Giacomo Quadrio

Drawing App

Specifica tecnica

Sommario

Il presente documento riporta l'insieme delle scelte progettuali che si intendono adottare durante lo sviluppo del software Drawing App.

Informazioni documento

Nome: specifica_tecnica_1.0.pdf
Versione: 1.0
Data creazione: 31/05/2012
Data ultima modifica: 10/06/2012
Stato del documento: Approvato

Registro delle modifiche:

Versione	Data	Modifiche effettuate
1.0	10/06/2012	Effettuate correzioni generali
0.7	08/06/2012	Inserito capitolo 4, prototipo interfaccia
0.6	08/06/2012	Terminato capitolo 3, specifica dei componenti
0.5	05/06/2012	Iniziato capitolo 3, specifica dei componenti
0.4	05/06/2012	Inserito sotto capitolo 2.4 API utilizzate
0.3	04/06/2012	Inseriti capitolo tecnologie, sotto capitoli 2.1, 2.2, 2.3
0.2	1/06/2012	Inseriti capitolo 1 - introduzione
0.1	31/05/2012	Impostata struttura generale

Tabella 1: registro delle modifiche

Indice generale

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire e mostrare le scelte progettuali, ad alto livello, che sono state adottate per realizzare il prodotto. Al suo interno verranno presentate le principali classi che lo compongono.

1.2 Scopo del prodotto

Il prodotto ha lo scopo di realizzare un'applicazione dedicata a dispositivi mobile come tablet e smartphone, con lo scopo di permettere all'utente di prendere appunti con varie modalità. L'applicazione prevede quindi che siano implementate varie funzionalità tipiche dei programmi dedicati al disegno o alla stesura di appunti ed anche altre funzionalità più particolari come la possibilità di registrare l'audio ambientale e sincronizzarlo con il testo o tratti presi.

1.3 Glossario

Per praticità e a scanso di equivoci, verrà creato un apposito documento denominato "Glossario" che conterrà tutti i termini e le sigle utilizzate nel presente documento. Questi termini verranno contrassegnati all'interno del testo con un'apposita sottolineatura.

1.4 Riferimenti

1.4.1 Normativi

- Documento di Analisi dei Requisiti, v 1.0 (Analisi_dei_requisiti_v1.0.pdf)

1.4.2 Informativi

- JQuery (http://docs.jquery.com/Main_Page)
- PhoneGap (http://docs.phonegap.com/en/2.0.0/guide_getting-started_index.md.html)
- JSON2 (https://github.com/douglascrockford/JSON-js)
- HTML Canvas 2D Context (http://dev.w3.org/html5/2dcontext/)

2 Tecnologie

Questa sezione descriverà i linguaggi, le tecnologie e API utilizzate nello sviluppo dell'applicazione.

2.1 Hardware

- Personal Computer con S.O. Mac OSX Lion 10.7.4
- Smart Phone LG Optimus One P500
- Tablet Acer Iconia A500

2.2 Editor e versionamento

- Eclipse 3.7.2 (per la realizzazione ed i test dell'applicativo Android)
- Sublime Text 2 (per la realizzazione del codice Javascritp, HTML5 e CSS)
- Git (strumento per il versionamento)

2.3 Strumenti utilizzati

2.3.1 HTML5

Linguaggio markup per la progettazione delle pagine web. Nonostante sia ancora in fase di definizione è stato scelto perché offre nuovi comandi e funzionalità di base più avanzate rispetto alle sue versioni precedenti. Tali funzionalità sarebbero altrimenti disponibili unicamente tramite plug-in o estensioni proprietarie specifiche del browser che si sta utilizzando. Una delle fundionalità principali utilizzate è il Canvas che, tramite l'ausilio di JavaScript, permette di creare animazioni o, nel caso specifico dell'applicazione oggetto dello stage, di disegnarvi all'interno.

