Resoconto Stage
Versione: 1.0
Mail: mattia.cerantola@gmail.com

Mattia Cerantola

MindSlide Mobile

Resoconto Stage

Sommario

Il presente documento riassume le attività svolte durante la realizzazione del progetto di stage *MindSlide Mobile*, ponendo particolare attenzione sugli standard studiati e sul comportamento dei dispositivi `mobile`.

Informazioni documento

Nome	resoconto_stage_1_0.pdf
Versione	1.0
Data creazione	30/06/2011
Data ultima modifica	06/07/2011
Stato del Documento	Formale ad uso esterno
Distribuzione	Mattia Cerantola Dott. Stefano Boldrin Dott.sa Ombretta Gaggi Dott. Gregorio Piccoli

Registro delle modifiche:

Versione	Data	Modifiche effettuate
1.0	06/07/2011	correzione ortografico/sintattica dell'intero documento.
0.5	06/07/2011	inserita la sezione *Device e loro comportamenti (3.2)*.
0.4	05/07/2011	inserita la sezione *Touch e multitouch events (3.1)*.
0.3	04/07/2011	correzione ortografico/sintattica delle parti redatte; inserite tabelle e figure; aggiunta la sezione *Funzionalità sviluppate e confronto con i requisiti (2.2)*.
0.2	01/07/2011	inserito il capitolo *Conclusioni (4)*.
0.1	30/06/2011	prima stesura del documento: inseriti i capitoli da *Introduzione (1)* a *Applicazioni (2.1)* incluso.

Tabella 1: registro delle modifiche

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è quello di riassumere le attività svolte durante la realizzazione del progetto di stage *MindSlide Mobile*. Gli argomenti trattati in questo documento sono i seguenti:

1. *Implementazioni e modifiche apportate al software di partenza (2)*;
2. *Standard esaminati (3)*;
3. *Conclusioni (4)*.

1.2 Strumenti e risorse

Gli strumenti e le risorse impiegate nella realizzazione del software `MindSlide Mobile` sono:

1.2.1 Hardware

- Personal Computer con S.O. *Microsoft Windows 7*;
- *Acer Iconia A500*;
- *Apple iPad 1.0*;
- *Apple iPhone 3.0*.

1.2.2 Software

Linguaggi

- `HTML5`;
- `CSS`;
- `JavaScript`.

Editor

- *Eclipse 3.5.2* (per la realizzazione della documentazione in LaTeX): `http://www.eclipse.org`
- *Notepad++ 5.9* (per la realizzazione del codice `javascript`, `HTML5` e `CSS`): `http://notepad-plus-plus.org`.

1.3 Glossario

Al fine di eliminare ambiguità e incomprensioni, tutti i termini tecnici e le sigle utilizzate nel presente documento sono riportati nel documento *Glossario*, fornito come allegato.

2 Implementazioni e modifiche apportate al software di partenza

In questo capitolo verranno analizzate e riepilogate le funzionalità prodotte per i `device mobile` rispetto al prodotto di partenza destinato ai dispositivi fissi, arrivando così al prodotto finale *MindSlide Mobile*.

Nota: per una visione completa dei requisiti, della progettazione effettuata e dei test di verifica si rimanda ai documenti *analisi_requisiti_1_0.pdf, specifica_tecnica_1_0.pdf, resoconto_test_1_0.pdf*.

2.1 Applicazioni

Le scorciatoie sul desktop permettono di creare una simil-applicazione di *MindSlide Mobile* senza dover usare nessun linguaggio specifico per il `device` in uso: la possibilità di usare la pagina web anche in modalità offline rende l'utilizzo di questo software estremamente agevole e portabile. Vediamo ora di seguito come si presenta la scorciatoia (evidenziata con un cerchio rosso) sui vari dispositivi.

