono appese all'indirizzo della pagina selezionata, mentre utilizzando POST queste verranno inviate in modo invisibile e senza i limiti del GET.
Il limite principale del GET è il numero di caratteri che si possono inviare, solitamente pari ad un massimo di 256 caratteri. Per il metodo POST il limite può essere imposto dal server dell'host che fornisce il servizio, configurato per non ricevere più di un certo numero di MB di dati per ogni richiesta POST, solitamente pari a 8MB, limite trascurabile per la maggior parte delle applicazioni.

Richiesta GET

Per inviare una richiesta http al server utilizzando il metodo GET è necessario compiere tre operazioni:

- inizializzazione dell'oggetto XMLHttpRequest
- definire una funzione di richiamo (call-back) quando arrivano i risultati
- inviare la richiesta

La prima operazione viene effettuata per mezzo del metodo open() avente la seguente sintassi: open(method, url). Esso prevede 2 parametri :

- method: parametro di tipo stringa indicante il metodo di invio dati
- url: è il nome della pagina a cui viene inviata la richiesta seguita dai parametri che si vogliono inviare

La seconda operazione viene effettuata assegnando alla proprietà onreadystatechange dell'oggetto un event handler function. Tale gestione viene chiamata ogni volta che la proprietà readyState di XMLHtppRequest cambia.
L'invio viene effettuato con il metodo send, la cui sintassi è la seguente: send(data).
Forniamo qui di seguito un esempio di richiesta GET:

```
var ajax= assegnaXMLHttpRequest() ;
if(ajax)
{
  ajax.open("get","ajax.html?user=pippo&tipo=1");
  ajax.onreadystatechange = handleResponse;
  ajax.send(null);
}
```

Richiesta POST

Quando viene inviata una richiesta GET, tutti i parametri fanno parte dell'URL. Con POST invece, questi dati sono inviati nel corpo della richiesta HTTP. Questo può essere fatto semplicemente inserendoli come parametri dei metodi send.
Bisogna inoltre ricordarci di impostare le corrette intestazioni HTTP tramite l'istruzione ajax.setRequestHeader("content-type", "application/x-www-form-urlencoded"), prima di effettuare la richiesta, altrimenti non sarà possibile accedere a questi parametri dal lato server.

Forniamo qui di seguito un esempio di richiesta POST:

```
var ajax= assegnaXMLHttpRequest() ;
if(ajax)
{
  ajax.open("post","logging.php")
  ajax.onreadystatechange = handleResponse;
  ajax.send("user=pippo");
}
```

Ricevere la risposta dal server

Tramite la proprietà onreadystatechange è possibile definire un event handler.
Tale event handler è una funzione che viene chiamata ogni volta che cambia la proprietà readyState dell'oggetto XMLHttpRequest. readyState è una proprietà che rappresenta lo stato della richiesta per mezzo di una valore intero che va da 0 a 4. Esistono cinque stati:

- 0 Uninitialized: l'oggetto XMLHttpRequest esiste, ma non è stato richiamato alcun metodo per inizializzare una comunicazione
- 1 Open: è stato chiamato il metodo open() ma il metodo send non è ancora stato eseguito
- 2 Sent: il metodo send() è stato eseguito ed ha effettuato la richiesta
- 3 Receiving: i dati in risposta cominciano a essere letti
- 4 Loaded: l'operazione è stata completata

Questo ordine non è sempre identico e non è sfruttabile allo stesso modo su tutti i browser.

L'unico stato supportato da tutti è il quarto, che a prescindere dalla riuscita dell'operazione, indica che l'operazione è stata completata e che il cambio di stato non andrà oltre quel valore. Per sapere effettivamente se lo scambio dati è avvenuto con successo questa variabile non è quindi sufficiente. Occorre testare il valore della proprietà status, che contiene esattamente il codice http di risposta restituito dal server. Se l'interazione è avvenuta con successo la variabile status assume il valore 200.

Le proprietà responseText e responseXML contengono i dati restituiti dal server a operazione ultimata.
La proprietà responseText contiene un dato di tipo stringa mentre responseXML contiene un oggetto XML qualora i dati restituiti siano un documento XML, null altrimenti.
Molti sviluppatori utilizzano sempre la proprietà responseText a prescindere dalla natura dei dati: nel caso i dati restituiti siano un documento XML, a responseText verrà comunque assegnata la sua rappresentazione testuale.
Nel caso di dati strutturati sembra comunque più elegante la scelta della proprietà responseXML, che contiene un oggetto Javascript XML DOM. Accedere agli elementi del documento XML è del tutto simile ad accedere agli elementi del DOM da Javascript con i metodi getElementsByTagName() , getAttribute(), documentElement().

Di seguito proponiamo una semplice funzione di gestione che una volta fatti gli opportuni controlli mostra le informazioni inviate dal server.

```
function handleResponse()
{
  if(ajax.readyState == 4)
   {
     if(ajax.status == 200)
            alert(ajax.responseText);
   }
}
```

Richiesta asincrona/sincrona

Solitamente le richieste http attraverso XMLHttpRequest sono asincrone.
Questo significa che durante la richiesta il browser continua l'esecuzione del codice e permette all'utente di continuare a interagire con la pagina.
Lo stato di avanzamento della richiesta viene monitorato attraverso l'evento onreadystatechange chiamato ogni volta che cambia lo stato di avanzamento della richiesta che può essere gestito da una funzione di call back da noi creata.

Se impostiamo il terzo parametro del metodo open() a false, la richiesta diventa sincrona: l'esecuzione dello script viene arrestata fino a quando i dati non tornano dal server. L'esecuzione di javascript si ferma all'istruzione send e, una volta ricevuti i dati, prosegue con il codice seguente di elaborazione del risultato.
In caso di richiesta sincrona alcuni browser (come firefox) non scatenano l'evento onreadystatechange, anche perché non ha molto senso, mentre altri come IE lo scatenano ugualmente.
Di seguito proponiamo un esempio di richiesta sincrona:

```
var ajax= assegnaXMLHttpRequest() ;
if(ajax)
{
  ajax.open("post","logging.php")
  ajax.onreadystatechange = handleResponse;
```

```
  ajax.send(“user=pippo”);
  document.getElementById(“output”).innerHTML= “Returned data” + XMLHttpRequest;
}
```

9.3.2.2 *Il nostro uso di AJAX*

Per l’invio dei dati si è pensato di sostituire l’applet con un AJAXscript e di utilizzare il metodo POST dato che esso ci consente di inviare più di 256 caratteri.
Effettuando alcune prove si è scoperto che mentre IE e Mozilla Firefox accettano il valore TRUE come parametro, esso non è accettato da Safari. Si è quindi fatto un browser detection in modo che quando il browser è Safari si imposta il parametro a false.

```
function flush()
{
        //assegna oggetto AJAX
        ajax=assegnaXMLHttpRequest();
        if(ajax)
        {
        //impostazione richiesta asincrona in POST sul file specificato
        var browser=navigator.userAgent.toUpperCase();
        if(browser.indexOf("SAFARI")<0)
                {ajax.open("post","logging.php", true);}
            else
                {ajax.open("post","logging.php", false); }
        //imposto il giusto header
        ajax.setRequestHeader("content-type", "application/x-www-form-urlencoded");
        ajax.onreadystatechange = handleResponse;
        //effettuo la richiesta
        ajax.send("log="+log);
        }
        else
         alert("IL BROWSER NON SUPPORTA AJAX");
}

function handleResponse()
{
 if(ajax.readyState == 4)
  {
   if(ajax.status == 200)
          alert(ajax.responseText);
  }
}
```

Teoricamente lo script avrebbe dovuto funzionare anche con il browser di Nokia e il browser di iPhone. Tuttavia, effettuando delle prove, ci siamo resi conto che il trasferimento non avveniva nel caso di Nokia.

Facendo un po' di ricerca sul web riguardo l'argomento ci si è accorti di utenti che lamentano il mancato funzionamento dell'impostazione degli headers[46], fondamentale nel caso in cui si voglia effettuare una richiesta in modalità POST.
Si è quindi deciso di provare a modificare la funzione in modo da effettuare più richieste GET successive di massimo 256 caratteri ciascuna, anziché un invio completo.

```
var num=0;
var tot=0;
var l=250;

function flush()
{
        num++;
        //determine il numero di richieste get che devo effettuare
        var numchar=log.length;
        div=numchar/l;
        var parteInt  = Math.floor(div);
        var resto=numchar%l;
        tot=parteInt+((resto != 0) ? 1 : 0);
        var index=0;
        if(num==1)
                index=0;
          else
                index=l*(num-1);
        if (num!=tot)
                stringa=log.substr(index, l);
          else
                stringa=log.substr(index);
        var primoblocco="logging.php?log="+stringa+"&num="+num+"&tot="+tot+"&t="+idTest;
        if(ajax)
        {

        ajax.open("get",primoblocco, true);
        ajax.onreadystatechange = handleResponse;
        //effettuo la richiesta
        ajax.send(null);
        }
        else
          alert("IL BROWSER NON SUPPORTA AJAX");
}

function handleResponse()
{
if(ajax.readyState == 4)
 {
```

[46] http://discussion.forum.nokia.com/forum/showthread.php?t=125900
http://discussion.forum.nokia.com/forum/showthread.php?p=444927

```
 if(ajax.status == 200)
        {
        if(num==tot)
                {
                 alert(ajax.responseText);
                parent.location="stop.html";
                }
            else
                {
                flush();
                }
        }
 }
}
```

Nello script si dichiarano inizialmente delle variabili globali: num che tiene il conto del numero di invii che sono stati effettuati, tot inizialmente impostata a 0 ma destinata a contenere il numero totale di pacchetti di dati che devono essere inviati, l che consente di impostare il numero di byte che si vogliono inviare alla volta.
La funzione flush(), chiamata per la prima volta al momento della terminazione del test, si occupa di:

- calcolare il numero di pacchetti di *l* byte ciascuno che devono essere inviati (numero di invii da effettuare per trasferire al server tutto il contenuto della variabile log)
- estrarre dalla variabile *log* la *num*-esima sottostringa di *l* byte (solo l'ultima stringa potrà avere un numero di byte diverso da 250)
- inviare al server il contenuto del file di log e di altre informazioni necessarie (il numero di invii effettuati, il numero totale di pacchetti, l'id del test)

Facendo delle prove ci siamo resi conto che il trasferimento dei dati avviene correttamente con i browser desktop e con il browser di iPhone mentre con il browser di Nokia sembra che ci siano ancora problemi: alla pressione del link stop viene inviato solo il primo pacchetto al server. Facendo dei test di debugging ci siamo resi conto che la chiamata alla funzione flush() presente in handleResponse() non viene mai chiamata. Affinché tutti i dati vengano trasferiti è necessario premere più volte il bottone stop fino a che non compare un messaggio che attesta l'avvenuto trasferimento di tutti i dati. Crediamo che questo comportamento sia dovuto ad un bug, dato che in tutti i browser funziona.

9.3.3 Software lato server che riceve i dati e salva su file

Nel software originario, una volta che il test viene terminato, l'applet invia le informazioni ad una servlet che si occupa di salvarle su un file di log preceduti da una opportuna intestazione.
Dato che abbiamo modificato il modo in cui si effettua l'invio dei dati, non è più possibile sfruttare questa servlet programmata per:

- Ricevere i dati con il metodo POST
- Ricevere tutti i dati in un'unica volta

Si rende quindi necessario la modifica della servlet oppure la sostituzione di questa con qualche altra tecnologia lato server che si occupi di ricevere i dati un po' alla volta, e solo alla fine salvarli su un file.

Noi si è optato per l'uso di PHP:

```
<?php
 session_start();
 $log=$_GET['log'];
 $tot=$_GET['tot'];
 $num=$_GET['num'];
 $idTest=$_GET['t'];
 $ip=$_SERVER['REMOTE_ADDR'];

 if($num==1)
 {
  $data=date ("D d F Y H:i:s");
  $hostname=gethostbyaddr($_SERVER['REMOTE_ADDR']);
  $log="Test logging:\r\nHtime: " . $data ."\r\n"."ADDR ".$ip."\r\n"."HOST ".$hostname."\r\n".$log;
 }

 $_SESSION['log']=$_SESSION['log'].$log;

  if($num==$tot)
  {
  $_SESSION['log']=str_replace("*","\r\n",$_SESSION['log']);
  $filename=$ip." ".time().".txt";
  if(!file_exists("logs/".$idTest))
                    Mkdir("logs/".$idTest,0777);
  $scrivi_file=fopen("logs/".$idTest.'/'.$filename,"w");
  fwrite($scrivi_file,$_SESSION['log']);
  fclose($scrivi_file);
  echo ("Il test e' terminato: i dati sono stati registrati");
  }
?>
```

Lo script riceve, oltre alle informazioni di log, anche altre informazioni importanti: il numero totale di pacchetti in cui è stato suddiviso il messaggio dal client, il numero di pacchetti ricevuti fino a quel momento. Dopodiché, se si tratta del primo pacchetto ricevuto, prima della memorizzazione delle informazioni su una variabile del server, inserisce una intestazione contenente varie informazioni (data, hostname, ecc.). In tutti gli altri casi lo script si limita al salvataggio delle informazioni sul server. Solo al momento in cui il numero di pacchetti ricevuti è uguale al numero totale di pacchetti, si procede al salvataggio del file di log sul server.
Per garantire che ad ogni invio, al client venga associata la stessa variabile (*$_SESSION['log']*) contenente i dati ricevuti precedentemente si è scelto di utilizzare il meccanismo delle sessioni.
Una caratteristica di http è infatti quella di essere un protocollo privo di stato, per cui il web server non riesce ad associare in automatico i dati ad una determinata richiesta.
Varie soluzioni sono state proposte per superare questo limite (utilizzo di cookies, passaggio di dati tramite query string o tramite campi di moduli). Le sessioni rappresentano un'altra possibile soluzione allo stesso

problema. Esse consentono di identificare un determinato client per un determinato periodo di tempo e di associargli dei dati[47].

[47] Quando un utente accede per la prima volta ad una pagina PHP impostata per creare una sessione, a questi viene associato un ID univoco che verrà utilizzato da PHP come chiave per recuperare l'array $_SESSION salvato per quel determinato utente. Tale ID generalmente viene salvato da PHP all'interno di un cookie sul client. Ad ogni nuova richiesta al server, il client invierà anche il proprio ID per farsi riconoscere.

