UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Vecchio Ordinamento in Ingegneria Informatica

Reingegnerizzazione dell'applicazione "Guida ai Servizi" del Comune di Treviso

Laureando:
Gianandrea Moro

Relatore:
Prof. Maristella Agosti

Anno Accademico 2011/2012

Abstract

Il Comune di Treviso era dotato di un'applicazione di tipo CMS (Content Management System) che serviva per descrivere su web i servizi che l'Amministrazione offre alla cittadinanza. L'obsolescenza del software ha portato all'esigenza di reingegnerizzare tale applicazione.
Vengono descritte le attività svolte: raccolta e analisi dei requisiti, progettazione, implementazione e verifica dei risultati ottenuti.
Particolare rilievo è stato dato alla progettazione del database e alle soluzioni adottate relative all'implementazione del motore di ricerca e alla struttura del codice.
Nella stesura del codice, in cui sono state utilizzate diverse tecniche tra cui AJAX (Asynchronous JavaScript and XML), si è posta particolare attenzione alla riusabilità.
Nel realizzare l'applicazione si è dovuto tener conto anche degli aspetti normativi relativi all'accessibilità.
Il sistema è stato realizzato su una piattaforma open source che utilizza strumenti quali Linux, Apache, MySQL e PHP.

Indice

1 Introduzione

La tesi descrive la reingegnerizzazione di un'applicazione di tipo CMS (Content Management System), chiamata “Guida ai Servizi”, sviluppata dal Comune di Treviso allo scopo di descrivere e pubblicare sul sito web istituzionale i servizi che l'Amministrazione mette a disposizione.

Il progetto è nato dall'esigenza di sostituire il precedente sistema ormai inadeguato alle nuove normative sull'accessibilità e basato su software proprietario di difficile personalizzazione.

La reingegnerizzazione è avvenuta seguendo le attività del ciclo di vita di un sistema informatico: studio di fattibilità, raccolta e analisi dei requisiti, progettazione, implementazione, validazione, collaudo e funzionamento [1].

Il sistema è stato realizzato su una piattaforma LAMP (Linux, Apache, MySQL, PHP) [10] ed è stato sviluppato principalmente con risorse interne all'ente e con l'adozione di software open source.

Il lavoro è stato svolto in qualità di dipendente del Comune di Treviso all'interno della struttura dei Servizi Informatici dell'ente, sotto la supervisione dell'Ing. Marcello Missagia e dell'Ing. Roberto Meneghetti, e in collaborazione con la Dott.ssa Patrizia Cadel, responsabile dell'Ufficio Relazioni con il Pubblico (in seguito indicato come URP).

Le attività svolte sono state la raccolta e l'analisi dei requisiti, la progettazione, l'implementazione, il collaudo e la messa in funzione. Nell'implementazione il sottoscritto ha coordinato il lavoro di Raffi Tchakerian e Silvia Boscolo, studenti dell'Università di Architettura di Venezia Corso di disegno industriale, che hanno curato la grafica dell'applicazione, e di Maria Berlese e Michele Terragno, studenti dell'Istituto Tecnico Einaudi di Montebelluna, che hanno curato il motore di ricerca.

La tesi descrive le fasi che hanno portato alla realizzazione del progetto. Nel secondo capitolo verranno elencate le funzionalità desiderate per il sistema e i vincoli legali e tecnologici cui si è dovuto sottostare. Queste informazioni sono state il frutto di un'attività di consultazione degli attori del sistema, dell'analisi del software precedente e della lettura della normativa. Per quel che riguarda la normativa, una delle maggiori problematiche è stata l'accessibilità intesa come la capacità di un sistema informatico di erogare servizi e fornire informazioni fruibili, senza discriminazioni, anche da parte di coloro che a causa di disabilità necessitano di tecnologie assistive o di configurazioni particolari. Relativamente ai vincoli tecnologici verrà esposta l'infrastruttura informatica utilizzata composta

interamente da software open source, come da indicazione dell'ente. Nel terzo, quarto e quinto capitolo verrà descritta la procedura che ha portato all'implementazione del database cui si appoggia il sistema. Nel sesto capitolo verrà spiegata la realizzazione del motore di ricerca che è una funzionalità molto importante in un'applicazione come questa che si propone di fornire informazioni al pubblico. In particolare verrà illustrato l'algoritmo che è stato ideato e la sua implementazione. Il settimo capitolo sarà dedicato alla struttura del codice e alle motivazioni che hanno portato all'adozione di un particolare paradigma di programmazione. Nello stesso capitolo verranno esposte le soluzioni impiegate per la gestione della programmazione concorrente. Verranno poi riportate le principali problematiche che si sono dovute affrontare nella realizzazione delle interfacce di back-end e di front-end dell'applicazione. Verranno descritte la scelta dell'editor wysiwyg (What You See Is What You Get) utilizzato nel back-end e la gestione del template grafico di front-end. Seguiranno alcune statistiche di utilizzo del sistema realizzato e il confronto con quanto era stato previsto. La tesi si concluderà, nell'ottavo capitolo, con la verifica del raggiungimento degli obiettivi, con l'analisi delle criticità esistenti e con le future possibilità di sviluppo.

1.1 Obiettivi dell'applicazione

L'obiettivo principale dell'applicazione "Guida ai Servizi" del Comune di Treviso è quello di veicolare via web ai cittadini le informazioni relative ai servizi forniti dall'ente. In particolare è necessario spiegare, in modo semplice e preciso, i suddetti servizi e fare in modo che queste informazioni siano facilmente reperibili da parte della cittadinanza.

Ogni servizio è gestito o erogato da un apposito ufficio dell'ente. Ogni ufficio è quindi il depositario di tutte le informazioni necessarie ai servizi di propria competenza.

Scopo dell'applicazione che si intende realizzare è anche quello di deviare su web le richieste di informazioni che pervengono attraverso altri canali, quali il telefono, il fax o la mail. In questo modo il cittadino è maggiormente autonomo nel reperimento delle informazioni, che tra l'altro può avvenire 24 ore su 24, e l'ufficio viene sgravato dal carico di lavoro dovuto all'attività informativa.

L'applicazione deve quindi gestire il flusso documentale, necessario alla composizione delle informazioni da pubblicare, in modo semplice e veloce per non pesare sulle altre attività degli uffici e averne quindi un ritorno in termini di efficienza.

Considerando le eterogenee abilità nell'uso degli strumenti informatici del personale degli uffici coinvolti, la semplicità dell'interfaccia è un aspetto rilevante.

1.2 Breve descrizione dell'infrastruttura Comunale coinvolta

Il Comune di Treviso è un ente organizzato in Aree, Settori e Servizi, che assolvono a varie mansioni, quali opere pubbliche, gestione urbanistica, controllo delle attività commerciali, tutela del territorio, promozione di attività culturali, etc.

Il progetto ha coinvolto praticamente tutti gli uffici Comunali in quanto ognuno fornisce al pubblico un servizio che è stato descritto e pubblicato nella "Guida ai Servizi". I principali uffici che hanno collaborato alla nascita e allo sviluppo del progetto sono i Servizi Informatici e l'URP.

Il personale dei Servizi Informatici ha la funzione di gestire l'infrastruttura informatica (hardware e software) dell'ente tra cui il sito internet istituzionale (www.comune.treviso.it). Tra i tecnici vi sono professionalità con mansioni sia sistemistiche che di programmazione, rendendo così possibile lo sviluppo in-house di diverse soluzioni, come quella oggetto del presente elaborato.

Da diverso tempo l'Amministrazione ha scelto di adottare, per quanto possibile, soluzioni open source e questo si riflette su molti degli strumenti su cui è basato il progetto.

L'URP ha la funzione di gestire i rapporti tra l'ente e la cittadinanza utilizzando diversi canali di comunicazione, che vanno dal contatto diretto (sportello) alla comunicazione telematica (web). Una caratteristica importante dell'URP, molto utile al fine del progetto, è quella di saper tradurre e spiegare con semplicità il linguaggio adottato dal personale dell'ente, che è talvolta troppo tecnico.

Altri uffici, quali l'ufficio Risorse Umane, hanno collaborato individuando le risorse di stage che, coordinate dai Servizi Informatici, hanno aggiunto importanti tasselli al sistema.

2 Analisi dei requisiti

In questa fase si è proceduto alle interviste con il personale dei principali uffici coinvolti, in particolare con l'URP, alla lettura della normativa e all'analisi della precedente implementazione dell'applicazione.

2.1 Storia dell'applicazione

Inizialmente la soluzione per pubblicare le schede informative era implementata su piattaforma Lotus Domino di IBM [11] e sviluppata dalla ditta "inRebus". L'ambiente Lotus metteva a disposizione dell'applicazione una buona integrazione con l'anagrafica dei dipendenti Comunali cui era affidato il compito di proporre le informazioni e consentiva inoltre una buona base per l'implementazione del flusso documentale necessario. Tuttavia nella versione installata non era possibile controllare con esattezza l'output, cioè l'html generato. Con l'avvento della nuova normativa, che verrà esposta nel capitolo 2.2, questo si è palesato come un grosso limite che ha impedito di proseguire con lo sviluppo nella stessa piattaforma.

Lotus Domino è caratterizzato da un'architettura del database che è document-oriented per cui non è stato possibile ricavarne direttamente uno schema logico ma è stato possibile esaminarne i dati. Il database infatti consisteva in una collezione di documenti che ospitavano talvolta campi diversi.

L'esperienza acquisita con quell'implementazione della "Guida ai Servizi" è stata comunque preziosa per l'individuazione degli use cases e di altri requisiti del sistema. In sintesi, si tratta di reingegnerizzare un sistema legacy data decomponibile, quindi trattabile [2]. Si tratta infatti principalmente di un sistema semistrutturato in cui i componenti applicativi sono separabili in due livelli: un livello costituito dai servizi d'acceso ai dati e uno composto dai servizi di presentazione e logica applicativa fusi in un unico blocco. In questi sistemi è possibile accedere direttamente ai dati ma non alla logica applicativa.

Infine, la scomparsa della ditta realizzatrice è stato il motivo per cui si è proceduto alla realizzazione del nuovo sistema con l'utilizzo di risorse interne all'ente.

2.2 Vincoli legali e tecnologici

Dall'adozione da parte delle Pubbliche Amministrazioni (in seguito indicate come PA) del web come canale di comunicazione, il legislatore ha progressivamente

completato il quadro normativo anche al fine di garantire equità di accesso alle informazioni.

A partire dalla circolare numero 3 del 13 marzo 2001 della Funzione Pubblica [12] sono stati introdotti i concetti di "usabilità" e di "accessibilità". Dove per "usabilità" si intende che *le informazioni devono essere organizzate e strutturate in maniera da garantire la massima fruibilità*, e per "accessibilità" si intende che deve essere garantita la consultazione dei siti web *anche da parte di individui affetti da disabilità fisiche o sensoriali, o condizionati dall'uso di strumenti con prestazioni limitate o da condizioni ambientali sfavorevoli*.

Nel tempo la normativa sull'accessibilità si è evoluta, passando per importanti provvedimenti quali la Legge numero 4 del 9 gennaio 2004 [3], più conosciuta con il nome di "Legge Stanca".

Allo stato attuale il testo di riferimento che riassume tutti i requisiti di legge sono le "Linee guida per i siti web della PA - Edizione 2011" [13]. L'esigenza di evidenziare al pubblico la data di aggiornamento dei contenuti delle schede informative è indicata dalle linee guida. Nello stesso testo si trova l'elenco dei contenuti minimi che devono essere presenti nei siti di una PA. Sono inoltre suggerite tecniche di realizzazione che adottano template grafici di tipo tableless in cui la disposizione grafica dei contenuti avviene con l'ausilio di fogli di stile (CSS).

Di particolare importanza per il progetto "Guida ai Servizi" è la normativa sulla pubblicazione della modulistica, infatti ai sensi dell'art 57 del CAD (Decreto legislativo 7 marzo 2005, n. 82 "Codice dell'Amministrazione digitale" [4]), *le PA non possono richiedere ai cittadini l'uso di moduli e formulari che non siano stati pubblicati. In caso di mancata pubblicazione, i procedimenti possono essere avviati senza la presentazione del modulo/formulario*.

L'articolo 57 infatti recita:

"*1. Le pubbliche amministrazioni provvedono a definire e a rendere disponibili anche per via telematica l'elenco della documentazione richiesta per i singoli procedimenti, i moduli e i formulari validi ad ogni effetto di legge, anche ai fini delle dichiarazioni sostitutive di certificazione e delle dichiarazioni sostitutive di notorietà.*

2. Trascorsi ventiquattro mesi dalla data di entrata in vigore del presente codice, i moduli o i formulari che non siano stati pubblicati sul sito non possono essere richiesti ed i relativi procedimenti possono essere conclusi anche in assenza dei suddetti moduli o formulari."

Nelle linee guida inoltre è scritto che "*Il servizio di consultazione dell'elenco dei procedimenti e la modulistica disponibile on line dovranno essere raggiungibili dalla home page del sito istituzionale, in posizione ben evidente, e correlata alle sezioni informative sui procedimenti*".

Per quanto riguarda i vincoli tecnologici, da tempo il Comune di Treviso ha adottato la politica di utilizzare quanto più possibile software libero. Gran parte dei server ha linux come sistema operativo e molti applicativi installati sono realizzati con software open source.
La "Guida ai Servizi" evidentemente deve poter contare su un'infrastruttura web.
L'ente ha scelto di sviluppare l'applicazione su una piattaforma di tipo LAMP, acronimo composto dalle iniziali di Linux (sistema operativo), Apache (web server), MySQL (database software) e PHP (linguaggio di script), che sono le principali componenti per la realizzazione di un'applicazione web.
La scelta delle componenti di questa infrastruttura, dettagliate qui di seguito, è stata guidata principalmente dalla necessità di uniformare la "Guida ai Servizi" con le altre applicazioni dell'ente. Tale necessità nasce dall'esigenza di semplificare le future operazioni di manutenzione.

