



UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA

*Corso di Laurea in Informatica*

**SVILUPPO PAGINA WEB PER LA PRESENTAZIONE  
DELLE PERFORMANCE DI UN'APPLICAZIONE  
TRAMITE PROXY SPECIALIZZATO**

**APPROCCIO AL PROBLEMA IN MODO ESPLORABILE  
TRAMITE LA LIBRERIA TANGLE**

Laureando  
**Michela Abbita**

Relatore: **Prof. Ombretta Gaggi**

## Sommario

Questo documento intende presentare il lavoro svolto durante il mio stage all'interno dell'azienda Zucchetti s.p.a. di Padova.

Viene presentato il bisogno tecnologico espresso dall'azienda e l'idea di base proposta per risolverlo.

L'intera attività è stata scandita da un Piano di Lavoro, che ha permesso il rilascio settimanale di documenti, prodotti e funzionalità aggiuntive.

In questo documento vengono spiegate nel dettaglio tutte le componenti architettoniche utilizzate per rendere più agevole la comprensione e l'esposizione del prodotto finale.

Attraverso l'uso dei Casi d'uso vengono esposti gli obiettivi preposti e infine analizzato il codice del prodotto finale.

# Indice

<b>1</b>	<b>Azienda</b>	<b>5</b>
<b>2</b>	<b>Il progetto</b>	<b>7</b>
2.1	Proposta di stage . . . . .	7
2.2	Ideazione e filosofia d'azione . . . . .	8
<b>3</b>	<b>Lo stage</b>	<b>9</b>
3.1	Piano di Lavoro . . . . .	9
3.2	Le attività preventivate . . . . .	9
3.3	Fasi di sviluppo effettive . . . . .	10
3.4	Diagramma . . . . .	10
<b>4</b>	<b>Premesse architettureali</b>	<b>12</b>
4.1	HTML . . . . .	12
4.2	DOM . . . . .	13
4.3	Canvas . . . . .	14
4.4	CSS . . . . .	15
4.5	JavaScript . . . . .	15
4.6	Mootools . . . . .	17
4.7	Tangle . . . . .	17
4.7.1	Inizializzazione . . . . .	18
4.7.2	Classi tangle standard e data-var . . . . .	18
4.7.3	Nuove classi . . . . .	19
4.7.4	Funzionalità utili . . . . .	19
4.8	Ajax . . . . .	19
4.9	JSON . . . . .	20
4.10	Tomcat . . . . .	20
<b>5</b>	<b>Sviluppo Prototipi</b>	<b>21</b>
5.1	Pagina 1 . . . . .	21
5.1.1	Attività . . . . .	21
5.1.2	Pagina 1-performance.js . . . . .	22
5.1.3	Pagina 1-index.html . . . . .	24
5.2	Pagina 2 . . . . .	26
5.2.1	Attività . . . . .	26
5.2.2	Pagina 2-performance.js . . . . .	26
5.2.3	Pagina 2-index.html . . . . .	31
5.3	Pagina 3 . . . . .	33
5.3.1	Attività . . . . .	33
5.3.2	Pagina 3-performance.js . . . . .	33
5.3.3	Pagina 3-index.html . . . . .	37
5.4	Pagina 4 . . . . .	39
5.4.1	Attività . . . . .	39
5.4.2	Pagina 4-performance.js . . . . .	40
5.4.3	Pagina 4-index.html . . . . .	41
<b>6</b>	<b>Sviluppo Prodotto finale</b>	<b>42</b>
6.1	Attività . . . . .	42

<b>7</b>	<b>I casi d'uso</b>	<b>46</b>
7.1	Casi d'uso	47
7.1.1	UC1 - Visualizzazione grafici	49
7.1.2	UC2 - Regolazione zoom	50
7.1.3	UC3 - Visualizzazione repo singolo	51
7.1.4	UC4 - Visualizzazione repo generale	53
7.1.5	UC5 - Regolazione tetto massimo tempo di risposta	54
7.1.6	UC6 - Regolazione KBrate	55
7.2	Definizione del prodotto	56
7.2.1	Script Tangle	56
7.2.2	Script utilizzato da onmouseover	62
7.2.3	Body	63
7.2.4	generalrepo	63
7.2.5	singlerepo	67
7.2.6	Canvas per la creazioni dei grafici	69
<b>8</b>	<b>Specifica tecnica</b>	<b>71</b>
8.1	stile	71
8.2	Tangle.js	71
8.3	mootools.js	71
8.4	TangleKit.js	71
8.5	BVTouchable.js	71
8.6	sprintf.js	71
8.7	MyScriptTangle.js	71
8.8	MyScriptMouse.js	71
8.8.1	body	72
<b>9</b>	<b>Conclusioni</b>	<b>73</b>
9.1	Prodotto finale	73
9.2	Confronto con l'azienda	73
9.3	Conoscenze pregresse	73
<b>A</b>	<b>Glossario</b>	<b>75</b>
<b>B</b>	<b>Riferimenti Bibliografici</b>	<b>81</b>

## Elenco delle figure

1	logo Zucchetti spa . . . . .	5
2	Pagina Web . . . . .	7
3	Rappresentazioni ottenibili con Canvas e tangle . . . . .	8
4	Diagramma di Grantt . . . . .	11
5	Sistema di riferimento per le misure sul Canvas . . . . .	15
6	Prototipo numero 1 . . . . .	21
7	canvas . . . . .	23
8	grafico utenti . . . . .	23
9	Tangle.PlotUtenti . . . . .	25
10	Elemento esplorabile . . . . .	26
11	Prototipo numero 3 - i primi tre grafici . . . . .	29
12	plot ottenuto con il Canvas . . . . .	31
13	index.html-pagina 2 . . . . .	32
14	interpolazione lineare . . . . .	34
15	index.html-pagina 3 . . . . .	37
16	Prototipo numero 4 - i primi tre grafici . . . . .	39
17	Files requests . . . . .	42
18	Prototipo pagina web finale . . . . .	43
19	Scenari sull'andamento generale e particolare . . . . .	44
20	Scenari sull'andamento generale e particolare . . . . .	44
21	Pagina Web . . . . .	45
22	Diagramma generale dei casi d'uso . . . . .	47
23	Diagramma del caso d'uso 1 . . . . .	49
24	Diagramma del caso d'uso 2 . . . . .	50
25	Diagramma del caso d'uso 3 . . . . .	51
26	Diagramma del caso d'uso 4 . . . . .	53
27	Diagramma del caso d'uso 5 . . . . .	54
28	Diagramma del caso d'uso 6 . . . . .	55
29	Rendering del Canvas . . . . .	62
30	Report generale . . . . .	63
31	Mouse onmouseoverout . . . . .	67
32	Mouse mousemove ma non selezione le barre . . . . .	68
33	Report generale . . . . .	69

## 1 Azienda



Figura 1: logo Zucchetti spa

Zucchetti è un'azienda italiana che produce soluzioni software, hardware e servizi per aziende, banche, assicurazioni, professionisti e associazioni di categoria per quanto riguarda:

- attività gestionali;
- gestione documentale (DMS);
- gestione del personale e delle risorse umane in aziende di grandi dimensioni;
- conservazione sostitutiva;
- rilevazione presenze, controllo accessi e videosorveglianza (hardware e software);

In aggiunta, l'azienda produce anche servizi complementari per:

- sviluppo progetti – system integration;
- soluzioni per la gestione privacy;
- modulistica e prodotti per l'informatica e l'ufficio;
- robotica e automazione.

Le origini del gruppo risalgono all'omonimo studio commercialista che nel 1978 realizza il primo programma in Italia per l'elaborazione in automatico delle dichiarazioni dei redditi.

Una rete distributiva che supera i 1.000 Partner sull'intero territorio nazionale e oltre 85.000 clienti con oltre 2.100 addetti, il gruppo Zucchetti è uno dei più importanti protagonisti italiani del settore dell'IT.

L'azienda opera con:

- aziende di qualsiasi settore e dimensione, banche e assicurazioni;
- professionisti, associazioni di categoria;
- pubblica amministrazione locale e centrale (Comuni, Province, Regioni, Ministeri, società pubbliche ecc.).

Attraverso personale altamente qualificato e preparato, il gruppo Zucchetti garantisce:

- un competente supporto pre-vendita (analisi delle esigenze, studio della soluzione ecc.);
- un tempestivo e valido servizio post-vendita (installazione, assistenza ecc.);
- una puntuale formazione per sfruttare al meglio tutte le potenzialità delle proprie soluzioni;

- un costante aggiornamento su tematiche civilistiche, contabili, fiscali, previdenziali ecc.

Zucchetti è anche sinonimo di sicurezza. Attraverso una Server Farm di proprietà, infatti, assicura elevati standard di sicurezza informatica, logica (sistemi antintrusione), applicativa e fisica (sistemi antincendio, controllo accessi ecc.), nonché la continuità dei servizi di outsourcing in tutte le condizioni.

I partner Zucchetti possiedono qualità e competenze specifiche per realizzare soluzioni mirate ad ogni esigenza del cliente, sviluppando così una relazione di valore che dura, si sviluppa e cresce nel tempo.

A Padova è presente la Zucchetti SPA - Research e Development Area, dove ho effettuato lo stage. La Zucchetti promuove nel corso dell'anno molteplici occasioni d'incontro con il pubblico come convegni orientati alla presentazione di nuove soluzioni, iniziative organizzate da società esterne a cui la Zucchetti partecipa, fiere e manifestazioni a livello sia nazionale che locale, momenti d'incontro e di confronto con il mondo della scuola e delle università.

## 2 Il progetto

### 2.1 Proposta di stage

L'azienda possiede una Proxy, che cattura le prestazioni di un'applicazione software. Lo stage si propone di realizzare una pagina web dinamica, che elaborando i dati provenienti dal Proxy, riporti una serie di rappresentazioni grafiche che mostrino l'andamento corrente di tali prestazioni.

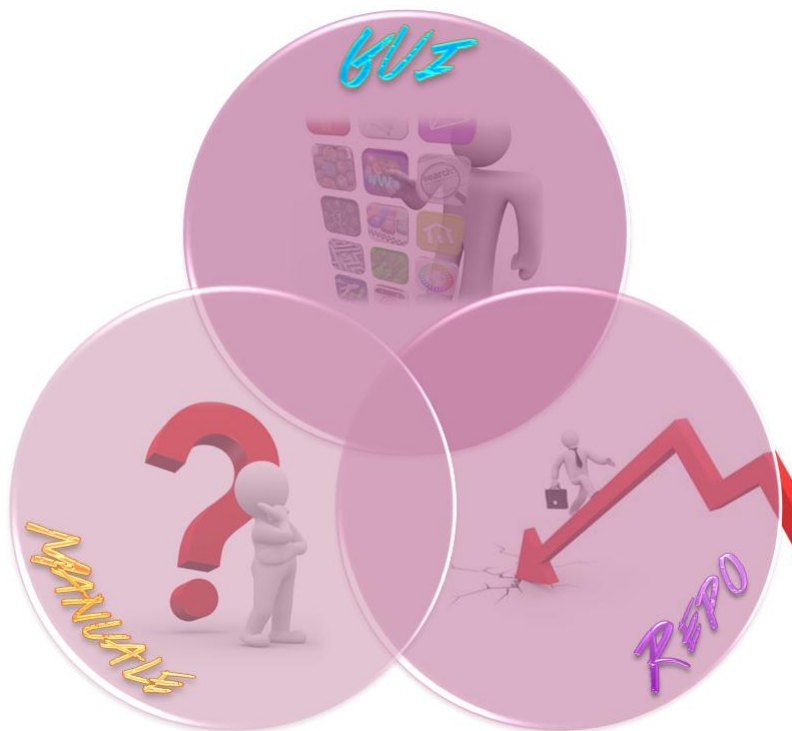


Figura 2: Pagina Web

Si richiede che l'utente sia aiutato, attraverso una spiegazione in linguaggio naturale, a comprendere:

- in che situazione si trova l'applicazione web;
- punti in cui si verificano cali delle prestazioni;
- scenari importanti e valori inaspettati;

Questa pagina web si deve porre a metà strada tra un'interfaccia utente, un manuale e un report di tali informazioni.

Inoltre deve essere possibile regolare alcuni parametri per cambiare la modalità di visualizzazione.



### 2.2 Ideazione e filosofia d'azione

Il Proxy specializzato cattura dati relativi alle performance di una pagina web. Tali dati sono agglomerati di dati spesso complessi dai quali è difficile far emergere informazioni utili. Si ha, quindi, la necessità di elaborare questi dati e visualizzarli attraverso un'interfaccia utente che permetta a utilizzatori occasionali di relazionarsi con efficacia con tali procedure complesse.

A tal proposito è stato chiesto di esplorare le possibilità fornite da Tangle, una libreria JavaScript che prometteva un nuovo approccio al problema. Tangle, e in particolare il suo ideatore Bret Victor, usa un differente modo di interazione alle pagine web. L'obiettivo è quello di cambiare il rapporto della gente con il testo. La gente vede il testo come informazioni da consumare, lui vuole che i lettori diventino attivi.

I lettori non dovrebbero semplicemente acquisire le nozioni statiche scritte nelle pagine, ma dovrebbero generalizzare esempi specifici e elaborare esempi di tali generalità. Si vuole che i lettori sviluppino una lettura critica dei documenti che li porti a una comprensione profonda dell'argomento trattato. Questa libreria che permette la creazione di documenti *reattivi*. Un documento reattivo permette al lettore di giocare con le ipotesi dell'autore e vedere le conseguenze aggiornate immediatamente, sviluppando un'intuizione di come un sistema funziona.

Il lettore può esplorare scenari alternativi più fiducioso della bontà delle conclusioni. Questo si potrebbe rendere con fogli di calcolo, ma si tratta sempre di modelli matematici e agglomerati di dati spesso di difficile comprensione, dove la barriera dell'esplorazione è alta.

Con questa nuova idea la spiegazione, che fruisce attraverso linguaggio naturale e la grafica, cambia a secondo degli scenari, accompagnando in modo user-friendly tutti gli utenti che lo desiderano nella propria esplorazione con un semplice click o trascinamento.

Il documento così può essere letto a più livelli, a seconda del grado di interesse del lettore.



Figura 3: Rappresentazioni ottenibili con Canvas e tangle

Si presta bene a una lettura veloce, dove il lettore occasionale può leggere il testo così com'è. Il lettore curioso, invece può regolare vari parametri ed esplorare nuove situazioni.

L'utente è aiutato dall'uso di molteplici rappresentazioni come istogrammi, areogrammi, plot di funzioni ma anche schemi attivi e calcoli intermedi auto aggiornabili. Il lettore è attratto dalla novità del widget interattivo e della sottigliezza con cui questo è integrato con la spiegazione. La Fig.3 mostra alcune delle rappresentazioni ottenibili. Con queste attrattive si cerca di rendere il semplice lettore, un lettore attivo!

### 3 Lo stage

Prima di tutto è stata fatta la pianificazione del lavoro, cioè l'impostazione del lavoro in termini di attività da svolgere, orari e vincoli.

Vengono spiegate le idee iniziali che si intende giungere e mostrato il diagramma di Gantt con le attività preventivate.

#### 3.1 Piano di Lavoro

Prima di iniziare lo stage è stato redatto un piano di lavoro con l'aiuto del tutor esterno. Tale documento ha permesso di tracciare i principali obiettivi con cadenza settimanale. Come precedentemente descritto ogni settimana ha dei prodotti in uscita che segnano l'avvenuto adempimento delle scadenze o i ritardi.

L'unione di tutte le attività deve consentire di terminare il progetto proposto dall'azienda. Gli obiettivi settimanali sono:

- l'acquisizione di tecniche e strumenti utili;
- un prodotto prototipale;
- delle funzionalità aggiuntive;
- una documentazione riguardante uno o più dei precedenti punti.

La durata delle attività è bilanciata in modo che la loro somma non superi le 320 ore. Dopo aver suddiviso il progetto nelle attività principali, viene inserito anche il diagramma di Gantt, che ha permesso di inserire dipendenze fra le attività e di distribuirle nell'arco temporale.

#### 3.2 Le attività preventivate

Il piano di lavoro ha preventivato queste attività settimanali:

- **Settimana 1** Viene dedicata allo studio di Tangle, una libreria JavaScript che permette di creare documenti reattivi. Questa attività è fondamentale allo sviluppo delle attività successive.
- **Settimana 2** Si intende realizzare tre pagine web statiche in HTML, che contengono parti interattive con fonti dati interne.
- **Settimana 3** Si intende realizzare pagine web dinamiche in JavaScript, che sono interattive con fonti dati interne.
- **Settimana 4** Si intende realizzare pagine statiche e dinamiche che accedono a fonti dati esterne via XML, Http e Java. Inoltre viene redatto in uscita un documento che illustri come la libreria utilizzata sia stata impiegata nelle pagine sviluppate fino a questa fase.
- **Settimana 5** Viene condotto uno studio su un documento interattivo che presenta le informazioni contenute nei log di un Proxy, il quale testa le prestazioni di un'applicazione Web.
- **Settimana 6** Si intende realizzare un documento interattivo che presenti in maniera user-friendly i dati provenienti dal Proxy.

- **Settimana 7** Viene effettuata un'attività di verifica su tutti i documenti in uscita e sul codice interno alla pagine.
- **Settimana 8** E' prevista la stesura della documentazione riguardante il funzionamento della pagina interattiva che presenta i dati del Proxy. Vengono inoltre esposti i risultati che emergono dalla fase di test e verifica della settimana settimane e il quadro riassuntivo delle attività effettuate durante lo stage.

### 3.3 Fasi di sviluppo effettive

Attraverso una serie di prototipi<sup>[g]</sup> sono emerse le situazioni che si vogliono modellare e che scenari possono mostrare queste situazioni. Il rilascio dei prototipi è stato utile perché l'uso delle interfacce grafiche ha permesso di esaminare le modalità nelle quali dovevano essere modificati alcuni parametri, e scegliere quella più user-friendly.

### 3.4 Diagramma

Il diagramma di Gantt è utile nelle attività di management di un progetto. Esso è costruito partendo da un asse orizzontale - a rappresentazione dell'arco temporale totale del progetto, suddiviso in fasi incrementali (ad esempio, giorni, settimane, mesi) - e da un asse verticale - a rappresentazione delle mansioni o attività che costituiscono il progetto.

Barre orizzontali di lunghezza variabile rappresentano le sequenze, la durata e l'arco temporale di ogni singola attività del progetto (l'insieme di tutte le attività del progetto ne costituisce la Work Breakdown Structure). Queste barre possono sovrapporsi durante il medesimo arco temporale ad indicare la possibilità dello svolgimento in parallelo di alcune delle attività.

Man mano che il progetto progredisce, delle barre secondarie, delle frecce o delle barre colorate possono essere aggiunte al diagramma, per indicare le attività sottostanti completate o una porzione completata di queste. Una linea verticale è utilizzata per indicare la data di riferimento.

Un diagramma di Gantt permette dunque la rappresentazione grafica di un calendario di attività, utile al fine di pianificare, coordinare e tracciare specifiche attività in un progetto dando una chiara illustrazione dello stato d'avanzamento del progetto rappresentato; di contro, uno degli aspetti non tenuti in considerazione in questo tipo di diagrammazione è l'interdipendenza delle attività, caratteristica invece della programmazione reticolare, cioè del diagramma PERT. Ad ogni attività possono essere in generale associati una serie di attributi: durata (o data di inizio e fine), predecessori, risorsa, costo.

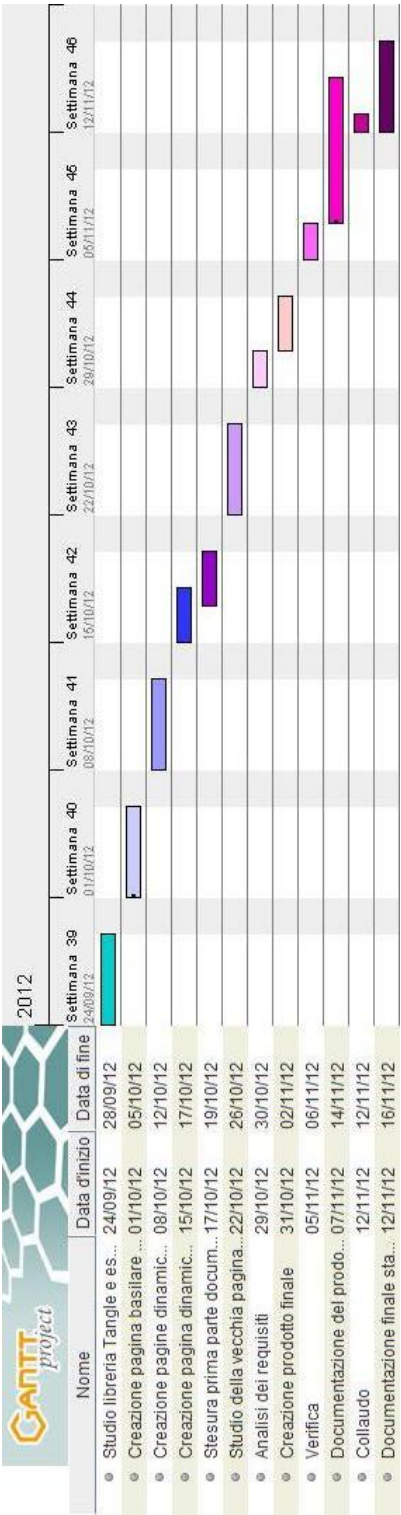


Figura 4: Diagramma di Grantt

## 4 Premesse architetture

In questa sezione vengono descritte le maggiori scelte architetture. Verranno spiegate le funzionalità dei componenti che ci serviranno ai fini del progetto. In alcuni casi si scenderà a una spiegazione più in dettaglio delle funzionalità allo scopo di alleggerire e rendere scorrevole la trattazione relativa al prodotto finale.