2.3.2 Javascript

Linguaggio di scripting orientato agli oggetti, interpretato e che viene comunemente utilizzato nei siti web. Questo linguaggio è utilizzato all'interno del progetto poiché permette una notevole interazione con l'elemento Canvas di HTML5 oltre che la definizione di funzioni in grado di interagire con il markup HTML.

2.3.3 JQuery 1.7.2

Libreria di funzioni JavaScript, cross-browser per le applicazioni web e che permette una semplificazione della progettazione lato client delle pagine HTML. La libreria JQuery è stata scelta in quanto mette a disposizione molte funzioni utili per interagire in maniera efficiente con Canvas tramite eventi mouse e touch screen, oltre che ad altre funzioni base che velocizzano e semplificano numerose operazioni.

2.3.4 JSON

JSON (JavaScript Object Notation) è un formato adatto per lo scambio dei dati in

applicazioni client-server. Tale strumento è utilizzato per effettuare parsing di elementi da Javascript a stringa e viceversa. Verrà utilizzato all'interno dell'applicativo tramite la libreria Javascript JSON2.js e servirà per il salvataggio in memoria e ripristino delle pagine del canvas.

2.3.5 Android SDK

Ambiente di sviluppo che permette di realizzare applicazioni Android e sarà quindi utilizzato, insieme a PhoneGap, per la realizzazione dell'applicativo mobile.

2.3.6 PhoneGap – Cordova 1.8.1

Framework mobile di sviluppo open source che permette di realizzare applicazioni mobile utilizzando JavaScript, HTML5 e CSS3 anziché i linguaggi nativi per gli specifici dispositivi. Tale Framework consente infatti di realizzare applicazioni native basate interamente sulle tecnologie sopraelencate, fornendo però delle API in grado di accedere a quelle risorse hardware a cui normalmente non si potrebbe aver accesso. Il risultato finale sarà quindi un'applicazione ibrida.

2.4 API utilizzate

2.4.1 Canvas 2D API

HTML5 mette a disposizione un tag che permette di disegnare all'interno del browser ovvero, come già accennato, il tag <canvas>. Esso permette di disegnare forme, linee, renderizzare testo, importare immagini, manipolare pixel ed esportare il contenuto in un file statico. Il canvas viene dichiarato con la seguente sintassi:

```
<canvas id = "cvs"></canvas>
```

Una volta dichiarato, avremo a disposizione un'area bianca di disegno. Per poter però accedere al *drawing context* e disegnare, dobbiamo utilizzare JavaScript:

```
<script>
      var canvas = document.getElementById("cvs");
      var context = canvas.getContext("2d");
</script>
```

Il context è quindi il nostro accesso diretto all'area di disegno del canvas ovvero l'unico mezzo attraverso il quale è possibile disegnarvi all'interno. Sul context verranno quindi richiamati i vari metodi per tracciare forme e linee. Nello specifico per l'applicazione sviluppata si sono utilizzati i seguenti metodi: strokeStyle, beginPath, moveTo, lineTo, stroke, closePath.
Sull'oggetto context verranno richiamati anche gli appositi metodi necessari a modificare dimensione e colore durante la stesua del tratto. Nel dettaglio di seguito è riportato un esempio di modifica della dimensione del tratto:

```
      if(this.id == 'one'){
            lineW = 1;
            context.lineWidth = lineW;
```

```
        }
```

2.4.2 PhoneGap recording API

PhoneGap mette a disposizione delle API per poter accedere all'hardware del dispositivo mobile ed effettuare operazioni di registrazione dell'audio e salvataggio del file, in formato MP3, all'interno della memoria del dispositivo. Il metodo per la registrazione verrà richiamato su di un oggetto di tipo Media:

```
var mediaRec = new Media(src, onSuccess, onError)
```

Tale oggetto richiede in input un percorso *src* dove effettuare il salvataggio del file e dei parametri per la gestione degli errori.

