Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Sviluppo di un'interfaccia Web 2.0 per il Sistema Informativo Archivistico Regionale del Veneto

Giuliano De Boni

Relatore: Dr. Nicola Ferro

Correlatore: Ing. Gianmaria Silvello

Anno Accademico 2012/2013

Dedicato alla mia famiglia, a tutte le persone che mi hanno sostenuto in questi anni e anche un po' a me stesso, per tutti i sacrifici fatti in questi anni di studio.

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

Lo scopo di questa tesi è la reingegnerizzazione del SIAR, il Sistema Informativo Archivistico Regionale del Veneto, ed in particolare lo sviluppo di una nuova interfaccia grafica per il sistema. Per analizzare il contesto in cui si trova il SIAR è necessario introdurre i concetti di Digital Libraries e di archivio. Negli ultimi anni le Digital Libraries, in italiano biblioteche digitali, sono diventate un argomento di particolare interesse per la comunità scientifica e non solo. Il termine Digital Library è stato usato per descrivere una varietà di concetti ed entità e in letteratura ne sono state date numerose definizioni, che verranno spiegate in dettaglio. Un archivio, invece, è un complesso di documenti, dove come documento si vuole indicare un insieme di informazioni memorizzate su qualsiasi supporto o tipologia, prodotte o ricevute e conservate da un ente o da una persona nello svolgimento delle proprie attività o nella condotta dei propri affari. La parte riguardante le Digital Libraries viene completata introducendo alcuni modelli di Digital Libraries ed esempi di Digital Libraries, mentre per quanto concerne gli archivi, vengono descritti i principali standard tradizionali e gli standard informatici.

Il SIAR è un progetto supportato dalla Regione Veneto in collaborazione con l'Università di Padova. L'analisi di questo sistema inizia rivedendo la struttura e l'architettura del primo sistema SIAR, che si divide in tre livelli: infrastruttura di scambio dati, infrastruttura di gestione dei metadati ed interfaccia utente; i tre livelli vengono poi visti in dettaglio. Alla base del SIAR si trova il modello dati di NESTOR, infatti il SIAR è un'implementazione concreta dei modelli che costituiscono NESTOR. NESTOR definisce due modelli dati complementari basati su insiemi annidati: il Nested Set Model (NS-M) e l' Inverse Nested Set Model (INS-M).

Il lavoro di questa tesi si occupa di rivedere l'attuale interfaccia utente del SIAR, verso un nuovo sistema basato sulla piattaforma open source Liferay (portale Web) e sul paradigma delle Portlet. Inoltre, la nuova interfaccia viene progettata e sviluppata aderendo agli standard e alle convenzioni del Web 2.0. Prima di procedere alla descrizione delle componenti software e dell'ambiente di sviluppo è necessario introdurre una metodologia di analisi per poter effettuare un'analisi e una valutazione del nuovo SIAR, confrontandolo con gli altri sistemi informativi archivistici attualmente in uso in Italia, tra cui il SAN, Siusa, Archimista, Arianna, ecc., in modo da poter mettere in evidenza i vantaggi che derivano per il nuovo sistema. Allo scopo sono stati introdotti alcuni parametri di valutazione, che permettono di analizzare il sistema in base all'aderenza alla pratica archivistica e alle sue caratteristiche architetturali.

Successivamente viene descritto in dettaglio l'ambiente di lavoro utilizzato per la progettazione e lo sviluppo della nuova interfaccia Web del SIAR. Viene riportato in dettaglio il portale open source Liferay, già accennato in precedenza. Il modello di sviluppo utilizzato per la creazione dell'interfaccia utente del nuovo SIAR si basa sulla progettazione di componenti, le Portlet, lato client e, quindi, vengono introdotti brevemente i linguaggi utilizzati: HTML, i Css, Javascript, JQuery, JQuery UI, Alloy UI e la tecnologia AJAX, che permette di effettuare richieste asincrone ai Web Services.

Dopo aver descritto le tecnologie usate, viene spiegato il processo di reingnerizzazione del SIAR, introducendo le principali componenti che ne prendono parte, tra cui i concetti di portale Web, di Portlet e di REST Web Services. In seguito vengono analizzati i motivi che hanno portato a questa scelta e i vantaggi che ne derivano, tra cui il concetto di riutilizzo delle Portlet sviluppate

e la possibilità di personalizzare il portale Liferay, adeguandolo alle necessità che si presentano in fase di sviluppo ed in quelle successive. Infine viene effettuata un'analisi del nuovo SIAR, in base ai parametri di valutazione per sistemi informativi archivistici introdotti.

Infine, nell'ultima parte della tesi viene descritto il dettaglio del prototipo realizzato, che si basa sullo sviluppo di Portlet specifiche. Ogni Portlet sviluppata permette di gestire una delle risorse: produttori, consumatori, metadati (ad esempio concept e namespace, le due Portlet sviluppate in questa tesi). Ogni Portlet condivide una struttura di base, composta da tre componenti: **Nuovo contenuto/risorsa**, che permette di inserire all'interno del sistema SIAR un nuovo contenuto (che varia in base al tipo di Portlet), la **Gestione contenuti/risorse**, che permette di visualizzare e modificare le risorse già presenti nel sistema e la parte di **Ricerca contenuti/risorse**, che fornisce le funzionalità di ricerca e recupero delle risorse memorizzate e mantenute dal SIAR, con la possibilità di effettuare query personalizzate e selettive.

Abstract

The purpose of this thesis is the reengineering of the SIAR, the "Sistema Informativo Archivistico Regionale del Veneto", and, in particular, the development of a new graphic interface for the system. In order to analyze the context where the SIAR is, it's necessary to introduce the concepts of digital library and archive. In the last years digital libraries, in italian biblioteche digitali, are become an argument of particular interest for the scientistic community. The term digital libray has been used to describe many concepts and entities and in the literature it has been given many definitions, which will be explained in detail. An archive, instead, is composed by a complex group of documents, where a document is a set of informations stored in a support, which is produced and stored by public istitutions or by people, who are doing their activities or business. The part of digital libraries are completed introducing some digital libraries model and their implementations. The section that completes the archives describes the classical standard and the computer standards.

The SIAR is a project supported by the Regione Veneto in collaboration with the University of Padua. The analysis of this system starts by focusing of the architecture of the first SIAR, which are composed of three levels: exchange data infrastructure, metadata management and user interface; every level will be shown in detail. The most important part of the SIAR is the data model of NESTOR, because it is an implementation of models which constitute NESTOR. NESTOR define two data models, which are complementary, based of nested sets: the nested set model (NS-M) and the inverse nested set model (INS-M).

The work of this thesis is to review the current user interface of the SIAR, toward a new system based on the open source platform Liferay (Web portal) and on the paradigm of the Portlet. Furthermore, the new interface is designed and developed by adhering to the standards and conventions of Web 2.0. Furthermore, the new interface is designed and developed by adhering to the standards and conventions of Web 2.0. Before proceeding to the description of software components and development environment it's necessary to introduce a method of analysis in order to make an analysis and evaluation of the new SIAR, comparing it with other archival information systems currently in use in Italy, including the SAN, SIUSA, Archimista, Ariane, etc., in order to highlight the advantages of the new system. In order were introduced some evaluation parameters, which allow to analyze the system on the basis of adherence to archival practice and its architectural features.

Next, the work environment used for the design and development of the new web interface SIAR is described in detail. It's reported in detail the open source portal Liferay, already mentioned earlier. The development model used for creating the use interface of the new SIAR is based on the design of components, portlet and, therefore, are introduced briefly the languages used: HTML, CSS, Javascript, JQuery, JQuery UI, Alloy UI and AJAX technology, which allows to make asynchronous request to web services.

After describing the technologies used, the process of reengineering of the SIAR is explained, introducing the main components that take part, including the concepts of the Web portal, portlet and REST Web Services. Then the reasons that have led to this choice and the benefits derived therefrom are analyzed, including the concept of re-use of portlets developed and the ability to customize the Liferay portal, adapting it to the needs of specific settings in the

development phase and in subsequent. Finally, an analysis of the new SIAR is performed, based on the evaluation parameters for archival information systems introduced.

Finally, the last part of the thesis describes the detail of the prototype, which is based on the development of specific portlets. Each Portlet developed allows to manage a resource: producers, consumers, metadata (eg concept and namespace, the two portlets developed in this thesis). Each portlet shares a basic structure, consisting of three components: **new content/resource**, which allows to insert inside the system SIAR a new content (which varies according to the type of portlet), the management **content/resources**, which allows to view and modify the resources already present in the system and the research **content/resources**, which provides search capabilities and resource recovery stored and maintained by the SIAR, with the ability to perform custom queries and selective.

1 Introduzione

La tesi è stata suddivisa in cinque capitoli (oltre all'introduzione e alla conclusione), che verranno introdotti e descritti brevemente nelle sezioni successive.

1.1 Digital Libraries e archivi

In questo capitolo vengono introdotte le Digital Libraries e gli archivi, in quanto lo scopo di questa tesi è lo sviluppo di un'interfaccia web 2.0 per il Sistema Informativo Archivistico Regionale del Veneto.

Nella prima sezione vengono dati alcuni cenni sulle Digital Libraries e sugli Archivi. All'inizio viene fatta un'introduzione generale sulle Digital Libraries, con una breve overview basata sulla letteratura dell'argomento. Poi vengono spiegati i principali modelli di Digital Libraries: il DELOS Digital Library Refence Model e il 5S Model, ed alcuni esempi di Digital Libraries, sviluppate negli ultimi anni: Etana-DL, DSpace e Fedora DL, DelosDLMS ed Europeana. Dopo aver visto le Digital Libraries, vengono introdotti gli archivi, analizzandone le proprietà mettendo in risalto le differenze principali con le Digital Libraries. Successivamente vengono presentati i principali standard tradizionali e gli standard informatici per gli archivi.

Nella seconda sezione vengono introdotti alcuni parametri di confronto per i sistemi informativi archivistici. Dopo aver descritto tali parametri viene fatta una breve panoramica sullo stato dell'arte nella Pubblica Amministrazione, presentando i principali sistemi sviluppati ed attualmente in uso: il Sistema Archivistico Nazionale (SAN), Siusa (Soprintendenza per i Beni Culturali), Archimista (Regione Lombardia) e IBC Archivi (Regione Emilia Romagna).

Nella terza sezione invece vengono descritti in breve i principali software proprietari per la gestione degli archivi: Archivio Storico, Arianna e GEA 5.0.

Nell'ultima sezione viene presentata una tabella comparativa dei Sistemi Informativi Archivistici descritti nelle sezioni precedenti, sia quelli sviluppati nell'ambito della Pubblica Amministrazione che quelli sviluppati da aziende private.

1.2 SIAR: il Sistema Informativo Archivistico Regionale del Veneto

In questo capitolo viene introdotto il SIAR, ovvero il Sistema Informativo Archivistico Regionale del Veneto. Il SIAR è un progetto supportato dalla Regione Veneto in collaborazione con l'Università di Padova.

Nella prima sezione viene descritta la prima versione del sistema SIAR, focalizzandosi sulle varie fasi di progettazione e sviluppo del progetto: ideazione, analisi dei requisiti, progettazione, modello dei dati, sviluppo e verifiche.

Nella seconda sezione viene spiegata l'architettura del primo sistema SIAR, che si divide in tre livelli: infrastruttura di scambio dati, infrastruttura di gestione dei metadati ed interfaccia utente. I tre livelli vengono visti in dettaglio.

Nella terza sezione viene introdotto il modello NESTOR. Il NESTOR Model è alla base del SIAR; infatti il SIAR è un'implementazione concreta dei modelli che costituiscono il Nestor. NESTOR definisce due modelli dati complementari basati su insiemi annidati: il Nested Set Model (NS-M) e l' Inverse Nested Set Model (INS-M), che vengono descritti nella sezione.

1.3 Tecnologie utilizzate

In questo capitolo viene descritto e analizzato in dettaglio l'ambiente di lavoro utilizzato per la progettazione e lo sviluppo della nuova interfaccia utente del SIAR.

All'inizio vengono introdotti i concetti di portale Web e di REST Web Service, le due componenti fondamentali della nuova architettura del SIAR che verrà vista nel prossimo capitolo.

Nella seconda sezione viene presentato e descritto in dettaglio l'ambiente di lavoro. Inizialmente, viene introdotto Liferay Portal, illustrandone le principali caratteristiche ed i suoi punti di forza; successivamente vengono descritte le Portlet e la loro struttura.

Nella terza sezione viene inizialmente illustrato il procedimento seguito per l'installazione e la configurazione dell'ambiente di sviluppo usato per la creazione delle Portlet. Come già accennato, il modello di svilippo utilizzato per la creazione dell'interfaccia utente del nuovo SIAR si basa sullo sviluppo di componenti, le Portlet, lato client. Il motivo di questa scelta è dovuto al fatto che tutta la logica di business è implementata tramite dei Web Services di tipo REST che forniscono l'accesso e la manipolazione dei contenuti del SIAR e lo scopo di questa tesi è quello di creare la nuova interfaccia grafica per accedere e gestire tali contenuti. In questa sezione vengono introdotti brevemente i linguaggi utilizzati: HTML, i Css, Javascript, JQuery, JQuery UI, Alloy UI e la tecnologia AJAX, che permette di effettuare richieste asincrone ai Web Services.

1.4 Analisi e progettazione

Il lavoro di questa tesi si occupa di rivedere l'attuale interfaccia utente del SIAR, basata su pagine HTML, verso un nuovo sistema basato sulla piattaforma Open Source Liferay (portale Web) e sul paradigma delle Portlet, che verrano viste più in dettaglio in seguito nel capitolo successivo. Quindi nella prima sezione del capitolo viene spiegato il processo di reingnerizzazione del SIAR, introducendo le principali componenti che ne prendono parte, tra cui i concetti di portale Web, di Portlet e di REST Web Services. Infine, dopo aver spiegato le nuove componenti, viene effettuata un'analisi del nuovo SIAR, in base ai parametri di valutazione per sistemi informativi archivistici definiti nel capitolo relativo all'introduzione alle Digital Libraries e agli archivi (sezione 2.2.1), con lo scopo di valutare e confrontare il sistema con quelli analizzati nel capitolo precedente.

Nella seconda sezione vengono analizzati i motivi che hanno portato a questa scelta (illustrata nella sezione precedente) e i vantaggi che ne derivano, tra cui quelli relativi alla decisione di utilizzare un portale Web come Liferay, con una struttura delle pagine basata sulle Portlet (componenti Web indipendenti e riutilizzabili) e un'architettura di tipo REST per implementare i Web Services che accedono ai dati ed ai metadati del sistema informativo archivistico regionale del Veneto.

Nella sezione finale, invece, viene effettuata un'analisi del SIAR 2.0, tenendo conto dei parametri di confronto introdotti nel capitolo 2.

1.5 Descrizione prototipo realizzato

Nell'ultimo capitolo viene introdotta e descritta l'interfaccia Web del SIAR, realizzata attraverso lo sviluppo di alcune Portlet. Viene introdotta la struttura di base che viene condivisa da ogni Portlet e vengono illustrati i principali dettagli implementativi per giustificare le scelte fatte in precedenza. Ogni Portlet è basata su tre componenti principali: **Nuovo contenuto/risorsa**, che permette di inserire all'interno del sistema SIAR un nuovo contenuto (che varia in base al tipo di Portlet), la **Gestione contenuti/risorse**, che permette di visualizzare e modificare le risorse già presenti nel sistema e la parte di **Ricerca contenuti/risorse**, che fornisce le funzionalità di ricerca e recupero delle risorse memorizzate e mantenute dal SIAR, con la possibilità di effettuare query personalizzate e selettive. Nella sezione finale del capitolo viene descritta la procedura usata per la progettazione e lo sviluppo del tema usato nel portale.

2 Digital libraries e archivi

In questo capitolo vengono introdotte le Digital Libraries e gli archivi, che sono un caso particolare delle prime. Dopo un'introduzione teorica, costituita da una breve overview sulla letteratura corrente, vengono illustrati i principali modelli di Digital Libraries ed alcuni esempi reali, come Etana, Europeana ecc. Successivamente vengono introdotti gli archivi e ne vengono illustrate le principali caratteristiche, tra cui il vincolo archivistico ed il principio del rispetto dei fondi. Per completare il discorso sugli archivi, vengono illustrati i principali standard tradizionali (ISAD(G), ISAAR(CPF)) es informatici (EAD). Dopo aver spiegato il concetto di sistema informativo archivistico, viene fatta una breve panoramica sui principali sistemi informativi archivistici sviluppati ed utilizzati in Italia. La panoramica viene terminata con un confronto tra i sistemi stessi, in base ad alcuni parametri introdotti e spiegati (architettura del sistema, esportazione dei dati, aderenza agli standard archivistici).

2.1 Cenni su Digital Libraries e archivi

2.1.1 Introduzione alle Digital Libraries

Negli ultimi anni le Digital Libraries [Agosti, 2009], in italiano biblioteche digitali, sono diventate un argomento di particolare interesse per la comunità scientifica e non solo. Il termine "Digital Library" è stato usato per descrivere una notevole varietà di concetti ed entità e in letteratura ne sono state date numerose definizioni. Alcune sono state formulate nell'ambito di specifici progetti di realizzazione di Digital Libraries dagli stessi sviluppatori, altre sono legate al mondo della ricerca e riflettono le diverse interpretazioni della biblioteca digitale. Negli anni novanta la DLI (Digital Library Initiative, negli USA) introdusse le Electronic Libraries; la biblioteca elettronica venne definita come un servizio, un'architettura, un insieme di risorse elettroniche, un database (che raccoglieva qualsiasi tipo di informazione, testuale, grafica, ecc.) e una serie di applicazioni in grado di ricercare, recuperare ed utilizzare le informazioni disponibili in un formato digitale e quindi utilizzabili da sistemi informatici. La biblioteca elettronica definiva quindi una biblioteca automatizzata che usava ogni tipo di strumentazione elettronica necessaria al suo funzionamento: grossi calcolatori, PC, terminali. Il termine *"elettronico"* era stato introdotto proprio per fare riferimento all'attrezzatura usata per la lettura dei dati e non per la tipologia delle informazioni eleborate. In questo senso *"elettronico"* definisce documenti e servizi inaccessibili senza attrezzature adeguate. Inizialmente gli obiettivi della DLI erano modesti, rispetto agli standard attuali. La ricerca si era focalizzata in tre aree:

1. recupero di dati e metadati di tutti i tipi, tra cui informazioni di tipo testuale, immagini, tracce sonore ecc.;
2. progettazione e sviluppo di algoritmi e software avanzati (per la ricerca, il recupero, il filtraggio, la sintesi di grandi quantità di informazioni di ogni tipologia);
3. pregettazione e utilizzo di database distribuiti su tutto il territorio.

Successivamente anche Borgman [Borgman, 1996, 1999] riprese i concetti espressi dalla DLI, definendo le Digital Libraries come insiemi di risorse elettroniche e tecniche applicate alla

creazione, ricerca e utlilizzo di informazioni. In questo senso possono essere considerate come un'evoluzione di sistemi di memorizzazione e recupero di dati su reti distribuite. Le Digital Libraries vengono costruite, organizzate e conservate da una comunità di utenti e sono utilizzate per supportare e sopperire alla necessità di una comunità di reperire e usare informazioni. In una comunità possono esistere gruppi che interagiscono attraverso uno scambio di dati, informazioni e conoscenze. La definizione data in precedenza estende la portata delle Digital Libraries verso numerose discipline, per includere l'intero ciclo di creazione, ricerca, recupero e utilizzo di informazioni; provando ad analizzare le cose secondo un altro punto di vista, le Digital Libraries sono viste anche come database e quindi anche la ricerca in questo ambito si è spinta verso lo sviluppo di strutture di memorizzazione, di algoritmi di recupero di dati, filtraggio e ricerca performanti, in modo da supportare l'incremento esponenziale della quantità di dati eleborati e memorizzati.

Un'altra definizione, introdotta da Oppenheim e Smithson [Oppenheim and Smithson, 1999], si focalizza sullo sviluppo delle nuove tecnologie digitali. Secondo gli autori, una Digital Library è un servizio informativo, in cui tutte le risorse informative sono disponibili in formato digitale e le funzioni di acquisizione, archiviazione, preservazione, recupero e accesso sono realizzate attraverso l'uso di tecnologie digitali. Gli autori, tuttavia, usano anche il termine *"biblioteca ibrida"* coniato da Rusbridge [Rusbridge, 1998] per definire le attuali biblioteche che, nella transizione al digitale, continuano ad integrare i servizi tradizionali delle biblioteche con i nuovi servizi. Con il termine *"biblioteca ibrida"* Rusbridge vuole indicare la combinazione di tecnologie e supporti informativi diversi, per i servizi di una biblioteca in transizione. Secondo Rusbridge la biblioteca ibrida dovrebbe essere *"disegnata per mettere insieme tecnologie diverse nel contesto di una biblioteca reale e per cominciare a sperimentare sistemi integrati e servizi sia nell'ambiente elettronico che in quello a stampa"*. È importante osservare che in inglese hybrid non indica la compresenza di elementi diversi nella stessa realtà, ma invece la trasformazione e la crescita da una specifica realtà a un'altra, in cui anche gli elementi di continuità si trovano a essere completamente rinnovati; la giusta traduzione dovrebbe essere quella di biblioteca in transizione. Il termine *"biblioteca ibrida"* nel tempo è caduto in disuso a favore del più diffuso termine Digital Library; tuttavia molte delle attuali Digital Libraries sono essenzialmente delle biblioteche ibride.

Tra le definizioni elaborate in ambito bibliotecario la più rilevante, perché identifica il servizio della Digital Library e la più diffusa è quella della Digital Libraries Federation (DLF). La Digital Libraries Federation precisa: *"Le Digital Libraries sono organizzazioni che forniscono le risorse, compreso il personale specializzato, per selezionare, organizzare, dare l'accesso intellettuale, interpretare, distribuire, preservare l'integrità e assicurare la persistenza nel tempo delle collezioni digitali così che queste possano essere accessibili prontamente ed economicamente per una comunità definita o per un insieme di comunità*". La comunità bibliotecaria focalizza la propria attenzione ai servizi offerti al pubblico e vede la Digital Library come estensione e/o come aggiunta di nuovi servizi delle biblioteche pubbliche o istituzionali. Estensione dei servizi, dal punto di vista dei bibliotecari, significa migliorare i servizi esistenti e ampliare le risorse informative attuali. Aumentare i servizi significa invece avviare nuove funzionalità, servizi a supporto degli utenti e in alcuni casi aumentare anche il target di utenza istituzionale.

Arms [Arms, 2000] raccoglie un insieme di obiettivi per le Digital Libraries, che devono:

- venire realizzate con la speranza che possano migliorare l'accesso all'informazione rispetto al passato.
- portare le informazioni direttamente all'utente. L'informazione diventa accessibile da ogni luogo;
- consentire una ricerca avanzata e la manipolazione dell'informazione digitale;

- condividere l'informazione. Molte aziende e istituzioni private e pubbliche usano Internet e le Digital Libraries come infrastruttura per la condivisione dell'informazione, ad esempio per la preparazione cooperativa di documenti e il loro ri-uso;
- consentire un veloce accesso all'informazione che è sempre aggiornata;
- permettere una migliore collaborazione tra gruppi e istituzioni univerisitarie e di ricerca.

Sostanzialmente sono tre le componenti fondamentali di una Digital Library:

1. collezione: è formata da dati testuali, metadata, documenti video e sonori e può comprendere sia una collezione permanente, sia una collezione cui si accede in determinati tempi;
2. servizi di accesso: devono dare la possibilità di recuperare in maniera veloce e semplice i dati che si vogliono ricavare e di estendere la ricerca anche a documenti collegati tra loro. I sistemi di accesso comprendono l'interfaccia utente, i sistemi di ricerca, identificazione e i sistemi di navigazione e di connessione all'informazione desiderata;
3. utenti: le esigenze sia di singole persone sia di utenti appartenenti ad istituzioni o community sono diventate sempre più importanti. Le applicazioni che si interfacciano con le Digital Libraries devono essere intuitive e performanti.

Negli ultimi anni, si sono diffuse maggiormente le Digital Libraries che raccolgono libri, film, documentari, musica ecc.. I nuovi tipi di sistemi devono essere in grado di manipolare collezioni formate da elementi di diversa natura e sono chiamati Digital Library Systems. Questi sistemi sono costituiti da complessi componenti software in grado di elaborare raccolte complesse e diversificate di oggetti digitali. Inoltre, lo sviluppo di Digital Library system si è orientato verso soluzioni utilizzabili anche dai non addetti ai lavori, mentre inizialmente le Digital Library erano pensate per comunità specializzate (centri di ricerca, università, enti pubblici ecc.). In questo contesto molte raccolte di dati stanno diventando fruibili via Web. Anche nel caso in cui le banche dati siano possedute dalle biblioteche o da altri enti, sia pubblici sia privati, e memorizzate su altri supporti di memorizzazione (cd, dvd ecc.), è ormai prevalente la tendenza a renderne la consultazione per gli utenti, sia locali che remoti, il più integrata e omogenea possibile, attraverso apposite applicazioni Web. L'utilizzo di applicazioni Web ha sicuramente il vantaggio di rendere accessibile le Digital Libraries potenzialmente in tutto il mondo (dove è disponibile una connessione ad Internet); un altro vantaggio è quello di riutilizzare tecnologie già consolidate per presentare all'utente un interfaccia anche graficamente gradevole, oltre che intuitiva.

2.1.2 Modelli di Digital Libraries

Come già accennato all'inizio, il termine Digital Library è stato coniato agli inizi degli anni Novanta per indicare una nuova tipologia di sistema informativo. Tuttavia, nel corso degli anni, tali tipologie di sistemi sono state sviluppate in ambienti diversi, adottando una grande varietà di approcci e soluzioni. Il risultato di questo processo ha portato alla creazione di sistemi anche molto diversi tra loro. L'importanza di introdurre dei modelli teorici risiede nel fatto di poter ottenere dei sistemi con caratteristiche ben definite, che possono risultare anche compatibili.

Il Digital Library Manifesto e il DELOS Digital Library Refence Model

Il Digital Library Manifesto [Agosti et al., 2007, Candela et al., 2009] si basa sulle conoscenze sviluppate da gruppi di ricerca europei sulle tematiche delle Digital Libraries, inclusi i gruppi che hanno fatto parte della Rete DELOS, e sui risultati degli eventi DELOS dedicati a tali tematiche. La prima dichiarazione che gli autori del Manifesto scrivono, per poter discriminare

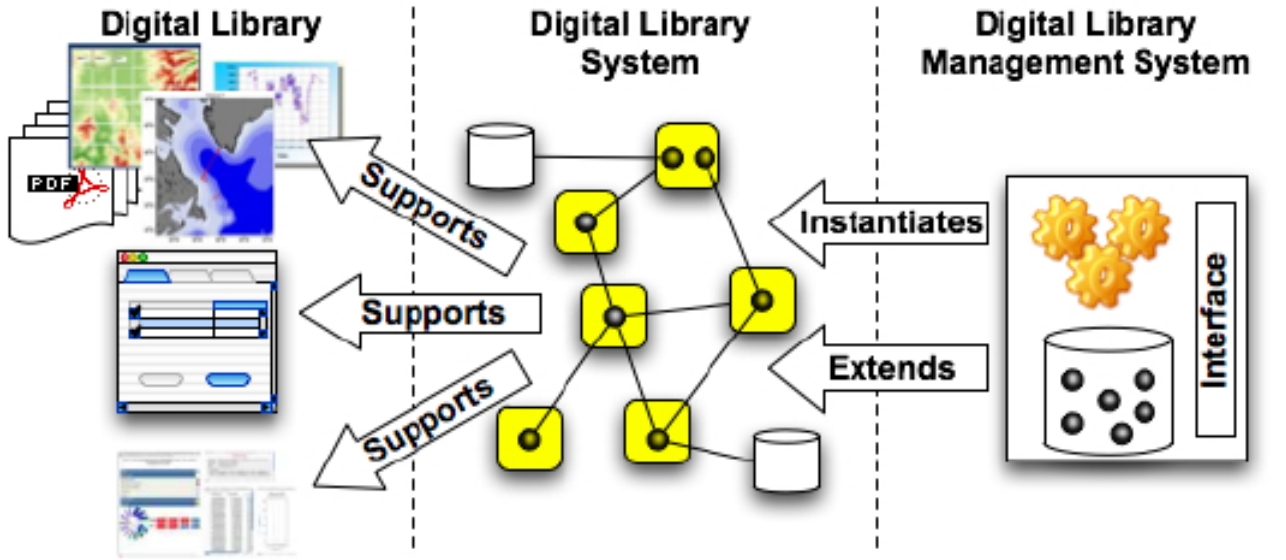
tra le diverse tipologie di sistemi per Digital Libraries, consiste nel distinguere tre componenti fondamentali:

- Digital Library: è un'organizzazione (eventualmente virtuale) che in modo organico raccoglie, gestisce e preserva per il lungo periodo un ricco contenuto digitale (Content) ed offre alle sue comunità di utenti (User) le funzionalità (Functionality) specializzate a trattare tale contenuto con livelli di qualità (Quality) misurabile e secondo regole (Policy) definite;

- Digital Library System: è un sistema software, basato su un'architettura (Architecture) ben definita (eventualmente distribuita), che fornisce tutte le funzionalità richieste da una specifica Digital Library. Gli utenti interagiscono con la Digital Library attraverso il corrispondente DLS;

- Digital Library Management System: è un altro software che fornisce l'infrastruttura software adatta a produrre e amministrare un Digital Library System, fornito di tutte le funzionalità considerate fondamentali per una qualsiasi Digital Library.

Figura 2.1: DELOS Reference Model [Agosti et al., 2007]



Il Reference Model è una struttura concettuale utile a rappresentare le entità significative di un certo universo di interesse, insieme alle relazioni tra le stesse. Le entità inizialmente introdotte nel Manifesto sono state organizzate in una gerarchia di domini, cioè gruppi di concetti e relazioni, ognuno identificato da un nome, che modellano specifici aspetti dei sistemi di tale mondo. In tal modo ciascuno dei tre sistemi in cui si articola il mondo delle Digital Libraries può essere modellato attraverso le entità e le relazioni catturate da questi domini a diversi livelli di astrazione. Complessivamente il modello consiste di 218 concetti e 52 relazioni. Il dominio più alto nella gerarchia è Digital Library Domain, che concettualmente racchiude tutti gli elementi necessari a rappresentare i tre sistemi per Digital Libraries. Questo dominio è diviso in due classi principali: Resource Domain e Complementary Domain. Il Resource Domain contiene tutti gli elementi fondamentali del mondo delle Digital Libraries ed è suddiviso nei domini: Content Domain, User Domain, Functionality Domain, Policy Domain, Quality Domain e Architecture Domain. Il Complementary Domain contiene tutti gli altri domini che, benché non costituiscano il centro di interesse delle Digital Libraries, sono tuttavia necessari per rappresentarne i sistemi.

5S Model

Il 5S Model [Goncalves, 2004] si basa su una serie di cinque concetti astratti o formalismi, posti come i fondamenti per la definizione e progettazione di una Digital Library; in questo modello gli oggetti digitali di una Digital Library, come metadati, collezioni, servizi, ecc., possono essere descritti in maniera rigorosa attraverso composizioni di oggetti matematici. 5SL [Goncalves, 2004], invece, è un linguaggio XML [W3C][1] che implementa il modello 5S; esso consente di definire le specifiche di alto livello delle componenti del modello, in quanto definisce:

- quali sono i dati e i metadati supportati (Stream Model);
- come vengono strutturate e organizzate tali informazioni (Structures Model);
- le proprietà logiche e le operazioni dei componenti di una Digital Libraries (Spaces Model);
- il comportamento e le caratteristiche delle Digital Libraries (Scenarios Model);
- le varie società degli Actors e degli User/Manager che operano nel contesto (Societies Model).

La tabella 2.1 riussume le componenti del modello 5S, sottolineando i formalismi e gli obiettivi di ciascun modello.