2.2.1 Web server (Apache)

Il server web [14] è un servizio o tipo di server che si occupa di fornire su richiesta dell'utente (client) files di qualsiasi tipo, tra cui pagine web successivamente visualizzabili dal browser presente sul dispositivo dell'utente. Le informazioni inviate dal server web all'utente viaggiano in rete trasportate dal protocollo HTTP.
In particolare Apache (il nome completo è "Apache HTTP Server Project") [15] è un server web open source sviluppato e distribuito, con una licenza che è compatibile con la terza versione della GNU GPL [16], dalla Apache Software Foundation.
L'applicazione "Guida ai Servizi" è stata implementata utilizzando la versione 2.0.52 di Apache.

2.2.2 Database server (MySQL)

MySQL [17] è un database relazionale open source che, installato in qualità di server, consente la gestione di accessi multiutente ai database che ospita. Tramite interfaccia a carattere supporta la maggior parte della sintassi SQL. Molti

linguaggi di programmazione possiedono delle librerie di funzioni per l'utilizzo di MySQL.
Attualmente MySQL è distribuito sia con licenza GPL che con una licenza commerciale. Per la "Guida ai Servizi" è stata adottata la licenza GPL.
L'applicazione "Guida ai Servizi" è stata implementata utilizzando la versione 4.1.20 di MySQL.

2.2.3 Storage engine (MyISAM)

MyISAM è lo storage engine predefinito in MySQL. Esso utilizza la struttura ISAM (Indexed Sequential Access Method). È un motore di immagazzinamento dei dati estremamente veloce che richiede poche risorse, sia in termini di memoria RAM, sia in termini di spazio su disco. Il suo limite principale consiste nel mancato supporto delle transazioni [18].
Per poter utilizzare le transazioni si sarebbe dovuto adottare uno storage engine diverso quale l'innoDB. Ciò nonostante, vista l'impossibilità di effettuare ricerche full-text su campi di tipo blob e la scarsità di esperienza su quella tipologia di tabelle (sarebbe stata la prima applicazione sviluppata internamente all'ente con l'ausilio di innoDB), si è preferito adottare lo storage engine MyISAM.

2.2.4 Linguaggi di script (PHP, javascript)

PHP [19] (acronimo di "PHP: Hypertext Preprocessor") è un linguaggio di scripting interpretato lato server, con licenza open source e libera (ma incompatibile con la GPL), particolarmente adatto alla programmazione Web ovvero alla realizzazione di pagine web dinamiche.
L'elaborazione di codice PHP produce codice html da inviare al browser dell'utente che ne fa richiesta. I vantaggi dell'uso di PHP sono la velocità di esecuzione e la semplicità di manutenzione perché è un linguaggio scarsamente tipizzato e il codice sorgente non ha bisogno di alcun compilatore. Sono inoltre disponibili le librerie per l'interazione con il database server MySQL.
L'applicazione "Guida ai Servizi" è stata implementa utilizzando la versione 4.3.9 di PHP.
Per il lato client dell'applicazione si è fatto uso di JavaScript, che è un linguaggio di scripting orientato agli oggetti che viene interpretato ed eseguito dai browser. In particolare è stato utilizzato JavaScript per l'inserimento dell'editor wysiwyg, per il sistema di importazione dei files e per la validazione dei dati inseriti dagli utenti.

2.3 Descrizione delle necessità

Come già esposto, la necessità è quella di veicolare ai cittadini le informazioni relative ai servizi forniti dall'ente. In particolare è necessario spiegare in modo semplice e preciso i suddetti servizi e fare in modo che queste informazioni siano facilmente reperibili da parte della cittadinanza. I conoscitori dei servizi sono ovviamente gli uffici stessi che li erogano, che d'ora in poi chiameremo anche "referenti". Gli addetti alla comunicazione con i cittadini sono il personale dell'URP, che d'ora in poi chiameremo anche "redattori". Da qui l'esigenza di costruire un sistema che metta in relazione i vari uffici del Comune con l'URP e che produca delle schede informative chiare e fruibili.
È importante che l'URP sia costantemente informato su tutti i procedimenti e i servizi in modo che sia in grado di informare a sua volta l'utenza anche attraverso altri canali, quali il telefono o lo sportello.
Nelle schede informative si individuano due particolari tipi di dettagli che sono i moduli e gli allegati. Si tratta di oggetti che completano le informazioni della scheda. Gli allegati ad esempio possono ospitare dei preziari, della tabelle, o altro. Queste informazioni non cambiano sostanzialmente il contenuto del procedimento proposto al cittadino ma spesso costituiscono delle parti schematiche e suscettibili di molti cambiamenti nel tempo.

È molto importante che l'applicazione venga realizzata in modo da consentire una semplice manutenzione da parte di un gruppo eterogeneo di programmatori e sviluppatori. Il codice generato deve essere quanto più leggibile e auto-esplicativo possibile, così come la struttura del database.

Sono da distinguere due versanti dell'applicazione e le relative necessità: il back-end in cui operano gli uffici Comunali e il front-end che è quanto viene messo a disposizione del pubblico sul sito web.

2.3.1 Il back-end

Il sistema deve realizzare principalmente un flusso documentale in cui i referenti propongono una scheda descrittiva al redattore, il quale la corregge e la pubblica definitivamente.
Il flusso deve essere tale per cui l'effetto nel front-end si deve avere solo in seguito all'approvazione di un redattore. Ciò significa che mentre una scheda è in fase di modifica, nel front-end deve restare visibile la sua versione precedente.

Questo flusso documentale per cui una scheda resta pubblicata mentre è in preparazione un suo aggiornamento, serve per evitare un vuoto informativo. È infatti stato valutato molto più grave l'assenza dell'indicazione di un servizio offerto che il suo mancato aggiornamento (nel senso di dettagli minori che non portano disagio all'utenza).

É stato richiesto che i redattori possano supportare il lavoro di compilazione delle schede da parte dei referenti in ogni fase del flusso. Questo si è rivelato utile soprattutto per i primi mesi di messa in opera dell'applicazione poiché in questo modo si è potuto operare una formazione sull'utilizzo della stessa.

Nel caso un servizio venga cessato o fortemente modificato, è necessario un meccanismo di spubblicazione delle informazioni.

Non è necessario che i moduli e gli allegati siano soggetti all'approvazione dell'URP, perché essi sono necessariamente costituiti da un linguaggio tecnico non modificabile. Eventualmente nella scheda informativa può esservi una spiegazione sulla loro lettura e compilazione. Per questo la modifica di moduli e allegati, da parte dei referenti, può avere subito effetti nel front-end. L'URP ne deve essere solamente avvisato.

È stato richiesto anche un sistema di reportistica in grado di riassumere l'uso e le modalità di reperimento delle informazioni da parte del pubblico, questo per "tarare" meglio la disposizione nel front-end e il contenuto delle schede.

I servizi descritti dalle schede sono erogati o gestiti dai vari uffici Comunali cui i referenti fanno capo. Da questo derivano due conseguenze importanti, una relativa al front-end che sarà affrontata nel capitolo 2.3.2, e l'altra relativa al dominio di gestione delle schede da parte dei referenti. Tale dominio viene individuato dagli uffici, nel senso che i referenti sono chiamati a gestire le schede informative dell'ufficio cui appartengono.

Tuttavia non è sempre possibile collocare un referente in un unico ufficio. Accade infatti che un referente debba gestire servizi di più uffici o che a una persona sia delegato il compito di gestire le schede di uffici cui non appartiene. Tutto questo è necessario anche per sopperire ai casi di assenze o alle esigenze di turnazione del personale.

In sintesi è opportuno che i referenti possano gestire solo le schede degli uffici cui appartengono, che un ufficio possa contare su più referenti per la redazione delle schede e che un referente possa essere collocato in più uffici.

E' utile inoltre che il flusso documentale di back-end sia assistito da un sistema di messaggistica e-mail che informi gli utenti dei vari passaggi di stato di schede, moduli e allegati.

2.3.2 Il front-end.

Il front-end deve proporre al cittadino i contenuti delle schede informative con un interfaccia quanto più semplice e fruibile. L'aspetto grafico deve essere simile a quello del sito principale dell'ente (www.comune.treviso.it), per non dare al pubblico una sensazione di confusione e di "uscita" dal percorso di navigazione.

Il fatto che un servizio appartenga a un ufficio significa che se un cittadino avesse bisogno di ulteriori informazioni o chiarimenti su quel servizio, dovrebbe essere messo in condizione di individuare con precisione l'ufficio di riferimento. Una scheda non può essere orfana dei contatti dell'ufficio cui il cittadino deve fare riferimento.

La ricerca delle informazioni deve avvenire sia tramite la visita di un albero di categorie che tramite una ricerca libera. La gestione delle categorie deve essere riservata ai redattori che in autonomia individuano l'organizzazione migliore delle informazioni. Nella categorizzazione ha un rilievo particolare la sezione riservata alla modulistica che, come spiegato nel capitolo 2.2 è anche un'esigenza legale.

Ogni categoria deve essere decorata con un'icona. Alcune categorie devono condividere lo stesso aspetto grafico per far capire all'utente che si tratta di argomenti affini, mentre altre devono avere ognuna un un'icona distinta. Come le categorie, anche le relative icone devono poter essere gestite in autonomia dai redattori.

In una pagina web che risponda alle direttive sull'accessibilità ogni oggetto non di testo deve essere dotato di un titolo e di un testo alternativo. Il testo alternativo ad esempio viene letto dai browser per ipovedenti quale descrizione dell'immagine. Le icone non possono fare eccezione e devono quindi essere corredate da un testo descrittivo.

3 Progettazione concettuale della base di dati

Lo scopo della progettazione concettuale è quello di rappresentare le specifiche informali della realtà di interesse in termini di una descrizione formale e completa ma indipendente dai criteri di rappresentazione utilizzati nei sistemi di gestione di basi di dati.

Il modello concettuale utilizzato è il modello Entità-Relazione in cui è possibile rappresentare i concetti ricavati dall'analisi dei requisiti. I costrutti principali di questo modello sono:

Entità: rappresentano le classi di oggetti che hanno proprietà comuni ed esistenza autonoma ai fini dell'applicazione di interesse.

Associazioni (Relazioni): rappresentano i legami logici, significativi per l'applicazione di interesse, tra due o più entità.

Attributi: descrivono le proprietà elementari delle entità o delle relazioni che sono di interesse ai fini dell'applicazione.

Per una trattazione più approfondita del modello Entità-Relazione si rimanda al capitolo 5.2 in [1].

Nello schema concettuale è stato da subito evidente che gli utenti, gli uffici, i moduli e gli allegati sono delle entità perché la loro esistenza è "autonoma" rispetto all'applicazione in esame. Le schede informative invece potevano essere considerate come delle relazioni fra le precedenti entità. Tuttavia l'indipendenza che può esserci tra le schede, i moduli e gli allegati, e soprattutto la centralità del concetto di scheda informativa nel sistema stesso, hanno portato a formulare uno schema concettuale in cui la scheda è anch'essa un entità.

Lo schema entità-relazione risultante dall'analisi concettuale è quello riportato in figura 3.1.

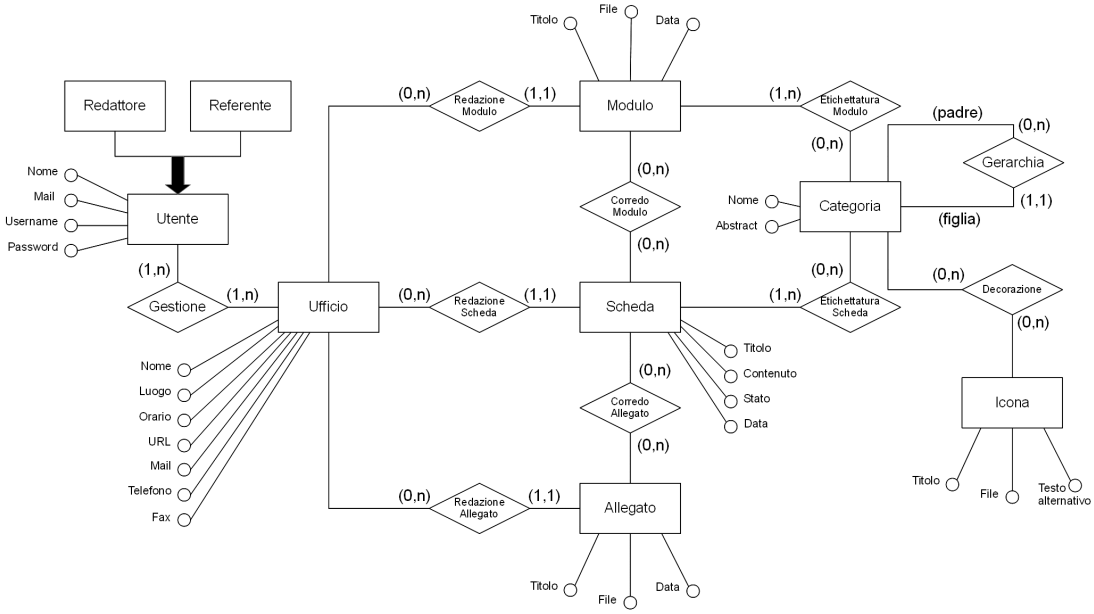

Figura 3.1: Schema E-R relativo alla progettazione concettuale

4 Progettazione Logica della base di dati

In questa fase è stato riorganizzato lo schema E-R ottenuto nel capitolo 3 operando una ristrutturazione. Si è tenuto conto dei volumi dei dati e delle caratteristiche delle operazioni principali che l'applicazione deve compiere. Prodotto finale di questa fase di progettazione è lo schema logico, ottenuto traducendo lo schema E-R ristrutturato.