Acer Iconia

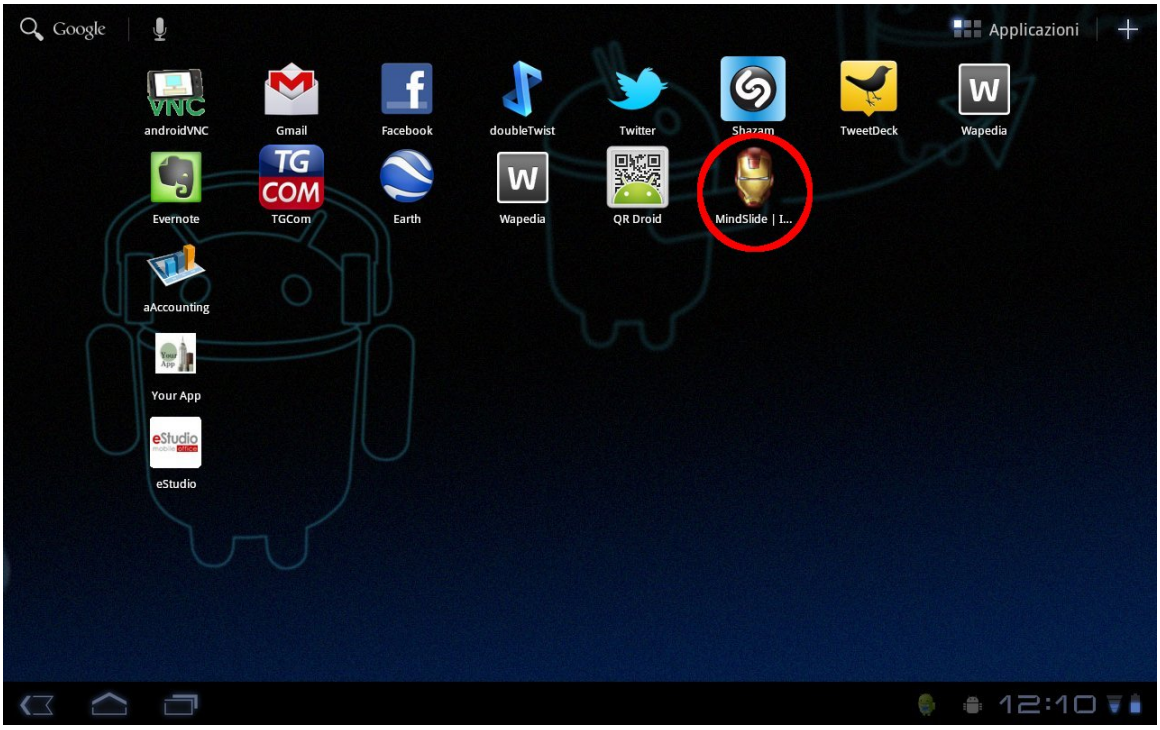

Figura 1: Iconia - App

Apple iPad

Figura 2: iPad - App

Apple iPhone

Figura 3: iPhone - App

Esattamente come per la sua versione base che ne permetteva l'utilizzo su PC fisso, l'unico requisito per usare `MindSlide Mobile` in modalità offline è quello di visitare almeno una volta in modalità online il sito dell'applicazione

`http://padova.zucchetti.it/MSM/`

così da permettere il download dei componenti necessari nel `local storage` del `browser`.

2.2 Funzionalità sviluppate e confronto con i requisiti

Con il supporto del documento *analisi_requisiti_1_0.pdf* ed in particolare facendo riferimento ai requisiti stabiliti all'inizio del progetto di stage, passiamo ora all'analisi delle fuzionalità sviluppate.

RF_1: Il software è correttamente caricato ed eseguito su dispositivi `mobile`;

RF_1.1: Il software fornisce funzionalità diverse a seconda delle caratteristiche del dispositivo sui cui viene eseguito;

RF_1.1.1: Il software distingue su quale dispositivo è in esecuzione e quali sono le periferiche input disponibili; tuttavia potrebbe non funzionare su ogni dispositivo `mobile` (si veda *Device e loro comportamenti (3.2)*);

RF_1.1.2: Il software consente l'utilizzo del `touch` per eseguire qualsiasi operazione in tutte e 3 le interfacce, esattamente come accade con l'uso del mouse su PC; tutte le funzionalità legate ad altri eventi (`multitouch, gesture, accelerometro, etc.`) sono comunque eseguibili anche tramite `touch`;

RF_1.1.3: Il software consente lo spostamento della `MindMap` in modalità `MindMap View` tramite il `drag&drop` via `touch` (`pan`);

RF_1.1.4: Il dispositivo consente, se dotato di `multitouch`, il ridimensionamento della `MindMap` in modalità `MindMap View` tramite la `gesture pinch` e `reverse-pinch`.