Per avviare la registrazione verrà richiamato il seguente metodo `mediaRec.startRecord()`, mentre per terminare dovrà essere richiamato il metodo `mediaRec.stopRecord()`.

2.4.3 PhoneGap play API

Analogamente alle API per la registrazione dell'audio, PhoneGap ne mette a disposizione anche per effettuare la riproduzione di specifiche tracce in formato MP3. L'oggetto utilizzato per effettuare la riproduzione sarà sempre un oggetto di tipo *media* con in input una variabile *src* per indicare la fonte da cui leggere l'audio. I metodi per la riproduzione e pausa sono poi i seguenti:

```
myMedia.play();
myMedia.Pause();
```

2.4.4 PhoneGap getPicture API

Con PhoneGap è possibile accedere direttamente alla memoria del dispositivo per poter caricare immagini all'interno della propria applicazione. Il caricamento avverrà tramite l'apertura dell'applicazione galleria, selezionando l'immagine interessata. Il metodo per effettuare tali operazioni è il seguente:

```
navigator.camera.getPicture(canvasManager.onPhotoURISuccess,
      canvasManager.onFail, { quality: 50, destinationType:
      navigator.camera.DestinationType.FILE_URI, sourceType:
      navigator.camera.PictureSourceType.PHOTOLIBRARY });
```

Tramite l'indicazione della fonte *PHOTOLIBRARY* verrà indicato quindi che la fonte delle immagini è la libreria fotografica. Il metodo infatti rende possibile il caricamento tramite immagine scattata dalla fotocamera.

2.4.5 PhoneGap saveFile e readFile API

Tra le API disponibili ve ne sono anche alcune dedicate alla lettura e scrittura di file all'interno della memoria del dispositivo. E' infatti possibile effettuare delle chiamate a funzioni che esplorano la memoria del dispositivo, cercano determinati file e se questi

non sono presenti vengono creati e salvati con un apposito contenuto. La funzione per effettuare queste operazioni è la seguente:

```
window.requestFileSystem(LocalFileSystem.PERSISTENT,  0,  saveFile.gotFS,
saveFile.failW);
```

E' poi possibile anche selezionare i file per effettuarne la lettura del loro contenuto e riutilizzarlo all'interno dell'applicativo. Tale funzione è la seguente:

```
window.requestFileSystem(LocalFileSystem.PERSISTENT,  0,  readFile.gotFS,
readFile.failR);
```

3 Specifica dei componenti

3.1 self

Classe che si occupa di appendere gli eventListener ai vari elementi interattivi dell'applicazione.

3.2 trackDrawer

Classe che si occupa di definire le funzioni che disegnano all'interno del canvas la linea continua, nonché la registrano le coordinate dei punti tracciati.

3.2.1 Metodi

- *preDraw: function(event)*
  Al verificarsi dell'evento touchStart, la classe prepara le operazioni di disegno all'interno del canvas
- *draw: function(event)*
  La funzione prende le coordinate derivanti dal verificarsi dell'evento touchMove ed invoca il metodo move per tracciare le linee. Al suo interno viene effettuata anche la registrazione, in un apposito array, di tutte le coordinate dei punti tracciati all'interno del canvas
- *move: function(i, changeX, changeY)*
  Funzione che effettua il tracciamento dei tratti da un punto A ad un punto B

3.3 trackTools

Classe che contiene al suo interno i metodi per poter modificare il tratto della linea tracciata dall'applicazione.

3.3.1 Metodi

- *selectTool: function(event)*
  Funzione che permette di cambiare lo strumento di disegno attualmente utilizzato
- *changeColor: function(event)*
  Funzione che permette di cambiare il colore del tratto
- *changeSize: function(event)*
  Funzione che permette di cambiare lo spessore del tratto

3.4 audioManager

Classe che si occupa di effettuare le operazioni di registrazione e riproduzione dell'audio in sincronia con la stesura del tratti su canvas.