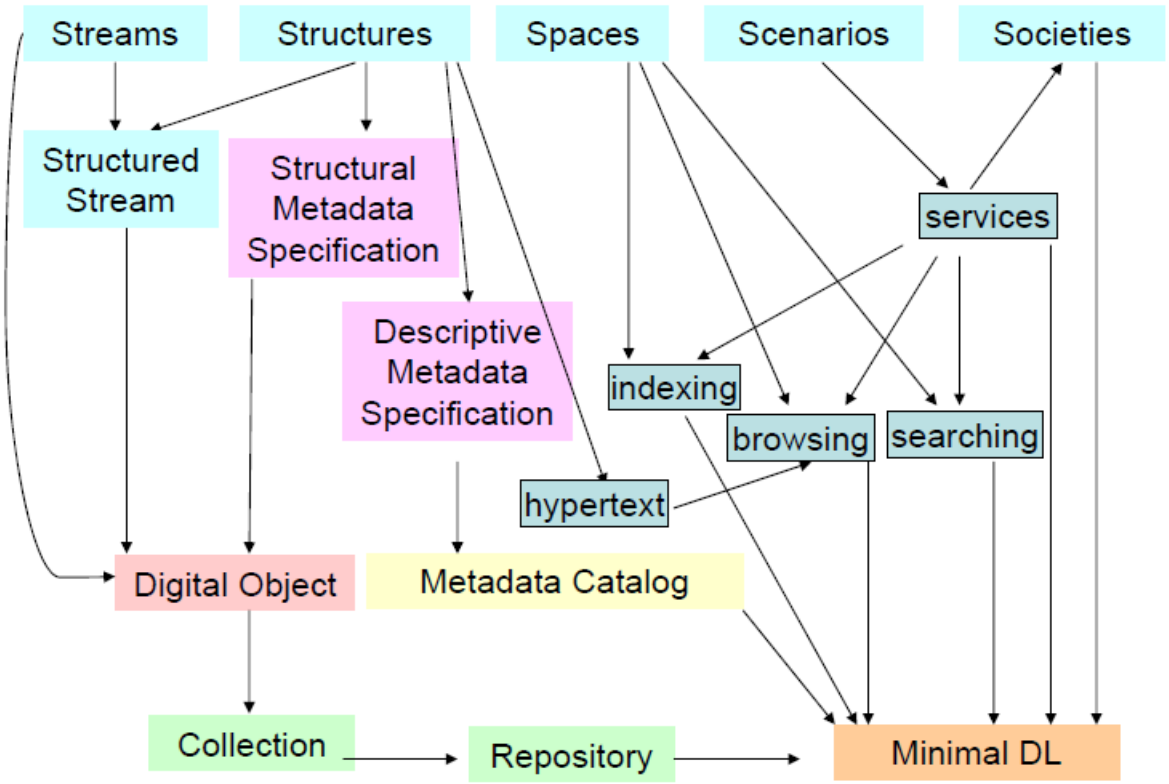
Tabella 2.1: Caratteristiche modello 5S

Modello	**Primitive**	**Formalismi**	**Obiettivi**
Stream Model	Testo, video, audio, Software	Sequenze, tipi	Descrive le proprietà dei contenuti
Structural Model	Collezioni, cataloghi, Hypertext, documenti, metadata	Grafici, con nodi, link, etichette	Specifica come sono organizzati i contenuti
Spatial Model	Interfaccia utente, modelli di ricerca	Insiemi, operazioni, spazi vettoriali ecc.	Definisce come vengono visualizzate le informazioni
Scenarios Model	Servizi, eventi, azioni	Sequenze, diagrammi, schemi	Dettaglio del comportamento del sistema
Societies Model	Community, utenti, manager, classi, relazioni, attributi	Modelli ad oggetti, pattern di progetto	Definisce i manager responsabile dei servizi e gli utenti che usano tali servizi

Invece, nella figura 2.2 viene riportato un diagramma [Fox, 2006] che rappresenta sinteticamente le cinque componenti e le relazioni tra loro:

[1] http://www.w3.org/XML/

Figura 2.2: Diagramma 5S Model [Fox, 2006]



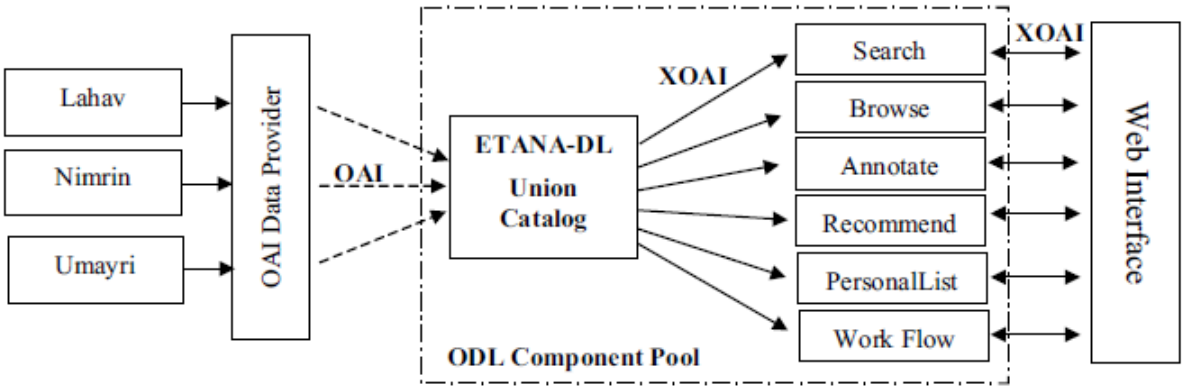
2.1.3 Esempi di Digital Libraries

Di seguito vengono introdotti brevemente alcuni esempi di Digital Libraries, sviluppate negli ultimi anni.

Etana-DL

Etana-DL [Ravindranathan et al., 2004] (Electronic Tools for Ancient Near Eastern Archaeology Digital Library) è un'implementazione di Digital Library basata sul 5S Model.

Figura 2.3: Architettura di Etana-DL [Ravindranathan et al., 2004]



Questa Digital Library è in grado di gestire informazioni complesse usando il paradigma Client-Server attraverso il protocollo OAI-PMH (Open Archive Initiative Protocol for Metadata Harvesting). È utilizzato per raccogliere (o collezionare) i metadati dei documenti contenuti in un

archivio in modo che altri servizi possano essere costruiti utilizzando metadati da più archivi. Il protocollo OAI-PMH verrà descritto dettagliatamente in seguito.

Lo schema visibile sopra nella figura 2.4, riporta com'era inizialmente l'architettura di Etana-DL. Sono presenti tre Data Provider e un catalogo centrale (Union Catalog) che si occupa di recuperare e mantenere i documenti digitali. Inoltre, sono presenti dei servizi DL utilizzabili per effettuare operazioni sulla Digital Library, quali la ricerca e la visualizzazione dei documenti. I servizi comunicano tra loro usando il protocollo XOAI, basato su XML.

DSpace e Fedora DL

I primi tipi di DLMS sono stati i sistemi di gestione di repository; in generale, si tratta di software con una gamma limitata di funzionalità. L'architettura è centralizzata. Un rappresentante di questa tipologia di sistemi è DSpace [Smith et al., 2003], sviluppato dalla biblioteca del MIT (Massachusetts Institute of Technology) e dai laboratori HewlettPackard a partire dal 2000. Il sistema, concepito per la gestione di repository istituzionali, aveva fra i suoi obiettivi:

- l'inserimento dei documenti digitali attraverso una procedura interattiva oppure utilizzando altri software di caricamento dati, compatibili con il sistema;
- la distribuzione del patrimonio digitale sul Web;
- la presenza di funzionalità di ricerca e di recupero delle informazioni e dei dati di interesse;
- la memorizzazione e conservazione del patrimonio digitale.

L'organizzazione delle informazioni all'interno di DSpace riflette la struttura di un'organizzazione di ricerca. Il repository è organizzato in comunità (ciascuna corrispondente a un laboratorio, centro o dipartimento), che mantengono le collezioni di documenti digitali. I vantaggi di DSpace sono l'efficienza e la semplicità delle sue procedure per l'istallazione, la configurazione e l'uso. Lo svantaggio principale è la sua limitata flessibilità, che lo rende adatto solo a specifici contesti.

La flessibilità è stata invece il maggior obiettivo perseguito dai progettisti del sistema Fedora (Flexible Extensible Digital Object Repository Architecture) [Lagoze and Payette, 1998], sviluppato circa nello stesso periodo in cui veniva introdotto DSpace. Fedora è stato inizialmente progettato dal Digital Library Research Group presso la Cornell University con un finanziamento della National Science Foundation; successivamente il suo sviluppo è proseguito in collaborazione con la biblioteca della University of Virginia Library. A differenza di altri sistemi di gestione di repository, utilizzabili tramite un'interfaccia utente-sistema non modificabile, Fedora è stato concepito come sistema-base su cui costruire altri sistemi e applicazioni per gli utenti, cioè come servizio fruibile per costruire applicazioni più sofisticate. Allo scopo di perseguire questo obiettivo, Fedora è stato implementato tramite dei Web Services.

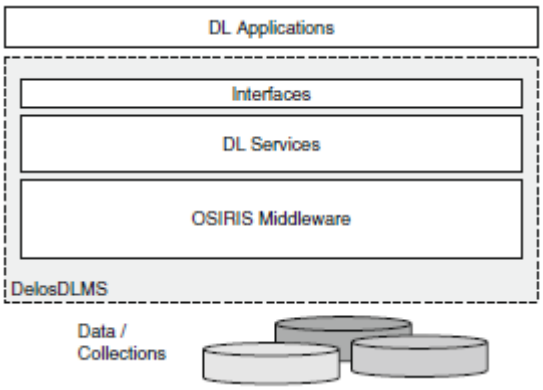
DelosDLMS

Si tratta di un prototipo [Ioannidis et al., 2008] di nuova generazione di Digital Libraries, costituito da un'infrastruttura DL distribuita, un insieme di serivizi di sistema base e una serie di servizi DL specifici e più sosfisticati. L'architettura del sistema è organizzata su due livelli logici:

1. un middleware che utilizza il paradigma SOA (Service-Oriented Architecture), un'architettura software adatta a supportare l'uso di servizi Web per garantire l'interoperabilità tra diversi sistemi;
2. una libreria estendibile di servizi di DL e interfacce che possono essere utilizzate per costruire applicazioni DL.

Il prototipo è basato sulla piattaforma OSIRIS/ISIS, un ambiente middleware (livello logico 1, di cui sopra) sviluppato dal Politecnico di Zurigo e poi esteso dall'Università di Basilea. OSIRIS è un middleware generico per l'esecuzione di processi distribuiti e decentralizzati, con la caratteristica di essere fault tolerant. ISIS, invece, è un prototipo di applicazione per il recupero delle informazioni all'interno di collezioni di dati digitali.

Figura 2.4: architettura di DelosDLMS [Ioannidis et al., 2008]



Come si può notare dallo schema di figura 2.4, al di sopra di OSIRIS è presente un livello costituito da servizi DL. Questi servizi sono di vario tipo: servizi basilari e generici, che forniscono il supporto per l'archiviazione, l'indicizzazione e il recupero di contenuti digitali, e servizi più specifici e complessi.

Europeana

Europeana è una Digital Libraries per cercare risorse in tutta Europa; tra le risorse disponibili e consultabili (copyright permettendo) si trovano: libri, riviste, film, mappe, foto, musica ecc. Fin dal momento del lancio del prototipo, la biblioteca è stata costantemente aggiornata e nel 2012 erano accessibili oltre 23 milioni di oggetti digitali.

Europeana fornisce delle API che permettono ad istituzioni, università ed altre organizzazioni culturali di:

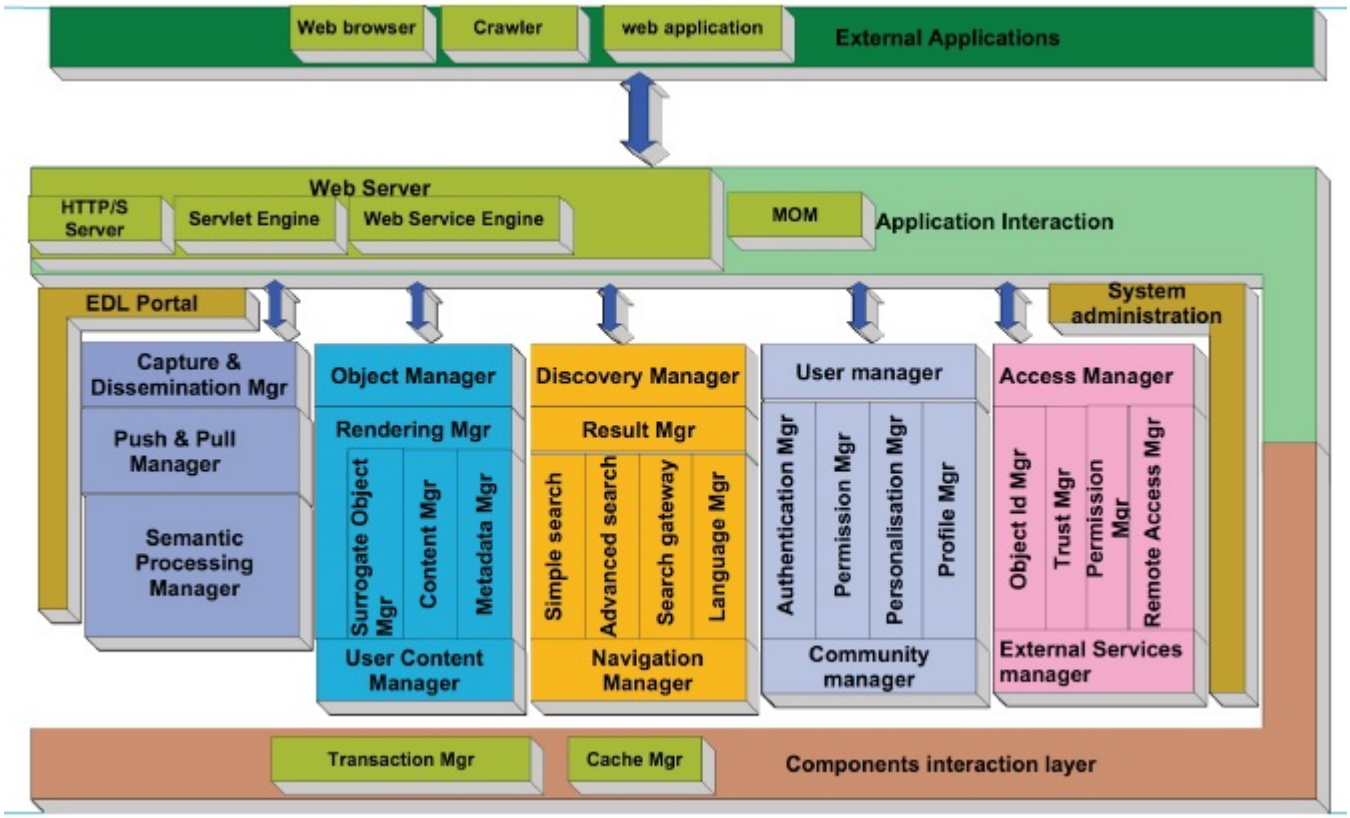
- accedere ai contenuti di Europeana stessa;
- aggiungere nuovi contenuti alla biblioteca;
- sviluppare e costruire applicazioni usando le funzionalità di Europeana.

Europeana non è un portale Web, invece Europeana Portal è una Web Application che utilizza le API per accedere ai contenuti della Europeana Digital Library. Nella figura 2.5 viene mostrata la struttura dell'architettura di questo sistema.

Il modello EDM (Europeana Data Model) è alla base dell'architettura di Europeana. Durante lo sviluppo di Europeana si sono presentati alcuni limiti. La prima difficoltà tecnica da superare è stata lo sviluppo di un insieme condiviso di metadati, che potesse mettere ordine fra tutti i differenti formati e standard usati dalle varie istituzioni coinvolte: ESE (European Semantic Elements). Il modello ESE non ha risolto tutte le problematiche, in quanto, ad esempio, la maggior parte dei metadati è in formato testuale e non URI, ponendo così dei limiti alla capacità di interconnessione tra i dati. Per superare questi problemi è stato allora progettato un nuovo modello, stavolta basato su RDF, che prende il nome di EDM (European Data Model). RDF[2] (Resource Description Framework) è lo strumento proposto dal W3C per descrivere i metadati

[2]http://www.w3.org/RDF/

Figura 2.5: Architettura di Europeana [*www.europeana.eu*]



relativi ad una risorsa, mettendo a disposizione un linguaggio per esprimere la semantica di una risorsa. EDM permette di risolvere molti dei problemi derivanti dall'uso di ESE, poichè:

- consente di distinguere tra un oggetto e la sua rappresentazione digitale;
- permette di differenziare tra un oggetto e i metadati che lo descrivono;
- prevede il supporto per risorse contestuali, in modo da poter connettere direttamente questo ultime all'oggetto di interesse.

2.1.4 Introduzione agli Archivi

Gli archivi esistono da secoli ma le definizioni concettuali di archivio sono più recenti. Una possibile definizione di archivio è quella di "*complesso di documenti prodotti o comunque acquisiti durante lo svolgimento della propria attività da magistrature, organi e uffici dello stato, enti pubblici e istituzioni private formate da famiglie e persone*" [Freda, 2008]. Ciò significa che l'archivio è il supporto e il risultato ottenuto da qualsiasi attività, sia individuale che istituzionale, sia semplice che complessa, posta in essere da quello che è il suo soggetto produttore. Gli archivi non nascono solo come beni culturali ma anche come strumenti di operatività quotidiana e si contraddistinguono per la loro trasversalità, che ne fa al tempo stesso beni culturali e strumenti di efficienza giuridica, amministrativa e operativa. Gli archivi in senso ampio possono essere considerati serbatoi di memoria,; la memoria, però, non ha necessariamente una profondità cronologica e allora gli archivi conservano anche la memoria del presente: certificati, titoli di studio, carte di identità, passaporti, contratti sono solo alcuni esempi di documenti utilizzati nella vita quotidiana. Negli archivi convivono dunque valori di natura giuridica, politica ed economica e valori di natura storica o culturale.

Rispetto a quanto introdotto da [Freda, 2008], altri studiosi hanno dato diverse definizioni di archivio: per Eugenio Casanova [Casanova, 1929] l'archivio è "la raccolta ordinata degli atti di un ente o individuo che si è costituita durante lo svolgimento della sua attività ed è stata conservata per il conseguimento di scopi politici, giuridici e culturali di quell'ente o individuo". La parola atti è sinonimo di documenti e in questo significato è usata anche nel linguaggio della Pubblica Amministrazione. Casanova sottolinea il fatto che un archivio, per essere tale, deve essere ordinato e, tra gli usi dello stesso, individua la finalità dell'archivio negli scopi giuridici politici e culturali. Giorgio Cencetti [Cencetti, 1937] definisce l'archivio come "complesso degli atti spediti e ricevuti da un ente o individuo per il conseguimento dei propri fini oppure per l'esercizio delle proprie funzioni" e sostanzialmente questa definizione concorda con quella di [Casanova, 1929]. Infine, Brenneke [Brenneke, 1968] considera l'archivio come "la totalità di scritti e altri documenti che si sono formati presso persone fisiche o giuridiche in base alla loro attività e quali fonti documentarie sono destinate a conservazione permanente in un determinato luogo". Per Brenneke il punto centrale sta nella produzione e nella conservazione degli atti archivistici, mentre Casanova e Cencetti si focalizzano sugli scopi e sulle motivazioni che portano alla costituzione dell'archivio.

Proprietà degli archivi

E' importante capire la distinzione fra archivio e biblioteca perché non è sempre del tutto chiara. Come è già stato accennato, la biblioteca è una collezione di libri, ordinata secondo un sistema, estraneo ai libri stessi e dipendente da un criterio (o più criteri) che può essere o meno ben definito. Il libro, cioè il singolo componente della biblioteca, ha due caratteristiche fondamentali (non condivise dagli elementi di un archivio):

- pluralità, poichè i libri sono solitamente delle copie;
- autonomia, senza legami necessari con altri libri.

Invece, l'archivio è un complesso di documenti [Bonfiglio-Dosio, 2003], dove, come documento, si vuole indicare un insieme di informazioni memorizzate su qualsiasi supporto o tipologia, prodotte o ricevute e conservate da un ente o da una persona nello svolgimento delle proprie attività o nella condotta dei propri affari; i documenti di un archivio hanno, a differenza dei libri di una biblioteca, la proprietà di unicità.

E' importante tenere conto del fatto che il concetto di documento non è vincolato dal legame con un particolare supporto di memorizzazione: possono essere documenti archivistici anche i file audio e video, le registrazioni ottiche o magnetiche e ogni altra forma di memorizzazione di informazioni su un qualsiasi tipo di supporto. Secondo la Legge 241/90, art.22: "è considerato documento amministrativo ogni rappresentazione grafica, fotocinematografica, elettromagnetica o di qualunque altra specie del contenuto di atti anche interni formati dalla pubblica amministrazione o comunque utilizzati ai fini dell'attività amministrativa".

Non tutte le forme di scrittura o di registrazione, prodotte e trasmesse all'interno delle organizzazioni, costituiscono documenti archivistici; lo sono quelle legate da un vincolo archivistico e che vengono organizzate e memorizzate nell'archivio da un soggetto produttore. Il concetto di vincolo archivistico è molto importante, infatti consente di distinguere tra una semplice raccolta di documenti ed un archivio. Il vincolo archivistico [Valacchi, 2009] è il nesso che collega in maniera logica e necessaria la documentazione costruita dal soggetto produttore. Tale vincolo esiste anche in assenza di attività esterne (numerazioni, classificazioni ecc.) sull'archivio e si manifesta in maniera indipendente all'interno dell'archivio. A titolo di esempio [Valacchi, 2009], è possibile vedere il modo in cui viene creato un vincolo archivistico, come risultato di un procedimento amministrativo. Prendendo il caso di una concessione edilizia, la pratica viene aperta con una richiesta di concessione, cui farà seguito la risposta dell'ente (Comune,

Provincia, ecc.) che chiede documenti a sostegno della richiesta. A seguito della comunicazione dell'ente, l'interessato alla concessione produrrà a sua volta i documenti richiesti e così via fino a quando viene rilasciato il permesso. L'esempio descritto risulta molto semplificato, ma dà la possibilità di vedere come il procedimento generi una serie di documenti legati appunto da un vincolo di involontarietà (la collocazione del documento è determinata dal flusso e non da scelte individuali), che lega ogni documento con altri in un contesto più ampio.

Il vincolo archivistico permette quindi di caratterizzare l'archivio, differenziandolo da una raccolta o da una collezione di documenti. Partendo da questo concetto, è possibile individuare come conseguenza diretta il fatto che un archivio, per definirsi tale, deve ricostituire l'ordine originario dei documenti, l'ordine che i documenti avevano al momento della loro nascita e che rispecchia il modo di essere e di funzionare dell'ente che li ha prodotti. Questo concetto si chiama "*respect des fonds*"[3] (rispetto dei fondi). Analizzando in maniera letterale la definizione appena data, è possibile individuare il concetto espresso: i documenti provenienti da fondi diversi non devono essere mescolati fra loro. In un contesto più ampio, invece, *respect des fonds* significa non solo il rispetto dei fondi, ma anche, all'interno di ciascuno di essi, la ricostituzione dell'ordine originario dei documenti. Con il concetto di *provenance* si intende, invece, la relazione fra i documenti archivistici e l'ente o la persona che li ha posti in essere e/o accumulati e usati nello svolgimento della propria attività personale o istituzionale, come già accennato.

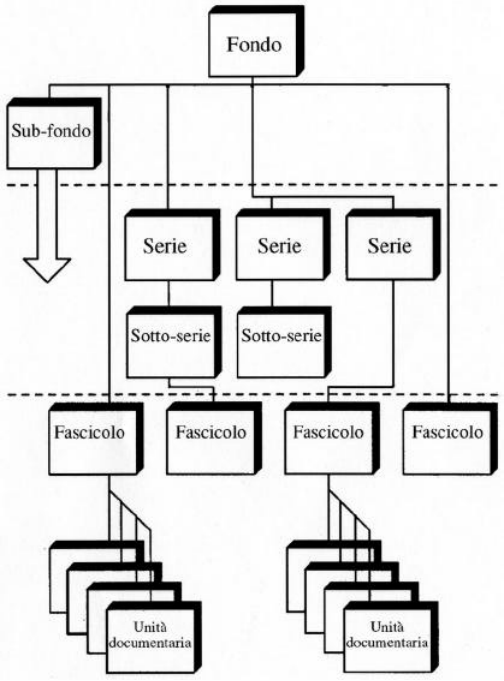
Per rappresentare correttamente i concetti descritti sopra, viene utilizzata una struttura gerarchica ad albero; una struttura di questo tipo è in grado di modellare in maniera opportuna la realtà di interesse e quindi consente di mantenere le relazioni tra i documenti di un archivio; inoltre, per collocare le singole entità che compongono un archivio [Vitali, 2010], cioè serie, unità, documenti (di seguito viene riportato uno schema esplicativo (Figura 2.6) e nel paragrafo successivo una breve spiegazione), all'interno del contesto archivistico di appartenenza è necessario introdurre il concetto di descrizione archivistica. Il comitato del Consiglio Internazionale degli Archivi (International Council of Archives) ha elaborato un modello teorico, definito come standard, di rappresentazione/descrizione degli archivi: ISAD(G) (verrà descritto più in dettaglio nella sezione successiva 2.1.5). La stesura di standard di descrizione archivistica permette di elaborare dei modelli di rappresentazione che siano condivisi e il cui livello di astrattezza è tanto maggiore quanto più ampia è la realtà da rappresentare. Gli obiettivi della standardizzazione possono essere considerati i seguenti:

- elaborazione di regole e accorgimenti che consentano di esplicitare la ricchezza informativa accumulata dal materiale archivistico lungo il percorso di produzione uso e conservazione;
- l'individuazione di modelli che consentano la circolazione delle informazioni;
- la semplificazione del recupero delle informazioni.

Concettualmente, la descrizione archivistica costituisce un'attività imprescindibile al fine del perseguimento degli obiettivi di valorizzazione e comunicazione delle fonti archivistiche, che non deve essere confusa come una semplice attività di catalogazione o schedatura, anche se queste operazioni fanno parte del processo di descrizione. Si tratta di un'attività complessa, che prende in considerazione tutte le entità informative che concorrono a definire un fondo archivistico e le relazioni che intercorrono tra tali entità. In questo senso le descrizione archivistica può essere definita come "*processo di raccolta, organizzazione ed analisi delle informazioni necessarie per la identificazione, la gestione e l'interpretazione del materiale conservato in un istituto archivistico*" e come "*l'illustrazione del contesto e del sistema archivistico in genere*".

[3] La definizione in lingua francese data nel [Walne, 1988]: *"Principe du respect des fonds, principe de provenance. Principe fondamental selon lequel les archives d'une meme provenance ne doivent pas etre mélangées à celles d'une autre provenance; ce principe inclut parfois le principe de respect de l'ordre primitif".*

Figura 2.6: Struttuta gerarchica di un archivio secondo ISAD(G) [ICA, 1999]

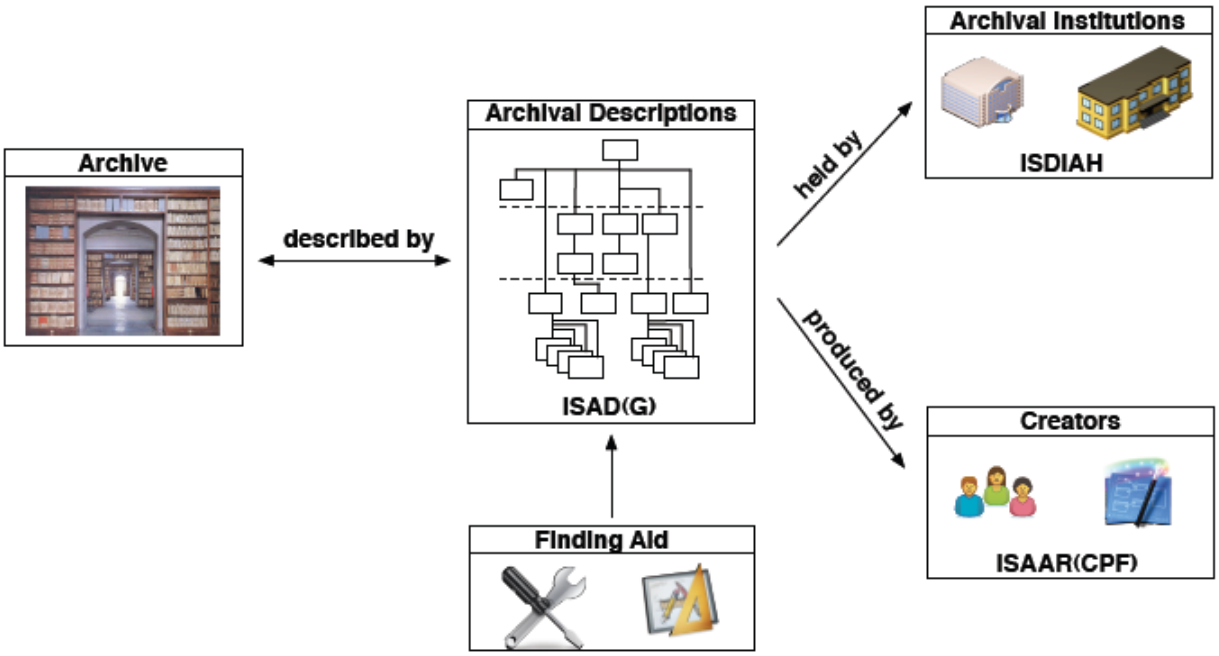


Semplificando, si possono considerare unità descrittive del contesto in esame le descrizioni di soggetti produttori, soggetti conservatori e ambiti politici istituzionali, mentre sono unità descrittive relative al contenuto quelle in merito ai diversi livelli del fondo e alle singole unità archivistiche. Esaminando la figura 2.6, è possibile notare che, secondo ISAD(G), un fondo può essere costituito da sottofondi, con questo termine si intende la partizione di un fondo contenente un insieme di documentazione correlata, corrispondente a suddivisioni amministrative dell'istituzione o dell'organismo produttore. A sua volta un fondo ed i suoi sub-fondi possono essere composti da serie archivistiche. Una serie archivistica è un raggruppamento di documentazione omogenea per tipologia o contenuto. Le serie possono essere suddivise in sottoserie, composte da fascicoli che a loro volta possono contenere i documenti archivistici. Secondo le logiche e le prassi definite dagli standard di descrizione ognuna di queste unità descrittive è un'entità autonoma e per ognuna di queste unità la tradizione archivistica prima e gli standard hanno individuato gli appropriati elementi descrittivi, cioè quelle informazioni che servono ad identificare e a rendere fruibili la singola unità. Il processo di descrizione si completa poi con la definizione delle corrette relazioni tra ognuna di queste unità descrittive.

Gli attori del sistema sono [Vitali, 2010]:

- produttori: "*L'ente, la famiglia o la persona che ha posto in essere, accumulato e/o conservato la documentazione nello svolgimento della propria attività personale o istituzionale*" [ISAD, Glossario]. Ogni archivio ha un proprio soggetto produttore, anche se all'archivio stesso possono corrispondere uno o più soggetti produttori che hanno nel corso della sua storia contribuito alla sua formazione, organizzazione e ordinamento. Sostanzialmente ci sono due categorie di produttori: pubblici e privati. Nella categoria dei soggetti pubblici si distinguono quelli statali, che comprendono le amministrazioni centrali e periferiche dello stato, e quelli non statali, che comprendono le amministrazioni di enti locali. Alla categoria dei soggetti privati appartengono i singoli, sia persone fisiche che giuridiche, i nuclei familiari, l'associazionismo senza scopo di lucro e le associazioni d'impresa costituite per finalità di lucro.;
- conservatori: soggetti, pubblici o privati, che conservano fondi archivistici e li rendono

Figura 2.7: schema descrizione archivistica [Silvello, 2011]



disponibili per la ricerca e la consultazione;

- strumenti di ricerca: termine generico con cui si indicano i vari tipi di strumenti messi a disposizione per rendere possibile e facilitare l'accesso ai dati presenti in un archivio. Tra questi, ci sono strumenti appositamente predisposti per la ricerca storico-archivistica, quali gli inventari, le guide e i censimenti descrittivi.