RF_1.1.5 (modificato come da *resoconto_settimanale*, sezione 2.4.2): In modalità `Presentation View` è possibile passare da una `slide` all'altra (successiva, precedente, padre, prima o ultima che sia) utilizzando il trascinamento di una o più dita sullo schermo. E' comunque sempre possibile raggiungere le `slide` toccando le relative icone del menù `slide` collocate nell'angolo superiore destro dello schermo;

RF_2: Il software è dotato di fogli di stile che si adattano al dispositivo su cui viene eseguito;

RF_2.1: I fogli di stile adattano correttamente gli elementi a schermo in base all'orientamento del dispositivo (`landscape` e `portrait`);

RV_1: Il software aderisce alle principali linee guida dei progetti OpenSource;

RV_2: Il software ha una licenza di tipo opensource;

RQ_1: Il progetto è accompagnato da opportuna documentazione;

RA_1: Il software utilizza la tecnologia `HTML5`;

RA_2: Il software è compatibile con più dispositivi possibili (si veda *Device e loro comportamenti (3.2)*);

RA_2.1: Il software NON utilizza linguaggi, librerie o tool mirati ad uno o più specifici dispositivi hardware.

Tutte i requisiti e le funzionalità collegate sono stati testati con successo; per maggiori informazioni si veda il documento *resoconto_test_1_0.pdf*.

2.3 Interfacce grafiche

Le interfacce grafiche implementate sono largamente descritte e analizzate nel documento *resoconto_test_1_0.pdf*, fornito come allegato.

3 Standard esaminati

Gli standard relativi alle tecnologie `touch` ed ai `device` portatili (quali `tablet` e `smartphone`) sono stati largamente analizzati durante questa esperienza di stage e sono risultati essere uno degli argomenti più importanti affrontati nella realizzazione del progetto.

Purtroppo essi non sono ancora stati regolamentati e definiti al 100%, soprattutto a causa della comparsa e diffusione solo recente dei `tablet` e degli `smartphone` sul mercato mondiale; andremo di seguito ad analizzare le norme riguardanti gli standard ed i dispositivi studiati.

3.1 Touch e multitouch events

Esattamente come per mouse e tastiera, `javascript` mette a disposizione diverse funzioni che consentono di intercettare gli input di tipo `touch`.

All'interno di una pagina web (come nel caso di *MindSlide Mobile*) `javascript` traccia il numero di dita presenti a schermo, salvando i dati in un array. Andiamo ora ad analizzare le funzioni relative agli eventi `touch`:

- **touchstart:** si attiva quando un dito viene appoggiato sulla superficie dello schermo. Esso tiene conto di quante dita sono presenti sullo schermo ed incrementa il numero di elementi presenti nell'array touch ogni volta che viene richiamato, modificandone anche la sua lunghezza (length);
- **touchmove:** si attiva quando un dito già appoggiato sulla superficie dello schermo si sposta. Esso traccia i movimenti delle singole dita sullo schermo, fornendo le coordinate X,Y della posizione attuale delle singole dita;
- **touchend:** si attiva quando un dito viene rimosso dalla superficie dello schermo. Esso ha principalmente il compito di decrementare la lunghezza dell'array legato agli eventi `touch`, dichiarando così concluso l'uso di un particolare dito. Se viene richiamata con un solo dito presente su schermo vuol dire che non ci sono più dita sulla superficie e quindi l'array `touch` viene distrutto.

La creazione dell'array ed il suo popolamento sono sequenziali: per ogni nuovo dito appoggiato l'array incrementa la propria lunghezza; questo implica che l'indice X dell'array corrisponde al dito appoggiato per X+1-esimo.

Per quanto riguarda il `multitouch` esso può essere gestito tramite l'array visto sopra, tracciando e calcolando opportunamente (a seconda dell'operazione che si desidera realizzare) la distanza tra 2 o più punti. Per fare un esempio con questo sistema sono stati implementati il `pinch` ed il `reverse-pinch` sulla `mindmap`.