Le architetture utilizzate sono in parte dettate esplicitamente dalle richieste dell'azienda, in parte legate indirettamente a queste ultime.

### 4.1 HTML

HTML, **H**yper**T**ext **M**arkup **L**anguage è il linguaggio usato per i documenti ipertestuali disponibili nel web.

L'HTML non è un linguaggio di programmazione ma solamente un linguaggio di markup che descrive le modalità di impaginazione, formattazione o visualizzazione grafica (layout) del contenuto, testuale e non, di una pagina web attraverso tag di formattazione.

L'HTML supporta l'inserimento di script e oggetti esterni quali immagini o filmati. E' stato concepito per definire il contenuto logico e non l'aspetto finale del documento.

In particolare noi utilizzeremo l'HTML5. Le novità introdotte dall'HTML5 sono finalizzate soprattutto a migliorare il disaccoppiamento tra struttura, definita dal markup, caratteristiche di resa (tipo di carattere, colori, ecc), definite dalle direttive di stile, e contenuti di una pagina web, definiti dal testo vero e proprio.

Inoltre l'HTML5 prevede il supporto per la memorizzazione locale di grosse quantità di dati scaricati dal browser, per consentire l'utilizzo di applicazioni basate su web (come per esempio le caselle di posta di Google o altri servizi analoghi) anche in assenza di collegamento a Internet.

In particolare:

- sono fissate in modo rigido le regole per la struttura del testo in capitoli, paragrafi e sezioni;
- sono resi dei controlli per i menu di navigazione;
- vengono deprecati o eliminati alcuni elementi che hanno dimostrato scarso o nessun utilizzo effettivo;
- vengono estesi a tutti i tag una famiglia di attributi, finalizzati all'accessibilità, finora previsti solo per alcuni tag;
- è introdotta la geo localizzazione, dovuta ad una forte espansione di sistemi operativi mobili sistema alternativo ai normali cookie, chiamato Web Storage, più efficiente, il quale consente un notevole risparmio di banda;
- possibilità di utilizzare alcuni siti offline;
- sono introdotti elementi specifici per la resa di contenuti multimediali (tag video e audio); infine viene introdotto il tag Canvas che permette di utilizzare JavaScript per creare grafica, animazioni e bitmap.

La novità introdotta da questo standard, ai fini da noi richiesti, è il supporto dei Canvas.

### 4.2 DOM

Il **Document Object Model** (abbreviato DOM) è un'astrazione della struttura dei documenti come modello orientato agli oggetti. E' un standard W3C (World Wide Web Consortium).

DOM permette di visualizzare un documento ben formato sotto forma di albero. E' una piattaforma e un'interfaccia indipendente dal linguaggio che consente ai programmi e script di accedere e aggiornare dinamicamente il contenuto, la struttura e lo stile di un documento.

Il DOM definisce gli oggetti, le proprietà di tutti gli elementi del documento e le modalità (interfaccia) per accedervi.

Tutti i nodi sono accessibili attraverso l'albero. Il loro contenuto può essere modificato o cancellato, e nuovi elementi possono essere creati.

L'albero inizia dal nodo radice e termina verso i nodi di testo al livello più basso della struttura.

Il DOM XML fornisce i metodi (funzioni) per attraversamento dell'albero XML e per l'accesso, inserimento ed eliminazione dei nodi. Tuttavia, prima di accedere e manipolare un documento XML deve essere caricato in un oggetto XML DOM.

Un parser XML legge l'XML e lo converte in un oggetto XML DOM:

```
if (window.XMLHttpRequest)
{
    xhttp=new XMLHttpRequest();
}
else // IE 5/6
{
    xhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xhttp.open("GET","books.xml",false);
xhttp.send();
xmlDoc=xhttp.responseXML;
```

Esso:

- crea un oggetto XMLHTTP;
- apre l'oggetto XMLHTTP;
- invia una richiesta XML HTTP al server;
- imposta la risposta come un oggetto XML DOM;

Per motivi di sicurezza, i browser moderni non consentono l'accesso tra domini. Ciò significa, che sia la pagina web che il file XML che si vuole caricare, deve trovarsi sullo stesso server.

Con la funzione *getElementsByTagName()* si ricava un elenco dei nodi con il nome del tag specificato, nello stesso ordine in cui appaiono nel documento di origine.

L'oggetto *XMLHttpRequest* viene utilizzato per lo scambio di dati con un server dietro le quinte.

Ha un vantaggio notevole, in quanto è possibile:

- aggiornare una pagina web senza ricaricarla;
- richiedere dati al server dopo che la pagina è stata caricata;
- ricevere dati da un server dopo che la pagina è stata caricata;
- inviare i dati a un server in background;

### 4.3 Canvas

Canvas è una estensione dell' HTML standard che permette la resa dinamica di immagini gestibili attraverso un linguaggio di scripting.

Per creare un Canvas basta inserire nel codice HTML:

```
<canvas id="prova" width="150" height="150"></canvas>
```

Il Canvas consiste in una regione disegnabile dotata di due attributi: *width* e *height*; entrambi non sono obbligatori e possono essere impostati tramite DOM.

Il codice JavaScript può accedere all'area con delle funzioni per il disegno. Alcuni usi possibili di Canvas includono grafici, animazione, composizioni fotografiche ecc. E' buona norma fornire un id, perché questo rende molto più facile identificazione da parte dello script.

Con:

```
var canvas = document.getElementById( 'prova' );
```

recuperiamo il nodo DOM del Canvas . L'istruzione:

```
var ctx = canvas.getContext( '2d' );
```

crea una superficie di disegno di dimensioni fissate in uno spazio bidimensionale.

Il Canvas è dotato di una griglia di coordinate. Una unità nella griglia corrisponde a 1 pixel sulla tela. L'origine di questa griglia è posizionato in alto a sinistra (coordinate (0,0)). Relativamente a questa origine vengono posizionati tutti gli elementi.

Tutte le forme sono realizzate a partire da linee e rettangoli. Per creare un rettangolo si utilizza la seguente funzione:

```
fillRect (x,y,width,height)
```

Per creare un percorso

```
ctx.beginPath ();  
ctx.moveTo (75,50);  
ctx.lineTo (100,75);  
ctx.lineTo (100,150);  
ctx.fill ();
```

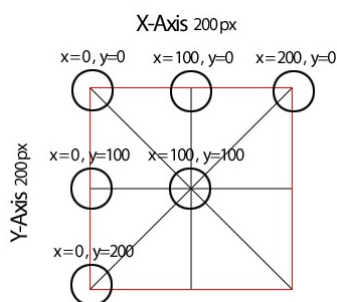


Figura 5: Sistema di riferimento per le misure sul Canvas

I percorsi vengono creati come una lista di sotto-percorsi (linee, archi, ecc) che vengono unite insieme con `fill()` e creano una forma.

Per applicare i colori a una forma, rendendole piene ci sono due importanti proprietà che possiamo usare: *FillStyle* e *StrokeStyle*. Il *StrokeStyle* imposta il colore del contorno forma e *FillStyle* è per il colore di riempimento.

Per impostare lo spessore di linea corrente si usa:

```
ctx.lineWidth=18;
```

I valori devono essere numeri positivi. Di default questo valore è impostato a 1,0 unità.

### 4.4 CSS

Il CSS (**C**ascading **S**tyle **S**heets o Fogli di stile) è un linguaggio usato per definire la formattazione di documenti HTML.

L'introduzione del CSS si è resa necessaria per separare i contenuti dalla formattazione e permettere una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine HTML che per gli utenti.

Inoltre avendo dei fogli di stile separati, a seconda del device nel quale viene visualizzata la pagina può essere impostato un foglio di stile specifico, per permettere migliore usabilità, efficienza e resa grafica.

Il supporto completo e corretto delle specifiche CSS non è offerto da nessun browser attuale. Tuttavia esistono browser che si avvicinano molto a questo risultato ed altri che invece ne sono molto lontani

### 4.5 JavaScript

JavaScript è un linguaggio di scripting orientato agli oggetti comunemente usato nei siti web

Ha in sé le funzionalità tipiche dei linguaggi di programmazione ad alto livello (strutture di controllo, cicli, ecc.) e consente l'utilizzo dell'object oriented.

Un aspetto importante è che il codice viene eseguito direttamente sul client e non sul server. Il vantaggio è che, anche con la presenza di script particolarmente complessi, il server



non viene sovraccaricato a causa delle richieste dei client. JavaScript viene utilizzato soprattutto in quanto linguaggio di scripting all'interno di un altro programma.

L'esempio tipico di programma ospite per uno script JavaScript è quello del browser. Quando viene visitata una pagina web che contiene il codice di uno script JavaScript, viene caricato in memoria lo script ed eseguito dall'interprete contenuto nel browser. Le interfacce che consentono a JavaScript di rapportarsi con un browser sono chiamate DOM (Document Object Model).

Molti siti web usano la tecnologia JavaScript lato client per creare potenti applicazioni web dinamiche con l'uso dei DOM.

Con il DOM, JavaScript riesce a creare HTML dinamico che:

- può modificare tutti gli elementi, attributi e stili CSS HTML della pagina;
- è in grado di reagire a tutti gli eventi della pagina;
- può controllare i valori nei campi di input, nascondere o visualizzare determinati elementi, rimpiazzare delle immagini, adattare valori di layout ecc. .

Un semplice esempio per prendere il nodo e modificarne il contenuto è:

```
<p id="prova">First.</p>
<script>
function myFunction()
{
  x=document.getElementById("prova");
  x.innerHTML="Second";
}
</script>
```

Le funzioni JavaScript sono spesso eseguite in risposta ad eventi sollevati da immagini, cursori, e molte altre parti. Un evento è qualcosa che accade nel documento, per esempio il mouse è sopra un oggetto o si fa click oppure si è usciti da un determinata area. E' possibile collegare un evento a una funzione JavaScript e in questo modo cambiare CSS, abilitare nuovi eventi, validare un input ecc.

Un esempio di codice potrebbe essere:

```
<img src = "normal.png"
      onclick      = "myJavascriptFunction()"
      onmouseover  = "this.src='rollover.png'"
      onmouseout   = "this.src='normal.png'"
/>
```

Gli script possono essere posizionati nell'header, nel body o in file esterni.

L'utilizzo dei file esterni è stato incentivato per aumentare l'utilizzo dello script da parte di pagine web diverse. I file JavaScript esterni hanno l'estensione .js .

Una variabile dichiarata (con *var*) all'interno di una funzione JavaScript è locale ed il suo scope è definito solo all'interno di tale funzione. Si possono avere variabili locali con lo stesso nome in diverse funzioni, perché le variabili locali vengono riconosciuti solo dalla funzione in cui sono dichiarate. Le variabili dichiarate fuori da una funzione, diventano globali e tutti

gli script e le funzioni possono accedervi.

L'operatore `+` può essere utilizzato anche per concatenare variabili stringa o valori di testo insieme. Ad esempio:

```
txt1="pippo ";  
txt2="pluto ";  
txt3=txt1+txt2;
```

Il risultato di `txt3` sarà: *pippopluto*.

### 4.6 Mootools

MooTools è un framework JavaScript con una sintassi orientata agli oggetti. Un framework JavaScript mette insieme oggetti, funzionalità e metodi sotto la stessa sintassi per unificare in uno o più script molte funzionalità per permettere la creazione di applicazioni di nuova generazione, come effetti ed animazioni.

Uno dei grandi vantaggi che si ottengono nell'utilizzare un framework è che la resa non dipende dal browser in uso. Browsers differenti hanno infatti modalità di resa grafica e gestione del DOM e degli eventi molto differenti tra loro.

All'interno di MooTools, è possibile creare richieste Ajax complesse, creare richieste in formato JSON, creare animazioni multi-funzione, settare o modificare i cookie, lavorare con gli stili, le proprietà e le dimensioni degli elementi, utilizzare i selettori CSS.

Mootools è il framework che sta alla base della libreria *Tangle*, sul quale si basa il nostro progetto. Si è ritenuto utile ai fini della spiegazione delle pagine darne un breve cenno.

Un metodo che troveremo sarà:

```
addEventListener(event, function())
```

esso aggiunge all'oggetto d'invocazione un evento e lo abbina a una funzione. Esempio:

```
window.addEventListener('domready', function() {...});
```

Questa parte di codice dice di ridefinire l'evento *domready* e di abbinarlo alla funzione definita nel secondo parametro. L'evento *domready* viene sollevato quando il DOM è stato caricato.

### 4.7 Tangle

Tangle è una libreria JavaScript per la creazione di documenti reattivi. Si dà la possibilità di esplorare la pagina in modo interattivo, giocando con i parametri e vedendo immediatamente il documento aggiornato.

È sufficiente scrivere un documento con HTML e CSS e usare speciali attributi HTML per indicare le variabili e un file JavaScript per specificare come le variabili vengono calcolate. Tangle collega tutti i dispositivi.

Ai fini di rendere più scorrevole la spiegazione delle singole pagine, diamo qui una breve trattazione della sua struttura principale.

### 4.7.1 Inizializzazione

Per iniziare, bisogna creare un oggetto `tangle`:

```
var tangle = new Tangle(document, {  
    initialize: function () { },  
    update:    function () { }  
});
```

i due parametri sono rispettivamente, il contenitore dove agisce `tangle`, spesso una parte di documento selezionata con il DOM, e le funzioni che questo `tangle` deve implementare.

Indispensabili sono l'*initialize*, nel quale bisogna inserire tutti i parametri che al caricamento della pagina devono essere già settati, e l'*update* che verrà eseguito ogni volta che verrà modificato un oggetto `tangle`.

L'*update* serve per contenere tutte le funzioni che portano il dato ad aggiornarsi, di conseguenza al cambiamento, per esempio, di alcuni parametri di base.

### 4.7.2 Classi `tangle` standard e `data-var`

Per aggiungere un oggetto di una classe bisogna mettere nel file HTML:

```
<span data-var="cookies" class="TKAdjustableNumber"> cookie </span>
```

L'attributo *data-var* specifica una variabile. In genere si utilizza per visualizzare il valore della variabile o per specificare quale variabile di una classe dovrebbe regolare.

Se il codice HTML contiene:

```
<span class = "MyClass" data-var = "cookies">
```

quando `tangle` viene creato, `MyClass` sarà istanziata per tale elemento. Se vi è un metodo *initialize*, verrà chiamato e gli verrà passato il nome della variabile. Se c'è un metodo di aggiornamento, verrà chiamato quando la variabile specificata viene inizializzata o cambiata.

L'attributo *class* viene usato per fornisce il comportamento di un elemento. Un certo numero di classi di base sono inclusi con `TangleKit`, ma molte altre se ne possono creare.

Si descrivono qui i principali:

- **TKSwitch** se il valore assunto da *data-var* è *n* mostra il figlio *n*-esimo. I valori *true* o *false* indicano rispettivamente il primo e il secondo figlio.

Esempio:

```
<span data-var="index" class="TKSwitch">  
    <span>Mostrato se index e' pari a zero.</span>  
    <span>Mostrato se index e' pari a uno.</span>  
    <span>Mostrato se index e' pari a due.</span>  
</span>
```

- **TKIf** Se *data-var* ha valore false, nasconde il suo contenuto, altrimenti lo mostra.
- **TKSwitchPositiveNegative** mostra il primo figlio dell'elemento se il valore è positivo o pari a zero. Mostra il secondo figlio dell'elemento se il valore è negativo.
- **TKToggle** bisogna fare clic per scegliere un valore compreso tra 0 e 1.
- **TKAdjustableNumber** trascinando un widget, si può settare a un nuovo valore.  
*Attributi: data-min (opzionale): valore minimo, data-max (opzionale): valore massimo, data-step (opzionale): granularità di regolazione.*

### 4.7.3 Nuove classi

```
Tangle.classes.MyClass = {  
  initialize: function (element, opzioni, tangle, variabile) {... },  
  update: function (element, valore) {... }  
};
```

La creazione di una nuova classe tangle come si vede nell'esempio precedente è facile ed intuitiva.

La definizione di nuove classi è molto utile per definire un comando che regola una variabile o una vista che mostra una variabile ma anche per usi più creativi.

### 4.7.4 Funzionalità utili

```
var value = tangle.getValue(variable);
```

Il metodo *getValue(variable)* ritorna il valore corrente della variabile in entrata come parametro.

E' molto utile quando si stanno creando nuove classi e si ha la necessità di acquisire i dati provenienti da un tangle originario.

## 4.8 Ajax

AJAX, acronimo di **A**synchronous **J**avaScript and **X**ML, è una tecnica di sviluppo per la realizzazione di applicazioni web interattive.

Essa permette lo scambio di dati in background fra browser e server, che consente l'aggiornamento dinamico di una pagina web senza nessuna interazione da parte dell'utente.

La tecnica Ajax utilizza una combinazione di:

- HTML e CSS per la struttura e la formattazione;
- DOM manipolato con JavaScript per interagire e modificare la struttura;
- l'oggetto XMLHttpRequest per l'interscambio asincrono dei dati tra il browser dell'utente e il web server;
- JSON come formato di interscambio dei dati.

### 4.9 JSON

JSON, acronimo di **J**ava**S**cript **O**bject **N**otation, è un formato adatto per lo scambio dei dati in applicazioni client-server.

Viene usato in AJAX come alternativa a XML.

L'uso di JSON è molto semplice, infatti l'interprete è in grado di eseguirne il parsing tramite una semplice chiamata alla funzione eval().

Un esempio di JSON:

```
{
  "type": "menu",\\stringa
  "value": "4",\\numero
  "items": [
    {"value": "New", "action": "CreateNewDoc"},
    {"value": "Open", "action": "OpenDoc"},
    {"value": "Close", "action": "CloseDoc"}
  ]\\array
}
```

### 4.10 Tomcat

Tomcat è un contenitore servlet open source sviluppato dalla Apache Software Foundation. Fornisce una piattaforma per l'esecuzione di applicazioni Web sviluppate nel linguaggio Java.

La sua distribuzione standard include anche le funzionalità di web server tradizionale, che corrispondono al prodotto Apache.

Per fare eseguire le pagine basterà metterle nella cartella

tomcat\Tomcat 5.5\webapps\ROOT

e visualizzare la pagina all'indirizzo

localhost:8080

## 5 Sviluppo Prototipi

### 5.1 Pagina 1

#### 5.1.1 Attività

Inizialmente è stato dedicato del tempo allo studio di Tangle, la libreria JavaScript proposta dall'azienda che permette di creare documenti reattivi.

Si è partiti con lo studio di TangleKit.js, una collezione di componenti dell'interfaccia utente che permette di regolare i valori e visualizzare i risultati. Essa include anche alcune librerie utili come Sprintf, BVTouchable e MooTools, un framework Javascript.

Infine sono stati presi in esame gli esempi offerti dal sito dell'ideatore di Tangle.

Una volta acquisiti i concetti fondamentali si è passati alla realizzazione della prima versione della pagina web.

La pagina fornisce un modo per regolare il numero di utenti presenti nell'applicazione. Ogni volta che regoliamo questo numero si deve visualizzare nell'apposito diagramma a barre il dato aggiornato (vedi Fig.4).

Per questa fase di prototipazione i dati che compaiono sui vari grafici sono forniti da una funzione-simulazione.

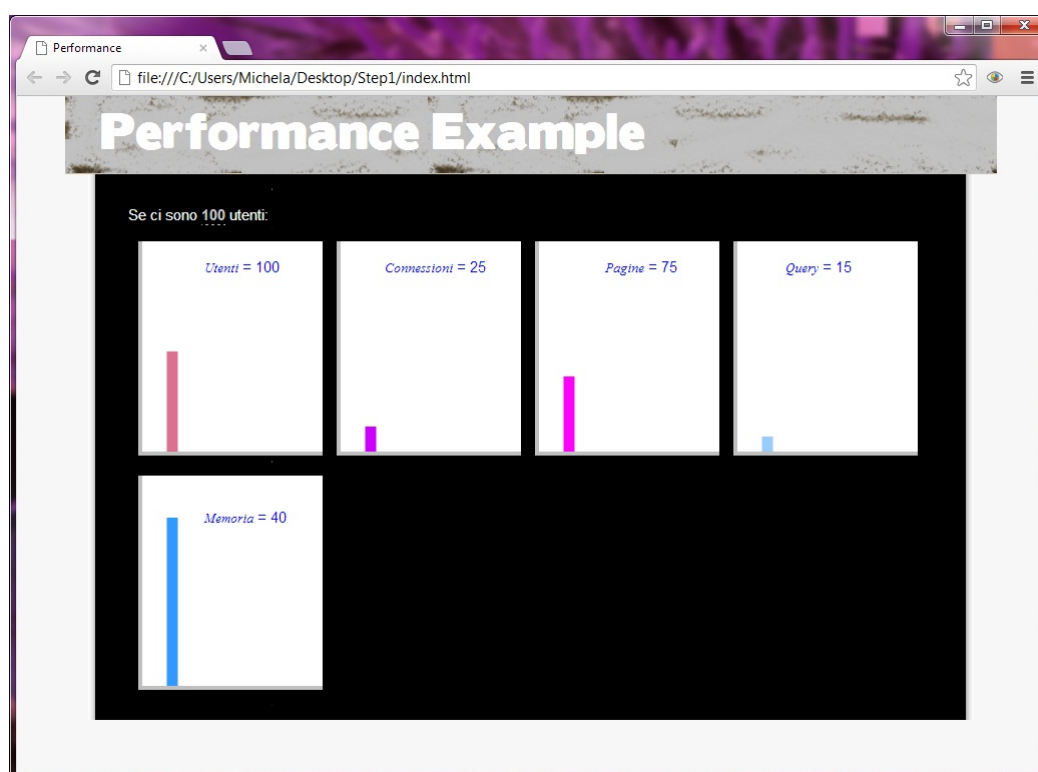


Figura 6: Prototipo numero 1

Sono stati creati cinque grafici:

- il primo rappresenta graficamente il numero di utenti presenti nella pagina; dato che può essere modificato e che si può visualizzare aggiornato.