I volumi dei dati, stimati in base all'implementazione della versione precedente dell'applicazione, sono riportati nella tabella 4.1:

Concetto	Tipo	Volume
Redattore	E	3
Referente	E	30
Utente	E	33
Gestione	R	60
Ufficio	E	60
Redazione Modulo	R	150
Modulo	E	150
Redazione Scheda	R	200
Scheda	E	200
Redazione Allegato	R	50
Allegato	E	50
Corredo Modulo	R	200
Corredo Allegato	R	100
Categoria	E	100
Etichettatura Modulo	R	200
Etichettatura Scheda	R	400
Gerarchia	R	100

Tabella 4.1: Tavola dei volumi

Le principali operazioni previste sono:

Operazione 1: Visita di una scheda, quindi recupero dei dati della scheda e dei relativi moduli, allegati e dati dell'ufficio di riferimento.

Operazione 2: Visita di un modulo.

Operazione 3: Visita dell'albero delle categorie, recupero dei dati della categoria, di tutta la relativa gerarchia e delle schede e dei moduli contenuti.

Operazione 4: Modifica di una scheda

Operazione 5: Modifica di un modulo

La quantità e il tipo delle operazioni sono riportate nella tabella 4.2,

Operazione	Tipo	Frequenza
Op. 1	I	250 al giorno
Op. 2	I	100 al giorno
Op. 3	I	300 al giorno
Op. 4	I	30 al mese
Op. 5	I	15 al mese

Tabella 4.2: Tavola delle operazioni

Non si è ritenuto necessario elencare tutte le operazioni possibili perché, altre operazioni, quali ad esempio l'inserimento di un nuovo ufficio o di un utente, sono così poco frequenti rispetto a quelle considerate che non hanno prodotto considerazioni significative al fine della traduzione dello schema E-R in schema logico.

4.1 Schema E-R ristrutturato

Nello schema E-R di figura 3.1 non sono presenti ridondanze quindi si è proceduto con l'eliminazione dell'unica generalizzazione presente. Si tratta di una generalizzazione totale ed esclusiva perché ogni occorrenza di ”Utente” è un'unica occorrenza di “Referente” o di “Redattore”. Si è quindi proceduto con l'accorpamento delle entità figlie al padre, aggiungendo all'entità “Utente” l'attributo “Ruolo”.

Non è stato necessario procedere poi ad alcun partizionamento o accorpamento di entità o associazioni.

La scelta degli identificatori primari merita invece una spiegazione più approfondita. Per ogni tabella entità si è infatti ritenuto opportuno aggiungere un ulteriore campo ID cui è stato dato il ruolo di chiave primaria. Nel capitolo 7.1 si vedrà come le chiavi delle entità si troveranno a far parte delle URL di alcune pagine dell'applicazione. L'assegnazione delle chiavi primarie su campi contenenti stringhe avrebbe presentato alcune criticità. Il sistema sarebbe stato suscettibile a impostazioni esterne quali la codifica del testo e inoltre i campi scelti sarebbero stati soggetti a troppe variazioni. A titolo esemplificativo, associare la chiave primaria per l'entità “scheda” al campo “titolo” (in coppia esternamente con l'ufficio di riferimento) avrebbe significato dover procedere a molti aggiornamenti

di tutti i vincoli referenziali, vista la suscettibilità al cambiamento del campo in questione.
Avendo adottato per tutte le entità il principio della presenza del campo ID si è potuto sviluppare un codice con delle caratteristiche di riusabilità più elevate.
Nel codice sviluppato risulterà particolarmente comodo far riferimento a una qualsiasi delle entità del progetto specificandone solamente il "tipo" (scheda, modulo, allegati, utente etc.) e un campo, comune a tutte, che è l'ID. L'accoppiata ID e "tipo" individuano univocamente un'entità.
Un esempio di questo è il sistema dei log. Una delle caratteristiche richieste all'applicazione è quella di essere in grado di generare delle statistiche di utilizzo. È pertanto necessario che sia in grado di raccogliere e analizzare dei log. I log per loro natura non necessitano di modifiche ma solo di inserimenti, per questo non è necessario un database per la loro raccolta. Tuttavia dovendo poi fare su di essi delle operazioni di analisi, con l'utilizzo di aggregazioni, risulta utile l'adozione di un database per la loro gestione.
Ai fini dell'applicazione si può considerare lo schema concettuale in figura 4.1.

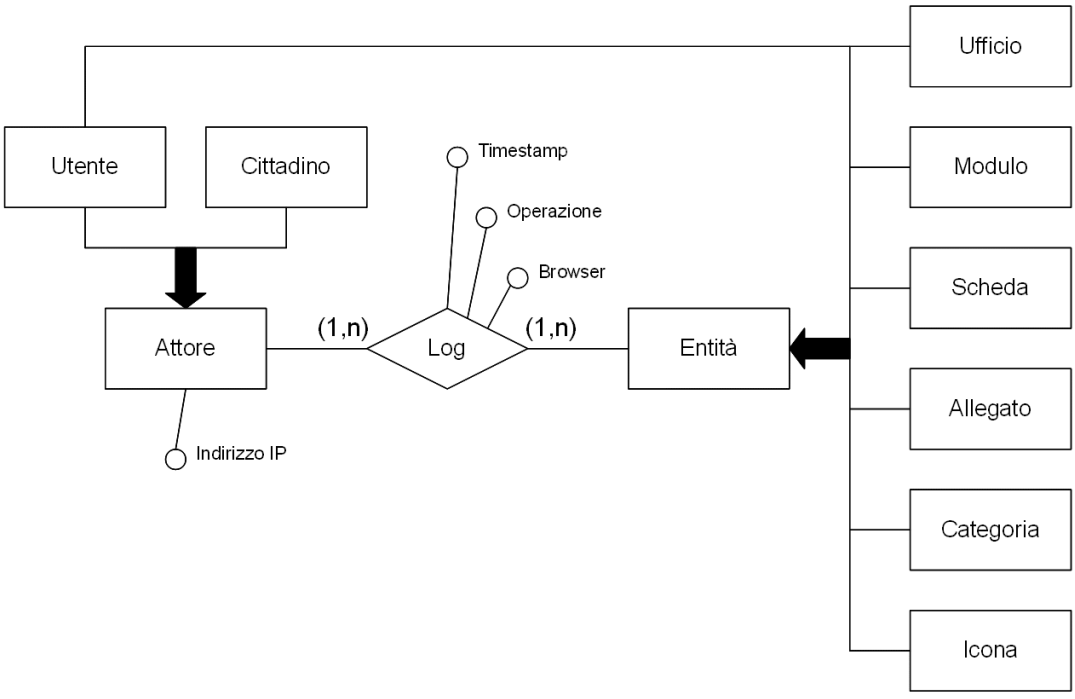

Figura 4.1: Schema concettuale dei log

La generalizzazione di "Entità" è stata risolta accorpando le figlie al padre, aggiungendo l'attributo "tipo". Questo è stato possibile proprio grazie alla presenza del campo ID in ogni entità.

Anche la generalizzazione dell'attore è stata risolta accorpando le figlie al padre, attribuendo un ID NULL all'attore "cittadino".
Lo schema diventa quello di figura 4.2

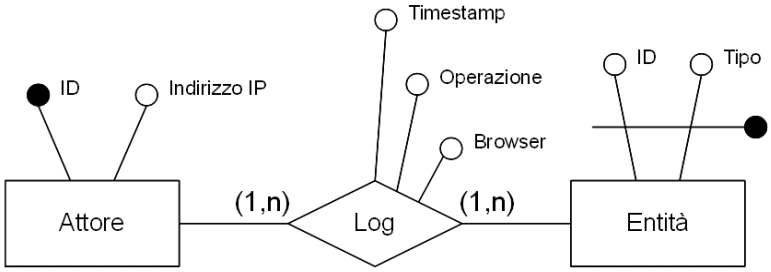

Figura 4.2: Schema concettuale dei log dopo l'eliminazione delle gerarchie

Ogni operazione fatta dall'applicazione sul database comporta l'aggiunta di un "log". L'operazione di scrittura di un "log" è la più frequente e avviene circa un migliaio di volte in un giorno. Si tratta quindi di una cardinalità molto elevata e destinata a una crescita costante. Per limitare il numero degli accessi necessari alla scrittura e alla consultazione si è proceduto a un accorpamento di entità.
Lo schema concettuale diventa quello di figura 4.3.

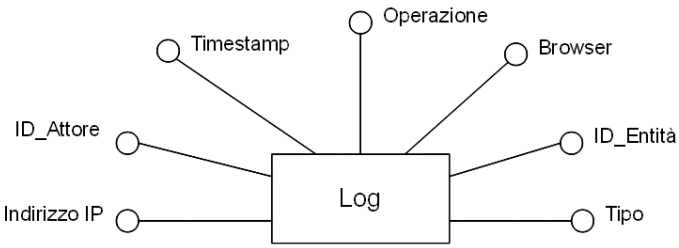

Figura 4.3: Schema concettuale dei log dopo accorpamento di entità

Lo schema logico che ne deriva è composto da un'unica tabella con i campi che corrispondono agli attributi dello schema concettuale ristrutturato.
In fase di progettazione fisica si è proceduto al cambiamento del nome di alcuni campi per motivi di uniformità con il database dell'applicazione.
Il contenuto della tabella "log" risulta facilmente leggibile. Un esempio è riportato nella tabella 4.3.

id_tipo	tipo	messaggio	id_utente	IP	timestamp	percorso	browser
12	ufficio	lock	39	192.168.2.249	2012-05-07 17:45:27		Mozilla/5.0 (W
12	ufficio	modificato	39	192.168.2.249	2012-05-07 17:45:48		Mozilla/5.0 (W
295	scheda	lock	33	192.168.2.249	2012-05-10 11:43:31		Mozilla/4.0 (c
295	scheda	modificata	33	192.168.2.249	2012-05-10 11:44:53		Mozilla/4.0 (c
295	scheda	lock	2	192.168.2.249	2012-05-10 12:00:20		Mozilla/5.0 (W
295	scheda	modificata	2	192.168.2.249	2012-05-10 12:01:44		Mozilla/5.0 (W
295	scheda	pubblicata	2	192.168.2.249	2012-05-10 12:01:44		Mozilla/5.0 (W
71	categoria	visitata	NULL	192.168.2.249	2012-05-10 13:00:32	Servizi per il cittadino -->	Mozilla/4.0 (c
281	scheda	visitata	NULL	192.168.2.249	2012-05-10 13:00:50	SUAP - Sportello Unico Attivita' Produttive --> Pubblici esercizi -->	Mozilla/5.0 (W
73	categoria	visitata	NULL	192.168.2.249	2012-05-10 13:00:55	Servizi per il cittadino --> Fare certificati e documenti -->	Mozilla/4.0 (c
248	allegato	visitato	NULL	192.168.2.249	2012-05-10 13:00:59	Scheda Somministrazione di alimenti e bevande in circoli privati affiliati	Mozilla/5.0 (W
281	scheda	visitata	NULL	192.168.2.249	2012-05-10 13:01:26	SUAP - Sportello Unico Attivita' Produttive --> Pubblici esercizi -->	Mozilla/5.0 (W
173	modulo	visitato	NULL	192.168.2.249	2012-05-10 13:01:32	@ Somministrazione di alimenti e bevande in circoli privati affiliati	Mozilla/5.0 (W
128	categoria	visitata	NULL	192.168.2.249	2012-05-10 13:01:41	Modulistica -->	Mozilla/4.0 (c
131	categoria	visitata	NULL	192.168.2.249	2012-05-10 13:01:44	Modulistica --> Anagrafe, Stato civile, Elettorale -->	Mozilla/4.0 (c
267	categoria	visitata	NULL	192.168.2.249	2012-05-10 13:01:47	Modulistica --> Anagrafe, Stato civile, Elettorale --> Stato Civile -->	Mozilla/4.0 (c
93	modulo	visitato	NULL	192.168.2.249	2012-05-10 13:01:55	Modulistica --> Anagrafe, Stato civile, Elettorale --> Stato Civile -->	Mozilla/4.0 (c
281	scheda	visitata	NULL	192.168.2.249	2012-05-10 13:03:15	SUAP - Sportello Unico Attivita' Produttive --> Pubblici esercizi -->	Mozilla/5.0 (W
267	categoria	visitata	NULL	192.168.2.249	2012-05-10 13:03:32	Modulistica --> Anagrafe, Stato civile, Elettorale --> Stato Civile -->	Mozilla/4.0 (c
38	allegato	visitato	NULL	192.168.2.249	2012-05-10 13:03:43	Scheda Somministrazione di alimenti e bevande in circoli privati affiliati	Mozilla/5.0 (W
281	scheda	visitata	NULL	192.168.2.249	2012-05-10 13:04:05	SUAP - Sportello Unico Attivita' Produttive --> Pubblici esercizi -->	Mozilla/5.0 (W
132	allegato	visitato	NULL	192.168.2.249	2012-05-10 13:04:10	Scheda Somministrazione di alimenti e bevande in circoli privati affiliati	Mozilla/5.0 (W
281	scheda	visitata	NULL	192.168.2.249	2012-05-10 13:04:16	SUAP - Sportello Unico Attivita' Produttive --> Pubblici esercizi -->	Mozilla/5.0 (W

Tabella 4.3: Esempio di contenuto della tabella "log"

È proprio grazie all'adozione degli ID nelle varie entità che si è potuto arrivare a questo sistema di log. Questo schema concettuale però non è relativo alla descrizione dei dati oggetto dell'applicazione ma all'organizzazione interna del meccanismo di funzionamento dell'applicazione ed è anche una conseguenza del linguaggio adottato. Nel capitolo 7.1 verrà descritto come l'uso del campo ID, quale riferimento alle entità, ha portato dei vantaggi relativi alla riusabilità del codice.

Come descritto più approfonditamente nel capitolo 7.2, l'applicazione utilizza lo stesso principio, cioè quello di individuare un'entità con la coppia di attributi ID e "tipo", anche per il sistema dedicato alla gestione della programmazione concorrente.

Lo schema E-R ristrutturato è quello di figura 4.4.