3.4.1 Metodi

- *recordAudioStart: function()*
  Funzione che avvia la registrazione della traccia audio. Verrà effettuata anche la registrazione delle informazioni della traccia all'interno di un array
- *recordAudioStop: function()*
  Funzione che termina la registrazione della traccia audio
- *onSuccess: function()*
  Messaggio restituito al termine di una corretta operazione di lettura o registrazione del file audio
- *onError: function(error)*
  Messaggio restituito al verificarsi di un errore durante le operazioni di lettura o scrittura
- *playAudioDraw: function()*
  Funzione che si occupa di riprodurre la traccia audio e nel contempo disegnare in maniera sincrona i tratti all'interno del canvas. La funzione si occuperà delle operazioni di preparazione del canvas e delle tracce audio da riprodurre, richiamando poi la funzione sincroDraw per il disegno effettivo del tratto
- *sincroDraw: function(x, a, c)*
  Funzione che si occupa di disegnare nel canvas i tratti necessari durante la riproduzione della traccia audio
- *defMediaList: function()*
  Funzione che definisce, al momento del cambio pagina, la lista dei brani da riprodurre

3.5 canvasManager

Classe che si occupa di fornire alcune operazioni basilari ed essenziali riguardanti l'interazione con il canvas.

3.5.1 Metodi

- *clearCanvas: function(event)*
  Funzione che si occupa di ripulire il canvas corrente da eventuali elementi disegnati su di esso. L'operazione si occupa anche di svuotare gli array contenenti le coordinate dei punti precedentemente tracciati
- *loadNewImg: function(event)*
  Funzione che si occupa di avviare il caricamento all'interno del canvas di un'immagine presente all'interno della memoria del dispositivo
- *onPhotoURISuccess: function(imageURI)*
  Funzione che si occupa di disegnare all'interno del canvas l'immagine precedentemente selezionata dalla funzione loadNewImage
- *onFail: function(message)*
  Funzione utilizzata per segnalare eventuali errori nel caricamento dell'immagine
- *copyArray: function(event)*
  Funzione che si occupa di copiare il contenuto dell'array principale dei tratti nell'array secondario di supporto
- *copyArrayInv: function(event)*
  Funzione che si occupa di copiare il contenuto dell'array ausiliario dei tratti nell'array principale
- *reDraw: function(event)*

Funzione che si occupa di ridisegnare, al partire da un array di punti, tutti i tratti di uno specifico canvas

3.6 history

Classe che si occupa di fornire le funzioni necessarie ad annullare e ripristinare un dato tratto disegnato all'interno del canvas.

3.6.1 Metodi

- *unDoLine: function(event)*
  Metodo che annulla l'ultima linea tracciata all'interno del canvas, a patto che ve ne fosse stata disegnata almeno una
- *reDo: function(event)*
  Metodo che ripete l'ultima linea tracciata all'interno del canvas, a patto che ve ne fosse stata disegnata almeno una e che questa fosse stata cancellata

3.7 navigatorTools

Classe che si occupa di esplorare le varie pagine dei canvas a disposizione dell'utente

3.7.1 Metodi

- *precPage: function()*
  Operazione che permette di visualizzare la pagina precedente a quella attuale, a patto che essa esista. Verrà quindi ripulito il canvas e caricato l'array di tratti richiesto
- *sucPage: function()*
  Operazione che permette di visualizzare la pagina successiva a quella attuale, a patto che essa esista. Verrà quindi ripulito il canvas e caricato l'array di tratti richiesto. Se questo non dovesse esistere, il canvas verrà ripulito e fornita una nuova pagina

3.8 saveFile

Classe che si occupa delle operazioni di salvataggio all'interno di un file di testo degli array dei tratti e degli array delle tracce audio.