- il secondo, che mostra il numero di connessioni al data base. Esse vengono calcolate come un quarto degli utenti presenti.
- nel terzo visualizza il numero di pagine servite. Tale numero è proporzionale al numero di utenti e si è deciso di definirle come tre volte il numero di connessioni.
- il quarto grafico raffigura le query servite in quel momento. Si è pensato di approssimare questo valore a un quinto rispetto al numero di pagine.
- l'ultimo rende visibile l'andamento degli accessi simultanei alla memoria. Sono possibili 50 accessi alla memoria contemporanei e questi accessi vengono richiesti sia dalla pagine che dalle query.

Le funzioni per ricavare i dati sono momentanee e a solo scopo di testare le proprie capacità e conoscenze nell'utilizzo di Tangle.

### 5.1.2 Pagina 1-performance.js

Come visto nella sez 4.6 al momento del caricamento del DOM, verrà eseguita la funzione, che compare come secondo parametro ad *addEventListener*.

La funzione non fa altro che identificare attraverso *getElementById()* i nodi interessati dalla modifica e creare un oggetto tangle che ha come collocazione i nodi selezionati precedentemente e come funzione l'inizializzazione della classe. Nell'inizialize vengono inizializzati i valori dei campi, nell'update vengono inserite le funzioni per aggiornare i campi.

Abbiamo usato delle funzioni semplici che potevano dare l'idea di un sistema che misura le prestazioni di un'applicazione web. In questa fase verranno calcolati i valori in una singola istantanea. Non si ha quindi la rappresentazione dello scorrere del tempo.

```
window.addEventListener('domready', function () {
    var container = document.getElementById("Performance");
    var tangle = new Tangle(container, {
        initialize: function () {
            this.utenti=100;
            this.conessioni=25;
            this.pagine=75;
            this.query=15;
            this.memoria=30;
        },
        update: function () {
            this.conessioni=Math.floor(this.utenti*(1/4));
            this.pagine=Math.floor(this.conessioni*3);
            this.query=Math.floor(this.pagine*1/5);
            var parziale=this.pagine + this.query
            this.memoria= parziale%50;
        },
    });
});
```

Nello stesso file aggiungo a Tangle una nuova classe. Questa nuova classe, *PlotUtenti*, rappresenta un oggetto tangle che rende graficamente il valore di *this.utenti* (vedi Fig.8).

```
Tangle.classes.PlotUtenti = {
    initialize: function (el, options, tangle) {
```

```
        this.tangle = tangle;
    },
    update: function (el, utenti) {
        var canvasWidth = el.get("width");
        var canvasHeight = el.get("height");
        var ctx = el.getContext("2d");

        ctx.fillStyle = "#fff";
        ctx.fillRect(0, 0, canvasWidth, canvasHeight);

        ctx.strokeStyle = "#DB7093";
        ctx.lineWidth = 11;
        ctx.beginPath();
        ctx.moveTo(30, canvasHeight)
        ctx.lineTo(30, canvasHeight - utenti);
        ctx.stroke();
    },
};
```

In particolare l'update ha come parametro il Canvas e il valore di utenti. L'oggetto preleva le dimensioni del Canvas, crea un ambiente grafico bidimensionale e disegna un rettangolo contenitivo delle dimensioni del Canvas di colore bianco.

Il Canvas utilizza una metrica in pixel che parte dal bordo in alto a sinistra (vedi Fig.7).

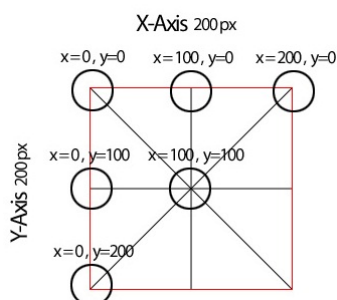


Figura 7: canvas

La barra che indica il valore degli utenti deve iniziare dal punto(30, canvasHeight) e finire nel punto (30, canvasHeight - utenti).

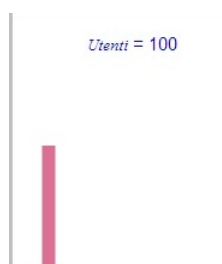


Figura 8: grafico utenti



Sono state aggiunte altre quattro classi quali PlotConessioni, PlotPagine, PlotQuery, PlotMemoria per rappresentare i grafici delle connessioni attive, delle pagine servite, delle query eseguite e degli accessi in memoria.

### 5.1.3 Pagina 1-index.html

Nell'head di questa pagina

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Performance</title>

  <link rel="stylesheet" href="style.css" type="text/css">

  <script type="text/javascript" src="Tangle.js"></script>

  <link rel="stylesheet" href="TangleKit/TangleKit.css" type="text/css">
  <script type="text/javascript" src="TangleKit/mootools.js"></script>
  <script type="text/javascript" src="TangleKit/sprintf.js"></script>
  <script type="text/javascript" src="TangleKit/BVTouchable.js"></script>
  <script type="text/javascript" src="TangleKit/TangleKit.js"></script>

  <script type="text/javascript" src="Examples/Performance.js"></script>
</head>
```

vengono inclusi i file della cartella tangleKit e Performance.js che è il file Javascript che è stato elaborato.

La parte interessata dal tangle è quella con id=Performance. Viene definito uno span con data-var=utenti e class=TKAdjustableNumber. Dunque lo span regola la variabile utenti attraverso una classe tangle, chiamata TKAdjustableNumber. Questo avviene attraverso il trascinamento a destra o a sinistra.

```
<div id="Performance">
  <div class="filterIndent" style="position:relative;">

    <div> Se ci sono
    <span data-var="utenti" class="TKAdjustableNumber">
      </span> utenti:
    </div>

    <canvas class="PlotUtenti" data-var="utenti" >
    </canvas>
    <div class="filterParams" >
      <i>Utenti</i> = <span data-var="utenti" ></span>
    </div>

    <canvas class="PlotConessioni" data-var="connessioni" >
    </canvas>
    <div class="filterParams" >
      <i>Connessioni</i> = <span data-var="connessioni" ></span>
```

```
</div>  
  
...  
  
</div>  
</div>
```

Il resto del codice serve per avere i grafici relativi con le rispettive intestazione.

Creiamo un Canvas con `class=PlotUtenti` e `data-var=utenti`. La classe `PlotUtenti` è definita nel file `Performance.js` e rappresenta l'istogramma a barra unica che fa vedere il numero di utenti.

Il grafico viene fornito di un parametro che fa vedere il valore numerico degli utenti.

Stessa trattazione è per gli altri tipi di Canvas e di plot.

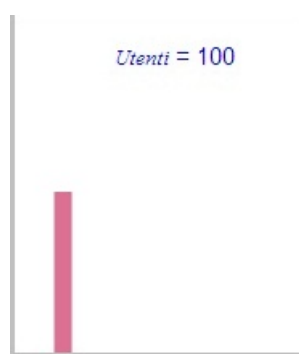
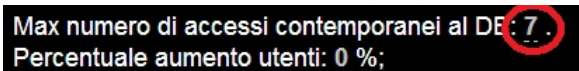


Figura 9: Tangle.PlotUtenti

### 5.2 Pagina 2

#### 5.2.1 Attività

Si è modificata la pagina precedentemente descritta. Questa nuova pagina deve introdurre



Max numero di accessi contemporanei al DB: 7  
Percentuale aumento utenti: 0 %;

Figura 10: Elemento esplorabile

un parametro che regola il numero di connessioni massime al database (vedi Fig.10). Inoltre si deve poter regolare, non il numero di utenti, ma la percentuale di crescita di quest'ultimi rispetto a un istante iniziale.

Aggiunta importante è quella dell'introduzione dell'asse temporale. Sono state creati nuovi algoritmi che potessero simulare le prestazioni del software in diversi istanti, vedi grafici nella Fig.11.

#### 5.2.2 Pagina 2-performance.js

Sono state implementate le funzioni che simulano le prestazioni del software. Viene inserito un parametro che regola il massimo numero di accessi simultanei alla memoria.

```
initialize: function () {  
    this.pcent=50;  
    this.cmax=6;  
  
}
```

I vari parametri nell'asse temporale in 12 unità vengono calcolati nel seguente modo.

Il campo `utenti1` si riferisce agli utenti presenti al tempo 1, `utenti 12` si riferisce agli utenti presenti al tempo 12. Stessa cosa per connessioni, query, pagine e memoria.

```
this.utenti1=(this.pcent/100);  
this.utenti2=this.utenti1+(this.pcent/100)*this.utenti1;  
...  
this.utenti12=this.utenti11+(this.pcent/100)*this.utenti11;
```

Gli utenti al tempo 1 è stato definito come uno gradino iniziale e calcolato con la percentuale di aumento rispetto a 0. I restanti invece vengono calcolati aggiungendo, al numero di utenti al tempo precedente, la percentuale di aumento rispetto a quest'ultimo.

```
var c=[];  
var i=1;  
var dt;  
this.query1=this.utenti1*3;  
if (this.query1<=this.cmax){  
    dt=1;  
    c[i]=this.query1;  
    i=i+1;  
}  
else {
```

```
dt=Math.ceil(this.query1/this.cmax);
for (j=0;j<=dt-1;j++){
    c[i+j]=this.cmax;

};
i=i+dt-1
c[i]=this.query1%this.cmax;
i=i+1;
};
this.tq1=0+dt;
....
this.connessioni1=c[1];
...
this.connessioni16=c[16];
```

Le query che vogliono avere risposta sono tre volte il numero degli utenti, per definizione della funzione iniziale. Se il numero di query in quell'istante non supera in numero massimo di connessioni allora vengono servite tutte contemporaneamente, cioè con uno intervallo di 1 unità di tempo rispetto al tempo precedente, se sono di più, le query non vengono servite in modo simultaneo, ma su più unità di tempo.

Le connessioni vengono abbinate a un array che contiene i valori delle connessioni in 12 unità di tempo. L'array viene inizializzato in questo modo: se il numero di query da servire sono minori o uguali al numero massimo di connessioni, le connessioni saranno quanto il numero di query, altrimenti, avremo uno o più unità di tempo (precisamente  $\text{Math.ceil}(\text{this.query2}/\text{this.cmax})-1$ ) in cui ci sono cmax connessioni attive e un'altra unità di tempo per le rimanenti.

Viene ipotizzato che ogni pagina per essere servita ha bisogno di avere tre query risposte.

```
var tcp=4;
this.pagine1=(1/3)*this.query1;
...
this.pagine12=(1/3)*this.query12;

this.tp1=this.tq1+tcp;
...
this.tp12=this.tq12+tcp;
```

Il tempo in cui le pagine vengono servite è pari al tempo di risposta delle query più il tempo di caricamento delle pagine (tcp=4).

La memoria viene acceduta dalle query e dalle pagine. Se il numero di accessi in quell'istante non supera in numero massimo di connessioni vengono evasi tutti gli accessi in memoria contemporaneamente. Se sono di più, le richieste vengono servite in più unità di tempo.

```
var r=this.query1+this.pagine1;
if(r<this.cmax){
    m[i]=r;
    i=i+1;
}
else{
    while((r-this.cmax)>=0){
```

```
        m[i]=this.cmax;
        i=i+1;
        r=r-this.cmax;
    };
    if (r>0){
        m[i]=r;
        i=i+1;
    };
};

...

this.memorial=m[0];
...
this.memorial2=m[11];
```

La memoria viene abbinata a un array che contiene tutti i valori degli accessi nelle 12 unità di tempo. L'array viene inizializzato in questo modo: se il numero di accessi è minore o uguale al numero massimo di connessioni, gli accessi saranno uguali al numero di richieste, altrimenti, avremo una o più unità di tempo in cui avremo il massimo numero di connessioni attive più un'eventuale ultima unità di tempo per completare le rimanenti richieste.

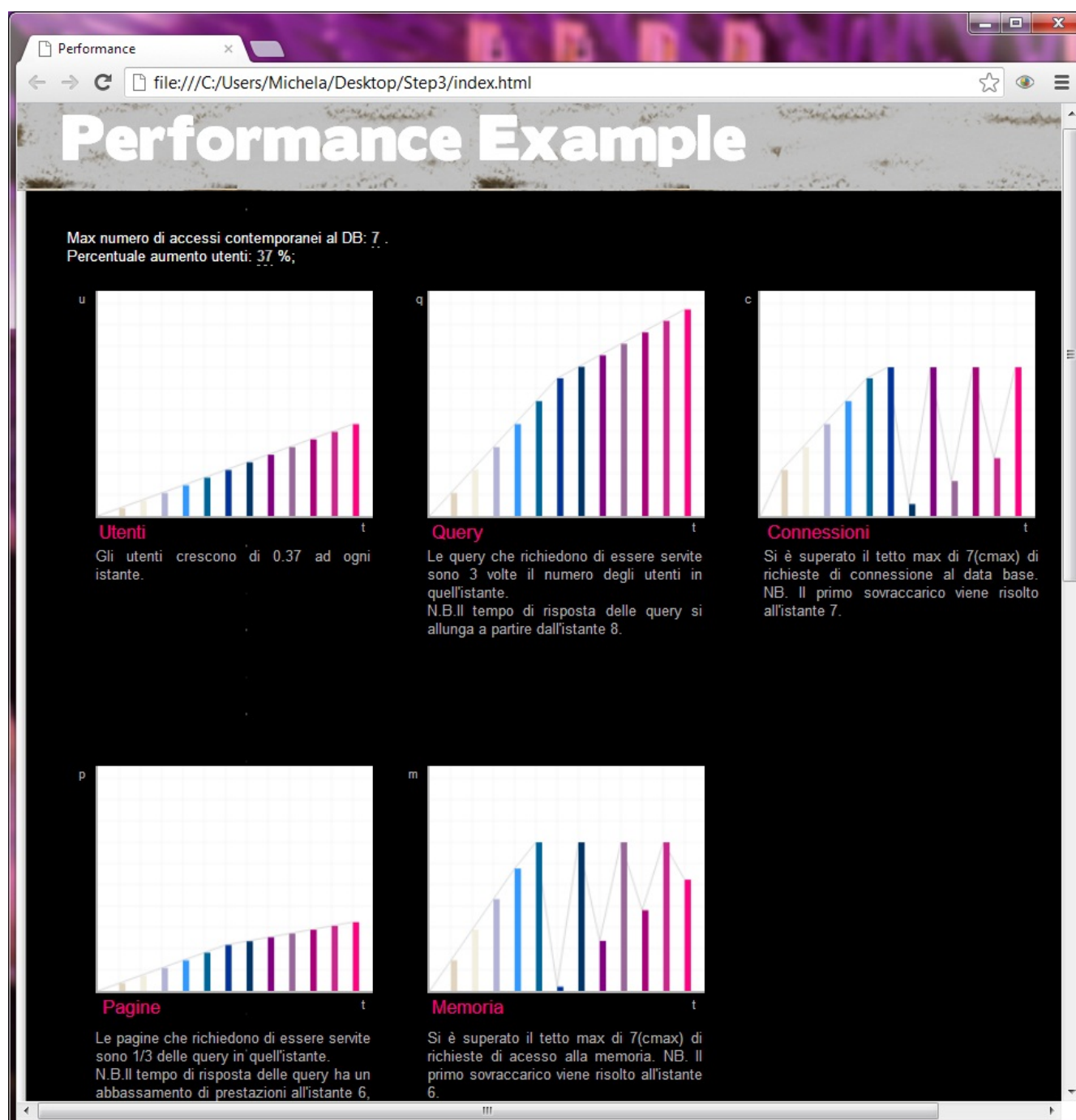


Figura 11: Prototipo numero 3 - i primi tre grafici

Inoltre le classi che regolano i plot dei Canvas hanno subito delle modifiche:

```
Tangle.classes.PlotUtenti = {  
  
    initialize: function (el, options, tangle) {  
        this.tangle = tangle;  
    },  
};
```

```
update: function (el, pcent, cmax) {

    var canvasWidth = el.get("width");
    var canvasHeight = el.get("height");
    var ctx = el.getContext("2d");

    var u1 = this.tangle.getValue("utenti1");
    ...
    var u12 = this.tangle.getValue("utenti12");

    var unit=this.getUnit();
    ctx.fillStyle = "white";
    ctx.fillRect(0, 0, canvasWidth, canvasHeight);

    //reticolato
    ctx.strokeStyle = "#EEEEEE";
    ctx.lineWidth = 0.3;
    ctx.beginPath();
    for (i=canvasHeight-unit; i>=0; i=i-unit)
    {
        ctx.moveTo(0, i);
        ctx.lineTo(canvasWidth, i);
    }
    ctx.stroke();

    ctx.strokeStyle = "#EEEEEE";
    ctx.lineWidth = 0.3;
    ctx.beginPath();
    for (i=unit; i<=canvasWidth; i=i+unit)
    {
        ctx.moveTo(i, 0);
        ctx.lineTo(i, canvasHeight);
    }
    ctx.stroke();

    //linea che congiunge
    ctx.strokeStyle = "#DCDCDC";
    ctx.lineWidth = 1;
    ctx.beginPath();
    ctx.moveTo(0, canvasHeight)
    ctx.lineTo(unit, canvasHeight - unit*u1);
    ...
    ctx.lineTo(unit*12, canvasHeight - unit*u12);
    ctx.stroke();

    //rettangoli
```

```
var color = ["#CDBFAC", ..., "#FF0080" ];
ctx.fillStyle = color[1];
ctx.fillRect(unit, canvasHeight - unit*u1, 5, u1*unit);
...
ctx.fillStyle = color[12];
ctx.fillRect(unit*12, canvasHeight - unit*u12, 5, u12*unit);

},

getUnit: function () { return 15; }
```

Come si può vedere dal codice, attraverso la funzione di tangle `getValue()` vengono presi i valori che servono da tangle e vengono inserite nelle variabili.

Successivamente si crea un reticolato per visualizzare meglio i valori nella resa grafica.

Il reticolato ha come unità un valore proveniente da una funzione, `getUnit()`. E' facile cambiare il valore di ritorno per visualizzare una vista più grande o più piccola. Poi vengono disegnati nel Canvas dei rettangoli di larghezza tre che rappresentano gli utenti nelle varie unità di tempo.

Sono state aggiunte anche una linea che congiunge tutti i rettangoli, per fornire graficamen-

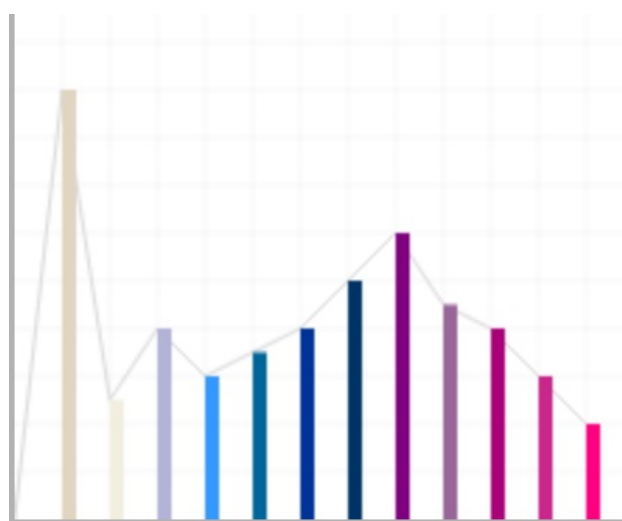


Figura 12: plot ottenuto con il Canvas

te l'andamento dell'afflusso di utenti. Le altre tre classi tangle, `PlotQuery`, `PlotConnessioni`, `PlotPagine`, `PlotMemoria` sono state modificate seguendo lo stesso principio.

### 5.2.3 Pagina 2-index.html

Viene introdotta un'altra variabile che è possibile trascinare, cioè `cmax`. Inoltre i grafici sono stati forniti del nome delle ordinate e delle ascisse. Sulle ordinate abbiamo messo il tempo.