Tipi di archivi

Negli ultimi anni la distinzione tra vari tipi di archivi è cambiata; possiamo distinguere tre tipi di archivi:

- Archivi "analogici": nella definizione di archivi analogici sono compresi tutti quei complessi archivistici che raccolgono in massima parte documenti cartacei, cui si possono aggiungere documenti registrati su supporti diversi quali cassette, audio e video, nastri magnetici ecc.

- Archivi informatici: per archivio informatico si intenderà invece una sedimentazione di documenti informatici prodotti, utilizzati, gestiti e conservati esclusivamente attraverso sistemi informatici. Dell'archivio informatico possono entrare a far parte anche copie di documenti originariamente analogici e poi sottoposti ad un processo di digitalizzazione.

- Sistemi archivistici integrati: questo tipo di archivi sono il risultato della fusione o della sovrapposizione delle due tipologie precedenti e dei relativi modelli di gestione. I soggetti produttori di questa raccolta sono molteplici, come molteplici sono i loro profili giuridici che vanno dal privato cittadino alle istituzioni pubbliche e universitarie. La realizzazione di archivi di questo tipo racchiude grandi potenzialità, soprattutto dal punto di vista della creazione e gestione di oggetti digitali, tra cui la nascita di grandi archivi digitali che riescono a memorizzare enormi quantità di dati.

2.1.5 Standard tradizionali per gli archivi

ISAD(G)

L'introduzione dell'informatica e il suo utilizzo crescente nella descrizione dei fondi d'archivio, l'esigenza di disporre di strumenti descrittivi che fossero tra loro confrontabili e che facilitassero lo scambio d'informazioni sul patrimonio archivistico, ha indotto il Consiglio internazionale per gli Archivi a istituire nel 1990 una Commissione sulle norme di descrizione, che nel 1994 ha realizzato un documento che contiene le norme generali per la descrizione di qualunque archivio. Le ISAD recepiscono tutto ciò che era stato elaborato in precedenza in materia di descrizione, con l'obiettivo di rendere applicabili a qualunque tipo di archivio e a qualunque livello di descrizione le seguenti regole:

1. descrizione a più livelli per preservare il contesto, la struttura del fondo e le sue suddivisioni;
2. adattare le informazioni al livello di descrizione, fornendo solo i dati appropriati al livello che in quel momento si sta descrivendo;
3. mantenere, qualunque sia il livello in cui ci si trova, il posto che l'unità archivistica occupa nella gerarchia complessiva del fondo;
4. evitare la ripetizione di informazioni inserite in un livello precedente.

Le informazioni sono raggruppate dalle ISAD in 6 zone o gruppi omogenei di informazioni. Ciascuna zona è, a sua volta, suddivisa in ulteriori elementi informativi che, in totale, sono 26. Ogni zona deve soddisfare un insieme omogeneo di necessità.

1. Zona d'identificazione: insieme delle informazioni che servono ad identificare l'unità archivistica di descrizione. Comprende:
   a) referenza, indicazione codificata dell'amministrazione o istituto;
   b) titolo;
   c) data di creazione dei documenti che sono presenti nell'unità;
   d) livello della descrizione;
   e) indicazioni materiali sull'unità di descrizione (quantità, dimensione, tipo).
2. Zona del contesto: insieme delle informazioni sull'origine e conservazione delle unità di descrizione. Comprende:
   a) nome del produttore;
   b) storia degli enti o notizie biografiche;
   c) data di costituzione dell'unità descritta;
   d) storia della conservazione;
   e) modalità di acquisizione all'Archivio.
3. Zona del contenuto: insieme delle informazioni sull'oggetto dell'unità di descrizione e sulla sua situazione archivistica. Comprende:
   a) presentazione del contenuto (oggetto e tipo dell'unità descritta);
   b) scarti, eliminazioni e regole di conservazione;
   c) accrescimenti;

   d) organizzazione dei documenti (struttura interna, ordine, metodi di ordinamento seguiti dagli archivisti).

4. Zona delle condizioni d'accesso e utilizzazione: insieme delle informazioni sulle possibilità di accesso e consultazione dell'unità di descrizione. Comprende:

   a) stato giuridico (archivio pubblico o privato);

   b) accessibilità (incondizionata, condizionata, vietata);

   c) diritti d'autore (condizione per la riproduzione);

   d) lingua dei documenti;

   e) caratteristiche materiali che hanno influenza sulla sua consultabilità (macchie, danni di roditori);

   f) strumenti di ricerca.

5. Zona delle fonti complementari: insieme delle informazioni relative ad altra documentazione che ha rapporto significativo con l'unità di descrizione. Comprende:

   a) localizzazione degli originali (se l'unità descritta è una riproduzione);

   b) esistenza di copie (su supporti non tradizionali);

   c) fonti complementari nell'ambito dello stesso istituto ;

   d) fonti complementari presenti in altri istituti;

   e) bibliografia.

6. Zona delle note: a differenza delle altre zone non ha una struttura predeterminata. Fornisce tutte quelle informazioni che non trovano posto in nessuna delle altre zone.

ISAAR(CPF)

Per quanto riguarda le informazioni relative al produttore della documentazione (previste dalle ISAD nella zona del contesto), la Commissione ha pubblicato nel 1995 un altro insieme di regole: ISAAR (descrizione standardizzata di un soggetto produttore di documentazione). Tale descrizione è composta da norme ed elementi utili a descrivere sinteticamente il soggetto produttore. Si tratta di uno standard internazionale per la descrizione di enti, persone e famiglie produttrici di documenti d'archivio. Lo scopo delle regole è dare criteri uniformi per la descrizione del soggetto produttore. I criteri sono due:

1. uniformità della loro denominazione;
2. descrizione delle loro caratteristiche.

Le regole ISAAR prevedono una descrizione divisa in 3 aree o zone:

1. area del controllo autorità;
2. area delle informazioni;
3. area delle note.

La prima area è quella di maggiore importanza. La denominazione del soggetto dovrà essere data sempre nello stesso formato. L'area delle informazioni deve fornire per gli enti notizie relative alla composizione, allo stato giuridico, all'epoca di funzionamento, alle sedi, alle competenze, alla struttura organizzativa, alle relazioni con gli altri enti. L'area delle note prevede l'indicazione dei criteri seguiti dal compilatore della voce, la bibliografia, la data di compilazione.

ISDF

International Standard for Describing Functions. Questo standard, di più recente pubblicazione, completa le descrizioni realizzate sulla base di ISAD e ISAAR, consentendo di descrivere in forma normalizzata le funzioni dei soggetti coinvolti nella produzione e nella eventuale successiva gestione del materiale archivistico.

ISDIAH

International Standard for Describing Institutions with Archival Holdings. Lo standard, anch'esso di recente pubblicazione e anch'esso destinato ad integrare le descrizioni messe a punto sulla base degli altri standard, consente di descrivere in maniera normalizzata i soggetti conservatori di materiale archivistico.

2.1.6 Standard informatici per gli archivi

EAD

Lo sviluppo di nuove tecnologie ha interessato anche gli archivi e ha permesso, come visto in precedenza, la nascita delle Digital Libraries. Uno dei principali standard per la descrizione archivistica è l'EAD [Pitti, 1999, Barbanti, 2006]: Encoded Archival Description. Si tratta di uno standard SGML/XML ideato per la codifica degli strumenti archivistici, ma applicabile anche a fondi di manoscritti, oggetti bibliografici o museali. L'Encoded Archivial Description è uno standard XML creato per semplificare il reperimento delle informazioni negli archivi, sostenuto dalla Library of Congress e dalla Society of American Archivists. L'obiettivo del progetto, nato nel 1993 presso l'Università di Berkeley, era quello di creare uno standard che descrivesse le collezioni degli archivi, dei musei e delle biblioteche, con un occhio di riguardo alle raccolte di manoscritti: gli elenchi generati con lo standard dovevano essere interpretabili dai calcolatori e semplici da ricercare, conservare e scambiare. Le linee guida per l'utilizzo dello standard sono state scritte dal Research Libraries Group e stabiliscono elementi e attributi utilizzabili per codificare i documenti in maniera conforme (non verranno trattati in dettaglio in questa tesi). Il DTD (un documento che permette di definire un nuovo insieme di tag che rispettino le esigenze di codifica di un documento o di un insieme di documenti specifici) dell'Encoded Archivial description specifica quali elementi devoso essere utilizzati per descrivere una collezione di documenti e il modo in cui tali elementi devono essere utilizzati. La codifica EAD inserisce all'interno di ogni documento una sezione detta eadheader in cui sono contenuti elementi, come il titolo del documento, e informazioni dettagliate necessarie per il reperimento e la catalogazione del documento: il creatore, la data di creazione, la storia delle sue revisioni, la lingua in cui è scritto ecc. Riassumendo, le caratteristiche principali dello standard EAD:

1. è un sistema per la gestione e la condivisione del patrimonio archivistico, di ausilio per la conservazione, accessibilità, utilizzo, conservazione fisica o trattamento del materiale;
2. utilizza la tecnologia XML per la conservazione e la comunicazione fisica dei dati. Ciò rende il modello dati indipendente da specifiche piattaforme hardware e software, per cui la continuità della struttura e del contenuto garantisce l'accettabilità e la validità nel tempo di ogni applicazione che lo utilizzi;
3. permette la conversione di strumenti di ricerca archivistici in formato cartaceo e la loro pubblicazione in formato elettronico;
4. presenta elementi e attributi i cui nomi cercano di essere il più possibile universali dal punto di vista della lingua e dell'applicazione, al fine di favorire l'interscambiabilità e la portabilità;

5. è aperto a proposte di modifica, preferendo l'aggiunta piuttosto che la modifica di elementi, in modo che le versioni successive della DTD EAD siano compatibili con quelle precedenti.

Esempio di codice EAD, frammento che rappresenta l'header di un documento EAD:

```
<eadheader audience="internal" countryencoding="iso3166-1"
dateencoding="iso8601"langencoding="iso639-2b"
relatedencoding="DC" repositoryencoding="iso15511"
scriptencoding="iso15924">
  <eadid countrycode="us" identifier="bachrach_lf"
        mainagencycode="NSyU">bachrach_lf</eadid>
  <filedesc>
        <titlestmt>
          <titleproper encodinganalog="Title">
                Louis Fabian Bachrach Papers
          </titleproper>
          <subtitle>
                An inventory of his papers at Blank University
          </subtitle>
          <author encodinganalog="Creator">Mary Smith</author>
        </titlestmt>
        <publicationstmt>
          <publisher encodinganalog="Publisher">
                Blank University
          </publisher>
          <date encodinganalog="Date" normal="1981">1981</date>
        </publicationstmt>
  </filedesc>
  <profiledesc>
        <creation>John Jones
        <date normal="2006-09-13">13 Sep 2006</date>
  </creation>
  <langusage>
        <language encodinganalog="Language" langcode="eng">
          English
        </language>
 </langusage>
</profiledesc>
</eadheader>
```

Alcuni punti deboli di EAD:

- complessità dello standard, che riguarda soprattutto la difficoltà di recepire ed implementare le innovazioni all'interno dello standard [Silvello, 2011];
- EAD è codificato in un unico file XML, dove sono memorizzate sia informazioni di tipo strutturale che di contenuto informativo; per accedere alle informazioni di interesse è necessario percorrere e navigare tutto l'albero gerarchico e questo può incidere negativamente sulle prestazioni del sistema; inoltre questo fatto impedisce anche una gestione flessibile dell'archivio [Silvello, 2011].
- EAD permette diversi gradi di libertà nella costruzione dell'XML; questo può risultare problematico nel caso in cui vengano usate, per processare i file, delle procedure automatizzate, in quanto non è sempre possibile conoscere in anticipo come viene costruito l'XML (soprattutto quando vengono importati da altri sistemi archivistici); infatti può

accadere che uno stesso campo descrittivo venga interpretato in maniere differente da archivisti diversi. [Silvello, 2011].

I sistemi informativi archivistici

I sistemi informativi archivistici sono costituiti da componenti software complessi e vengono utilizzati per gestire un archivio. Un sistema di questo tipo non è un database, ma il risultato di un processo/progetto di elaborazione culturale finalizzato ad una corretta rappresentazione ed utilizzazione di tutte le entità informative che caratterizzano il materiale archivistico. Nella realizzazione di un sistema informativo archivistico questa progettualità culturale è finalizzata soprattutto a recuperare il ruolo di mediazione esercitato dagli archivisti per favorire l'accesso alle fonti e si concretizza nella corretta restituzione delle informazioni relative sia al contenuto che al contesto. L'obiettivo principe di un sistema informativo archivistico è rendere più efficace quanto fatto manualmente dagli addetti e quindi di portare ad un'evoluzione delle funzionalità standard di un archivio, quali creazione, reperimento, conservazione ecc. di dati archivistici. Nello specifico un sistema informativo archivistico deve innanzitutto consentire di reperire il materiale che si sta cercando e di identificarlo in maniera univoca. Una volta garantiti questi risultati il sistema dovrà poi permettere all'utente di selezionare, tra ciò che si è reperito e identificato, ciò che è rilevante ai fini della ricerca che si sta conducendo e, naturalmente di ottenere (consultare) ciò che si è selezionato.

La maggior parte dei sistemi informativi archivistici analizzati prevede due distinti ambienti di lavoro, che permettono di distinguere anche tra due tipologie di utenti: l'utente archivista e l'utente finale (colui che accede ai contenuti senza modificarli). Il primo che possiamo definire è l'interfaccia autore, dove si creano le relazioni tra le entità informative di base e si manipolano le informazioni, creando i record descrittivi. In questa fase il sistema è gestito esclusivamente dagli archivisti e di questo ambiente l'utente finale non ha percezione. Su un altro versante, invece, si colloca l'interfaccia utente, l'ambiente cioè che consente l'accesso alle informazioni. Questa componente è essenziale e deve essere progettata con grande attenzione. In linea generale, l'interfaccia di consultazione deve consentire la diversificazione dei percorsi di ricerca, tramite diversi punti di accesso al sistema, e deve essere di facile utilizzo per gli utenti del sistema.

2.2 Stato dell'arte nella Pubblica Amministrazione: Sistemi di gestione degli Archivi

2.2.1 Parametri di confronto

Prima di descrivere alcuni dei più importanti sistemi informativi archivistici sviluppati ed utilizzati in Italia, viene introdotta una possibile lista di parametri di valutazione per i sistemi stessi, attraverso i quali sarà possibile, alla fine di questa sezione, mettere a confronto le caratteristiche dei vari sistemi per ricavarne i punti di forza e le possibili debolezze.

I parametri non sono tutti ugualmente importanti e hanno semantiche diverse. Alcuni riguardano l'aderenza alla pratica archivistica, altri le caratteristiche del sistema e altri il rilascio dei dati e le note legali. Nelle prossime sezioni i parametri vengono raccolti per le tipologia di interesse appena citate.

Raccolta metadati da altri sistemi archivistici:

Alcuni sistemi archivistici, in base alle loro finalità (ad esempio se un archivio è stato pensato e progettato per raccogliere dati da altri archivi), possono prevedere o meno una funzionalità

di raccolta e import di metadati provenienti da altri sistemi archivistici. Inoltre, tra quelli che permettono di effettuare l'import ci sono i sistemi che usano il protocollo OAI-PMH (Open Archive Initiative Protocol for Metadata Harvesting). L' OAI (Open Archives Initiative)[4] è un progetto nato con lo scopo di gestire e rendere disponibili gli archivi che contengono dati prodotti in ambiente accademico. Il modello funzionale di OAI è cosituito da due componenti:

- Data Provider: gestiscono uno o più archivi, detti anche Repositories, di collezioni di oggetti digitali e sono responsabili del loro mantenimento e della generazione dei metadati che li caratterizzano. Supportano il protocollo OAI-PMH per consentire l'accesso ai metadati sul contenuto. I Data Providers si occupano anche di mettere a disposizione i metadati, curandone la qualità e la completezza;
- Service Provider: gestiscono i servizi per l'aggregazione e l'indicizzazione dei metadati (ricerca, scoperta, localizzazione degli oggetti digitali) e interrogano gli archivi dei Data Provider, usando le richieste del protocollo OAI-PMH per recuperarne i metadati.

I Service Providers interrogano i data provider, da cui prelevano i metadati tramite il protocollo OAI-PMH (Protocol for Metadata Harvesting). I contenuti digitali vengono vista su tre livelli:

1. risorsa: è l'oggetto contenuto nei digital repository mantenuti dai data provider;
2. item: contenitore di tipo logico a partire dal quale vengono diffusi i metadati;
3. record: sono i metadati tramite l'XML secondo lo schema Dublin Core, ma possono essere gestiti anche metadati definiti in specifici domini di applicazione.

Il protocollo OAI-PMH[5] ha un insieme di comandi che vengono definiti:

- GetRecord: per recuperare i record;
- ListIdentifier: per ottenere la lista di identificatori;
- ListRecord: per la lista di identificatori;
- Identify: permette di ricavare informazioni generali sugli archivi e sugli stessi contenuti;
- ListMetadataFormats: restituisce il tipo di metadati usati;
- ListSets: per interrogare i repository.

I valori possibili per questo parametro sono quindi: si/no/uso protocollo OAI-PMH.

EAD

Questo parametro indica se un sistema implementa o meno lo standard per la descrizione archivistica EAD (visto in precedenza); i valori possibili sono si/no.

ISAD(G)-ISAAR (CPF)-ISDIAH

Questo parametro valuta se un sistema aderisce o meno ai standard per la descrizione archivistica ISAD(G)-ISAAR (CPF)-ISDIAH (visti in precedenza); anche qui i valori possibili sono si/no.

Architettura

Questo parametro riguarda l'architettura del sistema, dove con architettura si intende la struttura, le componenti che la compongono e come sono legate tra loro. Le possibili architetture sono:

[4] http://www.openarchives.org/
[5] http://www.openarchives.org/OAI/openarchivesprotocol.html

- centralizzata: un'architettura di questo tipo indica che il sistema è costituito da componenti che possono essere usate solo Stand Alone (un unico nodo di elaborazione); l'applicazione del sistema deve quindi gestire i dati, la logica di business e l'interfaccia utente;
- distribuita: un sistema di questo tipo viene utilizzato in un contesto diverso, costituito solitamente da un rete di nodi (un sistema che presenta un'interfaccia per il Web è distribuito). Nei sistemi distribuiti, gli strati software sono installati su livelli hardware, detti tier, dove un livello rappresenta in generale una macchina, di diversa capacità elaborativa. Da questo punto di vista, un'applicazione può essere configurata con:
  - un livello di distribuzione (Single Tier): tutti e tre i livelli software sono assegnati ad un'unica macchina, questa configurazione è quella classica terminale-host che caratterizzava i sistemi basati su mainframe;
  - due livelli di distribuzione (Two Tier): i livelli applicativi sono divisi tra la stazione di lavoro dell'utente e la macchina server che ospita le componenti di business e di logica dei dati. Sulla stazione di lavoro è realizzata la logica di presentazione e sulla macchina server quella d'accesso e di gestione dei dati;
  - tre livelli di distribuzione (Three Tier): i tre livelli applicativi sono il Client-tier (livello dell'interfaccia utente), il Business-tier (livello dove risiedono i componenti che implementano la logica di business) e il Resource-tier (livello che si occupa della gestione dei dati).

Le architetture distribuite hanno i seguenti vantaggi, rispetto a quelle centralizzate:

- connettività e collaborazione: possibilità di condividere risorse hardware e software (compresi dati e applicazioni);
- prestazioni e scalabilità: la possibilità di aggiungere risorse fornisce la capacità di migliorare le prestazioni e sostenere un carico che può aumentare (scalabilità orizzontale);
- tolleranza ai guasti, grazie alla possibilità di replicare risorse;
- uso di protocolli standard: favorisce l'interoperabilità di hardware e software di fornitori diversi.

Gli svantaggi, invece, sono:

- complessità: i sistemi distribuiti sono più complessi di quelli centralizzati;
- sicurezza: l'accessibilità in rete pone problematiche di sicurezza;
- non prevedibilità: i tempi di risposta dipendono dal carico del sistema e dal carico della rete, che possono cambiare anche rapidamente.

Portale/sito Web

Il parametro in esame indica se il sistema mette o meno a disposizione dell'utenza, che è interessata ad accedere ai dati dell'archivio, un sito o portale Web con una serie di funzionalità aggiuntive, come la visualizzazione della documentazione archivistica e la ricerca dei documenti. Un sito Web di questo tipo espone semplicemente i dati presenti nell'archivio mentre un portale Web viene creato da chi sviluppa il software e fornisce delle funzionalità aggiuntive oltre alla semplice visualizzazione dei dati. Oltre a quanto appena indicato, il concetto che si vuole analizzare con questo parametro riguarda anche la possibilità di creare o meno un sito o un portale a partire dai dati del sistema. I valori sono si/no.

DBMS

Questo parametro analizza il tipo di DBMS (DataBase Management System) utilizzato dal sistema in esame. La scelta del tipo di DBMS può essere critica, infatti un DBMS permette di accedere in modo semplice e efficiente ad una base di dati mantenendone la consistenza, la privatezza e l'affidabilità. I vantaggi sono:

- Accesso ai dati tramite un linguaggio universale. Ogni DBMS di una certa tipologia mette a disposizione un linguaggio di interrogazione (SQL nel caso relazionale).
- Atomicità delle operazioni. Un DBMS permette di effettuare sequenze di operazioni (transazioni) in modo atomico. Ciò significa che l'intera sequenza di operazioni viene eseguita con successo oppure nessuna di queste operazioni ha alcun effetto sui dati della base. L'atomicità delle transazioni permette di mantenere una stato della base di dati consistente con la realtà in esame.
- Accesso concorrente ai dati. Un DBMS permette a più utenti di accedere contemporaneamente alla base di dati.
- Privatezza dei dati. Un DBMS permette un accesso protetto ai dati. Utenti diversi possono avere accesso a diverse porzioni della base di dati e possono essere abilitati a certe operazioni su di esse.
- Affidabilità: un DBMS offre dei metodi per salvare copie dei dati (backup) e per ripristinare lo stato della base di dati in caso di guasti software e hardware (recovery).

I valori possibili sono costituiti dai principali DBMS disponibili:

- Oracle[6] e SQL Server[7], sono DBMS proprietari;
- MySql[8], PostgreSQL[9] e Derby[10] hanno invece una licenza open source.

Sistema operativo

Questo parametro riguarda il tipo di sistema operativo (oppure i sistemi operativi supportati, se il sistema è progettato per essere multipiattaforma) supportato dal sistema archivistico. I valori possibili sono: Windows, MacOS, Linux, Unix-platforms oppure indipendente dal sistema operativo.

Architettura Web application

Questo parametro stabilisce se un sistema archivistico mette a disposizione un portale Web basato sull'architettura di tipo REST e con il supporto alle Portlet, oppure se il sistema fornisce un'applicazione Web custom progettata appositamente per il sistema in esame. Il paradigma REST (Representational State Transfer) è uno stile architetturale per sistemi software distribuiti e indica una serie di principi architetturali per la progettazione di Web Service. Le Portlet, invece, sono componenti Web, sviluppate in Java, indipendenti e riutilizzabili che vanno a comporre un portale Web che supporta questa tecnologia (verranno introdotte e descritte in dettaglio nel capitolo 4); i valori possibili sono REST-Portlet/custom.

Reportistica

Questo parametro è relativo alle caratteristiche del sistema e riguarda il supporto alla reportistica offerto dal sistema in esame. Un criterio possibile, oltre alla presenza o meno della funzione di generazine di report, può essere quello basato sulle tipologie di report prodotte:

[6] http://www.oracle.com/us/products/database/overview/index.html
[7] https://www.microsoft.com/italy/server/sql/default.mspx
[8] http://www.mysql.it/
[9] http://www.postgresql.org/
[10] http://db.apache.org/derby/

- report completi: permettono di ottenere la stampa in dettaglio di tutto l'albero archivistico;
- report riassuntivi: permettono di ottenere delle statistiche sui documenti contenuti nel sistema informativo archivistico (inventario);
- report generici: i report vengono generati dal sistema ma le informazioni sul tipo di report non sono disponibili.

Funzionalità di ricerca

Questo parametro invece stabilisce se un sistema possiede delle funzionalità di ricerca che permettono di recuperare i dati di interesse; ad esempio se un utente vuole ricercare un dato documento, oppure se vuole trovare una pratica burocratica. I valori possibili possono essere: ricerca di base, ricerca avanzata, ricerca Google-like.

Licenza

Questo parametro riguarda il tipo di licenza d'utilizzo del sistema. Esistono licenze di vario tipo, di seguito vengono elencate le più diffuse:

- GPL: la General Public License è una licenza per software libero ed comunemente indicata anche con l'acronimo GNU GPL. A differenza delle licenze per software proprietario, la GNU GPL assicura all'utente la libertà di utilizzo, di copia, di modifica e di distribuzione del software, con la sola restrizione che il codice risultante deve ereditare la stessa licenza dei codice sorgente originario;
- BSD: le licenze BSD sono una famiglia di licenze permissive per software. Il loro nome deriva dal fatto che la licenza BSD originale (detta anche licenza BSD con 4 clausole) fu usata originariamente per distribuire il sistema operativo Unix Berkeley Software Distribution (BSD), una revisione libera di UNIX sviluppata presso l'Università di Berkeley. La differenza principale con la GPL è costituita dal fatto che, mentre la GPL impone che ogni lavoro derivato sia rilasciato secondo le stesse modalità, la licenza BSD consente di modificare e ridistribuire il software anche con modalità differenti;
- proprietaria: questo tipo di licenza, invece, impone che il software abbia delle restrizioni sul suo utilizzo, sulla sua modifica, riproduzione o ridistribuzione e sono imposte da chi ne detiene il diritto d'autore.
- Creative Commons: queste licenze offrono sei diverse articolazioni dei diritti d'autore per artisti, giornalisti, docenti, istituzioni ecc.. Il detentore dei diritti può non autorizzare a priori usi prevalentemente commerciali dell'opera (opzione Non commerciale) o la creazione di opere derivate (Non opere derivate); inoltre se sono possibili opere derivate, può imporre l'obbligo di rilasciarle con la stessa licenza dell'opera originaria (Condividi allo stesso modo). Le combinazioni di queste scelte generano le sei licenze CC, disponibili anche in versione italiana.

Esportazione dati

La possibilità di esportare i dati contenuti nell'archivio è un proprietà importante per un sistema informativo archivistico; i dati che vengono esportati possono essere:

- dati testuali, in formato CSV;
- dati complessi, in formato XML EAD;
- dati complessi, in formato XML non EAD;
- dump SQL.

2.2.2 Sistema Archivistico Nazionale

Il Sistema Archivistico Nazionale è stato creato grazie a un'iniziativa della Direzione generale per gli archivi, d'intesa con le Regioni, dell'Unione province d'Italia (UPI), dell'Associazione nazionale comuni italiani (ANCI) e delle Province autonome di Trento e Bolzano. Il SAN permette di accedere alle informazioni sul patrimonio archivistico italiano pubblicate nel Web dai diversi sistemi di descrizione archivistica che aderiscono ad esso. Lo scambio dei dati e dei metadati avviene mediante l'utilizzo del protocollo OAI-PMH.

Il SAN consente di recuperare diversi tipi di dati e contenuti:

- le descrizioni di alto livello degli archivi, provenienti dai sistemi aderenti a SAN;
- le risorse digitali (documenti, inventari di fondi, pubblicazioni nel web) organizzate in un Archivio Digitale;
- i contenuti Web;
- le risorse bibliografiche;
- i contenuti editoriali e scientifici tesi che interessano il patrimonio archivistico italiano.

Il portale del SAN, raggiungibile tramite il link *http://san.beniculturali.it/*, è formato da diverse sezioni; di seguito verranno riportate le più importanti:

- **Il SAN.** Costituisce l'area del portale in cui viene descritto il sistema. Si occupa di fornire le informazioni in merito all'utilizzo dei contenuti, ai soggetti che contribuiscono ad alimentare il repository degli archivi esposto dal portale e agli standard di progetto. Altre informazioni riguardano il mondo degli archivi in Italia e all'estero oltre che le normative italiane.
- **Cerca nel SAN.** Costituisce l'area del portale che consente di effettuare le ricerche sui contenuti degli archivi presenti nel sistema. Le funzioni di ricerca consentono di definire criteri specifici che insistono sia su oggetti archivistici che su documenti digitali. La consultazione dei singoli risultati ottenuti è caratterizzata da informazioni che dettagliano ogni singolo elemento; nel caso di documenti digitali è possibile fruirne attraverso le funzionalità presenti sul proprio personal computer.
- **Portali tematici.** Costituisce l'area del portale SAN che descrive ed espone i link che permettono di raggiungere i siti che si occupano di rendere fruibili oggetti archivistici e documenti digitali caratterizzati dall'appartenere a specifiche tematiche.
- **Strumenti per le ricerche.** Compone l'area del portale che contiene gli strumenti di ausilio per le ricerche operate sugli archivi.

2.2.3 SIUSA (Soprintendenza per i Beni Culturali)

SIUSA [CRIBC, 2001] è un'evoluzione, sia dal punto di vista concettuale sia dal punto di vista tecnico, del primo sistema informativo sviluppato negli anni ottanta: l'Anagrafe degli archivi italiani. SIUSA ha adottato lo standard proposto da ISAD(G) e ISAAR(CPF) e lo ha esteso, proponendo descrizioni separate tra loro correlate di tutte le entità appartenenti sull'archivio. Il SIUSA prevede la realizzazione di due database informativi, fisicamente e logicamente distinti ma tra loro collegati: il descrittivo e il gestionale. L'ambito descrittivo si articola in tre principali oggetti (database): il Complesso archivistico, il Soggetto conservatore e il Soggetto produttore. Il Complesso archivistico è formato da un'insieme di documenti che presentano caratteri di unitarietà ed omogeneità, quali fondi, subfondi, serie, sottoserie.

Figura 2.8: Home del Portale del SAN

SIUSA infatti non si configura come database di tipo inventariale. Si propone invece come punto di accesso primario per la ricerca generale su tutto il patrimonio archivistico nazionale non statale. Nella scheda del Soggetto conservatore sono presenti le informazioni riguardanti la conservazione della documentazione, che potranno essere utili all'utente per la consultazione. Le informazioni in merito al Soggetto produttore sono, come da norme ISAAR(CPF), trattate in maniera separata e interrelata rispetto alla descrizione della documentazione archivistica.

Il software SIUSA è costituito da un insieme di programmi sviluppati per memorizzare le descrizioni archivistiche, per renderle disponibili agli utenti e per recuperare il pregresso del progetto Anagrafe, con la possibilità di importare altre descrizioni archivistiche memorizzate su sistemi diversi e eterogenei. La parte più complessa è costituita da tre sezioni separate e autonome:

1. Il Datentry per l'immissione e l'aggiornamento dei dati sia dell'Ambito Descrittivo che dell'Ambito Gestionale.
2. Il sistema per la gestione degli utenti (inserimento e aggiornamento degli utenti e dei loro relativi diritti sugli Oggetti Archvistici di loro competenza.
3. Il sistema per il Search e il Retrieval dei dati contenuti nella base di dati.