Per i dispositivi compatibili sono previste anche funzioni riguardanti le `gesture`: esse si occupano di gestire più di un `touch` contemporaneo, permette effetti quali `pinch, rotate, reverse-pinch`, etc. Esse sono:

- **gesturestart:** si attiva quando viene appoggiato sulla superficie dello schermo un numero di dita maggiore di 1. Se per esempio non è presente nessun dito a schermo e vi si appoggia il primo questo evento non è invocato; se un dito è già presente e si appoggia il secondo viene invocato *gesturestart*;
- **gesturechange:** si attiva quando vengono eseguiti movimenti delle dita premute a schermo con un numero di dita maggiore di 1;
- **gestureend:** si attiva quando la `gesture` termina, ovvero sullo schermo rimangono appoggiate unicamente 0 o 1 dita.

Queste funzioni, tuttavia, risultano essere attulmente compatibili solo con i dispositivi *apple* ma, come detto sopra, è possibile creare lo stesso effetto elaborando operazioni aritmetiche derivabili dai vari dati dell'array `touch`.

Come si può leggere sul sito del `W3C` relativo a questi standard (`http://www.w3.org/TR/2011/WD-touch-events-20110505/`), essi sono attualmente ancora in fase di bozza e non sono pienamente definiti; la difficoltà principale nel definire degli standard concreti risiede nella molteplicità di informazioni che devono essere interpretate dai dispositivi `touch`. A differenza di mouse e tastiera che dispongono di un numero di tasti abbastanza elevato a cui attribuire funzioni diverse, il tocco e/o la pressione a schermo risultano estremamente difficili da interpretare ed i fattori in gioco sono tantissimi: numero di dita a schermo, movimenti delle singole dita, pressione esercitata, durata temporale della pressione, `gesture` su un particolare elemento a schermo, etc.

Ciò rende molto complicato stabilire norme ben definite, soprattutto considerando che i dispositivi hardware che sfruttano questa tecnologia tendono ad essere tra loro estramamente diversi, con comportamenti e tecnologie uniche (e quasi sempre non trasparenti). Nella prossima sezione analizzeremo il comportamento dei `device` studiati rispetto a questi standard.

3.2 Device e loro comportamenti

Il problema principale dei dispositivi `touch` è il seguente: essendo gli standard ancora incompleti (si veda *Touch e multitouch events (3.1)*) risulta essere totalmente assente anche un controllo da parte degli organi competenti sui produttori di hardware, che hanno così totale libertà di utilizzo delle funzioni legate al `touch`, senza dover rispettare nessuna norma/regola.

A differenza delle altre periferiche di input canoniche (mouse e tastiera), come già detto nella sezione precedente il `touch` non è sempre interpretato correttamente, a causa delle molteplici funzionalità che possono essere scatenate da un singolo ed unico input. Inoltre spesso i `device` cercano di rendere navigabili anche siti web non dotati di eventi `javascript touch`, ma unicamente di quelli mouse e keyboard. Come si può facilmente intuire il tocco è ben diverso dal mouse, in quanto non è sempre presente sullo schermo, non è per forza unico (più dita contemporaneamente sullo schermo), non traccia linee per passare da un punto all'altro, etc. Risulta quindi complicato (e non sempre possibile) cercare di "adattare" eventi pensati per mouse e tastiera a input di tipo tattile.

Anche l'orientamento del dispositivo può risultare, seppure in misura molto inferiore rispetto al `touch`, complicato da gestire. Aanalizzeremo ora come i `device` studiati si sono comportati rispetto a questi due aspetti.

3.3 Touch

I siti web che prevedono eventi `javascript` attualmente non includono quasi mai eventi `touch`, a causa della scarsa lungimiranza di chi progetta il sito; questo limite obbliga il `device` a cercare di interpretare secondo i propri criteri gli eventi mouse e keyboard, poichè quelli `touch` non risultano gestiti in quanto non previsti.