```
<div>Max numero di accessi contemporanei al DB:
<span data-var="cmax" class="TKAdjustableNumber" ></span> .</div>
<div> Percentuale aumento utenti:
```



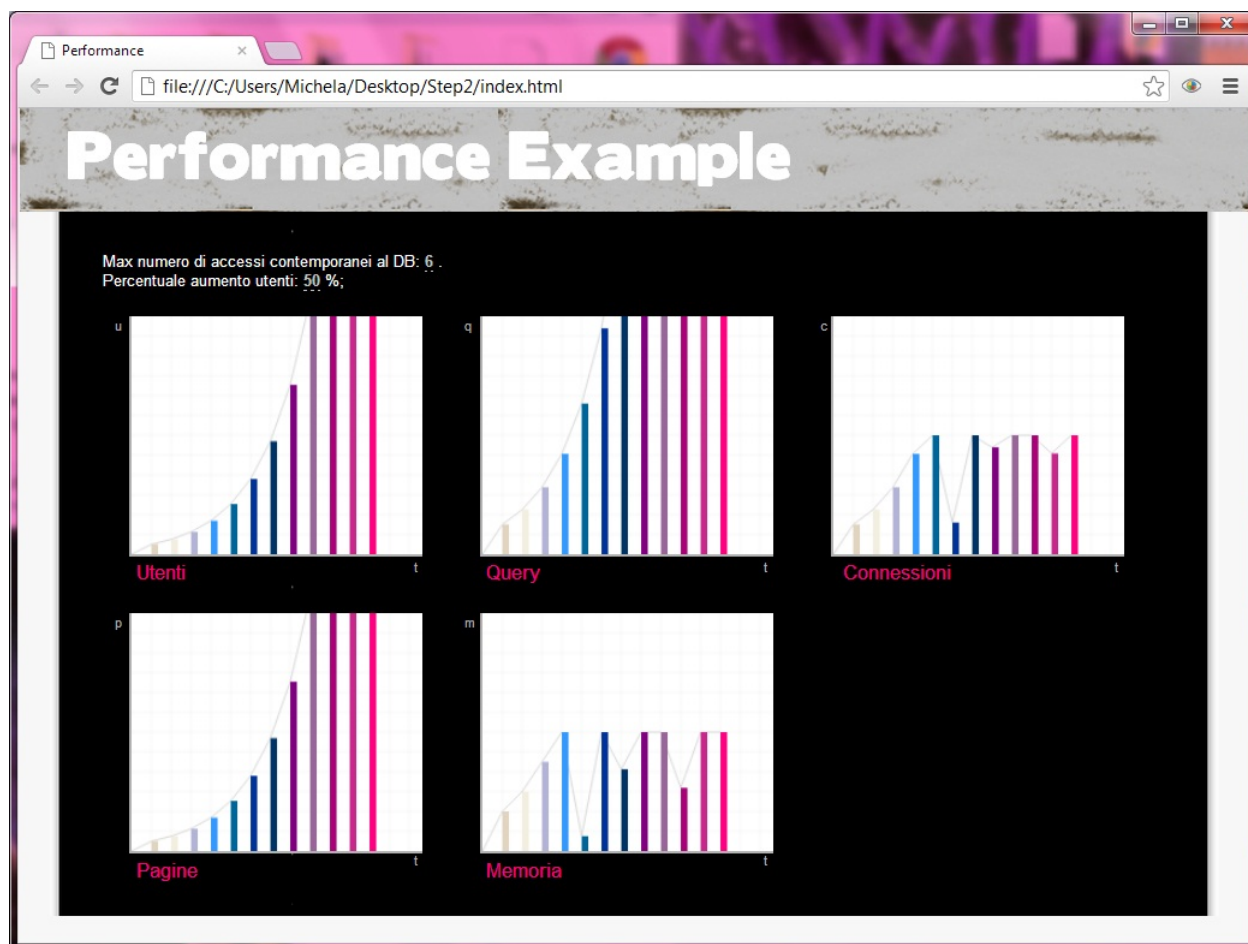


Figura 13: index.html-pagina 2

```
<span data-var="pcent" class="TKAdjustableNumber" ></span> %;</div>
```

```
<canvas class="PlotUtenti" data-var="pcent cmax" ></canvas>
```

```
<div class="target" >Utenti</div>
```

```
<div class="parametri" >u</div>
```

```
<div class="parametri" >t</div>
```

### 5.3 Pagina 3

#### 5.3.1 Attività

Successivamente si sono modellati gli scenari, cioè quelle situazioni che ci interessava analizzare andamento dei vari grafici: semplici spiegazioni testuali di cosa succede nel grafico, situazioni critiche o di stallo, indicazioni su come risolvere una determinata situazione, vedi spiegazione sotto i grafici della Fig.6.

Attraverso la regolazione della percentuale di aumento degli utenti e del numero massimo di connessioni al database, si possono esplorare diverse situazioni, per intraprendere un approccio critico sull'argomento.

#### 5.3.2 Pagina 3-performance.js

Le principali modifiche introdotte sono:

- il codice è stato reso più mantenibile, lavorando solo su array e passando soltanto alla fine i dati alle variabili che saranno prelevate per la visualizzazione (es.utenti1,utenti2,...,utenti12).
- a seconda dei valori inseriti serviva un diverso grado di precisione nell'arrotondamento, è stato quindi necessario inserire una funzione getPrec().

```
u = [];  
u[0] = 0;  
this.a = Math.round(this.getPrec() * this.pcent * 0.01) / this.getPrec();  
for (i = 1; i <= 12; i++) {  
    u[i] = Math.round(this.getPrec() * (i * this.a)) / this.getPrec();  
}  
this.utenti1 = u[1];  
...  
this.utenti12 = u[12];
```

- il nostro vecchio schema definisce come query1 le query arrivate al primo istante e servite all'istante tq1. Sono stati identificati i valori delle query in ogni istante. E' stato, quindi, realizzata una interpolazione lineare. Innanzitutto è stato calcolato come nella precedente versione le query e il tempo per servirle, inserendole nell'array queryp e tq.

```
var cn = []; cn[0] = 0;  
var queryp = []; queryp[0] = 0;  
var tq = []; tq[0] = 0;  
var i = 0;  
for (k = 1; k <= 12; k++)  
{  
    var dt;  
    queryp[k] = u[k] * 3;  
    if (queryp[k] <= this.cmax) {  
        dt = 1;  
        cn[i] = queryp[k];  
        i = i + 1;  
    }  
    else {  
        dt = Math.ceil(queryp[k] / this.cmax);
```

```
        for (j=0;j<=dt-1;j++){
            cn[i+j]=this.cmax;
        };
        i=i+dt-1
        cn[i]=queryp[k]%this.cmax;
        i=i+1;
    }
    tq[k]=tq[k-1]+dt;
}

this.connessioni1=cn[1];
...
this.connessioni16=cn[16];
```

Successivamente è stato realizzato un'interpolazione lineare, per fare in modo di visualizzare il numero di query servite per ogni unità di tempo.

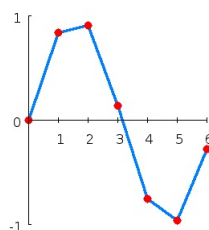


Figura 14: interpolazione lineare

Dati due punti  $A(x_1, y_1)$  e  $B(x_2, y_2)$  si ha un coefficiente angolare  $m = (y_2 - y_1) / (x_2 - x_1)$ . L'equazione della retta passante per i due punti è data dall'equazione  $y = m(x - x_1) + y_1$ . Avendo gli array precedentemente descritti, abbiamo calcolato l'equazione delle rette passanti per i punti `queryp[1], queryp[2]` e per `queryp[2], queryp[3]` ecc e presi valori discreti di tali rette.

In alcuni casi quando il valore del tempo era già discretizzato lo abbiamo semplicemente prelevato.

```
var q=[];
q[0]=0;
var j=1;
for (i=1;i<=12;i++)
{
    var ok=false;
    while (!ok){
        if (tq[j]<i){
            j++;
        };
        if (tq[j]==i) {
            q[i]=queryp[j];
            ok=true;
        }
        else {
```

```
        var m=(tq[j]-tq[j-1])/(queryp[j]-queryp[j-1]);
        var p=i-tq[j-1];
        q[i]=p/m+queryp[j-1];
        ok=true;
    };

}
};

this.query1=q[1];
...
this.query12=q[12];
```

- abbiamo annesso gli scenari. Gli scenari sono le situazioni in cui si può trovare la nostra applicazione web. Devono evidenziare situazioni di stallo e trovare i punti chiave. Sono state create le variabili `scenarioIndex`, `scenarioqIndex`, `scenarioIndex`, `scenarioIndex`, `scenarioIndex` rispettivamente per indicare le situazioni in cui si trova il grafico degli utenti, quello delle query, quello delle connessioni, quello delle pagine e quello della memoria. L'indice del grafico degli utenti assume lo scenario numero 1 quando gli utenti crescono più del 100%, 0 altrimenti.

```
this.scenarioIndex=0;
if (this.a>=1){this.scenarioIndex=1;}
```

L'indice del grafico delle query assume:

- valore 0 se il trend di crescita è al pari o in salita.
- valore 1 se il trend di crescita diminuisce. Vuol dire che ci stiamo avvicinando a una situazione di stallo. L'istante in cui si verifica questa situazione è `iq1`.
- valore 2 se siamo nello `scenarioIndex=1` e c'è una seconda diminuzione di crescita. L'istante in cui si verifica questa situazione è `iq2`.

```
this.scenarioqIndex=0;
for (i=2; i<=11 && (this.scenarioqIndex==0); i++){
    if ( q[i]>this.cmax){
        this.scenarioqIndex=1;
        this.iq1=i+1;
    }
}
if (this.scenarioqIndex==1){
    for ( i=this.iq1+1; (i<=11)&&(this.scenarioqIndex==1); i++){
        if ( q[i]>2*this.cmax){
            this.scenarioqIndex=2;
            this.iq2=i+1;
        }
    }
}
```

L'indice del grafico delle pagine assume:

- valore 0 se il trend di crescita è al pari o in salita.
- valore 1 se il trend di crescita diminuisce. Vuol dire che ci stiamo avvicinando a una situazione di stallo. L'istante in cui si verifica questa situazione è ip1.
- valore 2 se siamo nello scenarioqIndex=1 e c'è una seconda diminuzione di crescita. L'istante in cui si verifica questa situazione è ip2.

```
this.scenarioIndex=0;
for(i=2; i<=11 && (this.scenarioIndex==0); i++){
    if( (Math.round(this.getPrec()*(p[i]-p[i-1]))/this.getPrec()) >
        (Math.round(this.getPrec()*(p[i+1]-p[i]))/this.getPrec()) )
        {
            this.scenarioIndex=1;
            this.ip1=i;
        }
    }
    if(this.scenarioIndex==1){
        for(i=this.ip1+1; (i<=11)&&(this.scenarioIndex==1); i++){
            if( (Math.round(this.getPrec()*(p[i]-p[i-1]))/this.getPrec()) >
                (Math.round(this.getPrec()*(p[i+1]-p[i]))/this.getPrec()) )
                {
                    this.scenarioIndex=2;
                    this.ip2=i;
                }
            }
        }
    }
```

L'indice del grafico delle connessioni assume

- valore 0 se con un solo accesso si riesce a rispondere e tutte le richieste pendenti.
- valore 1 se bisogna fare un turno per risolvere le richieste pendenti. L'istante in cui si verifica questa situazione è ic

```
this.scenarioIndex=0;
for(i=2; i<=11 && (this.scenarioIndex==0); i++){
    if( cn[i]>cn[i+1] ){
        this.scenarioIndex=1;
        this.ic=i+1;
    }
}
```

L'indice del grafico degli accessi alla memoria assume

- valore 0 se con un solo accesso si riesce a rispondere e tutte le richieste pendenti.
- valore 1 se bisogna fare un turno per risolvere le richieste pendenti. L'istante in cui si verifica questa situazione è im.

```
this.scenarioIndex=0;
for(i=1; i<=11 && (this.scenarioIndex==0); i++){
    if( m[i]>m[i+1] ){
        this.scenarioIndex=1;
    }
}
```

```
    this.im=i+1;
  }
}
```

### 5.3.3 Pagina 3-index.html

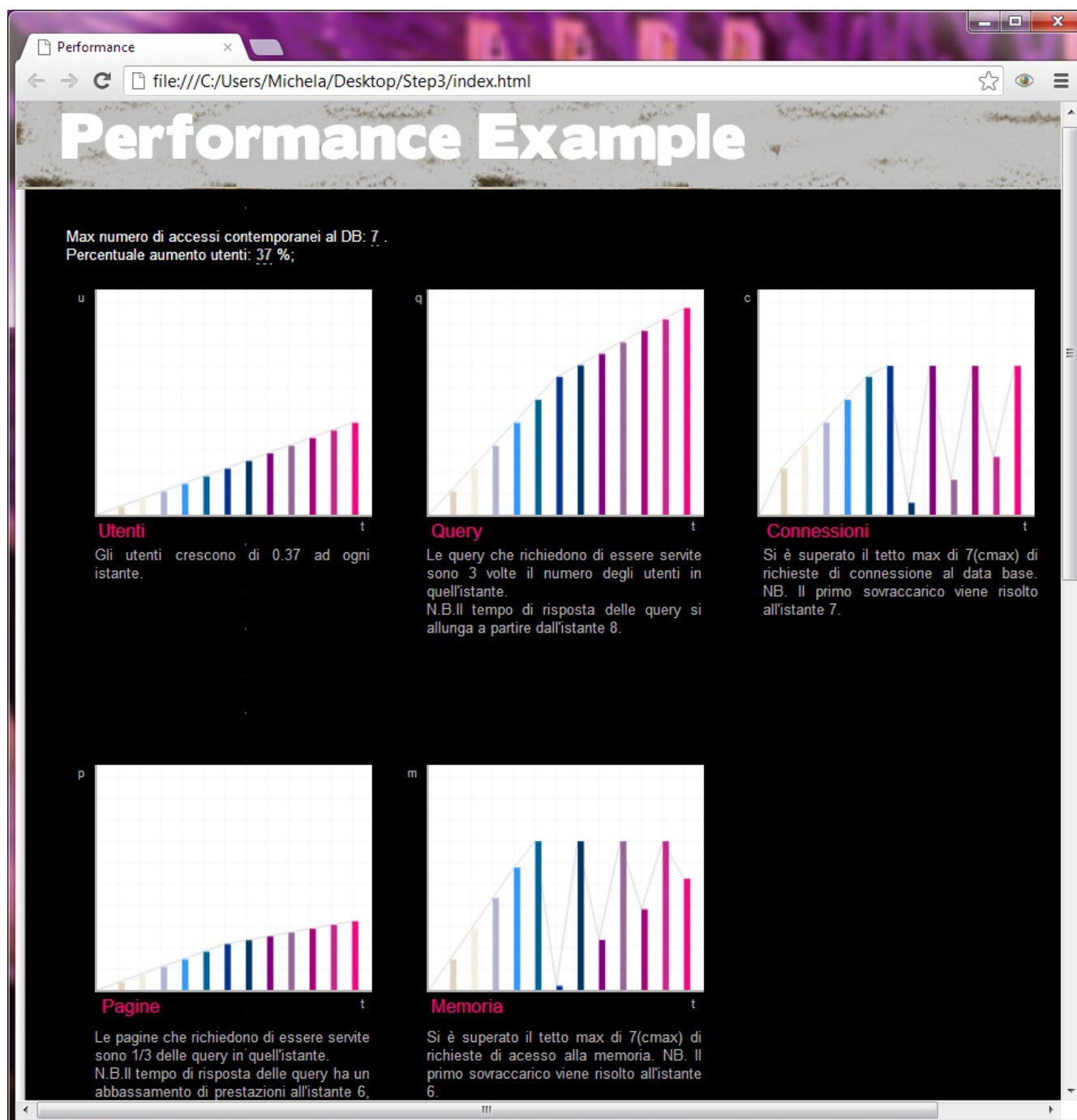


Figura 15: index.html-pagina 3

Le uniche aggiunte rispetto alla pagina precedente sono i tag per visualizzare gli scenari alternativi, in questo modo:

```
<div class="spiegazioni su">
  <span data-var="scenariouIndex" class="TKSwitch">
    <span>Gli utenti crescono di <span data-var="a" ></span></span>
    <span>Attenzione utenti in forte aumento.</span>
  </span>
</div>
```

La classe TKSwitch mostra solo uno dei figli, a seconda del valore assunto dalla variabile scenariouIndex.

### 5.4 Pagina 4

#### 5.4.1 Attività

La pagina è in grado di importare i dati relativi agli utenti nei vari istanti da una fonte dati esterna. E' stata caricata su Tomcat, un contenitore che opera all'interno di un server. I dati relativi agli utenti sono in JSON e la richiesta è fatta con Ajax.

La fonte dati esterna è il file, nel quale sono presenti i dati sugli utenti in formato JSON.

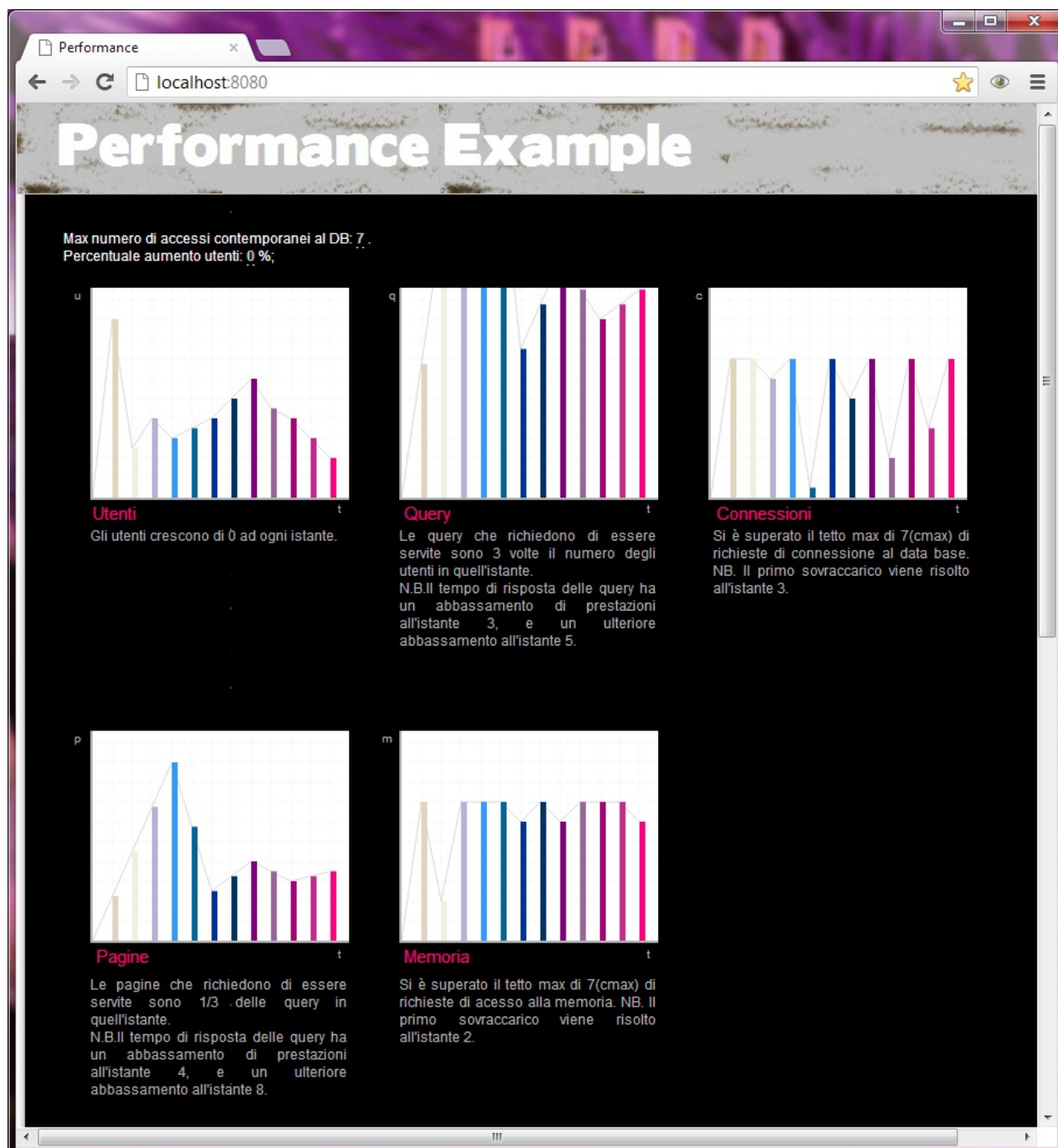


Figura 16: Prototipo numero 4 - i primi tre grafici



Sono state leggermente modificate le funzioni per riuscire a prendere come valore iniziale del grafico utente, il valore della fonte dati esterna e permettere l'aumento o la diminuzione proporzionale di questi dati (vedi Fig.7).

Con questa pagina si è conclusa la fase di prototipazione del prodotto finale ed è stata stesa una documentazione sulle pagine realizzate fino a quel momento.

### 5.4.2 Pagina 4-performance.js

In questa pagina bisogna fare in modo di prelevare i dati da una fonte dati esterna. La fonte dati è il file my.json nel quale sono presenti i dati sugli utenti in formato JSON. Per prelevare questi dati abbiamo dovuto mettere la nostra pagina in Tomcat e aggiungere una richiesta xmlhttprequest. Il file JSON è molto semplice:

```
// file my.json
{utenti:[0,2,2.5,4,3,3.5,4,5,6,4.5,4,3,2,5]}

//performance.js
...
var tangle = new Tangle(document.getElementById("Performance"), {

    initialize: function () {
        this.pcent=0;
        this.cmax=7;

        //prelevo i valori degli utenti dal file my.json
        var xmlhttp;
        xmlhttp=( (window.XMLHttpRequest)?
            new XMLHttpRequest():new ActiveXObject("Microsoft.XMLHTTP"));
        xmlhttp.open("GET","my.json",false);
        xmlhttp.send();
        jsontext=xmlhttp.responseText;
        var obj = eval("(" + jsontext + ")");
        this.f1=eval( obj.utenti[1]);
        ...
        this.f12=eval( obj.utenti[12]);
    },
    ...
}
```

Come si può vedere nel codice, durante l'inizializzazione ho creato un elemento xmlhttp che a seconda se il browser supporti XMLHttpRequest oppure no crea un nuovo elemento di quel tipo o per Internet Explore un ActiveXObject. Questo oggetto apre il file e preleva il suo contenuto.

Successivamente tramite la funzione eval() avviene il parser del contenuto e tramite una semplice espressione *utenti[i]* prelevo i-simo elemento dell'array. Questi valori serviranno per inizializzare i valori delle variabili utenti1, utenti2, utenti3, utenti4, utenti5, utenti6, utenti7, utenti8, utenti9, utenti9, utenti10, utenti11, utenti12.

Infine è stato modificato il metodo con cui si ricavano gli utenti per rendere possibile anche un pcent negativo.