10. PROBLEMA 3: RIADATTAMENTO DELL'INTERFACCIA GRAFICA DEL LOGGING TOOL

10.1 Problema

Se proviamo a visualizzare il tool così com'è sui due dispositivi mobili, otteniamo i seguenti risultati:

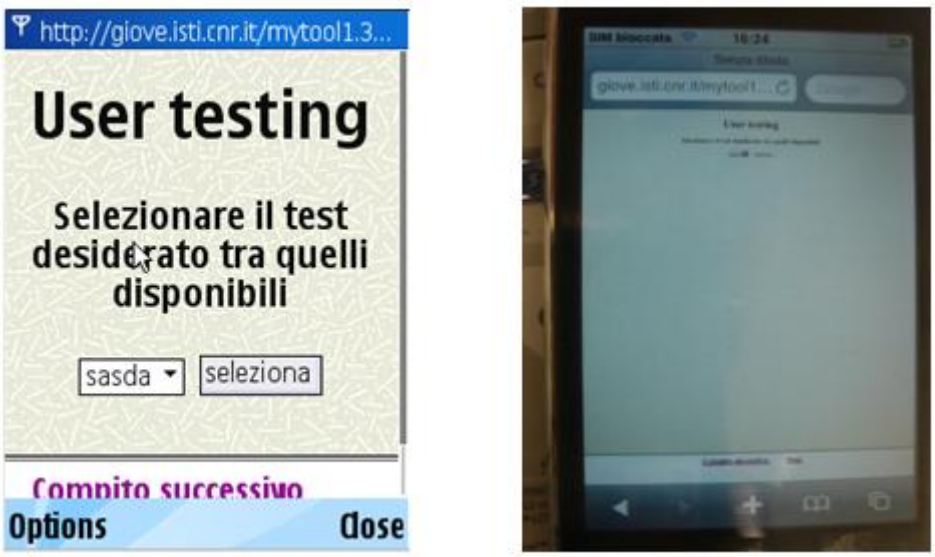

Figura 42 Prima prova di visualizzazione dell'interfaccia grafica del logging tool

Come si può vedere su Nokia N95 la pagina non necessita di adattamenti[48], mentre con Safari on iPhone il testo appare troppo piccolo e poco leggibile.
Nel paragrafo seguente cercheremo di spiegare per via teorica il perché di questa differenza. Il risultato ottenuto è dovuto al diverso comportamento dei browser nella visualizzazione di siti web (in particolare un modo differente di intendere il concetto di viewport).

10.2 Analisi del problema

10.2.1 La funzione di formattazione dei browser web

Prima di entrare nello specifico del problema riteniamo sia utile focalizzare l'attenzione su come generalmente si comportano i browser nella resa a video di un documento web.
Il rendering di un documento in un browser web moderno si basa sul modello di formattazione CSS2.
La funzione di formattazione nel compiere le sue operazioni, deve rispettare alcuni *vincoli interni* e alcuni *vincoli esterni* .
I vincoli interni sono quelli imposti dalla struttura e dagli stili del documento, per esempio:

---

[48] La prova di visualizzazione sul browser di Nokia è stata effettuata utilizzando il simulatore di "S60 3rd Edition SDK for Symbian OS" fornito da Nokia. Facendo delle prove sul dispositivo vero e proprio abbiamo ottenuto un risultato diverso. Il risultato è forse dovuto alle diverse versioni di browser. Crediamo che la soluzione a questo problema si possa ottenere aggiornando il firmware del Nokia N95.

- Se un box contiene un'immagine non può essere più stretta dell'immagine
- Se il documento imposta la larghezza di una colonna ad un certo numero fisso di pixels, essa non può diventare più stretta del valore specificato.
- La dimensione minima di un box contenente testo è dato dalla larghezza della parola più lunga in esso contenuta

I vincoli esterni sono quelli imposti dal browser e l'ambiente. Il più importante vincolo esterno è la larghezza del viewport del browser, cioè l'area a disposizione per la visualizzazione della pagina web.
Il processo di formattazione cerca di rendere il documento largo quanto il viewport, rispettando contemporaneamente i vincoli interni.
Capita spesso che questo non sia possibile: i vincoli interni spesso limitano la grandezza minima dei vari box che costituiscono il layout generato. In questi casi, il documento diventa più largo (o più stretto) del viewport e vedere l'intero documento potrebbe richiedere scrolling orizzontale.

Per capire meglio il concetto si propone un esempio: il sito del corso di laurea in informatica umanistica. Proviamo a visualizzare il sito su un browser desktop, modificando la larghezza del viewport.
Su browser desktop la viewport è definita dalla grandezza della finestra[49].

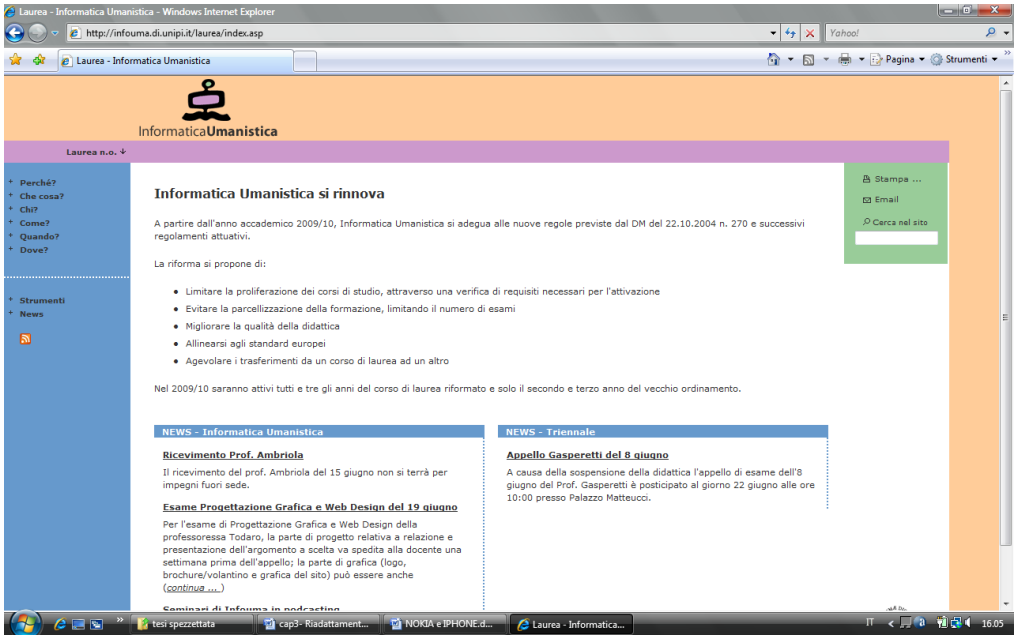

Figura 43 Visualizzazione del sito infouma.di.unipi.it su browser desktop

Se proviamo a ridimensionare il viewport riducendo la finestra otteniamo:

[49] Ridimensionando la finestra del browser si ridimensiona il viewport

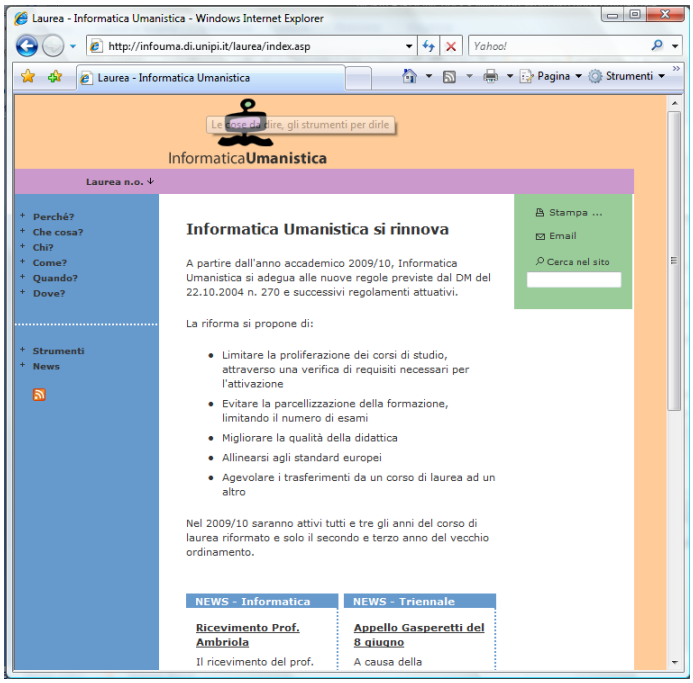

Figura 44 Visualizzazione del sito infouma.di.unipi.it su browser desktop dopo aver effettuato un ridimensionamento della finestra

La funzione di formattazione ha cercato di rendere il documento largo quanto la larghezza del viewport: si è ridotta la larghezza del contenitore principale, si è ridotta la larghezza del testo e delle colonne delle news. Questo è stato reso possibile dal fatto che questi elementi non hanno una larghezza fissa in pixel. Si noti che alcuni elementi hanno mantenuto la stessa dimensione: per esempio, la colonna azzurra a sinistra, il box verde in alto a destra. Probabilmente perché per questi elementi sono state impostate delle dimensioni fisse che ne impediscono il ridimensionamento in base alla finestra del browser.

Se proviamo a ridurre ulteriormente la finestra del browser arriviamo ad un punto in cui i vari elementi non riducono più la loro dimensione per adattarsi alla finestra. Questo avviene perché non è possibile scendere sotto una larghezza minima (solitamente determinata dal contenuto del box stesso).
In questi casi, il documento ottenuto come output della funzione di formattazione è più largo del viewport. Per vedere l'intero documento, il browser fornisce la barra per lo scrolling orizzontale:

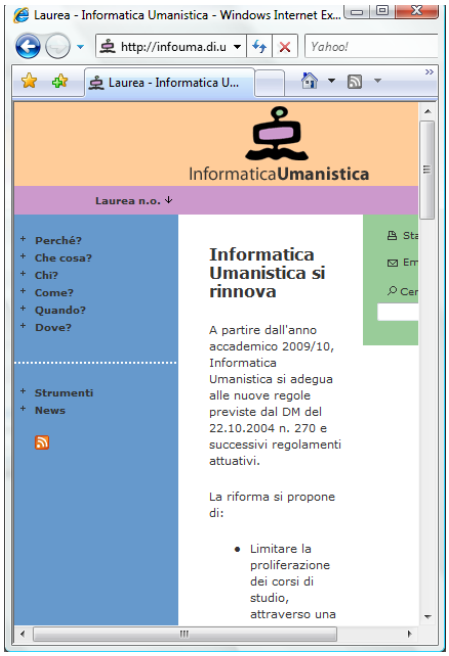

Figura 45 Visualizzazione del sito infouma.di.unipi.it su browser desktop con finestra del browser ridotta fino alla comparsa della barre di scorrimento

10.2.2 Il concetto di viewport sul browser di Nokia e di iPhone

Lo stesso procedimento spiegato sopra vale anche per i browser mobili.

Esiste però una differenza fondamentale: sui browser mobili le finestre non sono ridimensionabili. La finestra del browser occupa tutto il display.
Se per viewport intendiamo, come abbiamo fatto nel caso dei dispositivi desktop, la finestra del browser, possiamo affermare che il viewport coincide con lo schermo del dispositivo meno elementi che fanno parte dell'interfaccia come scrollbars ecc.

È quello che fa il browser di Nokia.
La funzione di formattazione cerca di ridurre la larghezza della pagina in modo che rientri nel display. Se la pagina risultante non entra all'interno del viewport, il browser fornisce la possibilità di navigare nella pagina facendo uso dello scrolling.

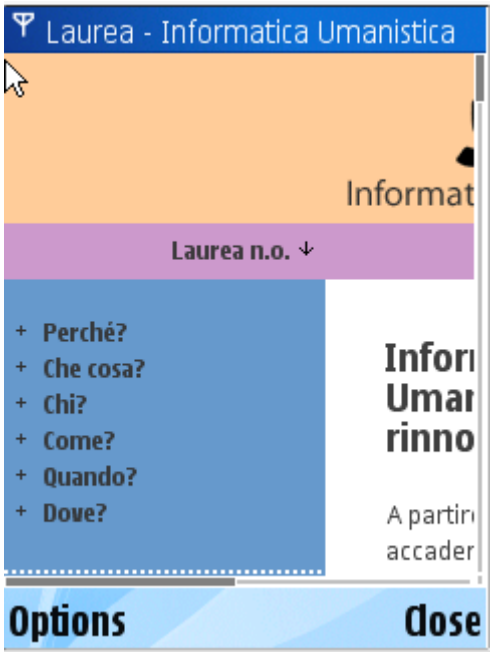

Figura 46 Visualizzazione del sito infouma.di.unipi.it sul browser di Nokia

Diverso è invece il concetto di viewport per Safari on iPhone. Esso non ha più a che fare con le dimensioni effettive della finestra del browser (che è fissa), ma è un generico riquadro avente certe dimensioni.
Di default la larghezza del viewport è di 980px[50], ma il progettista può impostarla impostando le dimensioni desiderate e altre proprietà all'interno del metatag viewport[51], della pagina che si vuole visualizzare.
La funzione di formattazione del browser prenderà come vincolo esterno le dimensioni di questo riquadro virtuale e non quelle della finestra, e compierà tutte le operazioni che abbiamo descritto sopra.
Una volta renderizzata la pagina web all'interno di questo riquadro virtuale, essa viene automaticamente scalata (settando la proprietà initial-scale) così che l'intera pagina entri nello schermo di iPhone.

[50] 980px è considerata la larghezza media delle pagine web.
[51] Non fa parte dell'HTML standard. è un'estensione di Apple

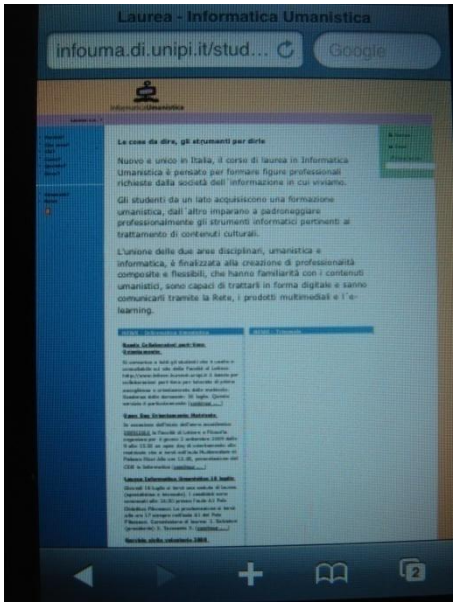

Figura 47 Visualizzazione del sito infouma.di.unipi.it sul browser di iPhone

Adesso apparirà chiaro il perché dei due diversi risultati: esso è determinato proprio da due modi diversi di concepire il viewport.

10.3 Riadattamento dell'interfaccia su iPhone

Intuitivamente, se vogliamo che i due browser visualizzino il sito allo stesso modo, occorrerà che il viewport all'interno del quale viene "disegnata" la pagina sia lo stesso. In particolare, vogliamo che il viewport su iPhone corrisponda a quello di Nokia N95 che è largo 240px.

Fortunatamente iPhone fornisce la possibilità di impostare valori diversi per il viewport facendo uso del metatag viewport. Nella pagina contenente la struttura a frame del sito si è dunque inserito il metatag viewport con il parametro width impostato a 240px.

```
<head>
<title></title>
<meta name="viewport" content="width=240" initial-scale="1.0">
</head>
<frameset rows="92%, *">
<frame name="site" src="go.php" title="sito web"  />
<frame name="hidden" src="hidden.html" title="test" />
</frameset>
</html>
```

Questo ha permesso di ottenere il seguente risultato.