3.3.1 iPad/iPhone

I dispositivi *apple* attuano la seguente interpretazione degli eventi `touch` all'interno del `browser`:

- il tocco rapido (inferiore ai 0.4 secondi) viene interpretato come "sfioramento" e invoca la funzione equivalente a mouseover; questo comportamento, in alcuni casi non sempre chiariti, viene applicato anche a seguito di pressioni ben più lunghe del tempo sopra citato, risultando di difficile previsione;

- una volta toccato un elemento esso rimane "evidenziato" (effetto equivalente ad un cursore del mouse che rimane sopra un'area impostata per reagire all'evento *onmouseover*). Questo effetto perdura finchè non viene toccata un'area diversa da quella evidenziata sulla pagina web o finchè non si tocca ancora una volta l'elemento evidenziato, innescando una chiamata equivalente al click del mouse;
- il trascinamento di un dito sulla pagina causa lo scorrimento di essa in orizzontale e/o verticale (se dotata di `scrollbar`);
- il trascinamento di 2 o più dita su un'area dotata di `scrollbar` interne causa lo scorrimento di quest'area, lasciando invariato lo scorrimento dell'intera pagina web;
- il `pinch e reverse-pinch` sulla pagina consentono di zoomare come se stessimo attivando/disattivando una lente di ingrandimento, consentendo in molti casi di toccare elementi altrimenti troppo piccoli nella visualizzazione di default;

3.3.2 Iconia

Il dispositivo *acer* attua la seguente interpretazione degli eventi `touch` all'interno del `browser`:

- il dispositivo non attua alcuna distinzione tra `touch` e sfioramenti ed il tocco su un oggetto attiva simultaneamente sia l'effetto *onmouseover* che il click;
- cliccando su un'area esterna l'effetto *over* causato dall'interazione precedente verrà disattivato, ma se l'area appena toccata è dotata di eventi sensibili all'hover o al click questi verranno immediatamente attivati come visto nel punto precedente;
- il trascinamento di un dito sulla pagina causa lo scorrimento di essa in orizzontale e/o verticale (se dotata di `scrollbar`);
- il trascinamento di un dito su un'area dotata di `scrollbar` interne causa lo scorrimento di quest'area, lasciando invariato lo scorrimento dell'intera pagina web;
- il `pinch e reverse-pinch` sulla pagina consentono di zoomare come se stessimo attivando/disattivando una lente di ingrandimento, consentendo in molti casi di toccare elementi altrimenti troppo piccoli nella visualizzazione di default;

3.4 Orientation

`Javascript` e i `CSS` sono in grado, grazie alle ultime implementazioni, di percepire l'orientamento del dispositivo, fornendo alcune interessanti funzionalità che consentono di ottimizzare la visualizzazione delle pagine web, massimizzandone l'utilizzo.

Il foglio di stile ha il compito di definire la disposizione degli elementi e consente, grazie alle *media query*, di stabilire regole di stile nel caso la pagina sia visualizzata in orizzontale (`landscape`) o in verticale (`portrait`). Esso si occupa autonomamente di percepire l'orientamento del `device` e non richiede nessuna implementazione o funzionalità particolare; il comportamento riscontrato è uguale per tutti i dispositivi testati.

`Javascript` è invece il vero e proprio strumento che si occupa di dialogare con il dispositivo e consente, intercettandone l'orientamento, di avviare determinate funzioni a seconda di come sia orientato il `device`. Esattamente come succedeva per le funzionalità legate al `touch`, esso viene interpretato in maniera leggermente diversa a seconda di quale dispositivo lo stia eseguendo; analizziamo ora i casi testati.

3.4.1 iPad/iPhone

I dispositivi *apple* attuano la seguente interpretazione degli eventi `orientation` all'interno del `browser`:

- il gestore *onorientationchange* applicato al *body* permette di invocare determinate funzioni al cambio dell'orientamento del `device`;
- la funzione *window.matchMedia* consente di accedere alle informazioni del dispositivo, effettuando una interrogazione e potendo così determinare il suo orientamento.

3.4.2 Iconia

Il dispositivo *acer* attua la seguente interpretazione degli eventi `orientation` all'interno del `browser`:

- il gestore *onorientationchange* applicato al *body* permette di invocare determinate funzioni al cambio dell'orientamento del `device`;
- la funzione *window.matchMedia* non risulta essere attualmente compatibile con la tecnologia *Android*; per ovviare a questo problema è stato necessario far interagire i comportamenti di `javascript` con il `CSS`: dato che i fogli di stile sono in grado di percecipire autonomamente l'orientamento del `device` ed applicano correttamente stili diversi a seconda che ci si trovi in `landscape` o in `portrait` si è pensato di attivare determinate funzioni a seconda dei valori di certi elementi.
  Un elemento marcato come invisibile all'interno della pagina web è stato impostato, tramite foglio di stile, per avere *font-size* pari a 11px nel caso la pagina sia in `landscape`, 10px nel caso sia in `portrait`. `Javascript`, dopo aver importato dinamicamente il `CSS` così da assicurarsi che il foglio di stile venga caricato prima delle funzioni che intendono interrogarlo, si occupa di controllare i valori *font-size* dell'elemento invisibile, in modo tale da capire se il `device` si trovi in `landscape` o in `portrait` ed applicare i comportamenti corretti a seconda dell'orientamento.