```
this.utenti1=this.f1*(1+this.a);
this.utenti2=this.f2*(1+this.a);
...
```

```
this.utenti12=this.f12*(1+this.a);  
var u=[0,this.utenti1,...,this.utenti11,this.utenti12];
```

### 5.4.3 Pagina 4-index.html

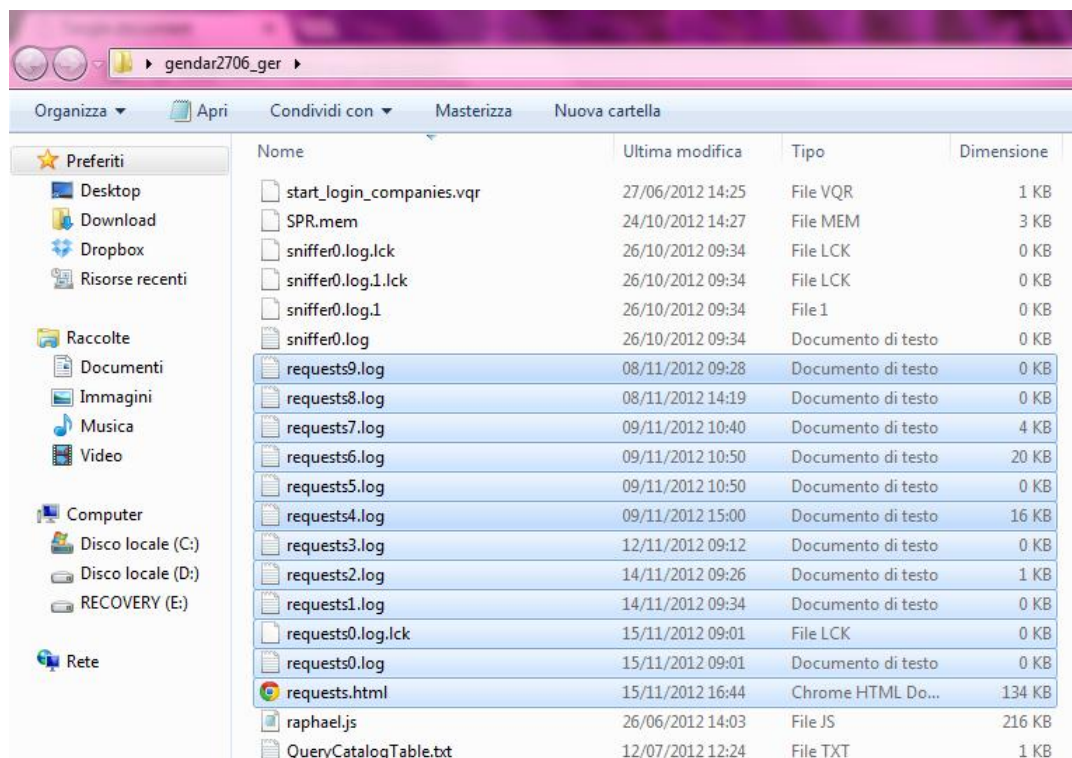
Questa pagina rimane invariata rispetto la precedente, cambia solamente il file .js a cui fa riferimento.

## 6 Sviluppo Prodotto finale

### 6.1 Attività

Nella fase finale è stato studiato il Proxy, inizialmente in uso in azienda. E' stata condotta un'analisi della pagina che il Proxy va a testare, della vecchia interfaccia grafica, ma soprattutto del log del Proxy.

Il Proxy, ad ogni cattura, mette i risultati in un insieme di file con numero sequenziale (vedi



Nome	Ultima modifica	Tipo	Dimensione
start_login_companies.vqr	27/06/2012 14:25	File VQR	1 KB
SPR.mem	24/10/2012 14:27	File MEM	3 KB
sniffer0.log.lck	26/10/2012 09:34	File LCK	0 KB
sniffer0.log.1.lck	26/10/2012 09:34	File LCK	0 KB
sniffer0.log.1	26/10/2012 09:34	File 1	0 KB
sniffer0.log	26/10/2012 09:34	Documento di testo	0 KB
requests9.log	08/11/2012 09:28	Documento di testo	0 KB
requests8.log	08/11/2012 14:19	Documento di testo	0 KB
requests7.log	09/11/2012 10:40	Documento di testo	4 KB
requests6.log	09/11/2012 10:50	Documento di testo	20 KB
requests5.log	09/11/2012 10:50	Documento di testo	0 KB
requests4.log	09/11/2012 15:00	Documento di testo	16 KB
requests3.log	12/11/2012 09:12	Documento di testo	0 KB
requests2.log	14/11/2012 09:26	Documento di testo	1 KB
requests1.log	14/11/2012 09:34	Documento di testo	0 KB
requests0.log.lck	15/11/2012 09:01	File LCK	0 KB
requests0.log	15/11/2012 09:01	Documento di testo	0 KB
requests.html	15/11/2012 16:44	Chrome HTML Do...	134 KB
raphael.js	26/06/2012 14:03	File JS	216 KB
QueryCatalogTable.txt	12/07/2012 12:24	File TXT	1 KB

Figura 17: Files requests

Fig.17).

Sono state avanzate proposte sul modo migliore per presentare i dati del Proxy, su che parametri si volevano fare regolare nella GUI e la linea grafica generale.

Il prodotto finale rispetto alla Pagina 5 completa gli scenari importanti. Questi scenari sono relativi all'andamento complessivo dei record catturati dal log del Proxy.

Per la presentazione dei dati è stata realizzata una pagina (vedi Fig.18) che presenta i grafici relativi al tempo di domanda/risposta delle pagine e dei byte trasportati.

Si è reso necessario l'introduzione di uno strumento di zoom, realizzato sempre con la libreria Tangle e di una sezione che mostrasse i dettagli di ogni singolo record, quando il mouse si trova sopra la parte di grafico relativo.

Per l'esposizione degli scenari finali si è pensato alla seguente scaletta.

Alcuni dati sono riassuntivi delle performance e non interagibili, altri invece grazie alla regolazione di alcuni parametri sono esplorabili, come il bitrate, la percentuale di record con

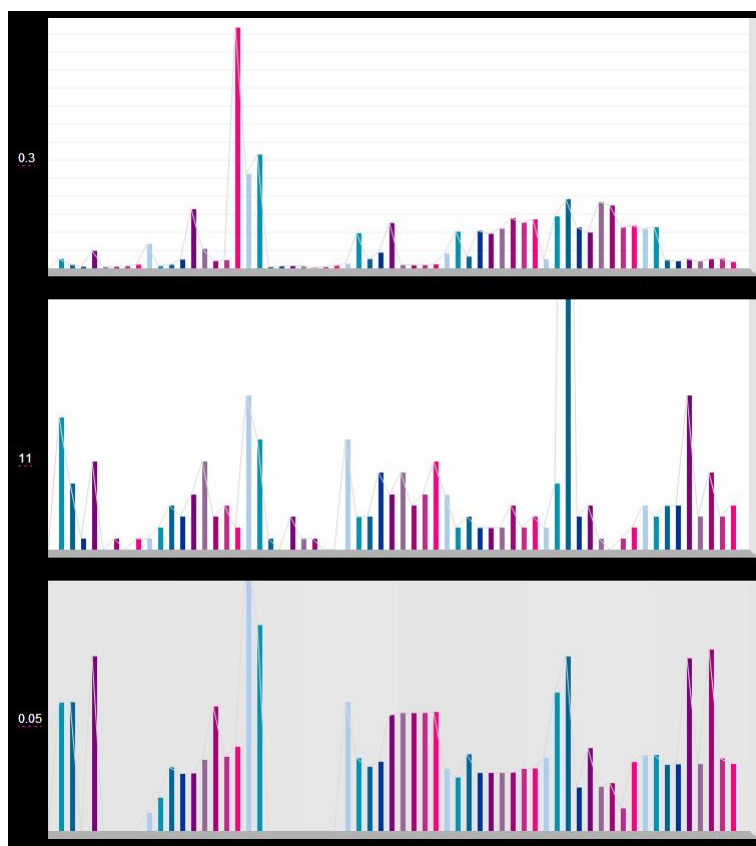


Figura 18: Prototipo pagina web finale

un tempo di risposta maggiore a una certo tetto massimo ecc. La Fig.18 mostra la resa grafica degli scenari. Questa pagina è il frutto di un confronto con il tutor esterno. Si è cercato di aggiungere tutte le funzionalità che erano necessarie al reale utilizzo di questa pagina di presentazione. La resa grafica finale è mostrata dalla Fig.20. E' stato fatto un collaudo informale della pagina, per testare l'effettivo funzionamento, anche in situazioni limite.

Tutti i requisiti richiesti sono stati soddisfatti.

E' stato redatto un documento di fine stage che spiega le varie fasi di realizzazione del prodotto finale, ne analizza di dettagli tecnici e da una valutazione generale sull'intera attività di stage.

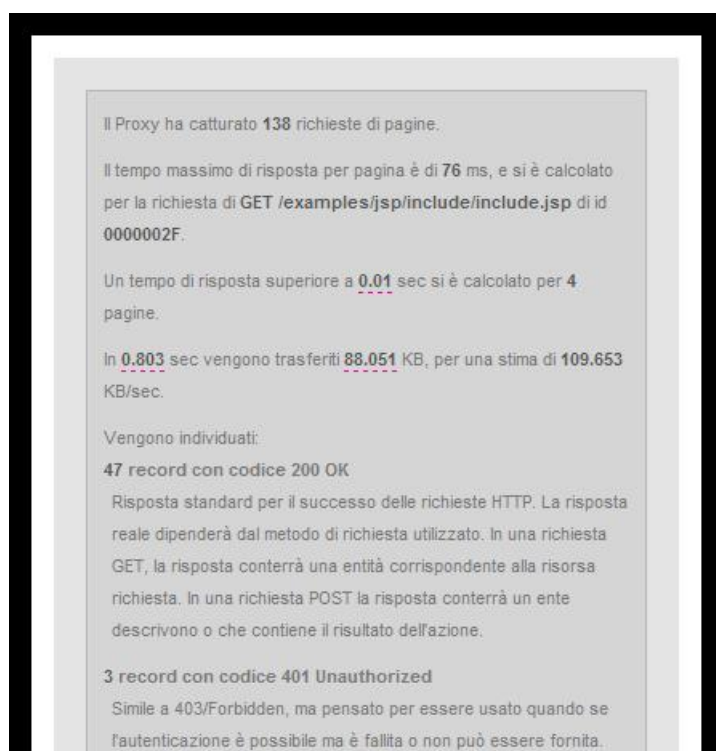


Figura 19: Scenari sull'andamento generale e particolare I parte

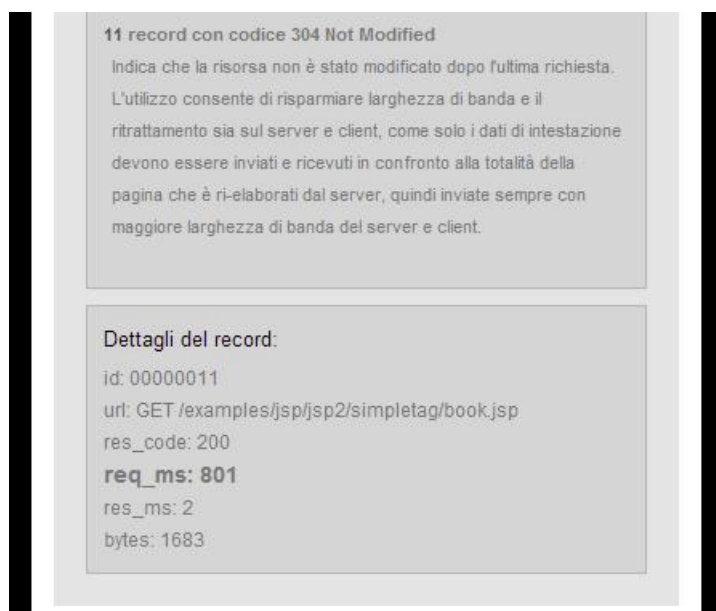


Figura 20: Scenari sull'andamento generale e particolare II parte

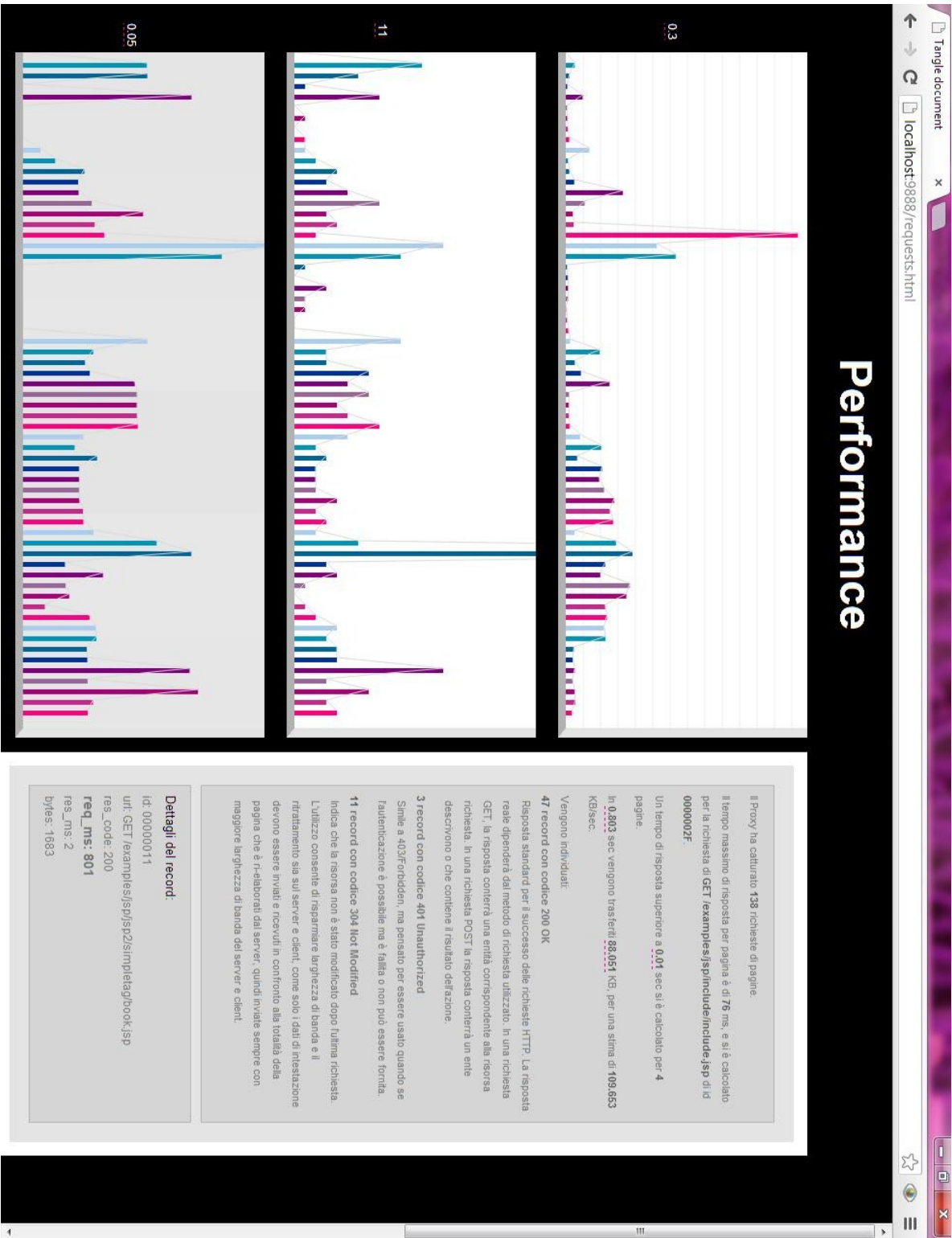


Figura 21: Pagina Web

### 7 I casi d'uso

Finito lo sviluppo prototipale si sono fissati i requisiti definitivi del prodotto finale. I casi d'uso(UC) sono stati usati per esprimere in maniera esaustiva e non ambigua, la raccolta dei requisiti al fine di produrre software di qualità.

Il tracciamento dei requisiti consiste nel valutare ogni requisito focalizzandosi sugli attori che interagiscono col sistema, valutandone le varie interazioni. In UML sono rappresentati dagli Use Case Diagram.

Il caso d'uso individua e descrive gli scenari elementari di utilizzo del sistema da parte degli attori che si interfacciano con esso (esseri umani oppure da sistemi informativi esterni).

Un caso d'uso deve essere elementare, cioè non scomponibile in casi d'uso più semplici che abbiano ancora senso compiuto per gli attori coinvolti.

I casi d'uso verranno raggruppati per area funzionale e per pre-condizioni generali comuni.

Le relazioni fra casi d'uso ed attori possono essere rappresentate in un apposito diagramma UML.

Inoltre le relazioni fra casi d'uso e requisiti possono essere mantenute attraverso una apposita matrice di tracciamento, che riporta la mappatura fra casi d'uso e requisiti funzionali. In generale ciascun requisito funzionale dovrebbe rispecchiarsi in almeno un caso d'uso, e nessun caso d'uso dovrebbe implicare requisiti funzionali non presenti nel documento dei requisiti o in contraddizione con essi.

## 7.1 Casi d'uso

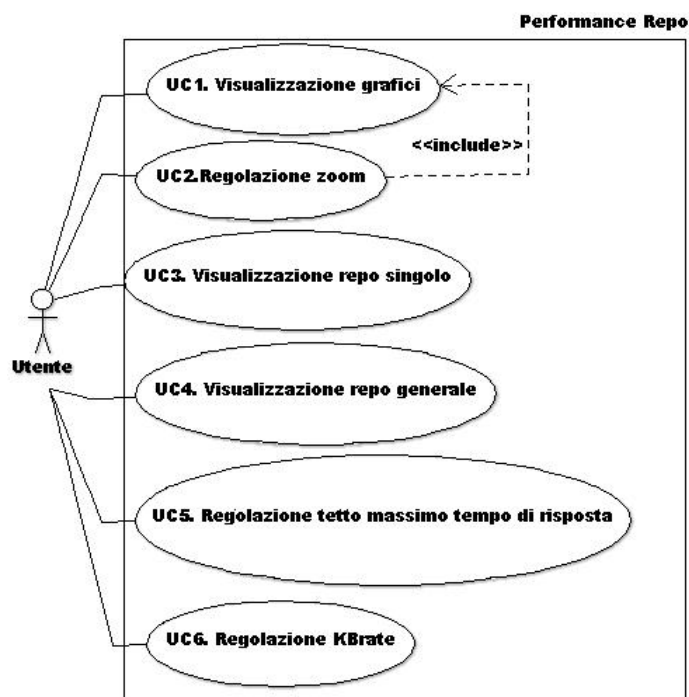


Figura 22: Diagramma generale dei casi d'uso

***Precondizione:***

Performance Repo è aperto

***Attore principale:***

Utente

***Sequenza principale:***

L'utente può fare le seguenti azioni:

- Visualizzare i grafici, che presentano i dati provenienti dal Proxy;
- Regolare lo zoom dei grafici. Ogni volta che verrà regolato lo zoom il rispettivo grafico si potrà visualizzare ingrandito o rimpicciolito;
- Visualizzare repo singolo;
- Visualizzare il repo generale;
- Regolare un tetto massimo di tempo di risposta delle richieste;
- Regolare il KBrate;

***Postcondizione:***

L'utente riesce ad interagire con la pagina, che visualizza le informazioni catturate dal Proxy e interagire con esse.



### *Sequenza alternativa:*

Il Proxy non ha catturato nessuna richiesta. Viene dato il relativo messaggio nel repo generale.

### 7.1.1 UC1 - Visualizzazione grafici

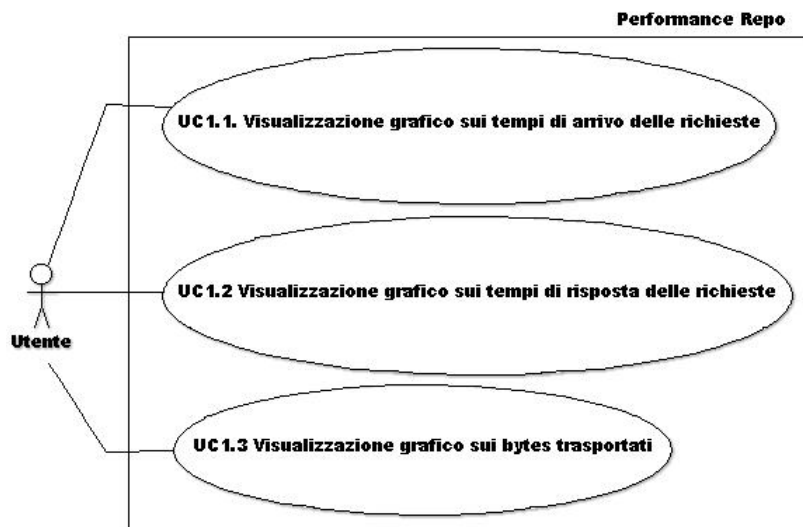


Figura 23: Diagramma del caso d'uso 1

***Precondizione:***

Performance Repo è aperto. Si visualizzano dati sui grafici.

***Attore principale:***

Utente

***Sequenza principale:***

L'utente visualizza:

- il grafico che presenta i tempi di arrivo delle richieste a partire all'istante di cattura;
- il grafico che presenta i tempi di risposta delle richieste;
- il grafico che presenta i bytes trasportati;

***Postcondizione:***

L'utente riesce a visualizzare i grafici.

***Sequenza alternativa:***

I record catturati sono superiori ai limiti di visualizzazione della pagina, quelli in esubero non verranno mostrati graficamente.

### 7.1.2 UC2 - Regolazione zoom

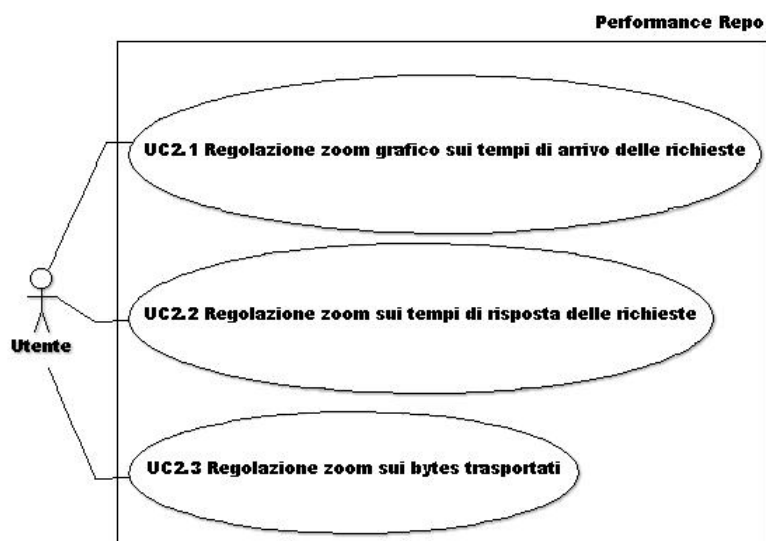


Figura 24: Diagramma del caso d'uso 2

***Precondizione:***

Performance Repo è aperto. Si visualizzano dati sui grafici.

***Attore principale:***

Utente

***Sequenza principale:***

L'utente può giocare con il valore delle unità di misura dei tre grafici.

***Postcondizione:***

L'utente riesce cambiare il valore delle unità di misura.

***Sequenza alternativa:***

L'utente tenta di impostare un valore di zoom che è fuori l'intervallo di tempo per quel determinato grafico. Non si riesce più a trascinare il paramento in quella direzione.

### 7.1.3 UC3 - Visualizzazione repo singolo

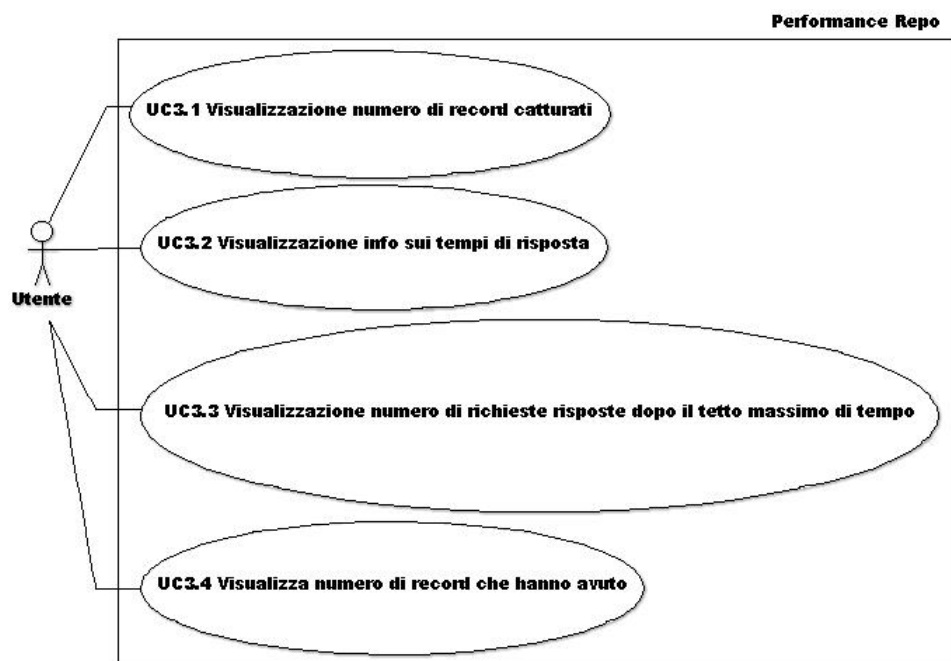


Figura 25: Diagramma del casi d'uso 3

**Precondizione:**

Performance Repo è aperto. Si visualizzano dati sui grafici.

**Attore principale:**

Utente

**Sequenza principale:**

L'utente visualizza:

- id, cioè l'identificativo del record;
- url, l'indirizzo della pagina;
- res\_code, il codice di stato della richiesta;
- req\_ms, il tempo di arrivo della richiesta;
- res\_ms, il tempo di risposta della richiesta;
- bytes trasportati;

**Postcondizione:**

L'utente riesce a visualizzare i dettagli di un singolo record.

**Sequenza alternativa:**

Si possono verificare le seguenti sequenze alternative:

- il mouse dell'utente è fuori del grafico. Viene mostrato un relativo messaggio;
- il mouse dell'utente è dentro il grafico ma non sta puntando nessuna barra. Viene mostrato un relativo messaggio.

#### 7.1.4 UC4 - Visualizzazione repo generale

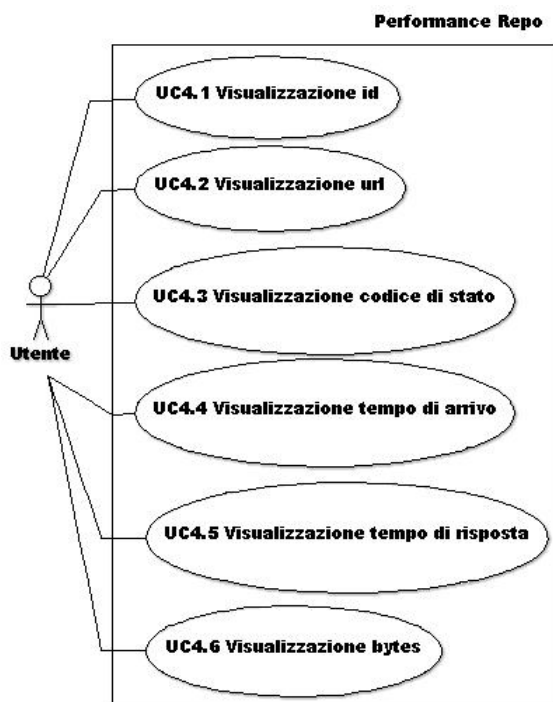


Figura 26: Diagramma del caso d'uso 4

***Precondizione:***

Performance Repo è aperto

***Attore principale:***

Utente

***Sequenza principale:***

L'utente visualizza:

- il numero di record catturati;
- tempo di risposta massimo catturato, id e url di tale record;
- numero di richieste che sono state risposte in un tempo superiore ad un tetto massimo;
- numero di richieste che hanno avuto un determinato codice di stato e la descrizione di tale stato. Vengono rilevati fino a 8 codici di stato.

***Postcondizione:***

L'utente riesce a visualizzare un repo generale delle informazioni provenienti dal Proxy.

***Sequenza alternativa:***

Non vi sono dati catturati. Viene mostrato un messaggio relativo.

#### 7.1.5 UC5 - Regolazione tetto massimo tempo di risposta

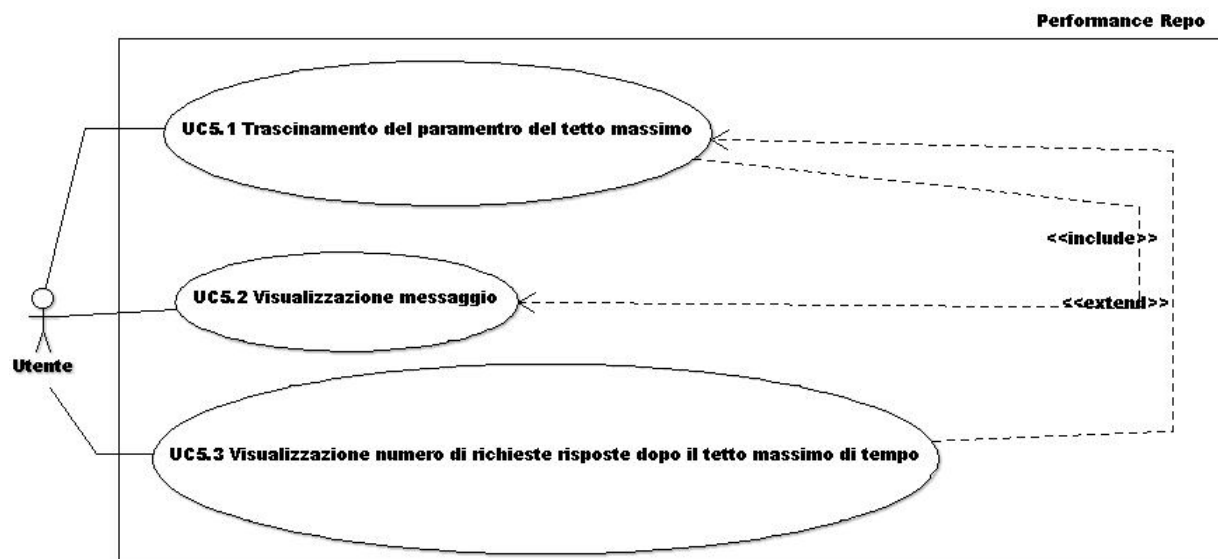


Figura 27: Diagramma del caso d'uso 5

***Precondizione:***

Performance Repo è aperto. Si visualizzano dati sui grafici.

***Attore principale:***

Utente

***Sequenza principale:***

Ogni volta che l'utente trascina il paramentro del tetto massimo sul tempo di risposta gli viene mostrato il numero di richieste che superano tale tempo. Se il numero di richieste supera il 30 del totale verrà mostrato un avviso.

***Postcondizione:***

L'utente riesce a giocare con il paramentro che regola il tetto massimo del tempo di risposta delle richieste.

***Sequenza alternativa:***

L'utente tenta di impostare un tetto massimo che è fuori l'intervallo deciso. Non si riesce più a trascinare il paramentro in quella direzione.

### 7.1.6 UC6 - Regolazione KBrate

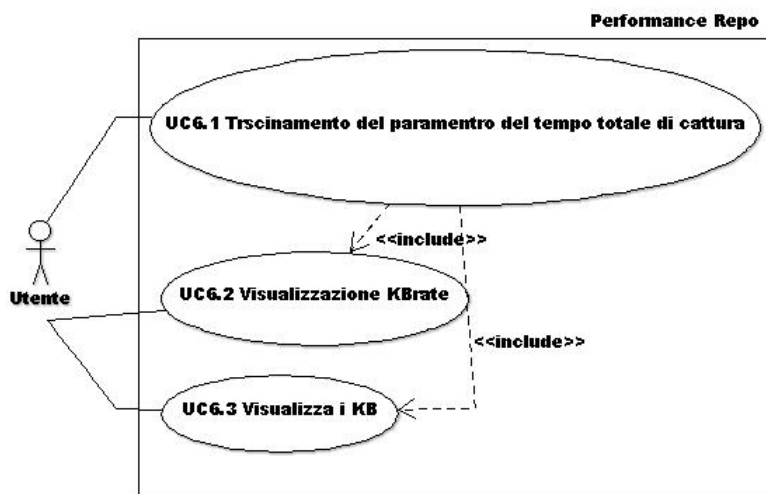


Figura 28: Diagramma del caso d'uso 6

**Precondizione:**

Performance Repo è aperto. Si visualizzano dati sui grafici.

**Attore principale:**

Utente

**Sequenza principale:**

Ogni volta che l'utente trascina il parametro dei secondi riesce a visualizzare i KB trasportati e il KBrate. Ogni volta che l'utente trascina il parametro del KB-rate vede cambiare i secondi che ci vogliono per trasportare i KB totali.

**Postcondizione:**

L'utente riesce a giocare con il parametro che regola i secondi e il KB.

**Sequenza alternativa:**

L'utente tenta di impostare i valori regolabili fuori dall'intervallo deciso. Non si riesce più a trascinarli in quella direzione.



### 7.2 Definizione del prodotto

In questa sezione si analizzerà il codice sorgente del prodotto finale. Esso si articola in molti script. Verranno spiegati in dettaglio quelli creati per la realizzazione della pagina, lasciamo alla sezione *'Specifica tecnica'* la trattazione degli altri script di supporto non sviluppati durante lo stage.

#### 7.2.1 Script Tangle

Lo script inizia con l'acquisizione dei dati provenienti dal log del Proxy. Il log del Proxy è un file composto da un sequenza di stringhe di questo tipo:

```
2012-11-09 14:21:20 00000001 GET
/examples/implicit-objects.jsp?parameters%20omitted
HTTP/1.1 200 from request:34 ms from first response: 5 ms 3210 bytes
```

```
2012-11-09 14:21:23.403 00000002 GET
/examples/jsp/jsp2/el/implicit-objects.jsp?parameters%20omitted
HTTP/1.1 200 from request:13 ms from first response: 8 ms 3210 bytes
```

Da ora in poi con *record* si identificherà una n-pla di questi dati:

- **id** (*es. 00000001*): serve per identificare in modo univoco il record. E' sequenziale e espresso in forma esadecimale su 8 bit.
- **url** (*es. /examples/implicit-objects.jsp?parameters%20omitted*) : è una stringa con l'indirizzo della pagina che si vuole richiedere.
- **res\_code** (*es. 200*) : è un numero di 3 cifre che rappresenta lo stato. Quando il server riceve una richiesta per una pagina da un sito, restituisce un codice di stato HTTP in risposta alla richiesta. Questo codice di stato fornisce informazioni sullo stato della richiesta.
- **req\_ms** (*es. 34*): è un numero che rappresenta il tempo in ms in cui è stata istanziata la richiesta della pagina rispetto all'inizio della cattura.
- **res\_ms** (*es. 5*): è un numero che rappresenta il tempo in ms in cui viene servita la pagina.
- **bytes** (*es. 3210*): è un numero che dice quanti byte è grande la pagina.

Il ciclo for preleva il file log più recente. La variabile *p* rappresenta l'espressione regolare per un record. Viene applicata l'espressione regolare al testo proveniente dal file del log e vengono estrapolati i dati utili:

```
var dbid={};var dburl={};var dbres_code={};var dbreq_ms={};
var dbres_ms={};var dbbytes={};
var http=new XMLHttpRequest()
var p=/[^\s]+\s+[^\s]+\s+([^\s]+\s+([^\s]+\s+)[^\s]+\s+([0-9]+\s+
    +from request:([0-9]+\s)ms+from first response: +([0-9]+\s)
    ms+([0-9]+\s)+bytes/g
var i=0;
for (this.l=0;this.l<10;this.l++){
    http.open('GET','requests'+this.l+'.log?'+Math.random(),false)
```

```
http.sendAsBinary?http.sendAsBinary(""):http.send("")
var match = p.exec(http.responseText)
var i=0;
while(match!=null) {
    dbid[i]=match[1];
    dburl[i]=match[2];
    dbres_code[i]=parseInt(match[3]);
    dbreq_ms[i]=parseInt(match[4]);
    dbres_ms[i]=parseInt(match[5]);
    dbbytes[i]=parseInt(match[6]);
    match = p.exec(http.responseText);
    i++;
}
}
```

Come visto nella sez 4.6 al momento del caricamento del DOM, viene eseguita la funzione relativa.// La funzione identifica attraverso *getElementById()* i nodi interessati dalla modifica e crea un oggetto *tangle* che viene collocato nei nodi selezionati precedentemente.

```
window.addEventListener('domready', function () {
    var tangle = new Tangle(document.getElementById("Performance"), {
        initialize: function () { ... },
        update: function () { ... }
    })
})
```

L'inizialize può essere divisa in più blocchi logici:

- inizializzazione dei valori per regolare gli zoom, relativamente *req\_ms*, *res\_ms* e *bytes*. In particolare questi parametri regolano la larghezza in px dell'unità di misura dei relativi grafici.

```
this.unity_req_ms=0.3;
this.unity_res_ms=11;
this.unity_KB=0.05;
```

- creazione del grafico su cui verranno tracciati questi dati. Un singolo grafico è in grado di contenere fino a 62 valori della stessa categoria(es.bytes) dei record catturati. La variabile *cont* inizialmente assume il numero totale di richieste catturate dal log del Proxy. Successivamente se le richieste sono maggiori di 62, *cont* assumerà il valore 62, e in *nrich* verrà messo il numero di richieste totali. Le ultime *rich* - *cont* richieste contribuiscono ai fini del stime ma non vengono visualizzate.

```
this.cont=0;
for(var i = 0; dbid[i]!==undefined; i++){
    this.cont++;
}
if(this.cont>62){this.nrich=this.cont;this.cont=62;}
else {this.nrich=this.cont;}
```

- il contenuto di *dbid[]* viene concatenato in una stringa - *dbidStringa* - per poterla trasportare nelle altre classi *tangle*. Stessa cosa avviene per i restanti db (*dburl*, *dbres\_code*, *dbreq\_cod*, *dbres\_code*, *dbbyte*):

```
this.dbidStringa=dbid[0];
for(var i=1;i<this.cont;i++){
    d=dbid[i];
    this.dbidStringa=this.dbidStringa+','+d;
}
this.dburlStringa=dburl[0];
for(var i=1;i<this.cont;i++){
    d=dburl[i];
    this.dburlStringa=this.dburlStringa+','+d;
}
...
```

- Il ciclo for fa scorrere gli indici e vengono estratti alcuni dati utili:
  - viene identificato il record che ha il tempo di risposta massimo. In *maxres\_ms*, *maxres\_msurl*, *maxres\_msid* viene messo rispettivamente il tempo massimo di risposta, url e l'id di tale record.
  - vengono conteggiati il numero di richieste che hanno ricevuto come *res\_code* un determinato codice. Questi dati vengono messi in *cont200*, *cont401*, *cont404* ecc.
  - vengono conteggiati i byte totali che vengono trasferiti durante la cattura del Proxy. Questo dato viene messo in *totaleKB*. Alla fine del ciclo in *totaleKB* verrà inserito il valore in KB e non più in byte e fatto un arrotondamento alla seconda cifra decimale.
  - viene calcolato l'intervallo di tempo in cui avviene la cattura. Questo dato viene messo in *totalet*. Alla fine del ciclo in *totalet* verrà inserito il valore in secondi e non più in ms.
  - viene calcolato il numero di KB che vengono trasferiti nell'intervallo di tempo della cattura. Esso esprimerà quindi i KB/sec. Tale valore verrà inserito in *KBsec* con un arrotondamento alla seconda cifra decimale.
  - viene calcolato il numero di richieste servite con tempo superiore a 1 secondo. Questo dato viene messo in *nmaxres\_ms*.

```
this.cont200=0; this.cont401=0; this.cont404=0; ...
this.totaleKB=0;this.nmaxres_ms=0;
this.totalet=0; this.maxres_ms=0;
for (var i=0; i<this.cont; i++) {
    if (this.maxres_ms<dbres_ms[i])
    {
        this.maxres_ms=dbres_ms[i];
        this.maxres_msurl=dburl[i];
        this.maxres_msid=dbid[i];
    };
    if(dbres_ms[i]>60) this.nmaxres_ms++;
    if (this.totalet<(dbreq_ms[i]+dbres_ms[i]))
        {this.totalet=dbreq_ms[i]+dbres_ms[i]; };
    this.totaleKB= this.totaleKB+dbbytes[i];
    if(dbres_code[i]==200){
        this.cont200++;
    }
```

```
        }
    else if (dbres_code[i]==401){
        this.cont401++;
    }
    ...
}
this.totalet=this.totalet/1000;
this.totaleKB=Math.round( this.totaleKB/1024 *1000)/1000 ;
this.KBsec= Math.round( this.totaleKB/this.totalet*1000)/1000;
```

- Si vuole avere un dato che dice se sono stati registrati dati oppure se non c sono record. Questo viene fatto da - *scenariocont* - che è 1 se ci sono record da visualizzare, 0 altrimenti.
- Si vuole avere un dato dice ci sono record con un determinato stato. Questo viene fatto da *scenario200*, *scenario401* ecc. che assumono valore 1 se esiste un record con *res\_code=200*, *400* ecc, altrimenti 0.

```
this.scenariocont=1;
if (this.cont==0){this.scenariocont=0;};

this.scenario200=0;
if (this.cont200>0){this.scenario200=1;}

this.scenario401=0;
if (this.cont401>0){this.scenario401=1;}
...
```

L'*update* è invocata quando viene modificato il tangle. Essa può essere divisa in due parti:

- Quando viene modificato *totaleKBvar* o *totaletvar*, viene ricalcolato il *Kbrate*.

```
this.KBsecvar= Math.round( this.totaleKBvar/this.totaletvar*1000)/1000;
```

- Si vuole ottenere il numero di record che hanno tempo di risposta maggiore a un parametro - *res\_msvar* - , inizializzato con 0.01 sec, che verrà modificato via interfaccia grafica.

```
this.nmaxres_msvar=0;
for (var i=0; i<this.cont; i++) {
    if (dbres_ms[i]>(this.res_msvar*1000)) this.nmaxres_msvar++;
}
this.nmaxres_msvar=this.nmaxres_ms;
```

- Con la modifica del valore di *res\_msvar* viene ricalcolato il numero di record che vengono risposte con un tempo superiore a esso.
- Se questo numero è superiore del 30% delle richieste totali, verrà messo il flag - *scenario30percento* - a 1, 0 altrimenti.

```
for (var i=0; i<this.cont; i++) {
    if (dbres_ms[i]>(this.res_msvar*1000)) this.nmaxres_msvar++;
}

this.scenario30percento=0;
if (this.nmaxres_msvar>(this.cont/100)*30){
    this.scenario30percento=1;
}
```

Nello script vengono definite le classi *plot* che mostrano graficamente i primi - *cont* - valori dei tempi di arrivo della richiesta, dei tempi di risposta e dei Kb trasportati. Tutti e tre queste classi hanno una struttura comune.

Anche le classi di *tangle* hanno un *initialize* e una *update*. In questo caso l'*initialize* crea un oggetto con il valore del *tangle* d'invocazione. L'*update* può essere divisa in più parti logiche:

- *cont* viene inizializzato con il valore corrente di *cont* proveniente dalla classe *tangle*.
- gli array *dbid[]*, *dburl[]*, *dbres\_code[]*, *dbreq\_ms[]*, *dbres\_ms[]*, *dbbytes[]* vengono ricostruiti a partire da *dbidStringa*, *dburlStringa*, *dbres\_codeStringa*, *dburlStringa*. Queste stringhe hanno i valori iniziali del db separati da , . Con *split* viene prelevato il dato utile e lo inserisco nell'array.