Le tre parti sono indipendenti e sviluppate con linguaggi diversi. Al momento:

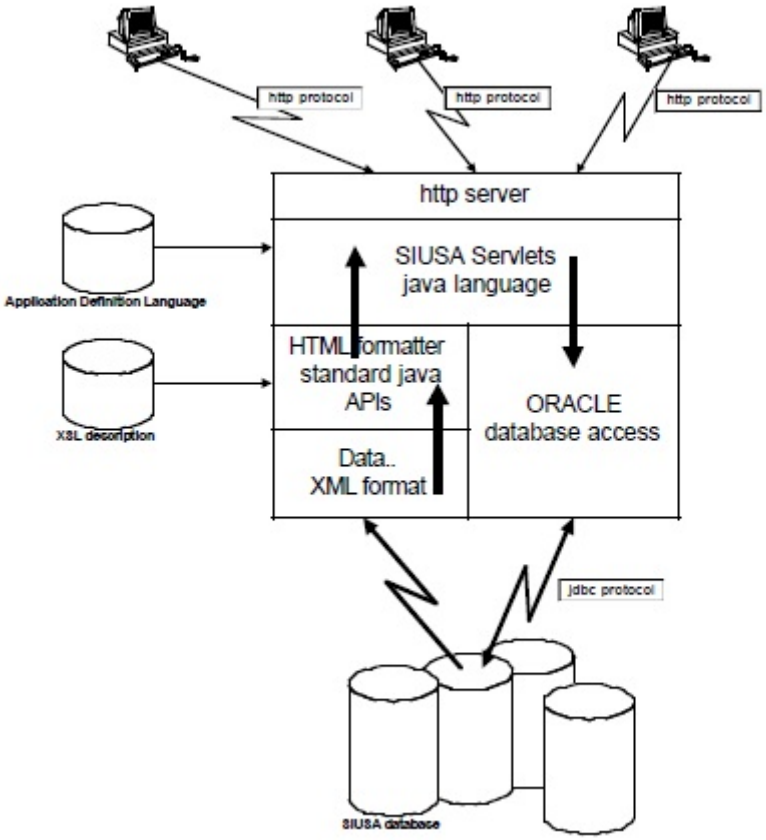
- Il Datentry è stato sviluppato in Java.
- La gestione degli utenti è stata sviluppata in linguaggio Perl.
- Il sistema di Search/Retrieval, è stato anch'esso sviluppato, in Perl.

Per utilizzare il sistema il link è il seguente: *http://siusa.archivi.beniculturali.it/*

2.2.4 Archimista (Regione Lombardia)

Archimista è un'applicazione open source per la schedatura, l'ordinamento e la descrizione degli archivi storici. È stato realizzata sulla base di un accordo tra Regione Lombardia, Regione

Figura 2.9: Schema del Dateentry di SIUSA [CRIBC, 2001]



Piemonte e Direzione Generale per gli Archivi che hanno conferito un incarico ad hoc all'Università degli Studi di Pavia. L'applicazione è stata sviluppata dalla Cooperativa Codex di Pavia. Archimista mira a sostituire nell'uso i software Guarini Archivi e Sesamo e l'obiettivo è quello di creare inventari e censimenti. Archimista è orientato al web ed è sviluppato per essere compatibile con qualunque browser moderno. Le funzionalità del software sono concentrate sulle operazioni di riordino, classificazione e gestione massiccia di singole unità (anche attraverso viste tabellari che ne semplifichino le operazioni ripetitive). Le principali entità in gioco sono i complessi archivistici, le unità archivistiche, i soggetti produttori e i soggetti conservatori descritti nel rispetto degli standard internazionali (rispettivamente ISAD(G) per i primi due, ISAAR(CPF) e ISDIAH). Sono previste anche una serie di schede accessorie per memorizzare anche dati di altra natura. Un'altra entità accessoria è data dagli oggetti digitali collegabili con qualsiasi livello dei complessi archivistici, con le unità archivistiche con i soggetti produttori e conservatori e con le risorse stesse. Ad esempio gli oggetti digitali potranno essere costituiti da un semplice pdf contenente un testo introduttivo o, soprattutto nel caso dei complessi archivistici e delle unità archivistiche e documentarie, immagini da legare alla descrizione.

Di seguito vengono riportati alcune schermate di esempio di archimista.

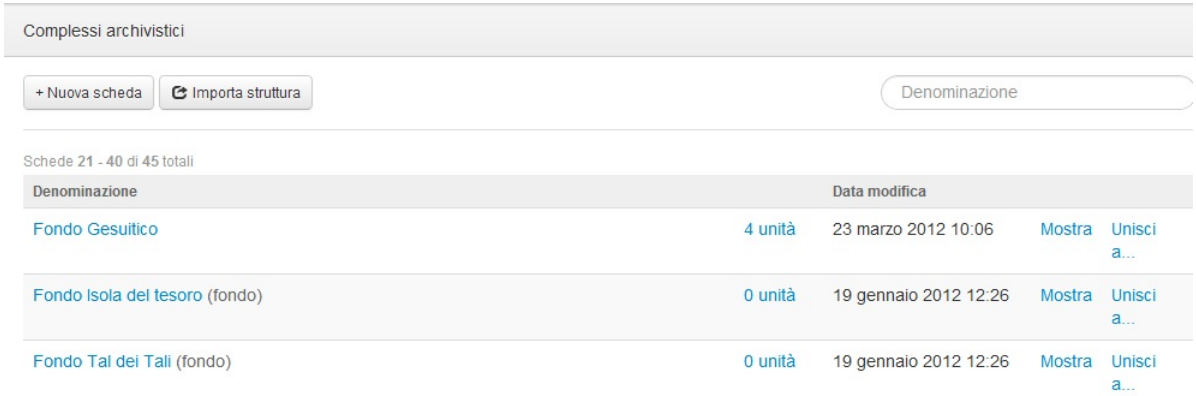
La bacheca è la home dell'applicazione. Fornisce il panorama delle principali schede compilate. Nell'esempio qui sotto sono presenti 47 complessi archivistici, 30 produttori e 15 conservatori. È possibile creare una nuova scheda cliccando sul comando omonimo. Nel caso del soggetto produttore, il comando "Nuova scheda" permetterà la scelta della tipologia (ente, persona, famiglia) che in ogni caso potrà essere cambiata in seguito.

Si accede all'ambiente "Complessi archivistici" da "Bacheca" o dal menu "Schede" presente in alto. Vengono mostrati i complessi di primo livello in lavorazione nel formato elenco; per

Figura 2.10: Schermata iniziale di Archimista

ciascuno sono indicati la denominazione, gli estremi cronologici, la tipologia (archivio, fondo, superfondo, ecc.), il numero di unità archivistiche e la data di ultima modifica:

Figura 2.11: Schermata dei Complessi archivistici di Archimista



Il sistema è disponibile per il download al sito *http://siusa.archivi.beniculturali.it/*.

2.2.5 IBC Archivi (Regione Emilia Romagna)

IBC Archivi è un'iniziativa dell'Istituto per i beni culturali della Regione Emilia-Romagna, sviluppata dal servizio di Soprintendenza per i beni librari e documentari e finalizzata alla creazione, gestione e pubblicazione in rete di risorse informative relative agli archivi storici emiliano-romagnoli e agli istituti ed enti che li conservano. Il progetto riguarda gli archivi storici di interesse locale appartenenti o conservati per lo più dagli enti locali, comuni e province del territorio regionale. Può però essere esteso a soggetti diversi, pubblici e privati, mediante la formalizzazione di accordi specifici. In vista della massima diffusione delle informazioni e della loro integrazione con le risorse della rete, IBC Archivi adotta standard e protocolli di comunicazione che consentono lo scambio di dati e l'interoperabilità con altri software e sistemi informativi.

All'indirizzo *http://archivi.ibc.regione.emilia-romagna.it/* è possibile utilizzare il sistema.

Con le voci del menu **Ricerca** si possono consultare informazioni a livello regionale sugli archivi ("Quale documentazione si conserva?"), sui loro soggetti produttori ("Quali enti, famiglie e persone hanno prodotto nel corso della loro storia la documentazione?") e sui loro soggetti conser-

vatori ("Dove si conserva la documentazione e come accedervi?"). Selezionando Inventari online è possibile accedere direttamente alla consultazione delle descrizioni archivistiche disponibili, effettuare ricerche su uno o più archivi, navigare nelle strutture dei fondi archivistici:

- la voce **Archivi** dà accesso alla lista dei complessi archivistici censiti; l'esito della ricerca sarà una descrizione sintetica del complesso archivistico, a partire dal quale si potrà consultare, se disponibile, l'inventario on line e navigare fino alla descrizione del soggetto conservatore del complesso archivistico;
- la voce **Soggetti produttori** fornisce una lista degli enti pubblici o privati, delle famiglie o persone che hanno prodotto nel corso del tempo gli archivi attualmente consultabili nella sezione Inventari on line;
- la voce **Soggetti conservatori** dà accesso alla lista degli istituti culturali censiti come conservatori di complessi archivistici. L'esito della ricerca sarà una descrizione del soggetto conservatore con informazioni di contatto e dati analitici relativi tra l'altro a sedi, orari, servizi, attività, modalità di accesso, attrezzature e strumenti di ricerca disponibili per l'utenza;
- la voce **Inventari online** permette di consultare le basi dati disponibili, effettuare ricerche multiarchivio e navigare nelle strutture gerarchiche dei complessi archivistici descritti.

La costruzione del sistema informativo Ibc ha previsto il recupero integrale delle diverse basi dati preesistenti: sono stati importati in xDams, con la codifica XML dei dati secondo la DTD EAD, gli inventari informatizzati gestiti dall'applicativo Sesamo. Nella piattaforma è stata importata, ed integrata in un unico repository, la banca dati del CAStE-R, ovvero le analitiche informazioni descrittive degli istituti conservatori raccolte con il Censimento degli archivi storici dell'Emilia Romagna e gestite con un applicativo ad hoc.

2.3 Altri prodotti proprietari

Archivio Storico

E' un software creato da AICoD (Azienda per l'Informazione e la Comunicazione Digitale), una web agency specializzata nello sviluppo di siti Internet e piattaforme web per le amministrazioni pubbliche, le aziende e i privati. La piattaforma, i cui obiettivi sono quelli di fornire solidità, affidabilità e velocità, guida l'utente nella fase di compilazione e gestione dei contenuti. L'obiettivo è di fornire un archivio digitale completo e catalogabile con un motore di ricerca per una gestione veloce dell'archivio, la conversione in automatico di tutti i file multimediali e la gestione dell'archivio fotografico, audio e video, sia in formato leggero che in alta definizione.

Arianna

Si tratta di un insieme di prodotti, sviluppati da Hyperborea, per la descrizione e gestione di archivi storici e di deposito. In particolare, il software Arianna3:

- fornisce strumenti per l'attività di descrizione, di riordino e di indicizzazione;
- è conforme agli standard internazionali di descrizione archivistica ISAD e ISAAR;
- è compatibile con il tracciato del Sistema informativo unificato delle Soprintendenze archivistiche (SIUSA);
- adotta lo standard XML/EAD;

- supporta tutti i più diffusi DBMS (MySQL, PostgreSQL, Oracle o SQLserver) e può gestire contemporaneamente più banche dati appartenenti anche a sistemi archivistici diversi;

- presenta un'interfaccia in inglese e italiano;

- adotta lo standard UNICODE, permettendo di produrre descrizioni archivistiche con i caratteri di tutte le lingue;

- utilizza un motore di indicizzazione "full-text" che permette di effettuare qualsiasi tipo di ricerca all'interno delle banche dati archivistiche.

Arianna Web è invece un software progettato per pubblicare su Internet banche dati archivistiche in formato XML/EAD, in grado di interoperare con molti software per la realizzazione di descrizioni archivistiche utilizzando mappature specifiche tra i vari modelli descrittivi e il formato XML/EAD. L'applicativo può restituire le informazioni sia in forma testuale, pubblicando tutti i campi che sono stati compilati in fase di descrizione, sia pubblicando anche riproduzioni digitali collegate alla descrizione archivistica.

GEA 5.0

Gea 5.0 è un software per descrivere, gestire e consultare gli archivi storici, sviluppato dal Consorzio BAICR Sistema Cultura. In particolare, l'organizzazione delle informazioni archivistiche rispetta la tradizione descrittiva nazionale ed è conforme agli standard internazionali ISAD(G) (per la descrizione archivistica), ISAAR(CPF) (per i record d'autorità) e ISDIAH (per i soggetti conservatori). L'applicativo genera file XML basati sugli standard EAD. Alcune funzionalità sono:

- struttura archivistica: con il livello di descrizione "Superfondo" è possibile strutturare un complesso di fondi che può riguardare o entità reali cui per convenzione non si attribuisce il livello di fondo (ad esempio, un ministero) o aggregazioni di altro tipo (cronologiche, tematiche, geografiche); la struttura del fondo può essere determinata preventivamente o in corso d'opera, così come l'assegnazione delle singole schede ai diversi livelli della struttura; inoltre, sono attive funzioni di spostamento di schede vincolate al rispetto di coerenze archivistiche;

- schede biografiche: richiamabili tramite link a partire da alcuni campi delle schede archivistiche;

- strumenti di ordinamento: per ogni livello archivistico è possibile dettare criteri e sottocriteri di ordinamento diversi (numerico, cronologico, alfabetico, per segnatura originaria, per classificazione, per protocollo) assegnando poi alle sequenze ordinate una numerazione, sempre modificabile;

- navigazione semantica: garantita dalla possibilità di stabilire collegamenti ipertestuali tra schede diverse sulla base dei contenuti o dei contesti di riferimento;

- ricerca: generica, per campo o per tipo di scheda;

- stampe e reportistica.

2.4 Tabella comparativa

Di seguito viene introdotta una tabella comparativa dei Sistemi Informativi Archivistici descritti nelle sezioni precedenti.

Tabella 2.2: Confronto tra i sistemi

Sistema	SAN	Siusa	Archimista	IBC Archivi	Arianna	Gea 5.0
Architettura del sistema	distr.	3-tier	distr.	distr.	centr.	centr.
Licenza	prop.	prop.	GPL	prop.	prop.	prop.
Esportazione dati	EAD	N.D.	EAD	N.D.	dati grezzi	dati grezzi
Raccolta metadati da altri sistemi archivistici	OAI-PMH	si	si	OAI-PMH	no	no
EAD	si	no	no	si	si	si
ISAD(G)-ISAAR (CPF)-ISDIAH	si	si	si	si	si	si
Portale/Sito Web	si	no	si	si	si	no
DBMS	N.D.	Oracle	N.D.	N.D.	tutti	N.D.
Sistema Operativo	ind.	ind.	ind.	ind.	Windows	N.D.
Architettura REST/Portlet	REST-Portlet	custom	custom	REST-Portlet	custom	custom
Reportistica	r.gener.	r.gener.	r.compl.	r.gener.	r.gener.	r.gener.
Funzionalità di ricerca:	avanzata	di base	avanzata	avanzata	avanzata	di base

Sulle colonne vengono inseriti i sistemi archivistici che vengono confrontati.

Sulle righe, invece, vengono dati i parametri di confronto, già introdotti all'inizio di questa sezione (tra parentesi vengono indicate le abbreviazioni usate nella tabella):

- Architettura del sistema: centralizzata (centr.)/distribuita (distr.)/monolitica/3-tier.
- Licenza: tipo di licenza; GPL/BSD/licenza proprietaria (prop.)/Creative Commons.

- Export dati: dati grezzi/formato XML EAD/dati complessi, in formato XML non EAD/dump SQL;
- Raccolta metadati da altri sistemi archivistici: sì/no/protocollo OAI-PMH.
- EAD: si/no.
- ISAD(G)-ISAAR (CPF)-ISDIAH: si/no.
- Portale/Sito Web: sì/no.
- DBMS: Postgresql/MySQL/Oracle/tutti.
- Sistema Operativo: Windows/Linux/MacOS/Unix/indipendente dal sistema (ind.).
- Architettura Web Application: i valori possibili sono REST-Portlet/custom.
- Reportistica: report completi (r. compl.)/report riassuntivi (r. riass.)/report generici (r. gener.).
- Funzionalità di ricerca: di base, avanzata, ricerca Google-like.

Infine viene usato il valore N.D. (Non Disponibile) per indicare se uno o più parametri non sono disponibili oppure non ricavabili per un dato sistema archivistico.

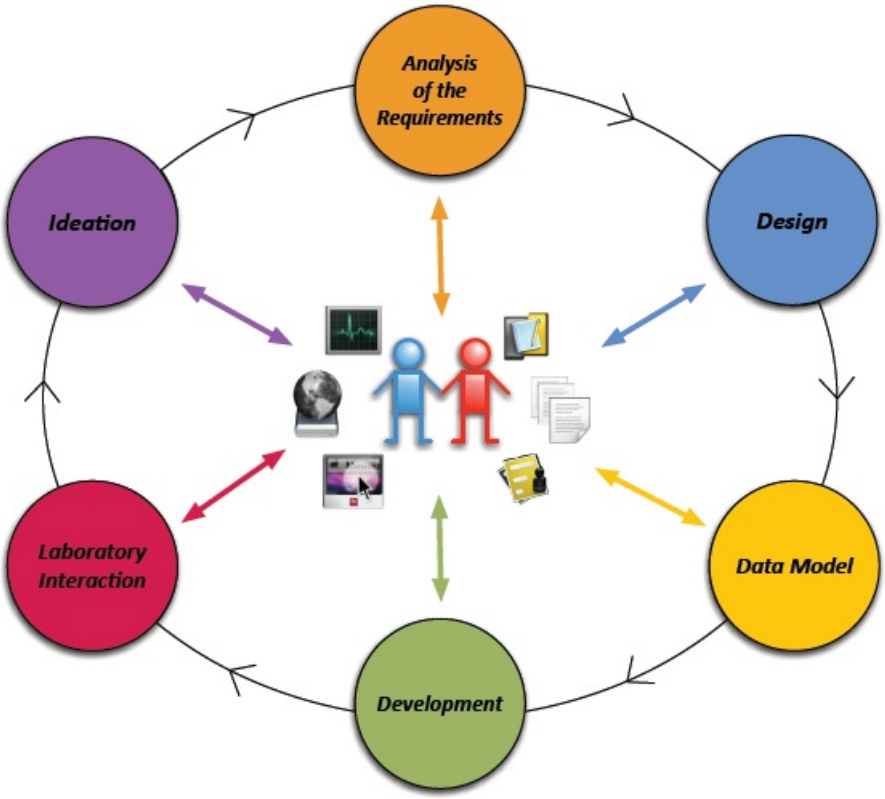
3 SIAR: Sistema Informativo Archivistico Regionale del Veneto

In questo capitolo viene introdotto il SIAR, ovvero il Sistema Informativo Archivistico Regionale del Veneto. Nelle prime due sezioni verrà descritta la prima versione del sistema SIAR, in particolare verranno presentate brevemente la struttura e il modello dei dati del SIAR. Nella terza sezione si parlerà del modello NESTOR, sul quale si basa l'architettura del SIAR.

3.1 Descrizione del progetto SIAR

Il SIAR [Agosti et al., 2011] è un progetto portato avanti dalla Regione Veneto in collaborazione con l'Università di Padova. Lo scopo fondamentale del progetto è lo sviluppo di un sistema informativo archivistico regionale. Sul territorio regionale sono presenti numerosi archivi, tra i quali gli Archivi e biblioteche musicali veneti del Novecento, Archivi del Patriarcato di Venezia, Archivi delle Aziende Sanitarie ecc., che raccolgono dati e informazioni in numerosi ambiti d'interesse per il cittadino e per la comunità. Come già detto, al progetto SIAR hanno collaborato specialisti e operatori di sistemi informatici, operatori archivisti e dei beni culturali. Lo sviluppo del SIAR è stato suddiviso in sei passi:

Figura 3.1: Progetto SIAR [Silvello, 2011]



1. Ideazione. Il primo passo è stato quello dell'analisi del problema e della definizione degli obiettivi da raggiungere. La prima attività ha compreso lo studio approfondito degli archivi, dei dati che raccolgono, delle relazioni tra i dati e delle problematiche che sorgono nella gestione degli stessi ed è stata portata avanti grazie alla stretta collaborazione tra informatici e operatori archivistici. In particolare, le relazioni tra i documenti archivistici ma anche quelle tra i documenti stessi e l'ambiente che li crea e li conserva sono di tipo gerarchico. Per questo motivo le relazioni vengono rappresentate nel sistema con delle strutture al albero. Per descrivere l'archivio si usa la descrizione archivistica, descritta nel capitolo precedente, e lo standard fondamentale adottato dalla comunità informatica è l'EAD (presentato anche questo nel capitolo precedente). Il gruppo di lavoro ha effettuato uno studio per individuare i vantaggi ed i difetti dello standard EAD, in modo da poter rappresentare correttamente le informazioni presenti in un archivio.
2. Analisi dei requisiti. Questa attività ha portato alla luce i principali requisiti che il sistema deve rispettare:
   a) mantenere la struttura gerarchica con cui sono gestite e mantenute le entità archivistiche, in modo da non andare a modificare le relazioni esistenti e quindi anche la struttura dell'archivio;
   b) estendere il formato dei metadati utilizzati nella descrizione archivistica, tenendo conto di tutti gli attori del sistema; sono previsti formati per descrivere correttamente anche le istituzione pubbliche, i produttori, i conservatori ecc.;
   c) utilizzare allo stesso tempo formati differenti di metadati;
   d) autorizzazione e sicurezza: la gestione di dati archivistici deve essere consentita solo al personale autorizzato.
3. Progettazione. L'attività di progettazione si è occupata di soddisfare i requisiti elencati al punto precedente e di come implementare il sistema definito in precedenza, utilizzando le tecnologie a disposizione e allo stato dell'arte. La definizione del formato dei metadati da utilizzare si è basata sulle indicazioni ottenute dal catalogo italiano per le risorse digitali, proposto dal SAN (Sistema Archivistico Nazionale), con lo scopo principale di garantire la compatibilità del SIAR con il SAN.
4. Modello dei dati. Il modello dei dati utilizzato nel SIAR è stato definito nel modello NESTOR (NEsted SeTs for Object hieRachies) [Ferro and Silvello, 2009]. Il NESTOR Model definisce a sua volta due modelli dati basati su insiemi annidati: il Nested Set Model (NS-M) e l' Inverse Nested Set Model (INS-M). Questi due modelli permettono di rapprensentare un insieme di dati e metadati organizzati secondo una struttura gerarchica e sono stati utilizzati per definire i metadati usati nel SIAR. La differenza sostanziale tra queste due nuove strutture e la generica struttura ad albero sta nel tipo di relazione tra un nodo e l'altro: nell'albero troviamo relazioni binarie tra i vari nodi, mentre in NESTOR viene introdotta la relazione di inclusione tra insiemi e sottoinsiemi.
5. Sviluppo. L'attività di sviluppo si è occupata dell'implementazione vera e propria del sistema SIAR, utilizzando le tecnologie dello stato dell'arte, tra cui anche il protocollo OAI-PMH. Il sistema SIAR è sviluppato come un'applicazione Web e per essere utilizzato basta disporre di un pc con un browser e una connessione ad internet. Questo lavoro di tesi si occuperà di rivedere l'interfaccia Web del sistema.
6. Verifica di laboratorio. Gli operatori archivistici testano le funzionalità del SIAR. La fase di test viene utilizzata, oltre, che per la verifica della corretta implementazione delle funzionalità con la ricerca di eventuali malfunzionamenti e problemi, anche per individuare

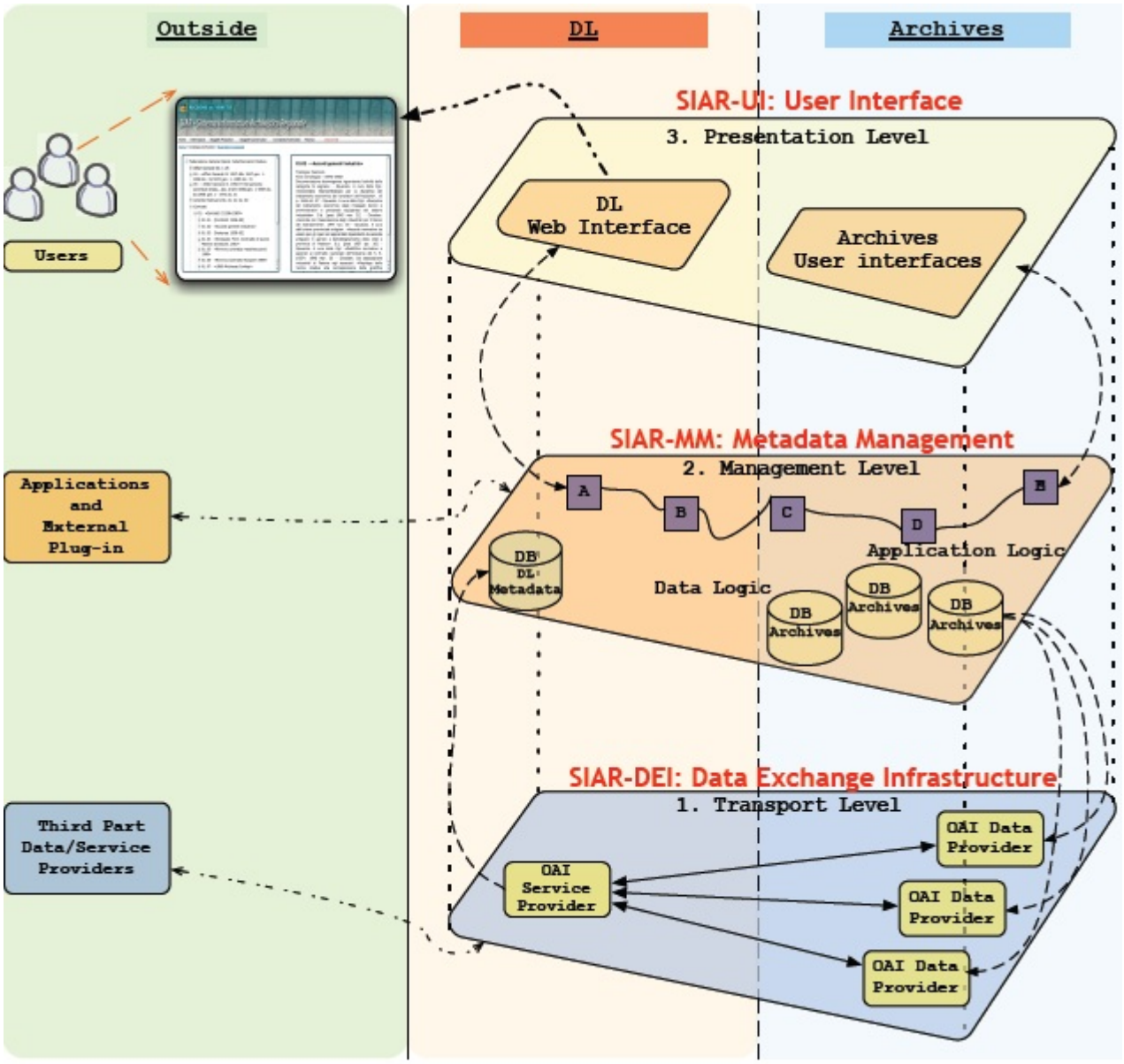
nuove attività da implementare per future possibili esigenze degli utenti archivisti del sistema.

3.2 Architettura e modello dei dati

L'architettura del primo sistema SIAR si divide in tre livelli:

1. infrastruttura di scambio dati [Agosti et al., 2011] ;
2. infrastruttura di gestione dei metadati [Ferro and Silvello, 2009, 2010] [Silvello, 2011];
3. interfaccia utente.

Figura 3.2: Architettura SIAR [Silvello, 2011]



Il livello di trasporto è gestito tramite il protocollo OAI-PMH (Open Archive Initiative Protocol for Metadata Harvesting).

Il modello funzionale basato su questo protocollo è composto da due parti:

1. data provider: questa componente si occupa di gestire uno o più repository, collezioni di oggetti e documenti digitali; le attività fondamentali sono responsabili della creazione e della memorizzazione dei metadati che descrivono e caratterizzano gli archivi stessi. Supportano il protocollo OAI per consentire l'accesso ai metadati sul contenuto. Il data provider, inoltre, mette a disposizione i metadati e ne cura la qualità e la completezza. Nel caso del SIAR, gli archivi locali dislocati sul territorio regionale sono i data provider;
2. service provider: gestiscono i servizi per l'aggregazione e l'indicizzazione dei metadati (ricerca, scoperta, localizzazione degli oggetti digitali) e interrogano gli archivi dei data provider usando le richieste del protocollo OAI per catturarne i metadati. Inoltre forniscono interfacce utente che si avvalgono della tipologia dei portali e dei Middle Ware (OAI-PMH, OpenUrl, Z39.50, ISO ILL, NCIP). Nel caso del SIAR il service provider è la Regione Veneto, che costituisce un repository centrale, contenente tutti i dati degli archivi locali e al quale possono avere accesso gli utenti autorizzati.

Il compito fondamentale di questo protocollo è quello di raccogliere i dati e i documenti nei repositories e per questo motivo è stato scelto per essere usato nel progetto SIAR; infatti il SIAR deve recuperare i documenti memorizzati negli archivi locali e deve raccoglierli nel repository centrale.

L'altra componente fondamentale del SIAR è il secondo livello, quello della creazione e gestione dei metadati. E' composto da tre parti:

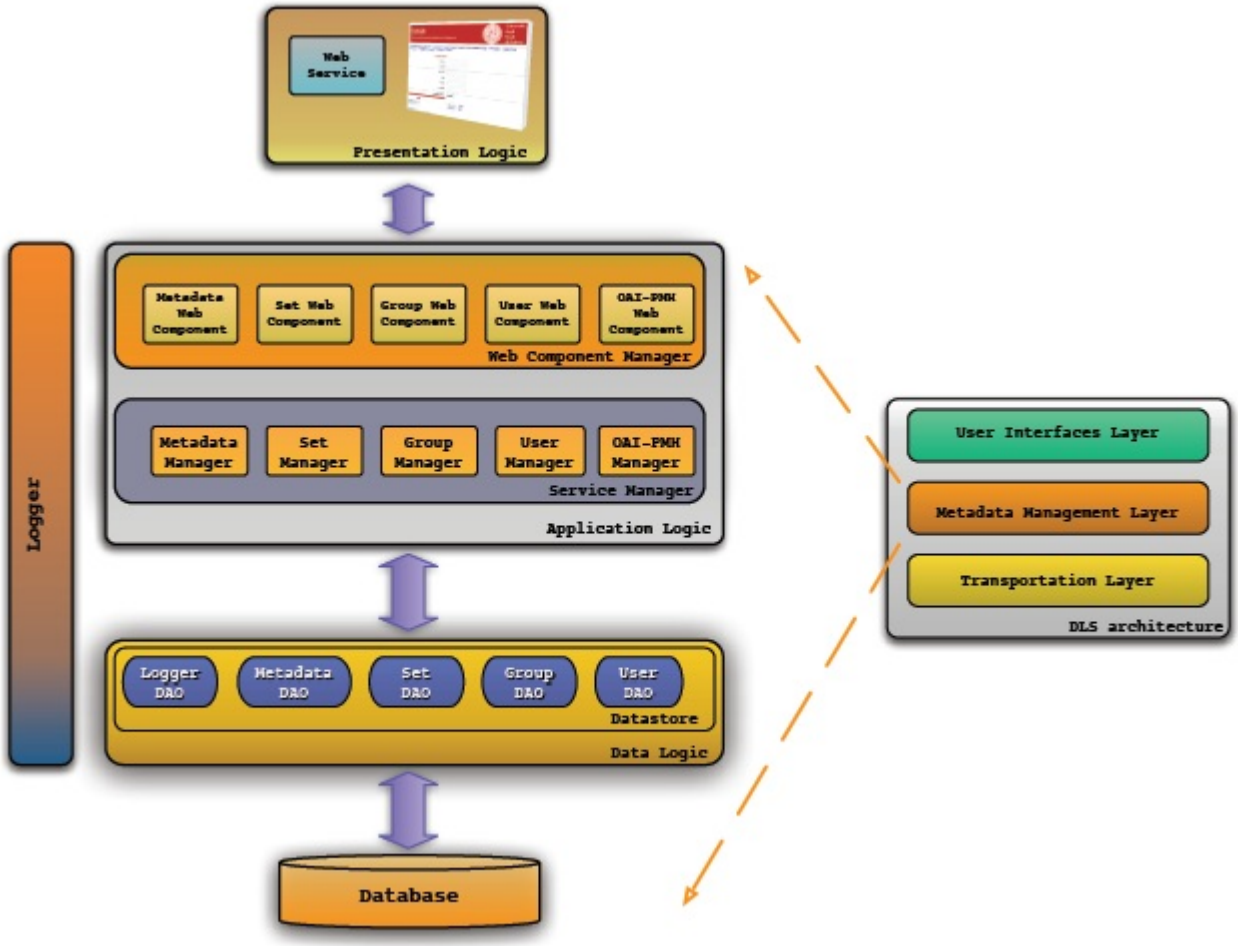
- un database relazionale;
- data logic: si compone di un datastore. Il datastore del sistema è indipendente da qualsiasi RDBMS e definisce e implementa tutti i metodi che vengono poi utilizzati dalle componenti di application logic. E' formato da diversi oggetti chiamati DAO (Data Access Objects); i DAO si inseriscono tra il database e le componenti che accedono al database stesso, permettendo di gestire le connessioni e di creare, leggere e modificare i dati memorizzati. Nel datastore del SIAR sono stati creati cinque DAO:
  - il DAO per creare e gestire i metadata;
  - il DAO per creare e gestire i set;
  - il DAO per creare e gestire gli utenti;
  - il DAO per creare e gestire i gruppi;
  - il DAO per creare e gestire i log;

  Nella figura 3.3 viene riportato il diagramma che rappresenta il livello di gestione e creazione dei metadati e le relazione tra le parti che lo compongono;
- application logic: è costituita da un service manager e da un Web component manager. In seguito viene riportato uno schema che riassume questo componente (figura 3.4).