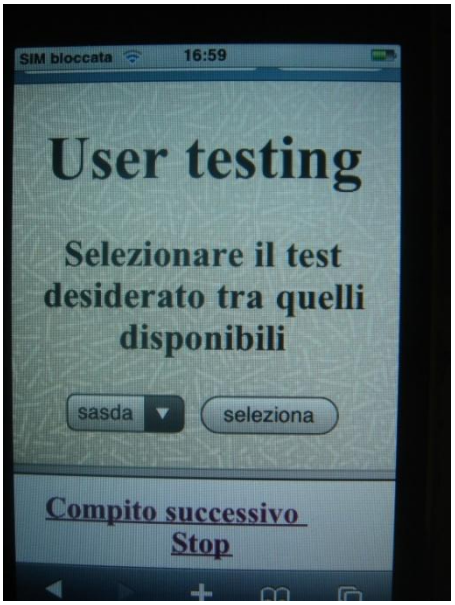

Figura 48 Prova di adattamento per iPhone

Tuttavia, un nuovo problema si determina al momento in cui andiamo a visualizzare il sito su cui eseguire il task.

Figura 49 Prova di adattamento per iPhone

Avendo settato il viewport a 240px la pagina del sito viene visualizzata in questo modo. Ai fini del test noi vogliamo però che le pagine siano visualizzate come avviene per default e cioè con una viewport di 980px. Per fare questo abbiamo creato un javascript che al momento in cui viene caricata la pagina accede al DOM e modifica il tag viewport.

```
<script type="text/javascript" src="/ultimo/siti/script.js"></script><script type="text/javascript">
 var meta=parent.document.getElementsByTagName('meta');
 meta[0].setAttribute('content','width=980');
 </script>
```

Tale script deve essere inserito all'interno di ogni pagina web del sito da testare.
Così facendo il sito viene visualizzato correttamente:

Figura 50 Prova di adattamento per iPhone

11. PROBLEMA 4: RILEVAMENTO DI ALCUNI GESTI SPECIFICI DI IPHONE

11.1 Il modello di interazione di iPhone

11.1.1 Touchscreen

La grande novità introdotta da iPhone è l'interazione tramite tocchi sullo schermo. Gli utenti utilizzano le loro dita per interagire con iPhone per selezionare, navigare, leggere contenuto web e usare applicazioni. Ci sono parecchi vantaggi derivanti dall'uso delle dita per interagire con il dispositivo:

- le dita sono immediatamente disponibili
- le dita sono capaci di compiere gesti compositi, dato che iPhone è multitouch
- danno all'utente un senso di immediatezza e connessione al dispositivo che è impossibile raggiungere utilizzando un tipo di input esterno come un mouse.

Ci sono, tuttavia, alcuni svantaggi dovuti al fatto che il dito non è un mouse:

- Un dito è generalmente più grande e quindi meno accurato del tradizionale dispositivo di puntamento: non è possibile fare selezioni precise
- Il dito non è persistente: mentre il puntatore del mouse è sempre sul display, lo stesso non vale per il dito. Per questo motivo non si può fare affidamento su tecniche che presumono che il mouse si sposti da un punto A ad un punto B attraverso tutto lo spazio nel mezzo.
  In particolare questo determina il fallimento di alcune categorie di eventi: drags e hover.

11.1.2 Come si comporta iPhone quando l'utente interagisce

Per meglio capire che cosa succede al momento che questi gesti vengono effettuati da parte dell'utente conviene distinguere due livelli:

- ➔ Livello dispositivo/OS
- ➔ Livello applicazione

1) Quando l'utente tocca lo schermo del dispositivo, il sistema operativo di iPhone registra questi tocchi (UITouch objects), insieme a delle proprietà temporali e spaziali.
   Una volta riconosciuto l'insieme di tocchi appartenenti alla stessa sequenza[52], iPhone li impacchetta all'interno di un oggetto chiamato UIEvent e lo pone all'interno della coda degli eventi dell'applicazione corrente.

[52] L'insieme dei tocchi che vanno da quando il primo dito tocca lo schermo e quando l'ultimo dito lascia lo schermo costituisce una MULTI-TOUCH SEQUENCE

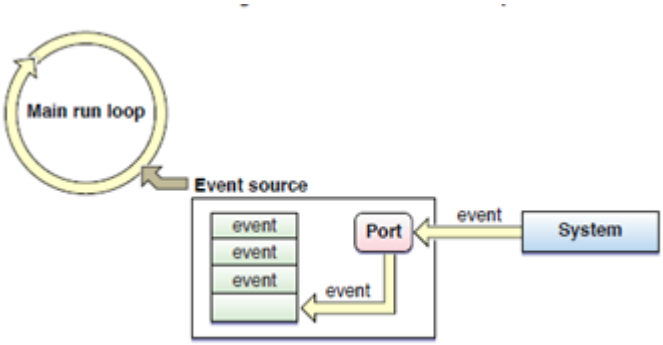

Figura 51 Processing events in the main run loop

2) L'oggetto UIApplication, che sta gestendo l'applicazione, estrae un evento dalla cima della coda e lo invia perché sia gestito. Tipicamente l'evento viene inviato all' oggetto UIWindow, che rappresenta la finestra principale dell'applicazione, e questo lo invia al **first responder**[53] perché venga gestito. Se il first responder non gestisce l'evento, passa l'evento al prossimo responder nella **catena dei responder** per vedere se può gestirlo. Perché un responder object sia in grado di gestire un evento è necessario che implementi un metodo (dell'interfaccia UIResponder) per la gestione di eventi.

11.1.3 Interazione con Safari on iPhone

Abbiamo visto come un'applicazione generica può gestire gli eventi prodotti dall'interazione con l'utente.
In questo paragrafo vedremo un'applicazione specifica: Safari on iPhone.
Safari on Iphone è programmato per riconoscere determinati gesti all'interno di una sequenza di tocchi e "reagire" di conseguenza, con un determinato comportamento. I gesti possono essere di due tipi: "one finger" o "two-finger", a seconda che coinvolgano l'uso di uno o più dita.
Nella tabella seguente riportiamo i gesti riconosciuti da Safari, una loro descrizione e le azioni svolte per default dal browser.

Tabella 9 Gesti riconosciuti da iPhone, descrizione e azione di default del browser collegata

gesti	descrizione	azioni
tap	L'utente tocca e rilascia velocemente lo schermo	Per premere o selezionare un controllo o un link (analogo al singolo click del mouse) Ricevi l'evento onClick per questo gesto
double tap	Consiste in due veloci taps successivi	Per fare zoom-in e centrare un blocco di contenuto o una immagine Per fare zoom-out (se hai già zoomed-in)
flick	appoggiare il dito sullo schermo e "swipe" velocemente nella direzione desiderata	Per fare scrolling verticale (scroll) e orizzontale (pan)

[53]Un **responder object** è un oggetto che ha la potenzialità di rispondere a eventi e gestirli.
La classe UIResponder è la base per tutti i responder object: essa fornisce un'interfaccia programmativa non solo per la gestione di eventi ma anche per il comportamento personalizzato dei responder objects.
Notare che la classe UIApplication, UIView, e tutte le classi che discendono da UIView (incluso UIWindow) ereditano direttamente o indirettamente da UIResponder.
Il **first responder** è il *responder object* nell'applicazione (solitamente un UIView object) che è il destinatario corrente dei tocchi.

		rapidamente
Drag	appoggiare il dito sullo schermo e muoverlo velocemente nella direzione desiderata, senza lasciarlo dallo schermo	Per fare scrolling verticale (scroll) e orizzontale (pan)
Pinch open	Mettere il pollice e un dito (o altre due dita) vicini sullo schermo e allontanarli uno dall'altro senza che lascino lo schermo	Per fare zoom in
pinch out	Mettere il pollice e un dito (o altre due dita) lontani sullo schermo e avvicinarli uno dall'altro senza che lascino lo schermo	Per fare zoom out
Touch and hold	Toccare e trattenere il dito fermo sullo schermo, finché non è visualizzata l'informazione o è avvenuta l'azione	Per mostrare informazioni aggiuntive (es:indirizzo di un link), ingrandire il contenuto sotto le dita, o svolgere azioni specifiche e funzionalità delle applicazioni buit-in
two finger scroll	È un drag svolto con due dita che si muovono insieme nella stessa direzione	Per fare scroll up e down (a seconda della direzione del movimento) all'interno di una text area, un frame inline, o un elemento con possibilità di overflow Puoi ricevere l'evento mousewheel per questo gesto

11.1.3.1 Eventi tradizionali

Per alcuni[54] di questi gesti, (e purché l'elemento interessato rispetti certe condizioni[55]), Safari on iPhone invia un **evento** direttamente all'applicazione web.

Tali eventi corrispondono ai tipici eventi solitamente generati dal mouse sui browser desktop (come *mousemove, mouse down, mouseup, mouseover, mouseout, click*), e che per questo motivo abbiamo definito **standard**. Anche le form e i documenti generano I tipici eventi che ci si aspetterebbero su desktop (*blur, focus, load, upload, reset,submit, change, abort*). Tutti gli eventi che non sono citati sopra sono non supportati.

Su iPhone gli eventi non sempre sono generati nello stesso momento dei browser desktop. Per capire meglio il comportamento di iPhone si vedano i seguenti flowcharts che scompongono il gesto nelle singole azioni e mostrano il momento preciso di generazione dell'evento.

[54] È importante notare, che non si riceve un evento per la maggior parte di questi gesti. Piuttosto, la maggior parte di questi gesti sono interpretati da Safari on iPhone e applicati al modo in cui iPhone mostra i contenuti, invece di essere passati direttamente alla pagina web

[55] La generazione di un evento è condizionale nel senso che dipende dal fatto che l'elemento selezionato è cliccabile o "scrollabile". Un elemento cliccabile è: un link, elemento di un form, area di una mappa immagine, o qualsiasi altro elemento che hanno impostati i gestori *mousemove, mousedown, mouseup, onclick.*
Un elemento scrollabile è un qualsiasi elemento con appropriati stili di overflow, aree di testo, e elementi iframe scrollabili.
Se un elemento non è cliccabile o scrollabile, nessun evento verrà generato.
A causa di questo comportamento da parte di Safari onIphone, potrebbe capitare che alcuni elementi non si comportino come ci si aspetta. È il caso per esempio di alcuni menu che utilizzano l'evento *onmouseover.* Per risolvere questo problema è sufficiente aggiungere `onclick = "void(0)"`, così che l'elemento sia riconosciuto come cliccabile.

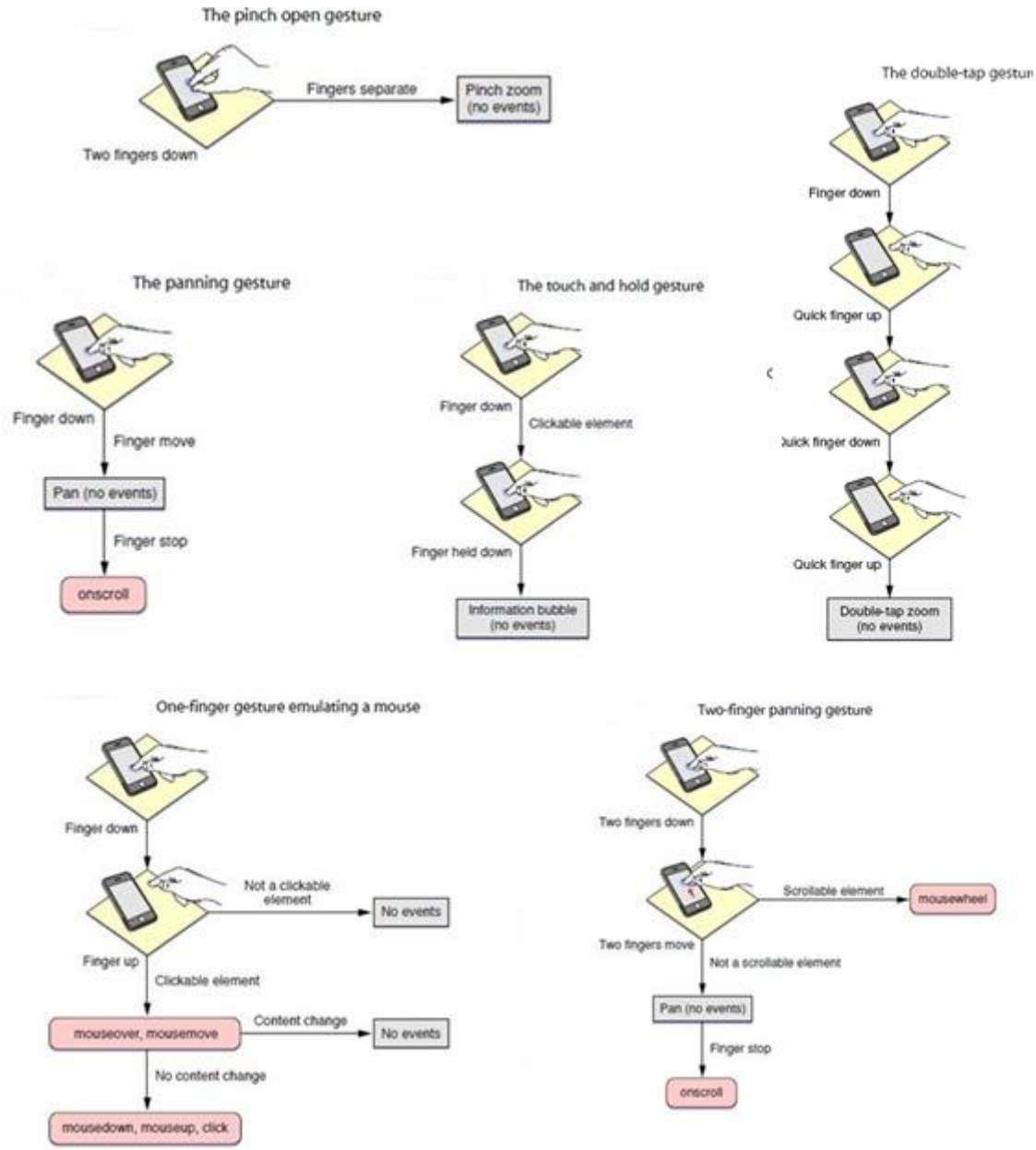

Figura 52 Flowcharts : scomposizione dei gesti nelle singole azioni che li compongono e eventi correlati

11.1.3.2 Eventi specifici di iPhone

Quando le dita si muovono sullo schermo, oltre agli eventi (che io ho definito standard) che emulano quelli generati dai movimenti del mouse, i DOM objects interessati ricevono degli eventi specifici di Iphone (che corrispondono a quelli solitamente ricevuti e gestiti al livello del browser).

Tali eventi possono essere gestiti all'interno della pagina web, ottenendo la possibilità di personalizzare la gestione degli eventi in modo simile a quanto avviene per le applicazioni native. Si può addirittura disabilitare il comportamento di default di Safari così che esso non interferisca con la nostra gestione personalizzata degli eventi al livello dell'applicazione web.