3.8.1 Metodi

- *saveCanvas: function(event)*
  Funzione che avvia il salvataggio degli array dei tratti all'interno dell'apposito file di testo. Chiamerà la funzione gotFS per verificarne l'esistenza
- *gotFS: function(fileSystem)*
  Funzione che verifica l'esistenza del file che si andrà a scrivere e richiamerà la funzione di scrittura. Se non esiste verrà creato
- *gotFileEntry: function(fileEntry)*
  Funzione che si occupa della conversione da array ad oggetto JSON stringa degli array contenenti le informazioni degli array dei tratti e tracce audio. La funzione si occupa poi di scrivere tali oggetti JSON stringa all'interno dell'apposito file

- *failW: function(error)*
  Funzione che si occupa di segnalare eventuali errori durante la scrittura all'interno del file di salvataggio

3.9 readFile

Classe che si occupa delle operazioni di lettura del file di testo memorizzato nel dispositivo e conversione dell'oggetto JSON nel relativo oggetto array necessario per effettuare il ripristino del canvas.

3.9.1 Metodi

- *loadCanvas: function(event)*
  Funzione che avvia la lettura degli oggetti stringa JSON dell'apposito file memorizzato all'interno del dispositivo. Chiamerà la funzione gotFS per verificarne l'esistenza
- *gotFS: function(fileSystem)*
  Funzione che verifica l'esistenza del file che si andrà a leggere. Se il file non è presente non verrà effettuata alcuna operazione di caricamento
- *gotFileEntry: function(fileEntry)*
  Funzione che si occupa di convertire da oggetto stringa JSON ad array di dati per poter definire gli array necessari al ripristino del canvas
- *failR: function(event)*
  Funzione che si occupa di segnalare eventuali errori durante la lettura del file di salvataggio

4 Prototipo di interfaccia

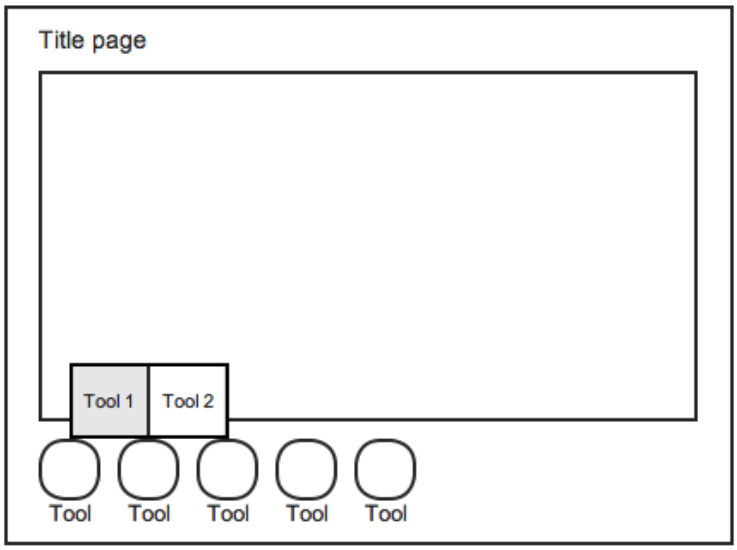

Immagine 1: bozza della struttura della pagina web

Il prototipo dell'interfaccia è stato ideato con un layout molto semplice. Vi è un’area principale centrale di disegno definita tramite Canvas; sotto tale area è poi presente una toolbar adibita ad ospitare i vari strumenti di disegno. Si è scelto una struttura di questo tipo proprio per fornire la maggior quantità di spazio disponibile per il disegno anche su dispositivi dotati di schermi non particolarmente grandi.

I vari sotto menù dedicati alla selezione dello strumento, dei colori e della grandezza del tratto sono stati realizzati tramite menù a scomparsa, questo sempre in ottica di non affollare eccessivamente l’interfaccia principale e permettere quindi di guadagnare maggiore spazio.
Sono stati poi scelti colori neutri e molto chiari sempre per mettere in risalto l'area di disegno e, di conseguenza, i tratti tracciati.