

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Il Facoltà di Ingegneria

Corso di INGEGNERIA INFORMATICA

Laurea in SISTEMI DISTRIBUITI

Sviluppo di app HTML5 per l'accesso a un portale scientifico XWiki

Candidato

Marco Zaccheroni

Relatore

Prof. Andrea Omicini

Correlatore

Stefano Mariani

Anno Accademico 2011/2012 - Sessione II

A Caterina, perchè non saprei come fare senza di lei,
ai miei genitori, che mi hanno permesso di arrivare fino a qui,
a tutti i miei amici, la cui provvidenziale ironia
mi ricorda sempre di non prendermi troppo sul serio.

Indice

Introduzione	7
1 Background	9
1 Le novità dell'HTML 5	9
2 Le App HTML 5	11
3 Piattaforma XWiki	12
3.1 Architettura di XWiki	13
3.2 Estendere XWiki	13
2 Internet e il mondo mobile	15
1 Difficoltà e limiti nell'accesso ad un sito classico in mobilità . .	15
2 Adattare il sito esistente o crearne uno ad hoc?	18
3 CASO REALE: <i>accedere ad un portale scientifico XWiki da dispositivo mobile</i>	20
3 Come sviluppare una app HTML5	23
1 Analisi dei requisiti e progettazione dell'app	23
2 Scelta degli strumenti	24
2.1 Il Client	24
2.2 L'Application Server	25
3 Implementazione	26
3.1 Client: <i>jQuery mobile</i>	26
3.2 Application Server: <i>Tomcat</i>	29
4 App HTML5 per l'accesso ad un portale scientifico XWiki	35
1 Analisi dei requisiti	35
2 Progettazione	36

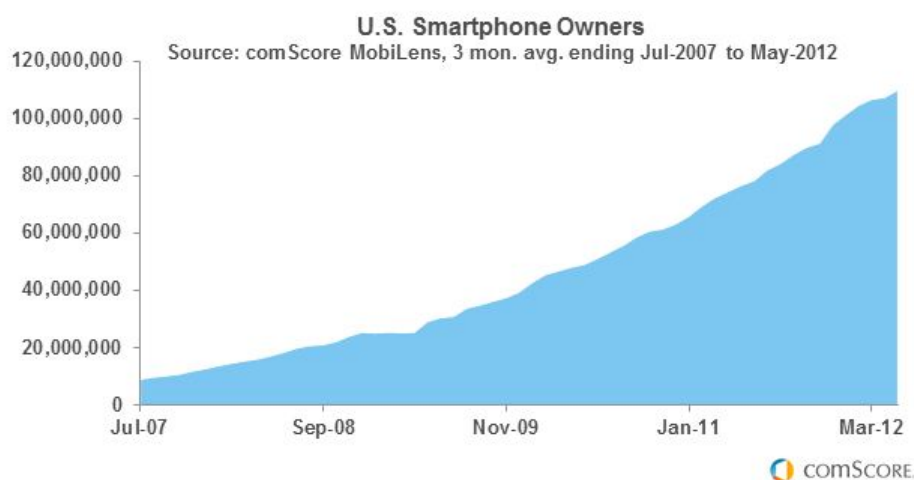
3	Implementazione	37
3.1	Configurazione Application Server	37
3.2	Client	38
3.2.1	Configurazione	38
3.2.2	Elementi comuni a tutte le pagine	40
3.2.3	Pagine dell'applicazione	42
3.2.4	Il foglio di stile	45
3.2.5	l'Application Cache	45
3.3	Il DataBase di XWiki	48
3.4	Le Servlets	49
	Conclusioni e sviluppi futuri	53
	Appendice - Codice completo	55
	Bibliografia	99

Introduzione

Fino a qualche anno fa era impensabile poter portare in tasca un dispositivo con capacità computazionali paragonabili a quelle di un computer. Tutto è cambiato durante il 2007, in una fredda giornata di gennaio un uomo chiamato Steve Jobs disse al mondo che Apple aveva intenzione di *reinventare il telefono* presentando l'iPhone.

Non è sbagliato dire che quello fu l'inizio di una vera e propria **rivoluzione** nel mondo della comunicazione mobile, si passò infatti da telefoni che si comportavano come piccoli computer a piccoli computer che si comportano *anche* come telefoni.

Gli smartphone si sono diffusi a macchia d'olio nella popolazione mondiale, un recente studio condotto dalla società di ricerca comScore ha infatti evidenziato che solo negli Stati Uniti i possessori di smartphone sono passati dall'essere appena 9 milioni nel Giugno del 2007 a circa 110 milioni nel Maggio del 2012.



Grazie alla possibilità di collegarsi ad internet in qualsiasi momento e da qualsiasi luogo, i dispositivi mobili hanno quindi portato un radicale cambiamento nel paradigma di utilizzo del web, introducendo la necessità di accedere in maniera più intuitiva e diretta ai contenuti.

Questo comporta un necessario cambiamento anche all'interno del web stesso, nella struttura delle sue pagine.

Lo scopo di questa tesi è quindi quello di mostrare in che modo sia possibile, tramite le nuove tecnologie, cambiare il web per renderlo più adatto ad una consultazione dai dispositivi mobili. Per farlo viene presentato il processo di sviluppo di una app HTML 5 per l'accesso al portale **APICe** del Dipartimento di Informatica: Scienza e Ingegneria (DISI) dell'Alma Mater Studiorum-Università di Bologna, che si appoggia sulla piattaforma XWiki.

Nell'intento di chiarire e approfondire i temi finora solo accennati, la tesi si articola come segue:

- nel capitolo 1 vengono fornite le descrizioni sulle tecnologie che sono alla base della tesi: lo stato dell'arte delle tecnologie del web e le caratteristiche di una piattaforma XWiki;
- nel capitolo 2 viene presentato il problema dell'accesso ad un sito - *classico* da dispositivo mobile e si analizzano le possibili soluzioni, applicandole anche al caso reale di XWiki;
- nel capitolo 3 invece ci si concentra su cosa significhi sviluppare un'app HTML5 per un sito mobile e quali siano gli strumenti necessari,
- infine nel capitolo 4 viene mostrato il processo di sviluppo di una applicazione HTML5 per l'accesso a un portale scientifico XWiki.

Capitolo 1

Background

*Il Web è ben lungi dall'essere fatto,
è solo in una fase farraginoso di costruzione.*

- Tim Berners-Lee -

In questo capitolo vengono introdotti i principali argomenti su cui si concentra questa tesi: si descrive dapprima che cosa è l'HTML5, quali sono le principali innovazioni che ha introdotto e come queste lo rendono il protagonista del web di domani, in seguito viene presentata la piattaforma XWiki descrivendone il funzionamento e l'architettura.

1 Le novità dell'HTML 5

L'HyperText Markup Language (HTML) è un linguaggio di markup con il quale è possibile definire la struttura di una pagina web. È stato creato al Cern nel 1991 da *Tim Berners-Lee* ed è diventato famoso nel 1994 grazie alla diffusione su larga scala di Internet.

Negli anni si è passato da un tipo di web orientato ai documenti, equivalente ad un classico libro che l'utente poteva semplicemente *consultare*, ad uno più orientato alle applicazioni, dove l'utente oltre che a consultare può anche creare contenuti, grazie a strumenti web sempre più elaborati.

L'HTML però non è riuscito a stare al passo con questo cambiamento, obbligando di fatto gli sviluppatori web ad utilizzare strumenti di terze parti, come *Flash*, plugin che per anni è stato il punto di riferimento per la creazione di animazioni, video e contenuti interattivi per il web.

Per questo motivo nasce l'**HTML 5** [?], con il preciso scopo di fornire agli sviluppatori uno strumento efficace ed efficiente per poter realizzare soluzioni innovative che non necessitino di alcun strumento *esterno*.

Nonostante il nome faccia pensare al contrario, questa innovazione non è da assegnare solo a miglioramenti del linguaggio di markup ma deriva dalle nuove dall'enorme potenziale delle nuove funzionalità del **CSS 3** [2] e delle ultime API di **javascript** [3].

Anche se è ancora abbastanza lontano dalla sua forma finale (il W3C dovrebbe rilasciare le specifiche definitive entro un anno o due) l'HTML5 porta con sé numerose novità [4] ed è utilizzato da un numero sempre maggiore di sviluppatori, anche se non completamente supportato dalla totalità dei browser.

Le principali innovazioni sono:

- semplificazione della struttura di una pagina, grazie all'introduzione di markup semantici (come `<header>`, `<nav>`, `<footer>`, `<aside>`);
- utilizzo delle pagine offline, possibilità di salvare file (e dati) in locale;
- possibilità di creare un canale di comunicazione full-duplex tra client e server, senza dover ricaricare ogni volta la pagina (Web Sockets);
- accesso diretto alle API dei device su cui viene visualizzato il sito, così da poter accedere, ad esempio, alla geolocalizzazione;
- supporto nativo per gli elementi multimediali, introduzione dei tag `<video>`, `<audio>` e `<canvas>`;
- supporto per l'esecuzione di script in background (Web Workers).

2 Le App HTML 5

Negli ultimi anni c'è stato un aumento esponenziale dei dispositivi (e quindi delle persone) connessi ad internet, questo è stato possibile anche grazie all'enorme successo dei dispositivi mobili (smartphone e tablet).

É quindi indispensabile che ci siano siti web che possano essere facilmente utilizzabili anche in mobilità, tenendo conto di tutte le differenze che ci sono con la consultazione da desktop.

Grande importanza hanno quindi le app HTML 5 (o Web App): sono siti web che mostrano funzionalità ed interfaccia simile a quelle delle applicazioni native installate sui dispositivi mobili, così da massimizzare l'esperienza di visita e la possibilità di accesso ai contenuti.

Pregi delle app HTML5:

- sono universali, non dipendono né dall'hardware né del sistema operativo del dispositivo in cui vengono utilizzate, necessitano solo di un browser che supporti adeguatamente gli ultimi standard;
- sono più economiche e veloci da sviluppare rispetto alle app native offrendo funzionalità del tutto similari;
- non hanno il bisogno di sottoporsi al processo di approvazione del market (App store, Marketplace e Windows store) quindi possono contenere qualsiasi tipo di contenuto.

Difetti:

- non hanno (ancora) il completo accesso alle API del device su cui viene utilizzata (rubrica, fotocamera, messaggi,...);
- sono meno reattive delle app native, dato che parte dell'elaborazione risiede sul server, le prestazioni dell'app dipendono molto dalla qualità della rete;
- hanno meno visibilità e diffusione, quindi minore possibilità di guadagno.

3 Piattaforma XWiki

Uno spazio **Wiki** è sito web (nella sua accezione più basilare, ovvero un *insieme di documenti ipertestuali*) in cui sono gli utenti stessi a creare e modificare i contenuti, realizzando così il vero obiettivo di Internet ovvero la condivisione della conoscenza.

Lo spazio Wiki più famoso è indubbiamente **Wikipedia**, l'enciclopedia gratuita nata nel 2001 è ora disponibile in 285 lingue diverse e conta più di **4.100.000** articoli per la sola versione Inglese (quella principale).

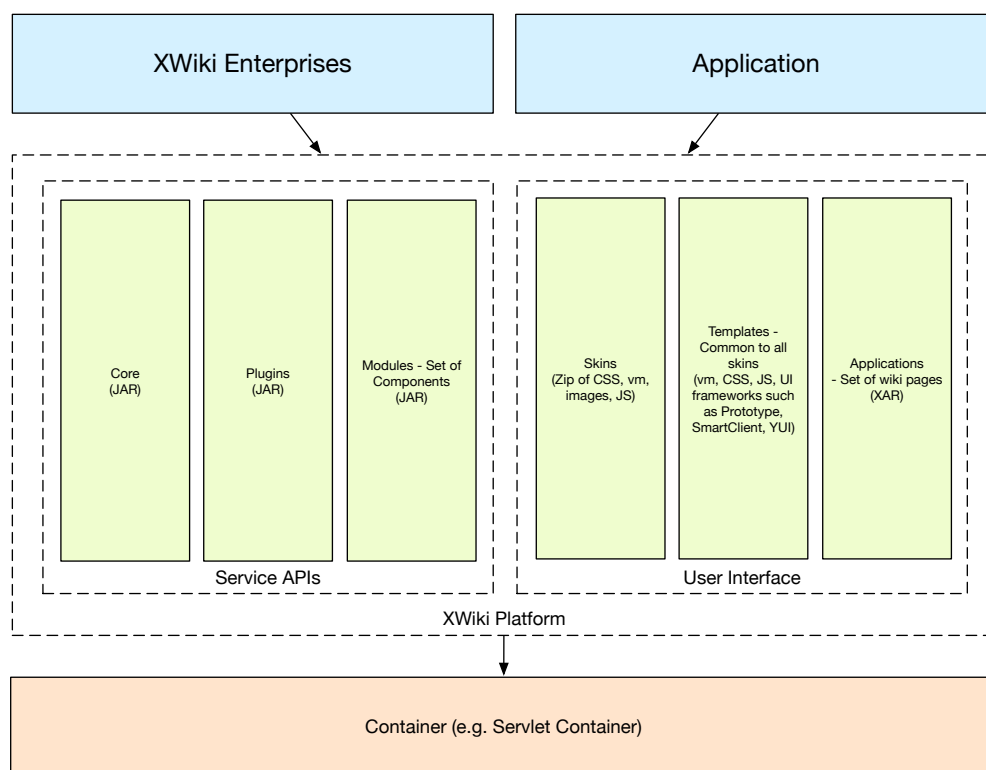
Le pagine Wiki vengono scritte tramite un linguaggio di markup creato appositamente, il *wikitext* che si basa sulla notazione `[[...]]` per inserire hyperlink ad altre pagine. Il wikitext si distingue dagli altri linguaggi dello stesso tipo perché non consente di applicare stili o di interagire con linguaggi di scripting. Le pagine vengono interpretate dal motore wiki che si occupa di tradurle in semplici pagine HTML interpretabile da qualsiasi browser.

Il vero limite di una piattaforma Wiki quindi deriva direttamente dai limiti del linguaggio con cui ne vengono create le pagine, i contenuti possono essere solo di tipo testuale e la struttura può essere solo quella di una *semplice* enciclopedia.

Per superare questi limiti nasce **XWiki**, una piattaforma open source basata su **Java**, come suggerisce il nome ha anch'essa l'obiettivo della creazione e condivisione della conoscenza ma lo mette in pratica con strumenti decisamente più potenti, le pagine possono essere create sfruttando linguaggi di scripting come **Groovy** e **Velocity**, che derivando direttamente da Java permettono potenzialmente la creazione di qualsiasi tipo di contenuto. [5]

3.1 Architettura di XWiki

Come descritto nella relativa documentazione [6], XWiki ha un'architettura di tipo modulare:



La parte fondamentale è costituita dal **Wiki Platform** dove sono contenuti i pacchetti che formano le service APIs, cuore operativo della piattaforma, e la User Interface. Al Wiki Platform si collegheranno quindi diverse altre componenti a seconda del tipo di wiki che si vuole creare. Tra queste è doveroso citare **XWiki Enterprise**, che permette di creare un wiki professionale con caratteristiche avanzate come la gestione/creazione di blog, rights management, pdf export, skinning e con un motore molto avanzato per lo scripting.

3.2 Estendere XWiki

L'aspetto più importante della piattaforma XWiki è che può essere liberamente estesa, non solo nei contenuti ma proprio nella struttura. È infatti possibile creare componenti che interfacciandosi alla XWiki Platform ne estendono le funzionalità. Questi componenti possono essere:

- Applicazioni (set di pagine);
- Plugin realizzati in Java;
- Script da inserire nelle pagine XWiki;
- Skin o rielaborazioni di quelle esistenti;
- Moduli scritti in Java.

Capitolo 2

Internet e il mondo mobile

*Il Web è più un'innovazione sociale
che un'innovazione tecnica*
- Tim Berners-Lee -

In questo capitolo si analizzano dapprima le motivazioni per cui i siti classici non sono adatti per la navigazione da dispositivi mobili come smartphone e tablet, poi si ragiona sulla necessità di creare soluzioni dedicate a questi device e le modalità con cui si possono realizzare. Infine si applicheranno questi ragionamenti al caso pratico di un portale scientifico XWiki, discutendo su come si possa adattare al mondo mobile.

1 Difficoltà e limiti nell'accesso ad un sito classico in mobilità

Per sito *classico* si intende un sito web con una forte componente statica, progettato e realizzato per essere consultato da un computer desktop (fisso o portatile). Solitamente questi siti presentano un elevato numero di scritte e di collegamenti, perlopiù testuali.

Non è difficile capire quali siano le motivazioni per cui queste tipologie di siti non sono adatti alla consultazione da dispositivo mobile:

schermi troppo piccoli anche se negli ultimi tempi gli schermi di tablet e smartphone hanno raggiunto risoluzioni molto elevate, la loro grandezza fisica resta decisamente inferiore a quella di un computer desktop, i contenuti delle pagine web risultano quindi molto difficili (se non impossibili) da consultare.