Gli eventi specifici di iPhone sono:

touchstart – generato quando un dito tocca l'iPhone

touchmove – generato quando un dito si sposta sullo schermo
touchend – generato quando un dito lascia il telefono
touchcancel- generato quando il sistema cancella il tocco
gesturestart- generato quando due o più dita toccano l'iPhone
gesturechange- generato quando due o più dita si muovono sullo schermo
gestureend- generato quando uno o nessun dito è rimasto sullo schermo

Per capire meglio l'ordine in cui tali eventi sono generati si supponga che un utente compia un generico gesto con due dita. Il flusso di eventi generati per questo gesto sono:

1. **touchstart** per il dito 1. Inviato quando il primo dito tocca lo schermo
2. **gesturestart** inviato quando il secondo dito tocca lo schermo
3. **touchstart** per Il dito 2. Inviato immediatamente quando il secondo dito tocca lo schermo
4. **gesturechange** per il gesto corrente. Inviato quando entrambe le dita si muovono sullo schermo
5. **gestureend**. Inviato quando il secondo dito lascia lo schermo
6. **touchend** per il dito 2. Inviato immediatamente dopo gestureend quando il secondo dito lascia lo schermo
7. **touchend** per il dito 1. Inviato quando il primo dito lascia lo schermo

Registrazione dei gestori di eventi

È possibile registrare un gestore di evento per questi eventi specifici di iPhone sia all'interno di un attributo HTML (`div ontouchstart="myTouchStart(event);">`) sia tramite Javascript (`.addEventListener("touchstart",myTouchStart,false);`).
Agli event handler viene passato un event-object. L'evento passato al gestore come parametro è un TouchEvent object se stiamo gestendo eventi di tipo touch, è un GestureEvent object se stiamo gestendo eventi di tipo gesture.

Oggetto TouchEvent

Un touch event object incapsula tutti i tocchi (touch objects) che sono correntemente sullo schermo. Ogni touch object corrisponde alla presenza di un dito sullo schermo.
Tale evento possiede una serie di proprietà a cui è possibile accedere:

target – l'oggetto target che ha generato il tocco
changedTouches – un'array contenente tutti i più recenti tocchi cambiati sulla pagina
targetTouches – un'array di tutti i tocchi per l'elemento target
touches – un array di tutti i tocchi sulla pagina[56]

Si noti che tutti i touch events appartenenti alla stessa sequenza sono inviati all'elemento che ha ricevuto l'evento touchstart indipendentemente dalla corrente locazione dei tocchi.

[56] Notare che changedTouches, targetTouches e touches contengono ciascuno liste di tocchi sostanzialmente differenti:

- targetTouches e Touches contengono ciascuno una lista delle dita che sono attulalmente sullo schermo
- changedTouches elenco solo l'ultimo tocco che è avvenuto.

È importante comprendere questa differenza, soprattutto perché, nel caso si lavori con touchend o gestureend event, non si avranno più a disposizione dita sullo schermo. Per cui targetTouches e touches dovrebbero essere vuoti. È comunque possibile vedere l'ultima cosa che è accaduta guardando l'array changedTouches.

Oggetto GestureEvent

Se stiamo lavorando con Gesture events l'event object passato alla funzione di gestione dell'evento.
I gesti hanno anche un enorme vantaggio: aggiungono due nuove proprietà agli event-object:

Tabella 10 I gesti aggiungono due nuove proprietà agli event objects

Property	Summary
`rotation`	How much the fingers have rotated
`scale`	How much a pinch has zoomed in (<1) or out (>1)

Queste proprietà ci permettono di misurare l'ampiezza dello zoom e della rotazione.

Proprietà del touch object

Una volta acceduto al tocco, usando una variabile tipo **event.touches[0],** è possibile accedere a proprietà aggiuntive di quel tocco:

Tabella 11 Proprietà dell'oggetto touch

Property	Summary
`clientX` or `clientY`	X or Y location relative to the current browser screen (absent any scroll offset for the overall web page)
`identifier`	Unique identifying number for the event
`pageX` or `pageY`	X or Y location relative to the overall web page
`screenX` or `screenY`	X or Y location relative to the user's overall computer screen (which is of limited use)
`target`	The target object that generated the touch

La maggior parte di queste proprietà ci dicono dove un tocco è avvenuto sulla pagina, usando una varietà di diversi sistemi di coordinate.
Mettendo questo insieme con le informazioni di base dell'evento che abbiamo discusso, è possibile cominciare a costruire programmi sofisticati che rilevano dove un utente sta toccando lo schermo e compie determinate azioni in base a questa informazione.

11.2 Quali gesti abbiamo deciso di riconoscere nel nostro tool?

Abbiamo visto che l'utente può interagire con iPhone compiendo dei gesti sullo schermo.
Per alcuni di questi gesti Safari on iPhone invia un **evento tradizionale** direttamente all'applicazione web:

- TAP: viene generato l'evento onClick per questo gesto
- FLICK: viene generato l'evento onScroll per questo evento
- TWO FINGER SCROLL: viene generato l'evento mousewheel per questo gesto

Rimangono fuori da questa lista alcuni gesti che sarebbe utile poter registrare durante il test. Tali gesti sono:

- PINCH IN
- PINCH OUT
- DOUBLE TAP

Sfruttando la gestione degli eventi specifici di iPhone si è tentato di aggiungere degli event handler per il riconoscimento di questi gesti.

11.2.1 Riconoscimento di un semplice gesto: il tap

Prima di tentare il riconoscimento di gesti complessi come pinch e double-tap abbiamo provato a riconoscere il gesto più semplice di tutti: il singolo tap.
Il tap consiste nel toccare e rilasciare velocemente lo schermo.
In base a quanto abbiamo esposto nel paragrafo 4.1.2.3, abbiamo riflettuto su quali siano gli eventi specifici di iPhone generati quando un dito tocca e rilascia lo schermo. Essi sono:

1) touchstart: il dito tocca la superficie
2) touchend: il dito lascia la superficie

Si noti che una caratteristica particolare di questo gesto è il fatto che l'evento touchemove non viene mai generato. Si può sfruttare proprio questa proprietà nel riconoscimento di questo gesto come segue:

```
document.addEventListener("touchmove",move,false);
document.addEventListener("touchend",end,false);

var flag=0;
function move(event)
{
 flag=1;
}

function end(event)
{
 if (flag==0)
        script.iPhoneEvent("TAP");
    else
        flag=0;
}
```

11.2.2 Riconoscimento del double tap

Per riconoscere questo gesto abbiamo riflettuto sul fatto che il double-tap consiste in due tap successivi a distanza di tempo ravvicinata sullo stesso elemento. Abbiamo sfruttato questa proprietà nel codice che segue:

```
document.addEventListener("touchmove",move,false);
document.addEventListener("touchend",end,false);

var flag=0;
var first=0;
var firstTap=0;
var firstTaptarget=0;

function move(event)
{
```

```
 flag=1;
}

function end(event)
{
  if (flag==0)
  {
    if (first==0)
     {
       first++;
       firstTap=(new Date()).getTime();
       firstTaptarget=event.target;
     }
else
    {
      first=0;
      taptime=(new Date()).getTime();
      if((new Date()).getTime()-firstTap<1000 && firstTaptarget==event.target)
                                                    script.iPhoneEvent("MULTITAP");
    }
  }
else
   flag=0;
}
```

11.2.3 Riconocimento dei gesti di zoom (pinch-in/pinch-out)

Riconoscere quando l'utente effettua zoom-in o zoom-out sulla pagina è abbastanza semplice se utilizziamo i gesture events.

In base a quanto abbiamo esposto nel paragrafo 4.1.2.3, i gesture events generati quando si compie questo gesto sono:

1) gesturestart: quando le due dita toccano lo schermo
2) gesturechange: quando le due dita si muovono contemporaneamente sullo schermo
3) gestureend: quando le dita rilasciano lo schermo

Si è deciso quindi di gestire l'evento gesturechange e l'evento gestureend.
Quando si verifica gesturechange il gestore legge il valore della proprietà scale dell'oggetto event, che fornisce la misura di quanto è stato fatto zoom in(<1) o zoom-out(>1). Nel gestore gestureend() si è poi controllato il valore della scala e in base a quello abbiamo stabilito se era stato effettuato uno ZOOM IN o ZOOM OUT. La variabile flag è stata usata per controllare che al momento in cui viene richiamato il metodo gestureend() fosse avvenuto prima l'evento gesturechange: tale situazione potrebbe non capitare nel caso in cui due dita toccano lo schermo e subito lo rilasciano.

Forniamo di seguito il codice che si è utilizzato.

```
document.addEventListener("gesturechange",gesturechange,false);
document.addEventListener("gestureend",gestureend,false);
```

```
var flag=0;
var scala=0;

function gesturechange(event)
{
 scala=event.scale;
 flag=1;
}

function gestureend(event)
{
 if(flag)
 {
 if(scala>1)
          script.iPhoneEvent("ZOOM IN");
 else
  script.iPhoneEvent("ZOOM OUT");
 }
 flag=0;
}
```

11.3 Considerazioni finali

Sfruttando la gestione degli eventi specifici di iPhone, siamo riusciti ad implementare una serie di gestori in grado di riconoscere alcuni gesti importanti di iPhone. Tuttavia, per il momento non siamo riusciti ad integrare i vari blocchi di codice mostrati sopra per il riconoscimento simultaneo di tutti questi gesti.

12. PROBLEMA 5: PROBLEMATICA USO DI FRAME

12.1 Problema

Come è già stato detto, l'applicazione risulta essere costituita da due frame: uno contenente le pagine del sito che si vogliono testare, l'altro contenente uno script che si occupa della ricezione, concatenamento e infine invio delle informazioni di log al server. Oltre allo script, questo secondo frame contiene i pulsanti "Compito successivo" e "Stop" che servono rispettivamente per andare al compito successivo e per terminare il test. Il test termina comunque per conto proprio una volta che si sono esauriti i compiti che lo compongono.

Figura 53 Struttura a frame dell'applicazione

La scelta di utilizzare i frame per strutturare il test non è stato un motivo estetico. Essa è stata fatta per due motivi funzionali:

- Consente di mantenere fissi sullo schermo i link "Compito successivo" e "Stop"
- Garantisce la persistenza delle variabili dello script che si occupa della memorizzazione delle informazioni relative agli eventi durante tutta la sessione del test.

E in effetti su tutti i principali browser desktop tutto funziona correttamente: entrambi gli obiettivi sono raggiunti.

Se però andiamo a provare il sito così strutturato sul browser di iPhone e del Nokia N95, ci accorgiamo che qualcosa non va come ci aspetteremmo: il secondo frame non sempre è visibile sullo schermo (mentre non ci sono problemi per quanto riguarda lo script).

Figura 54 I browser mobili non visualizzano il secondo frame fisso nella parte inferiore dello schermo

Questo è dovuto al diverso comportamento dei browser mobili nella visualizzazione delle applicazioni web strutturate a frame. Sui browser desktop ogni frame è caratterizzato da una propria dimensione che viene mantenuta a prescindere dalla grandezza della pagina che viene in esso caricata. Più precisamente, nel caso in cui la pagina sia di dimensioni maggiori, il browser visualizza le barre di scorrimento[57].

Su iPhone e Nokia N95 notiamo che i frame non contengono barre di scorrimento, per cui ciascun frame del frameset viene espanso per adattarsi al proprio contenuto. Questo comportamento rende impossibile mantenere fissi sullo schermo i link nel caso in cui la pagina visualizzata nel frame superiore abbia dimensioni in px maggiori rispetto ad esso.
Ne derivano alcune conseguenze:

- L'utente che vuole andare al compito successivo o terminare il test è costretto a raggiungere la fine della pagina
- Durante questa interazione l'utente sporca il file di log con eventi inutili ai fini del test

12.2 Tentativi di soluzione

Per risolvere il problema occorre trovare un modo alternativo ai frame per mantenere le informazioni fisse sullo schermo. Se ci pensiamo i CSS includono proprietà che consentono il posizionamento fisso degli elementi. Impostando il valore di position a fixed, il box viene sottratto al flusso normale del documento e posizionato tramite le proprietà **top, left, right o bottom.** Il posizionamento avviene rispetto al viewport. La particolarità che ci interessa di questa modalità di posizionamento è il fatto che un box posizionato in modo *fixed,* in caso di scrolling, non scorre insieme al resto del documento, ottenendo lo stesso effetto che si ottiene con i frame.
Abbiamo deciso quindi di provare a nascondere il secondo frame e fornire due link all'interno di un div posizionato in modo fixed nella parte bassa del viewport.
Abbiamo, cioè provato ad inserire all'interno delle pagine del sito da testare il seguente codice:

[57] È lo stesso procedimento svolta dal browser per il rendering della pagina all'interno della finestra. In effetti ogni frame è da considerare come una finestra del browser.

* *
* Compito successivo *
*   *
* Stop *
* *

Questo codice crea un div, e gli attribuisce gli stili specificati come proprietà style. Questo div contiene due link: Compito successivo e Stop. Notare che alla pressione dei link devono essere invocate le stesse funzioni che prima venivano chiamate cliccando sugli omonimi link presenti nel secondo frame (ora nascosto).
Per raggiungere tali funzioni presenti in un frame diverso rispetto a quello in cui ci troviamo adesso è possibile scrivere top.hidden.[nome funzione], dove top indica la finestra principale e hidden è il nome del secondo frame[58].

L'effetto ottenuto, visualizzato su browser desktop è il seguente:

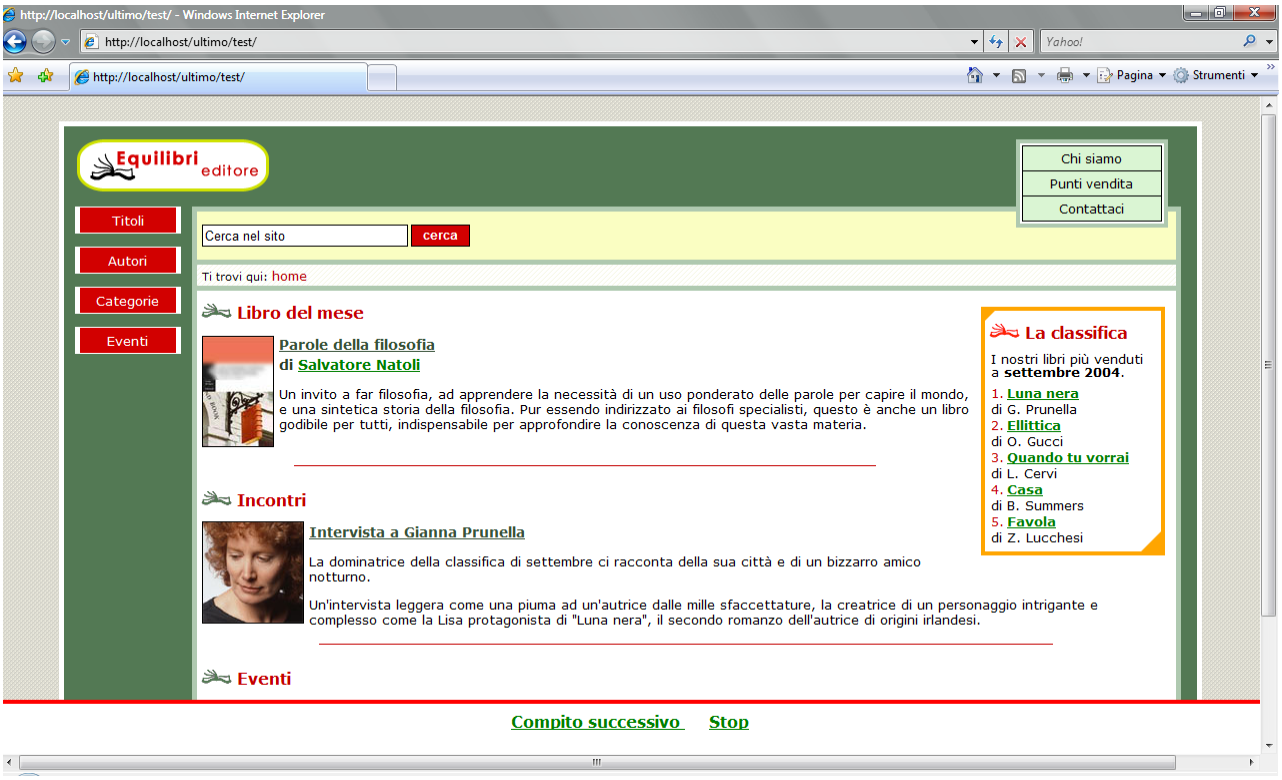

Figura 55 Sostituzione del frame inferiore con un div posizionato in maniera fissa in basso, mentre il frame originario viene reso invisibile

Facendo scrolling del documento o ridimensionando la finestra il div mantiene sempre la stessa posizione.