  Il service manager definisce le funzionalità previste per il SIAR, mentre il Web component manager definisce i metodi che verranno utilizzati dai Web services esposti dal sistema. Il service manager del SIAR contiene cinque service, uno per ogni DAO del datastore, e un service di tipo OAI-PMH che implementa il protocollo descritto in precedenza, occupandosi di ricevere e gestire le chiamate esterne creando come risposta il codice XML con i metadati richiesti. La parte di autenticazione degli utenti al sistema è gestita dallo user service.

Riassumendo, il database e il datastore hanno il compito di creare e gestire le quattro entità principali del SIAR: metadata, set, user and group. Il service manager implementa i servizi del

Figura 3.3: Metadata management [Silvello, 2011]



SIAR, mentre il Web component manager implementa i metodi che permettono di interagire con i Web services esposti dal SIAR. In questo livello devono essere mantenute le relazioni di tipo gerarchico che sono presenti tra le entità del sistema, secondo le specifiche del modello NESTOR [Silvello, 2011], che verrà introdotto e descritto brevemente in seguito.

Infine, il livello di presentazione del SIAR è costituito dall'interfaccia utente. Le interfacce sono fondamentalmente due:

1. la prima è pensata per gli utenti generici come archivisti, utenti della pubblica amministrazione, ricercatori, ecc. che vanno ad utilizzare le funzionalità fondamentali del SIAR;

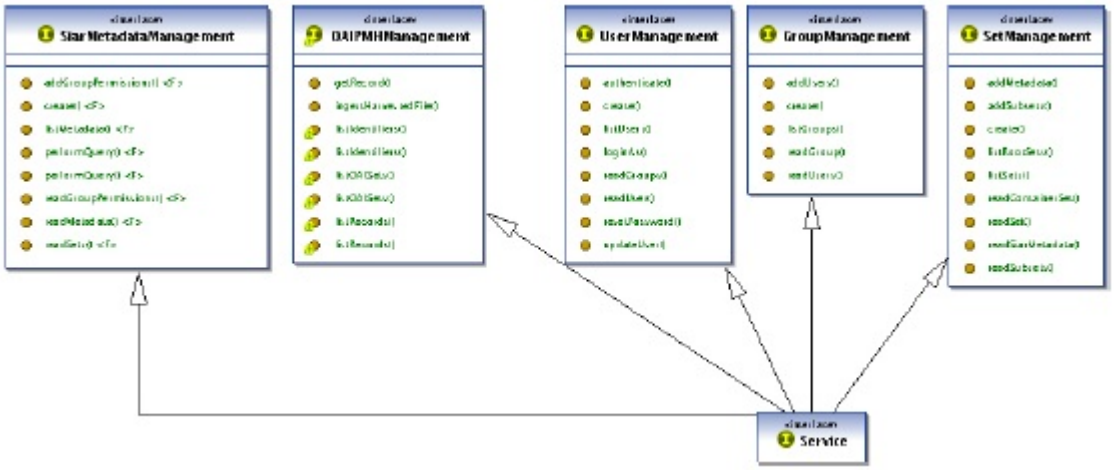
2. la seconda è dedicata agli utenti che si occupano di creare, memorizzare e gestire i metadati che descrivono l'archivio.

Per questo scopo sono stati definiti due ruoli utente:

1. utenti generici, che hanno accesso in sola lettura ai metadati dell'archivio;

2. utenti archivisti, che hanno la possibilità di creare e gestire i metadati dell'archivio.

Figura 3.4: Application logic [Silvello, 2011]



3.3 Modellazione del SIAR: il NESTOR Model

Il NESTOR Model [Ferro and Silvello, 2009, 2010, Silvello, 2011] è alla base del SIAR; infatti il sistema in esame è un'implementazione concreta dei modelli che costituiscono NESTOR.

NESTOR definisce due modelli dati complementari basati su insiemi annidati (Figura 3.5):

1. il Nested Set Model (NS-M);
2. l' Inverse Nested Set Model (INS-M).

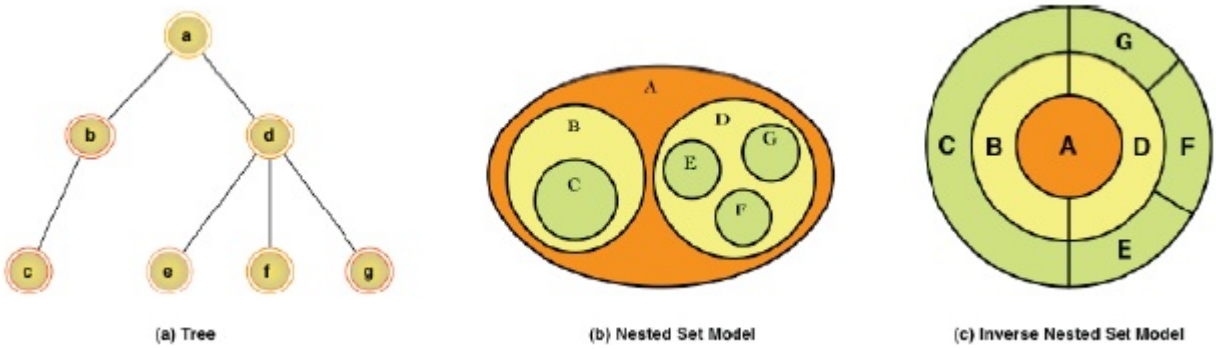
L'idea principale alla base di questo modello si basa sul concetto di organizzazione di insiemi annidati. Infatti, una struttura basata su insiemi annidati non è solamente in grado di mantenere tutte le caratteristiche e le funzionalità di una struttura gerarchica ad albero, ma permette di aggiungerne di nuove. Tra le principali possiamo distinguere la flessibiltà del modello e la possibilità di selezionare ed estrapolare rapidamente alcuni sottoinsiemi di dati, in modo che rispondano ai requisiti di interesse. La differenza sostanziale tra NESTOR e la classica struttura gerarchica ad albero è rilevabile dal tipo di relazione matematica che esprime il collegamento tra i vari elementi: nell'albero i nodi sono legati tra loro da una relazione matematica, mentre all'interno di NESTOR gli elementi sono collegati tramite la relazione di inclusione tra insiemi.

Nella figura 3.5 vengono mostrati due esempi dei modelli introdotti sopra, entrambi rappresentanti la struttura gerarchica ad albero della figura 3.5(a). Nel primo caso, cioè quello del NS-M, il nodo radice dell'albero è rappresentato tramite l'insieme principale, mentre ogni nodo foglia è costitituito da un sottoinsieme che non contiene ulteriori sottoinsiemi; la struttura gerarchica viene mantenuta grazie all'organizzazione degli insiemi e dei sottoinsiemi che rapprensetano i nodi degli alberi e alle relazioni che intercorrono tra gli stessi. Anche nel caso del secondo modello, l' INS-M, la struttura ad albero e le relative relazioni sono mantenute, ma la struttura risulta complementare a quella del primo modello: ogni nodo dell'albero viene mappato tramite un insieme, ma in questo modello ogni nodo padre diventa un sottoinsieme degli insiemi dei suoi nodi figli e il nodo radice risulta essere l'unico sottinsieme senza ulteriori sottoinsiemi.

Nella Figura 3.6, invece, viene riportato uno schema con tutte le componenti fondamentali del modello.

I modelli sopra rappresentati sono stati analizzati durante la fase di progettazione e sviluppo del SIAR, andando a verificarne la possibile applicazione anche al contesto degli archivi in modo da poter creare un sistema che superi i limiti riscontrati durante la fase di ideazione del SIAR. Al termine di quest'analisi è stato possibile affermare che il NESTOR è in grado di rappresentare

Figura 3.5: NESTOR Model [Silvello, 2011]



correttamente un archivio digitale. Ad esempio, la struttura ad albero rappresentata nella figura 2.6 nel capitolo 2, che rappresenta un possibile archivio con fondi, sottofondi, serie ecc., può essere rappresentata correttamente usando NESTOR, senza perdere le relazioni gerarchiche tra i vari componenti. Inoltre, i modelli di NESTOR permettono di utilizzare un qualsiasi formato di metadati, consentendo in questo modo agli archivisti di utilizzare quello più adatto alle loro esigenze. Infine, l'uso di NESTOR consente di ottenere una separazione ben definita tra la fase di modellazione e quella di progettazione e scelta della tecnologia da usare per la costruzione del sistema informativo archivistico.

4 Tecnologie utilizzate

In questo capitolo viene descritto e analizzato in dettaglio l'ambiente di lavoro utilizzato per la progettazione e lo sviluppo della nuova interfaccia utente del SIAR. Inizialmente vengono introdotti i concetti di portale Web e di REST Web Service. Successivamente vengono introdotte e spiegate le componenti principali: il portale Web Liferay e le Portlet. Infine, nella sezione successiva vengono descritti i linguaggi lato Client (Javascript, JQuery, JQuery UI, Alloy UI) usati durante lo sviluppo del codice.

4.1 Introduzione

4.1.1 Introduzione ai portali Web

Il termine portale Web, per quanto relativamente recente, è uno dei più utilizzati (e spesso abusati) tanto dai profani quanto dagli esperti di applicazioni Web e di Internet. Inoltre, spesso non viene fatta distinzione tra un sito Web ed un portale Web e i due termini vengono usati indifferentemente per riferire l'uno o l'altro. Un portale Web [Rich Sezov, 2012] può essere definito come un insieme di risorse Web, anche indipendenti tra loro, messe a disposizione degli utenti che usufruiscono dei contenuti e dei servizi del portale.

Le differenze sostanziali tra un sito e un portale possono essere riassunte nel seguente modo:

- un sito Web è composto da un insieme di documenti ipertestuali (pagine Web) correlati tra loro e caratterizzati da una prestabilita struttura grafica;
- un portale Web è invece composto da un insieme di elementi presenti all'interno di diverse pagine Web, la cui disposizione può essere modificata da ciascun utente e la cui struttura grafica può essere personalizzata in modo indipendente dagli altri componenti. Quindi un portale è sostanzialmente un aggregatore di informazioni, che offre un servizio di navigazione sul Web, facilitando la ricerca di contenuti di qualunque natura. Alcuni esempi di portali Web sono: portali turistici, portali di enti pubblici, ecc.

La definizione di portale Web, come riportato da Wikipedia[1] è: *"un sito Web che costituisce un punto di partenza, una porta di ingresso, ad un gruppo consistente di risorse internet o di una intranet"*. Inoltre un portale, per essere tale, dovrebbe presentare nella sua pagina principale l'accesso a strumenti quali motori di ricerca, liste di discussione e directory Web e deve prevedere una gestione dei contenuti basata su CMS (Content Management System). In ogni caso per un portale devono esssere fondamentali le capacità di convogliare informazioni da e verso l'esterno, tramite l'inserimento di news e feed da altri siti, e l'utilizzo di Web Application per fornire servizi.

Generalmente i portali sono costruiti e mantenuti utilizzando delle componenti software dinamiche chiamate Portlet [Sarin, 2012]. Una Portlet è un modulo Web, sviluppato in Java, riusabile all'interno di un portale Web. Attraverso le Portlet è possibile generare contenuti dinamici e, di conseguenza, gestirne la personalizzazione. Inoltre, tramite queste componenti è possibile integrare contenuti provenienti da diverse sorgenti.

[1] http://it.wikipedia.org/wiki/Portale_Web

4.1.2 Introduzione ai REST Web services

I Web Services per interagire con il SIAR sono stati sviluppati seguendo il paradigma REST. REST (Representational State Transfer) è uno stile architetturale per sistemi software distribuiti. La sua definizione è apparsa per la prima volta nel 2000 nella tesi di Roy Fielding "*Architectural Styles and the Design of Network-based Software Architectures*" discussa presso l'Università della California (Irvine). In questa tesi si analizzavano alcuni principi alla base di diverse architetture software, tra cui appunto i principi di un'architettura software che consentisse di vedere il Web come una piattaforma per l'elaborazione distribuita. Il concetto centrale per un sistema RESTful è quello di risorsa. Una risorsa è una qualunque entità che possa essere indirizzabile tramite il Web, cioè accessibile e trasferibile tra un client e un server. Le risorse possono essere di qualunque tipo:

- un articolo di un sito di notizie;
- uno studente di una qualche università;
- un'immagine in una Web gallery;
- ecc.

I principi fondamentali di un'architettura REST sono i seguenti:

1. client-server: il server offre uno o piu risorse e rimane in attesa di richieste. Il client, che desidera accedere ad una o più risorse, invia una richiesta che il server può scegliere di accettare o meno;
2. stateless: ogni richiesta isolata deve contenere tutte le informazioni necessarie per essere comprensibile, senza aver bisogno di riferimenti a precedenti richieste. Questo principio permette una implementazione più semplice del server e una sua migliore scalabilità, in quanto ogni risorsa può essere concretizzata dopo una richiesta. Si possono distinguere due tipi di stato:
   a) application state: rappresenta lo stato dell'applicazione client, il quale è differente per ogni possibile utente. L'application state rappresenta il percorso che il client ha seguito attraverso le risorse.
   b) resource state: rappresenta lo stato della risorsa, è uguale per tutti i client ed e mantenuto sul server finchè la risorsa non viene eliminata. Questo stato è restituito al client sotto forma di rappresentazione. Il server non mantiene l'application state.
3. caching: il risultato di una richiesta puo essere memorizzato da un client o da un intermediario (vicino al client) in modo da diminuire la latenza. Per poter essere memorizzata in cache una risposta deve comunque essere etichettata come cacheable;
4. layered: è possibile inserire dei livelli tra client e server sotto forma di componenti intermediari, i quali trasmettono i messaggi e possono offrire ulteriori servizi. Ogni livello è visibile solo dal suo immediato vicino in modo che i due livelli siano disaccoppiati tra loro e migliorare la flessibilita in caso di aggiornamenti;
5. Uniform interface: questo è il principio chiave dell'architettura REST, in quanto è ciò che la differenzia dalle altre architetture di rete. Ogni risorsa deve essere accessibile attraverso un insieme standard di operazioni che permetta al client di identificarla e di interagirvi. Il concetto base di questo principio è l'uniformità: non è rilevante quali standard siano stati scelti per accedere alle risorse ma è importante che chiunque voglia accedervi lo faccia nel medesimo modo. Questo principio è descritto da ulteriori quattro vincoli:
   a) ogni risorsa, come già detto in precedenza deve essere identicata in modo univoco (URI);

b) le risorse possono essere manipolate solo attraverso una loro rappresentazione e utilizzando metodi standard comuni a tutti i client;

c) ogni messaggio contiene tutte le informazioni necessarie per essere processato. Questo è diverso dal principio stateless sopra citato, poichè in esso si fa riferimento all'indipendenza dai soli stati precedenti;

d) il client è responsabile del mantenimento del proprio stato e può effettuare una transizione solo attraverso hyperlinks.

6. Code on demand: è un vincolo opzionale che permette di aggiungere funzionalità al client quando risulta essere necessario.

Il meccanismo per indicare quali operazioni effettuare sulle risorse è basato sul principio dell'uso esplicito dei metodi HTTP, sfruttando i metodi (o verbi) predefiniti di questo protocollo: GET, POST, PUT e DELETE. REST stabilisce quindi una mappatura uno a uno tra le tipiche operazioni CRUD (creazione, lettura, aggiornamento, eliminazione di una risorsa) e i metodi HTTP.

Tabella 4.1: Metodi REST

Metodo HTTP	**Operazione CRUD**	**Descrizione**
POST	Create	Crea una nuova risorsa
GET	Read	Ottiene una risorsa esistente
PUT	Update	Aggiorna una risorsa o ne modifica lo stato
DELETE	Delete	Elimina una risorsa

4.2 Descrizione ambiente di lavoro

4.2.1 Liferay portal

4.2.1.1 Introduzione e caratteristiche

Liferay Portal è un portale Web Open Source sviluppato in Java, che utilizza moderne tecnologie Web 2.0; è basato su un'architettura Service-Oriented (SOA), che rende possibile lo sviluppo di applicazioni a partire dai servizi Web esistenti. Viene utilizzato da aziende in tutto il mondo, tra cui Lufthansa, Telefonica O2, il Ministero della Difesa Francese, York University, ecc., e comprende una lista di funzionalità che lo mettono a confronto diretto con altri portali proprietari, con il vantaggio di non avere oneri di licenza. Liferay è sviluppato della Liferay Inc. ed stato concepito nel 2000. Esistono due versioni per questo progetto:

- Community Edition (CE), completamente gratuita (è distribuita sotto licenza GNU LGPL), destinata a sviluppatori, community ecc.;
- Enterprise Edition (EE), versione a pagamento, più stabile della precedente e destinata per usi professionali (aziende ecc.).

La Liferay Inc. è una software house che sviluppa software a pagamento ampliando e modicando software open source già esistenti. Il quartiere generale della società si trova a Los Angeles, in California (USA). Liferay è stata fondata nel 2000 da Brian Chan per creare un portale enterprise per le organizzazioni no profit. Nel 2004 la compagnia si è fusa con la compagnia tedesca Liferay GmbH sotto l'unico nome Liferay Inc.

I vantaggi principali di questo portale, che risultano essere funzionali per lo sviluppo e la progettazione della nuova interfaccia grafica del SIAR, sono:

- compatibilità con i principali Server Container, come ad esempio Apache Tomcat[2], Glassfish[3], JBoss[4], Oracle Weblogic[5], WebSphere[6];
- supporta i principali database, tra i quali MySQL, Oracle, SQL Server, PostgresSQL, JDataStore, Sybase, SAP, Apache Derby, Firebird, Hypersonic;
- è multipiattaforma, in quanto sono disponibili le versioni compatibili con i sistemi Linux, Unix, Windows e Mac OS;
- utilizza l'ultima versione di Java J2EE, la Java Platform Enterprise Edition 6;
- è compatibile con le specifiche JSR-168 e JSR-286 descritte nel capitolo precedente;
- fornisce il supporto ad AJAX[7];
- incorpora la libreria Javascript Alloy UI, descritta nella sezione 4.3.2.6;
- fornisce un gran numero di Portlet pronte all'uso che possono anche essere personalizzate;
- supporta il drag and drop delle Portlet, per favorire la personalizzazione delle pagine Web del portale;
- ricerca di risorse: possibilità di etichettare i documenti, i contenuti Web, i messaggi dei forum ed altri elementi in modo dinamico, condividendo i tag con gli altri utenti del portale. Gli utenti possono quindi ricercare tramite le etichette assegnate alle informazioni o gruppi di informazioni comuni;
- creazione di pagine personali: gli utenti abilitati possono avere uno spazio personale dove aggiungere proprie informazioni e decidere se renderle pubbliche o tenerle private. È possibile personalizzare lo spazio messo a disposizione tramite il drag & drop delle Portlet;
- supporto multilingua;
- supporto a LDAP;
- supporto all'autenticazione Single Sign On (SSO): il portale consente agli utenti di accedere ai contenuti e applicazioni da un unico punto di accesso. Liferay può infatti aggregare diversi sistemi applicativi rendendoli disponibili accedendo una volta sola con il massimo della riservatezza;
- sistema di autorizzazioni basato sul ruolo di un utente: per garantire che le persone accedano con diritto alle sole informazioni/dati per le quali sono autorizzate, gli amministratori del portale possono assegnare ai singoli utenti o gruppi di utenti diversi ruoli per attribuire loro differenti livelli di accesso e differenti diritti di modifica;

[2]http://tomcat.apache.org/
[3]http://glassfish.Java.net/
[4]http://www.jboss.org/overview/
[5]http://www.oracle.com/technetwork/middleware/Weblogic/overview/index.HTML
[6]http://www-01.ibm.com/software/it/Websphere/
[7]http://www.w3.org/TR/XMLHttpRequest/

- gestione centralizzata di tutti i contenuti, risorse, utenti, comunità, ruoli attraverso un pannello di controllo personalizzabile con la possibilità di aggiungere o togliere alcune sue parti;
- configurazione dell'interfaccia del portale: Liferay dispone di un'interfaccia utente intuitiva e semplice da utilizzare per tutti i membri di un'organizzazione.
- sistema di gestione dei contenuti: il CMS incluso all'interno di Liferay fornisce un insieme di funzionalità integrate con le funzioni di collaborazione e fornisce un repository centralizzato per conservare e gestire contenuti da visualizzare sul Web;
- Web Publishing, che può essere usato per creare pagine Web in modo veloce ricorrendo a dei contenuti riutilizzabili, anche grazie a dei modelli (template) per layout flessibili;
- Flexibile Template Mechanism (XSL/VM): i modelli creati per i Journal articles (così vengono chiamati i contenuti Web) possono essere realizzati in XSL o Velocity (VM) offrendo così agli sviluppatori la flessibilità di disegnare le pagine Web; Document e Image Library (anche se nella versione più recente di Liferay queste due vengono unite in un'unica Library), che provvede un deposito centralizzato per i servizi della libreria, basato sulla specifica Java Content Repository (standard JSR-170) per trattare diversi tipi di documenti (PDF, DOC, ecc.) che possono essere salvati sotto un unico URL;
- Portal Publishing & Staging: la funzione di publishing permette di modificare pagine Web in tempo reale pubblicando i nuovi contenuti a seconda delle esigenze dell'utente; lo staging, invece, consente di creare varie copie della pagina modificata, in modo da avere uno storico.

4.2.1.2 Funzioni di collaborazione (Collaboration Suite)

Liferay offre anche una serie di funzioni di collaborazione, cioè le funzionalità tipiche del Web 2.0 (feed RSS, sistemi di invio email, instant messaging, creazione gestione sondaggi, wiki, blogs. . .) che sono diventate caratteristiche necessarie anche dei moderni portali applicativi. Infatti, un portale, oltre a costituire uno strumento di presentazione di contenuti e di integrazione di applicazioni Web, può anche costituire un ottimo strumento di collaborazione tra utenti. La collaborazione può essere intesa in vari modi: interazione tra i membri di un team di progetto, tra i dipendenti di una azienda o tra cittadini e amministrazioni pubbliche; nel caso del SIAR la collaborazione può avvenire tra gruppi diversi di archivisti o di addetti alla gestione degli archivi in generale. L'insieme di questi strumenti permettono un'interazione e un coinvolgimento degli stessi utenti nella creazione e nella classificazione dei contenuti, un'interazione reciproca o con altre istituzioni.

Le principali funzionalità di questa suite sono riassunte brevemente di seguito:

- wiki: Liferay implementa un sistema di wiki;
- bacheca elettronica: un sistema di bacheca elettronica (''message boards") permette di condividere idee ed annunci all'interno di un gruppo. Liferay mette a disposizione dei report sull'attività svolta nella bacheca, riportando i post recenti, gli utenti attivi. Ogni thread è visibile via feed RSS ed ogni post può inviare una mail di avviso che permette di rispondere al post da client di posta;
- blog: Liferay implementa un sistema di blog. Tra le funzionalità sono disponibili l'editor WYSIWYG, social bookmarking, notifiche email per i contributi e commenti, sistemi di rating dei contributi, sottoscrizione via RSS, scheduling delle pubblicazioni;
- instant messaging: sono comprese delle funzioni di instant messaging;

- calendari: è possibile impostare ed utilizzare calendari di gruppo (basati sulle Community);
- avvisi: è possibile inviare messaggi di tipo broadcast a gruppi di utenti. Ogni utente può settare le modalità di ricezione degli avvisi tramite Web alert via portale, SMS, email o altre modalità di delivery impostate dall'amministratore;
- sondaggi: sono disponibili delle Portlet per l'impostazione, la presentazione e la raccolta dati di sondaggi;
- tracking delle attività: la Portlet "Attività Recenti" tiene traccia delle attività più recenti effettuate sul portale;
- RSS: Liferay consente di condividere tutte le tipologie di contenuto tramite feeding RSS.

4.2.1.3 Architettura di Liferay

SOA

Come detto in precedenza, Liferay ha un struttura che si basa sull'architettura SOA [Piraccini and Rossini, 2005]. SOA è un modello architetturale per la creazione e la progettazione di sistemi software residenti su reti di computer, che focalizza l'attenzione sul concetto di servizio. Un sistema costruito seguendo la filosofia SOA è costituito da applicazioni, chiamate servizi, che possono risiedere su locazioni diverse all'interno di una rete. I servizi possono anche essere visti come funzionalità di business realizzate tramite un componente che implementa un'interfaccia. Ogni servizio mette a disposizione una delle funzionalità e può utilizzare quelle rese accessibili da altri servizi andando a costruire, seguendo questo pattern, applicazioni di complessità maggiore.

SOA è costituito da principi e linee guida architetturali indipendenti da qualsiasi tecnologia e definisce una serie di proprietà che i servizi devono soddisfare per essere riutilizzabili e facilmente integrabili in ambiente eterogeneo:

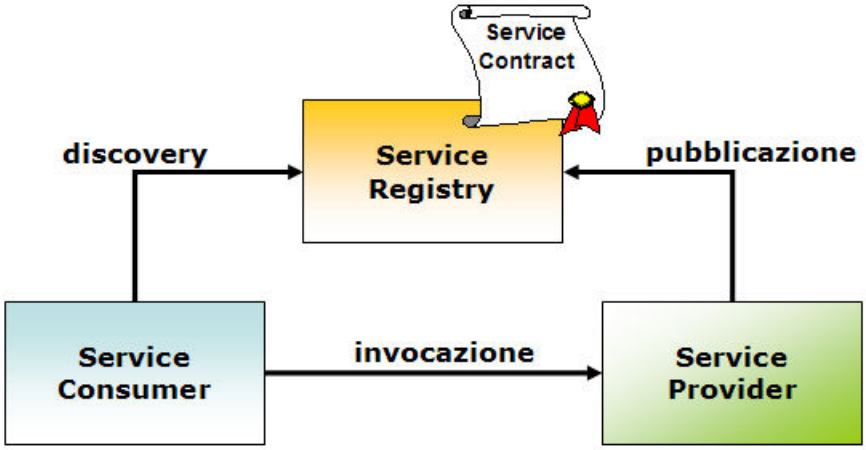
- i servizi devono essere ricercabili e recuperabili dinamicamente;
- i servizi devono essere modulari;
- i servizi devono definire delle interfacce esplicite e indipendenti dall'implementazione;
- i servizi devono avere un'interfaccia distribuita e devono essere accessibili in maniera trasparente;
- i servizi devono essere componibili, ovvero utilizzabili in processi di business complessi

Per ottenere questi requisiti, le applicazioni SOA definiscono dei ruoli:

- Service Consumer: l'entità che richiede il servizio; può essere un'applicazione o un altro servizio.
- Service Provider: l'entità che fornisce il servizio e che ne espone l'interfaccia.
- Service Contract: l'entità che definisce il formato per la richiesta di un servizio e della relativa risposta.
- Service Registry: Direttorio in rete dei servizi consultabili.

I servizi devono essere in grado di comunicare tra di loro attraverso un canale di comunicazione: il SOA Service bus. Da un punto di vista architetturale, il SOA bus è un layer che mette a disposizione uno strato di comunicazione tra i servizi. Lo scopo dell'Enterprise Service Bus (ESB) è fornire un'infrastruttura che centralizzi funzionalità quali supporto alla comunicazione

Figura 4.1: I ruoli SOA [Piraccini and Rossini, 2005]



sincrona ed asincrona basata su messaggi, intelligent routing, supporto alla trasformazione dei dati, ecc.

Architettura

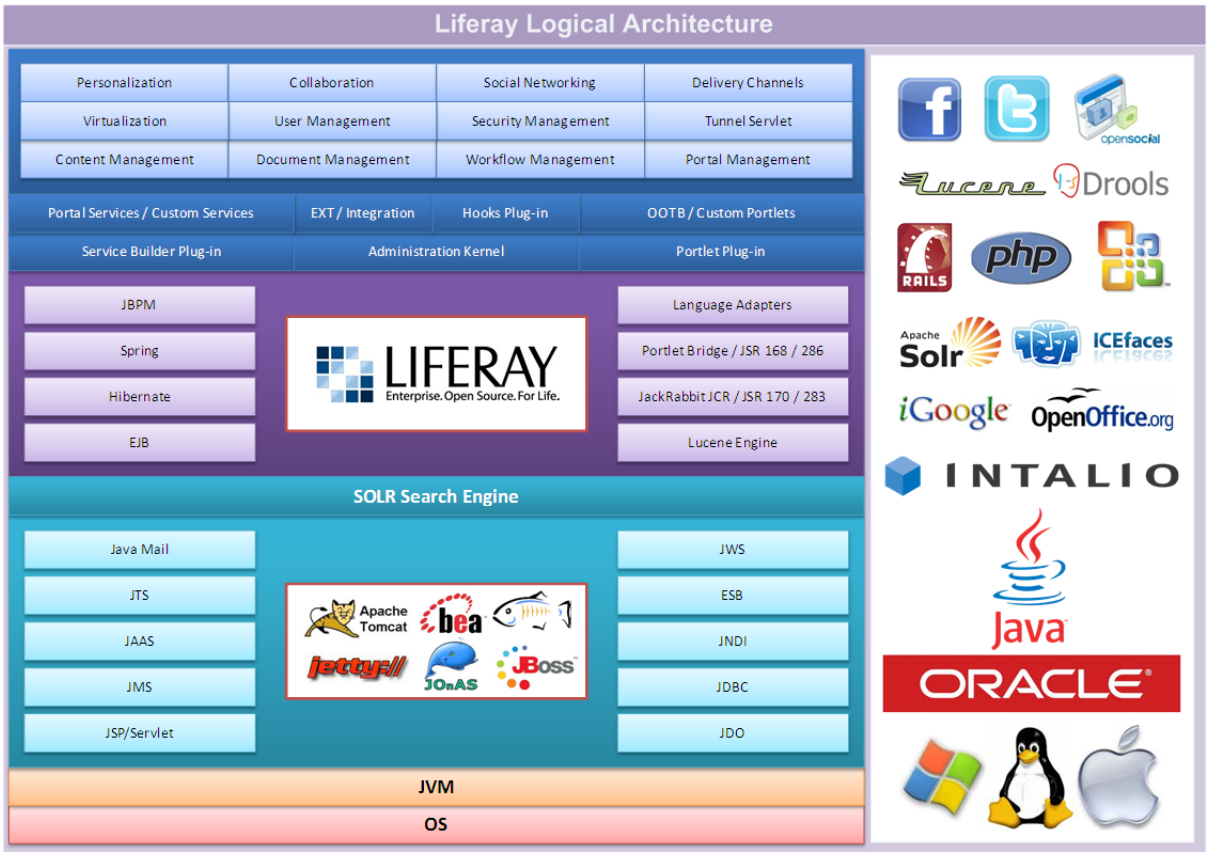
Liferay, come detto in precedenza, supporta i principali sistemi operativi: Windows, Linux e Mac OS. Appena sopra il sistema operativo si trova la JRE che mette a disposizione la Java Virtual Machine su cui viene eseguito il portale. L'istanza di Liferay viene eseguita su uno dei numerosi Application Server supportati (elencati sopra) e sul sito ufficiale del progetto sono disponibili delle versioni bundled, cioè versioni in cui il portale è precompilato e preconfigurato su un'applicazion server. Il server fornisce la connettività e l'interoperabilità usando un Enterprise Service Bus (ESB) e rende disponibili molti dei servizi che vengono utilizzati da Liferay, tra cui: JNDI, JDBC, JTS, JMS, JAAS, JDO, JWS, JSP / Servlet, e JavaMail. Attraverso questo ESB il portale può essere integrato con altre applicazioni per il Business Process Management (BPM) (usate per ottimizzare e monitorare i processi aziendali), repository JCR (Content Repository API for Java) ed altri. Inoltre vengono utilizzati Hibernate e i driver JDBC per la connessione con uno qualsiasi dei database supportati. Il Portlet Brideg fornisce il supporto per gli standard JSR 168/286 delle Portlet. Liferay usa il framework Spring per gli strati dei dati di business e dei servizi e per la gestione delle transazioni. Spring è un framework che ha come obiettivo principale quello di gestire la complessità nello sviluppo di applicazioni enterprise in ambiente Java. Le caratteristiche fondamentali di Spring sono due:

1. il pattern Inversion of Control (IoC) che serve a minimizzare le dipendenze fra gli oggetti, facilitando il riutilizzo delle componenti e rendendo più modulare lo sviluppo. Il concetto alla base dell'IoC è che le dipendenze degli oggetti devono essere risolte da un' infrastruttura esterna che ha la responsabilità di istanziare gli oggetti e di fornire loro le relative dipendenze. Quindi, il ciclo di vita degli oggetti è gestito da un'entità esterna (Container). Una tecnica usata per attuare l'IoC è la Dependency Injection (DI), cioè il Container si fa carico di "iniettare" le dipendenze nell'istanza di un oggetto automaticamente e a runtime; per fare questo, utilizza costruttori e metodi setter.