Intuitivamente questa operazione sembra molto complicata e poco razionale e verrebbe da chiedersi come mai non sia stata usata la funzione *window.orientation* (per esempio con *orientation* == 0 || *orientation* == 180 per l'orientamento `portrait` e *orientation* == 90 || *orientation* == 270 per l'orientamento `landscape`); questa soluzione, tanto semplice quanto immediata non è però applicabile per il seguente motivo: l'orientamento di default dei dispositivi *apple* (corrispondente quindi ad un *orientation* == 0) è `portrait`, mentre quello di *acer* è `landscape`. Ciò portava ad avere comportamenti corretti su un dispositivo e comportamenti opposti sull'altro; con la soluzione vista sopra questo problema è stato aggirato con successo.

Questa diversità ha dimostrato ancora una volta come la mancanza di standard possa rendere complicata se non addirittura impossibile la programmazione omogenea su dispositivi diversi.

4 Conclusioni

4.1 Rapporto preventivo/consuntivo

Di seguito viene riportata la tabella riassuntiva delle ore dedicate alle varie attività con un confronto diretto tra le quelle pianificate a preventivo e quelle effettive a consuntivo, seguita dai relativi diagrammi ed istogrammi.

Attività	Data inizio	Data termine	Preventivo	Consuntivo	Scostamento
Formazione/analisi	21/04/2011	02/05/2011	80	40	-40
Progettazione	03/05/2011	10/05/2011	40	40	0
Codifica	11/05/2011	03/06/2011	80	100	+20
Verifica	06/06/2011	01/07/2011	80	70	-10
Documentazione	27/04/2011	15/07/2011	20	50	+30
TOTALE	**21/04/2011**	**15/07/2011**	**300**	**300**	**0**

Tabella 2: riassunto ore per attività

Come si può notare dalle date di inizio/termine legate alla stesura della documentazione essa è stata presente in ogni fase del progetto, documentando tutti i risultati ed obiettivi raggiunti al termine di ogni attività e portando valore e qualità al progetto stesso.