12.2.1 Prove sul browser NOKIA

Provando questa nuova soluzione sul browser NOKIA, non si ottiene però il risultato sperato:

- Il secondo frame non viene nascosto

[58] Sono state necessarie anche lievi modifiche al codice delle funzioni riguardanti i percorsi del file. Ci siamo accorti che quando richiamiamo da un frame A una funzione che si trova all'interno di un frame B, eventuali path relativi dei file sono determinati relativamente alla posizione della pagina caricata nel frame A. Se non avessimo effettuato tali modifiche i file non sarebbero stati trovati.

- il DIV non viene posizionato in modo fisso rispetto al viewport che coincide con lo schermo, ma rispetto all'intera pagina web.

Non abbiamo trovato nessun modo per nascondere il secondo frame, mentre per cercare di risolvere il primo problema si è provato a impostare le coordinate del posizionamento fisso con la proprietà top anziché con la proprietà bottom.
Se andiamo a visualizzare il risultato otteniamo il risultato di figura Figura 56 Tentativo di visualizzare un div con posizione fissa in basso dello schermo sul browser di Nokia, in una pagina web contenuta in un frame,ma facendo scrolling vediamo come anche in questo caso ci sia qualcosa che non va: il div non rimane fisso rispetto al viewport ma rispetto all'intera pagina web.

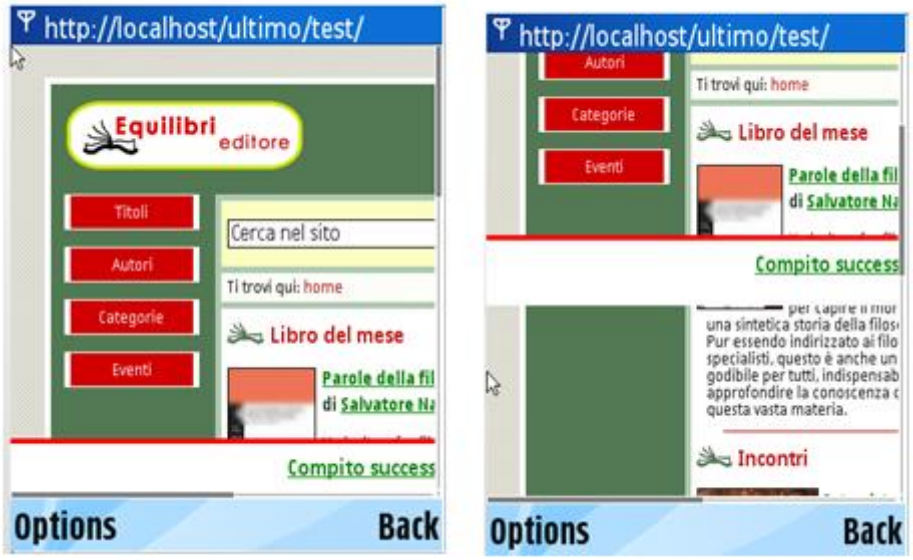

Figura 56 Tentativo di visualizzare un div con posizione fissa in basso dello schermo sul browser di Nokia, in una pagina web contenuta in un frame

A questo punto ci siamo domandati se il posizionamento fisso non fosse supportato in generale da questo browser o se questo comportamento si avesse solo nel caso di uso all'interno di frame.
Abbiamo provato, quindi, a visualizzare direttamente la pagina web (solitamente visualizzata all'interno del frame 1) e ci siamo accorti che in questo caso il posizionamento fisso funziona.

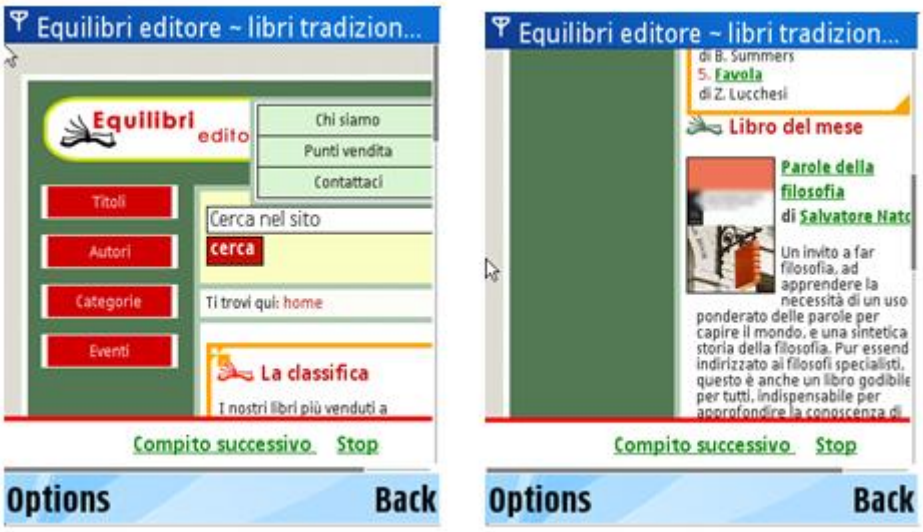

Figura 57 Tentativo di visualizzare un div con posizione fissa direttamente in una pagina web

12.2.2 Ipotesi di una soluzione alternativa priva di frame per Nokia N95

Abbiamo visto che:

- per mantenere i link fissi sullo schermo possiamo usare il posizionamento fisso usato dai CSS
- il posizionamento fisso usato all'interno dei frame non funziona sul browser NOKIA.
- il posizionamento fisso senza uso dei frame funziona sul browser NOKIA
- non si riesce a nascondere i frame sul browser NOKIA

Sembrerebbe quindi che la soluzione sia da ricercare nell'eliminazione totale dei frame.
Come già sappiamo Il frame è stato utilizzato per:

- mantenere fissi sullo schermo i link "Compito successivo" e "Stop"
- Garantire la persistenza delle variabili dello script che si occupa della memorizzazione delle informazioni relative agli eventi durante tutta la sessione del test.

Supponiamo quindi di eliminare i frame: il primo obiettivo potrebbe essere raggiunto senza problemi utilizzando il posizionamento fisso (funzionante in assenza di frame), mentre il secondo obiettivo – la persistenza dei dati – non sarebbe più raggiungibile in nessun modo[59].
Questo è vero se vogliamo una persistenza lato client.

Potremmo però pensare di inviare le informazioni relative agli eventi, mano a mano che questi accadono direttamente al server, ed utilizzare qui il meccanismo delle sessioni per garantire la persistenza dei dati.

12.2.3 Prove sul browser di iPhone

Effettuando alcune prove con iPhone ci siamo accorti che il posizionamento fisso non funziona mai: né nel caso di uso dei frame né nel caso di pagine web semplici.

Al momento non abbiamo trovato soluzioni al problema.

[59] Ci sarebbero i cookie ma presentano dei limiti nella dimensione e quindi sulla quantità di dati che è possibile memorizzare: per sessioni di test molto lunghe si potrebbero perdere delle informazioni.

13. CONFIGURABILITA' DEL LOGGING TOOL

Nel tool originario, per preparare un test era necessario:

1) Inserire le pagine del sito da testare nella cartella principale del tool (sul server)
2) Per ogni pagina inserire nell'head una riga di codice per l'inserimento del javascript
3) Nella cartella test/task aprire il file task.js e impostare la variabile NTask al numero dei task che saranno presenti nel test
4) Sempre nella cartella task, per ogni task ***i*** che desideriamo che sia presente nel test, creare un file nominato task***i***.htm contenente la descrizione del compito

Questo modo di procedere possiede alcuni **svantaggi** tra cui:

a) È necessario l'accesso al server per poter configurare il tool per un test
b) è un modo di procedere poco intuitivo per i non "addetti ai lavori"
c) può richiedere molto tempo soprattutto se si considerano le operazioni 2 e 4
d) non permette di avere più test pronti per la somministrazione, a meno di non creare nuove cartelle sul server ognuna contenente il tool configurato per ogni singolo test

Ci siamo posti quindi l'obiettivo di superare questi problemi. In particolare:
il problema a e b possono essere risolti sviluppando una **interfaccia di configurazione del test** che permetta la configurazione del test via http. In pratica si tratta di una serie di pagine html che guidano l'utente nel processo di configurazione di un nuovo test o nella visualizzazione, modifica o cancellazione di test esistenti. Esse permettono in modo abbastanza semplice la configurazione di un test anche ad utenti poco esperti, e permettono di eseguire da remoto operazioni come la 1 e la 4, che inizialmente richiedevano l'accesso diretto al server.

Inoltre **automatizzando alcune operazioni** (come l'inserimento del javascript all'interno di ogni pagina HTML del sito da testare, e la creazione automatica delle pagine HTML di descrizione dei task), è possibile ottenere un buon risparmio di tempo ovvero risolvere il problema c.

Infine, poiché vogliamo che il medesimo tool permetta di eseguire test diversi (o comunque tenga in memoria le impostazioni di altri test per esempio già eseguiti), è stato scelto di **memorizzare tutte le informazioni** raccolte in fase di configurazione del test tramite interfaccia **in un database.**

Diventa quindi possibile una volta avviato il tool scegliere tra una lista di test presenti quello che si intende eseguire.

13.1 Architettura dello strumento di configurazione

In generale, dal punto di vista architetturale, lo strumento di configurazione del logging tool si compone essenzialmente di due parti:

- Le pagine contententi le interfacce di configurazione
- Un database che raccoglie le informazioni relative al test

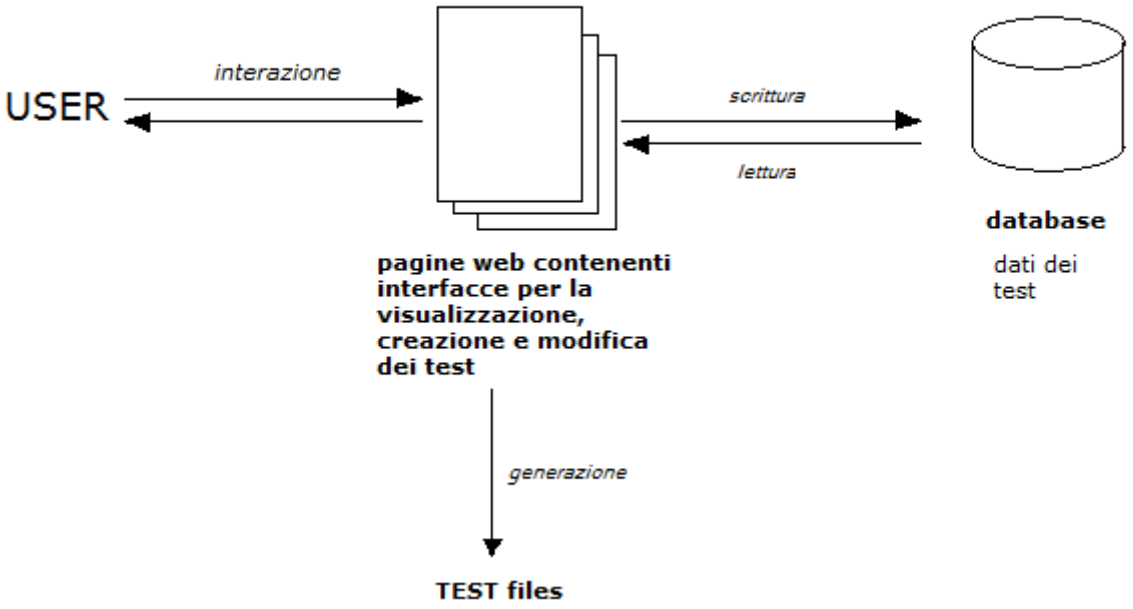

Figura 58 Funzionamento generale dello strumento di configurazione del logging tool

In generale, l'utente che decide di configurare, modificare, o cancellare un test, accede ad un'applicazione web presente nella cartella admin del logging tool. Tali pagine scambiano informazioni con un database che può contenere tutte le informazione sui test. Per esempio in fase di creazione o modifica di un test le informazioni inserite nell'interfaccia di configurazione vengono memorizzate sul database; mentre per la visualizzazione delle informazioni sui test già esistenti queste verranno prelevate dal database.

Alla fine del processo di creazione del test, verranno generati tutti i file necessari al funzionamento del test all'interno delle specifiche cartelle del logging tool:

- pagine web da testare (esse vengono inserite nella cartella siti/[idTest]/[idTask]
- pagine web contenenti la descrizione del compito (nella cartella test/task/[idTest]) come task*i*.htm

Analizziamo più in dettaglio la struttura del database e il modo in cui sono state implementate le interfacce di configurazione.

13.1.1 Il database

L'utilizzo del database consente di memorizzare tutte le informazioni dei test raccolte in fase di configurazione , e quindi di mantenerlo in memoria per un eventuale uso (anche in forma modificata) in futuro.

Per quanto riguarda il database è stato utilizzato MySQL. Il database progettato ha uno schema piuttosto semplice.