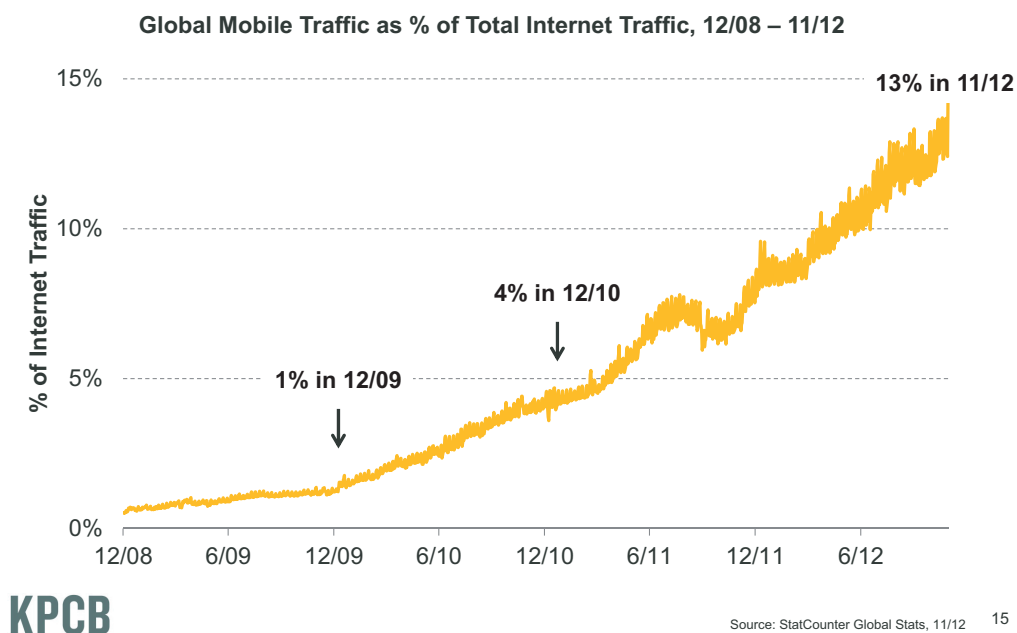
differente paradigma di interazione per anni l'interazione sul web ha seguito solamente il paradigma del **point & click**, l'utente utilizzando il mouse sposta il puntatore su di un elemento grafico o testuale, per poi cliccare ed ottenere un risultato. Con l'avvento degli schermi touch, però l'interazione è mutata, seguendo un paradigma molto più naturale, utilizzando delle dita. Bisogna notare il fatto che un dito umano è nettamente più grande del puntatore del mouse, questo rende notevolmente probabile un errore nella selezione di un hyperlink piuttosto che un altro se sono disposti in maniera molto vicina tra loro.

tecnologie non supportate a causa di limitate nella capacità di elaborazione o di decisioni aziendali, non tutti i tipi di contenuti sono supportati dai dispositivi mobili:

- tutti i device con sistema operativo iOS, non permettono la visualizzazione di contenuti Flash;
- gli archivi compressi non sono gestibili, se non appoggiandosi ad app di terze parti;
- alcune tipologie di documenti, anche multimediali (es .doc, .xls, e .wav) non sono supportati o non vengono gestiti correttamente.

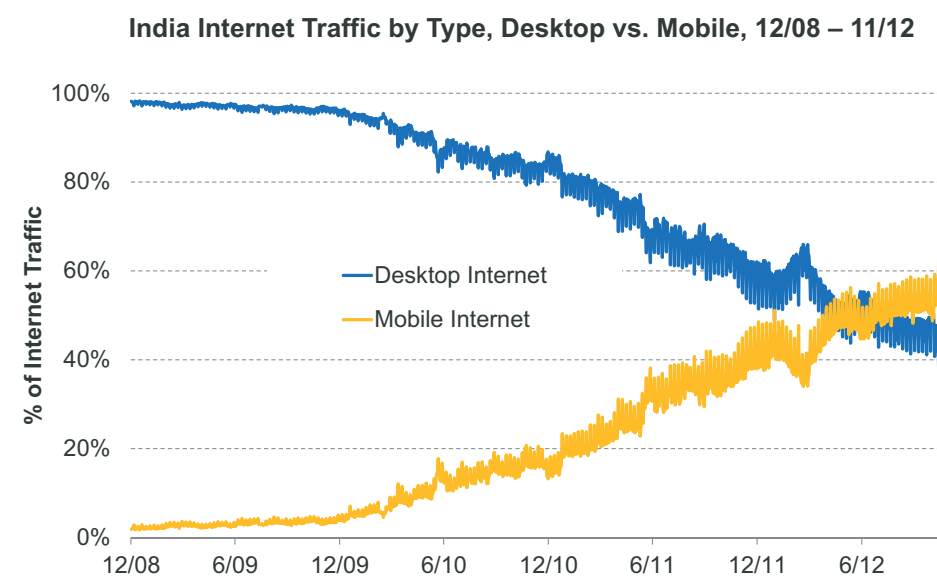
Come indicato nei grafici a pagina seguente, tratti dal rapporto *Internet Trends* pubblicato nel 2012 da Mary Meeker [7], il traffico web generato da dispositivi mobili aumenta notevolmente ogni anno, è infatti passato dal 4% a fine 2010 al 13% nel novembre 2012 e ben presto supererà quello generato dai computer desktop, come ad esempio è già successo in India.

Global Mobile Traffic Growing Rapidly to 13% of Internet Traffic

**KPCB**

15

In India, Mobile Internet Traffic Surpassed Desktop Internet Usage in May, 2012 - Other Countries to Follow...

**KPCB**

Source: StatCounter Global Stats, 11/12

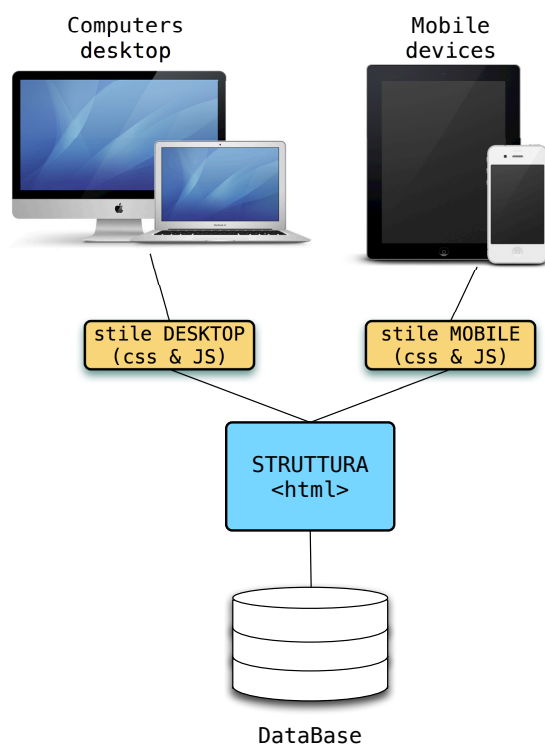
16

Questa tendenza rende quindi indispensabile l'esistenza di siti web dedicati alla navigazione da dispositivo mobile: nella prossima sezione verranno analizzate le possibili modalità con cui questi possono essere creati.

2 Adattare il sito esistente o crearne uno ad hoc?

Quando si decide di creare una versione *mobile* di un sito, bisogna innanzitutto scegliere se realizzare una **versione adattata** del sito normale, oppure se creare un sito **ad hoc**, appositamente concepito e strutturato per la visualizzazione su dispositivi mobili.

Adattare il sito esistente potrebbe essere la scelta più semplice, il lavoro si concentrerebbe esclusivamente sul design da applicare alla struttura già esistente nelle pagine. Il risultato sarà il seguente:



Si noti però che questa strada è realmente *semplice* solo se il sito da cui si parte è stato creato applicando una netta separazione tra codice (HTML) e design (CSS), se così non è stato questo processo di adattamento potrebbe rivelarsi altamente complesso.

Bisogna inoltre considerare che un sito concepito per l'ambiente desktop ha solitamente una struttura molto complessa ed articolata ed è ricchissimo di informazioni, situazione ben lontana da quella che permette un'ottimale consultazione da dispositivo mobile che richiede semplicità e sintesi .

Altra problematica risiede nelle modalità in cui vengono applicati i diversi design al sito, non tutti i dispositivi mobili infatti gestiscono correttamente gli attributi `media` e `media queries` attraverso i quali vengono attuati gli adattamenti.

Questa soluzione comporta però notevoli vantaggi in sede di manutenzione del sito, dato che si avrà un'unica piattaforma da gestire anziché due distinte.

Creare un sito dedicato In questo caso si ha invece un controllo totale sulla struttura e sul design del sito che verrà visualizzato, così da poter garantire la migliore esperienza di visita all'utente; sarà possibile effettuare una suddivisione più efficace dei contenuti ed una organizzazione più efficiente delle funzionalità offerte dal sito.

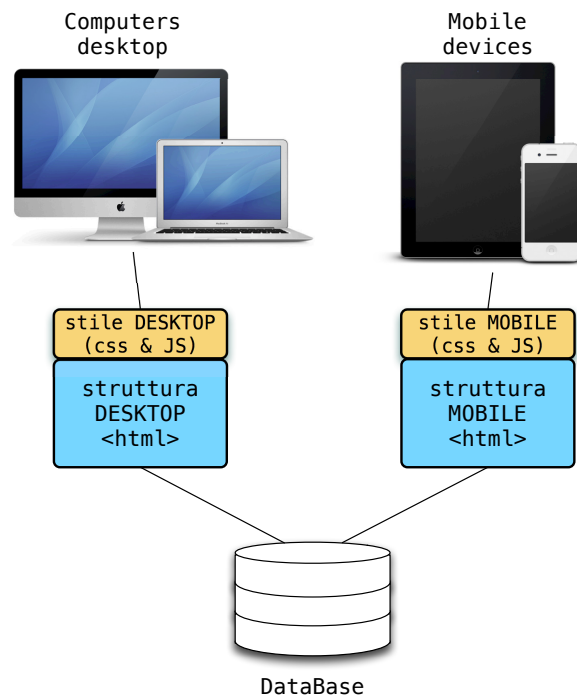
Grazie al controllo sulla struttura delle pagine sarà più semplice seguire le linee guida per una ottimale consultazione del sito da dispositivi mobili [8], le principali sono:

- **tagliare le caratteristiche**, eliminare tutte le cose che non sono fondamentali per l'uso in mobilità;
- **tagliare i contenuti**, ridurre il numero di parole e rinviare informazioni secondarie a pagine secondarie;
- **ingrandire gli elementi dell'interfaccia**, così da evitare il problema del dito grosso di cui si è anche parlato in precedenza.

I principali svantaggi di questa scelta sono in fase di creazione, dovendo costruire ex-novo il sito occorreranno tempi di sviluppo più lunghi e costi maggiori, e in fase di manutenzione dove ovviamente si avranno due distinte

piattaforme da gestire: quella desktop e quella mobile.

La struttura finale del sistema sito desktop/sito mobile sarà dunque il seguente:



3 CASO REALE: *accedere ad un portale scientifico XWiki da dispositivo mobile*

Come introdotto nel capitolo precedente, XWiki offre una piattaforma per la condivisione avanzata della conoscenza. Questo lo rende uno strumento perfetto per la creazione di un **portale scientifico** dove la comunità degli studiosi può collaborare e pubblicare i risultati delle ricerche.

Un ottimo esempio è **APICe** che, come riportato nella home page, tramite XWiki realizza un laboratorio di ricerca che ospita, fisicamente o virtualmente, singoli studiosi, gruppi di ricerca e progetti interdisciplinari, che includono studenti di dottorato, collaboratori e ricercatori, principalmente (ma non esclusivamente) appartenenti al Dipartimento di Informatica: Scienza e

Ingegneria (DISI) dell'Alma Mater Studiorum-Università di Bologna.

La grande quantità di funzionalità e di informazioni all'interno di ogni pagina e la mancanza di un supporto nativo per la visualizzazione in mobilità, costituiscono un grosso ostacolo per la consultazione di un portale XWiki, e di conseguenza di APICe, tramite smartphone e tablet.

Le alternative possibili per consentire un accesso ottimale ai contenuti della piattaforma tramite i dispositivi mobili sono:

Modificare la skin di visualizzazione adattando la struttura preesistente delle pagine così da disporre i contenuti in maniera ottimale.

Può essere realizzato tramite l'inserimento di uno script velocity all'interno del metodo `view` del motore XWiki, in modo che venga riconosciuto il tipo di dispositivo che effettua la richiesta di visualizzare la pagina, nel caso si tratti di smartphone o tablet viene poi caricata una skin differente da quella classica, progettata appositamente per le caratteristiche dei dispositivi mobili.

Creare un'app HTML 5 così da poter ricostruire liberamente la struttura delle pagine, eliminando funzioni e caratteristiche difficilmente utilizzabili in ambito mobile e organizzando in maniera più funzionale la divisione dei contenuti. Può essere realizzata utilizzando uno dei tanti framework disponibili per creare web app e sfruttando le caratteristiche del Servlet Container, che sta alla base della piattaforma XWiki, per accedere al DataBase in cui i dati sono memorizzati.

Nei seguenti capitoli di questa tesi si è scelto di mostrare un esempio implementativo di una semplice **app HTML5** per l'accesso ad un portale scientifico, per le seguenti motivazioni:

- permette una maggiore libertà nella divisione ed organizzazione dei contenuti, in modo da garantire più facilmente le caratteristiche di sintesi e semplicità per le pagine;
- consente di creare un'interfaccia utente del tutto simile a quella delle app native, rendendo la navigazione nel sito intuitiva;

- è più performante, salvando sul dispositivo mobile tutti i componenti statici dell'applicazione (immagini, file di scripting e di stile, pagine statiche) in modo da minimizzare il traffico dati.

Un esempio riguardo alla modifica di una skin di visualizzazione è invece possibile trovarlo nella tesi *Configurazione di un portale XWiki per la visualizzazione su dispositivi mobili: il caso di APICe* di **Buscarini Andrea**.

Capitolo 3

Come sviluppare una app HTML5

*Chiedersi se un computer possa pensare
non è più interessante del chiedersi
se un sottomarino possa nuotare.*

- Edsger Dijkstra -

Lo scopo di questo capitolo è quello di mostrare le tappe principali per lo sviluppo di un'app HTML5, dalla scelta della struttura in base alla tipologia di app che si è interessati a creare, la scelta degli strumenti ottimali ed i metodi in cui effettuare l'implementazione tramite piccoli esempi pratici.

1 Analisi dei requisiti e progettazione dell'app

Prima di tutto occorre stabilire *che cosa* deve essere in grado di fare l'app HTML 5 che si ha intenzione di sviluppare. Si devono decidere le caratteristiche principali e il tipo di funzionalità che essa dovrà offrire.

In seguito si passa alla definizione di *come* devono essere realizzate queste funzionalità, Questa fase prende il nome di *progettazione* ed ha l'obiettivo di determinare quale sarà la struttura operativa dell'applicazione.

Possiamo riconoscere principalmente due tipologie di app:

- **client-only**, sono applicazioni *autosufficienti*, non hanno il bisogno di prelevare contenuti da fonti esterne (es. database) ed hanno quindi una struttura semplice ad un solo livello, il client appunto.
- **client/server**, sono quelle applicazioni in cui c'è una divisione fisica (oltre che logica) tra la parte che si occupa dell'interfaccia grafica e delle funzionalità semplici e la parte che si occupa della gestione/creazione dei contenuti.

In base a come (e dove) viene implementata la *business logic* (insieme degli algoritmi che si occupano della creazione dinamica dei contenuti) queste applicazioni possono avere una struttura a due o 3 livelli logico-funzionali:

2-tier architecture il client si collega direttamente al sistema database dove sono conservati i dati.

3-tier architecture il client si occupa solamente della GUI e dell'im-paginazione dei contenuti che ottiene connettendosi ad un *application server* in cui risiede la business logic dell'app che effettua il collegamento al database.

Indubbiamente l'architettura più interessante è quella **3-tier architecture**, in quanto permette una migliore suddivisione dei compiti tra i vari componenti funzionali dell'applicazione, facilitando non solo il processo di creazione ma anche quelli successivi di gestione ed estensione.

2 Scelta degli strumenti

Una volta definite le caratteristiche e struttura dell'app si può passare alla fase di implementazione dell'applicazione; prima però è di fondamentale importanza scegliere accuratamente gli strumenti con cui realizzarne le componenti, in modo da poter coniugare al meglio prestazioni e funzionalità.

2.1 Il Client

Nei capitoli precedenti si è detto diverse volte che il principale vantaggio dello sviluppo di una web app è quello di poter realizzarne un'unica applicazione che può essere utilizzata da più dispositivi, a prescindere dall'hardware e

dal Sistema Operativo che hanno. Questo però non è completamente vero, è necessario considerare sempre quali sono le caratteristiche del browser utilizzato e la difficoltà nell'aggiornare il software da parte degli utenti con scarse capacità tecniche (che a causa della diffusione mainstream di smartphone e tablet sono la maggioranza).