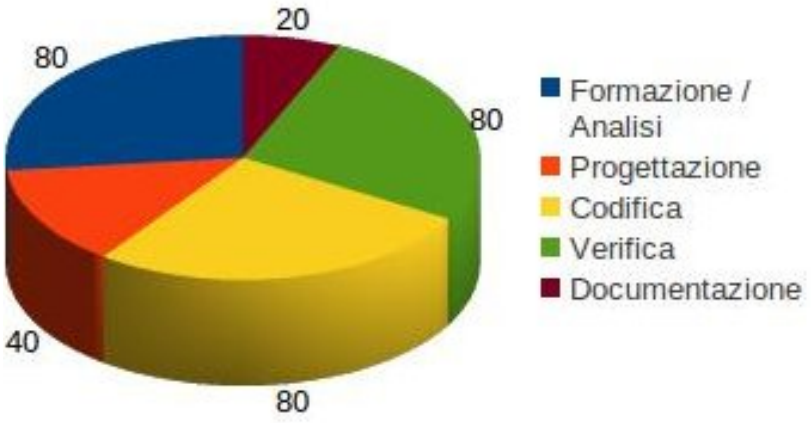

Figura 4: ore a preventivo

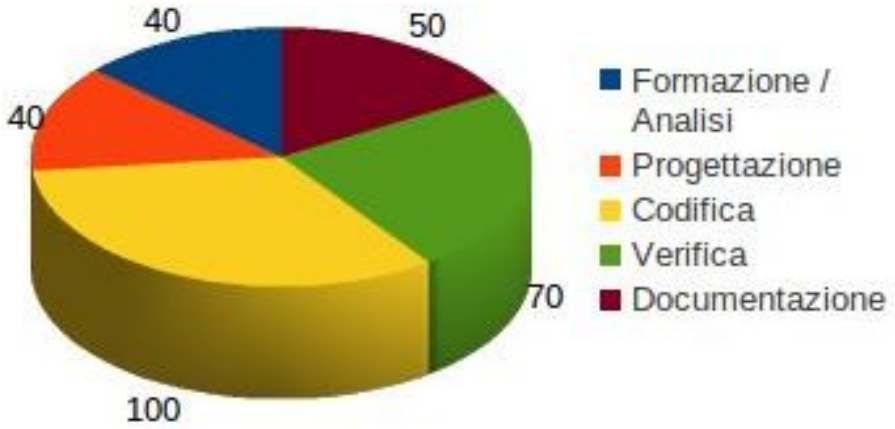

Figura 5: ore a consuntivo

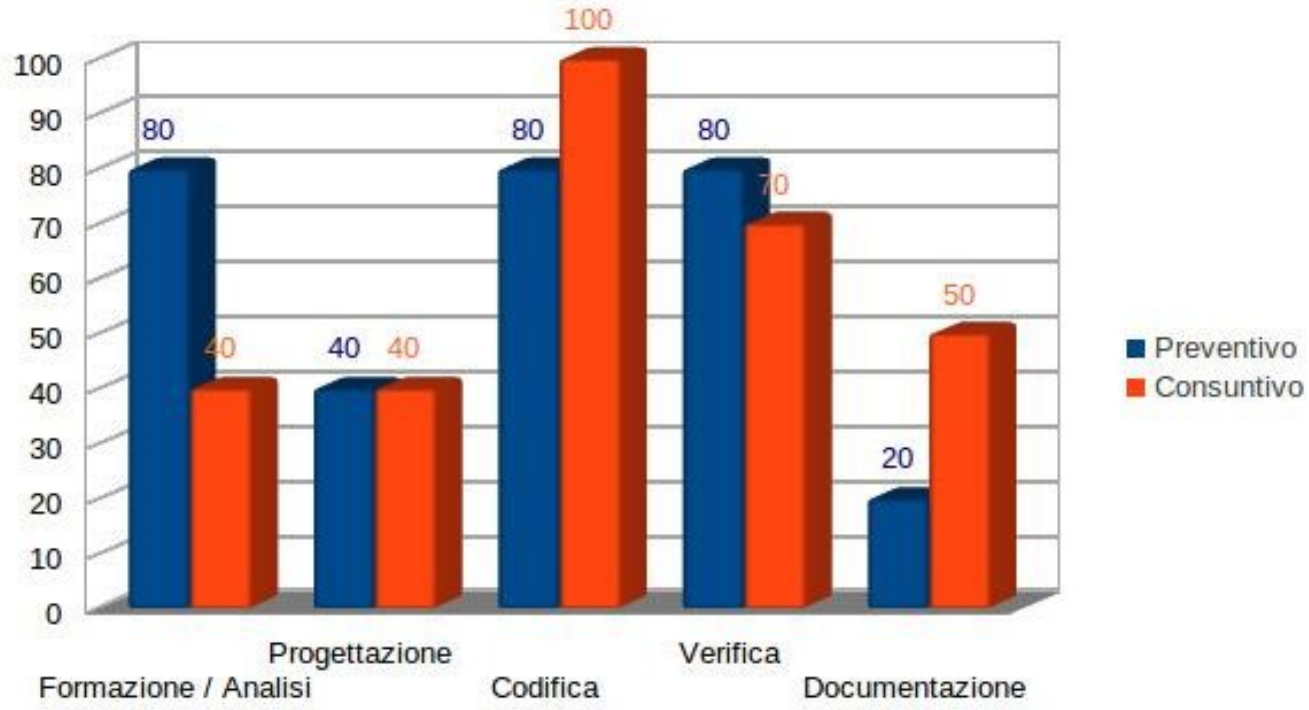

Figura 6: confronto ore preventivo/consuntivo

Come si nota dalla tabella *riassunto ore per attività (2)* e dall'istogramma *confronto ore preventivo/consuntivo (6)* c'è un discostamento su diverse attività rispetto alle ore preventivate: in particolare la formazione ha richiesto molte ore in meno rispetto a quelle pianificate, a causa soprattutto del numero ridotto di norme ben definite ed alla quasi totale assenza di standard `touch`.