Esso è formato essenzialmente da tre tabelle:

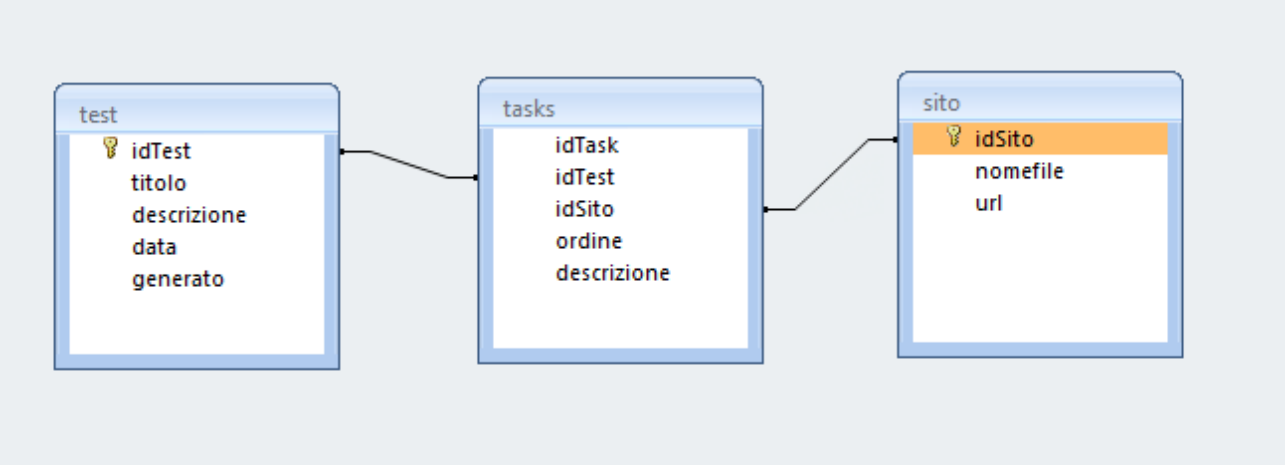

Figura 59 Schema del database

- La tabella TEST contenente le informazioni riguardanti il test quali il titolo, la descrizione, la data e un campo di controllo (generato) [60]
- La tabella TASKS contenente le informazioni relative ad un task quali il test a cui si riferisce, l'ordine rispetto agli altri task presenti in quel test, la descrizione, il sito su cui viene eseguito quel task
- Infine una tabella SITO contenente l'id del sito, il nome del file (.zip) contenente tutte le pagine del sito, e l'url del sito (importante perché a partire dalla cartella zip, verrà estratto il sito e messo in una cartella).

Volendo motivare le relazioni che intercorrono tra le varie tabelle potremmo dire:

- Un test è formato da più tasks (relazione 1-N)
- Un task si riferisce ad un test specifico. (relazione N-1)
- Un task viene eseguito su un solo sito (relazione 1-1)
- Un sito appartiene ad un solo task (relazione 1-1)

Quest'ultima affermazione si è scelta per semplicità. Ci ha evitato le problematiche connesse al riconoscimento e alla scelta da parte dell'utente tra i siti precedentemente inseriti. Ne consegue però che, se per esempio, abbiamo due task che devono essere svolti sullo stesso sito, occorre inserire la cartella .zip contenente il sito per due volte ed essi saranno considerati come file distinti all'interno del database (infatti hanno un id diverso) pur essendo uguali.

13.1.2 L'interfaccia di configurazione

L' interfaccia di configurazione deve permettere:

- La creazione di un nuovo test
- La visualizzazione/modifica/ cancellazione dei test già esistenti

[60] Esso serve per controllare che oltre al fatto che sul database sono presenti il dato del test, siano già stati generati anche i file utilizzati dal tool per lo svolgimento del test

Si è deciso quindi di creare una prima pagina index.php che consenta la scelta tra queste due categorie di operazioni:

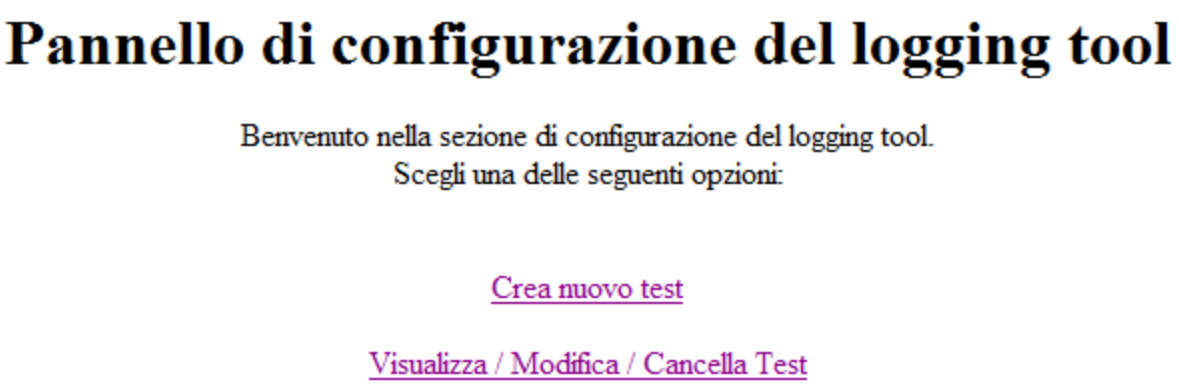

Figura 60 Schermata iniziale del tool di configurazione

13.1.2.1 Creazione di un nuovo test

Scegliendo "crea nuovo test", il sito ci guida attraverso le fasi di creazione del test:

- Inserimento titolo e descrizione del test (creanuova.php)

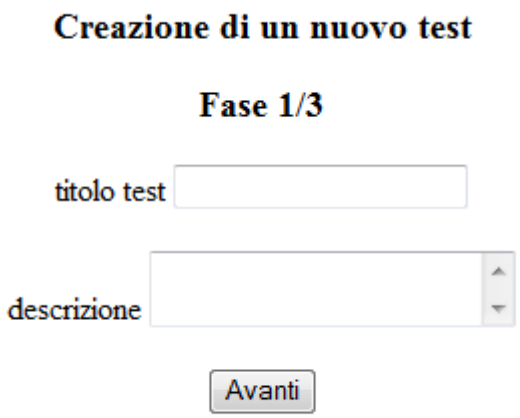

Figura 61 Interfaccia per la creazione di un nuovo test

- Inserimento dei vari task di cui è composto il test, e dei siti su cui ciascun task deve essere eseguito. (instasks.php)

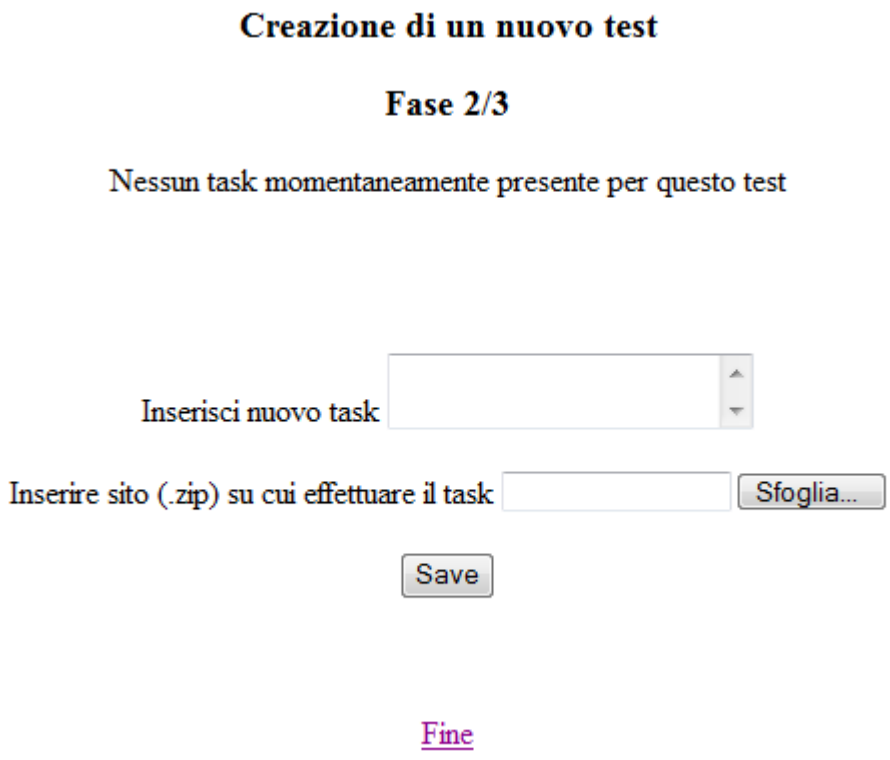

Figura 62 Interfaccia per l'inserimento dei task all'interno del test

La pagina instasks.php mostra in alto la lista dei task che compongono il test (se presenti) altrimenti mostra un avviso che avverte della assenza di task.
Per inserire un nuovo task l'utente può utilizzare l'interfaccia sottostante che prevede:

- La digitazione di un testo che descriva chiaramente il task
- Il sito nel quale l'utente del test dovrà effettuare il task.

Si è scelto di utilizzare il formato zip per il trasferimento dei file del sito sul server, in quanto tale formato ci permette di raggruppare in un unico file tutti i file che lo compongono.
Alla pressione del tasto Save la descrizione del task e il file vengono inviate al server[61], e la pagina instask.php ricaricata di nuovo per consentire l'inserimento di un nuovo task.
Una volta inseriti tutti i task desiderati l'utente può procedere con la pressione del link Fine.
Alla pressione del link "fine" verrà eseguito il codice presente nella pagina genera test.php

Il codice si occupa, per ogni task appartenente al test, di:

1) Decomprimere il sito associato a quel task nella cartella siti/[idTest]/[idTask]
2) Per ogni file estratto nella cartella siti/[idTest]/[idTask] inserire nell'head la riga <script type="text/javascript" src="/ultimo/siti/script.js"></script>
3) Generare nella cartella test/task/[idTest] un file di nome task[i].htm

Vediamo in dettaglio come abbiamo risolto a livello di codice ognuna di queste operazioni.

[61] Il file zip verrà salvato nella cartella siti/[idTest]/[idTask]. Il nome del file sarà il valore dell'identificatore univoco del task all'interno del database

Operazione 1: *Decomprimere il sito associato a quel task nella cartella siti/[idTest]/[idTask]*

Per trattare con i file zip occorrono librerie php che si occupino della gestione di questo tipo di file.

Da una ricerca sul web si è scoperto che a partire dalla realease 5.2 di PHP è stata introdotta come libreria di default un'estensione (prima disponibile tramite il repository PECL) per la gestione completa dei file Zip. L'estensione prende il nome di php_zip.dll[62]. Essa si caratterizza per le ottime performance dovute al fatto che è implementata nativamente ed è molto più completa rispetto a molte altre soluzioni che possiamo trovare sulla rete. L'estensione per la gestione dei file compressi espone un oggetto chiamato `ZipArchive` che rappresenta un archivio zip, ed una serie di funzioni che permettono di leggere e scrivere all'interno di archivi a chiunque preferisse un approccio procedurale. Per una corretta gestione degli archivi zip la libreria espone anche una serie di costanti che possono o devono essere utilizzate durante la chiamata a determinati metodi; queste costanti sono esaustivamente spiegate nella documentazione ufficiale. Proponiamo sotto la porzione di codice rigorosamente commentato che si occupa,per ogni task, di decomprimere il sito ad esso associato nella cartella siti/[idTest]/[idTask]

```
// Interrogo il database per ottenere la lista dei task e dei siti associati per il //test di
idTest =$idTest
$sql="SELECT tasks.idTask, tasks.ordine, tasks.idSito, tasks.descrizione, sito.url FROM
test JOIN tasks ON test.idTest=tasks.idTest JOIN sito ON tasks.idSito=Sito.idSito WHERE
test.idTest='$idTest' ORDER BY ordine";
$ris=mysql_query($sql,$conn) or die("Errore1" .mysql_error());
$cont=0;

//per ogni task
while($record=mysql_fetch_assoc($ris))
 {
        $cont++;
         //creo una nuova istanza della classe ZipArchive
         $archivia = new ZipArchive();

        //tento l'apertura del documento compresso tramite il metodo open
         if ($archivia->open( $record['url'])!==TRUE)
         {
         @exit("Impossibile aprire <$nome_file>\n");
         }
        /* invoco il metodo extractTo per estrazione file nel percorso indicato come
        parametro  */
         $archivia->extractTo('../siti/'.$idTest.'/'.$record['idTask']);
        // chiudo l'archivio
         $archivia->close();
```

[62] Sulle modalità d'uso di questa libreria si veda http://php.html.it/articoli/leggi/2036/le-funzioni-per-la-gestione-del-file-zip-con-php-52/ . Per la documentazione ufficiale si veda http://it2.php.net/manual/en/class.ziparchive.php

[…]

***Operazione 2**:* Per ogni file estratto nella cartella siti/[idTest]/[idTask] inserire nell'head il javascript

Per inserire il javascript all'interno dell'head di ciascuna pagina estratta, essendo i file XHTML a tutti gli effetti file XML, sono state ricercate librerie PHP capaci di trattare con questo tipo di file[63].

Le librerie che possiamo considerare standard che permettono l'accesso in lettura, modifica ed eventualmente interrogazione di file XML sono fondamentalmente DOM e SAX. PHP fornisce un'implementazione di entrambe le librerie, anche se a partire dalla versione 5 espone un sistema molto più semplice e diretto per accedere ai file XML: la libreria SimpleXML[64] abilitata di default all'interno del motore PHP.

La libreria SimpleXML presenta il grosso vantaggio di avere un'interfaccia a oggetti molto semplice e intuitiva da utilizzare. Rispetto al più prolisso DOM richiede meno righe di codice e, a differenza del SAX (che ci obbliga a tenere traccia manualmente delle gerarchie durante la gestione degli eventi generati), mantiene intatta la struttura del file XML anche nella sua rappresentazione interna. Infine, SIMPLEXML permette di eseguire query XPath sull'oggetto per recuperare nodi.

Proponiamo di seguito il codice PHP che si occupa, di esplorare la cartella siti/$idTest/$idTask e per ogni file HTML (solo quelli) che trova inserire all'interno del tag HEAD la riga per l'inserimento del Javascript.

```
//invocazione iniziale della funzione
$dir = '../siti/'.$idTest.'/'.$record['idTask'];
        addJsInDir($dir);


function addJsInDir($dir)
{
//ottengo un array di directory e file contenuti in $dir
 $dirfile=scandir($dir);
//scorro l'array
 for($i=0;$i<count($dirfile);$i++)
  {
  //controllo se il file è un file o una directory
//se è un file
   if(!is_dir($dir.'/'.$dirfile[$i]))
   {
      //controllo se è un file HTML
      $path_info = pathinfo($dir.'/'.$dirfile[$i]);
      $ext=(String)$path_info['extension'];

      if(strcmp($ext,'html')==0 ||strcmp($ext,'htm')==0) //se l'estensione è html o html
      {
          //carico il file xml
```

[63] Per una trattazione completa sulle librerie per la gestione di file XML tramite PHP si veda "Pro PHP XML and Web Services" di Robert Richard.

[64] Istruzioni per l'utilizzo della libreria SimpleXML sono disponibili in "Pro PHP XML and Web Services" di Robert Richard, ma anche sul web http://php.html.it/articoli/leggi/1763/simplexml-gestire-xml-in-php-5/. Per la documentazione ufficiale si veda http://us2.php.net/simplexml

```
        $xml = simplexml_load_file($dir.'/'.$dirfile[$i]);
        if($xml!=false)
        {
           //aggiungo la riga del js
           $nuovo=$xml->head->addChild("script","");
           $nuovo->addAttribute("type","text/javascript");
           $nuovo->addAttribute("src","/ultimo/siti/script.js");
           file_put_contents($dir.'/'.$dirfile[$i], $xml->asXML());}
        }
    }
   else //se il file è una directory (diversa da . e ..) richiamo la funzione su quella directory
        if($dirfile[$i] != '.' && $dirfile[$i] != '..')
                        addJsInDir($dir.'/'.$dirfile[$i]);
 }

}
```

Poiché una cartella può ricorsivamente contenere altre cartelle e così via, abbiamo deciso di definire una funzione, in modo da poterla richiamare in maniera ricorsiva fino all'esplorazione di tutto l'albero dei file contenuti nella directory di partenza.