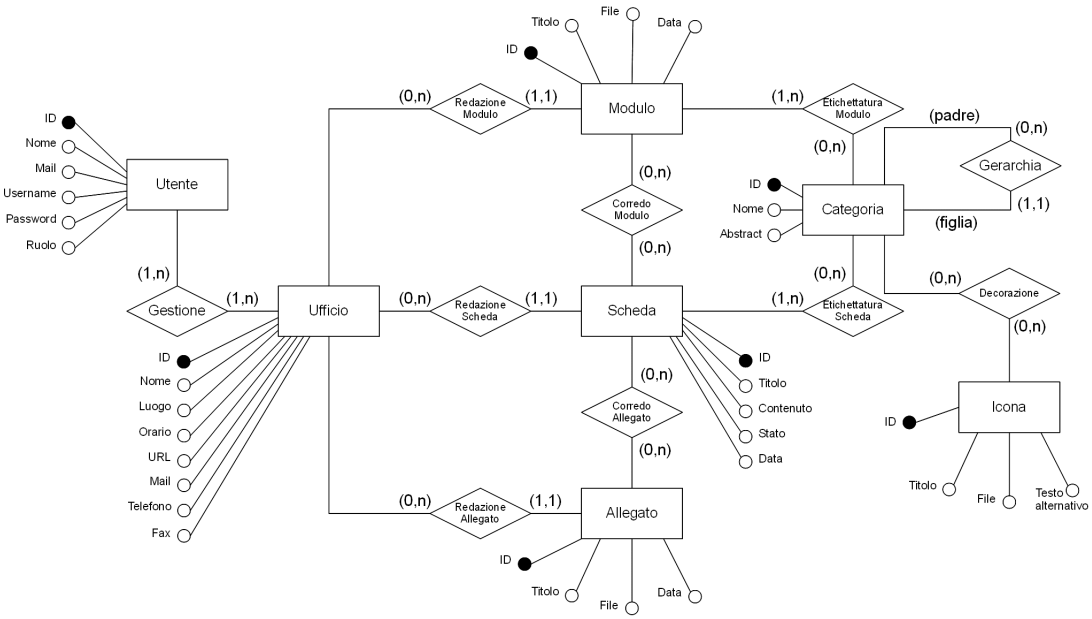

Figura 4.4: Schema E-R ristrutturato

4.2 Schema logico

Lo schema logico, risultante dalla traduzione dello schema E-R, è quello visibile in figura 4.5.

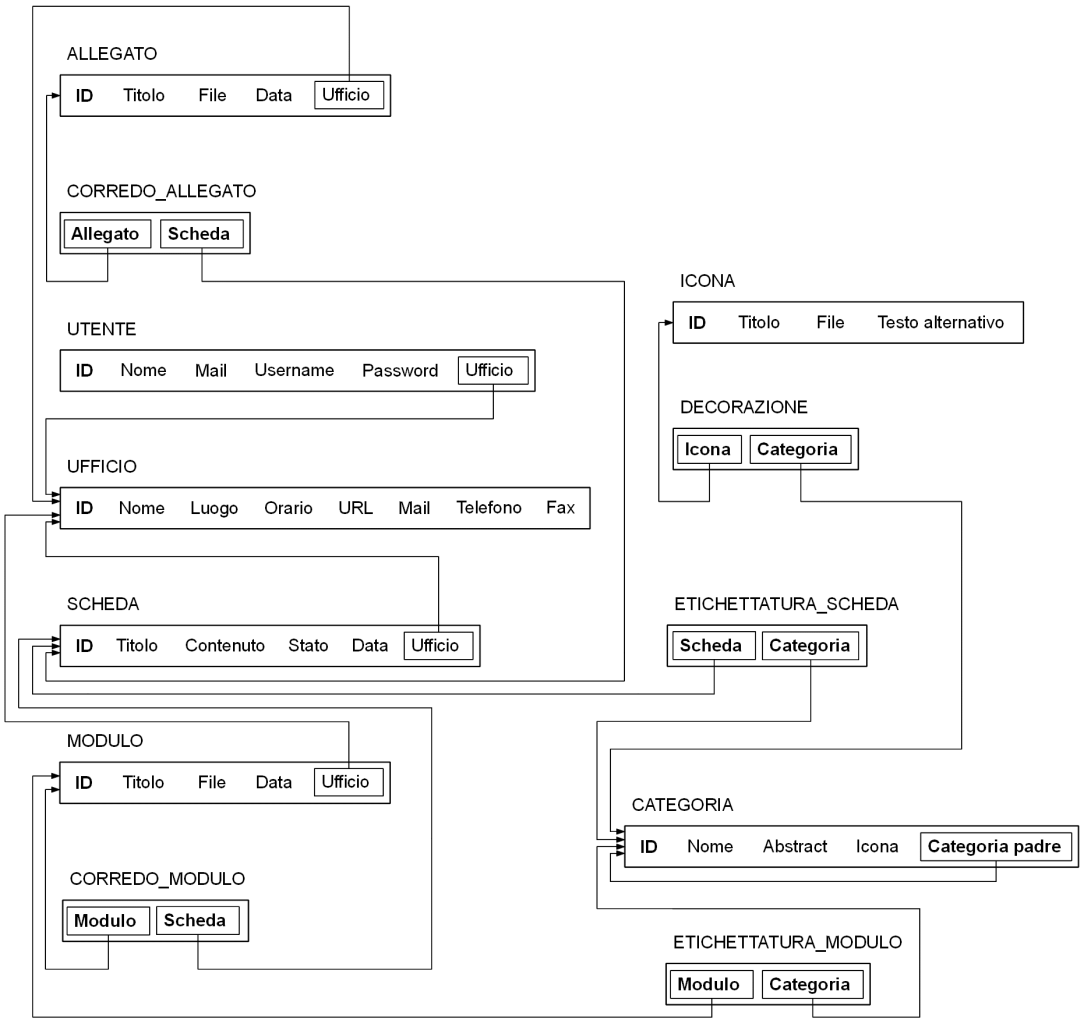

Figura 4.5: Schema logico

5 Progettazione Fisica della base di dati

Lo storage engine MyISAM non supporta le transazioni per cui nella progettazione fisica non sono state definite chiavi esterne.
Nel database le tabelle di relazione sono state rinominate con il prefisso "afferenza", in questo modo è possibile capire facilmente la struttura del database individuando immediatamente le relazioni che vi sono tra le entità.
La relazione "Etichettatura Modulo", ad esempio, è stata implementata nella tabella chiamata "afferenza_modulo_categoria". Da questo si capisce che "modulo" e "categoria" sono delle entità e che sono in relazione tra loro con una cardinalità di tipo molti a molti.
Oltre alla tabella "log", nel database vi sono delle altre tabelle di servizio quali la tabella "template" che ospita l'interfaccia grafica del front-end, la tabella "blocco" che gestisce l'accesso concorrente in modifica delle entità e la tabella "log-sys" che ospita i messaggi dovuti a eventuali errori dell'applicazione.

Nella progettazione fisica di un database è opportuno considerare anche la definizione degli indici. L'aggiunta di un indice è utile per velocizzare le operazioni di recupero dei dati ma comporta un rallentamento per le operazioni di aggiornamento perché il sistema, a ogni modifica, deve procedere anche all'aggiornamento degli indici.
Per accedere alle entità si è usato spesso il campo ID, come spiegato nel capitolo 7.1. Il campo ID è già chiave primaria delle varie entità per cui non occorre definire su di esso alcun indice. MySQL infatti definisce automaticamente un indice per ogni chiave primaria. Tutte le tabelle di afferenza, ossia le tabelle di relazione molti a molti, sono composte da due campi che sono gli ID delle entità che collegano. La chiave primaria in tali relazioni è già definita dalla coppia dei due ID.
Le operazioni che non si basano sugli ID non sono molte. A dimostrazione, per ricostruire l'albero delle categorie è necessario effettuare una ricerca in base al campo "Categoria padre". Si tratta di un'operazione molto frequente perché necessaria alla navigazione del front-end. L'aggiornamento di una categoria non è invece una cosa molto frequente, per cui si è scelto di adottare un indice sull'attributo "Categoria padre" della tabella "categoria".
Più delicato è il caso delle queries che utilizzano i dati della tabella "log", ad esempio per effettuare le analisi statistiche. Si tratta di una tabella destinata a ospitare migliaia di tuple. La scrittura dei log è l'operazione più frequente perché

al minimo ne avviene una a ogni pagina visitata da un cittadino. Quindi è opportuno valutare attentamente se l'attività di aggiornamento degli indici comporta dei rallentamenti significativi sulla navigazione del front-end. Per questo si è preferito verificare le scelte effettuate dopo la messa in funzione dell'applicazione utilizzando un'analisi empirica.
Il database deve essere interrogato per capire quante volte è stata visitata una scheda da quando è stata creata. In questo caso le informazioni vengono estratte utilizzando i campi "id_tipo", "tipo" e "messaggio". Si è visto empiricamente che, grazie all'aggiunta di un indice costruito sui tre attributi sopra citati, il tempo di esecuzione di una query di questo tipo è passato da 2,95 secondi a 0,32 secondi, mentre la navigazione nel front-end non ha subito particolari rallentamenti.
E' stato anche richiesto di avere la medesima informazione riferita però a un determinato intervallo di tempo, e di generare una classifica delle entità più visitate. Il recupero di queste informazioni è basato sul campo "timestamp" che descrive in quale secondo è avvenuta l'operazione cui si riferisce il log. Un ulteriore indice creato su questo campo, che praticamente ospita valori tutti diversi, avrebbe la stessa cardinalità della tabella, quindi non apporterebbe miglioramenti[1].
Dopo la messa in funzione dell'applicazione si è visto che le queries che generano le classifiche impiegano dai 6 agli 8 secondi che, vista la scarsa frequenza di queste operazioni, rappresentano un tempo accettabile.

Per l'implementazione del database è stato utilizzato il software PhpMyAdmin [20]. Si tratta di un client, scritto in codice PHP, che mette a disposizione una pratica interfaccia grafica per gestire i database MySQL.
Per l'analisi dei tempi di esecuzione delle queries sono stati usati strumenti come lo Slow Query Log di MySQL in combinazione con il comando mysqldumpslow.

1 Lo storage engine adottato non consente la creazione di un indice parziale sul campo "timestamp".

6 Motore di ricerca

In un'applicazione come questa che si propone di fornire informazioni al pubblico è molto importante il recupero delle stesse anche tramite una ricerca libera. Per questo ll motore di ricerca ha un ruolo fondamentale. In particolare è essenziale che l'esito della ricerca proponga le informazioni ordinate per pertinenza.

Il tempo destinato a questa parte del progetto non era tale da permettere l'utilizzo di avanzate tecniche di information retrieval, quindi è stato adottato un approccio diverso.

Partendo dall'analisi delle ricerche effettuate più frequentemente sul sito istituzionale e dall'analisi del contenuto delle schede informative della precedente versione della guida, sono state individuate le principali problematiche.

Si è notato che alcune ricerche coinvolgono parole molto comuni quali i verbi (ad esempio “devo fare la carta d'identità”), che alcune ricerche utilizzano un lessico non presente nelle schede informative ma che fanno un chiaro riferimento ad alcuni servizi forniti dall'ente (per esempio “cancello” molto probabilmente riferito a “passo carraio”), che vi sono molti servizi simili tra loro che di conseguenza usano un lessico molto comune nelle relative schede descrittive (“Carta d'identità”, “Carta d'identità per cittadini minorenni” e “Carta d'identità per cittadini stranieri”) e che alcuni concetti non sono individuabili con una parola sola (ad esempio la parola “vendita” può essere riferita sia alla “vendita al dettaglio” che alla “vendita di liquidazione”).

Per tutto questo si è preferito implementare un sistema di ricerca personalizzato e parametrizzabile, piuttosto che utilizzare costrutti già esistenti quali MATCH e AGAINST di MySQL.

Sono stati adottati di conseguenza alcuni accorgimenti. In particolare è stato aggiunta la possibilità di associare alle schede e ai moduli delle “parole chiave”, ed è stato ideato un algoritmo basato su una somma pesata.

L'adozione delle “parole chiave”, oltre a essere uno strumento utile per il recupero e l'ordinamento dei risultati in base alla pertinenza, serve per risolvere la problematica della ricerca con lessico non appropriato o non presente nelle schede.

L'algoritmo di ricerca si compone di diverse fasi.

- Nella prima fase viene fatta un'operazione di elaborazione della stringa da cercare. Vengono individuate le singole parole ed eliminate le stopwords (articoli, congiunzioni, etc.). Vengono poi ricavate le radici delle parole

togliendo le desinenze più comuni, questo per includere casistiche come il plurale/singolare.

- Nella seconda fase vengono recuperate dal database tutte le schede e i moduli che contengono nel "titolo", nel "contenuto" o nelle "parole chiave" almeno una delle parole cercate.
- Nella terza fase, con l'utilizzo di una somma pesata, viene assegnato a ogni scheda/modulo recuperato un valore calcolato in base al numero di occorrenze trovate e al tipo di campi in cui sono state trovate queste occorrenze. Viene quindi ordinato l'elenco di schede/moduli recuperati in base a questo valore.

Il valore assegnato a ogni/scheda modulo è il risultato della somma di due componenti.

Una prima componente considera, per ogni parola cercata, il numero di occorrenze trovate nel "titolo", nel "contenuto" e nelle "parole chiave" della scheda/modulo. Viene dato un peso diverso alle occorrenze per ogni tipo di campo. È risultato utile attribuire maggior peso alle occorrenze del campo "titolo" e del campo "parole chiave" piuttosto che quelle relative al campo "contenuto". Questo per rispondere al problema del lessico comune. Si è visto infatti che le parole del titolo hanno una caratterizzazione/differenziazione maggiore di quelle presenti nel contenuto.

Una seconda componente considera quante parole, tra quelle da cercare, sono presenti contemporaneamente nel "titolo", nel "contenuto" o nelle "parole chiave". Anche qui viene dato un peso diverso a seconda se c'è presenza nel "titolo", nel "contenuto" o nelle "parole chiave". Questa seconda componente serve per rispondere al problema che alcuni concetti sono identificati da più di una parola.

Ognuna di queste due componenti ha un peso parametrizzato.