2. l'Aspect Oriented Programming (AOP), un paradigma di programmazione che mira a disaccoppiare tutte le caratteristiche di un sistema che sono logicamente indipendenti dal sistema stesso.

Nella figura 4.2 viene riportato uno schema dell'architettura di Liferay, con tutte le componenti che lo costituiscono.

Figura 4.2: Architettura di Liferay [rif. *www.liferay.com*]



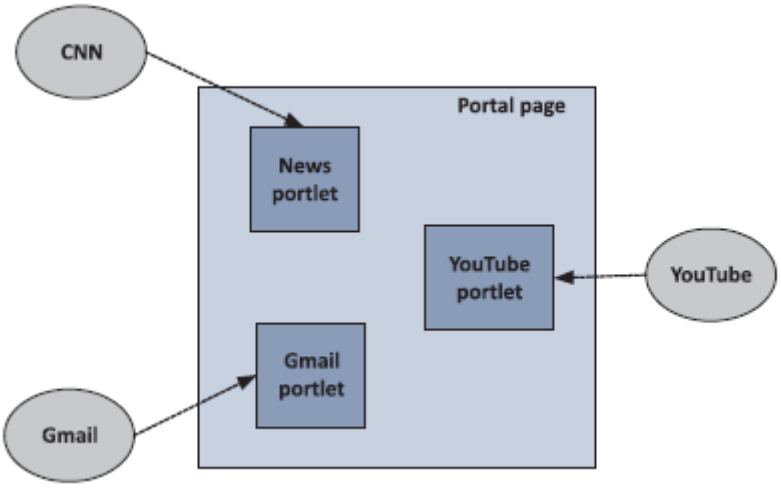
4.2.2 Portlet

4.2.2.1 Introduzione

Per descrivere in maniera corretta le Portlet è necessario introdurre brevemente anche il concetto di Servlet. Le pagine Web generate in modo dinamico possono contenere al loro interno codice sorgente che utilizza sia il linguaggio HTML, sia il linguaggio Java. Il difetto principale deriva dall'unione, nella stessa struttura, di due tipologie diverse di linguaggio: il linguaggio HTML, dedicato alla visualizzazione delle informazioni, e il linguaggio Java, usato per l'implementazione della logica applicatica, rendendo complicate le operazioni di manutenzione e riducendo la leggibilità del codice stesso. Per risolvere questo problema sono state sviluppate le Servlet: in esse la maggior parte del codice Java viene spostato in opportune classi Java, separando così il codice dedicato alla rappresentazione dei contenuti Web da quello relativo alle operazioni di gestione. Una Servlet è quindi una componente Web gestita da un contenitore e scritto in linguaggio Java, che genera pagine Web dinamiche. Le Portlet sono molto simili alle Servlet, con le quali condividono il modello di funzionamento di base: a fronte di una Request effettuata dal client (il browser), le Portlet eseguono alcune operazioni di base volte a modificare il loro stato, interagire con gli strati applicativi sottostanti e fornire una risposta coerente con la configurazione complessiva del portale. La differenza principale tra una Servlet e una Portlet è legata al contesto di esecuzione: le Servlet lavorano secondo un modello uno a uno con il Client (una Request viene gestita da una sola Servlet) e, quindi, rappresentano l'unico punto di ingresso allo strato Server da parte del Client. Nel caso delle Portlet, quando viene effettuata una richiesta, si possono eseguire contemporaneamente più Portlet ed ottenere una sola pagina di risposta, costruita opportunamente. Le Servlet sono gestite da un Servlet Container. Il Container si occupa della compilazione e dell'esecuzione delle classi di cui è composta Servlet:

quando il Server riceve una richiesta da parte di una pagina Web, il Servlet Engine esegue un'analisi sintattica della pagina e, quando rileva il codice Java, lo interpreta sostituendolo con codice HTML; successivamente la pagina viene restituita al Client, concantenando i vari frammenti di codice HTML prodotti.

Figura 4.3: Schema di una pagina Web di un portale [Sarin, 2012]



Come si può vedere in Figura 3.8, una pagina di un portale può essere costituita da più Portlet, che recuperano i dati e le informazioni da sorgenti diverse.

Le Portlet sono state definite inizialmente dalle specifice JSR-168 (Java Specification Request) come: "*un componente Web basato sulla tecnologia Java, gestito da un Portlet Container che processa richieste da parte dell'utente e genera contenuti dinamici. Le Portlet sono usate dai portali come componenti d'interfaccia utente pluggabili al fine di provvedere a un livello di presentazione per i sistemi informativi*" (rif. *http://jcp.org/en/jsr/overview*). Sono state rilasciate nel 2003 da un gruppo di lavoro a cui hanno partecipato Apache, BEA, Broadvision, IBM, Oracle, SAP, SUN, Sybase, TIBCO. Un'implementazione molto diffusa della specifica JSR168 è Apache Pluto; altre implementazioni sono state sviluppate da IBM, Oracle e BEA Systems. Fra le implementazioni Open Source di portali conformi allo standard ci sono Jetspeed 2 Portal Server (ancora sviluppato da Apache), JBoss Portal, Liferay Portal e Stringbeans Portal.

Le Java Portlet Specification v1.0, introdotte con le JSR-168, mettono a disposizione delle Portlet API, progettate per garantire interoperabilità tra le Portlet e i Portlet Containers, e in particolare definiscono alcune caratteristiche di queste componenti (che verranno descritte nel prossimo capitolo):

- le Portlet hanno un sistema di gestione delle richieste, comprendente le Action Requests e le Render Requests, per gestire le due fasi fondamentale della richiesta stessa: fase di Action Processing (richiesta d'azione) e fase di Rendering (visualizzazione dei risultati di un'azione invocata).
- Portlet Modes: il portale avvisa la Portlet in base alle azioni che deve processare e ai contenuti che dovrebbe generare;
- Window States, che indica la quantità di spazio che deve essere assegnata alla Portlet, in base al volume di dati che deve processare e visualizzare;
- Portlet Data Model, che permette alla Portlet di memorizzare le informazioni e i parametri necessari alla fase di renderizzazione dei risultati e di salvare i dati relativi alla sessione della Portlet e i dati relativi agli utenti preferenze delle portlet;

- lo sviluppo di un portale è un modo per integrare differenti applicazioni Web.

Successivamente, sono state scritte le JSR-286, conosciute anche come Java Portlet Specification v2.0. Sono state sviluppate allo scopo di migliorare le specifiche della versione precedente, andando a sopperire a lacune o a funzionalità incomplete. Alcune delle nuove funzionalità e caratteristiche includono le seguenti:

- Inter-Portlet Communication: comunicazione e scambio di dati tra le instanze di Portlet diverse, attraverso eventi (Portlet Events) e scambio di parametri pubblici di renderizzazione (Public Render Parameters). L'importanza di questo concetto sta nel fatto che le Portlet diventano riutilizzabili in scenari differenti e rende possibile la comunicazione ad una particolare Portlet di dati e informazioni che possono derivare dall'interazione con l'utente, attraverso una Portlet diversa.
- I contenuti generati dinamicamente sono resi disponibili direttamente alle Portlet.
- Introduzione del supporto ad AJAX[8] e allo scambio dati attraverso il formato JSON[9] nelle Portlet.
- Introduzione dei Filters e dei Listeners. In particolare, un filtro è un componente riutilizzabile che permette di sviluppare una logica di processo applicabile a più Portlet, andanto a preprocessare le richieste che giungono alle Portlet e a postprocessare le risposte alle chiamate. Un Listener, invece, è un componente presente all'interno di un Web Container che si occupa di rimanere in ascolto degli eventi che indicano quando il contesto dell'applicazione viene inizializzato, quando un attributo viene aggiunto o rimosso dalla sessione, ecc..

La nuova interfaccia utente del SIAR si baserà proprio sullo sviluppo di queste componenti software e quindi sullo sviluppo di un portale Web; nel prossimo capitolo verranno spiegati i motivi di questa scelta e i vantaggi che derivano dall'utilizzare un framework di questo tipo.

Le Portlet generano dei frammenti di codice markup (es. HTML, XHTM) che, integrati ed aggregati secondo determinati criteri e modi, formano la pagina Web completa. Il rapporto tra utente e Portlet attraverso il browser è implementato tramite il paradigma di scambio di richieste e risposte dei portali. Le Portlet possono generare contenuti diversi a seconda dell'utente del portale che le utilizza. Le Portlet, in modo analogo a quanto succede per le Servlet (introdotte nella sezione 3.4.1), delle quali sono un estensione, vengono gestite da un Portlet Container. Le caratteristiche principali che hanno sia le Portlet che le Servlet sono le seguenti:

- sono componenti Web basate su tecnologia Java;
- sono gestiste da un contenitore specializzato;
- generano contenuti dinamici;
- interagiscono con il client tramite un paradigma di richieste e risposte.

Analizzando invece le differenze principali con le Servlet, si rilevano che:

- le Portlet generano soltanto frammenti di markup, non pagine Web complete; successivamente il portale si occuperà di gestire l'aggregazione dei vari frammenti di markup;
- i client Web interagiscono con le Portlet attraverso il portale e non direttamente;
- più Portlet possono coesistere nella stessa pagina di un portale;

[8] http://www.w3.org/TR/2006/WD-XMLHttpRequest-20060405/
[9] http://www.json.org/

- le Portlet hanno un sistema di gestione delle richieste più raffinato delle Servlet, comprendente le ActionRequests e RenderRequests;
- le Portlet hanno delle modalità predefinite (view, edit, help mode) e degli stati delle finestre che ne definiscono la dimensione (minima, massima, normale);
- le Portlet possiedono funzioni atte ad avere una gestione degli URL indipendente dal portale;
- le Portlet non possono specificare le intestazioni HTTP delle risposte;
- le Servlet non sono appropriate per gestire l'aggregazione di contenuti, in quanto questo compito rimane in carico alla Servlet stessa, mentre per le Portlet esso è delegato al Portlet Container;
- le Servlet non hanno il supporto per la inter-comunicazione tra le stesse, per fare ciò è necessario invocare le Servlet con le API che mettono a disposizione. Nel caso delle Portlet, invece, è possibile abilitare la comunicazione tra Portlet usando gli eventi.

Il Portlet Container fornisce l'ambiente di runtime richiesto per la loro esecuzione e ne gestisce l'intero ciclo di vita, oltre a fornire uno spazio persistente per memorizzare le preferenze relative ad esse. Il contenitore si occupa si ricevere le richieste dal portale e di reindirizzarle alle Portlet opportune. Esempio di interazione tra un client e un portale è riportato nei passi seguenti.

- Un client, dopo aver fatto o meno l'autenticazione al portale, effettua una richiesta HTTP al portale.
- La richiesta viene ricevuta dal portale.
- Il portale determina se la richiesta contiene un'azione mirata per ciascuna delle Portlet associate alla pagina del portale.
- Se esiste una Portlet per quell'azione richiesta, il portale inoltra la domanda al Portlet Container che invocherà una chiamata ad essa.
- Il portale invoca le Portlet, tramite il contenitore, al fine di ottenere dei frammenti di markup che verranno inclusi nella pagina risultante.
- Il portale aggrega i vari markup prodotti dalle Portlet e invia al browser la pagina di risposta.

Per capire meglio com'è strutturata una pagina di un portale, nella figura 4.4 viene riportato un esempio. Ogni Portlet possiede un titolo e dei bottoni di controllo.

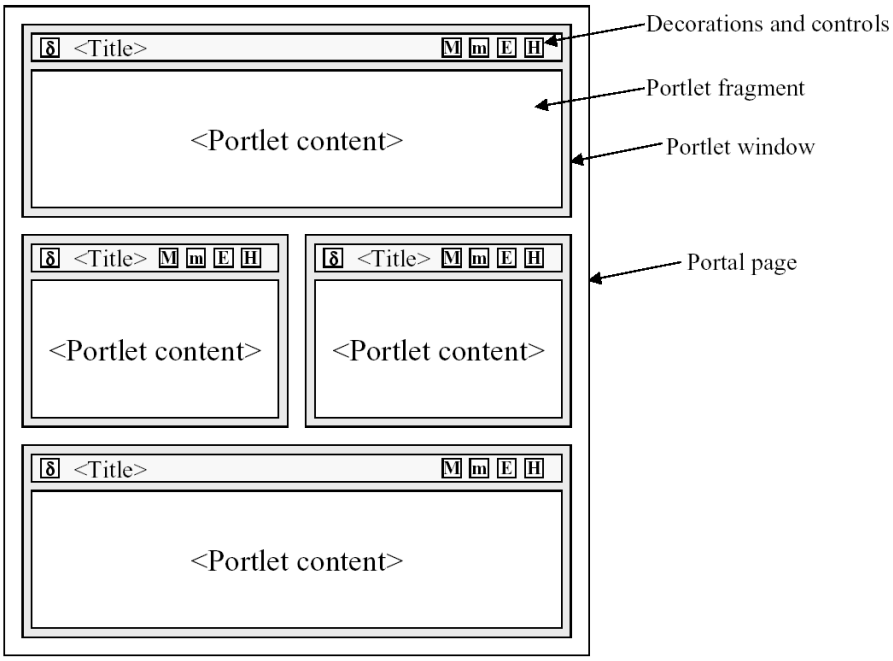
Nella figura 4.5, invece, un esempio di una pagina Web di Liferay:

4.2.2.2 Struttura Portlet

L'oggetto principale che caratterizza le specifiche è l'interfaccia `Portlet,` la quale è la principale astrazione delle Portlet API [Sarin, 2012]. Tutte le Portlet implementano questa interfaccia direttamente oppure estendendo una classe che la implementa. Le Portlet API includono la classe `GenericPortlet` che implementa l'interfaccia appena descritta e mette a disposizione le funzionalità di default. Una pratica comune tra gli sviluppatori è quella di estendere direttamente o indirettamente la classe `GenericPortlet` per l'implementazione delle proprie Portlet. Il ciclo di vita della Portlet è gestito principalmente da quattro metodi che vengono chiamati direttamente dal Portlet Container:

1. `init()`: è invocato quando la Portlet viene instanziata dal contenitore al fine di contenere la logica che preparerà l'oggetto a gestire le richieste.

Figura 4.4: Struttura tipo di una pagina Web in un portale. [rif. *www.jboss.com*]



2. `destroy()`: quando il contenitore deve occuparsi di eliminare una Portlet al fine di far pulizia se l'oggetto non viene più utilizzato o se il server viene spento.
3. `processAction()`: viene invocato dopo che l'utente ha effettuato una richiesta e serve a processare i dati dell'utente o ad effettuare altre operazioni di elaborazione.
4. `render()`: viene invocato quando si rende necessario renderizzare il contenuto della Portlet e quindi l'output dell'eleborazione del metodo precedente.

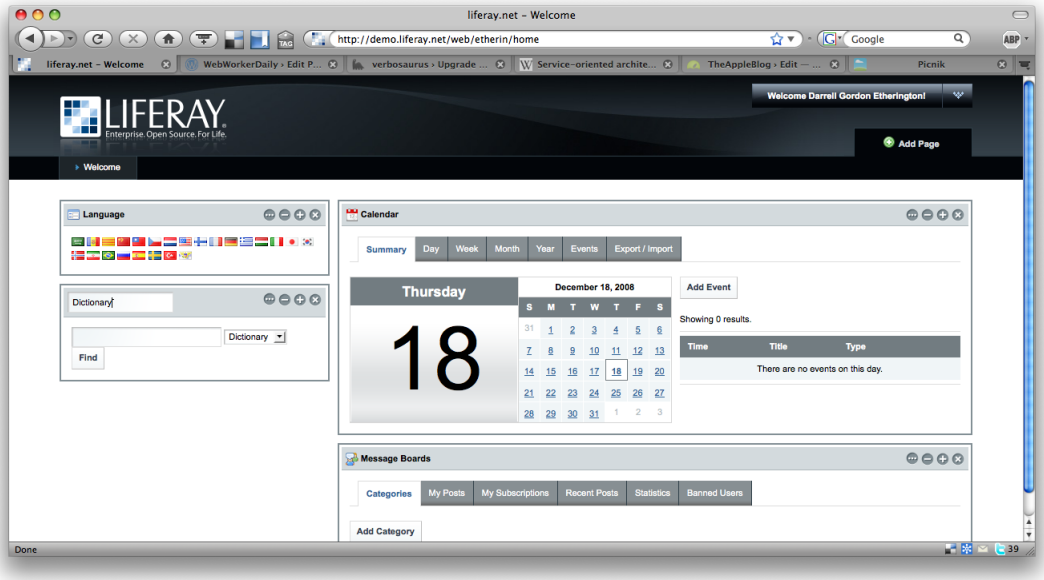
La `GenericPortlet` è dotata, in aggiunta ai metodi descritti sopra, di altre implementazioni specifiche del metodo `render()`, che lo specializzano ulteriormente a seconda della modalità di utilizzo della Portlet:

- `doView()`: viene usato per renderizzare la Portlet quando si trova in View Mode, cioè la modalità d'utilizzo in cui l'utente interagisce con essa.
- `doEdit()`: viene richiamato per renderizzare la Portlet quando si trova in Edit Mode, cioè la modalità in cui è possibile specificare le opzioni di personalizzazione e di configurazione della Portlet.
- `doHelp()`: viene utilizzato per renderizzare la Portlet quando si trova in Help Mode, per visualizzare la pagina relativa alla guida d'utilizzo della Portlet.

Il Portlet Container deve gestire per ogni Portlet il Portlet Mode ed il Window State. Il Window State indica la quantità di spazio all'interno della pagina del portale che può essere assegnato per una Portlet. Gli stati possibili sono: minimized, maximized o normal. La Portlet potrà decidere di utilizzare questa informazione per decidere quante informazioni renderizzare. La specifica JSR 168 definisce tre modalità standard di funzionamento delle Portlet:

- View: lo scopo di questa modalità è quello di generare il codice HTML che presenta all'utente lo stato corrente della stessa. Per esempio, una Portlet in view mode potrà includere un'interfaccia con cui l'utente potrà navigare ed interagire, oppure includere contenuti statici che non richiedono alcuna interazione con l'utente. Gli sviluppatori di Portlet dovranno implementare la modalità view sovrascrivendo il metodo `doView()` della classe GenericPortlet.

Figura 4.5: Pagina Web di Liferay [rif. *www.liferay.com*]



- Edit: con la modalità edit, la Portlet deve presentare i contenuti e la logica che permettono all'utente di personalizzarne il comportamento. In questa modalità vengono tipicamente settate le PortletPreferences. Gli sviluppatori di Portlet a seconda delle proprie necessità possono implementare la funzionalità edit sovrascrivendo il metodo doEdit della classe GenericPortlet.
- Help: l'help mode permette di avere informazioni di aiuto sulla Portlet. Gli sviluppatori possono implementare le funzionalità dell'help mode sovrascrivendo il metodo doHelp della classe GenericPortlet.
- Custom Portlet Modes: è possibile definire delle modalità di funzionamento personalizzate sulla base delle funzionalità specifiche. Per fare questo è necessario definire la nuova modalità attraverso il tag `customPortlet-mode` nel deployment descriptors

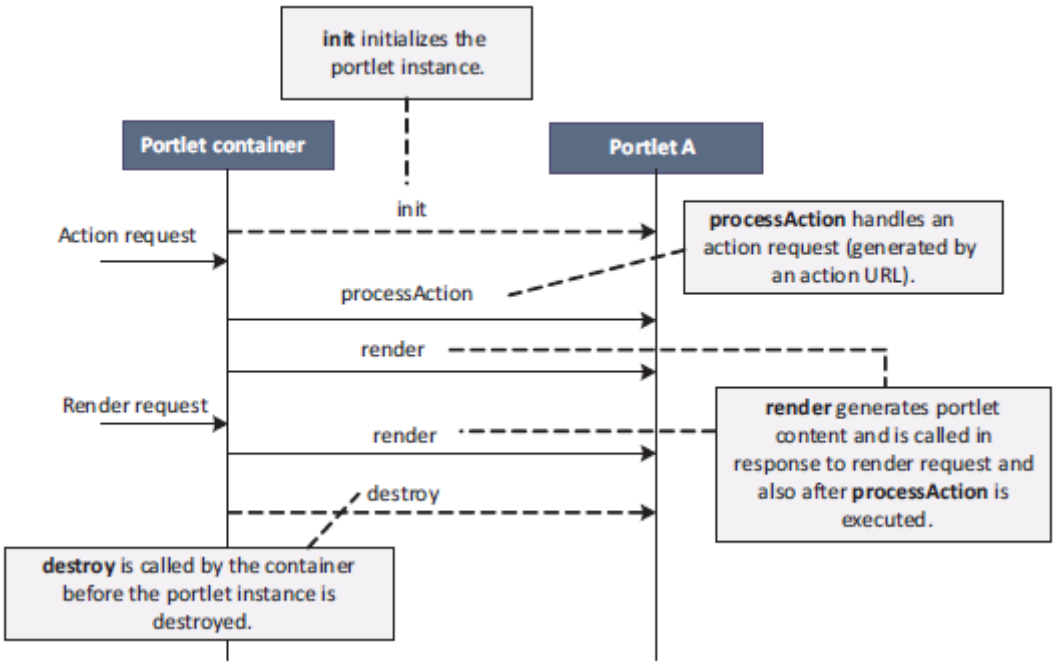
La disponibilità di queste tre modalità può cambiare a seconda del ruolo posseduto dall'utente del portale. Per esempio, potrebbe essere permesso solamente all'amministratore di accedere in edit mode, mentre per i normali utenti è possibile rendere disponibile solamente le modalità view ed help.

4.2.2.3 Portlet URL

Molto spesso all'interno di una Portlet si ha la necessità di creare dei collegamenti ipertestuali che facciano riferimento ad altre Portlet, come per esempio all'invio di un form di dati. Per garantire la completa portabilità delle Portlet non è possibile creare dei link a specifici URL (Uniform Resource Locator): per risolvere questo problema si ricorre perciò alle PortletURL [Sarin, 2012], oggetti generati dal portale al momento di una richiesta diretta verso la Portlet richiesta dall'utente. Nel dettaglio, la specifica JSR-168 permette la creazione di due tipi di PortletUrl:

- ActionURL: viene usato per processare un'azione e fa riferimento ad una ActionRequest.
- RenderURL: viene usato per generare un frammento di codice markup e fa riferimento ad una RenderRequest.
- ResourceURL: viene utilizzato per recuperare risorse come documenti pdf o altro.

Figura 4.6: Ciclo di vita di una Portlet [Sarin, 2012]



Il RenderUrl è un tipo speciale di ActionURL: viene utilizzato con lo scopo di impostare correttamente i parametri di configurazione per il render della Portlet nel momento in cui si presenta una richiesta di renderizzazione; non è perciò possibile usarlo per il passaggio di parametri proveniente da un modulo dati. La creazione dei due tipi di PortletUrl avviene mediante la chiamata dei metodi createActionURL e createRenderURL. Per il passaggio di parametri si può utilizzare il metodo setParameter applicabile all'oggetto PortletURL creato in precedenza.

4.3 Ambiente di sviluppo

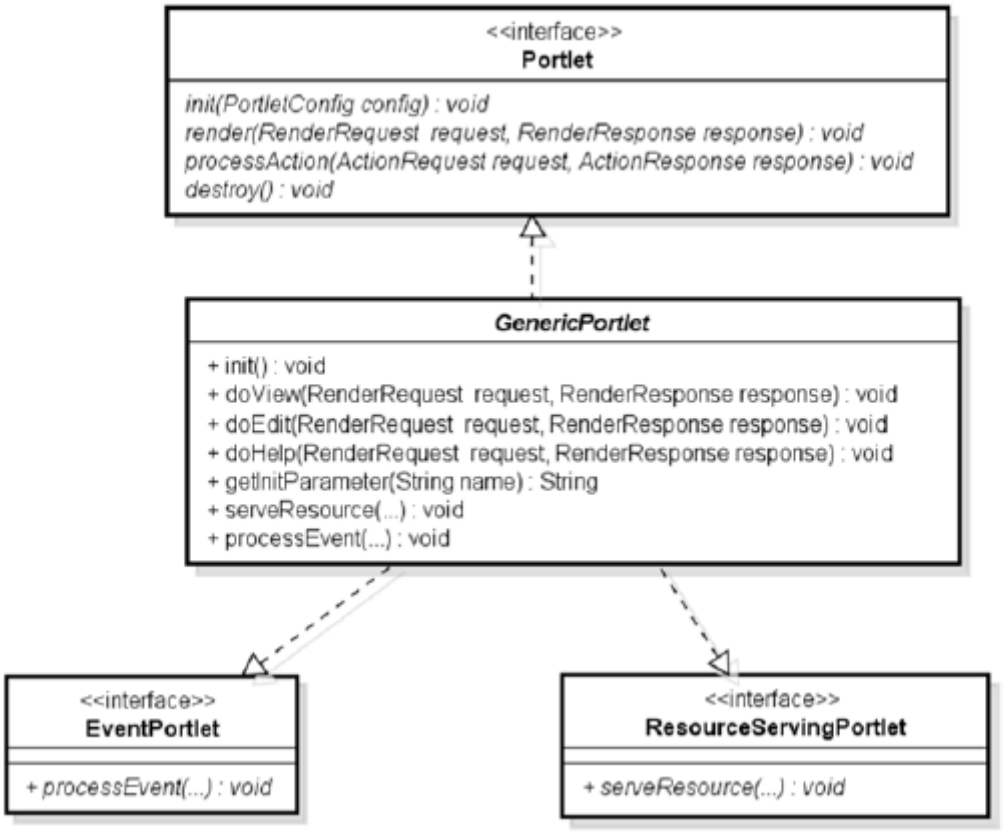
4.3.1 Sviluppo Interfaccia SIAR lato client

Il modello di sviluppo utilizzato per la creazione dell'interfaccia utente del nuovo SIAR si basa sullo sviluppo di componenti, Portlet, lato client. Il motivo di questa scelta è dovuto al fatto che tutta la logica di gestione è implementata tramite il Web Service di tipo REST, che fornisce l'accesso e la manipolazione dei contenuti del SIAR; lo scopo di questa tesi è quello di creare la nuova interfaccia grafica per accedere e gestire tali contenuti. In questa sezione vengono introdotti brevemente i linguaggi utilizzati: HTML, i Css, Javascript, JQuery, JQuery UI, Alloy UI e la tecnologia AJAX, che permette di effettuare richieste asincrone al Web Service.