Quindi, la funzione addJsInDir() si occupa di eseguire una scansione della cartella ottenuta come parametro tramite la funzione scandir() che restituisce un'array dei file e directory (di primo livello) contenute nella cartella. Per ciascun elemento dell'array controlla se si tratta di un file o una directory. Se si tratta di una directory viene richiamata la funzione addJsInDir su quella directory; se si tratta di un file, controlliamo che sia di formato HTML (estensione .html oppure .htm). Se il file è di formato HTML, utilizzando la funzione simplexml_load_file carichiamo all'interno di un oggetto SimpleXML il contenuto del file passato come argomento.

L'oggetto restituito rappresenta la root del nostro documento XML: esso espone una proprietà per ogni nodo figlio che può essere un array (nel caso ci siano più figli omonimi) o un elemento singolo e permette di accedere agli attributi utilizzando la sintassi che normalmente viene utilizzata per gli array.

Utilizzando i metodi (addChild() e addAttribute) forniti dalla libreria inseriamo all'interno del tag HEAD la riga <script type="text/javascript" src="/ultimo/siti/script.js"></script>.

Occupandoci in seguito dell'adattamento dell'interfaccia per iPhone (10.3) all'interno di ogni pagina del sito da testare sono state fatte inserire anche le seguemti righe:

```
<script type="text/javascript" src="/ultimo/siti/script.js"></script><script type="text/javascript">
 var meta=parent.document.getElementsByTagName('meta');
 meta[0].setAttribute('content','width=900');
 </script>
```

Per fare questo è bastato aggiungere nel codice:

```
$nuovo=$xml->head->addChild("script","
 var meta=parent.document.getElementsByTagName('meta');
 meta[0].setAttribute('content','width=900');");
        $nuovo->addAttribute("type","text/javascript");
        file_put_contents($dir.'/'.$dirfile[$i], $xml->asXML());
```

Operazione 3: Generare nella cartella test/task/[idTest] un file di nome task[i].htm

La creazione, per ogni task ***i*** che compone il test di un file nella cartella test/task/[idTest] un file di nome task ***i***.htm si ottiene semplicemente attraverso le funzioni offerte da PHp per la gestione di file di testo (fopen(), fwrite(), fclose()).

Proponiamo sotto il codice commentato:

```
$cont=0;
//per ogni test
while($record=mysql_fetch_assoc($ris))
    {
        //increment la variabile contatore
        $cont++;
        […]
        //creo la directory destinata a contenere i file dei task relative a questo test
        Mkdir("../test/task/".$idTest,0777);
        //apro (creo) un file in scrittura
        $scrivi_file=fopen("../test/task/".$idTest."/task".$cont.".htm","w");
        //inserisco il codice html in una variabile
        $pagina="<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'>
        <HTML>
        <HEAD>
        <link rel='stylesheet' href='task.css' type='text/css'>
        <title> Task$cont </title>
        <bgsound src='newtask.wav' loop='0'>
        </HEAD>
        <BODY>
        <div class='messaggio' align='center'>
        <h1> Compito numero $cont</h1>
        <p> $record[descrizione]</p>
        <br>
        <br>
        <p align='center'>
        <a href='../../../siti/$idTest/$record[idTask]/index.html'  target='site'
onClick='javascript:self.close()'  accesskey='a'> Vai avanti con il test </a>
        </p>
        </div>
        <!-- ... body of document ... -->
        </BODY>
        </HTML>";

// scrivo il cotnenuto della variabile nel file
        fwrite($scrivi_file,$pagina);
//chiudo il file
        fclose($scrivi_file);
```

13.1.2.2 Modifica e cancellazione di un test

Se si desidera cancellare o modificare un test, occorre fare click sul link "Visualizza, Modifica, Cancella Test" presente nella pagina iniziale. Verrà mostrata una tabella che elenca il nome dei test disponibili e fornisce due link per la modifica e la cancellazione.

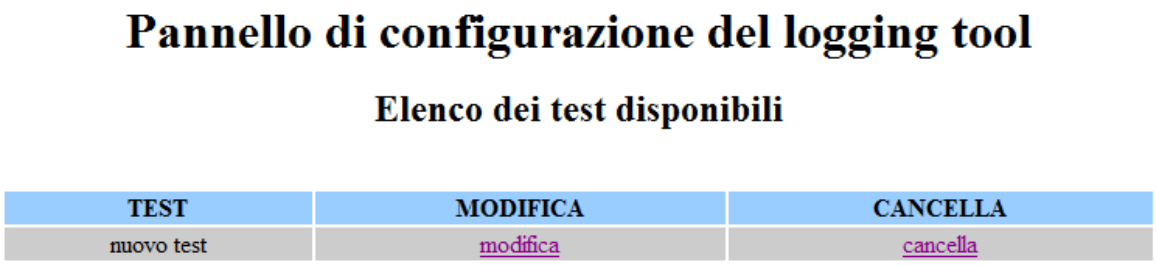

Figura 63 Interfaccia per la visualizzazione, modifica e cancellazione del test

Scegliendo *cancella* sarà mostrato un messaggio di conferma: premendo *ok* il test viene cancellato dal database. Scegliendo modifica sarà avviata la procedura per la modifica del nome, della descrizione, e dei task che compongono il test.

Per il modo in cui è stata strutturata, la procedura di modifica deve essere completata fino in fondo, se si vogliono evitare malfunzionamenti.

13.2 Modifiche della parte front-end del logging tool determinate dall'aggiunta dello strumento di configurazione

L'aggiunta dello strumento di configurazione ha determinato la necessità di apportare alcune modifiche alla parte front end del logging tool. Prima di tutto, poiché i test disponibili possono essere più d'uno è necessario modificare la pagina iniziale del test (go.php contenuta nel frame in alto) in modo da permettere la scelta del test che si vuole eseguire tra quelli disponibili.

Figura 64 Interfaccia iniziale del tool per la scelta del test da eseguire

Notare che una volta premuto il bottone *seleziona* l'id del test selezionato viene inserito nella query string.

Questo permette la visualizzazione della seguente schermata che riporta il numero di compiti che l'utente dovrà svolgere e fornisce una breve spiegazione all'utente riguardo il funzionamento del tool.

Figura 65 Interfaccia per l'inserimento del nome del l'utente che svolgerà il test

Alla fine viene richiesto all'utente l'inserimento del nome. La pressione del bottone "Inizia il test" determina l'invocazione della funzione getUsername() presente nel file go.php:

```
function getUsername()
{
//setto script al riferimento all'oggetto window del frame hidden
script= top.hidden;
//recupero il nome inserito dall'utente
 var name = document.forms[0].elements[0].value;

//invia il nome utente e il browser allo script presente nel lo script nel frame "hidden"
script.setUsername(name);
script.browser("*"+navigator.appName+" "+navigator.userAgent);

//recupero dalla querystring l'id del test che era stato richiesto
var querystring=location.search;
 var inizio = querystring.indexOf("t=");
 var fine=querystring.indexOf("&");
 idTest=querystring.substring(inizio+2,fine);

//invoco la funzione askfortask presente nello script (file task/task.js) inviandogli l'id del test
script.askfortask(idTest);
}
```

La funzione askfortask(), presente nel file task.js, riferito all'interno della pagina hidden.php, si occupa dell'impostazione della variabile Ntask al numero di task di cui è formato lo specifico test di id impostato come parametro.

```
function askfortask(t)
{
```

```
 idTest=t;
//assegna oggetto AJAX
ajaxobj=addXMLHttpRequest();
if(ajaxobj)
{
 var pagina="task/tasks.php?idTest="+idTest;
 ajaxobj.pen("get", pagina, true);
 ajaxobj.onreadystatechange = Response;
 //effettuo la richiesta
 ajaxobj.send(null);
}
else
  alert("IL BROWSER NON SUPPORTA AJAX");
}
```

Una volta memorizzato nella variabile globale (e quindi disponibile per tutta la sessione del test) il valore dell'idTest, esegue una richiesta ajax alla pagina task/tasks.php?idTest=… .

La pagina specificata contiene un semplice script php che richiede al database il numero di task che compongono il test, quindi lo stampano sulla pagina.

```
<?php
  include('connessione.php');
  $conn=Connettidb();
  $idTest=$_GET['idTest'];
  $sql="SELECT COUNT(*) as numero  FROM test JOIN tasks ON test.idTest=tasks.idTest WHERE
test.idTest='$idTest' ORDER BY ordine";
  $ris=mysql_query($sql,$conn) or die("Errore1" .mysql_error());
 $record=mysql_fetch_assoc($ris);
  echo "$record[numero]";
?>
```

A questo punto, la pagina contenente il numero di task, viene ricevuto dal client; la funzione di call-back Response() (anch'essa presente nel file task.js), si occupa di impostare la variabile globale NTask al numero dei task ricevuti in risposta:

```
function Response()
{
if(ajaxobj.readyState == 4)
  {
  if(ajaxobj.status == 200)
            NTask=ajaxobj.responseText;
  }
}
```

A questo punto tutte le variabili che ci servono (idTest, NTask) sono settate e disponibili per tutta la sessione di esecuzione del test.

Quando l'utente preme il pulsante Compito successivo nel frame "hidden" verrà richiamata la funzione nextTask() presente nel file task.js, la quale, incrementata la variabile t che tiene il conto del numero di task svolti, si occupa di visualizzare all'interno del frame *site* la pagina contenente la descrizione del compito:

```
function nextTask ()
{
 t += 1;
 var doc;
 for(var i = 0; i < parent.frames.length; i++)
   if (top.frames[i].name == "site")
    {
     doc = top.frames[i];
    }
         if (t <= NTask)
                  doc.location="task/"+idTest+"/task"+t+".htm";
          else
                  doc.location = ("task/endTask.htm");
}
```

13.3 Limiti attuali di questo strumento

Questo strumento di configurazione attualmente presenta dei limiti:

- Funziona solo con siti statici
- I file contenuti nel sito devono essere file XHTML ben formati.
  Infatti, a causa dell'utilizzo della libreria Simple XML se il file analizzato non è un file XML ben formato non viene aggiunta la riga del javascript e si verifica uno warning.
- Non permette il trasferimento di file superiori a un certo numero di MB
- Non consente la configurazione manuale (attraverso l'accesso diretto al server). Essa consentirebbe anche di ovviare alla problematica di cui sopra consentendo di trasferire file di qualsiasi dimensione all'interno del server.

Alcuni di questi limiti potranno essere superati in eventuali versioni future.
Altri aspetti che possono essere migliorati riguardano:

- l'interfaccia grafica dello strumento di configurazione
- rendere possibile, in fase di configurazione di un determinato test, selezionare gli eventi che vogliamo che siano registrati e quelli che invece vogliamo siano ignorati.

14. CONCLUSIONI E SVILUPPI FUTURI DEL TOOL DI LOGGING

Nel corso del nostro lavoro è emersa l'importanza di avere un tool automatico per la memorizzazione delle azioni svolte dall'utente nel corso dei test di usabilità.

Analizzando un vecchio tool sviluppato anni fa al laboratorio HIIS all'ISTI-CNR sono state individuate varie problematiche che non lo rendono utilizzabile con i dispositivi mobili:

- Incompatibilità dello script di rilevamento con i browser moderni
- Incompatibilità dell' applet java con i browser mobili
- Mancato rilevamento di alcuni gesti specifici di iPhone
- problemi di visualizzazione dell'interfaccia
- problemi legati all'uso dei frame

Riassumiamo brevemente l'analisi che abbiamo fatto di ogni problematica, le soluzioni proposte e i risultati ottenuti.

Incompatibilità dello script di rilevamento con i browser moderni

Il rilevamento degli eventi generati dall'interazione dell'utente con il sito, viene effettuato tramite uno script inserito in ogni pagina del sito da testare.
Lo script di rilevamento originario era stato progettato per essere eseguito all'interno dei browser della quarta generazione, IE 4 e Netscape 4.
I browser attuali hanno un modello di eventi diverso, il cosiddetto DOM Level 2 event model, a parte Explorer che non si è adeguato agli standard.
Si è quindi resa necessaria la modifica dello script in modo da consentire il rilevamento degli eventi su tutti i browser moderni sia desktop che mobili e in modo che le informazioni vengano inviate ad uno script contenuto nel frame *hidden* anziché all'applet, che abbiamo dovuto togliere per problemi di compatibilità con i browser mobili.

Incompatibilità dell' applet Java con i browser mobili

Un componente essenziale nell'architettura originaria del tool è un applet Java. Essa svolge le seguenti operazioni:

- ricevere le informative relative alle azioni degli utenti man mano che queste si verificano e concatenarle con quelle precedenti,
- inviare i dati al server alla conclusione del test

Poiché l'applet non è compatibile con i browser mobili, si è resa necessaria la ricerca di alternative.
La prima operazione è svolta da un javascript, mentre l'invio dei dati al server è ottenuto tramite AJAX.
Poiché il browser Nokia non supporta le richieste al server con metodo POST (o meglio non supporta il settaggio degli headers) è stato costruito uno script che esegue l'invio tramite tante richieste GET successive. Tale soluzione funziona su tutti i browser, mentre sul browser Nokia si verifica un comportamento anomalo.
È stato poi costruito uno script PHP che si occupa della ricezione lato server dei vari pacchetti di informazione e solo quando li ha ricevuti tutti le memorizza su un file di log.

Mancato rilevamento di alcuni gesti specifici di iPhone

Una grande novità introdotta da iPhone è l'interazione mediante tocchi sullo schermo. L'utente interagisce compiendo gesti con le dita sullo schermo. Durante l'interazione, ad alcuni di questi gesti vengono associati i tradizionali eventi javascript (per esempio al TAP singolo viene associato l'evento onClick); altri gesti come il pinch-in e pinch-out, oppure il double tap non hanno nessun evento associato.
Si è posto quindi il problema di come fare per rilevare quando l'utente compie tali gesti durante il test.
Dalla lettura di documentazione e manuali riguardanti iPhone abbiamo appreso che esso, oltre a sopportare il LEVEL 2 Event model, possiede un modello di eventi specifico, pensato appositamente per i dispositivi touchscreen.
Sfruttando quindi gli eventi specifici di iPhone siamo riusciti a riconoscere anche questi gesti singolarmente. Resta ancora da capire come integrare il codice per il riconoscimento dei vari gesti in modo che esso possa avvenire contemporaneamente.