Occorre quindi prestare molta attenzione alla scelta del framework con il quale implementare l'app, tra quelli esistenti attualmente, i più interessanti sono:

jQuery Mobile È il più famoso ed utilizzato, come suggerisce il nome è l'evoluzione di jQuery e consente di creare applicazioni che offrono buone funzionalità su un numero piuttosto elevato di device (non solo di ultima generazione). Utilizza un approccio basato sul *progressive enhancement*¹, dove viene data importanza fondamentale ai contenuti, per questo sono stati suddivisi i device in base alle loro capacità, i più evoluti fruiranno di una esperienza e di una interfaccia più completa rispetto a quelli obsoleti, ma tutti potranno comunque accedere ai contenuti e alle funzionalità dell'applicazione.

Sencha Touch Figlio del famoso ExtJS è un framework basato quasi completamente su JavaScript che permette di creare applicazioni molto complete con un'interfaccia che pochissimo ha da invidiare a quelle native. Tutte le caratteristiche dell'app vengono definite tramite il linguaggio di scripting, lasciando all'HTML il solo compito di includere lo script, questo permette una incredibile libertà di implementazione ma genera anche un handicap non indifferente, infatti funziona esclusivamente sui browser che implementano WebKit, quindi solo sugli ultime versioni di Safari (iOS) e Chrome (Android).

2.2 L'Application Server

Come detto parlando della fase di progettazione, l'Application Server costituisce il vero cuore dell'applicazione.

¹ **per approfondimenti:** Aaron Gustafson (2008), "*Understanding Progressive Enhancement*", <http://www.alistapart.com/articles/understandingprogressiveenhancement>;

A differenza di un classico Web Server, l'Application Server non si limita a rispondere alle richieste del client con una semplice pagina HTML, ma è capace di eseguire algoritmi di calcolo, ricerche ed elaborazioni complesse sui dati. Fornisce un vero e proprio *ambiente di esecuzione* per implementare la business logic, permettendo di utilizzare linguaggi di alto livello (come **Java**) per generare pagine dinamiche, in grado di cambiare aspetto e contenuti in base alle decisioni dell'utente.

Il più diffuso è sicuramente **Tomcat**, è un application server open source sviluppato dalla Apache Software Foundation. Basato su Java e sulle specifiche della tecnologia J2EE, è tecnicamente un **Servlet Container** e **JSP Engine**, cioè un software capace di gestire lato server le web app costituite da componenti Servlet e da pagine JSP [10].

Le **Servlet** sono classi Java che vengono eseguite ogni qualvolta l'application server riceve una richiesta particolare, costituiscono il cuore della business logic, in cui vengono effettuate tutte le operazioni destinate alla ricerca ed elaborazione dei dati.

Le **JSP** (Java Server Page) sono invece pagine in cui convivono HTML e Java, con lo scopo di creare pagine web dinamiche che cambiano in base alle necessità dell'utente.

3 Implementazione

Verranno ora mostrati due esempi di realizzazione dei due livelli funzionali, client e server, introdotti nella sezione precedente.

Per la realizzazione del lato client, si è deciso di utilizzare jQuery mobile per le sue caratteristiche *HTML5-oriented* e per la elevata compatibilità con quasi la totalità dei dispositivi attualmente sul mercato.

3.1 Client: *jQuery mobile*

jQuery Mobile sfrutta la struttura semantica delle pagine del nuovo standard HTML5 e dei suoi metadata (grazie agli attributi **data-***) per definire le varie parti dell'interfaccia, in questo modo per creare una semplice appli-

cazione sarà sufficiente scrivere del normale codice HTML5, senza il bisogno di utilizzare tecnologie particolari [9].

Per rendere più chiaro il funzionamento del framework, di seguito è riportato il codice per definire la struttura di una semplice pagina:

```
<!DOCTYPE html>
<html>
<head>
<title>Web app name</title>
<meta name="viewport"
      content="width=device-width, initial-scale=1.0, maximum-
              scale=1.0, minimum-scale=1.0, user-scalable=no">
<link rel="stylesheet"
      href="http://code.jquery.com/mobile/1.0b3/jquery.mobile-1.0
          b3.min.css" />
<script type="text/javascript"
      src="http://code.jquery.com/jquery-1.6.3.min.js"></script>
<script type="text/javascript"
      src="http://code.jquery.com/mobile/1.0b3/jquery.mobile-1.0
          b3.min.js"></script>
</head>
<body>
<div data-role="page">
  <div data-role="header" data-position="fixed" data-id="
      head">
    <h1>Page Title</h1>
  </div>
  <div data-role="content">
    <h1>Page Content</h1>
  </div>
  <div data-role="footer" data-position="fixed" data-id="
      foot" >
    <div data-role="navbar" data-iconpos="top">
      <ul>
        <li><a href="#" data-icon="home" >Button 1</a></li>
        <li><a href="page2.html" data-icon="grid" >Button 2</
            a></li>
        <li><a href="page3.html" data-icon="search" >Button 3
            </a></li>
      </ul>
    </div>
  </div>
</div>
```

```
        </div>
      </div>
</div>
</body>
</html>
```

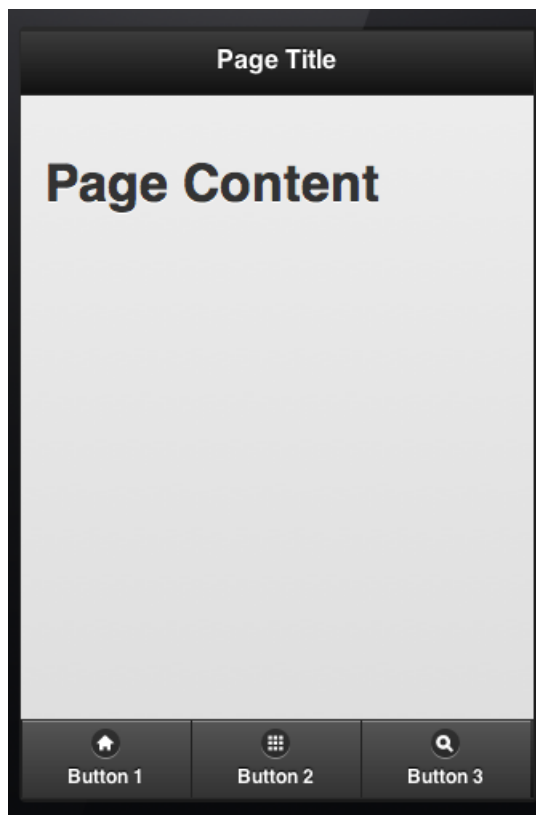
La prima porzione di codice interessante è quella relativa al meta tag **viewport**, qui vengono impostate alcune caratteristiche di default per la visualizzazione della pagina, in questo caso viene settata la larghezza della pagina deve coincidere con la grandezza dello schermo e viene disabilitato lo zoom. Vengono poi importati i file del framework, il foglio di stile del tema di default e i due file con gli funzioni javascript di jquery e jquery mobile.

Passando al body invece si può facilmente notare il largo utilizzo degli attributi **data-***, essi hanno lo scopo di impostare i metadata relativi ai vari elementi della pagina, metadata che verranno utilizzati dal framework per *interpretare* gli elementi ed assegnare loro la giusta posizione.

Analizziamo ora il significato dei principali attributi **data-*** utilizzati:

- **data-role**, assegna un valore semantico al tag di aggregazione (solitamente **div**) a cui viene assegnato:
 - **page** identifica il corpo complessivo della pagina, è infatti possibile anche specificare più pagine in uno stesso file html;
 - **header**, **header** e **navbar** creano gli elementi dell'interfaccia grafica, rispettivamente la barra in alto, quella in basso e l'oggetto che contiene i pulsanti per la navigazione tra le viste (pagine) dell'app.
- **data-position**, utilizzato esclusivamente per header e footer, assegnando lo stesso valore **data-id** su pagine diverse permette di rendere statico questo elemento, impedendo al browser di ricarcarlo;
- **data-icon** e **data-iconpos**, inseriscono delle icone ai pulsanti che vengono aggiunti alla pagina, il primo specifica il tipo di icona mentre il secondo la sua posizione.

il risultato dell'interpretazione della pagina sarà il seguente:



3.2 Application Server: *Tomcat*

Per ogni utente che si collega alla web app, Tomcat istanzia un contenitore chiamato **Sessione** per tutta la durata del suo collegamento. Quando l'utente effettua una richiesta, all'interno di questo contenitore vengono creati due oggetti: uno di nome **request** che corrisponde alla richiesta effettuata e uno di nome **response** come reazione.

A questo punto entrano in azione le Servlets, è necessario indirizzare request e response al giusto **Contesto** in cui sono presenti le risorse necessarie a soddisfare la richiesta, ci sono tre scenari possibili al momento della generazione dell'output:

- La Servlet esaudisce correttamente la richiesta e assegna a Tomcat il compito di consegnare direttamente il risultato al client, ad esempio nella forma di ipertesto HTML;

- La risorsa non è presente all'interno del contesto, Tomcat si occuperà quindi di effettuare una nuova ricerca in un contesto esterno, riavviando la procedura di input/output;
- La richiesta può essere soddisfatta all'interno del contesto indicato ma non con la risorsa richiesta (almeno non interamente) viene generata una **forward** verso un'altra risorsa (ad esempio verso una JSP che si occuperà di interpretare e integrare i risultati di una query con una struttura HTML).

Vediamo ora un piccolo esempio, supponiamo di avere un database MYSQL con una sola tabella, come rappresentato qui sotto:

DB: supereroi

Nome	Alter ego	Universo
Iron Man	Tony Stark	Marvel
Batman	Bruce Wayne	DC
Capitan America	Steve Rogers	Marvel
Green Arrow	Oliver Queen	DC
Wolverine	Logan	Marvel

tabella: Anagrafe

Come prima cosa occorre definire il contesto della risorsa, aggiungendo al file `<Tomcat Home>/conf/context.xml` le seguenti righe:

```
<Resource
  name="jdbc/eroifumetti" auth="Container"
  type="javax.sql.DataSource"
  maxActive="100" maxIdle="30" maxWait="120000"
  username="root" password="root"
  driverClassName="org.gjt.mm.mysql.Driver"
  url="jdbc:mysql://localhost/supereroi
/>
```

In questo modo sarà possibile effettuare la connessione al database dalla servlet passando il semplice riferimento al contesto.

Ora viene mostrata una semplice servlet con il compito di ottenere dal DB tutti i nomi dei supereroi e di passare questi risultati ad una pagina JSP che in cui verranno impaginati; per semplicità non si è riportata la gestione delle eccezioni:

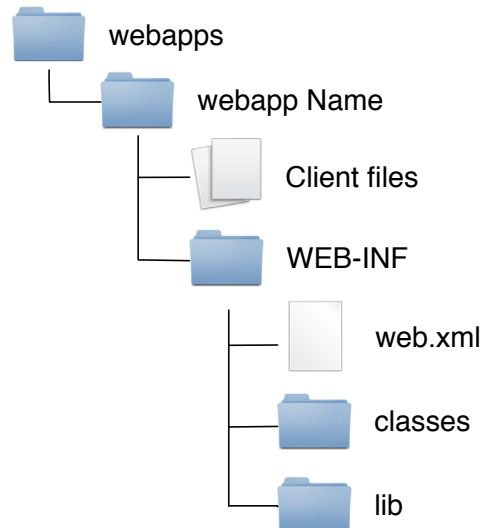
```
1 public class HeroesName extends HttpServlet {
2     public void doGet (HttpServletRequest request,
3         HttpServletResponse response) throws ServletException,
4         IOException{
5         InitialContext initCtx = new InitialContext();
6         DataSource dataSource = (DataSource) initCtx.lookup("java
7             :comp/env/jdbc/supereroi");
8         Class.forName("org.gjt.mm.mysql.Driver");
9         Connection dbconn = dataSource.getConnection();
10        PreparedStatement sql = dbconn.prepareStatement("select
11            Nome from Anagrafe;");
12        ResultSet results = sql.executeQuery();
13        while(results.next()){
14            // inserimento dei risultati in una struttura dati di
15            nome "res"
16        }
17        dbconn.close();
18        //passaggio dei risultati all'oggetto request
19        request.setAttribute("results", res);
20        //che viene rimandato alla pagina JSP dove a
21        RequestDispatcher requestDispatcher = getServletContext()
22            .getRequestDispatcher("/nomisupereoi.jsp");
23        requestDispatcher.forward(request, response);
24    }
25 }
```

Il metodo `doGet(...)` viene richiamato ogni qualvolta viene effettuata una richiesta alla servlet; osservando il codice alla riga 4 si può notare il riferimento al contesto creato in precedenza, seguito dall'esecuzione della query. Infine avviene l'assegnazione dei risultati all'oggetto *request* che viene poi consegnato alla pagina JSP dove vengono mostrati all'utente.

Affinchè la web app possa eseguire correttamente la Servlet, è fondamentale seguire una precisa organizzazione dei file all'interno di una directory in

[Tomcat Home]/webapps/.

L'organizzazione dovrà essere la seguente:



La cartella **WEB-INF** contiene tutti i file non visualizzabili dal Client per motivi di sicurezza del sistema:

- Il file **web.xml** viene definito **deployment descriptor** e permette a Tomcat di eseguire le Servlet descrivendone percorsi e argomenti necessari per l'inizializzazione.
- All'interno di **classes** sono invece presenti tutti i compilati Java per il funzionamento delle Servlet.
- Nel caso in cui Servlet e classi Java di supporto siano inseriti all'interno di uno o più archivi **Jar**, questi dovranno essere inseriti all'interno della cartella **lib**.

Per utilizzare la Servlet inserita sopra occorrerà modificare il file **web.xml** come segue:

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">
  <description>Super Heroes App</description>
```

```
<servlet>
    <servlet-name>MainServlet</servlet-name>
    <servlet-class>HeroesName</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>MainServlet</servlet-name>
    <url-pattern>/HeroesName</url-pattern>
</servlet-mapping>
</web-app>
```

Così facendo per richiamare la Servlet sarà sufficiente includerla in un attributo href, nel seguente modo:

```
<a href="/HeroesName">Heroes Name</a>
```

Concludendo, nella pagina JSP `nomisupereoi.jsp` finale, sarà necessario recuperare l'attributo `results` che gli è stato passato dalla servlet ed elaborarlo per inserirlo nel contesto della pagina:

```
<%
String[] names;
if (request.getAttribute("results") != null) {
    names = (String[]) request.getAttribute("results");
    for(int i = 0; i < names.length; i++) //elaborazione e
        stampa dei contenuti
} else {
    out.println("error");
}
%>
```


Capitolo 4

App HTML5 per l'accesso ad un portale scientifico XWiki

*La disumanità del computer sta nel fatto che,
una volta programmato e messo in funzione,
si comporta in maniera perfettamente onesta.*

- Isaac Asimov -

Lo scopo di questo capitolo è di fornire un esempio pratico di come una app HTML5 per l'accesso ad un portale scientifico possa essere implementata concretamente. L'obiettivo è quello di creare una applicazione che permetta di accedere allo **spazio pubblicazioni** del portale APICe.

1 Analisi dei requisiti

L'applicazione dovrà fornire tutti i contenuti riguardanti le pubblicazioni disponibili su APICe. Le pubblicazioni dovranno essere:

- suddivise per tipologia:
 - Articoli di rivista
 - Articoli in serie
 - Articoli di conferenze
 - Capitoli di libro
 - Volumi curati
- Indicizzate per:
 - co-autore;

- co-curatori;
- riviste & serie;
- tag.

Dovrà anche essere possibile effettuare la ricerca di una o più pubblicazioni filtrando per **parole chiave**, **anno** e **stato di pubblicazione**.

2 Progettazione

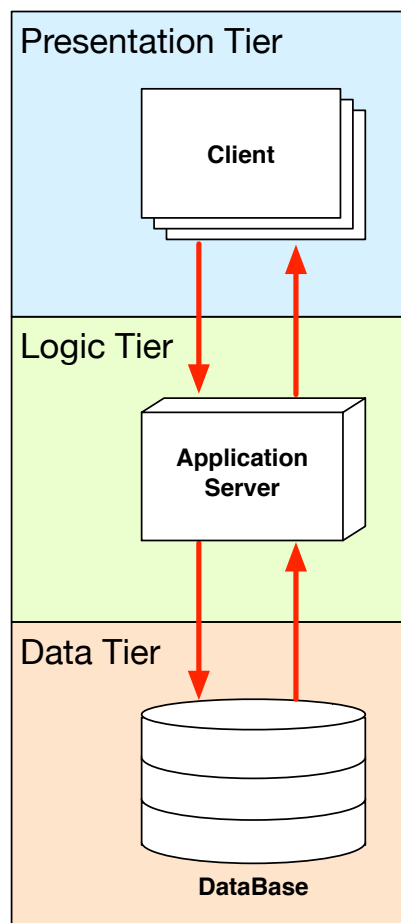
Per poter fornire le informazioni riguardo alle pubblicazioni è ovviamente necessario accedere al DataBase di XWiki, è necessario quindi realizzare un modello **client/server**.

Per permettere una implementazione più ordinata, con una adeguata divisione tra interfaccia utente e business logic, è opportuno strutturare l'applicazione secondo una **architettura three-tier**:

Presentation Tier: è il livello più esterno dell'applicazione, qui viene creata l'interfaccia utente con la principale funzione di *tradurre* i task e i risultati in qualcosa facilmente comprensibile;

Logic tier: livello che coordina l'applicazione, in cui è realizzata la **business logic**, processa le richieste dell'utente e crea le query da consegnare al DB, elabora poi i risultati e li passa al *presentation tier*;

Data Tier: qui risiede il database in cui vengono conservati i dati. Riceve le query dal *logic tier* e passa i risultati che verranno per l'elaborazione.



Il **Client** verrà realizzato utilizzando il framework **jQuery mobile** per le sue caratteristiche HTML5-oriented e per la elevata compatibilità con quasi la totalità dei dispositivi attualmente sul mercato.

Come **Application Server** verrà invece utilizzato **Tomcat**, in quanto grazie alle tecnologie JSP e Servlets permette l'implementazione della business logic tramite **Java**.

All'interno dell'applicazione verrà effettuato l'accesso diretto al DataBase **MySQL** di XWiki, senza utilizzare il middleware. Tale scelta punta semplicemente ad una più rapida e semplice implementazione della business logic.

3 Implementazione

3.1 Configurazione Application Server

Come spiegato nel capitolo precedente per un corretto funzionamento dell'applicazione è necessario modificare alcuni file di configurazione di Tomcat.

Per impostare il giusto **Contesto** in cui sono presenti le risorse (ovvero il database di xwiki) dove poter recuperare i dati necessari all'applicazione si è modificato il file [Tomcat Home]/conf/context.xml aggiungendo le seguenti righe:

```
<Resource name="jdbc/ApiceHTML5" auth="Container" type="javax
    .sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="120000"
    username="xwiki" password="xwiki" driverClassName="org.gjt.
        mm.mysql.Driver"
    url="jdbc:mysql://localhost/xwiki?
        useServerPrepStmts=false&
        amp;useUnicode=true&
        amp;characterEncoding=UTF-8&
        amp;sessionVariables=sql_mode=''"
/>
```

I valori per i parametri di configurazione del contesto sono stati prelevati direttamente dal file `hibernate.cfg.xml` di XWiki.

Inoltre occorre anche creare un file `web.xml` da inserire nella cartella `WEB-INF` nella `ROOT` dell'applicazione, il file dovrà contenere i parametri base dell'app e la descrizione delle Servlets, come segue:

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">

  <display-name>Apice Publications Space</display-name>
  <description>Simple app HTML5 to access the
    publication space inside APICe XWiki</description>

  <servlet>
    <servlet-name>MainServlet</servlet-name>
    <servlet-class>PubsList</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>MainServlet</servlet-name>
    <url-pattern>/PubsList</url-pattern>
  </servlet-mapping>

  ...

  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>

</web-app>
```

3.2 Client

3.2.1 Configurazione

Come già detto diverse volte nelle pagine precedenti, una web app può avere funzionalità ed interfaccia del tutto simili a quelli delle app native. Confi-

gurando di alcuni parametri all'interno dell'<head> è possibile permettere all'utente di aprire l'applicazione come se fosse realmente *installata* nel telefono, senza doverla ogni volta caricare con il browser. Tramite i browser di iOS e Android è infatti possibile creare dei collegamenti (chiamati Web Clip) direttamente dalla Home Screen, con tanto di icona personalizzabile, splash screen all'apertura e visualizzazione a tutto schermo.

- Per specificare la visualizzazione non scalabile a tutto schermo, senza i comandi del browser:

```
<meta name="viewport" content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, minimum-scale
    =1.0, user-scalable=no">
<meta name="apple-mobile-web-app-capable" content="yes">
```

- Per specificare diverse icone, a seconda della risoluzione del dispositivo:

```
<link rel="apple-touch-icon-precomposed" sizes="72x72"
    href="images/icon_72.png">
<link rel="apple-touch-icon-precomposed" sizes="114x114"
    href="images/icon_114.png">
<link rel="apple-touch-icon-precomposed" href="images/
    icon_57.png">
```

- Per specificare diverse splash screen, in base al tipo di dispositivo (rilevato tramite l'utilizzo di una *media query* in CSS3):

```
<!-- iPhone -->
<link rel="apple-touch-startup-image"
    media="(device-width: 320px)" href="images/
    splash_320x460.png">
<!-- iPhone (Retina) -->
<link rel="apple-touch-startup-image"
    media="(device-width: 320px) and (-webkit-device-pixel-
    ratio: 2)"
    href="images/splash_640x920.png">
<!-- iPad (portrait) -->
<link rel="apple-touch-startup-image"
    media="(device-width: 768px) and (orientation: portrait
    )"
    href="images/splash_768x1004.png">
```

Infine è necessario anche inserire i riferimenti ai file del framework ed al file di stile personalizzato:

```
<link rel="stylesheet" href="css/themes/default/jquery.mobile-1.2.0.css" />
<link rel="stylesheet" href="css/style.css" />
<script src="js/jquery.js"></script>
<script src="js/jquery.mobile-1.2.0.js"></script>
```

3.2.2 Elementi comuni a tutte le pagine

Per consentire una semplice navigazione all'interno dell'app sono stati aggiunti alcuni elementi statici:

Header barra orizzontale nella parte superiore della pagina, in cui è presente il logo dell'app e compaiono i tasti per tornare indietro o per tornare alla home, nel caso in cui non sia presente il footer.

```
<div data-role="header" data-position="fixed" data-id="
  "head" id="head">
  <a href="index.jsp" data-icon="back"
    data-iconpos="notext"
    data-rel="back" id="bt_back">back</a>

  <a href="index.jsp" data-icon="home"
    data-iconpos="notext" id="bt_home">home</a>
</div>
```

Analisi dei meta-tag significativi:

- **data-position=fixed** e **data-id**, indicano al browser di non coinvolgere l'elemento nelle animazioni per le transizioni e di non ricaricare ogni volta.
- **data-rel=back**, serve per la navigazione all'interno della cronologia, in modo che venga caricata la pagina visitata precedentemente, se questa funzionalità non è supportata dal browser verrà caricata la pagina index.jsp.

Il prodotto dell'elaborazione da parte del framework sarà:



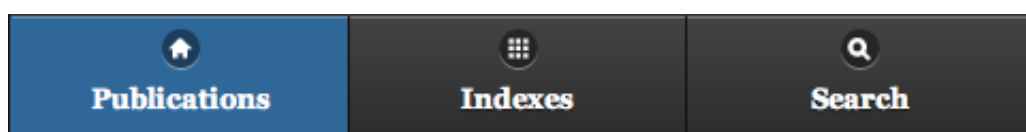
Footer barra orizzontale nella parte inferiore della pagine, contiene una navbar in cui ci sono i collegamenti per le tre viste principali dell'app. Non viene visualizzata nelle pagine in cui è presente un qualsiasi elenco di pubblicazioni.

```
<div data-role="footer" data-position="fixed" data-id="
  foot" id="foot">
  <div data-role="navbar" data-iconpos="top">
    <ul>
      <li><a href="#" data-icon="home" class="ui-btn-
        active ui-state-persist" id="bt_nav">
        Publications</a></li>
      <li><a href="indexes.jsp" data-icon="grid" id="
        bt_nav" data-transition="slide">Indexes</a></
        li>
      <li><a href="search.jsp" data-icon="search" id="
        bt_nav" data-transition="slide">Search</a></li
        >
    </ul>
  </div>
</div>
```

Analisi dei meta-tag significativi:

- `data-transition`, specifica il tipo di transizione nell'accedere alla pagina indicata nell'attributo `href`.
- `class=ui-btn-active ui-state-persist`, serve a forzare lo stato attivo del pulsante a cui viene assegnato, utilizzato in una navbar indica all'utente la pagina in cui è attualmente.

Il prodotto dell'elaborazione da parte del framework sarà:



3.2.3 Pagine dell'applicazione

L'applicazione è costituita da tre viste principali:

Publications in cui è presente una list view dove tramite la quale si può visualizzare l'elenco di tutte le pubblicazioni, oppure solamente di una particolare tipologia, questa è anche l'home page dell'applicazione;

Indexes che contiene una list view per accedere all'elenco dei co-autori, dei co-curatori, delle riviste & serie oppure dei tag;

Search da cui, tramite la compilazione di un form è possibile effettuare una ricerca per parole chiave, anno o stato di pubblicazione.

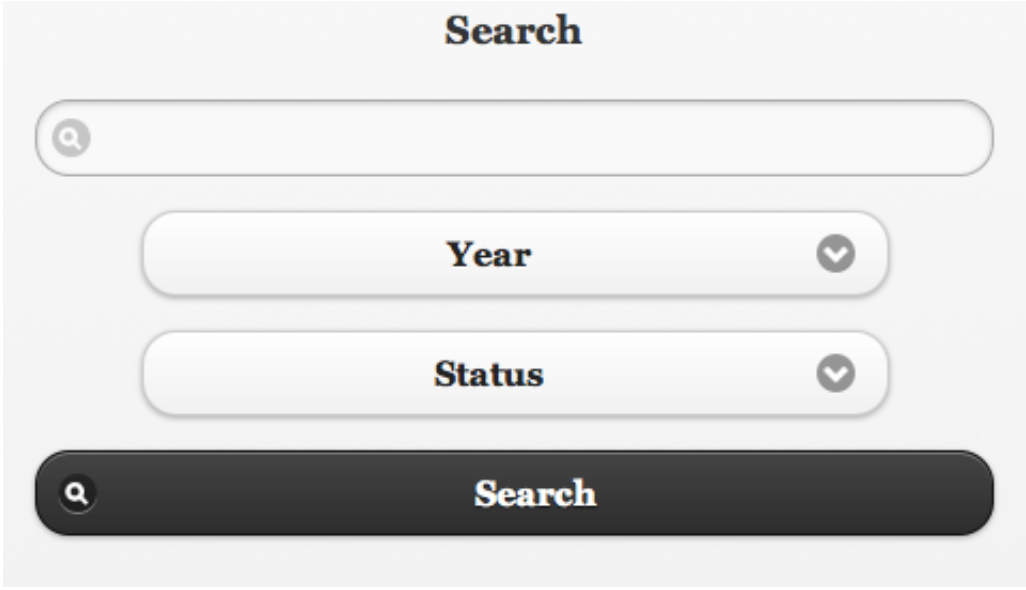
A titolo di esempio qui di seguito viene riportato parte del codice, del `content` della pagina Search, in quanto permette di fare alcune osservazioni interessanti sul funzionamento della tecnologia JSP e dell'Application Server. Per l'intero codice di tutte le pagine si rimanda all'appendice.

```
<form name="Search" action="PubsList?type=search" data-
  transition="slidefade">
  <input type="search" name="text" id="text"/>
  <div id="advanced-search">
    <div data-role="fieldcontain">
      <select name="year" id="year">
        <option value="">Year</option>
        <%
          int year = Calendar.getInstance().get(Calendar.YEAR
            );
          for (; year > 1979; year--) out.println("<option
            value=\"\" + year + \"\">\" + year + "</option>");
        %>
      </select>
    </div>
    <div data-role="fieldcontain">
      <select name="status" id="status">
        <option value="">Status</option>
        <option value="In press">In press</option>
        ...
      </select>
    </div>
  </div>
```

```
<input type="submit" value="Search" data-theme="a" data-  
    icon="search"/>  
</form>
```

Grazie all'attributo `action` della form, è possibile analizzare un esempio di comunicazione con le **Servlets**, il tutto avviene in maniera molto trasparente, è infatti sufficiente inserire il nome della servlet come se si trattasse di una semplice pagina del sito (aggiungendo eventualmente dei parametri da passargli), l'Application Server poi prenderà in carico la richiesta ed eseguirà la classe Java corrispondente. In questo esempio è inoltre possibile osservare la potenza delle JSP, queste infatti permettono di utilizzare del codice Java (delimitato da `<% %>`) all'interno dell'HTML per effettuare semplici calcoli o elaborazioni.

Il prodotto dell'elaborazione da parte del framework sarà:



Sono inoltre presenti altre due pagine, che permettono la creazione dei contenuti dinamici, in base ai risultati che vengono consegnati dalle Servlet:

Publications Founded è la pagina che dinamicamente crea una list view, i cui elementi sono i collegamenti alle singole pubblicazioni;

Publication contiene tutte le informazioni riguardo ad una specifica pubblicazione, tra cui il collegamento al file pdf (se presente), alla pagina di XWiki corrispondente e a tutte le pubblicazioni con gli stessi tag.

Per capire in che modo queste pagine dinamiche funzionano si supponga di richiedere all'applicazione, tramite il form visto sopra, tutte le pubblicazioni del 2012. La servlet **PubsList** interroga il database e inserisce i risultati in un array, che in seguito viene consegnato alla pagina **pubsfounded.jsp** per creare la listview sopracitata. Per motivi di chiarezza il codice per la creazione dinamica della listview viene mostrato in una versione semplificata, rimandando all'appendice per la versione completa e correttamente funzionante.

```
if (request.getAttribute("results") != null) {
    ListItem[] results = (ListItem[]) request.getAttribute("
        results");
    for(int i = 0; i < results.length; i++){
        out.println("<li><a href=\"PubInfo?id=" + results[i].
            getId() +
                "&name=" + results[i].getName() + "\" class=\"" +
                results[i].getName() +
                "\"><h2>" + results[i].getTitle() + "</h2><p><b>" +
                results[i].getYear() +
                " </b>" + results[i].getAuthor() + "</p></a></li>");
    }
}
```

Ottenendo un risultato simile a:



Allo stesso modo, una volta che si seleziona un elemento in particolare dalla listview, si effettua una richiesta ad una servlet, questa volta chiamata

PubInfo per le informazioni, specifiche di quella particolare pubblicazione. Viene effettuata un'altra interrogazione al database e questa volta viene passato un oggetto Java **Publication** alla pagina **publication.jsp**.

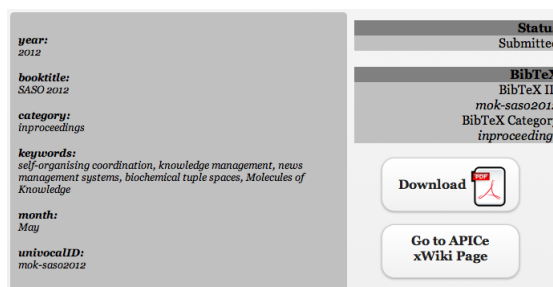
3.2.4 Il foglio di stile

Introducendo un foglio di stile personalizzato, oltre a quello del framework, è possibile sfruttare le potenzialità delle **media queries**¹ introdotte con il CSS3, queste permettono di definire all'interno di un unico file stili differenti, a seconda delle dimensioni dello schermo coi cui si accede alla pagina.

Per impostare regole particolari per la visualizzazione su schermi larghi al massimo 480px sarà sufficiente inserire il comando:

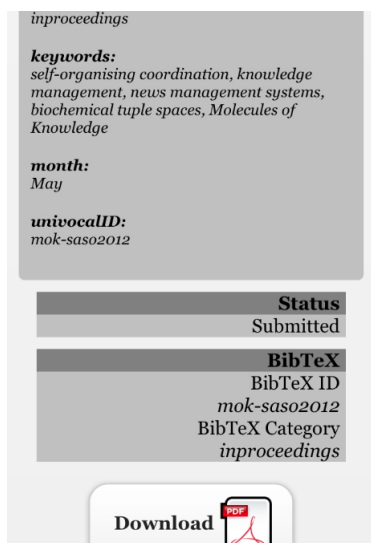
```
@media only screen and (max-device-width: 480px) {  
  ...  
}
```

É possibile vedere un semplice esempio nella pagina in cui vengono mostrati i dettagli di una pubblicazione. Nel caso questa venga consultata ad esempio da un iPhone (il cui schermo anche se realmente ha 960px di larghezza, nel CSS viene considerato da 480px con doppia densità di pixel) la sezione con i dettagli viene visualizzata in colonna, mentre se viene consultata da un dispositivo con risoluzione maggiore sarà disposta su due colonne.



Visualizzazione su iPad

¹ per approfondimenti: <http://www.w3.org/TR/css3-mediaqueries/>



Visualizzazione su iPhone

3.2.5 l'Application Cache

É una delle tante novità dell'HTML5, è un'estensione della *cache* normale. Indicando esplicitamente al browser quali file salvare, senza che questi debbano essere per forza visitati, l'appication cache permette la consultazione più o meno completa di un sito anche offline.

Il suo funzionamento è piuttosto semplice, è sufficiente creare un file all'interno del quale si specifica la lista di file che il browser deve salvare nella sua memoria e che deve mostrare anche quando si è offline, il formato è il seguente:

CACHE MANIFEST

CACHE:

index.jsp

indexes.jsp

search.jsp

error.jsp

nofound.jsp

offline.jsp

...

