# TECNOLOGIE WEB E ACCESSIBILITÀ

January 12, 2015

Bogdan Suierica 1008089
Campagna Simone 1005922
URL : tecweb/~scampagn/cgi-bin/index.cgi
Per quel che riguarda la parte amministrativa del sito ( funzionalità spiegata successivamente ) è disponibile l'utente amministratore "admin" con medesima password ('admin')

## 1 Abstract

Il progetto Quartier 6/7 ha lo scopo di presentare un gruppo di youtuber in cerca di nuovi volti e talenti . Per far ciò il gruppo mette a disposizione questo servizio nel quale si richiede ai nuovi utenti di esprimere una propria idea comica ( pagina "POSTA IDEA" ) tramite la quale, in seguito all'approvazione da parte dell'amministratore, essa potrà rientrare tra le idee pubblicate potendo avere l'apporto da parte del publico in merito a commenti denotanti miglioramenti dell'idea stessa o di altra specie.

## 2 Struttura file

I file sono organizzati principalmente in 2 macro cartelle ( come da configurazione server studenti.math.unipd.it ) ovvero :

**cgi-bin** cartella contenente solamente gli eseguibili

| | |
|---|---|
| function_—.cgi | script adottati negli eseguibili di pagina |
| —.cgi | eseguibili di pagina |

**public_html** contenente a sua 5 sotto cartelle , rispettivamente

| | |
|---|---|
| css | cartella fogli di stile |
| js | cartella script javascript |
| media | cartella contenente immagini varie e logo |
| template | cartella relativa ai temlate perl implementati dagli eseguibili in "../cgi-bin/*.cgi" |
| xml | cartella contenente i database |

## 2.1 Implementazione file

**cgi-bin**

function_user.cgi script riguardanti le info utente

function_block.cgi script competenti nella configurazione dei moduli di pagina

function_system.cgi script relativi per cookie e mantenimento dello stato utente

**pubic_html/css**

page_—.css implementazione stile specifico di pagina

Module_—.css stile riguardante il modulo di pagina

0_—.css implementazione stile di stampa

1_—.css implementazione stile mobile

2_—.css implementazione stile tablet

3_—.css implementazione stile desktop

**public_html/js**

function_module.js funzionalità di accessibilità su campi di modulo

**public_html/media** contenitore di media usati nel servizio

**public_html/template**

—.tmpl template di pagina o modulo

**public_html/xml**

ideeAccept.xml database riguardante le idee sottoposte a giudizio amministrativo quindi non ancora pubblicate definito dallo schema "ideeAccept.xsd"

idee.xml base di dati delle idee accettate definito ndallo schema "idee.xsd"

team.xml elenco degli ideatori definito dallo schema "team.xsd"

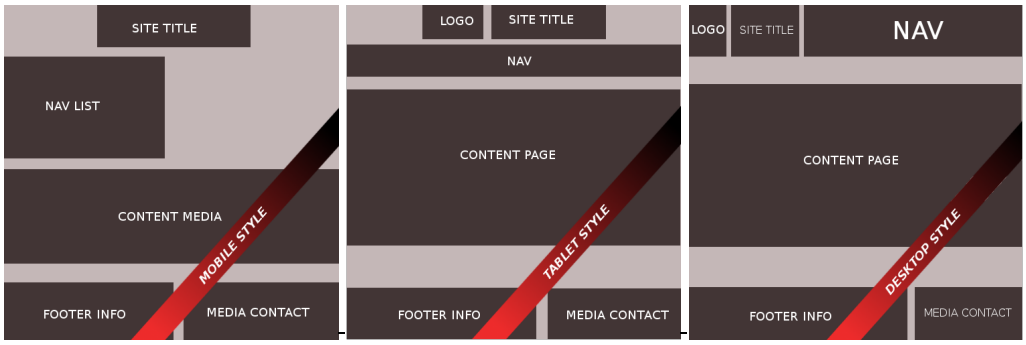
utenti.xml elenco dell'utenza definito dallo schema "utenti.xsd"

# 3 Layout

Il layout del servizio è strutturato in maniera tale che esso sia fruibile in ogni suo campo o testo a qualsiasi dispositivo *, ovvero definendo un layout per il small mobile, medium mobile e desktop.