In sintesi, il valore attribuito alle schede trovate, viene calcolato con la formula

$$V_{TOT} = P_r \cdot V_r + P_n \cdot V_n \quad ,$$

con

$$V_r = P_r^T \cdot \sum_{\forall w \in S} r^T(w) + P_r^K \cdot \sum_{\forall w \in S} r^K(w) + P_r^C \cdot \sum_{\forall w \in S} r^C(w)$$

e

$$V_n = P_n^T \cdot n^T(S) + P_n^K \cdot n^K(S) + P_n^C \cdot n^C(S) \quad .$$

Il significato dei simboli è il seguente:

w parola da cercare

S insieme delle parole da cercare (o stringa da cercare)

V_{TOT}: valore complessivo attribuito alla scheda trovata

P_r: peso della componente relativa al numero di occorrenze delle w

V_r: valore attribuito al numero di occorrenze delle w

P_n: peso della componente relativa al numero di parole di S presenti

V_n: valore attribuito al numero di parole di S presenti

P_r^T: peso assegnato al numero di occorrenze di w nel titolo

$r^T(w)$: numero di occorrenze della parola w nel titolo

P_r^K: peso assegnato al numero di occorrenze di w nelle parole chiave

$r^K(w)$: numero di occorrenze della parola w nelle parole chiave

P_r^C: peso assegnato al numero di occorrenze di w nel contenuto

$r^C(w)$: numero di occorrenze della parola w nel contenuto

P_n^T: peso assegnato alla presenza delle parole S nel titolo

$n^T(S)$: numero di parole di S trovate nel titolo

P_n^K: peso assegnato alla presenza delle parole S nelle parole chiave

$n^K(S)$: numero di parole di S trovate nelle parole chiave

P_n^C: peso assegnato alla presenza delle parole S nel contenuto

$n^C(S)$: numero di parole di S trovate nel contenuto

Il valore attribuito ai moduli trovati, viene calcolato nella stessa maniera ma considerando, al posto del campo "contenuto", il campo "file"[2]. Per cui si ha:

$$V_{TOT} = P_r \cdot V_r + P_n \cdot V_n \quad ,$$

$$V_r = P_r^T \cdot \sum_{\forall w \in S} r^T(w) + P_r^K \cdot \sum_{\forall w \in S} r^K(w) + P_r^F \cdot \sum_{\forall w \in S} r^F(w) \quad ,$$

$$V_n = P_n^T \cdot n^T(S) + P_n^K \cdot n^K(S) + P_n^F \cdot n^F(S)$$

dove (oltre a quanto prima illustrato)

P_r^F: peso assegnato al numero di occorrenze di w nel file

$r^F(w)$: numero di occorrenze della parola w nel file

P_n^F: peso assegnato alla presenza delle parole S nel file

$n^F(S)$: numero di parole di S trovate nel file

Il valore dei parametri della somma pesata è stato regolato empiricamente simulando delle ricerche tra quelle effettuate più frequentemente sul sito istituzionale e verificando i risultati della ricerca.
Il valore impostato dei parametri è riportato nella tabella 6.1.

2 Inteso come nome del file.

scheda	modulo
$P_r := 1$	$P_r := 1$
$P_n := 1$	$P_n := 1$
$P_r^T := 100$	$P_r^T := 100$
$P_r^K := 25$	$P_r^K := 25$
$P_r^C := 10$	$P_r^F := 10$
$P_n^T := 100$	$P_n^T := 100$
$P_n^K := 20$	$P_n^K := 19$
$P_n^C := 5$	$P_n^F := 5$

Tabella 6.1: Valore dei parametri del motore di ricerca

Per dare all'utente una misura dell'attinenza degli elementi trovati nel front-end è stato associato accanto a ogni elemento recuperato un numero di stelline proporzionali al valore attribuito con la formula precedente.

7 Implementazione e utilizzo dell'applicazione di basi di dati

In questa fase si è proceduto non solo alla realizzazione delle pagine html che fanno da interfaccia tra gli utenti del sistema e il database, ma anche alla scrittura di quelle parti di codice che completano la logica applicativa.

7.1 Struttura del codice

Il codice è stato diviso in files che a loro volta sono stati organizzati in directory. Questo è stato fatto per rispondere soprattutto a esigenze di leggibilità ma anche di sicurezza.
Si è infatti scelto di includere tutti gli script relativi al back-end in una cartella denominata "gestione", il cui accesso può venire ulteriormente controllato dalle opzioni che il server web Apache mette a disposizione. In questo modo si è potuto limitare l'accesso a questa cartella solo agli utenti con IP appartenete alla LAN dell'ente. La direttiva utilizzata nel file di configurazione di Apache "httpd.conf" è la seguente.

```
<Directory <<gestione>>/>
     Allow from <<IP lan interna>>
     Deny from All
</Directory>
```

Per gli stessi motivi di sicurezza si è scelto di posizionare al di fuori della document root del server web, la directory dove l'applicazione deposita e gestisce i files inseriti dagli utenti (moduli, allegati e icone). Questo garantisce che solo gli script della "Guida ai Servizi" consentono l'accesso e il controllo dei suddetti files.

Per rispondere alle esigenza di leggibilità e di riuso, nella stesura del codice si è cercato di seguire il paradigma MVC (Model View Controller) che suggerisce di separare il codice funzionale al modello, il codice relativo ai meccanismi di gestione e controllo, e il codice per la realizzazione dell'interfaccia con l'utenza.
Il codice funzionale al modello, cioè quello necessario per gestire gli elementi descritti nel database, è organizzato in classi, ognuna corrispondente a un'entità. Lo scopo di queste classi è quello di mettere a disposizione le funzioni che

interagiscono con il database. Queste classi sono organizzate in files che hanno lo stesso nome dell'entità cui si riferiscono. Ad esempio, corrispondentemente all'entità "modulo", è stato creato il file "modulo.php".
Nel file si trova la definizione della classe "modulo" che contiene tutti i metodi necessari per accedere agli attributi dell'entità "modulo" scritti nel database. Nel codice relativo alle pagine di gestione non sono presenti script SQL, separando così lo strato di codice relativo al modello da quello di controllo.
In particolare si è scelto di avere due costruttori per ogni classe.
Un costruttore senza parametri che scrive nel database l'entità a partire dai dati che arrivano dalla pagina con la form di inserimento. Questo costruttore rende persistente nel database un'entità e i suoi attributi e mette a disposizione nello script l'oggetto appena istanziato.
Un costruttore con un unico parametro che è l'ID dell'entità. Come si è visto nel capitolo 4.1, ogni oggetto derivante da un'entità è univocamente individuato dal tipo di entità cui si riferisce e dall'ID. Tale costruttore recupera dal database i dati e crea un oggetto PHP che può essere utilizzato dal codice. A titolo esemplificativo per recuperare dal database il modulo con ID uguale a 35, si può utilizzare il codice seguente:

```
$modulo_da_recuperare = new modulo (35);
```

A questo punto gli attributi del modulo sono utilizzabili tramite chiamate di questo tipo:

```
$modulo_da_recuperare->titolo
```

Il codice che si occupa del controllo dell'applicazione (Controller) è diviso in files corrispondenti ai passi dell'applicazione, ossia alle pagine html. Questo ha il vantaggio di rendere il listato facilmente individuabile quindi più semplice da correggere o modificare. Infatti gli script che si sono ottenuti sono piuttosto corti e inoltre è sufficiente guardare l'URL del browser per capire dove si trova il codice che si sta eseguendo.
Tutte le procedure, quali l'inserimento, la modifica o la cancellazione di un'entità, sono state divise in tre pagine con le seguenti funzionalità:

1. pagina di elenco e selezione dell'entità da gestire
2. pagina con form o azione di modifica dei campi/attributi dell'entità selezionata nella pagina precedente
3. pagina che effettua le modifiche precedentemente espresse nel database

Quest'ultima pagina non produce alcun output html ed è fatta in modo che non possa essere rieseguita più volte, come avviene con la pressione del tasto “back” del browser. Questo effetto si ottiene utilizzando al termine delle operazioni di scrittura nel database l'istruzione PHP

```
header("Location: pagina_di_selezione_delle_entità.php");
```

che restituisce in risposta al browser, non la pagina che effettua l'inserimento nel database, ma la prima pagina di elenco e selezione delle entità.

Lo schema di suddivisione delle procedure di inserimento/modifica/cancellazione delle entità in pagine PHP è visibile in figura 7.1 dove si può notare anche che il passaggio dei dati tra la prima e la seconda pagina avviene con metodo GET mentre tra la seconda e la terza avviene con metodo POST.

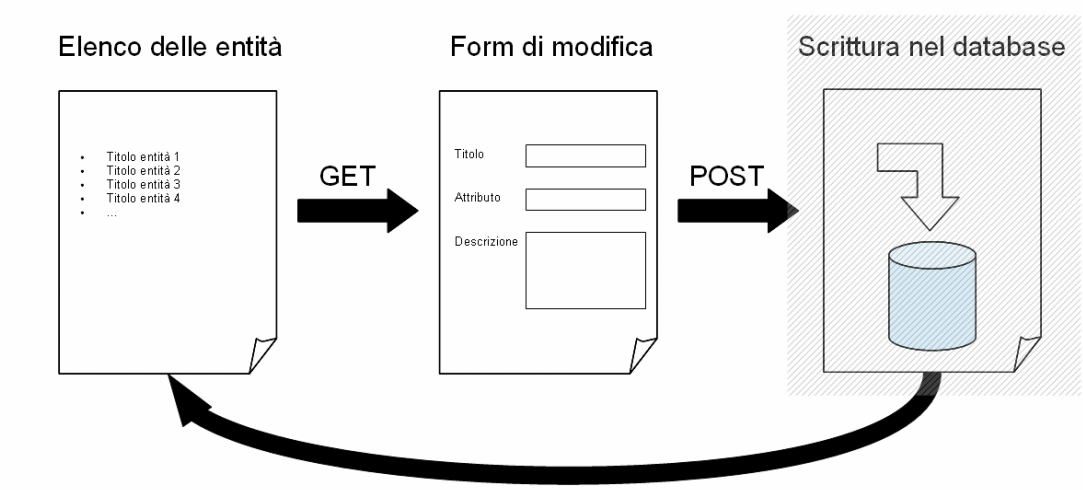

Figura 7.1: Suddivisione delle procedure di inserimento/modifica/cancellazione in pagine PHP

Tra la prima e la seconda pagina è stato adottato il metodo GET per realizzare un meccanismo più efficiente di comunicazione tra gli attori del sistema. Nelle mail che l'applicazione genera automaticamente per avvisare i referenti e i redattori dei passaggi di stato delle schede e dei moduli, si è voluto inserire un URL che porta direttamente alla pagina di modifica dell'entità cui la mail si riferisce. Il metodo GET è particolarmente adatto a questo scopo perché inserisce le variabili e il loro contenuto nell'URL.

La seconda pagina necessità di un'unica variabile che è l'ID dell'entità da modificare. Vista la cardinalità delle schede e dei moduli si tratta di un numero piuttosto piccolo quindi ampiamente compatibile con le dimensioni imposte dal metodo GET.

Quando un referente chiede la pubblicazione di una scheda, il redattore riceve una mail in cui compare un link che porta direttamente alla pagina in cui può operare delle correzioni e procedere alla pubblicazione.
Il link ha la seguente forma:
http://www.comune.treviso.it/GuidaAiServizi/gestione/gestione_scheda2.php?id_scheda=38
Anche tra le pagine che realizzano il front-end il passaggio dei dati avviene in modalità GET per mettere a disposizione dell'utenza degli URL che possono essere conservati o scambiati tramite mail. Nel front-end le variabili transate sono poche e si tratta quasi sempre di ID di categorie, schede, moduli o allegati.
La terza pagina di scrittura nel database necessità invece di più variabili dalla dimensione talvolta elevata. Il contenuto della scheda, ad esempio, è costituito da un testo html che può raggiungere dimensioni superiori ai 2000 caratteri che è opportuno non superare in un URL per mantenere una maggiore compatibilità con i browser attualmente presenti.
Per questo è stato adottato il metodo POST che consente il passaggio di variabili con dimensioni molto elevate.

Vi sono delle porzioni di codice comuni a molte pagine, come quelle che si occupano della connessione al database o dell'autenticazione.
L'accesso del personale alla precedente realizzazione dell'applicazione era basato sull'anagrafica delle utenze Lotus Domino, ossia l'anagrafica di tutti i dipendenti dotati di una casella di posta elettronica. Anche nella nuova versione il back-end prevede un meccanismo di login, basato su credenziali tipiche quali il nome utente e la password, ed è stato realizzato con le variabili di sessione di PHP. Questo sistema, per funzionare, richiede che sul browser degli utenti siano abilitati i cookies. L'elenco degli utenti è custodito nel database e come si è visto nel capitolo 4.1 si tratta di un'entità.
Gli script PHP che compongono il sistema di autenticazione sono collocati su files separati i quali vengono inclusi all'inizio di ogni pagina che necessita di un accesso controllato. Per questo motivo nelle pagine di gestione compare la seguente riga di codice:

```
include ("login.php");
```

Si è usato lo stesso criterio anche per i parametri generali dell'applicazione, come il dominio di riferimento, i dati per l'accesso al database, i percorsi delle

directory e altro, che sono stati raccolti in un unico file denominato "config.php" che viene incluso in ogni pagina.

La separazione tra il codice funzionale al controllo (controller) e il codice funzionale all'interfaccia grafica (view) è meno evidente. L'intestazione e il fondo delle pagine sono raccolte in files separati e vengono richiamati con l'istruzione "include:

```
include ("header_html.php");
include ("footer_html.php");
```

All'interno dello stesso script si possono distinguere due blocchi. Il primo, dedicato al controllo, gestisce le variabili che vengono poi utilizzate nel secondo blocco che si occupa di generare l'html necessario all'interfaccia.
Per rendere più netta la separazione sarebbe stato opportuno adottare un motore di template quali ad esempio "smarty" [21], tuttavia la lunghezza e la complessità dei file ottenuti non è tale da giustificare l'introduzione di un nuovo linguaggio.