Riadattamento dell'interfaccia per dispositivo mobile

A causa del diverso comportamento del browser di Nokia e di iPhone nella visualizzazione delle pagine web, il testo che guida l'utente durante il test appariva troppo piccolo e quindi illeggibile. Anche gli altri elementi come i campi di testo da compilare erano difficilmente utilizzabili dall'utente, se non dopo frequenti operazioni di zoom in e zoom-out che sarebbero da evitare il più possibile in quanto appesantiscono l'interazione. Fortunatamente iPhone presenta un metodo abbastanza semplice per ottimizzare la visualizzazione dei siti web: il metatag viewport.

Problematica uso di frame

Il tool utilizza i frame per due motivi:

- consente di mantenere fissi sullo schermo i link "Compito successivo" e "Stop",
- garantisce la persistenza delle variabili dello script che si occupa della memorizzazione delle informazioni relative agli eventi durante tutta la sessione del test.

Entrambi gli obiettivi sono pienamente raggiunti sui browser desktop. Si rilevano invece problematiche per quanto riguarda i browser mobili di iPhone e di Nokia.
Se la persistenza delle variabili dello script è pienamente raggiunta, non si riesce invece ad ottenere che i link rimangano fissi e quindi sempre disponibili sullo schermo.
Questo è dovuto ad un diverso modo di gestire i frame da parte di questi browser: a differenza dei browser desktop, dove i frame mantengono la loro dimensione a prescindere dalla pagina che viene in essi caricata, sui browser mobili i frame si dimensionano in base al contenuto.

Si è quindi fatto il seguente tentativo: nascondere il secondo frame e inserire al suo posto un div posizionato in modo fixed nella parte bassa del viewport.
Ci siamo accorti che sul browser Nokia il posizionamento fisso non funziona come vorremmo se utilizzato all'interno di un frame, mentre in iPhone questo stile non è proprio supportato. Al momento non si è riuscito a risolvere il problema[65].

Oltre alla risoluzione delle problematiche si è ritenuto utile dotare il tool di uno strumento di configurazione che permettesse di creare nuovi test e visualizzare o modificare test precedentemente

[65] Abbiamo solo ipotizzato una possibile soluzione che funziona per Nokia, la quale prevede l'eliminazione dei frame e l'utilizzo delle sessioni lato server per la persistenza dei dati.

inseriti. Esso consente inoltre di automatizzare alcune operazione consentendo un notevole risparmio di tempo nella preparazione del test.

14.1 La nuova versione del tool

A seguito di tutte queste modifiche è stato ottenuto un nuovo tool. Proviamo ad analizzarlo brevemente.

La figura mostra l'architettura del vecchio e del nuovo tool a confronto.

Lato client, gli eventi sono sempre rilevati tramite uno script inserito in ogni pagina del sito da testare. Tale script non è però lo stesso: esso è stato modificato in modo da funzionare su tutti i browser moderni. L'applet è stata sostituita da un javascript nella funzione di ricezione, memorizzazione dei dati e il concatenamento con quelli precedentemente ricevuti, mentre per l'invio dei dati al server è stato utilizzato un ajaxscript che esegue una serie di richieste GET successive.

Lato server, la servlet è stata sostituita da uno script php che si occupa della ricezione dei vari pacchetti ricevuti dallo script ajax, e solo quando li ha ricevuti tutti memorizza tutte le informazioni su un file (precedute da un'opportuna intestazione).

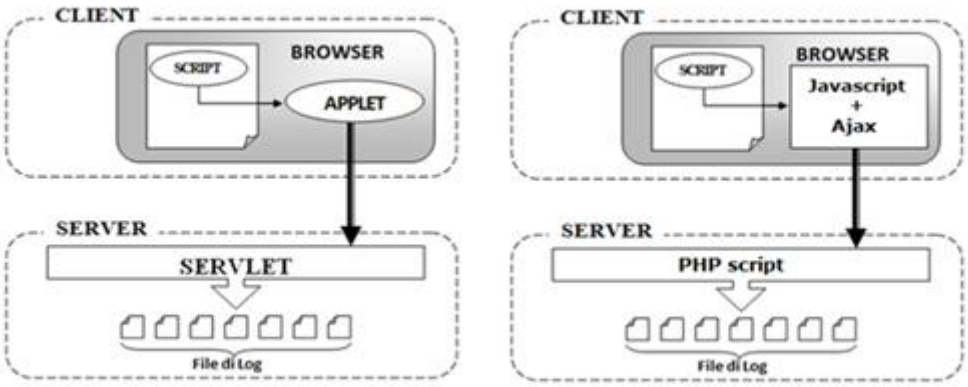

Figura 66 Architettura del vecchio e del nuovo tool a confronto

14.2 Tecniche implementative utilizzate

Per lo sviluppo di questa applicazione web sono state utilizzate le seguenti tecnologie o linguaggi:

- JAVASCRIPT per lo scripting lato client
- Ajax per lo scambio di alcune informazioni con il server
- PHP + diverse librerie
- SQL come linguaggio di interrogazione
- Server Apache come server web
- Server MySQL come server per il database

14.3 Organizzazione della cartella del tool

I vari file che compongono il tool si trovano nella cartella nominata *ultimo.*
I vari file una struttura gerarchica che descriviamo.
La cartella *admin* contiene tutto ciò che riguarda lo strumento per la configurazione del tool:

- La cartella *sites*: contiene i vari file zip caricati da interfaccia e contenente le pagine del sito che si vuole testare. Per evitare conflitti di nomi essi sono stati rinominati con l'id unico del task cui sono associati
- La cartella *utility*: contiene codice utile come per esempio quello contenuto in connessione.php per la connessione al database
- Vari files php che implementano il software di gestione del tool

La cartella *siti* contiene:

- Una cartella per ogni test che è stato generato: alla cartella viene dato come nome l'id univoco del test. All'interno di ciascuna cartella sono presenti altre cartelle, una per ogni task che compongono il test. Tali cartelle contengono i siti su cui si vuole eseguire il task. A ciascuna pagina del sito è stato aggiunto automaticamente in fase di generazione del test il riferimento allo script per il rilevamento degli eventi
- Il file *script.js:* contiene il codice javascript per il rilevamento degli eventi

Infine la cartella *test* contiene:

- I file html e php che implementano il front end del tool di logging
- Una cartella *task* contenente tutto ciò che riguarda i task che dovranno essere eseguiti nel corso del test:
    - Una cartella per ogni test (il nome assegnato è l'id univoco del test cui si riferisce): ogni cartella contiene un file html per ciascuno dei task che compongono il test
    - File connessione.php: per la connessione al database
    - endTask.html: pagina mostrata alla fine del test
    - task.js: codice javascript che si occupa di richiedere al server il numero di tasks di cui è composto il test per inizializzare la variabile NTask; contiene inoltre la funzione per l'avanzamento (da un compito al successivo) all'interno del test
- una cartella logs: contiene una cartella per ogni test configurato. Ciascuna di queste ultime cartelle contengono i file di logs generati dallo svolgimento dello specifico test.

14.4 Requisiti tecnici

14.4.1 Requisiti per l'installazione dell'applicazione lato server

I requisiti necessari per l'installazione dell'applicazione sono:

- Server APACHE
- Server MySQL
- PHP 5.2+ (abilitare estensione php_zip.dll)

14.2.1 Requisiti per l'utilizzo dell'applicazione lato client

Per utilizzare il tool è sufficiente un browser:

- il browser deve abilitare il supporto javascript
- il browser può essere IE 4+ o un qualsiasi browser (anche mobili) che supporti il DOM Level 2 Event Model (Mozilla Firefox, Safari, Opera, ecc.).
- Il browser deve supportare AJAX (GET method)

14.3 Guida all'installazione

Per l'installazione del tool sono necessari i seguenti passi:

- Copiare la cartella del tool nella cartella principale del server
- creare il database nel server MySQL (per questa operazione è possibile utilizzare il file DUMP presente nella cartella principale del software
- all'interno della cartella admin/utility e all'interno della cartella test/task/ modificare il file connessione.php con i parametri per la connessione al database

A questo punto, tramite il browser si acceda alla cartella admin per creare il test.

Per l'esecuzione del test è necessario invece accedere al server alla cartella test presente nella cartella del tool.

14.4 Sviluppi futuri

Oltre alla risoluzione delle problematiche che sono rimaste aperte altri possibili sviluppi futuri potranno riguardare:

- aumento delle possibilità di configurazione (per esempio la possibilità di selezionare quali tipi di eventi e su quali elementi devono essere rilevati, il tipo di informazioni da memorizzare per ciascun tipo di evento, formato di memorizzazione dei file di log)
- Miglioramenti dell'interfaccia grafica del tool
- Provare a ricostruire una simulazione o video dell'interazione dell'utente con il sito a partire dal file di log
- Provare a salvare i file di log in XML e trovare un modo per correlare ciascun evento generato con gli elementi della pagina, in modo da migliorare il processo di analisi dei log.
- Miglioramento dell'aspetto estetico del tool
- Miglioramento dell'usabilità
- Miglioramento dell'accessibilità per consentire lo svolgimento di test anche da parte di utenti disabili.

Bibliografia

Monografie e manuali:

Wagner, R., *Professional iPhone and iPod Touch Programming, Building Applications for Mobile Safari,* John Wiley & Sons, Inc., 2008

Allen, C.,Appelcline S., *iPhone in Action. Introduction to Web and SDK Development,* Manning, 2009

Flanagan, D., *JavaScript - The Definitive Guide, 5th Edition* , O'reilly, Aug. 2006

Flanagan, D., *JavaScript - The Definitive Guide, 3th Edition* , O'reilly, Aug. 1998

Kurose, J.F., Ross, K.W., *"Internet e reti di calcolatori",* McGraw-Hill seconda edizione, 2003

Ricci, F., *Sviluppare il Web Mobile*, Apogeo, 2009

Richard, R., *Pro PHP XML and Web Services,* Apress, 2006

Articoli scientifici:

Kaikkonen, A., "*Past, Present usage and the future",* Nokia Corporation, Finland

Paternò, F., "*Interfacce utente multi-dispositivo"* , in Alessandro Soro(ed.), Human Computer Interaction, 11-15

Roto, V., *Search on Mobile Phones,* Nokia research center, 2006

Arase, Y., Tajahiro, H., Uemukai, T., *OPA Browser: A Web Browser for Cellular Phones Users,* Proceedings of the 20th annual ACM symposium on User interface software and technology, p.71-80, 2007

Tajima, K., Ohnishi K, *Browsing Large HTML Tables on Small Screen,* Proceedings of the 21st annual ACM symposium on User interface software and technology, p. 259-268, 2008

Watters, C., Zhang, R., Duffy, J., *Comparing Table Views for Small Devices,* Proceedings of the 2005 ACM symposium on Applied computing, p. 975-980, 2005

Maekawa, T., Hara, T., Nishio, S., *Image Classification for Mobile Web browsing,* Proceedings of the 15th international conference on World Wide Web, p. 43-52, 2006

Liu, H., Xie, X., Ma, W., Hong-Jiang, *Automatic Browsing of Large Picture on Mobile Device,* Proceedings of the eleventh ACM international conference on Multimedia, p 148-155, 2003

Roto, V., popescu, A., Koivisto, A.J., Vartieinen, E., *Minimap – A Web Page Visualization Method for Mobile Phones,* Proceedings of the SIGCHI conference on Human Factors in computing systems , 2006

Grassel, G., Geisler, R., Vartiainen, E., Chauhan, D., Popescu, A., *The Nokia Open Source Browser*, 2003

Kaikkonen, A., Roto, V., *Perception of Narrow Web Pages on a Mobile Phone*, 2003

Ahmadi, H., Kong, J., *Efficient Web browsing on Small Screens*, Proceedings of the working conference on Advanced visual interfaces, 2008

Kaikkonen, A., Roto, V., *Navigating in a Mobile XHTML Application*, Proceedings of the SIGCHI conference on Human factors in computing systems, 2003

Baudisch, P., Xie, X., Wang, C., Ma, W., *Collapse to Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content*, Proceedings of the 17th annual ACM symposium on User interface software and technology, p. 91-94, 2004

Nichols, Hua, Barton, *Highlight: A System for Creating and Deploying Mobile Web Applications*, Proceedings of the 21st annual ACM symposium on User interface software and technology, p 249-258, 2008

Paternò, Santoro, Scorcia, *Automatically Adapting Web Sites for Mobile Access through Logical Descriptions and Dynamic Analysis of Interaztion Resources*,

Paternò, *Technological platforms, convergence and adaptation of interactive contents*

Yang, C.C., Wang, F. U., *Fractal Summarization for Mobile Devices to Access Large Documents on the Web*, Proceedings of the 12th international conference on World Wide Web, p. 215-224, 2003

Gupta, S., Kaiser, G., Neistadt, D., Grimm, P., *DOM-based Content Extraction of HTML Documents*, Proceedings of the 12th international conference on World Wide Web, p. 207-214, 2003

Ka Kit Hoi, Dik Luk Lee, Jianliand Xu, *Document Visualization on Small Displays*, Proceedings of the 4th International Conference on Mobile Data Management, p. 262-278, 2003

Efficient Web Browsing on Handhelf Devices Using Page and Form Summarization, ACM Transactions on Information Systems (TOIS), p. 82-115, 2002

Lam, H., Baudisch, P., *Summary Thumbnails: Readable Overviews for Small Screen Web Browsers,* Proceedings of the SIGCHI conference on Human factors in computing systems, p. 681-690, 2005

Buyukkokten, O., Garcia-Molina, H., Paepcke, A., Winograd, T., *Power Browser: Efficient Web browsing for PDAs,* Proceedings of the SIGCHI conference on Human factors in computing systems, p. 430-437, 2000

MacKay, B., *The Gateway: A Navigation Technique for migrating to Small Screen,* CHI '03 extended abstracts on Human factors in computing systems, p.684-685, 2003

Lieberman, H., *A Multi-Scale, Multi-Layer, Translucent Virtual Space,* Proceedings of the IEEE Conference on Information Visualisation, p. 126, 1997

Wobbrock, J.O., Forlizzi, J., Hudson, S.E., Myers, B.A, *WebThumb: Interazrion Techniques for Small-screen browser,* Proceedings of the 15th annual ACM symposium on User interface software and technology, p 205-208, 2002

Tesi di laurea

Roto, V.: *Web Browsing on Mobile Phones - Characteristics of User Experience*, Doctoral dissertation, Helsinki University of Technology. (2006)

Documentazione ufficiale Apple

iPhone Human Interface Guidelines for Web Applications

Safari Web Content Guide for iPhone 0S

Safari HTML Reference

Safari CSS Reference

WebKit DOM Programming Topics

Safari DOM Extensions Reference

Apple Javascript Coding Guidelines

iPhone Application Programming Guide (capitoli: The Core Application- Event handling)

UIEvent Class Reference

UITouch Class Reference

Documentazione Forum.Nokia.com

Getting Started with AJAX on the Nokia Web browser, version 2.0, 2007

Mobile Web Technologies Overview Version 1.0, 2008

Nokia Web Browser Design Guide version 1.0, 2007

S60 3rd Edition Feature Pack 1: Web browser Feature Summary version 1.0, 2006

S60 3rd Edition Feature Pack 1: Web Browser Product Description Version 1, 2006