Ecco di seguito un elenco dei layout :

mobile




## 3.1 Implementazione stili

Ogni pagina adotta un proprio foglio di stile nominato “page_” seguito dal nome di pagina competente. In essi vengono definiti 3 breakpoint specifici ( e non 4 dato che per il servizio in questione lo stile mobile e desktop risultano essere adattabili per le dimensioni di dispositivi tablet ) ovvero print, mobile/tablet ( max-width:1151px ) e desktop/tablet (min-width:1152px).

Successivamente vi sono i file Module_”module_name”.css, fogli di stile dei moduli di pagina ( header, userbar, breadcumbs, footer ) i quali adottano invece dei breakpoint convenzionali, potendo captare ( per quel che si può ) la tipologia di dispositivo attualmente collegato, includendovi quindi la corrispondente implementazione grafica ( file [0-3]_”name”.css ).

( Nota : si sarebbe voluto stabilire l’entità del dispositivo tramite degli script javascript, ma per gli ovvi motivi legati alle diverse configurazioni dei vari client ed prerogative d’accessibilità se ne è omesso l’uso )

Inoltre tutti i fogli di stile sono stati inseguito alla scrittura, ottimizzati nel peso facendo si che per stili mobili la media dei pacchetti richiesti per il solo stile non superi i 4,3KB, per tablet 4,5KB mentre per desktop 4,2KB.

# 4 Xml

Il database reperibile al path “./public_html/xml” consta di 3 file :

- “utenti.xml” : utenti del sito responsabili per l’approvazione ed amministratori

- "ideeAccept.xml" : racchiude le idee proposte degli utenti guest che saranno poi successivamente accettate dai soli utenti inscritti in "utenti.xml"
- "idee.xml" : racchiude le idee accettate dal admin in seguito alle richieste inviate e pubblicate alla pagina "idee.cgi"

### 4.1 Nota ai tipi d'utente e registrazione

Si è pensato per il servizio esposto di non concedere l'iscrizione a qualsiasi utente dato che essa è riservata ai soli utenti resposabili dell'approvazione delle idee inviate, permettendo quindi a chiunque voglia di poter postare la Sua.

Questo perché la maggior parte dei servizi riserva le proprie funzionalità alla sola utenza registrata, scoraggiando l'utente guest addentratosi per la prima volta nel sito e troppo pigro per la compilazione dei propri dati .

Dunque si è pensato ad una registrazione al volo nel momento della sottomissione dell'dea . Di conseguenza l'utenza destinata all'uso della barra di login è la sola responsabile dell'accettazione o meno dell'idee.

### 4.2 Utenti destinatari

Il servizio è destinato a tutti coloro che vogliono mettersi in mostra o dire "la propria" tramite qualche righe, ricercando consiglio in merito ad un qualche skatch comico o idea divertente balzatagli in mente, quindi non ancora del tutto messa in pratica ma che richiede forse qualche consiglio da chi, come noi, ne sa forse un po' di più

## 5 Accessibilità

Per quanto riguarda l'accessibilità sono stati sviluppati i seguenti punti

### 5.1 Colori

- I colori addottati del sito,a parte qualche oggetto, sono in scala monocromatica questo per favorire un risparmio all'alimentazione del dispositivo
- La combinazione di colori adottata permette l'uso del servizio anche a coloro che portano handicap da deuteranopia, protanopia e trinanopia.