7.2 Programmazione concorrente

Visto l'accesso contemporaneo ai dati dell'applicazione da parte degli utenti, si è reso necessario considerare le problematiche relative alla gestione di risorse condivise.

In fase di scrittura dei dati nel database è necessario accertarsi che, se la scrittura coinvolge più tabelle, questa avvenga in maniera consistente, evitando che altre scritture contemporanee disallineino i puntatori tra le tabelle. Come detto lo storage engine MyISAM non supporta le transazioni e le chiavi esterne che risolverebbero questo tipo di problema.
Visto l'esiguo numero delle scritture nel database, relative a modifiche di schede o altre entità, e la velocità con cui queste avvengono[3], è stato adottato un meccanismo di lock di tutte le tabelle. In questo modo non è necessario dichiarare anticipatamente quali saranno le tabelle coinvolte in ogni pagina che effettua delle operazioni di scrittura. Le pagine in questione risultano così avere una forma comune e questo ne migliora il riuso.
La sezione critica nel codice ha la seguente forma:

3 Avvengono circa 2 scritture al giorno che coinvolgono più tabelle contemporaneamente e raramente superano 3 Kbyte di dati inseriti per volta.

```
lockdb();
<operazioni di scrittura nel database>
unlockdb();
```

Per la realizzazione della funzione "lockdb()" è stato utilizzato il costrutto "LOCK TABLES" di MySQL in modalità "WRITE". In questo modo, nel momento in cui avviene l'aggiornamento dei dati, nel database non è possibile effettuare contemporaneamente nessun'altra operazione. Nel front-end non può così capitare di visualizzare schede con informazioni non congruenti, ad esempio con associazioni errate tra schede e moduli.
L'accesso ai dati da parte del pubblico non rappresenta un problema perché avviene solo in lettura. Più precisamente l'unica operazione di scrittura è quella di inserimento dei nuovi log che non permettono, per loro natura, un accesso in modifica da parte di nessuno, pertanto non si può considerare una criticità.

Nel back-end potrebbe avvenire che più referenti o redattori intendano modificare una stessa entità (scheda, modulo, allegato, etc.) che è dunque una risorsa condivisa. Diventa così necessario che un solo utente alla volta possa accedere alla pagina di modifica di una stessa entità. Per fare questo è stato implementato un semaforo binario [5].
Quando un utente desidera accedere alla pagina di modifica di un'entità viene verificato se la stessa è già in uso ad altri consultando una tabella di servizio chiamata "blocco".
Come per il sistema dei log, descritti nel capitolo 4.1, in questa tabella un'entità è individuata univocamente dall'attributo ID e dall'attributo "tipo".
La tabella blocco è costituita dai seguenti campi:

- **id_tipo:** chiave primaria dell'entità soggetta a modifica;
- **tipo:** tipo dell'entità soggetta a modifica (scheda, modulo, utente etc.);
- **id_utente:** chiave primaria dell'utente che sta effettuando la modifica;
- **timestamp:** orario di inizio dell'operazione di modifica.

Se l'entità è libera, l'applicazione la occupa iscrivendola nella tabella "blocco", altrimenti segnala all'utente che la risorsa è occupata e non fa comparire la form di modifica. Questo costituisce l'operazione di wait del semaforo binario.
Quando la scrittura nel database viene completata, o comunque quando l'utente visualizza una qualsiasi altra pagina, la risorsa viene liberata cancellandola dalla tabella "blocco". Questo costituisce l'operazione di signal del semaforo binario.

Purtroppo con l'utilizzo di PHP non è possibile sapere con certezza se un utente dopo aver compilato la form di modifica dell'entità ha completato la scrittura nel database. L'utente potrebbe infatti chiudere il browser, anche accidentalmente, senza procedere al salvataggio del proprio lavoro. In questo caso la risorsa potrebbe rimanere occupata per un tempo indefinito ossia finché l'utente non accede nuovamente all'applicazione. Per questo è stata data la possibilità ai redattori di forzare lo sblocco di una risorsa.
L'utilizzo della tabella "blocco" è in un una sezione critica del codice che viene eseguita all'inizio della pagina e ne permette un accesso esclusivo.
Questo è un esempio di codice che utilizza questa procedura:

```
lock_blocco();
$lock=islock("modulo", $id_modulo);
if ($lock!=FALSE) {
     $utente = new utente($lock->id_utente);
     echo "ATTENZIONE: modulo in redazione da ".$utente-
>nome." (".$lock->timestamp.")";
     include ("fine_pagina.php");
     include ("footer_html.php");
     die();
}
lock("modulo", $id_modulo);
unlock_blocco();
```

La funzione "islock" restituisce un oggetto "lock" che contiene i dati relativi all'occupazione della risorsa, se questa è occupata, mentre ritorna FALSE se è libera.
Si può notare che per confinare la sezione critica del codice non si è fatto uso della funzione "lockdb()" che occupa tutte le tabelle, ma si è preferito implementare la funzione "lock_blocco()" che occupa solo la tabella "blocco". Questo perché si tratta di una parte di codice ricorrente nell'applicazione e si è ritenuto utile una gestione meno invasiva dei lock del database.

7.3 Back-end

L'interfaccia di back-end, visibile in figura 7.4, è stata dotata di un menù, presente in tutte le pagine, le cui voci rimandano alle schermate di gestione delle varie entità e cambiano a seconda del ruolo dell'utente.
I referenti hanno la possibilità di gestire le schede, gli allegati, i moduli e gli uffici di propria competenza. I redattori, oltre a gestire i contenuti di tutti i referenti, possono gestire le categorie, le relative icone, gli utenti e possono accedere alla sezione delle statistiche.
I due diversi menù sono visibili in figura 7.2 e 7.3.

Figura 7.2: Menù del redattore

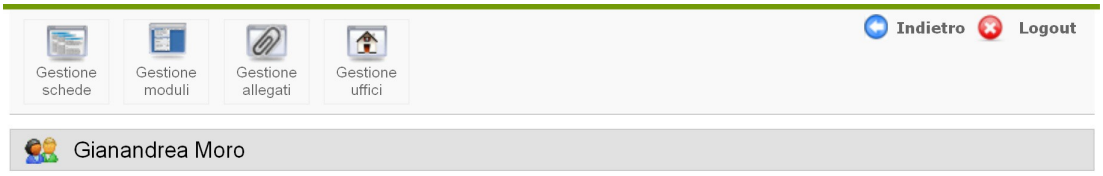

Figura 7.3: Menù del referente

Anche il codice relativo al menù, essendo comune a molte pagine è stato raccolto in un file e richiamato con l'istruzione "include":

```
include ("intestazione_utente.php");
```

Il sistema delle statistiche, consultabile dai redattori, si basa sui dati raccolti nei log. L'interfaccia permette di scegliere il periodo da esaminare e aggregare i risultati evidenziando non solo il numero di visite di ogni entità del sistema (schede, moduli, allegati e categorie) ma anche una serie di dettagli ulteriori. È possibile, per esempio, vedere quali sono stati i percorsi di categorizzazione o le ricerche libere che hanno utilizzato i cittadini per raggiungere una particolare scheda o un particolare modulo.

Tra le problematiche che si è dovuto affrontare relativamente al back-end sono di particolare rilievo l'interfaccia di editing del testo delle schede e il meccanismo di upload dei files relativi a moduli, allegati e icone. Questi sistemi devono essere semplici ed efficaci per rendere l'utilizzo dell'applicazione il più veloce e intuitivo possibile.

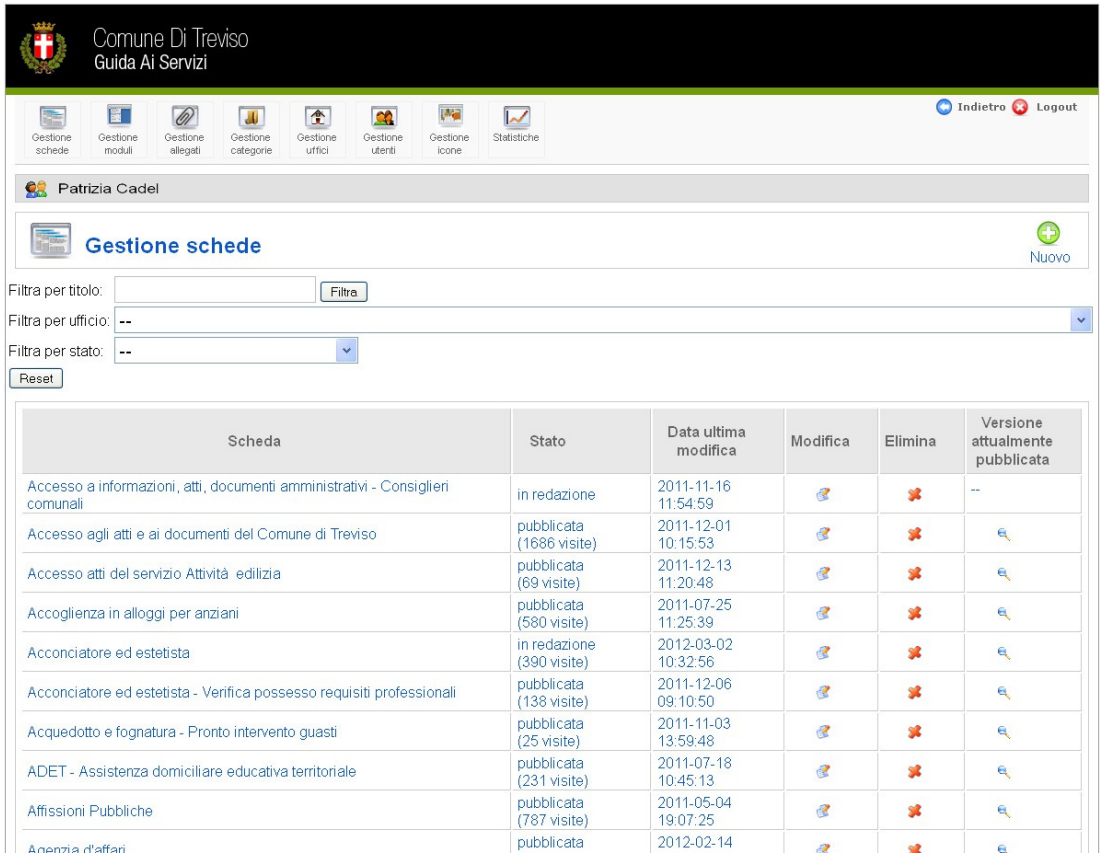

Figura 7.4: Schermata del back-end

7.3.1 Editor wysiwyg

Per gestire il contenuto della scheda informativa si è scelto di inserire nella form di inserimento/modfica un editor html di tipo wysiwyg. L'editor però deve poter mettere a disposizione solo le funzioni necessarie all'applicazione. Questo aspetto ha dei particolari risvolti sul fronte dell'accessibilità. Infatti, se un editor lasciasse modificare i font, le dimensioni o i colori dei caratteri in maniera indiscriminata ci sarebbe il rischio che gli utenti producano inconsapevolmente del codice html non compatibile con le direttive sull'accessibilità. Queste infatti

non permettono di imporre una dimensione fissa dei caratteri che devono invece essere sempre ridimensionabili a piacere dal fruitore della scheda.
E' stato quindi effettuato un confronto tra gli editor open source più diffusi.
In particolare sono stati valutati CKEditor [22], TinyMCE [23] e Xhina [24].
Tutti questi editor sono realizzati in codice javascript e sono tutti ricchi di funzionalità e molto personalizzabili. Per tutti è disponibile una traduzione in italiano più o meno completa.
Riassumo ora le caratteristiche distintive di ognuno.

7.3.1.1 CKEditor

CKEditor è il nome che dal 2009 caratterizza l'editor precedentemente conosciuto come FCKeditor. FCKeditor è nato nel 2003 a da allora è stato sviluppato con l'aggiunta di nuove funzionalità. Tra le caratteristiche che lo differenziano dagli altri c'è l'attenzione verso le tematiche sull'accessibilità. Si tratta di accessibilità perseguita applicando le indicazioni internazionali del W3C (World Wide Web Consortium) [25]. Il W3C è una comunità internazionale che ha lo scopo di sviluppare e definire gli standard del web al fine di massimizzarne le prestazioni. Esso definisce, ad esempio, la grammatica di linguaggi come l'html e i CSS. Nel 1997 il consorzio lanciò l'iniziativa WAI (Web Accessible Iniziative) [26] che ha come fine lo sviluppo di standard che consentano di rendere accessibili alle persona affette da disabilità i contenuti del web. Le WCAG (Web Content Accessibility Guidelines) [27], pubblicate nella versione 1.0 sul sito del W3C nel maggio del 1999, sono uno dei frutti dell'iniziativa. Si tratta delle linee guida (o criteri) da seguire per realizzare un'interfaccia web che sia accessibile anche a persone affette da cecità o altri difetti di vista, sordità, ulteriori limitazioni cognitive, limitazioni nella capacità di movimento, etc. Esse non sono scritte per una tecnologia specifica, come il linguaggio html, ma si tratta di criteri più generali. Il 25 gennaio 2001 è stata pubblicata sul sito del W3C la versione 2.0 delle WCAG[4]. Una pagina web può essere conforme alle WCAG 1.0, o 2.0 o entrambe. All'interno delle linee guida si distinguono tre livelli di conformità, indicati con le diciture A (livello basso), AA (livello medio), AAA (livello alto). A ogni criterio è associato un livello di conformità. Il livello di conformità di un'applicazione è stabilito in base a quali criteri questa implementa.
CKEditor è conforme alle WCAG 2.0 del livello doppia AA.

4 Alla data di scrittura della tesi l'ultimo aggiornamento delle WCAG 2.0 risale all'11 dicembre 2008.

Questo editor è anche conforme alle direttive della Section508 [28] che è una legge statunitense che descrive come il contenuto web debba essere accessibile anche agli utenti diversamente abili. La normativa italiana, citata nel capitolo 2.2, è fortemente ispirata dalle linee guida sull'accessibilità del W3C e dalla Section508, per questo il CKEditor risulta particolarmente adatto alle applicazioni web sviluppate in Italia.
CKeditor è compatibile con i più diffusi screenreaders, come ad esempio JAWS [29], è fruibile con il solo utilizzo della tastiera ed esiste anche una versione con una grafica caratterizzata da un alto contrasto tra i colori.
CKEditor è distribuito con le licenze GPL, LGPL [30], e MPL [31]. In più è disponibile con una licenza commerciale.