FALLBACK:

/ offline.jsp

Il file deve contenere obbligatoriamente nella prima riga il testo **CACHE MANIFEST** e puoi essere suddiviso in tre sezioni:

CACHE: i file indicati sotto verranno inseriti nella cache;

NETWORK: contiene i file che **non** devono essere salvati ma che richiedono la connessione internet;

FALLBACK: permette di indicare una risorsa di fallback da caricare nel caso in cui i file che corrispondono al pattern indicato (il pattern / indica tutti i file), che non sono stati inseriti nella cache, non siano raggiungibili.

Una volta creato il file è necessario specificarlo come attributo del tag **html** nella homepage dell'applicazione, così da creare la cache non appena si accede la prima volta:

```
<html manifest="cache.manifest">
```

Infine è necessario configurare l'Application Server in modo che riconosca il formato del file, per farlo occorre aprire il file [Tomcat Home]/conf/web.xml ed inserire:

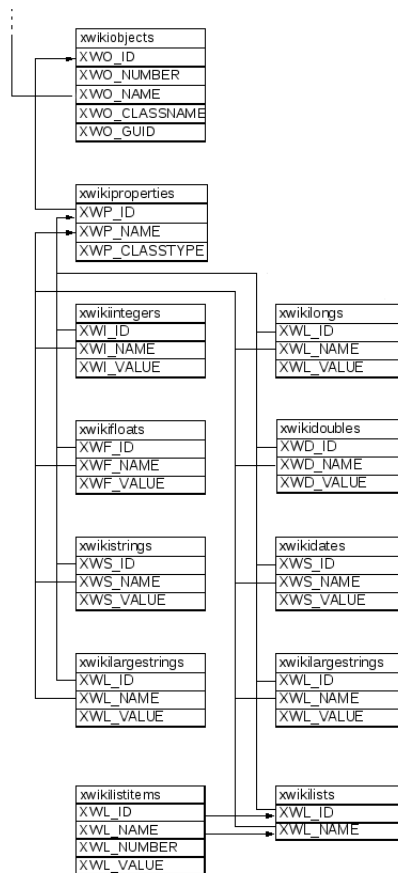
```
<mime-mapping>
  <extension>manifest</extension>
  <mime-type>text/cache-manifest</mime-type>
</mime-mapping>
```

3.3 Il DataBase di XWiki

All'interno del database di XWiki ad ogni pagina corrisponde un *oggetto*, cioè un record nella tabella **xwikiobjects**; per trovare le pubblicazioni sarà necessario controllare se il campo XWO_CLASSNAME contiene la stringa *Publications.PublicationClass*.

Il campo XWO_ID rappresenta invece l'id univoco dell'oggetto, tramite il quale si effettua il join con le altre tabelle ed è possibile recuperare tutte le informazioni riguardo ad ogni singola pubblicazione, come titolo, autori, estratto, anno, etc. All'interno poi della tabella **xwikiproperties** è possibile trovare l'intero elenco delle proprietà dell'oggetto, con il relativo classtype, necessario per sapere in quale tabella è conservato il dato, ad esempio il **titolo** di una pubblicazione è di tipo **string** ed è contenuto nella tabella **xwikistrings**.

Per una migliore comprensione di quanto appena detto, di seguito viene riportato lo schema **E/R** della parte significativa del database.



3.4 Le Servlets

All'interno dell'applicazione sono presenti tre diverse Servlet:

PubsList ha il compito di ricercare le informazioni base di ogni pubblicazione (titolo, anno e autori) per creare l'elenco dinamico all'interno della pagina **pubsfounded.jsp**;

Indexes interroga il database per indicizzare le pubblicazioni, tenendo anche il conto di quante pubblicazioni ci siano per ogni elemento di un indice;

PubInfo ha il compito ultimo di ricercare tutte le informazioni disponibili per una particolare pubblicazione che verranno poi passate alla pagina **publication.jsp** per l'impaginazione.

A titolo di esempio viene di seguito analizzato il codice relativo alla Servlet **PubsList**. Questa servlet si occupa di creare la lista delle pubblicazioni disponibile, filtrando i risultati in base alle richieste dell'utente. Le varie richieste vengono distinte tramite il parametro **type** che viene passato nell'url del link con cui viene evocata la servlet:

- type = **all**, tutte le pubblicazioni;
- type = **articles**, solo gli articoli di riviste;
- type = **series**, solo gli articoli in serie
- type = **papers**, solo gli articoli di conferenze;
- type = **books**, solo i capitoli di libro;
- type = **edite**, solo i volumi curati
- type = **search**, solo le pubblicazioni che corrispondono ai filtri settati nel form di ricerca, qui vengono passati anche i parametri **text**, **year** e **status**;
- type = **authors**, solo le pubblicazioni di un certo autore, attraverso il parametro **name**;
- type = **editors**, solo le pubblicazioni con un certo curatore, attraverso il parametro **name**;
- type = **js**, solo le pubblicazioni pubblicate su una particolare rivista o serie, attraverso il parametro **name**;
- type = **tags**, solo le pubblicazioni con un certo tag, attraverso il parametro **name**;

Come prima cosa viene recuperato dalla **request** il parametro **type** e a seconda del suo valore viene preparata la query, per semplicità di lettura è riportato solamente il caso in cui **type** sia **all**, per il codice completo si rimanda all'appendice:

```

1 public class PubsList extends HttpServlet {
2     public void doGet (HttpServletRequest request,
3         HttpServletResponse response) throws ServletException,
4         IOException{
5         String jspPage = "";
6         String select = "";
7         String from = "";
8         String condition = "";
9         String join = "";
10        String type = request.getParameter("type");
11        if (type.equals("all")){
12            select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
13                XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
14                XWL_VALUE as AUTHOR";
15            from = "xwikistrings as S, xwikilongs as L,
16                xwikilargeststrings as LS, xwikiobjects as O";
17            condition = "S.XWS_NAME = 'title' and L.XWL_NAME = '
18                year' and LS.XWL_NAME = 'author' and O.XWO_CLASSNAME
19                = 'Publications.PublicationClass' and O.XWO_NAME <>
20                'PublicationClassTemplate'";
21            join = "S.XWS_ID = O.XWO_ID and L.XWL_ID = O.XWO_ID and
22                LS.XWL_ID = O.XWO_ID";
23        }
24    }

```

Successivamente viene indicato il **contesto** specificato nel file di configurazione di Tomcat, si effettua la connessione al database e si esegue la query, si noti l'apertura del blocco try/catch per poter gestire le eccezioni:

```

15 try{
16     if (!select.equals("")){
17         InitialContext initCtx = new InitialContext();
18         DataSource dataSource = (DataSource) initCtx.lookup("java:
19             comp/env/jdbc/ApiceHTML5");
20         Connection dbconn;
21         ResultSet results;
22         PreparedStatement sql;
23         Class.forName("org.gjt.mm.mysql.Driver");

```

```
23     try {
24         dbconn = dataSource.getConnection();
25         sql = dbconn.prepareStatement("select " + select + " from
            " + from + " where " + condition + " and " + join + "
            order by YEAR desc;");
26         results = sql.executeQuery();
```

In seguito si controlla che la query abbia prodotto risultati, in caso positivo questi vengono inseriti in un array di `ListItem`, oggetto creato ad hoc per contenere i campi significativi delle pubblicazioni, che viene assegnato come attributo della `request`. Nel caso in cui non ci siano risultati invece si assegna alla variabile che indica la pagina verso cui inviare il *forwarding* il riferimento ad una pagina di errore: `nofound.jsp`.

```
27     if(results.next()){
28         results.last();
29         int size = results.getRow();
30         int i = 0;
31         ListItem[] res = new ListItem[size];
32         results.beforeFirst();
33         while(results.next()){
34             id = results.getString("ID");
35             name = results.getString("NAME");
36             title = results.getString("TITLE");
37             year = results.getString("YEAR");
38             author = results.getString("AUTHOR");
39             ...
40             //impostazione parametri da consegnare alla JSP
41             res[i] = new ListItem();
42             res[i].setId(id);
43             res[i].setName(name);
44             res[i].setTitle(title);
45             res[i].setYear(year);
46             res[i].setAuthor(author);
47             i++;
48         }
49         ...
50         dbconn.close();
51         request.setAttribute("results", res);
52         jspPage = "/pubsfounded.jsp?type=" + type;
53     }else{
```

```
54     jspPage = "/nofound.jsp";  
55 }
```

Infine c'è la gestione delle eccezioni, nel caso si entri nel ramo del `catch` si imposta il caricamento di una pagina dove viene mostrato il messaggio di errore. Alla fine del metodo avviene poi il *forwarding* verso il riferimento che si trova dentro alla variabile `jspPage` passando i parametri **request** e **response**.

```
56         }catch (SQLException s){  
57             request.setAttribute("error", s);  
58             jspPage = "/error.jsp";  
59         }  
60     }  
61     }catch (Exception err){  
62         jspPage = "/error.jsp";  
63     }  
64     request.setAttribute("from", "PubsList");  
65     RequestDispatcher requestDispatcher = getServletContext()  
66         .getRequestDispatcher(jspPage);  
67     requestDispatcher.forward(request, response);  
68 }
```

Conclusioni e sviluppi futuri

Giunti al termine è opportuno analizzare il lavoro svolto, cercando di individuare i risultati raggiunti e quelli mancati: in modo da offrire interessanti spunti per delle future argomentazioni che integrino ulteriormente il presente trattato confermandone o confutandone il contenuto.

Innanzitutto è bene ricordare l'obiettivo della tesi, al fine di poter valutare correttamente se esso sia stato raggiunto o meno: come è stato già detto, il proposito principale era quello di mostrare quali siano i metodi e le tecnologie per rendere il web più adatto al nuovo tipo di utilizzo che ne viene fatto, tramite i dispositivi mobili. Quello appena enunciato può essere considerato lo scopo di più alto livello della tesi, ma non è il solo. Un secondo obiettivo, non meno importante e sicuramente più concreto, è la presentazione del processo di sviluppo di una app HTML5 per i dispositivi mobili, capace di semplificare la navigazione e massimizzare la fruizione dei contenuti anche all'interno di uno spazio complesso ed articolato come un portale scientifico XWiki.

Si confida nel raggiungimento di tali obiettivi, ad opera dei seguenti capitoli:

- il capitolo 2 dovrebbe aver descritto in maniera esaustiva le motivazioni che rendono necessaria la creazione di soluzioni web mirate al mondo mobile e le modalità secondo cui queste possono essere realizzate;
- il capitolo 3 dovrebbe aver definito in maniera non ambigua gli strumenti essenziali allo scopo;
- infine il capitolo 4 dovrebbe aver mostrato un utile esempio di sviluppo di una app HTML5 per una piattaforma XWiki.

Analizzando il risultato ultimo della tesi, ovvero l'applicazione per l'accesso al portale scientifico APICe realizzata in HTML5, è opportuno effettuare alcune osservazioni.

Al fine di usufruire di un ottimale grado di libertà nell'organizzazione dei contenuti, "depurandoli" dalla struttura delle pagine di XWiki, è stato necessario prelevarli direttamente dalla fonte, ovvero dal DataBase. Questo però ha implicato la totale rinuncia a tutte le funzionalità di modifica e condivisione che vengono fornite dai vari componenti della piattaforma. È bene notare quindi che, seppure l'applicazione garantisce l'accesso ai contenuti del portale scientifico, questo viene effettuato in modalità "*sola lettura*".

A tale proposito si possono individuare alcune interessanti prospettive verso cui sviluppare ulteriormente questa applicazione:

- Tramite l'inserimento del supporto al middleware e gestendo adeguatamente l'accesso al DataBase, potrebbe essere possibile aggiungere direttamente all'interno dell'app la creazione di contenuti e il supporto ad alcune delle funzionalità XWiki che si appoggiano al DB;
- Considerando le caratteristiche fortemente *HTML-based* del framework jQuery mobile, potrebbe essere possibile implementarlo direttamente all'interno di una skin di XWiki, in modo tale che l'interazione tra le due tecnologie possa portare ad una comunque ottimale organizzazione dei contenuti.

Appendice - Codice completo

In questa appendice si riporta il codice completo di ogni componente creato nello sviluppo dell'app HTML5 all'interno del capitolo 4.

Pagine del Client

index.jsp

Homepage dell'applicazione, da cui è possibile accedere all'elenco delle pubblicazioni, divise per tipologia.

```
1 <!DOCTYPE html>
2 <html manifest="cache.manifest">
3 <head>
4   <title>Publications</title>
5   <meta name="viewport" content="width=device-width, initial-
      scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-
      scalable=no">
6   <meta name="apple-mobile-web-app-capable" content="yes">
7   <meta name="apple-mobile-web-app-status-bar-style" content=
      "black">
8   <link rel="apple-touch-icon-precomposed" sizes="72x72" href
      ="images/icon_72.png">
9   <link rel="apple-touch-icon-precomposed" sizes="114x114"
      href="images/icon_114.png">
10  <link rel="apple-touch-icon-precomposed" href="images/
      icon_57.png">
11
12  <!-- iPhone -->
13  <link rel="apple-touch-startup-image"
```

```
14         media="(device-width: 320px)"
15         href="images/splash_320x460.png">
16 <!-- iPhone (Retina) -->
17 <link rel="apple-touch-startup-image"
18       media="(device-width: 320px) and (-webkit-device-
19             pixel-ratio: 2)"
20       href="images/splash_640x920.png">
21 <!-- iPad (portrait) -->
22 <link rel="apple-touch-startup-image"
23       media="(device-width: 768px)
24             and (orientation: portrait)"
25       href="images/splash_768x1004.png">
26 <!-- iPad (landscape) -->
27 <link rel="apple-touch-startup-image"
28       href="images/splash_1024x748.png"
29       media="screen and (min-device-width: 481px)
30             and (max-device-width: 1024px) and (orientation:
31             landscape)" />
32 <!-- iPad (Retina, portrait) -->
33 <link rel="apple-touch-startup-image"
34       media="(device-width: 768px)
35             and (orientation: portrait)
36             and (-webkit-device-pixel-ratio: 2)"
37       href="images/splash_1536x2008.png">
38 <!-- iPad (Retina, landscape) -->
39 <link rel="apple-touch-startup-image"
40       href="images/splash_2048x1496.png"
41       media="screen and (min-device-width: 481px)
42             and (max-device-width: 1024px) and (orientation:
43             landscape)
44             and (-webkit-min-device-pixel-ratio: 2)" />
45
46 <link rel="stylesheet" href="css/themes/default/jquery.
47       mobile-1.2.0.css" />
48 <link rel="stylesheet" href="css/style.css" />
49 <script src="js/jquery.js"></script>
50 <script src="js/jquery.mobile-1.2.0.js"></script>
51 </head>
52 <body>
53 <div data-role="page" class="pages" id="home">
```

```

51     <div data-role="header" data-position="fixed" data-id="
        head" id="head">
52         
53     </div><!-- /header -->
54
55     <div data-role="content">
56         <h1>Publications</h1>
57         <p>Research and development work in APICe involves the
            use of a large amount of scientific publications,
            and produces a lot of original scientific materials.
            </p>
58         <p>In order to help the many activities in APICe, a
            handy and precise representation of such a huge
            collection of pieces of scientific literature is
            certainly more than useful. </p>
59         <p>This Publications space in the APICe space is meant
            to address such needs, by containing the scientific
            material considered relevant for the scientific and
            technical activities in APICe, either published
            directly by APICe people or by other researchers in
            the world.</p>
60         <br>
61         <br>
62         <ul data-role="listview">
63             <li><a href="PubsList?type=all" data-transition="
                slidefade">All Publications</a></li>
64             <li><a href="PubsList?type=articles" data-transition="
                slidefade">Journal Articles</a></li>
65             <li><a href="PubsList?type=series" data-transition="
                slidefade">Series Articles</a></li>
66             <li><a href="PubsList?type=papers" data-transition="
                slidefade">Conferences Paper</a></li>
67             <li><a href="PubsList?type=books" data-transition="
                slidefade">Book Chapters</a></li>
68             <li><a href="PubsList?type=edited" data-transition="
                slidefade">Edited Volumes</a></li>
69         </ul>
70     </div><!-- /content -->
71     <div data-role="footer" data-position="fixed" data-id="
        foot" id="foot">
72         <div data-role="navbar" data-iconpos="top">

```

```

73     <ul>
74         <li><a href="#" data-icon="home" class="ui-btn-active
              ui-state-persist" id="bt_nav">Publications</a></li>
75         <li><a href="indexes.jsp" data-icon="grid" id="bt_nav
              " data-transition="slide">Indexes</a></li>
76         <li><a href="search.jsp" data-icon="search" id="
              bt_nav" data-transition="slide">Search</a></li>
77     </ul>
78 </div><!-- /navbar -->
79 </div><!-- /footer -->
80 </div><!-- /page -->
81
82 </body>
83 </html>