### 5.2 Effetti visivi di supporto

Sono stati adottati degli effetti visivi di supporto per agevolare l'utente in ogni aspetto implementati per :

**menu**

1. Evidenziazione in contrasto della voce di menu attualmente selezionata

2. Definizione del colore delle voci di menu in pieno contrasto col background del "nav"
3. Differenziazione dei link già visitati dai precedenti con l'inscrizione in colore maggiormente flebile dei precedenti
4. Accentuazione dello stile della voce di menu attualmente puntata dal mouse

**userBar**

1. Comparsa di un bordo maggiorato al passaggio del mouse e click del mouse
2. Accentuazione dello stile alla selezione del bottone di login

**footer**

1. Aumento delle dimensioni del "media contact" al passaggio del mouse

## 5.3 Tag meta

1. Sono state adottate le giuste specifiche per quel che riguarda i titoli di pagina
2. Sono state definite le keywords relative e specifiche ad ogni pagina

## 5.4 Title e non vedenti

**submit**

ogni submit è stato provvisto di relativo title specifico (tralasciando ovviamente la parte amministrativa )agevolando il non vedente nella compilazione del form di commento o nell'invio d'una nuova idea .

Inoltre in caso di disabilitazione del javascript ( relativo alla stampa a schermo dei campi incompleti ) è stato redatto un gestore mostrante l'idea la cui compilazione ha creato l'errore .

**image** per le immagini invece, essendo esse non di contenuto eccetto il logo , non sono state dotate di relativi title non volendo appensantire la lettura già problematica per di suo, della pagina

**menu** ogni singola voce di menu è stata fornita di title potendo dare un'idea della pagina che sarà raggiunta dall'utente disabile

**link** ogni link è stato dotato di specifico title

## 5.5 Orientamento secondario

Per quel che riguarda il "Breadcumb" c'è da dire che essendo il servizio strutturato ad un singolo livello, dunque è pressoché inutile l'uso di tale provvisto di link alla sezione precedente. E' stato però scelto di inserirlo comunque, omettendo però eventuali link e relativi title che sarebbero solamente una ripetizione della voce di menu precedentemente letta.

## 5.6 Comodità d'uso

Per ogni modulo sono stati forniti delle semplici funzioni javascript agevolanti nell'inserimento di nuovi dati, ed evidenziando eventuali errori di compilazione

# 6 Perl

## 6.1 Architettura ed implementazione

### 6.1.1 Introduzione

Nel strutturare le funzionalità al servizio necessarie si è deciso di adottare un modello a Templating, incentivando quindi il riutilizzo del codice, oltre al mantenimento pulito delle pagine HTML sul quale lavorare.

Sono state adottate poi funzioni di controllo nelle varie pagine ( —.cgi ) implementate poi nei file function_—.cgi potendo rendere di più semplice comprensione della struttura del servizio.

### 6.1.2 I dati

I dati sono immagazzinati in formato Xml, come da specifica, di conseguenza il Modello dei dati è il DOM. L'istanziazione e la manipolazione del Modello dei dati avviene tramite il modulo Perl::LibXML, per favorire la rapidità di sviluppo. I motivi dietro la scelta di LibXML piuttosto che altri moduli simili sono la compatibilità con il server del progetto e la presenza di documentazione più estensiva rispetto ai concorrenti.

Script responsabili della scrittura dei dati, reperibili in function_block.cgi tra cui :

**gestRequestForIdea()**

relativa all'inserimento di un'idea dalla pagina Posta idea. tale non fa altro che creare il nuovo elemento ed attribuirgli un id univoco incrementale per poi scrivere il tutto su "ideeAccept.xml"

**accettaIdeeRequest()**

relativa invece all'accettazione da parte dell'amministratore dell'idea proposta. La funzione non fa altro che spostare il nodo inscritto nel db "ideeAccept.xml" e scriverlo in "idee.xml" accertandosi di usare un'attributo

id sempre incrementale ( ovvero il contratto per l'id è che sia pari all'id precedente ( se esistente, altrimenti 0) + 1.