7.3.1.2 TinyMCE

E' uno degli editor più diffusi grazie anche all'integrazione con il CMS Joomla [32] che lo propone come editor di default.
Tra le sue peculiarità vi sono i "compressors" che minimizzano la banda necessaria al caricamento dell'editor. Il codice Javascript, di cui è costituito TinyMCE viene compresso dal server con GZIP [33] e inviato al browser il quale lo decomprime e lo esegue.
I "compressors" sono disponibili in diversi linguaggi lato server, come PHP, ASP.NET (Active Server Pages .NET), JSP (JavaServer Pages) e CFML (ColdFusion Markup Language).
Questo sistema è utile soprattutto se il rallentamento nel caricamento dell'editor è dovuto alla carenza di banda. Per contro richiede un po' di potenza di calcolo in più sia da parte del server che da parte del client per le operazioni rispettivamente di compressione e di decompressione.
Nell'applicazione "Guida ai Servizi" si prevede che l'interfaccia di back-end verrà fruita principalmente tramite la LAN dell'ente, di conseguenza la banda necessaria all'editor non è una criticità. Per questo tale peculiarità non è essenziale ai fini del progetto.
Sono disponibili a pagamento dei moduli aggiuntivi per la gestione di file e immagini.
Il nucleo principale dell'editor è distribuito con licenza LGPL.

7.3.1.3 Xhina

Tra le sue peculiarità vi sono alcune funzionalità di "tidy" ossia di pulizia del codice. L'applicazione del filtro "tidy" consente di eliminare automaticamente i

tag html indesiderati che spesso compaiono in seguito a operazioni di copia e incolla di testi scritti con programmi di office automation. Questo tipo di operazioni è molto ricorrente nel modo di operare degli utenti della "Guida ai Servizi". In più si possono correggere o eliminare, sempre in maniera automatica, quegli elementi di formattazione frutto delle abitudini estetiche/grafiche dei diversi attori del sistema.
Grazie a questa funzionalità si ottiene non solo un codice html con grammatica di tipo strict, che è il primo requisito della "Legge Stanca" sull'accessibilità, ma anche un buon livello di uniformità nella formattazione grafica delle schede.
Tra i plugins a disposizione gratuita vi è un un file manager, chiamato "Extended File Manager", molto versatile che, opportunamente configurato, consente di caricare nel server e collegare nelle pagine html generate, qualsiasi tipo di file.
Xhina è distribuito con licenza BSD con 3 clausole [34].

La scelta è caduta su Xinha per la presenza delle funzionalità di "tidy" e di file manager. Al momento della scelta dell'editor non era ancora stato affrontato il problema del caricamento dei file e l'"Extended File Manager" di Xhina rappresentava una possibile soluzione. Come spiegato in seguito il problema è stato poi risolto con un altro sistema più semplice da utilizzare da parte dei referenti e dei redattori.

7.3.2 Importazione dei files

Inizialmente per il caricamento dei files relativi ai moduli e agli allegati si è sperimentato il file manager dell'editor Xhina, che però è risultato di difficile integrazione e soprattutto di difficile utilizzo da parte degli utenti.
E' stata poi utilizzata una tecnica AJAX (Asynchronous JavaScript and XML) [35] che era stata adottata inizialmente solo per il caricamento delle icone associate alle categorie. Questo sistema permette all'utente di gestire l'upload dei files in un'unica schermata senza passare per il submit definitivo della pagina.
AJAX è una tecnica di sviluppo per la realizzazione di applicazioni web interattive (dette anche RIA, Rich Internet Application). Lo sviluppo di applicazioni html con AJAX si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza il ricaricamento della stessa. AJAX è asincrono nel senso che i dati richiesti al server vengono caricati in background e visualizzati in tempi diversi rispetto alla generazione della pagina principale.

Nella pagina dell'applicazione dedicata all'inserimento dei moduli, l'utente, oltre a inserire gli attributi necessari (titolo, ufficio di riferimento, etc.), deve selezionare il file relativo, scegliendolo tra quelli presenti nel file-system del proprio computer. Al momento della selezione il file viene caricato sul server e validato controllando che le dimensioni non siano eccessive e che l'estensione sia tra quelle consentite. Tutto questo prima di procedere al salvataggio di quanto inserito nella form. In questo modo l'utente può verificare subito se il file selezionato è quello giusto, infatti, nella pagina stessa, al termine del caricamento, compare un link che, se cliccato, fa visualizzare il contenuto del file. In caso di errore è sufficiente procedere a una nuova selezione, sempre senza effettuare il submit di tutta la form.
Normalmente il caricamento dei file sul server si realizza dopo il submit, e la validazione e l'eventuale possibilità di correzione si ha nella pagina successiva. Tuttavia costringere l'utente alla visione di una schermata di verifica, anche se i dati caricati sono corretti, appesantisce la procedura di inserimento.
Con l'ausilio della tecnica AJAX l'esito dell'operazione di selezione del file è immediatamente visibile e, se gli altri dati inseriti sono congruenti, non è necessario proporre un'ulteriore pagina di controllo.
Considerando che spesso gli utenti incontrano difficoltà nella navigazione del file-system, la percezione che si è avuta con questo metodo è stata di rassicurante semplificazione.
In particolare è stato adottato lo script "PHP AJAX Image Upload" di Tim Wickstrom che è distribuito con licenza Creative Commons Attribution 3.0 United States License [36] basato su lavoro di AT Web Results, Inc. [37]. Tra i numerosi script a disposizione questo è risultato particolarmente versatile ed è inoltre dotato di una pratica barra di caricamento.
È stata adottata la tecnica AJAX solo nel sistema di caricamento dei files perché, in questo caso, il vantaggio avuto dal miglioramento della funzionalità dell'interfaccia è stato valutato più importante della maggior complessità del codice generato.

7.3.3 Meccanismo di pubblicazione e versionamento

Nel capitolo 2.3.1 è stato descritto il flusso documentale che deve essere implementato.
Nel back end si è scelto di distinguere quattro stati: "in redazione", "richiesta pubblicazione", "pubblicata", "richiesta di spubblicazione".

Nello stato “in redazione” la scheda può essere modificata dal referente e ogni modifica non deve produrre effetti nel front-end. Si tratta di una fase di produzione della scheda. Quando un referente ritiene la scheda completa ne richiede la pubblicazione e la scheda passa nello stato di “richiesta pubblicazione”.
Nello stato “richiesta pubblicazione” la scheda viene controllata dal redattore che la corregge e la pubblica.
Nello stato “pubblicata” la scheda nel back-end è identica a quella in front-end. Un redattore la può modificare e il risultato diventa immediatamente visibile in front-end. Se invece un referente la volesse modificare, si ritornerebbe allo stato “in redazione” per ricominciare l'iter. Un referente potrebbe anche chiedere solo la spubblicazione, e la scheda passerebbe in “richiesta spubblicazione”. Nel front-end la scheda resterebbe comunque visibile finché un redattore la spubblica, il che la riporta nello stato “in redazione”.
Ad ogni passaggio di stato gli attori interessati, cioè tutti i redattori e i referenti dell'ufficio cui è assegnata la scheda in lavorazione, vengono avvisati tramite una mail.
L'elenco completo delle transizioni tra gli stati di back-end si può vedere nello schema di figura 7.5. Nella stessa figura sono stati evidenziati con tratti diversi le transizioni che hanno effetto sul front-end.
E' importante notare che questi stati sono relativi al solo back-end. Nel front-end infatti una scheda è caratterizzata solo dalla presenza o meno, ossia dagli stati “pubblicata” o “non pubblicata”.
Da questo è scaturita la necessità di separare la gestione del back-end da quella di front-end.
Si è scelto così di tenere le tabelle che custodiscono i dati del front-end separate dalle altre tabelle. Questo ha anche il vantaggio di consentire la separazione fisica dei database. Attualmente tutte le tabelle sono all'interno dello stesso database ma il tutto è predisposto per separare i due versanti dell'applicazione. In questo modo si potrebbe custodire in una DMZ la parte relativa al front-end e all'interno della LAN aziendale la parte relativa al back-end incrementando il livello di versatilità e di sicurezza dell'applicazione. La pubblicazione in front-end è in sintesi una sincronizzazione, dei soli contenuti da pubblicare, tra il database di back-end a quello di front-end.
Di fatto si generano così due versioni delle schede, quelle pubblicate (in front-end) e quelle in redazione (in back-end). Per consentire una rapida distinzione le

tabelle relative alle schede sono state nominate "schedabackend" e "schedafrontend".

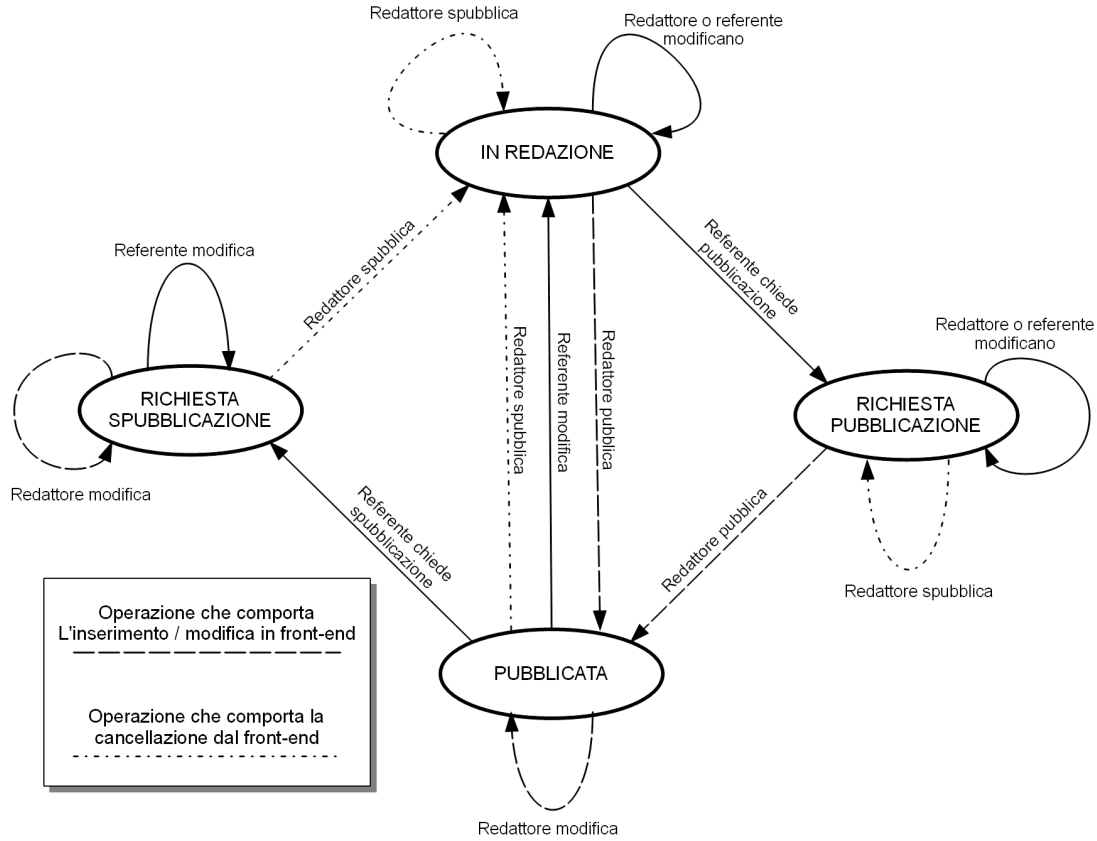

Figura 7.5: Diagramma degli stati e delle transizioni di back-end

7.4 Front-end

Il front-end è stato realizzato generando codice html caratterizzato da un layout fluido[5] con il quale si gestisce la disposizione grafica dei contenuti senza far uso del costrutto "table" che andrebbe riservato, come suggerito dalle linee guida, solo per la disposizione di dati tabellari.

La problematica più rilevante in questa fase ha riguardato la grafica, una delle esigenze è infatti quella di dare all'applicazione un aspetto simile a quello del sito principale dell'ente www.comune.treviso.it.

L'implementazione statica di un'interfaccia grafica non avrebbe permesso di seguire tempestivamente gli aggiornamenti che caratterizzano il sito. Ad esempio, le voci di menù, che fanno parte dell'intestazione, sono soggette a

5 Talvolta viene utilizzato anche il termine "layout liquido".

variazioni con una cadenza circa mensile. Con questa soluzione, i cambiamenti dell'intestazione avrebbero dovuto essere riportati nella “Guida ai Servizi” con la stessa cadenza.
Si è scelto dunque di implementare un meccanismo in cui l'applicazione ricava automaticamente il suo aspetto da una pagina web.
In particolare nel codice html del sito web dell'ente sono state inserite delle stringhe di demarcazione che servono al sistema per individuare i punti di taglio, ossia le porzioni di codice che costituiscono l'intestazione e il fondo pagina.
Le stringhe di demarcazione sono dei commenti html non visibili all'utente e per inserirle si è dovuto modificare il sito principale. Essendo quest'ultimo realizzato con il CMS Joomla, si è dovuto intervenire solo sul file che tale software utilizza come definizione del template grafico.
Il codice html che costituisce l'intestazione e il fondo pagina viene poi elaborato automaticamente per correggere il titolo, i metadati, i fogli di stile e i collegamenti espressi in modo relativo.
Questo meccanismo di “grab” provoca un rallentamento del sistema misurato in circa un secondo per ogni pagina visualizzata, poiché si somma il tempo dovuto alla visita del sito principale. Si è preferito allora fare in modo che questa operazione non avvenisse più frequentemente di un intervallo di tempo prefissato. Più precisamente, il template in uso è depositato nel database con la relativa data di generazione. A ogni visita l'applicazione verifica da quanto tempo è stato generato il template, se è da più di 30 minuti procede all'aggiornamento.
I parametri relativi a questo sistema, cioè l'URL della pagina da cui recuperare l'aspetto grafico e il numero di minuti dopo i quali considerare il template obsoleto, sono stati inseriti nel file di configurazione dell'applicazione.
Il risultato è visibile dal confronto delle figure 7.6 e 7.7.