```

indexes.jsp

Pagina da cui si accede all'elenco delle pubblicazioni, indicizzate per co-autori, co-curatori, riviste & serie e tag.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Indexes</title>
5     <meta name="viewport" content="width=device-width, initial-
              scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-
              scalable=no">
6     <meta name="apple-mobile-web-app-capable" content="yes">
7     <meta name="apple-mobile-web-app-status-bar-style" content=
              "black">
8
9     <link rel="apple-touch-icon-precomposed" sizes="72x72" href
              ="images/icon_72.png">
10    <link rel="apple-touch-icon-precomposed" sizes="114x114"
              href="images/icon_114.png">
11    <link rel="apple-touch-icon-precomposed" href="images/
              icon_57.png">
12
13    <link rel="stylesheet" href="css/themes/default/jquery.
              mobile-1.2.0.css" />

```

```

14     <link rel="stylesheet" href="css/style.css" />
15     <script src="js/jquery.js"></script>
16     <script src="js/jquery.mobile-1.2.0.js"></script>
17 </head>
18 <body>
19 <div data-role="page" class="pages" id="home">
20
21     <div data-role="header" data-position="fixed" data-id="
22         head" id="head">
23         
24     </div><!-- /header -->
25
26     <div data-role="content">
27         <h1 align="center">Indexes</h1>
28         <ul data-role="listview">
29             <li><a href="Indexes?type=authors" data-transition="
30                 slidefade">Authors</a></li>
31             <li><a href="Indexes?type=editors" data-transition="
32                 slidefade">Editors</a></li>
33             <li><a href="Indexes?type=js" data-transition="
34                 slidefade">Journals & Series</a></li>
35             <li><a href="Indexes?type=tags" data-transition="
36                 slidefade">Tags</a></li>
37         </ul>
38     </div><!-- /content -->
39
40     <div data-role="footer" data-position="fixed" data-id="
41         foot" id="foot">
42         <div data-role="navbar" data-iconpos="top">
43             <ul>
44                 <li><a href="index.jsp" data-icon="home" id="bt_nav"
45                     data-transition="slide" data-direction="reverse">
46                     Publications</a></li>
47                 <li><a href="#" data-icon="grid" id="bt_nav" class="
48                     ui-btn-active ui-state-persist">Indexes</a></li>
49                 <li><a href="search.jsp" data-icon="search" id="
50                     bt_nav" data-transition="slide">Search</a></li>
51             </ul>
52         </div><!-- /navbar -->
53     </div><!-- /header -->

```

```
45
46 </div><!-- /page -->
47
48 </body>
49 </html>
```

search.jsp

Contiene il form per effettuare la ricerca di una o più pubblicazioni in particolare.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Search</title>
5   <meta name="viewport" content="width=device-width, initial-
      scale=1">
6   <meta name="apple-mobile-web-app-capable" content="yes">
7   <meta name="apple-mobile-web-app-status-bar-style" content=
      "black">
8
9   <link rel="stylesheet" href="css/themes/default/jquery.
      mobile-1.2.0.css" />
10  <link rel="stylesheet" href="css/style.css" />
11  <script src="js/jquery.js"></script>
12  <script src="js/jquery.mobile-1.2.0.js"></script>
13  <%@ page
14    import = "java.util.Calendar"
15    %>
16 </head>
17 <body>
18 <div data-role="page" class="pages" id="home">
19
20   <div data-role="header" data-position="fixed" data-id="
      head" id="head">
21     
22   </div><!-- /header -->
23
24   <div data-role="content">
25     <h1>Search</h1>
```

```

26      <form name="Search" action="PubsList?type=search"
      data-transition="slidefade">
27      <input type="search" name="text" id="text"/>
28      <div id="advanced-search">
29          <div data-role="fieldcontain">
30              <select name="year" id="year">
31                  <option value="">Year</option>
32          <%
33              int year = Calendar.getInstance().get(Calendar.
                  YEAR);
34              for (; year > 1979; year--) out.println("<
                  option value=\"\" + year + \"\">\" + year + "</
                  option>");
35          %>
36          </select>
37      </div>
38
39      <div data-role="fieldcontain">
40          <select name="status" id="status">
41              <option value="">Status</option>
42              <option value="In press">In press</option>
43              <option value="Proof">Proof</option>
44              <option value="Camera-ready sent">Camera-ready
                  sent</option>
45              <option value="Revised">Revised</option>
46              <option value="Accepted">Accepted</option>
47              <option value="Accepted with revision">
                  Accepted with revision</option>
48              <option value="Rejected">Rejected</option>
49              <option value="Submitted">Submitted</option>
50              <option value="Draft">Draft</option>
51              <option value="Note">Note</option>
52          </select>
53      </div>
54      </div>
55      <input type="submit" value="Search" data-theme="a"
          data-icon="search"/>
56  </form>
57
58  </div><!-- /content -->
59

```

```

60     <div data-role="footer" data-position="fixed" data-id="
        foot" id="foot">
61     <div data-role="navbar" data-iconpos="top">
62     <ul>
63         <li><a href="index.jsp" data-icon="home" id="bt_nav"
            data-transition="slide" data-direction="reverse">
            Publications</a></li>
64         <li><a href="indexes.jsp" data-icon="grid" id="bt_nav
            " data-transition="slide" data-direction="reverse"
            >Indexes</a></li>
65         <li><a href="#" data-icon="search" id="bt_nav" class=
            "ui-btn-active ui-state-persist">Search</a></li>
66     </ul>
67     </div><!-- /navbar -->
68
69     </div><!-- /header -->
70
71 </div><!-- /page -->
72
73 </body>
74 </html>

```

pubsfounded.jsp

Crea dinamicamente la lista delle pubblicazioni tramite l'attributo **results** che gli viene passato dalle Servlets.

```

1 <!DOCTYPE html>
2 <html>
3 <%@ page
4     import = "java.io.*"
5     import = "java.sql.*"
6     import = "apice.ListItem"
7 %>
8 <head>
9     <title>
10     <%
11         //impostazione del titolo della pagina in base al
            contenuto
12         String pub_type = request.getParameter("type");
13         String name = request.getParameter("name");

```

```
14         if (pub_type.equals("all")){
15             out.println("All Publications");
16         } else if (pub_type.equals("articles")){
17             out.println("Journal Articles");
18             } else if (pub_type.equals("series")){
19                 out.println("Series Articles");
20         } else if (pub_type.equals("papers")){
21             out.println("Conferences Papers");
22         } else if (pub_type.equals("books")){
23             out.println("Book Chapters");
24         } else if (pub_type.equals("edited")){
25             out.println("Edited Volumes");
26         } else if (pub_type.equals("authors")){
27             if (name == null){
28                 out.println("Authors Index");
29             } else {
30                 String[] n = name.split(",");
31                 out.println(n[0] + "'s Publications");
32             }
33         } else if (pub_type.equals("editors")){
34             if (name == null){
35                 out.println("Editors Index");
36             } else {
37                 String[] n = name.split(",");
38                 out.println("Edited by " + n[0]);
39             };
40         } else if (pub_type.equals("js")){
41             if (name == null){
42                 out.println("J&S Index");
43             } else {
44                 out.println("Published in " + name);
45             }
46         } else if (pub_type.equals("tags")){
47             if (name == null){
48                 out.println("Tags Index");
49             } else {
50                 out.println("Publications with tag: " + name);
51             }
52         }
53     %>
54 </title>
```

```

55 <meta charset="ISO-8859-1">
56 <meta name="viewport" content="width=device-width, initial-
    scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-
    scalable=no">
57 <meta name="apple-mobile-web-app-capable" content="yes">
58 <meta name="apple-mobile-web-app-status-bar-style" content=
    "black">
59
60 <link rel="stylesheet" href="css/themes/default/jquery.
    mobile-1.2.0.css" />
61 <link rel="stylesheet" href="css/style.css" />
62 <script src="js/jquery.js"></script>
63 <script src="js/jquery.mobile-1.2.0.js"></script>
64
65 </head>
66 <body>
67 <div data-role="page" class="pages" id="home">
68     <div data-role="header" data-position="fixed" data-id="
        head" id="head">
69         <a href="index.jsp" data-icon="back" data-iconpos="
            notext" data-rel="back" id="bt_back">back</a>
70         
71         <a href="index.jsp" data-icon="home" data-iconpos="
            notext" id="bt_home">home</a>
72     </div><!-- /header -->
73
74     <div data-role="content">
75         <h1>
76         <%
77             if (pub_type.equals("all")){
78                 out.println("All Publications");
79             } else if (pub_type.equals("articles")){
80                 out.println("Journal Articles");
81             } else if (pub_type.equals("series")){
82                 out.println("Series Articles");
83             } else if (pub_type.equals("papers")){
84                 out.println("Conference Papers");
85             } else if (pub_type.equals("books")){
86                 out.println("Book Chapters");
87             } else if (pub_type.equals("edited")){
88                 out.println("Edited Volumes");

```

```

89         } else if (pub_type.equals("authors")){
90             if (name == null){
91                 out.println("Author's Index");
92             } else {
93                 out.println("Publications by <b>" + name + "</b>"
94                     );
95             }
96         } else if (pub_type.equals("editors")){
97             if (name == null){
98                 out.println("Editor's Index");
99             } else {
100                 out.println("Publications edited by <b>" + name +
101                     "</b>");
102             };
103         } else if (pub_type.equals("js")){
104             if (name == null){
105                 out.println("Journal & Series Index");
106             } else {
107                 out.println("Published in \"<b>" + name + "\"</b>"
108                     );
109             };
110         } else if (pub_type.equals("tags")){
111             if (name == null){
112                 out.println("Tags Index");
113             } else {
114                 out.println("Publications with tag \"<b>" + name
115                     + "\"</b>");
116             };
117         }
118     }
119     %>
120 </h1>
121 <ul data-role="listview" data-filter="true">
122     <%
123         ListItem[] results;
124         String from = "";
125         if (request.getAttribute("from") != null) {
126             from = (String) request.getAttribute("from");
127         }
128         if (request.getAttribute("results") != null) {
129             results = (ListItem[]) request.getAttribute("
130                 results");

```

```

125         for(int i = 0; i < results.length; i++){
126             if (from.equals("Indexes")){
127                 out.println("<li><a href=\"PubsList?type=" +
                             pub_type + "&name=" + results[i].getName() +
                             "\"><h2>" + results[i].getName() + "</h2><
                             span class=\"ui-li-count\">" + results[i].
                             getCount() + "</span></a></li>");
128             } else if (from.equals("PubsList")) {
129                 out.println("<li><a href=\"PubInfo?id=" +
                             results[i].getId() + "&name=" + results[i].
                             getName() + "\" class=\"" + results[i].
                             getName() + "\"><h2>" + results[i].getTitle
                             () + "</h2><p><b>" + results[i].getYear() + "
                             </b>" + results[i].getAuthor() + "</p></a
                             ></li>");
130             }
131         }
132         } else {
133             out.println("error - missing query results");
134         }
135         %>
136     </ul>
137 </div><!-- /content -->
138
139 </div><!-- /page -->
140
141 </body>
142 </html>

```

publication.jsp

Ha il compito finale di *costruire* la pagina della pubblicazione.

```

1 <!DOCTYPE html>
2 <html>
3 <%@ page
4     import = "java.io.*"
5     import = "java.util.ArrayList"
6     import = "java.util.LinkedHashMap"
7     import = "apice.Publication"
8 %>

```

```
9  <%
10      Publication pub;
11      if (request.getAttribute("publication") != null) {
12          pub = (Publication) request.getAttribute("publication")
13              ;
14      } else {
15          out.println("error - missing publication details");
16          pub = new Publication();
17      }
18  %>
19  <head>
20      <title><%=pub.getName()%></title>
21      <meta charset="ISO-8859-1">
22      <meta name="viewport" content="width=device-width, initial-
23          scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-
24          scalable=no">
25      <meta name="apple-mobile-web-app-capable" content="yes">
26      <meta name="apple-mobile-web-app-status-bar-style" content=
27          "black">
28
29      <link rel="stylesheet" href="css/themes/default/jquery.
30          mobile-1.2.0.css" />
31      <link rel="stylesheet" href="css/style.css" />
32      <script src="js/jquery.js"></script>
33      <script src="js/jquery.mobile-1.2.0.js"></script>
34
35  </head>
36  <body>
37      <div data-role="page" class="pages" id="home">
38
39          <div data-role="header" data-position="fixed" data-id="
40              head" id="head">
41              <a href="indexes.html" data-icon="back" data-iconpos="
42                  notext" data-rel="back" id="bt_back">back</a>
43              
44              <a href="index.jsp" data-icon="home" data-iconpos="
45                  notext" id="bt_home">home</a>
46          </div><!-- /header -->
47
48          <div data-role="content">
```

```

42     <h1><%=pub.getTitle()%></h1>
43     <h2 class="author" >
44     <%
45         String author = pub.getAuthor();
46         String editor = pub.getEditor();
47         if (!author.equals("")){
48             author = author.replace(", ", " ");
49             author = author.replace(" and ", ", ");
50             author = author.replace(" AND ", ", ");
51             out.println(author);
52         } else if (editor !=null) {
53             editor = editor.replace(", ", " ");
54             editor = editor.replace(" and ", ", ");
55             editor = editor.replace(" AND ", ", ");
56             out.println(editor + " (eds)");
57         }
58     %>
59 </h2>
60     <hr>
61     <%
62         if (!pub.getAbstract().equals("")) out.println("<p>" +
63             pub.getAbstract() + "</p><hr>");
64     %>
65     <div class="details">
66         <div class="pub_details">
67             <%
68                 ArrayList<String> not_print = new ArrayList<
69                     String>();
70                 not_print.add("title");
71                 not_print.add("status");
72                 not_print.add("doi");
73                 not_print.add("url");
74                 not_print.add("issn");
75                 not_print.add("issn-online");
76                 not_print.add("isbn");
77                 not_print.add("pdf-local");
78                 not_print.add("url-pdf");
79
78                 LinkedHashMap<String,String> map = pub.getDetails
79                     ();
79                 for(String str:pub.getDetailsLabels()){

```

```

80         String value = map.get(str);
81         if (str.equals("editor")){
82             value = value.replace(", ", " ");
83             value = value.replace(" and ", ", ");
84             value = value.replace(" AND ", ", ");
85         }
86         if (!not_print.contains(str)) out.println("<h4><b
            >" + str + " :</b><br>" + value + "</h4>");
87     }
88     %>
89     </div>
90     <div id="boxbar">
91         <div class="box" id="status">
92             <div class="titlebox">
93                 <b>Status</b>
94             </div>
95             <%=pub.getStatus()%>
96
97         </div>
98     <%
99         if (pub.hasWEB()){
100             out.println("<div class=\"box\" id=\"web\"><div
                class=\"titlebox\"><b>Web</b></div>");
101             if (pub.getDoi() != null) out.println("DOI <a href
                =\"http://dx.doi.org/\" + pub.getDoi() + "\"
                target=\"_blank\">" + pub.getDoi() + "</a><br>");
                ;
102             if (pub.getUrl() != null) out.println("<a href=\""
                + pub.getUrl() + "\" target=\"_blank\">Publisher
                's Page</a>");
103             out.println("</div>");
104         }
105         if (pub.hasBiBlio()){
106             out.println("<div class=\"box\" id=\"biblio\"><div
                class=\"titlebox\"><b>Biblio</b></div>");
107             if (pub.getIssn() != null) out.println("ISSN<br>" +
                pub.getIssn() + "<br>");
108             if (pub.getIssn_online() != null) out.println("ISSN
                on-line<br>" + pub.getIssn_online() + "<br>");
109             if (pub.getIsbn() != null) out.println("ISBN<br>" +
                pub.getIsbn() + "<br>");

```

```

110         out.println("</div>");
111     }
112
113     if (pub.hasBibTex()){
114         out.println("<div class=\"box\" id=\"bibtex\"><div
115             class=\"titlebox\"><b>BibTeX</b></div>");
116         out.println("BibTeX ID<br><i>" + pub.getUniID() + "
117             </i><br>");
118         out.println("BibTeX Category<br><i>" + pub.
119             getCategory() + "</i><br>");
120         out.println("</div>");
121     }
122
123     %>
124
125     <div class="buttons">
126     <%
127         if (pub.hasPDF()){
128             pub.setUrlPDFlocal("http://apice.unibo.it/xwiki/
129                 bin/download/Publications/" + pub.getName() +
130                 "/"");
131             out.println("<a href=\"\" + pub.getPDF() + \"\"
132                 class=\"pdf\" target=\"_blank\" data-role=\"
133                 button\" data-inline=\"true\" id=\"btn_img\">
134                 Download <img src=\"images/pdf.png\" alt=\"PDF
135                 \" id=\"pdf_img\"></a>");
136         }
137
138     %>
139
140     <a href="http://137.204.107.27/xwiki/bin/view/
141         Publications/<%=pub.getName()%>" class="to_xwiki"
142         target="_blank" data-role="button" data-inline="
143         true" id="btn_img"><div>Go to APICe<br>xWiki Page<
144         /div></a>
145
146 </div>
147 </div>
148 </div>
149 <hr>
150 <%
151     if (!pub.getTags().isEmpty()){
152         out.println("<div id=\"tags\"><b>Tags:</b><br>");