```
var cont=this.tangle.getValue("cont");

var dbidStringa=this.tangle.getValue("dbidStringa");
var dbid={};
var n=dbidStringa.split(",");
for (var i=0;i<cont;i++){
    dbid[i]=n[i];
}

var dburlStringa=this.tangle.getValue("dburlStringa");
var dburl={};
n=dburlStringa.split(",");
for (var i=0;i<cont;i++){
    dburl[i]=n[i];
}
...
```

- aggiungo un *Listener* al Canvas. Quando viene sollevato l'evento *mousemove* esegue la funzione.  
Questa funzione deve identificare la barra su cui si posa il mouse e restituire nell'apposto spazio i dati relativi a tale barra.  
Il *getBoundingClientRect()* restituisce le dimensioni della finestra del client, l'attributo *clientX* restituisce la coordinata corrente del mouse.  
In *mousePos* avremo il valore della coordinata x relativa al Canvas.  
Infine - *i* - sarà l'indice della barra sui cui è il mouse. Attraverso questo indice inserirò nella sezione apposita i dati relativi al record. Verrà evidenziato il dato che viene visualizzato nel grafico.

Ad esempio se il mouse è sopra il Canvas di classe *plot\_res\_code*, il valore relativo a *res\_ms* verrà evidenziato. Se il mouse si trova, però, su una parte del Canvas più a sinistra della prima barra o oltre le prime *cont* barre verrà eseguito il ramo else dell'if.

```
el.addEventListener('mousemove', function(evt) {
    var rect = el.getBoundingClientRect();
    root = document.documentElement;
    var mousePos = evt.clientX - rect.left - root.scrollLeft;
    var i = Math.floor(mousePos / 11) - 1;
    if (i < cont && i > -1) {
        document.getElementById("spiegazione").innerHTML = 'Dettagli del record: ';
        document.getElementById("id").innerHTML = "id: " + dbid[i];
        document.getElementById("req_ms").innerHTML = ("req_ms: " + dbreq_ms[i]).bold();
        ...
    }
    else {
        document.getElementById("spiegazione").innerHTML = 'Seleziona un record';
        document.getElementById("id").innerHTML = ' ';
        document.getElementById("url").innerHTML = ' ';
        ...
    }
}, false);
```

- Viene realizzato un disegno sopra la tela. I parametri *canvasWidth*, *canvasHeight* vengono inizializzati con i valori correnti di width e height di el, parametro della funzione che rappresenta il Canvas.
- Viene creato un spazio bidimensionale e disegnato un rettangolo bianco delle stesse dimensioni del Canvas.
- Viene creato un reticolato costituito da linee grigie orizzontali, che distano l'una dall'altra 60 unità dell'asse y. Le unità vengono prelevate dal tangle.
- Viene realizzato l'istogramma vero e proprio: si tratta di tanti rettangoli di altezza pari ai valori di *dbreq\_ms[]*, *dbres\_ms* e *dbbytes[]*.
- viene disegnata una linea che congiunge le barre.

```
var canvasWidth = el.get("width");
var canvasHeight = el.get("height");
var ctx = el.getContext("2d");
var unitx = this.getUnitx();
var unity = this.tangle.getValue("unity_req_ms");

ctx.fillStyle = "white";
ctx.fillRect(0, 0, canvasWidth, canvasHeight);

ctx.strokeStyle = "#EEEEEE";
ctx.lineWidth = 0.8;
ctx.beginPath();
```

```
for (i=canvasHeight-(60*unity); i>=0; i=i-(60*unity))
{
  ctx.moveTo(0, i);
  ctx.lineTo(canvasWidth, i);
}
ctx.stroke();

var color = ["#ABCDEF", "#0095B6", "#006699", "#003399",
"#800080", "#996699", "#AA0078", "#CA278C", "#FF0080" ];
for (var i=1; i<=cont; i++)
{
  ctx.fillStyle = color[i%9]
  ctx.fillRect(unitx*i, canvasHeight - unity* dbreq_ms[i-1],
                    5, unity* dbreq_ms[i-1]) ;
}

ctx.strokeStyle = "#DCDCDC";
ctx.lineWidth = 1;
ctx.beginPath();
ctx.moveTo(0, canvasHeight)
for (var i=1; i<=cont; i++)
{
  ctx.lineTo(unitx*i, canvasHeight - unity* dbreq_ms[i-1]);
}
ctx.stroke();
}
```

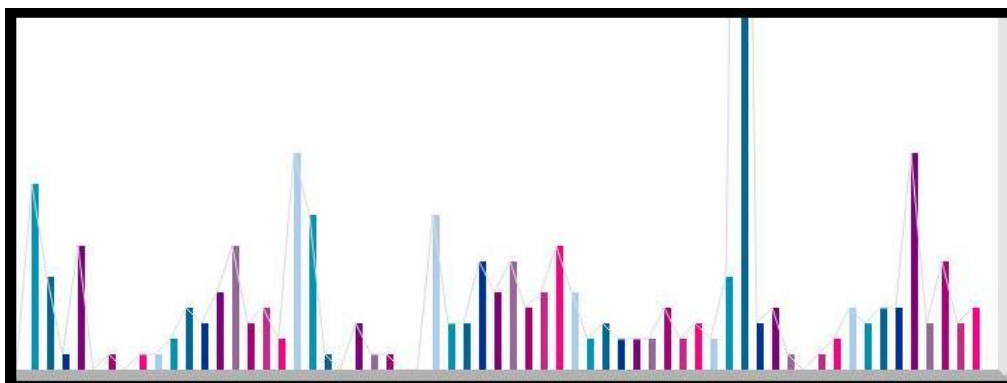


Figura 29: Rendering del Canvas

### 7.2.2 Script utilizzato da onmouseover

Questo ulteriore script è autonomo perché è uguale per ogni grafico e così può essere riutilizzato. Esso infatti aggiorna la parte dove vengono visualizzati i dati riguardanti i singoli record inserendo un messaggio con le istruzioni e eliminando i singoli dati.

```
function eraseSingleRepo(){
```

```
message = "Vai su un punto del grafico per vedere  
le informazioni in dettaglio.";  
document.getElementById(" spiegazione ").innerHTML=message;  
vuoto="";  
document.getElementById(" id ").innerHTML=vuoto;  
document.getElementById(" url ").innerHTML=vuoto;  
...  
};
```

### 7.2.3 Body

La parte coinvolta dalla classe tangle è quella con *id=Performance*. Essa si divide in due sezioni con id:

- generalrepo;
- singlerepo;

Queste due sezioni verranno trattate nello specifico nei prossimi capitoli.

### 7.2.4 generalrepo

Esso mostra delle stime sull'intero insieme di record. La sua struttura è la seguente:

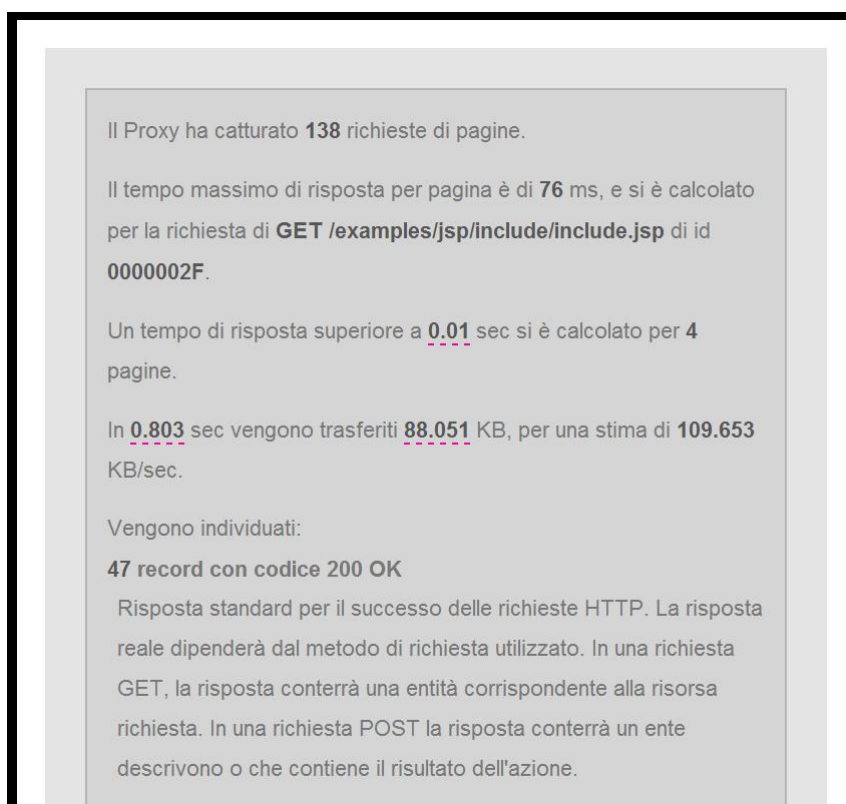


Figura 30: Report generale



```
<div data-var="scenariocont" class="TKSwitch">
  <div>Il Proxy non ha catturato nessuna attivita'.</div>
  <div id="standard"> ... </div>
</div>
```

Quando *scenariocont* assumerà valore 1, cioè non vi sono record nel log del Proxy, viene preso il primo div, che mostra il relativo messaggio. Altrimenti *scenariocont* assumerà valore uno e verrà mostrato il div con *id=standard*. Questa funzionalità è resa possibile grazie alla classe *TKSwitch* di tangle. Il div di *id=standard* è suddivisibili in più parti:

- Questo div mostra il numero totale di richieste catturate dal Proxy. La classe *ev* serve soltanto da formattazione. Evidenzia infatti, tutti i parametri che vengono generati dinamicamente.

```
<div>
  Il Proxy ha catturato
  <span data-var="nrich" class="ev"></span>
  richieste di pagine.
</div>
```

- In questa parte viene mostrato il massimo tempo di risposta, attraverso la variabile *maxres\_ms*. Viene anche specificato l'url e id relativo a tale richiesta tramite le variabili *maxres\_msurl*, *maxres\_msid*.

```
<div>
  Il tempo massimo di risposta per pagina e' di
  <span data-var="maxres_ms" class="ev"></span>
  ms, e si e' calcolato per la richiesta di
  <span data-var="maxres_msurl" class="ev" ></span>
  di id
  <span data-var="maxres_msid" class="ev"></span>.
</div>
```

- Questo div è più complesso. Quando la pagina è caricata verrà mostrato il numero di pagine che hanno avuto un tempo superiore a 10 ms. Attribuito al primo span la classe *TKAdjustableNumber*, facciamo in modo che questo parametro attraverso il trascinamento, possa cambiare assumendo valori da 0.001 a 100 con una discretizzazione millesimale.

```
<div>
  Un tempo di risposta superiore a
  <span data-var="res_msvar" class="TKAdjustableNumber ev"
  data-min="0.001" data-max="100" data-step="0.001"></span>
  sec si e' calcolato per
  <span data-var="nmaxres_msvar" class="ev" ></span>
  pagine.
</div>
```

- Questo div, grazie alla classe *TkIf*, mostra il suo contenuto solo se il valore di *scenario30percento* assume valore 1. La variabile *scenario30percento* sarà uguale a 1 solo se le pagine ad avere un tempo di risposta superiore a - *res\_msvar* - (valore che può cambiare grazie alle funzionalità descritte al punto precedente) sono più del 30% del totale.

```
<div data-var="scenario30percento" class="TKIf">
  NB. Il numero di pagine che superano
  <span data-var="res_msvar" class="ev" ></span>
  sec per essere servite , e' maggiore al 30%.
</div>
```

- Questo div quando la pagina viene caricata mostra l'intervallo di tempo totale nella quale vengono trasferiti dati (- *totaletvar* -), i Kb totali trasferiti (- *totaleKBvar* -) e l'arrotondamento del Kbrate espresso in KB/sec (- *KBsecvar* -). Grazie alla classe *TKAdjustableNumber* è possibile giocare con *totaletvar* e *totaleKBvar* per vedere come può cambiare il Kbrate.

```
<div>
  In
  <span data-var="totaletvar"
    class="TKAdjustableNumber ev"
    data-min="0.5" data-max="500" >
  </span>
  sec vengono trasferiti
  <span data-var="totaleKBvar"
    class="TKAdjustableNumber ev"
    data-min="1" data-max="3000" >
  </span>
  KB, per una stima di
  <span class="ev" data-var="KBsecvar">
  </span> KB/sec.
</div>
```

- Questo div mostra il suo contenuto solo se *scenario30percento=1*. *scenario30percento* assume valore 1 solo se ci sono più del 30% di pagine che superano - *res\_msvar* - per essere servite. La variabile *res\_msvar* può essere modificata dopo il caricamento della pagina, come mostrato al punto 3.

```
<div data-var="scenario30percento" class="TKIf">
  NB. Il numero di pagine che superano
  <span data-var="res_msvar" class="ev" ></span>
  sec per essere servite , e' maggiore al 30%.
</div>
```

- Questo div serve per capire che tipi di risposte abbiamo ricevuto dal server e in che numero. Vengono presentati così i vari scenari (- *scenario200* - , - *scenario404* -) con la relativa descrizione. Lo scenario apparirà solo se si è riscontrato almeno un record con quel codice.

```
<div>
  Vengono individuati:
  <div data-var="scenario200" class="TKIf">
    <dt><span class="ev" data-var="cont200" ></span>
      record con codice 200 OK <dt>
    <dd>Risposta standard ... risultato dell'azione. </dd>
  </div>
  ...
</div>
```

### 7.2.5 singlerepo

Questa sezione potrà assumere diverse sembianze a seconda di che evento si verifica:

- il mouse è fuori dal Canvas. Lo stato del mouse è *onmouseout*. Il Canvas esegue la funzione associata che inserisce nel div con id=spiegazione la stringa *Vai su un punto del grafico per vedere le informazioni in dettaglio.* e svuota il contenuto dei restanti div dentro *singlerepo*. Il div con id spiegazione, fornisce delle informazioni sulle azioni che si devono intraprendere per modificare il div.

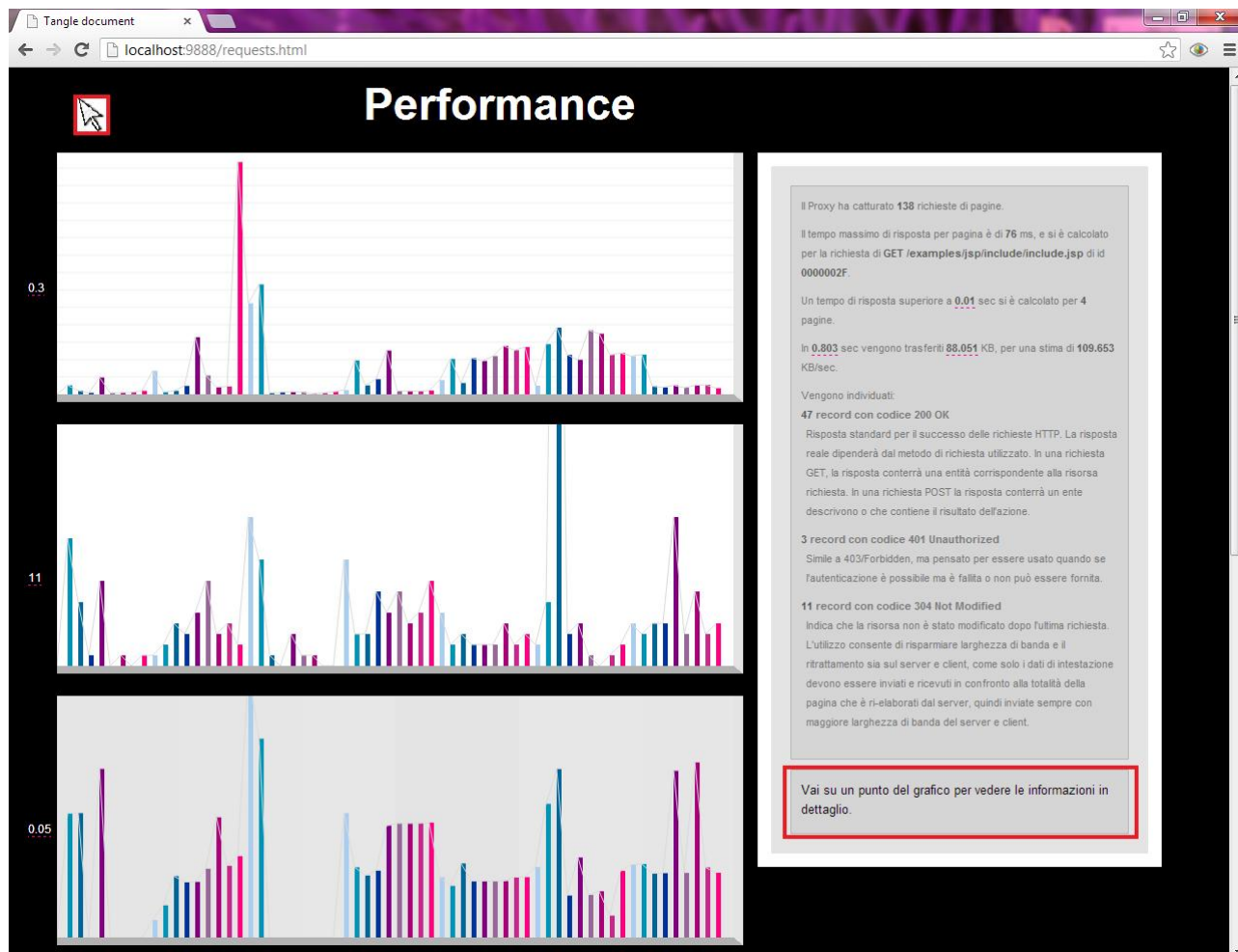


Figura 31: Mouse onmouseoverout

- il mouse è sul Canvas ma è più a sinistra della prima barra o più e oltre l'ultima barra. Lo stato del mouse è *mousemove*, ma non selezionando alcun record viene inserito un messaggio ('Seleziona un record.') del div spiegazione e tutti gli altri div svuotati.

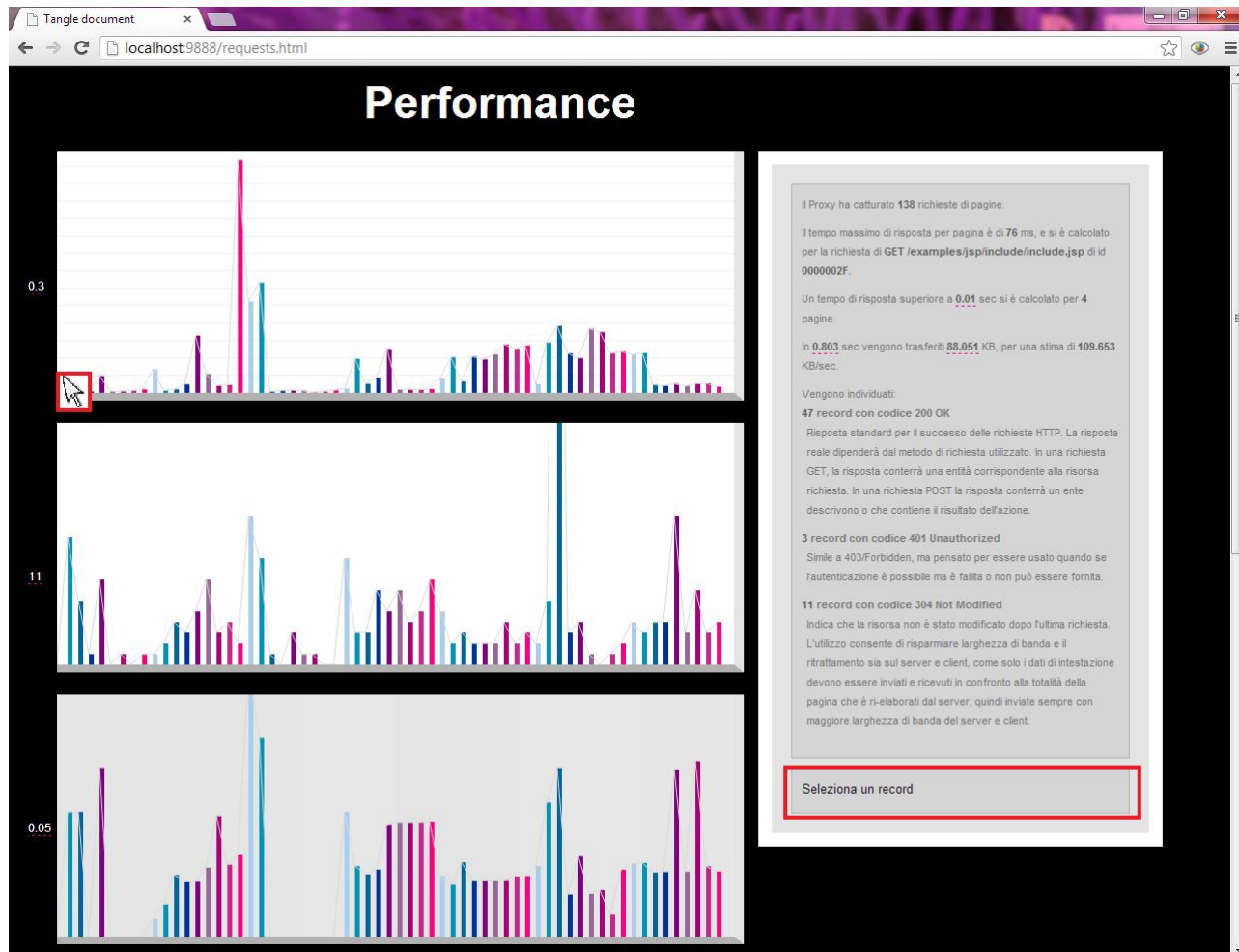


Figura 32: Mouse *mousemove* ma non selezione le barre

- il mouse è sul Canvas e su una barra. Vengono mostrati i dettagli (*id*, *url*, *res\_code* ecc.) nei rispettivi div.

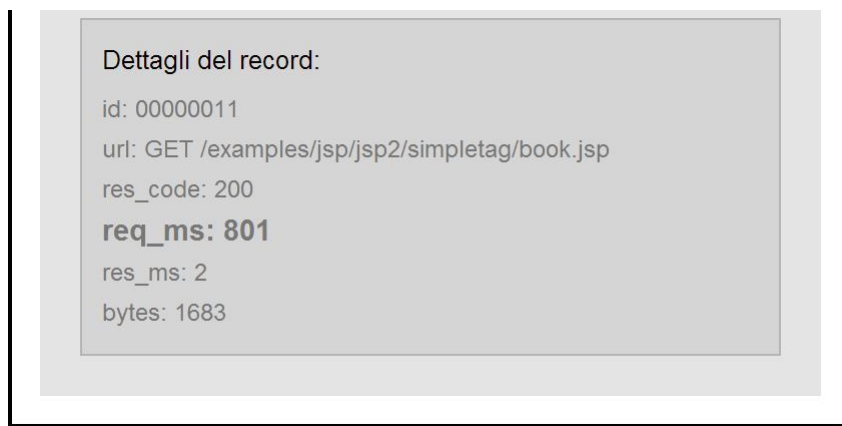


Figura 33: Report generale

```
<div id="singlerepo">
  <div id="spiegazione">
    Vai su un punto del grafico
    per vedere le informazioni
    in dettaglio.
  </div>
  <div id="id"></div>
  <div id="url"></div>
  <div id="res_code"></div>
  <div id="req_ms"></div>
  <div id="res_ms"></div>
  <div id="bytes"></div>
</div>
```

### 7.2.6 Canvas per la creazioni dei grafici

Il body si conclude con i tre Canvas per rendono visivamente gli istogrammi. Si è dotato il Canvas di uno zoom per riuscire a visualizzare sia l'andamento totale, che i valori più piccoli. Lo zoom è stato reso attraverso i valori - *unity\_req\_ms* -, - *unity\_req\_ms* -, - *unity\_bytes* - che rappresentano le unità di misure del grafico di *req\_ms*, *req\_ms*, *bytes*. Per ogni grafico l'intervallo e la discretizzazione dei valori delle unità è differente per permettere una migliore resa grafica e puntamento con il mouse.