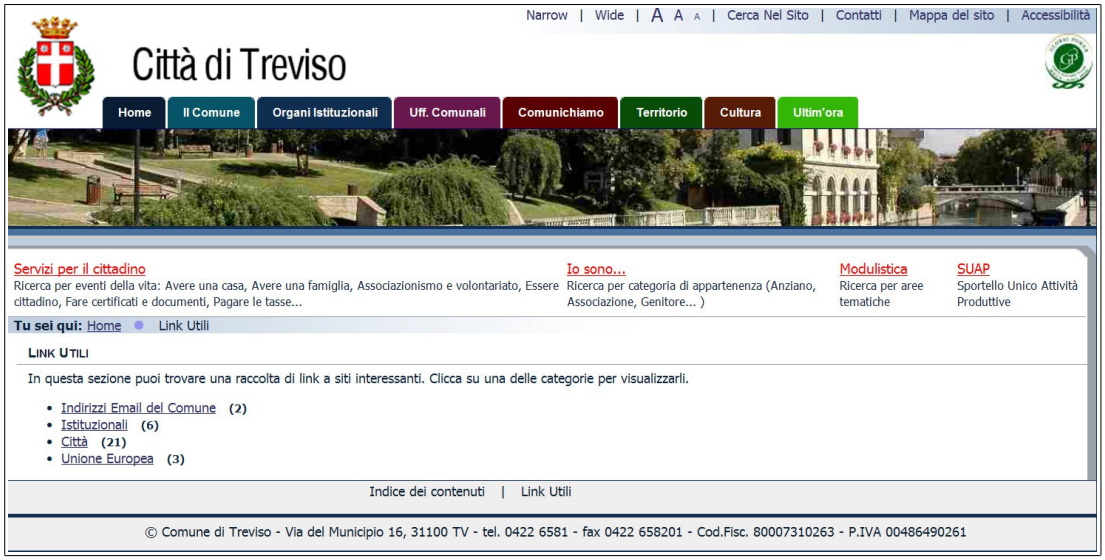

Figura 7.6: Schermata del sito web dell'ente

Figura 7.7: Schermata di front-end della "Guida ai Servizi"

Per rispondere a quanto richiesto nel capitolo 2.3.2 la pagina relativa a ogni scheda viene composta immettendo all'inizio le informazioni dell'ufficio di riferimento.

Come richiesto dalle linee guida alla fine di ogni scheda viene automaticamente riportata la data di ultimo aggiornamento delle informazioni.

Inoltre tra le categorie ce n'è una particolare che, posta al primo livello di profondità dell'albero, ospita tutti i moduli presenti nel sistema.

7.5 Utilizzo dell'applicazione

Rispetto alla versione precedente dell'applicazione si è verificato un aumento dei volumi gestiti. In ogni caso l'aumento non è tale da mettere in discussione le scelte tecniche effettuate o l'infrastruttura adottata che è in grado di supportare carichi molto maggiori.

7.5.1 Migrazione dei contenuti

Per popolare l'applicazione di contenuti si è scelto di partire da quelli presenti nella precedente versione dell'applicazione implementata su server Lotus Domino di IBM.

Utilizzando il linguaggio di script, proprietario di quella piattaforma, è stata realizzata una procedura che ha esportato quasi tutti i contenuti in linguaggio SQL. Il codice ottenuto è stato importato direttamente nel database.

È stata poi necessaria una fase di correzione manuale soprattutto per quel che riguarda l'inserimento delle "parole chiave" e l'associazione tra schede, moduli e allegati.

7.5.2 Quantità di dati inseriti

Allo stato attuale[6] l'applicazione ospita 321 schede informative (che occupano nel database 813,7 KB), 331 moduli e 219 allegati.

L'albero delle categorie è composto da 173 voci.

Gli utenti coinvolti sono 47 e gestiscono le informazioni relative a 105 uffici.

Il database ha una dimensione di 18,1 MB e conta 191.030 tuple (di cui 176.585 sono relative ai log).

Si può notare, dal confronto con la tabella 4.1, l'aumento dei volumi previsti. Il numero delle schede è aumentato del 61%[7], il numero dei moduli del 121%, il numero degli allegati del 338%, il numero delle categorie del 73%, il numero degli utenti del 42% e il numero degli uffici del 75%.

6 I dati si riferiscono al 1 giugno 2012.

7 Le percentuali sono arrotondate all'unità.

7.5.3 Statistiche di utilizzo del sistema

Mediamente in un mese[8] vengono visitate 9.262 schede informative, 5.241 moduli, 1.636 allegati e 12.736 categorie. Vengono effettuate 806 ricerche libere.

Si può notare, dal confronto con la tabella 4.2, anche l'aumento del numero delle operazioni previste. Il numero delle visite alle schede è aumentato del 23,5%, ai moduli del 74,7% e alle categorie del 41,5%.

8 La media è calcolata su un periodo che va dal 1 aprile 2012 al 1 giugno 2012.

8 Conclusioni

In questo lavoro, sono state esposte le procedure che hanno portato alla reingegnerizzazione di un software per la pubblicazione delle schede descrittive dei servizi erogati dal Comune di Treviso. La documentazione contenuta in questa tesi fornisce una panoramica completa del lavoro svolto e spiega le scelte tecniche effettuate.

Gli obiettivi proposti sono stati raggiunti, come si evidenzia dai dati sull'utilizzo del sistema. I contenuti gestiti dall'applicazione sono aumentati rispetto alla versione precedente. Tuttavia la mole non è tale da mettere in difficoltà l'architettura realizzata e in più si è ottenuto un sistema versatile e riusabile.

8.1 Altri utilizzi dell'applicazione

A conferma della versatilità e riusabilità dell'applicazione si può dire che questa è stata poi adottata anche per la realizzazione di un sistema di descrizione dei procedimenti interni riservato al personale dell'ente, e per la realizzazione di una guida dedicata allo sport praticabile nel territorio Comunale (InformaSport). Nei procedimenti interni si è assimilata la scheda di descrizione di un procedimento a una scheda di descrizione di un servizio, per cui il back-end è rimasto praticamente invariato mentre l'interfaccia di front-end è visibile solo all'interno della LAN. Allo stesso modo, per quel che riguarda l'InformaSport si è assimilato la scheda di descrizione dell'attività sportiva a una scheda di descrizione di un servizio.[9]

8.2 Criticità e prospettive di miglioramento

Nello sviluppo dell'applicazione ma soprattutto nella successiva fase di messa in funzione si sono evidenziate alcune criticità non risolte anche per l'esaurimento del tempo di lavoro assegnato al progetto. Tra le principali c'è la mancanza di indicizzazione del contenuto dei file relativi a moduli e allegati. Questo è un limite del motore di ricerca che propone agli utenti i moduli e gli allegati selezionandoli solo in base al titolo, al nome del file e alle parole chiave. Il risultato comunque positivo del funzionamento del motore di ricerca è dovuto anche alla competenza degli utenti che attribuiscono correttamente le parole chiave.

Un'altra criticità è rappresentata dal database non transazionale che costringe a delegare alla correttezza del codice PHP la gestione degli errori relativi alle

9 Al momento di scrittura della tesi questi sistemi sono in fase di realizzazione.

operazioni sui dati e la gestione della programmazione concorrente. Adottando un database transazionale, con l'ausilio delle chiavi esterne, si potrebbe ottenere un sistema ugualmente non suscettibile agli errori di inconsistenza dei dati ma caratterizzato da un codice di script più semplice.

Le prospettive di miglioramento e di sviluppo riguardano sia gli aggiornamenti dell'infrastruttura informatica che le richieste di nuove funzionalità che gli uffici coinvolti hanno avanzato dopo l'attivazione dell'applicazione.

Relativamente all'aggiornamento dell'infrastruttura si può notare che l'applicazione si comporta correttamente anche su piattaforme più recenti. Infatti il linguaggio PHP, di cui, al momento di scrittura della tesi, è stata rilasciata la versione 5.4, si è evoluto adottando numerose peculiarità orientate alla programmazione a oggetti. Quasi tutte le funzionalità della versione precedente di PHP, ossia quella utilizzata dall'applicazione oggetto della tesi, non sono state deprecate e sono presenti nella nuova versione, quindi, avendo implementato già una struttura a oggetti, in particolare per la parte relativa alla gestione del modello, e non avendo utilizzato funzioni deprecate, l'applicazione è compatibile anche con l'utilizzo di PHP 5.
Lo stesso vale per il database server MySQL e il server web Apache dei quali sono state utilizzate solo funzionalità di base difficilmente suscettibili a cambiamenti futuri.

Nella precedente realizzazione l'anagrafica degli utenti era integrata con il sistema di posta dell'ente. Questo consentiva di entrare nel programma senza immettere ulteriormente le credenziali di accesso. Anche se il passaggio per la schermata di login non è stato percepito come un grosso disagio potrebbe essere utile realizzare un sistema di SSO (Single Sign-On) in cui l'identità dell'utente è carpita in modo automatico. Per fare questo si potrebbe integrare il sistema con l'infrastruttura di dominio dell'ente ossia con l'anagrafica degli utenti abilitati ad accedere alle postazioni di lavoro computerizzate. Il sistema di dominio del Comune di Treviso è realizzato con Samba [38] che è un software open source.
Relativamente all'adozione di software di terze parti, quali ad esempio l'editor wysiwyg, essendo i prodotti disponibili in continua evoluzione, si può pensare di effettuare altre scelte a seconda delle nuove funzionalità disponibili, se utili allo scopo dell'applicazione.

E' stato chiesto di poter gestire dei collegamenti a siti esterni sottoponendoli alla categorizzazione dell'applicazione. Attualmente l'unico modo per inserire un link è quello di scriverlo nel contenuto di una scheda utilizzando l'editor wysiwyg. E' stato chiesto anche di aggiungere un sistema di customer satisfaction che permetta ai cittadini di valutare le informazioni presentate.
Ad una prima analisi di fattibilità queste nuove funzionalità sono compatibili con l'architettura generale dell'applicazione, confermando la bontà del sistema realizzato.

Ringraziamenti

Desidero ringraziare tutta la mia famiglia per il sostegno durante il percorso universitario e in particolare nonna Olinta.
Ringrazio il personale del Comune di Treviso che ha, in vario modo, collaborato alla realizzazione dell'applicazione oggetto della tesi.

Gianandrea

Bibliografia

[1] P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone, *Basi di Dati*. McGraw-Hill, Milano, 1999.

[2] C. Batini, B. Pernici, G. Santucci (a cura di), *Sistemi Informativi – Volume 5*. Franco Angeli, Milano, 2001

[3] *Gazzetta Ufficiale - Serie Generale n. 13 del 17-1-2004*. Istituto Poligrafico e Zecca dello Stato, 2004

[4] *Gazzetta Ufficiale - Serie Generale n. 112 del 16-5-2005.* Istituto Poligrafico e Zecca dello Stato, 2005

[5] G. Clemente, M. Moro, *Sistemi operativi – Edizione provvisoria.* Libreria Progetto, Padova, 1999

[6] M. Wandschneider, *Sviluppare applicazioni web con PHP e MySQL.* Apogeo, Trento, 2006

[7] B. Brinzarea, C. Darie, F. Cherecheş-Toşa, M. Bucica, *AJAX e PHP. Sviluppare applicazioni web dinamiche*. Hoepli, Trento, 2011

[8] N. C. Zakas, J. Mc Peak, J. Fawcett, *Ajax guida per lo sviluppatore.* Hoepli Trento, 2011

[9] A. Patton, *Practical LotusScript.* Manning Publications, 1999

Sitografia

[10] LAMP (software bundle). http://en.wikipedia.org/wiki/LAMP_(software_bundle)

[11] IBM Lotus Domino. http://www-142.ibm.com/software/products/it/it/domino

[12] PubbliAccesso.gov.it. http://www.pubbliaccesso.gov.it/

[13] Linee guida per i siti web della PA - Edizione 2011. http://www.funzionepubblica.gov.it/lazione-del-ministro/linee-guida-siti-web-pa/presentazione.aspx

[14] Server web. http://it.wikipedia.org/wiki/Server_web

[15] Apache HTTP Server Project. https://httpd.apache.org/

[16] GNU GPL. http://www.gnu.org/licenses/gpl.html

[17] MySQL. http://www.mysql.com/

[18] MyISAM. http://it.wikipedia.org/wiki/MyISAM

[19] PHP. http://www.php.net/

[20] PhpMyAdmin. http://www.phpmyadmin.net

[21] Smarty, PHP Template Engine. http://www.smarty.net/

[22] CKEditor. http://ckeditor.com/

[23] TinyMCE. http://www.tinymce.com/

[24] Xinha. http://trac.xinha.org/

[25] W3C. http://www.w3.org/

[26] WAI. http://www.w3.org/WAI/

[27] WCAG 2.0. http://www.w3.org/TR/WCAG/

[28] Section508. http://www.section508.gov/

[29] JAWS. http://www.freedomscientific.com/products/fs/jaws-product-page.asp

[30] LGPL. http://www.gnu.org/licenses/lgpl.html

[31] MPL. http://www.mozilla.org/MPL/1.1/

[32] Joomla. http://www.joomla.org/

[33] GZIP. http://www.gzip.org/

[34] BSD 3-Clause License. http://opensource.org/licenses/BSD-3-Clause

[35] AJAX. http://it.wikipedia.org/wiki/AJAX

[36] Creative Commons Attribution 3.0 United States License. http://creativecommons.org/licenses/by/3.0/us/

[37] AT Web Results, Inc. - PHP and AJAX Image Upload. http://atwebresults.com/php_ajax_image_upload/

[38] Samba. http://www.samba.org/