L'attività di verifica è stata leggermente inferiore a quanto preventivato, anche se risulta essere molto vicina a quanto previsto durante la fase di pianificazione.

Le attività che invece hanno richiesto ore in più rispetto a quelle inizialmente calcolate sono state la codifica e la stesura della documentazione. Questo è dovuto principalmente a due fattori:

1. confrontando i vari `device` il loro comportamento è risultato essere assolutamente poco omogeneo e spesso ci si è trovati di fronte ad eventi imprevedibili eseguendo la stessa operazione su apparecchi diversi. E' stato quindi necessario puntare massicciamente su una programmazione che cercasse di standardizzare gli eventi dei `device`, annullando quasi del tutto i comportamenti di default così da avere, a parità di input, gli stessi output su tutti i dispositivi. Questo era stato pianificato fin dall'inizio dello stage ma ha richiesto ore in più rispetto a quanto preventivato proprio a causa di assenza di comportamenti standard da parte dei diversi hardware;
2. conseguentemente a quanto detto nel punto 1 è stato necessario aumentare notevolmente anche il numero di ore dedicate alla documentazione, poichè l'azienda ospitante ha manifestato grande interesse riguardo questa differenza di comportamenti tra i vari dispositivi e la quasi totale assenza di standard e di conformità tra i `device`.

4.2 Conoscenze possedute

Le conoscenze da me possedute al momento dell'inizio dello stage sono state fondamentali per affrontare il progetto e riuscire a raggiungere tutti gli obiettivi prefissati entro il tempo stabilito. Esse riguardano principalmente:

- la struttura e le funzionalità del software di partenza *MindSlide*, progettato e programmato durante il progetto di ingegneria del software A.A. 2010/2011 dal gruppo *IronMad Project*, di cui il sottoscritto faceva parte;
- i linguaggi `HTML5, CSS e javascript`.

4.3 Conoscenze acquisite

Grazie all'esperienza di stage mi è stato possibile incrementare le conoscenze dei linguaggi sopra citati ed avere una maggiore consapevolezza del loro impatto all'interno di `browser` diversi: lavorando a stretto contatto con professionisti del settore ho potuto avere prova tangibile di come i `browser` siano tra loro estremamente diversi e che raramente (e solo in casi molto semplici) la navigazione su una pagina web risulti essere ugualmente fruibile su ognuno di essi (si veda il capitolo *Device e loro comportamenti (3.2)*).

Altre conoscenze acquisite sono senz'altro quelle legate alle funzionalità `touch, multitouch` ed agli strumenti portatili come i `tablet` e gli `smartphone` con cui non avevo mai avuto occasione di lavorare prima. Queste tecnologie, attualmente in fase di larghissima espansione, sono indubbiamente uno strumento destinato a crescere sempre di più, anche in base alla tendenza del mercato, orientato a soddisfare la sempre più comune esigenza di essere connessi ovunque ed in qualsiasi momento. Trovo quindi estremamente importante aver imparato ad utilizzare e lavorare con questi dispositivi, in quanto è prevedibile che in un futuro non troppo lontano buona parte della programmazione sarà incentrata sullo sviluppo di applicazioni locali e via web per questi `device`. Inoltre ho imparato come sia importante iniziare a sviluppare siti internet

prevedendo fin dalla progettazione la possibilità che essi siano visitati da dispositivi `touch`, così da permettere di massimizzare l'esperienza d'uso da qualsiasi hardware si stia utilizzando.

La conoscenza più grande che ho acquisito durante questo stage riguarda indubbiamente gli standard e la gestione degli eventi: questo progetto ha messo in luce come la mancanza di norme ben definite porti i produttori di hardware ad interpretare comandi ed eventi spesso in modo troppo libero, compromettendo la visibilità di molte applicazioni progettate/costruite su o per dispositivi diversi. Ho perciò imparato quanto sia importante rispettare gli standard vigenti e, nel caso questi non siano definiti o solamente abbozzati, attenersi ad una politica più generale possibile, senza introdurre nulla che non sia riproducibile agevolmente anche su piattaforme/sistemi diversi.