```
<div>
<span data-var="unity_req_ms" class="TKAdjustableNumber reg"
data-min="0.2" data-max="11"
data-step="0.1" data-format="p2" style="top:220px;"></span>
```

```
<canvas id="myCanvasreq_ms" class="Plot_req_ms"
data-var="unity_req_ms" width="700" height="250"
onmouseout="eraseSingleRepo()"></canvas>
</div>
```

I Canvas sono identificati da un id per avere delle ancore per la classe tangle, inoltre appartengono a delle classi tangle per permettere il rendering delle forme.

L'attributo *data-var* sul Canvas significa che quando viene modificato il valore di data-var anche il contenuto del Canvas cambierà.

Come precedentemente detto quando il mouse è fuori dal Canvas verrà eseguita la funzione *eraseSingleRepo()*.

### 8 Specifica tecnica

Nello specifica tecnica viene presentata la struttura della pagina *request.html*. Per comodità il file contiene tutte le librerie che servono per gli script Tangle. Il file è così composto:

#### 8.1 stile

Procura lo stile alla pagina. Da menzionare è che gli elementi plot hanno un bordo di colore grigio per rappresentare gli assi delle ordinate e delle ascisse e dargli una certa profondità. Vengono formattate anche le classi Tangle originali e quelle create ad hoc per la pagina.

#### 8.2 Tangle.js

E' la libreria utilizzata e contiene la definizione della classe Tangle. Essa definisce e implementa tutti i metodi applicabili ad un oggetto Tangle: le modalità di settaggio, di acquisizione delle variabili e formattazione dei dati.

#### 8.3 mootools.js

E' il framework JavaScript che abbiamo trattato nella sezione 4.6. Esso serve per il funzionamento di *BVTouchable.js* e *TangleKis.js*.

#### 8.4 TangleKit.js

Contiene le vere e proprie classi di tangle trattate già nella sezione 4.7.2 e i vari formati di visualizzazione dei valori presenti in data-var.

#### 8.5 BVTouchable.js

Definisce due nuove classi BVTouchable e BVTouches. Esse rappresentano rispettivamente gli oggetti sensibili e quelli che sono stati toccati. Vengono definiti e ridefiniti degli eventi e rese nuove funzionalità.

#### 8.6 sprintf.js

E' una libreria che serve a tangle per presentare la variabile in vari formati.

#### 8.7 MyScriptTangle.js

E' la parte che è stata creata durante questo progetto. Essa contiene lo script per istanziare e modificare l'oggetto Tangle. Essa provvede anche le istruzioni per caricare il contenuto del log del Proxy. Vengono inoltre aggiunte delle classi Tangle per la rappresentazione grafica degli istogrammi.

#### 8.8 MyScriptMouse.js

E' uno script aggiuntivo che fornisce una funzione chiamata dall'evento *onmouseoverout* del Canvas.



### 8.8.1 body

E' la parte HTML che presenta la struttura della pagina.

## 9 Conclusioni

### 9.1 Prodotto finale

L'azienda aveva l'esigenza di mostrare in modo chiaro le performance di alcuni applicativi. Era inoltre interessata a visualizzare colli di bottiglia, punti focali e dettagli utili.

Il prodotto finale è stato una pagina web che presenta in maniera fluida, accattivante e interattiva le prestazioni di un'applicazione web. L'azienda stessa ha fornito il Proxy specializzato per catturare questi dati.

La modalità esplorabile è stata resa attraverso l'uso della libreria Tangle, che si prestava bene a tale esigenza.

### 9.2 Confronto con l'azienda

L'introduzione all'interno dell'azienda è stata impegnativa ma edificante. Il mio progetto non presupponeva un lavoro di team. Le mie attività non si sono dunque dovute integrare all'interno dei processi produttivi dell'azienda. Nonostante questo ho trovato un'ottima risposta e considerazione quando ho chiesto delucidazioni su alcuni strumenti che il tutor mi aveva chiesto di usare e quando ho dovuto collaborare con la persona che si era occupato della vecchia pagina di presentazione dei dati provenienti dal Proxy.

L'azienda si presenta come un ambiente vivace e ricco di idee. Ho trovato persone instancabili e appassionate al proprio lavoro.

Ho avuto modo soprattutto di confrontarmi con il tutor interno che, sin dall'inizio, mi ha dato una idea positiva in quanto di ampie vedute e molto disponibile al confronto e con un'attenta valutazione delle proposte altrui. Ho notato che anche con gli altri colleghi c'è un forte scambio di opinioni riguardo le strade percorribili per la risoluzione dei problemi.

Il periodo di stage presso l'azienda si è concluso con il raggiungimento di tutti gli obiettivi prefissati, l'approvazione del prodotto finale e l'intento di aggiungere Performance Repo alla struttura aziendale.

### 9.3 Conoscenze pregresse

Credo che il corso di studi di Informatica fornito da questa Università fornisca un ottimo background culturale e di esperienze per il futuro. Fornire le giuste conoscenze è senz'altro importante ma specialmente nell'ambito informatico grandemente eterogeneo e in continua evoluzione credo che l'acquisizione di un adeguato metodo di studio, di un giusto metodo di approccio ai problemi e lo standard qualitativo che si richiede siano state di gran lunga le lezioni più utili per il futuro.

Le conoscenze acquisite durante il corso di Ingegneria del Software sono state fondamentali per la buona riuscita dello stage. La fase iniziale di analisi dei requisiti è stata agevolata dall'esperienza già avuta durante il corso: ho cercato di estrapolare i punti importanti durante il colloquio in cui si sono definiti i requisiti e insieme al tutor ho tracciato una sequenza di attività che mi avrebbe portato al prodotto finito. Sicuramente questa materia è stata indispensabile per la fase di documentazione. Grazie alle varie revisioni del progetto didattico ho imparato a capire quali documenti si devono redigere e il giusto grado di dettaglio.

Basilare è stato il corso di Tecnologie Web per la natura stessa del mio prodotto. Con questo stage ho approfondito la mia conoscenza su alcune parti che, se pure facenti parte del progetto didattico non sono riuscite a approfondire in prima persona. Lo stage è stata quindi un'ottima occasione per il completamento e potenziamento delle mie conoscenze.

Tra le materie opzionali nel mio Piano di studi ho inserito Sviluppo e gestione dei progetti. Pochi mesi prima dello stage ho studiato questa materia e curato insieme ad altri colleghi

un progetto che ritengo essermi stato di grande aiuto anche per lo stage. Nonostante gran parte delle nozioni trattate sono ricollegabili a Ingegneria del software, mi sono occupata personalmente della fase di pianificazione delle attività e questo mi ha permesso di stendere insieme al tutor un adeguato Piano di Lavoro.

# Glossario

## A

### API

Application Programming Interface (Interfaccia di Programmazione di un'Applicazione) indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione.

## B

### Back-end

Parte di un'applicazione responsabile dell'elaborazione dei dati ricevuti dalla parte front-end, durante l'elaborazione la parte back-end può interagire con il database di cui ha l'accesso diretto. Dopo aver elaborato i dati, solitamente questi vengono ritornati alla parte front-end.

### Bitmap

La grafica bitmap è una tecnica utilizzata per descrivere un'immagine in formato digitale. Un'immagine descritta con questo tipo di grafica è chiamata immagine bitmap. La grafica bitmap si contrappone alla grafica vettoriale.

### Bitrate

E' il numero di bit che vengono trasmessi o elaborati per unità di tempo. Il bit rate è quantificato utilizzando i bit al secondo ( bit / s ) unità, spesso in combinazione con un prefisso SI come kilo- (kbit / s), mega-(Mbit / s), giga- (Gbit / s) o tera- (Tbit / s).

### Browser

E' un programma che consente di usufruire dei servizi di connettività in rete e di navigare sul World Wide Web, appoggiandosi sui protocolli di rete forniti dal sistema operativo (a partire da quelli di livello applicativo come HTTP, FTP ecc.) attraverso opportune API, permettendo di visualizzare i contenuti delle pagine dei siti web e di interagire con essi. Quest'ultima funzionalità è supportata dalla capacità del browser di interpretare l'HTML e di visualizzarlo in forma di ipertesto.

### business intelligence

Ci si può solitamente riferire a:

1. un insieme di processi aziendali per raccogliere ed analizzare informazioni strategiche.
2. la tecnologia utilizzata per realizzare questi processi,
3. le informazioni ottenute come risultato di questi processi.

## C

### Client

si indica una componente che accede ai servizi o alle risorse di un'altra componente detta server. In questo contesto si può quindi parlare di client riferendosi all'hardware oppure al software. Esso fa parte dunque dell'architettura logica di rete detta client-server.

Un computer collegato ad un server tramite una rete informatica (locale o geografica) ed al quale richiede uno o più servizi, utilizzando uno o più protocolli di rete è un esempio di client

hardware.

Il termine client indica anche il software usato sul computer client per accedere alle funzionalità offerte dal server.

### **Cookie**

Sono stringhe di testo di piccola dimensione inviate da un server ad un Web client (di solito un browser) e poi rimandati indietro dal client al server (senza subire modifiche) ogni volta che il client accede alla stessa porzione dello stesso dominio. Il termine cookie - letteralmente biscotto - deriva da magic cookie, concetto ben noto in ambiente UNIX che ha ispirato sia l'idea che il nome dei cookie HTTP.

Sono usati per eseguire autenticazioni automatiche, tracking di sessioni e memorizzazione di informazioni specifiche riguardanti gli utenti che accedono al server, come ad esempio siti web preferiti o, in caso di acquisti on-line, il contenuto dei loro carrelli della spesa (shopping cart).

Ogni dominio o sua porzione che viene visitata col browser può impostare dei cookies. Poiché una tipica pagina internet, ad esempio quella di un giornale online, contiene oggetti che provengono da molti domini diversi e ognuno di essi può impostare cookies, è normale ospitare nel proprio browser molte centinaia di cookies.

## **D**

### **Discretizzare**

Rappresenta il processo di trasformazione di modelli matematici ed equazioni continue nelle controparti discrete. Questo processo è spesso necessario come primo passo per esaminare un problema fisico simulandolo attraverso l'uso di un calcolatore e/o analizzandolo tramite tecniche numeriche.

## **F**

### **Framework**

Struttura di supporto su cui un software può essere organizzato e progettato. Alla base di un framework c'è sempre una serie di librerie di codice utilizzabili con uno o più linguaggi di programmazione, spesso corredate da una serie di strumenti di supporto allo sviluppo del software.

## **G**

### **GUI**

Graphic User Interface. Interfaccia grafica di un sistema operativo, costituita da: scrivania, finestre (windows), barre di scorrimento, icone, menu e puntatore.

## **H**

### **Hosting**

Servizio che consiste nell'allocare su un server web le pagine di un sito web, rendendolo così accessibile dalla rete Internet.

## **I**

### **ID**

Identificatori, sono token lessicali che denominano delle entità. Il concetto è analogo a quello di un nome. Gli identificatori sono usati estensivamente in tutti i sistemi che manipolano l'informazione. Dare un nome alle entità rende possibile riferirsi ad esse, il che è essenziale

per ogni tipo di processo.

### **Istogramma**

è la rappresentazione grafica di una distribuzione in classi di un carattere continuo. È costituito da rettangoli adiacenti le cui basi sono allineate su un asse orientato e dotato di unità di misura (l'asse ha l'unità di misura del carattere e può tranquillamente essere inteso come l'asse delle ascisse). L'adiacenza dei rettangoli dà conto della continuità del carattere. Ogni rettangolo ha base di lunghezza pari all'ampiezza della corrispondente classe; l'altezza invece è calcolata come densità di frequenza, ovvero essa è pari al rapporto fra la frequenza (assoluta) associata alla classe e l'ampiezza della classe. L'area della superficie di ogni rettangolo coincide con la frequenza associata alla classe cui il rettangolo si riferisce e per tale caratteristica gli istogrammi rappresentano un tipo di areogramma. La somma delle aree dei rettangoli è uguale alla somma delle frequenze dei valori appartenenti alle varie classi. Volendo si può scegliere di rappresentare nell'istogramma le frequenze relative (anziché le semplici frequenze assolute) delle varie classi.

L'istogramma è uno dei sette strumenti della qualità, si costruisce partendo dalla massima escursione tra i dati dividendola per gli intervalli desiderati.

## L

### **Log**

File in cui si tengono registrate le attività compiute per esempio da un'applicazione, da un server, o da un interprete di comandi.

## M

### **Markup languages**

E' un insieme di regole che descrivono i meccanismi di rappresentazione (strutturali, semantici o presentazionali) di un testo che, utilizzando convenzioni standardizzate, sono utilizzabili su più supporti. La tecnica di composizione di un testo con l'uso di marcatori (o espressioni codificate) richiede quindi una serie di convenzioni, ovvero appunto di un linguaggio a marcatori di documenti.

## O

### **Offline**

it. disconnesso. Indice di disattivazione di una connessione con un host. In Internet, ad esempio, è possibile visitare un sito off-line, salvando le pagine HTML (ed eventuali altre risorse) sul proprio hard disk per una successiva visita in locale, anche se alcune funzionalità possono risultare assenti o limitate.

### **Open source**

Si indicano i programmi il cui codice sorgente è pubblico e disponibile per l'uso e la modifica: in altre parole libero, a differenza dei programmi gratuiti che vengono distribuiti senza costi ma senza poter apportare modifiche. Il codice Open Source viene tipicamente creato tramite collaborazione di programmatori, i quali apportano man mano miglioramenti al codice sorgente originario e diffondono il risultato a beneficio della comunità.

## P

### **Parser**

Analizzatore sintattico. Algoritmo di riconoscimento sintattico di un linguaggio. Il parser,

componente del compilatore, esegue l'analisi sintattica del flusso di token (ad esempio if, then, else, ?, +, \*, /, variabili) ottenuti dall'analizzatore lessicale. Il parser prende ciascun token e gli assegna un significato, verifica la coerenza delle istruzioni del programma con la sintassi del linguaggio sorgente, ovvero trova una regola della grammatica del linguaggio che giustifichi la presenza e la posizione del token; se riscontra incongruenze il parser genera messaggi di errore o avvertimenti, altrimenti produce una forma intermedia dicodice che il generatore di codice userà per compilare l'istruzione.

### **Plot**

Trama di un grafico

### **Proxy**

è un programma che si interpone tra un client ed un server facendo da tramite o interfaccia tra i due host ovvero inoltrando le richieste e le risposte dall'uno all'altro. Il client si collega al proxy invece che al server, e gli invia delle richieste. Il proxy a sua volta si collega al server e inoltra la richiesta del client, riceve la risposta e la inoltra al client. A differenza di bridge e router, che lavorano ad un livello ISO/OSI più basso in quanto sfruttano i NAT, i proxy nella maggior parte dei casi lavorano a livello applicativo; di conseguenza un programma proxy gestisce un numero limitato di protocolli applicativi.

### **Prodotto prototipale**

Un modello approssimato o parziale del sistema che vogliamo sviluppare che simula o esegue alcune funzioni del sistema finale, realizzato allo scopo di valutarne le caratteristiche (in particolare, la usabilità). Lo sviluppo prototipale è utile:

- per avere un rapido feedback sul progetto
- per tenere il design centrato sull'utente
- per sperimentare design alternativi
- per eliminare i problemi di scrivere il codice
- per superare il problema della non completa

Normalmente, l'utente può adottare una versione più recente senza discontinuità nell'uso, ma poi non può tornare indietro.

## Q

### **Query**

Interrogazione. Nei database indica il criterio in base al quale si effettua la ricerca (select query) o la modifica (action query) di un particolare record o insieme di record.

SQL è lo standard dei database query language.

## R

### **Rendering**

Processo di calcolo per la creazione delle texture a partire dai dati geometrici (wireframe) definiti per esempio con un programma CAD per una rappresentazione realistica di oggetti 3D e 2D.

# W

### **Web Storage**

Pagine web in grado di memorizzare i dati in locale all'interno del browser dell'utente. In precedenza, questo è stato fatto con i biscotti. Tuttavia, Web Storage è più sicuro e più veloce. I dati non sono inclusi in ogni richiesta al server, ma utilizzato solo quando richiesto. E' anche possibile memorizzare grandi quantità di dati, senza influire sulle prestazioni del sito. I dati vengono memorizzati in coppie chiave / valore, e una pagina web può solo accedere ai dati memorizzati da solo.

# S

### **Scripting**

E' un linguaggio di programmazione che supporta la scrittura di script, programmi scritti per un ambiente software che automatizzano l' esecuzione di compiti che potrebbero in alternativa essere eseguiti uno ad uno da un operatore umano.

Gli script possono essere scritti ed eseguiti on-the-fly, senza autorizzazione esplicita di compilazione e collegamento passi, sono in genere creati o modificati dalla persona eseguirle. Un linguaggio di scripting è di solito interpretato dal codice sorgente o bytecode. Il termine dello script è di solito riservata ai programmi di piccole dimensioni (fino a qualche migliaio di righe di codice).

### **Server**

Componente informatico che fornisce, a livello logico e a livello fisico, un qualunque tipo di servizio ad altre componenti attraverso una rete di computer.

### **Software**

Programmi eseguibili dal calcolatore (computer). Un programma è costituito da un insieme di istruzioni e dati (algoritmi e dati) che possono essere caricati nella memoria volatile ed eseguiti dalla CPU.

### **Stringa**

Sequenza di caratteri.

# U

### **user-friendly**

Tipo di software di facile usabilità. Prende l'utente per mano e, anche grazie ad una interfaccia grafica (GUI) accattivante e all'uso di menu, pulsanti, icone, mouse, lo accompagna amichevolmente alla finalizzazione del compito prefisso.

# W

### **Web**

Il World Wide Web è un servizio di Internet che permette di navigare ed usufruire di un insieme vastissimo di contenuti (multimediali e non) e di ulteriori servizi accessibili a tutti o ad una parte selezionata degli utenti di Internet.

### **W3C**

World Wide Web Consortium, anche conosciuto come W3C, è un'organizzazione non governativa internazionale che ha come scopo quello di sviluppare tutte le potenzialità del World Wide Web. Al fine di riuscire nel proprio intento, la principale attività svolta dal W3C consiste nello stabilire standard tecnici per il World Wide Web inerenti sia i linguaggi di markup



che i protocolli di comunicazione.

### **Widget**

Componente grafico di una interfaccia utente di un programma, che ha lo scopo di facilitare all'utente l'interazione con il programma stesso. In italiano è detto congegno[1] (o elemento) grafico; può essere una vera e propria miniapplicazione.

## **Riferimenti bibliografici**

- [1] Pagina web di tangle  
<http://worrydream.com/Tangle/>
- [2] Progetto Tangle  
<https://github.com/worrydream/Tangle>
- [3] Framework Mootools  
<http://mootools.net/docs/core>
- [4] Pagina web di tangle  
<http://worrydream.com/Tangle/>
- [5] Tomcat  
<http://tomcat.apache.org/>