**eliminaIdeeRequest()**

al contrario della precedente, questa invece non fa altro che eliminare semplicemente l'idea proposta

### 6.1.3 Views

La Vista dei dati avviene tramite pagine HTML. Il sito prevede alcune pagine statiche per esporre i dati sull'associazione, mentre le pagine come "IDEE" o "POSTA IDEA" saranno generate dinamicamente tramite script Perl.

Gli script responsabili delle varie Views sono gli script del file function_ block.cgi con le seguenti funzioni :

**setUserBar()**

subroutine la cui funzione è quella di settare la barra di login a seconda se esso sia stato eseguito o meno

**menuSection()**

subroutine la cui funzione non è altro si settare la voce di menu attualmente selezionata ( data la problematica del modello a Templating )

**getIdeaAccept()**

responsabile della stampa a schermo delle idee ancora non ancora esaminate, lette quindi dal DB "ideeAccept.xml"

**getIdea()**

funzione per la stampa delle idee accettate e quindi pubblicate ( pagina "idee.cgi" ) lette quindi da "idee.xml

### 6.1.4 Script

Tutti gli script relativi alle funzionalità di inserimento eliminazione e richieste di vario tipo sono inscritti anch'essi in "function_ block.cgi" tra cui :

**isThereIdea("db_ index")**

la funzione non fa altro che sulla base dell'indice passato ( 0 per "ideeAccept.xml" d 1 per "idee.xml" ) dire se vi sono o meno idee da poter esporre ( funzionalità adottata anche per l'inserimento di un nuovo id )

**getNewComment()**

reponsabile invece della gestione per la richiesta dovuta all'inserimento di un nuovo commento . Qui è giusto dire qualche parola in merito l'uso .

Per quel che riguarda la funzione in se non si fa altro che controllare l'integrità dei dati inviati dall'utente ( campi vuoti ad esempio lasciando

possibile l'uso di eventuali caratteri speciali e numeri ) e successivamente controllare se l'utente ha già commentato in precedenza il commento attraverso il confronto

$session->param('ableToComment') != $q->param('id_nuovo_commento')

questo per ovviare alle problematiche dovute ad un refresh di pagina . E' nota la possibilità del metodo ".remove('param name') disponibile su qualunque oggetto cgi ma ahimè non sempre si è dimostrato risultare corretto il suo funzionamento e dunque si è optato bloccare il commento fino a che, come giusto che sia, l'utente non visiti almeno 1 pagina diversa per poi ritornare nella pagina delle Idee.

**getNewRequestIdea()**

è una subroutines da controllo per :

- l'inserimento e/o cancellazione di nuove idee sulla base della circostanza che vede l'utente loggato o meno nel servizio
- l'inoltro di una qualche nuova idea
- il settaggio dei campi del modulo riportando eventuali errori in sede di compilazione della domanda o semplicemente mostrando a schermo il successo dell'avvenuto invio

Nota come fatta in precedenza per gestNewComment(), anche qui si è ricorso ad una varibile di sessione per evitare l'inserimento erroneo al refresh della pagina, è stata adottata la variabile "isAbleToSend()" avente come valore 0 o 1 a seconda che l'utente abbia già effettuato o no la richiesta. Tale verrà poi ripristinata, consentendo un nuovo invio, nel momento in cui l'utente visiti almeno un altra pagina .

### 6.1.5 Sessioni

Per sessioni e mantenimento dei dati fra di esse sono da citare :

**getSession()**

funzione responsabile del mantenimento dei dati di sessione

**setCookie()**

funzione responsabile per il settaggio lato client dei cookie utente

## 6.2 Considerazioni finali

Il servizio è ancora a livello prototipale e con una buona idea di base che potrà essere successivamente generalizzata ad idee di qualsiasi tipo. Funzionalità come eliminazione ad-hoc di commenti ed idee già postate, eliminazione/accettazione multipla di idee sono state omesse per non aumentare la complessità del servizio lasciandolo nella sua semplicità .

Un'altra nota di merito è per l'eliminazione dei commenti offensivi : essendo una delle prerogative di internet la libertà, non sono state adottate eventuali censure .