```

```
138         for(String tag:pub.getTags()){
139             out.println("<a href=\"PubsList?type=tags&name=\" +
                tag + "\" class=\"tag\" data-role=\"button\"
                data-inline=\"true\">" + tag + "</a>");
140         }
141         out.println("</div><hr>");
142     }
143     %>
144
145     </div><!-- /content -->
146
147 </div><!-- /page -->
148
149 </body>
150 </html>
```

nofound.jsp

Pagina evocata nel caso in cui la query effettuata nel database risulti vuota.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Nothing Found</title>
5     <meta charset="ISO-8859-1">
6     <meta name="viewport" content="width=device-width, initial-
        scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-
        scalable=no">
7     <meta name="apple-mobile-web-app-capable" content="yes">
8     <meta name="apple-mobile-web-app-status-bar-style" content=
        "black">
9     <link rel="stylesheet" href="css/themes/default/jquery.
        mobile-1.2.0.css" />
10    <link rel="stylesheet" href="css/style.css" />
11    <script src="js/jquery.js"></script>
12    <script src="js/jquery.mobile-1.2.0.js"></script>
13
14 </head>
15 <body>
16 <div data-role="page" class="pages" id="home">
```

```

17     <div data-role="header" data-position="fixed" data-id="
        head" id="head">
18     <a href="index.jsp" data-icon="back" data-iconpos="
        notext" data-rel="back" id="bt_back">back</a>
19     
20     <a href="index.jsp" data-icon="home" data-iconpos="
        notext" id="bt_home">home</a>
21 </div><!-- /header -->
22
23 <div data-role="content">
24     <div id="not-found-message">
25         
26         <div id="text-msg">
27             <h1 id="message"><b>Nothing Found</b></h1>
28             <h1 id="subs">go back and refine your search</h1>
29         </div>
30     </div>
31 </div><!-- /content -->
32
33 </div><!-- /page -->
34
35 </body>
36 </html>

```

offline.jsp

Viene caricata su indicazione del file manifest dell'app, nel caso in cui ci sia qualche risorsa non accessibile.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Offline</title>
5     <meta charset="ISO-8859-1">
6     <meta name="viewport" content="width=device-width, initial-
        scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-
        scalable=no">
7     <meta name="apple-mobile-web-app-capable" content="yes">
8     <meta name="apple-mobile-web-app-status-bar-style" content=
        "black">

```

```

9      <link rel="stylesheet" href="css/themes/default/jquery.
      mobile-1.2.0.css" />
10     <link rel="stylesheet" href="css/style.css" />
11     <script src="js/jquery.js"></script>
12     <script src="js/jquery.mobile-1.2.0.js"></script>
13
14 </head>
15 <body>
16 <div data-role="page" class="pages" id="home">
17     <div data-role="header" data-position="fixed" data-id="
        head" id="head">
18         <a href="index.jsp" data-icon="back" data-iconpos="
            notext" data-rel="back" id="bt_back">back</a>
19         
20         <a href="index.jsp" data-icon="home" data-iconpos="
            notext" id="bt_home">home</a>
21     </div><!-- /header -->
22
23     <div data-role="content">
24         <div id="not-found-message">
25             
26             <div id="text-msg">
27                 <h1 id="message"><b>No Connection</b></h1>
28                 <h1 id="subs">check your settings</h1>
29             </div>
30         </div>
31     </div><!-- /content -->
32
33 </div><!-- /page -->
34
35 </body>
36 </html>

```

File .manifest per l'Application Cache

```

1  CACHE MANIFEST
2  # ver 10-12-12
3
4  CACHE:

```

```
5 index.jsp
6 indexes.jsp
7 search.jsp
8 error.jsp
9 nofound.jsp
10 offline.jsp
11
12 js/jquery.js
13 js/jquery.mobile-1.2.0.js
14
15 css/style.css
16 css/themes/default/jquery.mobile-1.2.0.css
17 css/themes/default/images/ajax-loader.gif
18 css/themes/default/images/icons-18-black.png
19 css/themes/default/images/icons-18-white.png
20 css/themes/default/images/icons-36-black.png
21 css/themes/default/images/icons-36-white.png
22
23 images/logo.png
24 images/notfound.png
25 images/error.png
26 images/offline.png
27 images/pdf.png
28
29 FALLBACK:
30 / offline.jsp
```

Foglio di stile

```
1
2 * {
3     font-family: georgia, serif;
4 }
5
6
7 #head {
8     height: 70px;
9 }
10
```

```
11 #head img{
12     margin:0px auto;
13     display:block;
14     width: auto;
15     height: 70px;
16 }
17
18 #head {
19     background: #2F6799;
20     color: white;
21 }
22
23 .ui-btn-active{
24     background: #2F6799;
25     color: white;
26 }
27
28 #bt_back{
29     margin-top: 20%;
30     margin-left: 5%;
31 }
32
33 #bt_home{
34     margin-top: 20%;
35     margin-right: 5%;
36 }
37
38 #not-found-message {
39     width: 300px;
40     height: 200px;
41     position: absolute;
42     left: 50%;
43     top: 50%;
44     margin: -100px 0 0 -150px;
45 }
46
47 #notfound-img{
48     margin: 0px auto;
49     display: block;
50     width: 200px;
51     height: auto;
```

```
52     padding-bottom: 20px;
53 }
54
55 #message {
56     text-align: center;
57     font-size: xx-large;
58     padding: 0px;
59 }
60
61 #subs {
62     text-align: center;
63     font-size: medium;
64     color: #5B5B5B;
65 }
66
67 #err-msg {
68     text-align: center;
69 }
70
71 #err-img {
72     margin: 0 auto;
73     display: block;
74 }
75
76 [data-role="content"] h1 {
77     text-align: center;
78     padding-top: 0px;
79     margin-top: 0px;
80     padding-bottom: 10px;
81     font-size: large;
82 }
83
84 [data-role="content"] h2 {
85     font-size: small;
86     font-style: italic;
87     color: #5B5B5B;
88     background-color: transparent;
89 }
90
91 h2.author {
92     text-align: center;
```

```
93 }
94
95 [data-role="content"] h3 {
96     font-size: medium;
97     font-weight: normal;
98     color: black;
99     background-color: transparent;
100 }
101
102
103 [data-role="content"] h4 {
104     font-size: smaller;
105     font-style: italic;
106     font-weight: normal;
107     color: black;
108     background-color: transparent;
109 }
110
111 .pub_details {
112     width: 400px;
113     margin: 0 auto;
114     display: inline-block;
115     background: silver;
116     text-shadow: none;
117     padding: 10px;
118     border-radius: 5px;
119 }
120
121 #boxbar {
122     width: 250px;
123     margin: 0 auto;
124     vertical-align: top;
125     display: inline-block;
126 }
127
128 .box{
129     text-align: right;
130     text-shadow: none;
131     background: silver;
132     color: black;
133     padding-left: 5px;
```

```
134     padding-right:5px;
135     margin: 10px 5px;
136     width: 250px;
137     display: inline-block;
138 }
139
140 .titlebox {
141     background:grey;
142     color: black;
143     margin-left:-5px;
144     margin-right:-5px;
145     padding-right:5px;
146 }
147
148 .box a {
149     font-size: smaller;
150 }
151
152 .details {
153     max-width: 700px;
154     margin:0 auto;
155 }
156
157 .buttons {
158     margin: 0 auto;
159     max-width: 700px;
160     text-align: center;
161 }
162
163 #advanced-search {
164     margin: 0 auto;
165 }
166
167 [data-role="fieldcontain"] {
168     text-align: center;
169 }
170
171 #tags {
172     text-align: center;
173     margin: 0 auto;
174 }
```

```
175
176 #btn_img img{
177     vertical-align: middle;
178 }
179
180 #btn_img {
181     width: 170px;
182     height: 65px;
183     vertical-align: middle;
184 }
185
186 #pdf_img {
187     width: 50px;
188 }
189
190 #ext_img {
191     width: 25px;
192 }
193
194 /* iPad [portrait + landscape] */
195 @media only screen and (min-device-width: 768px) and (max-
    device-width: 1024px) {
196     #head, #head img {
197         height: 90px;
198     }
199 }
200
201 /* iPhone [portrait + landscape] */
202 @media only screen and (max-device-width: 480px) {
203
204     .pub_details {
205         min-width: 250px;
206         width: auto;
207         display: block;
208     }
209
210     #boxbar {
211         width: 270px;
212         display: block;
213     }
214 }
```

```
215     .box{
216         display: block;
217     }
218
219 }
220
221 /* iPhone [landscape] */
222 @media only screen and (max-device-width: 480px) and (
223     orientation: landscape){
224
225     #not-found-message {
226         width: 350px;
227         height: 200px;
228     }
229
230     #notfound-img{
231         height: 180px;
232         width: auto;
233         margin-left: -80px;
234         margin-top: 40px;
235         display: inline;
236     }
237
238     #text-msg {
239         height: 50px;
240         position: absolute;
241         display: inline-block;
242         margin-top: -150px;
243         margin-left: 120px;
244     }
245
246     #message {
247         font-size: large;
248     }
249
250     #subs {
251         font-size: small;
252     }
253 }
```

Servlets

PubsList

Ha il compito di ricercare le informazioni base di ogni pubblicazione (titolo, anno e autori) per creare l'elenco dinamico all'interno della pagina `pubsfounded.jsp`.

```
1  import java.io.*;
2  import java.sql.*;
3  import javax.naming.Context;
4  import javax.naming.InitialContext;
5  import javax.sql.DataSource;
6  import javax.servlet.*;
7  import javax.servlet.http.*;
8  import apice.ListItem;
9
10 public class PubsList extends HttpServlet {
11
12
13     public void doGet (HttpServletRequest request,
14                        HttpServletResponse response) throws ServletException,
15                        IOException{
16
17         String jspPage = "";
18         String select = "";
19         String from = "";
20         String condition = "";
21         String join = "";
22
23         String type = request.getParameter("type");
24
25         if (type.equals("all")){
26
27             select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
28                     XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
29                     XWL_VALUE as AUTHOR";
30
31             from = "xwikistrings as S, xwikilongs as L,
32                   xwikilargestrings as LS, xwikiobjects as O";
33
34         }
```

```
29      condition = "S.XWS_NAME = 'title' and L.XWL_NAME = '
        year' and LS.XWL_NAME = 'author' and O.XWO_CLASSNAME
        = 'Publications.PublicationClass' and O.XWO_NAME <>
        'PublicationClassTemplate'";
30
31      join = "S.XWS_ID = O.XWO_ID and L.XWL_ID = O.XWO_ID and
        LS.XWL_ID = O.XWO_ID";
32
33
34  } else if (type.equals("articles")){
35
36      select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
        XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
        XWL_VALUE as AUTHOR";
37
38      from = "xwikistrings as S, xwikistrings as S1,
        xwikistrings as S2, xwikilongs as L,
        xwikilargeststrings as LS, xwikiobjects as O";
39
40      condition = "O.XWO_NAME <> 'PublicationClassTemplate'
        and S.XWS_NAME = 'title' and S1.XWS_NAME = 'journal'
        and S1.XWS_VALUE <> '' and S2.XWS_NAME = 'status'
        and S2.XWS_VALUE = 'Published' and L.XWL_NAME = '
        year' and LS.XWL_NAME = 'author' and O.XWO_CLASSNAME
        = 'Publications.PublicationClass'";
41
42      join = "S.XWS_ID = O.XWO_ID and S1.XWS_ID = O.XWO_ID
        and S2.XWS_ID = O.XWO_ID and L.XWL_ID = O.XWO_ID and
        LS.XWL_ID = O.XWO_ID";
43
44
45  } else if (type.equals("series")){
46
47      select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
        XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
        XWL_VALUE as AUTHOR";
48
49      from = "xwikistrings as S, xwikistrings as S1,
        xwikistrings as S2, xwikilongs as L,
        xwikilargeststrings as LS, xwikiobjects as O";
50
```

```
51      condition = "O.XWO_NAME <> 'PublicationClassTemplate'
      and S.XWS_NAME = 'title' and S1.XWS_NAME = 'series'
      and S1.XWS_VALUE <> '' and S2.XWS_NAME = 'status'
      and S2.XWS_VALUE = 'Published' and L.XWL_NAME = '
      year' and LS.XWL_NAME = 'author' and O.XWO_CLASSNAME
      = 'Publications.PublicationClass'";
52
53      join = "S.XWS_ID = O.XWO_ID and S1.XWS_ID = O.XWO_ID
      and S2.XWS_ID = O.XWO_ID and L.XWL_ID = O.XWO_ID and
      LS.XWL_ID = O.XWO_ID";
54
55
56  } else if (type.equals("papers")){
57
58      select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
      XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
      XWL_VALUE as AUTHOR";
59
60      from = "xwikistrings as S, xwikistrings as S1,
      xwikistrings as S2, xwikilongs as L,
      xwikilargestrings as LS, xwikiobjects as O";
61
62      condition = "O.XWO_NAME <> 'PublicationClassTemplate'
      and S.XWS_NAME = 'title' and S1.XWS_NAME = 'category'
      ' and S1.XWS_VALUE = 'inproceedings' and S2.XWS_NAME
      = 'status' and S2.XWS_VALUE = 'Published' and L.
      XWL_NAME = 'year' and LS.XWL_NAME = 'author' and O.
      XWO_CLASSNAME = 'Publications.PublicationClass'";
63
64      join = "S.XWS_ID = O.XWO_ID and S1.XWS_ID = O.XWO_ID
      and S2.XWS_ID = O.XWO_ID and L.XWL_ID = O.XWO_ID and
      LS.XWL_ID = O.XWO_ID";
65
66
67  } else if (type.equals("books")){
68
69      select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
      XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
      XWL_VALUE as AUTHOR";
70
71      from = "xwikistrings as S, xwikistrings as S1,
```

```

    xwikistrings as S2, xwikistrings as S3, xwikilongs
    as L, xwikilargeststrings as LS, xwikiobjects as O";
72
73    condition = "O.XWO_NAME <> 'PublicationClassTemplate'
    and S.XWS_NAME = 'title' and S1.XWS_NAME = '
    booktitle' and S1.XWS_VALUE <> '' and S2.XWS_NAME =
    'status' and S2.XWS_VALUE = 'Published' and S3.
    XWS_NAME = 'category' and S3.XWS_VALUE <> 'book' and
    S3.XWS_VALUE like '%proceedings%' and L.XWL_NAME =
    'year' and LS.XWL_NAME = 'author' and O.
    XWO_CLASSNAME = 'Publications.PublicationClass'";
74
75    join = "S.XWS_ID = O.XWO_ID and S1.XWS_ID = O.XWO_ID
    and S2.XWS_ID = O.XWO_ID and S3.XWS_ID = O.XWO_ID
    and L.XWL_ID = O.XWO_ID and LS.XWL_ID = O.XWO_ID";
76
77
78    } else if (type.equals("edited")){
79
80        select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
            XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, S1.
            XWS_VALUE as AUTHOR";
81
82        from = "xwikistrings as S, xwikistrings as S1,
            xwikistrings as S2, xwikilongs as L,
            xwikilargeststrings as LS, xwikiobjects as O";
83
84        condition = "O.XWO_NAME <> 'PublicationClassTemplate'
            and S.XWS_NAME = 'title' and S1.XWS_NAME = 'editor'
            and S1.XWS_VALUE <> '' and S2.XWS_NAME = 'status'
            and S2.XWS_VALUE = 'Published' and L.XWL_NAME = '
            year' and LS.XWL_NAME = 'author' and LS.XWL_VALUE =
            '' and O.XWO_CLASSNAME = 'Publications.
            PublicationClass'";
85
86        join = "S.XWS_ID = O.XWO_ID and S1.XWS_ID = O.XWO_ID
            and S2.XWS_ID = O.XWO_ID and L.XWL_ID = O.XWO_ID and
            LS.XWL_ID = O.XWO_ID";
87
88    } else if (type.equals("search")){
89

```

```
90     String text = request.getParameter("text");
91     String year = request.getParameter("year");
92     String status = request.getParameter("status");
93
94     select = "distinct O.XWO_ID as ID, O.XWO_NAME as NAME,
95             S.XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
96             XWL_VALUE as AUTHOR";
97
98     from = "xwikistrings as S, xwikistrings as S1,
99           xwikilongs as L, xwikilargeststrings as LS,
100          xwikilargeststrings as LS1, xwikiobjects as O";
101
102     condition = "O.XWO_NAME <> 'PublicationClassTemplate'
103               and S.XWS_NAME = 'title' and S1.XWS_NAME = 'status'
104               and S1.XWS_VALUE like '%" + status + "' and L.
105               XWL_NAME = 'year' and L.XWL_VALUE like '%" + year +
106               "' and LS.XWL_NAME = 'author' and LS1.XWL_VALUE like
107               '%" + text + "%' and O.XWO_CLASSNAME = '
108               Publications.PublicationClass'";
109
110     join = "S.XWS_ID = O.XWO_ID and S1.XWS_ID = O.XWO_ID
111           and L.XWL_ID = O.XWO_ID and LS.XWL_ID = O.XWO_ID and
112           LS1.XWL_ID = O.XWO_ID";
113
114 } else if (type.equals("authors")){
115
116     String name = request.getParameter("name");
117
118     select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
119             XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
120             XWL_VALUE as AUTHOR";
121
122     from = "xwikistrings as S, xwikilongs as L,
123           xwikilargeststrings as LS, xwikiobjects as O";
124
125     condition = "S.XWS_NAME = 'title' and L.XWL_NAME = '
126               year' and LS.XWL_NAME = 'author' and O.XWO_CLASSNAME
127               = 'Publications.PublicationClass' and O.XWO_NAME <>
128               'PublicationClassTemplate' and LS.XWL_VALUE like
129               '%" + name + "%'";
130
131 }
```

```

112         join = "S.XWS_ID = O.XWO_ID and L.XWL_ID = O.XWO_ID and
                LS.XWL_ID = O.XWO_ID";
113
114     } else if (type.equals("editors")){
115
116         String name = request.getParameter("name");
117
118         select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
                XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
                XWL_VALUE as AUTHOR";
119
120         from = "xwikistrings as S, xwikistrings as S1,
                xwikilongs as L, xwikilargeststrings as LS,
                xwikiobjects as O";
121
122         condition = "S.XWS_NAME = 'title' and L.XWL_NAME = '
                year' and LS.XWL_NAME = 'author' and O.XWO_CLASSNAME
                = 'Publications.PublicationClass' and O.XWO_NAME <>
                'PublicationClassTemplate' and S1.XWS_NAME = '
                editor' and S1.XWS_VALUE like '%" + name + "%'";
123
124         join = "S.XWS_ID = O.XWO_ID and S1.XWS_ID = O.XWO_ID
                and L.XWL_ID = O.XWO_ID and LS.XWL_ID = O.XWO_ID";
125
126     } else if (type.equals("js")){
127
128         String name = request.getParameter("name");
129
130         select = "O.XWO_ID as ID, O.XWO_NAME as NAME, S.
                XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
                XWL_VALUE as AUTHOR";
131
132         from = "xwikistrings as S, xwikistrings as S1,
                xwikilongs as L, xwikilargeststrings as LS,
                xwikiobjects as O";
133
134         condition = "S.XWS_NAME = 'title' and L.XWL_NAME = '
                year' and LS.XWL_NAME = 'author' and O.XWO_CLASSNAME
                = 'Publications.PublicationClass' and O.XWO_NAME <>
                'PublicationClassTemplate' and (S1.XWS_NAME = '
                journal' or S1.XWS_NAME = 'series') and S1.XWS_VALUE

```

```
        like \"%\" + name + \"%\"";
135
136    join = "S.XWS_ID = O.XWO_ID and S1.XWS_ID = O.XWO_ID
        and L.XWL_ID = O.XWO_ID and LS.XWL_ID = O.XWO_ID";
137
138    } else if (type.equals("tags")){
139
140        String name = request.getParameter("name");
141
142        select = "distinct O.XWO_ID as ID, O.XWO_NAME as NAME,
        S.XWS_VALUE as TITLE, L.XWL_VALUE as YEAR, LS.
        XWL_VALUE as AUTHOR";
143
144        from = "xwikistrings as S, xwikistrings as S1,
        xwikilongs as L, xwikilargeststrings as LS,
        xwikiobjects as O, xwikiobjects as T, xwikilistitems
        as I";
145
146        condition = "S.XWS_NAME = 'title' and L.XWL_NAME = '
        year' and LS.XWL_NAME = 'author' and O.XWO_CLASSNAME
        = 'Publications.PublicationClass' and O.XWO_NAME <>
        'PublicationClassTemplate' and T.XWO_CLASSNAME = '
        XWiki.TagClass' and I.XWL_NAME = 'tags' and I.
        XWL_VALUE like \"%\" + name + \"%\"";
147
148        join = "S.XWS_ID = O.XWO_ID and S1.XWS_ID = O.XWO_ID
        and L.XWL_ID = O.XWO_ID and LS.XWL_ID = O.XWO_ID and
        O.XWO_NAME = T.XWO_NAME and I.XWL_ID = T.XWO_ID";
149
150    }
151    try{
152        if (!select.equals("")){
153            InitialContext initCtx = new InitialContext();
154            DataSource dataSource = (DataSource) initCtx.lookup("
                java:comp/env/jdbc/ApiceHTML5");
155            Connection dbconn;
156            ResultSet results;
157            PreparedStatement sql;
158            Class.forName("org.gjt.mm.mysql.Driver");
159            try {
160                dbconn = dataSource.getConnection();
```

```
161         sql = dbconn.prepareStatement("select " + select +  
162             " from " + from + " where " + condition + " and  
163             " + join + " order by YEAR desc;");  
164     results = sql.executeQuery();  
165     if(results.next()){  
166         results.last();  
167         int size = results.getRow();  
168         int i = 0;  
169         ListItem[] res = new ListItem[size];  
170         results.beforeFirst();  
171  
172         String id = "";  
173         String name = "";  
174         String title = "";  
175         String year = "";  
176         String author = "";  
177  
178         while(results.next()){  
179             id = results.getString("ID");  
180  
181             name = results.getString("NAME");  
182             name = name.replace("Publications.", "");  
183  
184             title = results.getString("TITLE");  
185  
186             if (results.getString("YEAR") == null){  
187                 year = "";  
188             } else {  
189                 year = results.getString("YEAR");  
190             }  
191  
192             author = results.getString("AUTHOR");  
193             author = author.replace(",", "");  
194             author = author.replace(" and ", ", ");  
195             author = author.replace(" AND ", ", ");  
196             if (type.equals("edited")) author = author + "  
197                 (eds)";  
198  
199             //impostazione parametri da passare  
200             res[i] = new ListItem();  
201             res[i].setId(id);
```

```

199         res[i].setName(name);
200         res[i].setTitle(title);
201         res[i].setYear(year);
202         res[i].setAuthor(author);
203
204         i++;
205     }
206     dbconn.close();
207     if (type.equals("authors") || type.equals("
        editors") || type.equals("js") || type.equals(
            "tags")){
208         request.setAttribute("name", name);
209     }
210         request.setAttribute("results", res);
211         jspPage = "/pubsfounded.jsp?type=" + type;
212     }else{
213         jspPage = "/nofound.jsp";
214     }
215     }catch (SQLException s){
216         request.setAttribute("error", s);
217         jspPage = "/error.jsp";
218     }
219
220     }
221 }catch (Exception err){
222     request.setAttribute("error", err);
223     jspPage = "/error.jsp";
224 }
225 request.setAttribute("from", "PubsList");
226 RequestDispatcher requestDispatcher = getServletContext()
    .getRequestDispatcher(jspPage);
227 requestDispatcher.forward(request, response);
228 }
229 }
```

Indexes

Interroga il database per indicizzare le pubblicazioni, tenendo anche il conto di quante pubblicazioni ci siano per ogni elemento di un indice. Viene caricata

su indicazione del file manifest dell'app, nel caso in cui ci sia qualche risorsa non accessibile.

```
1 import java.io.*;
2 import java.sql.*;
3 import java.util.ArrayList;
4 import java.util.LinkedHashMap;
5 import java.util.Collections;
6 import javax.naming.Context;
7 import javax.naming.InitialContext;
8 import javax.sql.DataSource;
9 import javax.servlet.*;
10 import javax.servlet.http.*;
11 import apice.ListItem;
12
13
14 public class Indexes extends HttpServlet {
15
16     public void doGet (HttpServletRequest request,
17         HttpServletResponse response) throws ServletException,
18         IOException{
19
20         String select = "";
21         String from = "";
22         String condition = "";
23         String join = "";
24
25         String type = request.getParameter("type");
26
27         if (type.equals("authors")){
28
29             select = "LS.XWL_VALUE as NAMES";
30
31             from = "xwikilargestrings as LS, xwikiobjects as O";
32
33             condition = "LS.XWL_NAME = 'author' and LS.XWL_VALUE <>
34                 '' and O.XWO_CLASSNAME = 'Publications.
35                 PublicationClass'";
36
37             join = "LS.XWL_ID = O.XWO_ID";
38
39         } else if (type.equals("editors")){
```

```
36
37     select = "S.XWS_VALUE as NAMES";
38
39     from = "xwikistrings as S, xwikiobjects as O";
40
41     condition = "S.XWS_NAME = 'editor' and S.XWS_VALUE <>
42         '' and O.XWO_CLASSNAME = 'Publications.
43         PublicationClass'";
44
45     join = "S.XWS_ID = O.XWO_ID";
46
47 } else if (type.equals("js")){
48
49     select = " S.XWS_VALUE as NAME, count(*) as NUM";
50
51     from = "xwikistrings as S, xwikiobjects as O";
52
53     condition = "(S.XWS_NAME = 'journal' or S.XWS_NAME = '
54         series') and S.XWS_VALUE <> '' and O.XWO_CLASSNAME =
55         'Publications.PublicationClass' group by S.
56         XWS_VALUE order by S.XWS_VALUE";
57
58     join = "S.XWS_ID = O.XWO_ID";
59
60 } else if (type.equals("tags")){
61
62     select = "L.XWL_VALUE as NAME, count(*) as NUM";
63
64     from = "xwikiobjects as P, xwikiobjects as T,
65         xwikilistitems as L";
66
67     condition = "P.XWO_CLASSNAME = 'Publications.
68         PublicationClass' and T.XWO_CLASSNAME = 'XWiki.
69         TagClass' and L.XWL_NAME = 'tags' group by L.
70         XWL_VALUE";
71
72     join = "P.XWO_NAME = T.XWO_NAME and L.XWL_ID = T.XWO_ID
73         ";
74
75 }
76
77 if (!select.equals("")){
```



```

97         }else{
98             map.put(n[i], ++count); //
                altrimenti incrementiamo il
                conteggio
99         }
100         if (!List.contains(n[i])){
101             List.add(n[i]);
102         }
103     }
104 } else if (type.equals("js") || type.equals("
    tags")){
105     String name = results.getString("NAME");
106     String count = results.getString("NUM");
107     List.add(name + "~" + count);
108 }
109 }
110 if(type.equals("authors") || type.equals("
    editors")){
111     Collections.sort(List);
112     String[] names = (String[]) List.toArray(new
        String[List.size()]);
113     ListItem[] res = new ListItem[List.size()];
114     for(i = 0; i < res.length; i++){
115         res[i] = new ListItem();
116         res[i].setName(names[i]);
117         res[i].setCount(map.get(names[i])); //con map
            .get(res[i]) ci restituisce il valore del
            contatore che corrisponde alla stringa in
            res[i]
118     }
119     request.setAttribute("results", res);
120 } else if (type.equals("js") || type.equals("tags
    ")){
121     String[] ls = (String[]) List.toArray(new
        String[List.size()]);
122     ListItem[] res = new ListItem[List.size()];
123     for(i = 0; i < res.length; i++){
124         //divido la stringa in due, con ;
125         // il primo elemento e' il nome, il secondo
            il conteggio
126         String[] el = ls[i].split("~");

```

```
127         res[i] = new ListItem();
128         res[i].setName(el[0]);
129         res[i].setCount(Integer.parseInt(el[1]));
130     }
131     request.setAttribute("results", res);
132 }
133 jspPage = "/pubsfounded.jsp?type=" + type;
134 }else{
135     jspPage = "/nofound.jsp";
136 }
137 dbconn.close();
138 }catch (SQLException s){
139     request.setAttribute("error", s);
140     jspPage = "/error.jsp";
141 }
142 }catch (Exception err){
143     request.setAttribute("error", err);
144     jspPage = "/error.jsp";
145 }
146 request.setAttribute("from", "Indexes");
147 RequestDispatcher requestDispatcher = getServletContext
    ().getRequestDispatcher(jspPage);
148 requestDispatcher.forward(request, response);
149 }
150
151 }
152
153 }
```

PubInfo

Ha il compito ultimo di ricercare tutte le informazioni disponibili per una particolare pubblicazione che verranno poi passate alla pagina Viene caricata su indicazione del file manifest dell'app, nel caso in cui ci sia qualche risorsa non accessibile.

```
1 import java.io.*;
2 import java.sql.*;
3 import java.util.ArrayList;
4 import java.util.LinkedHashMap;
```

```
5 import java.util.Collections;
6 import javax.naming.Context;
7 import javax.naming.InitialContext;
8 import javax.sql.DataSource;
9 import javax.servlet.*;
10 import javax.servlet.http.*;
11 import apice.Publication;
12
13
14 public class PubInfo extends HttpServlet {
15
16
17     public void doGet (HttpServletRequest request,
18         HttpServletResponse response) throws ServletException,
19         IOException{
20
21         String select = "";
22         String from = "";
23         String condition = "";
24         String join = "";
25
26         String jspPage = "/error.jsp";
27
28
29         String id = request.getParameter("id");
30
31         Publication pub = new Publication(id, request.
32             getParameter("name"));
33
34         select = "LS.XWL_VALUE as AUTHOR, LS1.XWL_VALUE as
35             ABSTRACT, L.XWL_VALUE as YEAR";
36
37         from = "xwikilongs as L, xwikilargeststrings as LS,
38             xwikilargeststrings as LS1";
39
40         condition = "L.XWL_ID = '" + id + "' and L.XWL_NAME = '
41             year' and LS.XWL_NAME = 'author' and LS1.XWL_NAME =
42             'abstract'";
43
44         join = "LS.XWL_ID = L.XWL_ID and LS1.XWL_ID = L.XWL_ID"
45             ;
46     }
47 }
```

```

38 try{
39     InitialContext initCtx = new InitialContext();
40     DataSource dataSource = (DataSource) initCtx.lookup("
41         java:comp/env/jdbc/ApiceHTML5");
42     Connection dbconn;
43     ResultSet results;
44     PreparedStatement sql;
45     Class.forName("org.gjt.mm.mysql.Driver");
46     //Class.forName("com.mysql.jdbc.Driver");
47     try {
48         dbconn = dataSource.getConnection();
49         sql = dbconn.prepareStatement("select " + select + "
50             from " + from + " where " + join + " and " +
51             condition + ";");
52         results = sql.executeQuery();
53         if(results.next()){
54             results.beforeFirst();
55             while(results.next()){
56                 pub.setAuthor(results.getString("AUTHOR"));
57                 pub.setAbstract(results.getString("ABSTRACT"));
58                 pub.setYear(results.getString("YEAR"));
59             }
60
61             sql = dbconn.prepareStatement("select XWS_NAME as
62                 NAME, XWS_VALUE as VALUE from xwikistrings where
63                 XWS_ID = ' " + id + " ' and XWS_VALUE <> ''");
64             results = sql.executeQuery();
65             while(results.next()){
66                 pub.addDetails(results.getString("NAME"),
67                     results.getString("VALUE"));
68             }
69
70             select = "I.XWL_VALUE as VALUE";
71
72             from = "xwikiobjects as T, xwikilistitems as I";
73
74             condition = "T.XWO_NAME = 'Publications.' + pub.
75                 getName() + ' ' and T.XWO_CLASSNAME = 'XWiki.
76                 TagClass' and I.XWL_NAME = 'tags'";

```

```
71         join = "I.XWL_ID = T.XWO_ID";
72
73         sql = dbconn.prepareStatement("select " + select +
74             " from " + from + " where " + join + " and " +
75             condition + ";");
76         results = sql.executeQuery();
77         while(results.next()){
78             pub.addTag(results.getString("VALUE"));
79         }
80
81         request.setAttribute("publication", pub);
82         jspPage = "/publication.jsp";
83     }else{
84         jspPage = "/nofound.jsp";
85     }
86     dbconn.close();
87 }catch (SQLException s){
88     request.setAttribute("error", s);
89     jspPage = "/error.jsp";
90 }
91 }catch (Exception err){
92     request.setAttribute("error", err);
93     jspPage = "/error.jsp";
94 }
95
96 RequestDispatcher requestDispatcher = getServletContext()
97     .getRequestDispatcher(jspPage);
98 requestDispatcher.forward(request, response);
99
100 }
101 }
```


Bibliografia

- [1] HTML 5 Specification,
<http://www.w3.org/TR/html5/>;
- [2] Introduction to CSS3,
<http://www.w3.org/TR/2001/WD-css3-roadmap-20010523/>;
- [3] Javascript APIs,
http://www.w3.org/TR/#tr_Javascript_APIs;
- [4] “*WHY HTML5 ROCKS*”,
<http://www.html5rocks.com/en/why>;
- [5] XWiki Documentation,
<http://www.xwiki.org/xwiki/bin/view/Main/Documentation>;
- [6] XWiki Architecture,
<http://platform.xwiki.org/xwiki/bin/view/DevGuide/Architecture>;
- [7] Mary Meeker (2012), “*Internet Trends*”, Stanford BASES;
<http://kpcb.com/insights/2012-internet-trends-update>
- [8] Jakob Nielsen (2012), “*Mobile Site vs. Full Site*”,
<http://www.useit.com/alertbox/mobile-vs-full-sites.html>;
- [9] jQuery Mobile: Demo and Documentation,
<http://jquerymobile.com/demos/1.2.0/>;
- [10] Apache Tomcat Documentation,
<http://tomcat.apache.org/tomcat-6.0-doc/index.html>;