4.3.1.1 HTML

HTML (HyperText Markup Language) è un linguaggio di markup (non di programmazione) utilizzato per la definizione della struttura di una pagina Web: viene interpretato dal browser Web, il quale lo elabora e genera la visualizzazione della pagina richiesta. Per la definizione della pagina vengono utilizzati i markup tag; un markup tag HTML (o semplicemente tag) è costituito da una keyword (parola chiave) compresa tra due parentesi acute, come ad esempio <body>. In genere i tag HTML vengono inseriti nella pagina in coppia, ad esempio e

Figura 4.7: GenericPortlet [Sarin, 2012]



. Il primo tag, in una coppia di tag, viene chiamato tag di apertura, mentre il secondo tag viene chiamato tag di chiusura. HTML è un linguaggio la cui sintassi è stata definita dal World Wide Web Consortium (W3C) ed è a questo organismo che è necessario fare sempre riferimento quando si parla di HTML. Ad esempio, sul sito ufficiale del w3c è possibile leggere: "*the World Wide Web Consortium (W3C) recommends lowercase in HTML 4 and demands lowercase tags in XHTML*", cioè si raccomanda di scrivere i tag HTML in minuscolo. Un semplice esempio:

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/HTML4/strict.dtd">
 <HTML>
  <head>
   <meta http-equiv="content-type"
    content="text/HTML; charset=UTF-8">
   <title>Titolo</title>
  </head>
  <body>
   **** content ****
  </body>
 </HTML>
```

HTML DOM (Document Object Model). Il DOM è un modo per rappresentare un documento HTML tramite oggetti. La rappresentazione avviene usando una struttura ad albero composta da nodi, in cui ogni elemento HTML è un nodo e ogni nodo è un oggetto. La logica usata nella costruzione dell'albero è la stessa che si utilizzava per costruire un albero genealogico: padre-figlio, cioè gli elementi possono essere annidati in altri elementi.

Non è scopo di questa tesi analizzare in dettaglio tutte le specifiche del linguaggio HTML.

4.3.1.2 Css

I CSS (Cascading Style Sheet - Fogli di stile a cascata), introdotti a partire dall'HTML 4.0, sono utilizzati per definire come una pagina deve essere visualizzata; se l'HTML descrive il contenuto della pagina, i fogli di stile a cascata si occupano della sua formattazione (colori, caratteri, bordi, ecc..). Esistono diversi vantaggi dovuti al loro utilizzo, che sono principalmente:

- riutilizzo dello stesso tema per più pagine Web senza bisogno di aggiunta di codice;
- separazione del codice relativo al contenuto da quello relativo alla rappresentazione grafica, facilitando quindi la manutenzione.

La sintassi generale per una qualsiasi regola css è la seguente: selettore {proprietà:valore; ...; proprietà:valore;}. Esempio:

```
body {
width:  100%;
background-color:  #55F;
margin:  50px; }
```

Il selettore indica l'elemento (o gli elementi) a cui viene applicata la regola, espressa tra le due parentesi graffe. In questo caso la regola sarà applicata all'elemento <body>. Tra le parentesi graffe viene appunto inserito l'elenco delle proprietà da modificare e i rispettivi valori.

Esistono quattro tipi di selettori, elencati nella tabella 4.2.

Tabella 4.2: Tipologia selettori CSS

Nome	Sintassi	Descrizione
markup tag	nomeMarkup	Associa uno stile agli elementi corrispondenti al tag HTML specificato (es. <p>,<a>)
Id	#nomeId	Associa, usando l'identificatore univoco dato con l'attributo "id", uno stile ad uno e un solo elemento HTML all'interno del documento.
Class	.nomeClass	Associa uno stile ad un gruppo di elementi che utilizzano la classe specificata
Pseudo-classe	:nomePsClass	Il selettore tramite pseudo-classe si basa sul concetto di pseudo-classe. Una pseudo-classe non definisce un elemento, ma un particolare stato di quest'ultimo. Ad esempio, quando viene cliccato un elemento, esso si troverà nello stato di "cliccato".

Per ciascuno di questi selettori può essere annesso un insieme di proprietà con lo scopo di definire un particolare stile. Qui di seguito vengono elencati alcuni esempi di proprietà CSS:

- background-color: sfondo di un elemento;
- text-color: colore del testo;

- margin: margine di elemento;
- height: altezza di un elemento;
- position: posizione rispetto alla pagina o all'elemento;
- overflow: visualizzazione corretta se il contenuto presente all'interno di un elemento eccede la sua dimensione;
- left: posizione dal margine sinistro;
- top: posizione dal margine superiore;
- pointer: tipo di puntatore.

I CSS possono essere usati in 3 modi distinti:

1. inline: vengono dichiarati come attributo del tag HTML a cui si riferiscono;
2. interno: vengono dichiarati all'interno del singolo file HTML;
3. esterno: vengono dichiarati in un apposito file esterno al documento HTML e poi richiamati dal codice HTML.

4.3.1.3 Javascript

Javascript è un linguaggio di scripting interpretato, sviluppato per aumentare l'interattività con l'utente nelle pagine Web. L'attività d'interpretazione del codice Javascript non richiede elevate risorse hardware; questa caratteristica è rilevante: infatti, essendo un linguaggio lato client, l'elaborazione deve essere fatta nel minor tempo possibile per non appesantire la pagina e, di conseguenza, renderla poco usufruibile da parte dell'utente finale. Le principali funzionalità di Javascript sono:

- creazione di componenti dinamici all'interno della pagina Web;
- possibilità di leggere e scrivere gli elementi HTML. Tramite Javascript è infatti possibile modificare la struttura del documento HTML in tempo reale, senza interagire con il server;
- controllo degli eventi generati dall'utente; è possibile scrivere il codice Javascript da eseguire quando una pagina Web ha terminato il caricamento, oppure quando un utente clicca su un determinato elemento HTML;
- interazione con i moduli Web compilati, per esempio per il controllo dei valori inseriti;
- lettura e scrittura di cookie per il salvataggio di informazioni nel browser;
- può essere utilizzato per avere informazioni sul browser dell'utente. In questo modo è possibile scrivere il codice custom a seconda del browser che sta interpretando la pagina Web.

Javascript è un linguaggio di scripting orientato agli oggetti. Javascript fu, originariamente, sviluppato da Brendan Eich, della Netscape Communications, con il nome di Mocha e successivamente di LiveScript, ma in seguito è stato rinominato JavaScript ed è stato formalizzato con una sintassi più vicina a quella del linguaggio Java. Javascript è stato standardizzato per la prima volta tra il 1997 e il 1999 dalla ECMA con il nome ECMAScript.

Javascript è un linguaggio case-sensitive, ossia fa differenza tra minuscole e maiuscole; questo significa che per Javascript una variabile denominata var1 è diversa da una variabile denominata Var1. Per dichiarare una variabile in Javascript bisogna usare la parola chiave var. Per scegliere il nome bisogna seguire le seguenti regole: i nomi delle variabili sono case-sensitive (cioè, come già detto, x è diverso da X) e il nome di una variabile deve iniziare con una lettera, il carattere

$ o un underscore. Javascript presenta molti costrutti che utilizzano una sintassi molto simile a quella di Java (cicli for, while, ecc.) ma che non verranno trattati in dettaglio in questa tesi.

Per usare un linguaggio di scripting in un browser è necessario comunicare al browser sia la presenza di uno script sia il linguaggio in cui lo script è scritto. Abbiamo visto che questa operazione viene eseguita tramite il tag `<script>` e il suo attributo `type` che, in questo caso, avrà valore `text/Javascript` (anche se questo può essere differente in base alla versione HTML usata). Esempio:

```
<HTML>
 <head>
  <h1>Titolo</h1>
  <script type="text/Javascript">
    alert("Hello, World!");
  </script>
 </head>
 ...
</HTML>
```

E' possibile anche inserire codice Javascript da un file esterno, aggiungendo l'attributo `src` al tag `script`:

```
<script type="text/Javascript" src="Javascript.js"></script>
```

4.3.1.4 JQuery

JQuery è una libreria JavaScript. Lo scopo della libreria è quello di rendere l'uso di Javascript il più semplice possibile. Il motto di JQuery è "write less, do more", cioè scrivere di meno, fare di più.

Ogni browser implementa uno specifico motore JavaScript con specifiche ed eccezioni proprie, e per questo motivo è spesso impossibile assicurare il funzionamento cross-browser di uno script. Inoltre, il linguaggio presenta ancora delle lacune che lo rendono incoerente rispetto ai linguaggi server-side come PHP o Ruby. L'introduzione di JQuery ha avuto lo scopo principale di superare queste limitazioni di compatibilità fra browser; questa libreria, infatti, crea un livello di astrazione che permette di scrivere un codice che funziona, teoricamente (vedremo che non sempre è così), su tutti i browser supportati.

Le principali caratteristiche della libreria sono elencate di seguito:

- selezione e manipolazione degli elementi HTML;
- selezione e manipolazione dei fogli di stile;
- gestione degli eventi HTML;
- effetti ed animazioni JavaScript;
- navigazione e modifica del DOM HTML;
- chiamate AJAX;
- possibilità di estendere la libreria con plugin: JQuery è una libreria espandibile e sono già presenti innumerevoli plugin per le più disparate funzionalità.

Per includere la libreria:

```
<HTML>
 <head>
  <script src="path/jquery-x.x.js">
```

```
  </script>
 </head>
 <body> ... </body>
</HTML>
```

Descrivere in dettaglio tutte le funzionalità di JQuery non rientra tra gli obiettivi di questa tesi e di seguito verranno descritte brevemente alcune funzionalità.

La sintassi

La logica di funzionamento di JQuery è basata su due passaggi:

1. selezione di un elemento HTML
2. associazione di un'azione all'elemento selezionato.

La sintassi di JQuery rispecchia tale semplicità: **$(selector).action().**

La sintassi di JQuery comprende:

- il simbolo del dollaro per indicare che il frammento di codice deve essere interpretato usando JQuery;
- un selettore (selector) usato per individuare uno o più elementi HTML;
- un'azione (.action()) che viene eseguita sugli elementi selezionati al punto precedente.

Esempi:

```
$("div").hide() - nasconde tutti i <div> della pagina
$("#test").hide() - nasconde l'elemento con id=test
```

Nella tabella 4.3 vengono elencati i principali selettori che si possono usare con JQuery.

Tabella 4.3: Selettori principali JQuery

Selettore	Esempio	Descrizione
*	$("*")	Seleziona tuti gli elementi
#id	$("#test")	Seleziona l'elemento con id=test
.class	$(".prova")	Seleziona tutti gli elementi con attributo class=prova
element	$("div")	Seleziona tutti gli elementi
:first	$("p:first")	Seleziona il primo paragrafo
:last	$("p:last")	Seleziona l'ultimo paragrafo
:even	$("li:even")	Seleziona solo gli elementi dispari
:odd	$("li:odd")	Seleziona solo gli elementi pari
:header	$(":header")	Seleziona tutti gli elementi <hn>, cioè <h1>, <h2>, ..., <h6>

:visible	$(“div:visible”)	Seleziona tutti i visibili
[attribute]	$(“[href]”)	Seleziona tutti gli elementi che hanno un attributo href
[attribute=value]	$(“[href=’index.htm’]”)	Seleziona tutti gli elementi che hanno href=”index.HTML”

Per gli elementi di tipo input sono disponibili alcuni selettori specifici, elencati nella tabella 4.4.

Tabella 4.4: Selettori di JQuery per gli elementi di tipo input

Selettore	**Esempio**	**Descrizione**
:input	$(“:input”)	Seleziona tutti gli elementi di tipo input che non sono solo quelli del tag <input>. A questa categoria appartiene, per esempio, anche il tag <button>
:text	$(“:text”)	Seleziona tutti gli elementi input con type=”text”
:radio	$(“:radio”)	Seleziona tutti gli elementi input con type=”radio”
:checkbox	$(“:checkbox”)	Seleziona tutti gli elementi input con type=”checkbox”
:submit	$(“:submit”)	Seleziona tutti gli elementi input con type=”submit”
:button	$(“:button”)	Seleziona tutti gli elementi input con type=”button”
:image	$(“:image”)	Seleziona tutti gli elementi input con type=”image”
:enabled	$(“:enabled”)	Seleziona tutti gli elementi input abilitati
:disabled	$(“:disabled”)	Seleziona tutti gli elementi input disabilitati
:selected	$(“:selected”)	Seleziona tutti gli elementi input selezionati
:checked	$(“:checked”)	Seleziona tutti gli elementi input checked

Su ogni elemento selezionato è possibile eseguire azioni di diverso tipo: applicare un effetto grafico, visualizzare un messaggio di avviso, modificare il colore di sfondo, ecc. JQuery prevede una serie di metodi che possono essere divisi in sei categorie, sicuramente non esaustive visto che per JQuery sono stati sviluppati centinaia di plugin per estendere le funzionalità della libreria:

1. metodi per la gestione degli eventi;
2. metodi per la gestione degli effetti;
3. metodi per modificare il DOM;
4. metodi per modificare i CSS;
5. metodi per la gestione delle chiamate AJAX;
6. metodi di utilità e di gestion degli array.

Metodi per la gestione degli eventi

Per evento si intende una qualunque interazione dell'utente con la pagina Web, tra cui: click del mouse su un elemento, inserimento dati tramite un form, ecc. Per gestire gli eventi di un determinato elemento HTML, JQuery mette a disposizione una serie di metodi e di seguito vengono elencati i più usati (non ha senso elencarli tutti in questo contesto):

- change(function): associa un gestore all'evento onchange dell'elemento, con gestore si intende il fatto che deve essere definita una funzione Javascript da eseguire quando si verifica l'evento;
- click(function): associa un gestore all'evento onclick dell'elemento;
- dblclick(function): associa un gestore all'evento doppio click dell'elemento;
- hover(inFunction,outFunction): associa un gestore con una o due funzioni all'evento relativo al passaggio del mouse sopra un elemento;
- keydown(function): associa un gestore all'evento relativo alla pressione di un qualsiasi tasto della Keyboard (tastiera);
- keypress(function): associa un gestore all'evento causato dalla ricezione da parte del browser di un carattere;
- keyup(function): associa un gestore all'evento relativo al rilascio di un qualsiasi tasto della Keyboard (tastiera);
- mousedown(function): associa un gestore all'evento relativo al click con il tasto sinistro del mouse sopra un determinato elemento;
- mouseenter(function): associa un gestore all'evento relativo al rilascio (dopo il click, il tasto rimane premuto) del tasto sinistro del mouse sopra un determinato elemento;
- ready(function): associa un gestore all'evento di caricamento della pagina Web; il codice definito all'interno della funzione associata viene eseguito quando il DOM è pronto, ma prima che le immagini e gli altri elementi grafici siano caricati;
- load(function): associa un gestore all'evento di caricamento della pagina Web; il codice definito all'interno della funzione associata viene eseguito quando tutti gli elementi che compongono la pagina sono stati caricati;
- resize(function): associa un gestore all'evento di ridimensionamento della finestra del browser in cui è contenuta la pagina Web;
- scroll(function): associa un gestore all'evento di scroll della pagina;

- submit(function): associa un gestore all'evento di submit di un form.

Esempio:

```
$("button").click (function() {
 -- codice da eseguire --
})
```

I metodi per la gestione degli effetti

JQuery consente di associare agli elementi HTML degli effetti grafici in modo semplice, attraverso l'uso di alcuni metodi che si occupano di associare l'effetto voluto ad un determinato elemento. Di seguito vengono elencati, come fatto in precedenza, i più comuni:

- animate(): è utilizzato per creare animazioni degli elementi HTML della pagina Web. La sintassi completa è la seguente: $(selector).animate(stili,durata,easing,callback), dove i parametri sono:
  - stili: contiene i nomi e i valori delle proprietà CSS da animare;
  - durata: la durata dell'animazione, espressa in millisecondi;
  - easing: l'algoritmo di easing che regola la progressione dell'animazione. JQuery supporta i valori swing (predefinito) e linear;
  - callback: la funzione che viene eseguita quando l'animazione è completa;
- fadeIn() e fadeOut(): la sintassi completa dei due metodi è simile: $(selector).fadeIn(duration,easing,c e $(selector).fadeOut(duration,easing,callback); i tre parametri sono opzionali:
  - il parametro duration indica il tempo per il quale deve durare l'effetto;
  - il parametro easing viene usato per modificare il comportamento previsto dalle animazioni predefinite;
  - il parametro callback indica l'eventuale funzione da eseguire al termine dell'animazione.
- hide(), show() e toggle(): questi tre metodi vengono utilizzati per nascondere o mostrare un elemento HTML. Anche in questo caso le sintassi sono identiche: $(selector).hide(duration, easing, callback), $(selector).show(duration, easing, callback) e $(selector).toggle(duration, easing, callback), dove i parametri hanno lo stesso significato ed uso dei due metodi descritti al punto precedente.

I metodi per modificare il DOM

I metodi di questo tipo consentono di manipolare il DOM HTML, cioè permettono di aggiungere, modificare e cancellare gli elementi HTML della pagina Web. Di seguito vengono elencati i principali:

- after(): inserisce testo semplice o codice HTML dopo il tag di chiusura dell'elemento selezionato;
- append(): inserisce testo semplice o codice HTML dopo l'elemento selezionato, ma all'interno dello stesso elemento;
- attr(): modifica o restituisce il valore di un attributo di un elemento;
- before(): inserisce testo semplice o codice HTML prima del tag di apertura dell'elemento selezionato;

- HTML(): imposta o restituisce il contenuto dell'elemento selezionato;
- insertAfter(): inserisce codice HTML dopo l'elemento selezionato;
- insertBefore() Inserisce codice HTML o elementi prima dell'elemento selezionato;
- remove(): rimuove gli elementi selezionati;
- removeAttr(): rimuove gli attributi selezionati;
- removeClass(): rimuove la classe selezionata.

I metodi per modificare i CSS

I metodi di questo tipo servono per modificare i fogli di stile della pagina Web. Di seguito, vengono elencati i principali:

- addClass(): aggiunge una o più classi all'elemento selezionato;
- css(): permette di modificare o ricavare il valore delle proprietà css associate all'elemento selezionato;
- height(): modifica o restituisce l'altezza dell'elemento;
- width(): modifica o restituisce la larghezza dell'elemento.

I metodi per la gestione delle chiamate AJAX

Il metodo principale per inviare richieste AJAX con JQuery è il metodo $.ajax(), da cui derivano gli altri metodi più specifici (post, get). I parametri della chiamata, che possono essere personalizzati, sono numerosi e, per questo motivo, il metodo accetta un'unico oggetto Javascript con i parametri di base ed altri opzionali per sovrascrivere i valori di default. I principali parametri da settare sono:

- url: URL della risorsa alla quale viene inviata la richiesta, ad esempio un Web service;
- type: il tipo di richiesta HTTP da effettuare. Questo attributo accetta i valori GET (defaut) e POST;
- data: i dati che vengono inviati al server; può essere un oggetto del tipo {chiave: valore, chiave2: valore}, oppure una stringa del tipo "chiave=valore&chiave2=valore2";
- dataType: la tipologia di dato che viene restituito in risposta; i tipi possibili sono: "xml", "HTML", "script", "json", "jsonp" e "text";
- success(data): la funzione che viene eseguita se la chiamata AJAX va a buon fine; il parametro rappresenta i dati restituiti dal server, nel formato indicato nella richiesta;
- error(XMLHTTPRequest, textStatus, errorThrown): la funzione che viene eseguita in caso di errore; i tre parametri sono, rispettivamente, l'oggetto della richiesta, lo stato e la descrizione testuale dell'errore rilevato;
- complete(XMLHTTPRequest, textStatus): consente di specificare una funzione che viene eseguita al termine della chiamata, indipendentemente dal fatto che la request sia andata o meno a buon fine;
- beforeSend(XMLHTTPRequest, textStatus): la funzione di callback che viene eseguita prima di inviare la richiesta al server. I due parametri sono l'oggetto della richiesta e una stringa contenente lo stato della richiesta;
- timeout: il tempo in millisecondi dopo i quali la richiesta viene considerata scaduta;
- async: definisce se la chiamata deve essere asincrona (true) o sincrona (false), nel cui caso bloccherà la pagina fino alla fine della chiamata (il valore di default è true);

- contentType: il tipo di contenuto inviato al server.

Esempio tratto dall'implemetazione della nuova interfaccia Web del SIAR:

```
$.ajax({
  url: url,
  type: 'GET',
  contentType: "application/json; charset=utf-8",
  dataType: "json",
  crossDomain: true,
  beforeSend:function(req){
  req.setRequestHeader('Authorization', hash);},
  error:function(xhr,ajaxOptions,thrownError){...},
  success: function(data) {...}
});
```

4.3.1.5 JQuery UI

Lo sviluppo di interfacce grafiche per il Web che risultino essere intuitive, utilizzabili e graficamente gradevoli è uno dei punti di importanza di questa tesi. L'utilizzo del plugin JQuery UI, insieme alla libreria Alloy (descritta in seguito) ha facilitato il lavoro di sviluppo. JQuery è l'attuale progetto ufficiale per la realizzazione di interfacce utente di JQuery

La struttura della libreria è estremamente modulare; infatti, sul sito del progetto è disponibile sia un download completo di tutto il codice (sia sorgente che compresso per l'ambiente di produzione), sia un builder, con il quale è possibile realizzare una versione personalizzata della libreria in modo da includere solo i moduli necessari all'applicazione. A titolo d'esempio, se l'applicazione utilizza solo i moduli "dialog" (per la creazione di finestre modali) e datepicker (il calendario), allora è possibile creare una versione di JQuery che implementa solo queste funzionalità, evitando di importare nella pagina Web codice non usato che impiega tempo per essere caricato dal Browser.

Uno dei nuovi metodi introdotti da JQuery UI è effect(). Nella sua forma più semplice accetta come parametro una stringa che rappresenta il tipo di effetto da applicare: 'blind', 'bounce', 'clip', 'drop', 'explode', 'fold', 'highlight', 'puff', 'pulsate', 'scale', 'shake', 'size', 'slide', 'transfer'; oltre al tipo di effetto, il metodo accetta altri due parametri (opzionali): un oggetto Javascript per la personalizzazione dell'effetto ed una funzione da eseguire alla fine dell'animazione. I parametri con i quali è possibile modificare le impostazioni di base degli effetti variano in base al tipo di effetto e sono elencati nella documentazione ufficiale.

Nelle interfacce delle applicazioni Web moderne, i controlli degli elementi delle pagine Web come le select, le checkbox e i campi di testo, risultano spesso limitativi per le esigenze di interazione con l'utente finale. In alcuni casi, ad esempio, il fatto di avere a disposizione un calendario a comparsa (datepicker) per la selezione delle date facilita l'utente, rispetto a dover impostare i valori tramite le select che rappresentano il giorno, il mese e l'anno. In JQuery UI sono presenti una serie di elementi chiamati Widget che servono per migliorare l'interazione con l'utente, in termini di usabilità e accessibilità. Il nome Widget potrebbe risultare fuorviante, visto che è spesso associato ad applicazioni desktop (soprattutto nel mobile). JQuery UI comprende una serie di Widget, ma di seguito vengono introdotti brevemente solo quelli usati durante lo sviluppo dell'interfaccia del SIAR:

- Datepicker.
- Dialog.
- Tabs.

Datepicker

Il Datepicker è usato per facilitare l'inserimento delle date, compresa la formattazione della data stessa. Le opzioni fornite da questo Widget sono numerose; di seguito vengono elencate le principali:

- monthNames / monthNamesShort: i nomi dei mesi in formato esteso (Gennaio) e abbreviato (Gen);
- dayNames / dayNamesShort / dayNamesMin: i nomi dei giorni della settimana in formato esteso (Lunedì), abbreviato (Lun) e minimo (Lu);
- dateFormat: il formato della data, in italiano dd/mm/yy;
- duration: indica la durata dell'animazione con cui il calendario appare. I valori accettati sono numeri (millisecondi) oppure stringhe ('slow', 'normal', 'fast');
- showAnim: indica il tipo di animazione da usare per mostrare il calendario.

I principali eventi che possono essere attivati sono, invece:

- beforeShow: viene attivato appena prima che il calendario venga mostrato. La funzione associata all'evento accetta due argomenti: un'istanza dell'input a cui è associato il Widget e un'istanza del datapicker stesso;
- onClose: evento attivato quando il datepicker viene chiuso; accetta come argomenti la data selezionata in formato stringa e un'istanza del datepicker stesso. L'evento viene lanciato anche se non è stata selezionata una data;
- onSelect: evento attivato quando il datepicker viene selezionato; accetta come argomenti la data selezionata in formato stringa e un'istanza del datepicker stesso.

Infine, i metodi specifici del Widget sono:

- hide / show: consente di nascondere e visualizzare il datepicker;
- getData / setData: permette di ricavare ed impostare la data. Per il primo metodo è necessario fornire un ulteriore argomento che rappresenta il formato della data;
- dialog: permette di visualizzare il datepicker usando una finestra popup, nel caso di JQuery UI viene usato un Dialog (descritto in seguito).

Esempio:

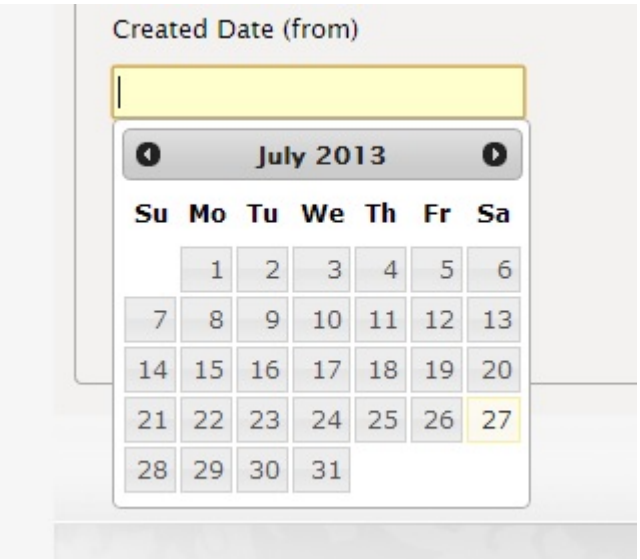
```
("#createdDate").datepicker({dateFormat: "yy-mm-dd"});
```

Dialog

Il Dialog viene usato per la creazione di finestre popup e di finestre modali. Le principali opzioni specifiche del Widget sono:

- autoOpen: definisce se la finestra debba essere aperta automaticamente quando il metodo è richiamato;
- show / hide: (stringa – null) indica una specifica animazione per mostrare e nascondere la finestra. Si deve sempre tenere presente che l'uso di animazioni in questi casi serve per dare maggior risalto al contenuto che si ta per mostrare e perciò esse dovrebbero essere utilizzate con parsimonia al fine di non rendere l'interazione troppo pesante;
- buttons: permette di aggiungere nella parte inferiore della finestra dei bottoni. L'opzione accetta un oggetto Javascript in cui la chiave rappresenta il testo del bottone, mentre il valore è una funzione di callback lanciata quando il bottone viene premuto;

Figura 4.8: Esempio di Datepicker



- dialogClass: definisce una classe CSS aggiuntiva per personalizzare graficamente la finestra;
- draggable / resizable: un valore booleano che, se impostato a false, rende la finestra fissa oppure non ridimensionabile;
- modal: permette di creare un Dialog modale. In questo modo tutti gli elementi esterni vengono bloccati e l'utente è obbligato ad interagire con essa. Questo tipo di Dialog è simile alle funzioni alert() e confirm() di JavaScript;
- title: indica il testo da usare come titolo della finestra.

Gli eventi attivabili principali sono:

- beforeClose: viene attivato quando l'utente cerca di chiudere la finestra. Se la funzione associata restituisce false, la finestra resterà aperta. Potrebbe essere utile nel caso in cui si voglia verificare che l'utente abbia compilato dei campi di input obbligatori presenti nella finestra;
- open / close: vengono attivati quando la finestra viene aperta o chiusa;
- dragStart / drag / dragStop: vengono lanciati quando la finestra è spostata;
- resizeStart / resize / resizeStop: vengono attivati quando si vuole ridimensionare la finestra.

Infine, i metodi specifici:

- open / close: con questi metodi è possibile mostrare e nascondere le finestre senza la necessità che l'utente le richiami direttamente;
- moveToTop: quando sono aperte più finestre nella pagina, potrebbero risultare sovrapposte le une alle altre. Con questo metodo è possibile portare in primo piano una delle finestre;
- isOpen: questo metodo restituisce true se la finestra è visibile, altrimenti restituisce false.

Esempio di definizione di un Dialog modale usato nell'applicazione sviluppata:

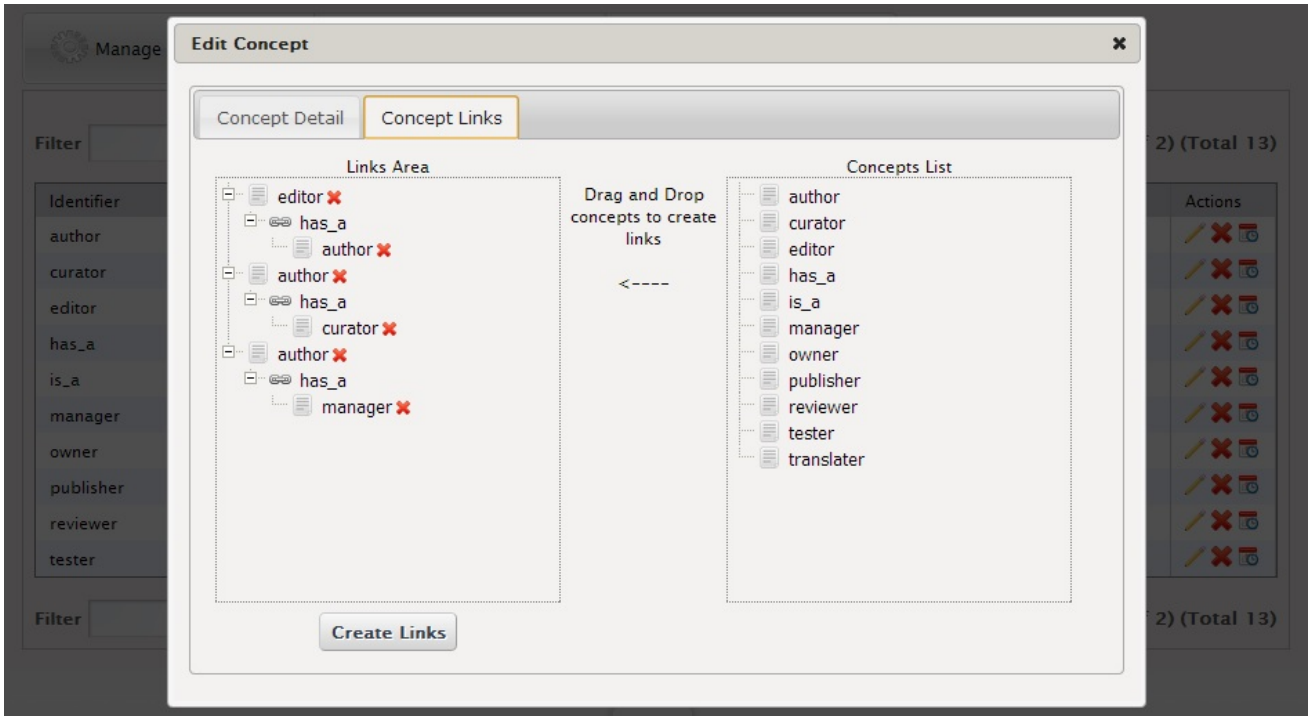
```
$("#dialogGetNamespace").dialog({
  autoOpen: false,
  modal: true,
```

```
  resizable: false,
  title: 'Search Namespace',
  width : 900,
  height : 500,
  open: function( event, ui ) {
    IUI.resources.concept.loadNamespaces();
  }
});
```

Per aprire e chiudere il Dialog:

```
$("#dialogGetNamespace").dialog("open");
$("#dialogGetNamespace").dialog("close");
```

Figura 4.9: Esempio di Dialog



Tabs

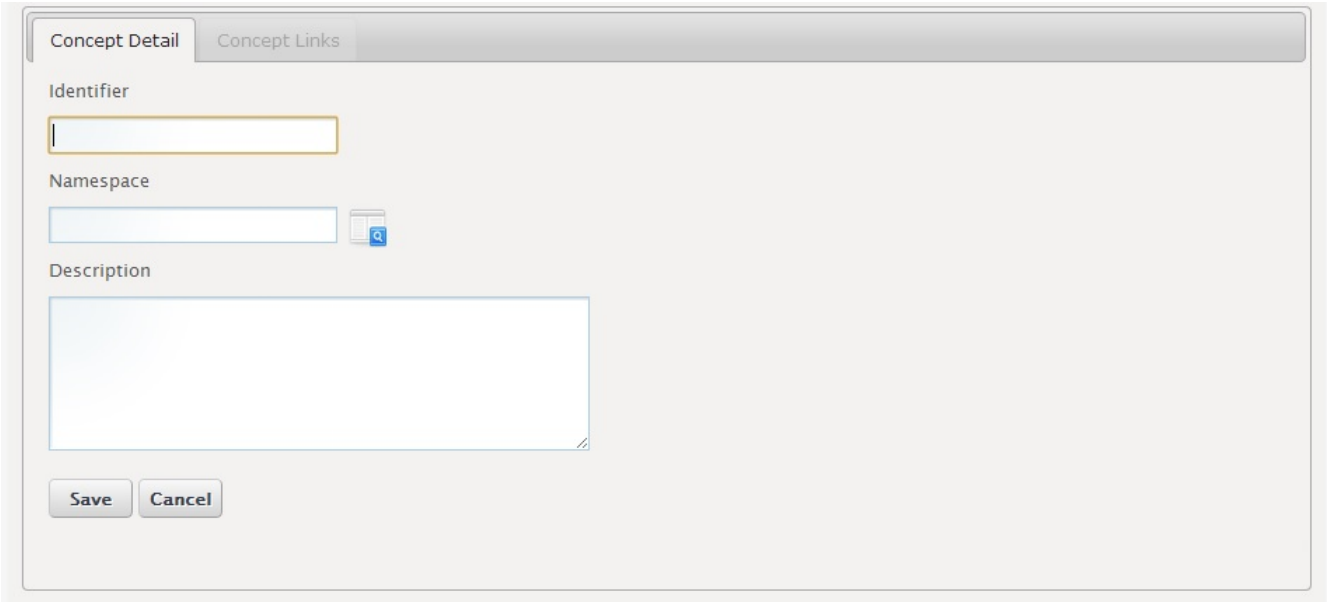
L'utilizzo delle schede (tab) per gestire blocchi di contenuto molto corposi o di diversa natura è divenuta una soluzione molto usata nello sviluppo di interfacce Web. Solitamente, l'uso delle tab è indicato nei casi in cui è necessario suddividere un contenuto in blocchi ben distinti. Le principali opzioni offerte dal Widget sono:

- ajaxOptions: è costituito da un oggetto JavaScript contenente le opzioni per la gestione del caricamento di contenuti nelle tab attraverso richieste AJAX;
- event: è l'evento che permette di visualizzare il contenuto di una scheda. In alcuni casi potrebbe essere preferibile associare l'apertura di una scheda all'evento mouseover oppure all'evento onclick;
- selected: indica l'indice (a partire da 0) della scheda di default al momento dell'inizializzazione del Widget.

Di seguito, un esempio di pagina strutturata a tabs:

```
$("#tabs").tabs({ disabled: [1] });
```

Figura 4.10: Esempio di una pagina strutturata a tabs



I principali eventi attivabili sono:

- select: viene lanciato quando viene selezionata una scheda, più precisamente quando è invocato l'evento dell'opzione event (di default click).
- load: viene attivato quando il contenuto di una scheda caricato da remoto viene caricato completamente;
- show: viene lanciato quando una scheda viene mostrata e diventa attiva;
- add / remove: viene lanciato quando viene aggiunta una nuova scheda;
- enable / disable: viene attivato quando una tab viene abilitata o disabilitata.

Tutte le funzioni di callback associate agli eventi accettano come argomenti l'evento Javascript nativo ed un oggetto con le seguenti proprietà:

- tab: etichetta della scheda selezionata;
- panel: elemento che contiene la scheda corrente;
- index: un numero intero che rappresenta l'indice della scheda selezionata (partendo da 0).

I metodi specifici sono:

- add: attraverso questo metodo è possibile aggiungere una scheda al Widget, associando ad essa un contenuto già presente nel documento oppure caricandolo da remoto. Il metodo ha due argomenti in input: il primo è una stringa che può essere legata ad un contenuto presente nel documento oppure può essere un URL da cui caricare in modo asincrono un frammento di HTML; il secondo argomento obbligatorio è una stringa da utilizzare come testo della nuova etichetta.
- remove: rimuove dal Widget una tab specifica in base al numero di indice passato come argomento (partendo da 0);
- select: rende attiva la scheda indicata come ulteriore argomento;
- load: ricarica il contenuto di una specifica scheda il cui contenuto viene caricato dinamicamente con chiamate AJAX. Per questo metodo è necessario fornire due parametri, che

indicano rispettivamente la posizione della scheda nel Widget (a partire da 0) ed il nuovo indirizzo da cui caricare i contenuti;

- length: restituisce il numero di tab caricate nel Widget;
- abort: cancella tutte le richieste AJAX ancora in sospeso per le tab caricate dinamicamente;
- rotate: attiva una rotazione automatica fra le schede del Widget.

4.3.1.6 Alloy UI

Alloy UI è la libreria Javascript nativa di Liferay; è stata progettata a partire da YUI, la libreria Javascript rilasciata da Yahoo. Per lo sviluppo del codice in questa tesi è stata utilizzata la versione 1.0 di Alloy UI, mentre attualmente è già stata rilasciata la versione 2.0. Uno dei vantaggi di utilizzare questa libreria nativa sta nel fatto che non è necessario importare lo script ed includerlo in ogni pagina in cui viene usato. Inoltre, Alloy UI fornisce alcune funzionalità e Widget che non sono presenti nella versione ufficiale di JQuery UI (si potrebbero ottenere in ogni caso, utilizzando dei plugin specifici) e che sono stati usati nel progetto. Il principale Widget utilizzato è la datatable di Alloy UI, di seguito viene riportato un esempio di definizione:

```
var columnDefs = [
  {key:"Actions"},
  {key:"Identifier",sortable: true},
  {key:"Prefix"},
  {key:"Description"}
];

var arrData= -- array con i dati --

AUI().use("datatable", "datatable-sort", function (A) {
  // A table from data with keys that work fine as column names
  var table = new A.DataTable.Base({
  columnset: columnDefs,
  recordset: arrData
  }).plug(A.Plugin.DataTableSort);

  table.render("#tableNSList");
});
```

Per utilizzare Alloy UI è necessario usare la funzione `AUI().use( -- parametri --, function)` per richiamare le componenti di Alloy da usare, in questo caso vengono utilizzati `"datatable"` e `"datatable-sort"`. Infine viene definita la funzione all'interno della quale è possibile scrivere il codice di Alloy; in questo esempio viene definita la datatable: `var table = new A.DataTable.Base({ ...})`, che accetta più parametri, tra cui un oggetto Javascript con le definizioni delle colonne e un array con i record da visualizzare in forma tabellare.

5 Analisi e progettazione

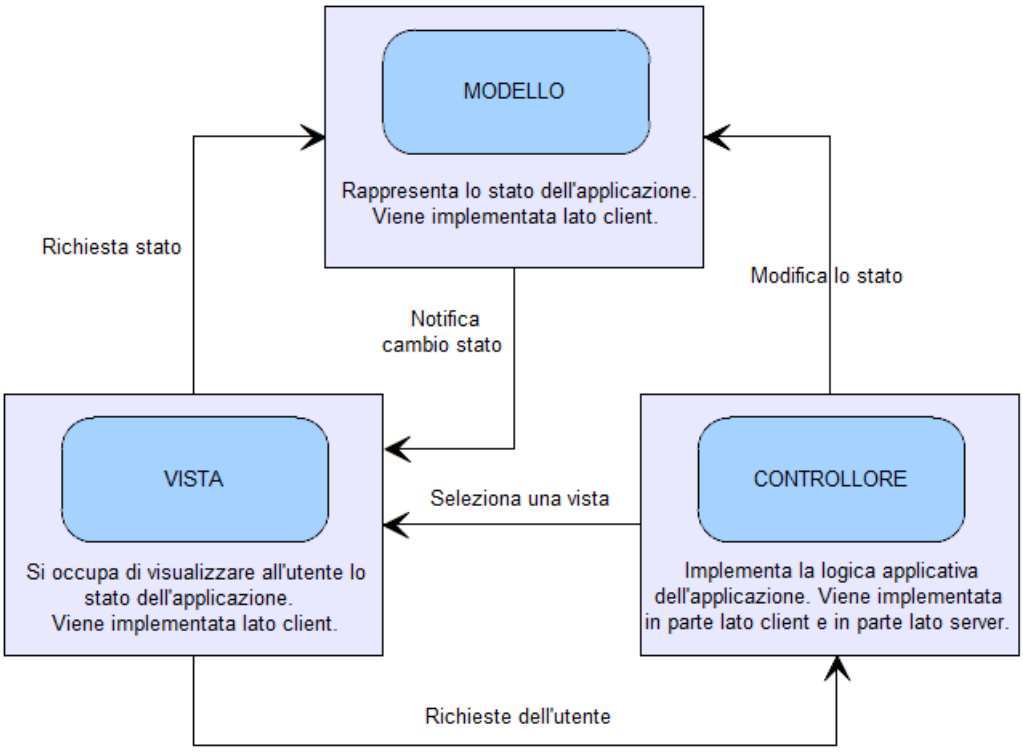
In questo capitolo viene illustrata la struttura della nuova interfaccia del SIAR. Dopo aver presentato la nuova architettura, vengono analizzati i motivi che hanno portato alle scelte effettuate e i vantaggi che ne derivano. Nella sezione finale, invece, viene effettuata un'analisi del SIAR 2.0, tenendo conto dei parametri di confronto introdotti nel capitolo 2.

5.1 Reingegnerizzazione del SIAR: SIAR 2.0

Il lavoro di questa tesi si occupa di rivedere l'attuale interfaccia utente del SIAR, basata su pagine HTML, verso un nuovo sistema basato sulla piattaforma Open Source Liferay (portale Web) e sul paradigma delle Portlet, introdotte nel capitolo precedente. Nel prossimo capitolo verrà descritto in dettaglio l'ambiente di lavoro utilizzato per la progettazione e lo sviluppo dell'interfaccia utente del nuovo portale Web del SIAR e il prototipo realizzato.

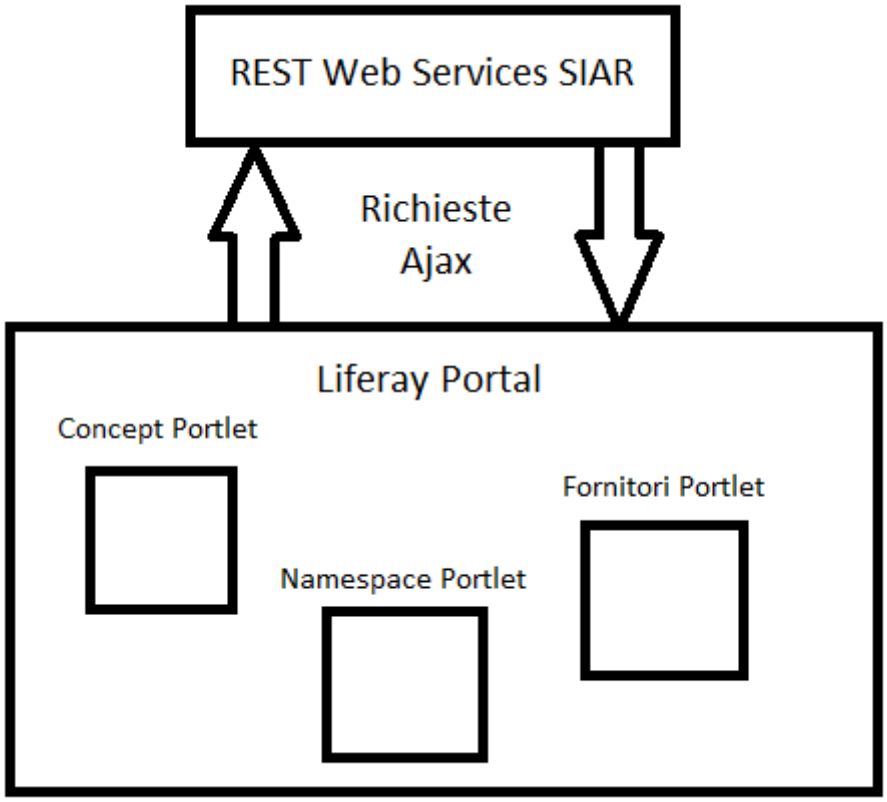
Lo schema della nuova architettura è riportato nelle figure 5.1 e 5.2. Liferay implementa il modello MVC. Il Model-View-Controller (tradotto in italiano Modello-Vista-Controllo) è un pattern architetturale molto diffuso nello sviluppo di sistemi software in grado di separare la logica di presentazione dei dati dalla logica di business.

Figura 5.1: Schema MVC



Nella nuova architettura, la logica di business viene implementata tramite i REST Web Services che permettono di accedera ai dati e ai metadati del SIAR. Invece, la logica di presentazione viene realizzata attraverso le Portlet

Figura 5.2: Schema architettura SIAR 2.0



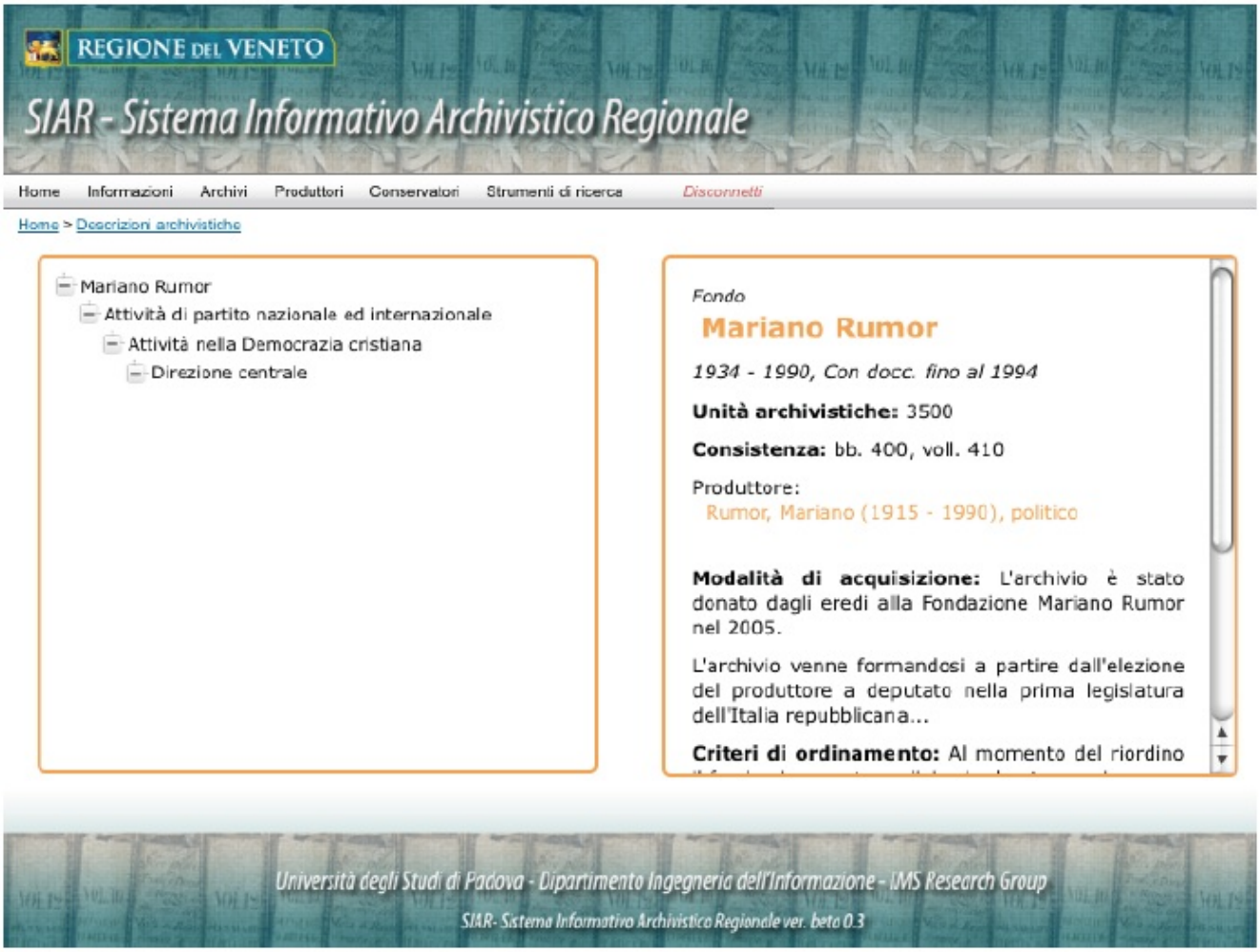
Di seguito viene riportata una schermata dell'attuale interfaccia grafica del SIAR (figura 5.2).

La scelta di utilizzare Liferay Portal per sviluppare la nuova interfaccia Web del SIAR è dettata da alcuni fattori per superare alcuni limiti della vecchia interfaccia, tra cui la possibilità di inserire applicazioni diverse all'interno di un unico ambiente omogeneo (il portale) e richiedendo un unico login all'utente. Un altro fattore importante riguarda la facilità di installazione e di configurazione delle Portlet e del portale, infatti usando Liferay è possibile creare, rimuovere e modificare i temi delle pagine del portale stessi, le finestre e le impostazioni di sicurezza. Inoltre, anche la gestione degli utenti, delle loro policies di accesso e la visibilità sui contenuti è semplificata. Infine, la scelta di sviluppare delle applicazioni Web attraverso l'utilizzo delle Portlet permette di beneficiare dei vantaggi portati da queste componenti, tra cui la riusabilità all'interno di pagine diverse del portale. I vantaggi vengono illustrati in maniera più dettagliata nella sezione successiva.

5.2 Vantaggi architettura proposta

La scelta di utilizzare un portale Web come Liferay con una struttura basata sulle Portlet (componenti Web indipendenti e riutilizzabili) e un'architettura di tipo REST è stata adottata con delle motivazioni ben precise. Innanzittutto è necessario comprendere il motivo per cui si è deciso di puntare sullo sviluppo di un portale e non sulla creazione di un semplice sito: come già accennato, un portale Web può essere considerato un "aggregatore" di contenuti, in grado

Figura 5.3: Software SIAR Silvello [2011]



di contenere un gran numero di componenti che accedono alle informazioni attraverso sorgenti dati diverse. Un sito Web è costituito da un insieme di pagine Web che condividono la stessa struttura e impostazione grafica, mentre un portale Web è composto da un insieme di elementi presenti all'interno delle pagine Web, la cui disposizione può essere modificata dall'utente e la cui struttura grafica può essere personalizzata in modo indipendente dagli altri componenti.

La scelta è ricaduta su Liferay grazie alla possibilità di realizzare portali aziendali, intranet ed extranet in modo semplice e veloce, ottimizzando i tempi di lavoro e favorendo la collaborazione e l'interazione tra i vari gruppi di utenti. Inoltre, questo portale racchiude funzionalità di CMS (Content Management System, un software il cui compito è facilitare la gestione dei contenuti di siti Web) native e permette lo sviluppo di nuovi componenti che si vanno ad integrare con la struttura globale dell'applicazione; è adatto per lo sviluppo di realizzazioni Web conformi alle caratteristiche del Web 2.0, permettendo un forte livello di interazione e personalizzazione da parte dell'utente ed un'elevata possibilità di configurazione dal lato dell'amministrazione. Grazie alla sua strutturazione "community", Liferay è particolarmente adatto per le realtà aziendali che necessitano di aree (pubbliche o private) distinte, per suddividere le informazioni gestite dai diversi reparti interni (commerciale, amministrativo, tecnico, etc.) o relative a sedi differenti.

Il primo sostanziale vantaggio di utilizzare un'architettura basata sulle Portlet è il concetto di riutilizzo delle Portlet stesse. In fase di sviluppo è possibile indicare, attraverso la proprietà *instanceable* presente nel file di configurazione della Portlet *liferay-Portlet.xml*, se la Portlet può

essere o meno istanziata più volte, cioè se può essere utilizzata (true) o meno (false) in più pagine oppure anche in una stessa pagina del portale. La possibilità di riusare le Portlet permette di avere una struttura di pagine Web estremamente flessibile, con la possibilità di fornire anche all'utente finale i permessi per poter personalizzare le Portlet. Infatti, un utente può costruire un portale personalizzato, scegliendo dove, come e in quale pagina disporre le Portlet con diversi contenuti. Inoltre le Portlet faciltano l'accesso a più sorgenti dati, consentendo all'utente di evitare di usare più applicazioni Web standard per effettuare le stesse operazioni. Ad esempio, si può verificare uno scenario di questo tipo: un utente vuole visualizzare i dati del meteo della propria città e necessità anche di un'applicazione per prenotare le proprie vacanze: di norma per effettuare queste operazioni è necessario accedere a due applicazioni Web distinte, invece, è possibile pensare che l'utente possa ottenere questi dati tramite due Portlet differenti ma che possono essere posizionate sulla stessa pagina.

Il secondo vantaggio, sempre derivante dalla modularità ottenuta tramite un'architettura basata sulle Portlet, include una strategia flessibile di progettazione e di sviluppo del codice. Infatti, è stato prevista la creazione di una singola Portlet per ogni componente del SIAR, tra cui le Portlet per gestire i produttori e i conservatori, ma anche quelle usate per la gestione dei metadati (tra cui le Portlet per i Concept ed i Namespace). Questo facilita eventuali modifiche o evoluzioni successive, in quanto, se si rende necessario introdurre cambiamenti su una singola Portlet, le modiche saranno circoscritte solo a quella.

Un'altra importante caratteristica di Liferay è data dalla possibilità di personalizzare il portale stesso, adeguandolo alle necessità che si presentano in fase di sviluppo e nelle fasi successive. Liferay permette di personalizzare il comportamento delle componenti del portale e anche tutte le Portlet che vengono messe a disposizione di default, utilizzando l'ambiente hook e il plugin ext.

Anche l'utilizzo di un Web Service che implementa il paradigma REST comporta alcuni vantaggi. Un servizio Web basato sulla metodologia REST ha come punto di forza l'utilizzo di URI, che permette di identificare univocamente una risorsa in Internet, di descrivere il meccanismo da usare per accedere alla risorsa e specifica anche dove può essere trovata. Inoltre, qualsiasi client e server che supporta invocazioni HTTP, può facilmente invocare un servizio REST. Per quanto riguarda la sicurezza, nelle comunicazioni tra servizi REST, gli apparati di sicurezza come firewall sono in grado di riconoscere l'intento di ogni messaggio, analizzando il comando HTTP utilizzato nella richiesta.

Nei servizi REST le richieste, come detto precendentemente, vengono indirizzate verso URI differenti, che vengono mappati sulle risorse. Il consumo di banda di un servizio REST è ridotto rispetto ad altre tipologie di Web Services, infatti viene inviato insieme alla richiesta un documento XML o altre informazioni oltre all'URI solo quando bisogna creare o aggiornare lo stato di una risorsa.

Infine, un'architettura basata su un Web Portal come Liferay, che implementa le specifiche JSR sulle Portlet e su Web Services progettati e sviluppati secondo il paradigma REST, ha permesso di rendere il SIAR indipendente dal sistema operativo utilizzato; l'unico vincolo imposto riguarda il browser Web: attualmente il SIAR è compatibile con Mozilla Firefox e Google Chrome.

5.3 Analisi del SIAR

In questa sezione verrà fatta una breve analisi del SIAR, in base ai parametri di valutazione per sistemi informativi archivistici definiti nel capitolo precedente (sezione 2.2.1), con lo scopo di valutare e confrontare il sistema con quelli analizzati nel capitolo precedente.

L'architettura del SIAR è, come visto in precedenza, a tre livelli: livello dati, livello di gestione dei metadati e livello interfaccia utente. Per quanto riguarda la raccolta metadati da altri sistemi archivistici, il SIAR era stato inizialmente progettato proprio per recuperare metadati da più fonti archivistiche, dislocate su tutto il territorio regionale, utilizzando il protocollo OAI-PMH. L'architettura del SIAR si basa su un'implementazione del modello NESTOR, che utilizza alcune funzionalità del già citato protocollo OAI-PMH e permette di superare alcune lacune, rilevate in precedenza, dello standard per la descrizione archivistica EAD. Lo standard EAD, infatti, soffre di alcuni problemi, tra cui l'eleborazione automatica dei file EAD, l'accesso ai dati riguardanti una particolare descrizione ed il fatto che un file EAD contiene informazioni riguardanti sia la struttura che il contenuto delle descrizioni archivistiche. Il problema dell'elaborazione automatica dei file XML EAD nasce dal fatto che non sempre è possibile conoscere in anticipo la struttura del file XML stesso perchè lo standard permette una certa libertà in questa fase: ad esempio, un campo descrittivo può essere valutato ed interpretato diversamente dagli archivisti e quindi può assumere significati diversi in base a chi lo elabora. L'accesso individuale a particolari descrizioni archivistiche risulta problematico perchè è necessario percorrere tutta la struttura gerarchica dell'archivio e questo è dovuto al fatto che EAD non richiede una descrizione standardizzata del materiale descritto. Un singolo file EAD rappresenta e descrive un intero archivio e contiene informazioni sia di tipo strutturale che di natura informativa (i contenuti archivistici); la gestione di file di queste dimensioni può risultare complicata. Infine, il SIAR aderisce ai standard per la descrizione archivistica ISAD(G), ISAAR(CPF) e ISDIAH.

Per quanto riguarda l'interfaccia utente, è stato progettato per il SIAR un portale Web basato sulla tecnologia delle Portlet, che vedremo in dettaglio nel prossimo capitolo. La strategia di progettazione pensata ed utilizzata prevede lo sviluppo di una singola Portlet per ogni elemento che compone il SIAR: ad esempio, è previsto lo sviluppo di una Portlet che gestisce i dati e le informazioni sui produttori di archivi. Come già accennato, le Portlet permettono all'utente finale di interagire con il sistema, attraverso le operazioni fondamentali di creazione, modifica e visualizzazione di una risorsa, e per espletatare queste funzionalità vengono utilizzati i Web Services di tipo REST progettati per il SIAR. La scelta di usare queste tecnologie ha permesso di rendere il sistema indipendente dal sistema operativo utilizzato dall'utente; l'unico vincolo inposto riguarda il browser Web: attualmente il SIAR è compatibile con Mozilla Firefox e Google Chrome. Infatti, le chiamate AJAX, usate per interagire con i Web Services del SIAR, vengono implementate usando la libreria Javascript JQuery, che non viene supportata correttamente dal browser Internet Explorer. L'applicazione sviluppata effettua le chiamate asincrone ai servizi usando i metodi forniti da JQuery (introdotti in seguito); questi metodi, a loro volta, richiamano l'oggetto XMLHttpRequest[1], che permette di effettuare richieste HTTP asincrone ad un Web Service. Questo oggetto viene richiamato in maniera differente, in base al browser in cui viene istanziato. Nel caso di Internet Explorer, ad esempio, questo oggetto è restituito da un ActiveXObject, mentre negli altri browser, come Mozilla, Safari, Chrome, Opera ecc., XMLHttpRequest è supportato in modo nativo. Successivamente, anche per le versioni 7, 8, 9 e 10 di Internet Explorer XMLHttpRequest è stato supportato nativamente, ma per i browser di questa famiglia rimangono dei problemi di parsing dei dati scambiati con i Web Services (supporto al parsing dei dati nel formato JSON e XML) e dei problemi di cross domain, cioè di richieste HTTP verso server che non appartengono allo stesso dominio in cui si trova l'applicazione. La cosiddetta Same-Domain-Policy è una restrizione presente nei recenti browser che impedisce al codice Javascript, tramite qualsiasi tipo di richiesta HTTP, di accedere a risorse che si trovano su server diversi rispetto a quello dell'applicazione.

Il DBMS utilizzato è PostgreSQL, un database relazionale open source.

[1] http://www.w3.org/TR/XMLHttpRequest/

Per quanto riguarda le funzionalità di ricerca, il SIAR mette a disposizione la possibilità di interrogare il sistema stesso scrivendo in maniera opportuna delle query e inviando la richiesta al Web Service.

Infine, la licenza con cui viene rilasciato il SIAR è la licenza open source GPL.

6 Descrizione prototipo realizzato

In questo capitolo viene introdotta e descritta l'interfaccia Web del SIAR, realizzata attraverso lo sviluppo di alcune Portlet. Viene introdotta la struttura di base che viene condivisa da ogni Portlet e vengono illustrati i principali dettagli implementativi per giustificare le scelte fatte in precedenza. Nella sezione finale del capitolo viene descritta la procedura usata per la progettazione e lo sviluppo del tema usato nel portale.

6.1 Installazione e configurazione ambiente di sviluppo

I software utilizzati e installati su ambiente Windows 7 per lo sviluppo della tesi sono i seguenti:

- JDK 1.7.
- PostgreSQL 9.2.
- Liferay-portal-6.1.1 (versione bundle con Tomcat 7).
- Liferay-plugins-sdk-6.1.1.
- Eclipse Juno con l'ambiente di sviluppo Liferay IDE 1.6.

Il primo passo è quello di installare il JDK 1.7; il JDK (Java Development Kit) è un kit che consiste nel compilatore Java (Javac) e da strumenti ad esso correlati (per citarne alcuni: Javadoc per la generazione automatica della documentazione e jdb il debugger) che consentono la realizzazione di applicazioni Java. È disponibile nella sezione download del sito di Oracle[1] per i diversi sistemi operativi.

Il secondo passo consiste invece nell'installazione del DBMS prescelto, ovvero PostgreSQL 9.2, scaricabile dalla sezione download del sito ufficiale[2] e nella creazione di un database (di tipo UTF-8) e di un ruolo utente che saranno utilizzati da Liferay. PostgreSQL presenta queste caratteristiche:

- un linguaggio nativo chiamato PL/pgSQL simile al linguaggio procedurale di Oracle PL/SQL, che offre particolari vantaggi nelle procedure che fanno uso intensivo di query.
- Wrapper per i più diffusi linguaggi di scripting come Perl, Python, Tcl, e Ruby che permettono di utilizzare la loro potenza nella manipolazione delle stringhe e nel link ad estese librerie di funzioni esterne.
- Licenza Open Source.
- Complesso sistema di autorizzazioni e gestione della multiutenza.
- Viste e procedure.
- Supporto per le transazioni ed i cursori.
- Chiavi ed integrità referenziale.

[1] *http://www.oracle.com/technetwork/Java/Javase/downloads/index.HTML*
[2] *http://www.postgresql.org/download/*

- Triggers.
- Capacità illimitata per ogni database.
- Numerosi tipi di dati complessi.

L'ultimo passo consiste nello scaricare e scompattare Liferay-portal-6.1.1 (versione bundle con Tomcat 7) e il Liferay-plugins-sdk-6.1.1. Decomprimendo lo zip, che contiene Liferay, si trovano le cartelle:

- data: contiene i file se viene usato hsql come DBMS, i dati di jackrabbit repository e gli indici di ricerca di lucene;
- license;
- application server: Tomcat 7.

A questo punto è necessario avviare il Tomcat e procedere alla configurazione di Liferay, accessibile all'indirizzo *http://localhost:8080.* E' possibile procedere utilizzando il Wizard di configurazione (senza dover intervenire sui file di properties), inserendo la lingua utilizzate e le impostazioni che rigurdano il database su cui si appoggerà Liferay. Al termine della configurazione il file portal-setup-wizards.properties viene modifcato come segue:

```
admin.email.from.name=Test Test

jdbc.default.password=liferay

liferay.home=C:/Users/Tesi/Desktop/liferay-portal-6.1.1

admin.email.from.address=test@liferay.com

jdbc.default.driverClassName=org.postgresql.Driver

jdbc.default.username=liferay

jdbc.default.url=jdbc:postgresql://localhost:5432/liferay6.1.1

setup.wizard.enabled=false
```

Come ambiente di sviluppo è stato utilizzato Eclipse Juno[3]. Il vantaggio principale di quest'ultimo è che offre un'ampia scelta di software di terze parti (plugin) che ne incrementano la produttività e le potenzialità. I principali vantaggi nell'utilizzo di questo IDE sono:

- Eclipse è stato inizialmente pensato per programmare in Java;
- consente la creazione di nuovi progetti, package e classi in modo semplice e veloce;
- gestione dei progetti molto flessibile tramite la vista "Project Explorer";
- la vista "Outline" consente di consultare tutte le varie informazioni sul contenuto del file aperto (classi, metodi, variabili);
- possibilità di usufruire di formattazione e completamento automatico del codice.

Per lo sviluppo delle Portlet è stata installata, all'interno di Eclipse, la Liferay IDE, disponibile nell'Eclipse Marketplace. La Liferay IDE viene configurata installando il Liferay-plugins-sdk-6.1.1. Il Plugins SDK è un ambiente usato per lo sviluppo di plugin e per realizzare Portlet, temi, layout, hooks, come software indipendente dal portale. In particolare, le Portlet create in Plugins SDK possono importare classi solamente dalle librerie Portal-Kernel e Portal-Service (e dagli altri file JAR presenti nella cartella /WEB-INF/lib della Portlet), ma non da Portal-Impl. Ciò forza le Portlet ad affidarsi completamente alle API del portale e a non dipendere dalle classi implementative definite in Portal-Impl. Per gestire le proprietà del portale, lingua e JSP relative a Portal-Impl si devono usare gli hook. L'ambiente SDK consente quindi di creare plugin hot-deployable per il portale Liferay, senza dover riavviare l'Application Server.

[3] *http://www.eclipse.org*

6.2 Sviluppo interfaccia utente

La struttura generale di ogni Portlet è basata su tre componenti principali:

1. Nuovo contenuto/risorsa: permette di inserire all'interno del sistema SIAR un nuovo contenuto, che varia in base al tipo di Portlet; infatti, ogni Portlet sviluppata ha il compito di gestire risorse differenti, come i produttori, i consumatori, oppure i metadati usati dal sistema (concept, namespace, ecc.);
2. Gestione contenuti/risorse: questa componete permette di visualizzare e modificare le risorse già presenti nel sistema;
3. Ricerca contenuti/risorse: funzionalità di ricerca e recupero delle risorse memorizzate e mantenute dal SIAR, con la possibilità di effettuare query personalizzate e selettive.

La struttura della libreria Javascript sviluppata segue questo modello: il codice Javascript viene inserito, in base al tipo di funzionalità, in file Javascript differenti e opportunamente nominati. E' stato definito un oggetto principale: `IUI`, che viene richiamato in tutti i file della libreria. In ogni file Javascript viene creato un oggetto, definito come parametro dell'oggetto principale `IUI`, su cui vengono sviluppati i metodi, che possono essere richiamati in altre parti di codice. Un esempio:

```
var IUI = IUI || {};
IUI.resources = {};
IUI.resources.concept = {};
...
this.postConcept = function(){ ... };
...
this.deleteConcept = function(){ ... };
...
```

Nel caso sopra riportato è stato definito l'oggetto `IUI.resources.concept` e i relativi metodi per la gestione delle risorse di tipo "*concept*".

6.2.1 Nuovo contenuto

L'inserimento di nuovi contenuti viene effettuato tramite un form, che contiene tutti i campi necessari per completare la creazione di una risorsa. Il metodo `IUI.resources.concept.postConcept()`, il cui codice viene riportato di seguito, permette di effettuare l'inserimento nel sistema di un nuovo contenuto.

```
this.postConcept = function(){
  //retrieve data from the form and create a object
  var conceptObj = {};
  conceptObj.identifier = $('#identifier').val();
  conceptObj.namespace = $('#namespace').val();
  conceptObj.description = $('#description').val();
  var jsonNS = JSON.stringify(conceptObj);

  //retrieve creator and right
  var creator = Connection.getCreator();
  var right = Connection.getRight();

  //create data to be sent
  var concept = '{"direct":{"file-metadata":{"creator":"'+creator+'",';
  concept += '"rights":"'+right+'"},"concept":';
  concept += jsonNS;
  concept += '}}';
```

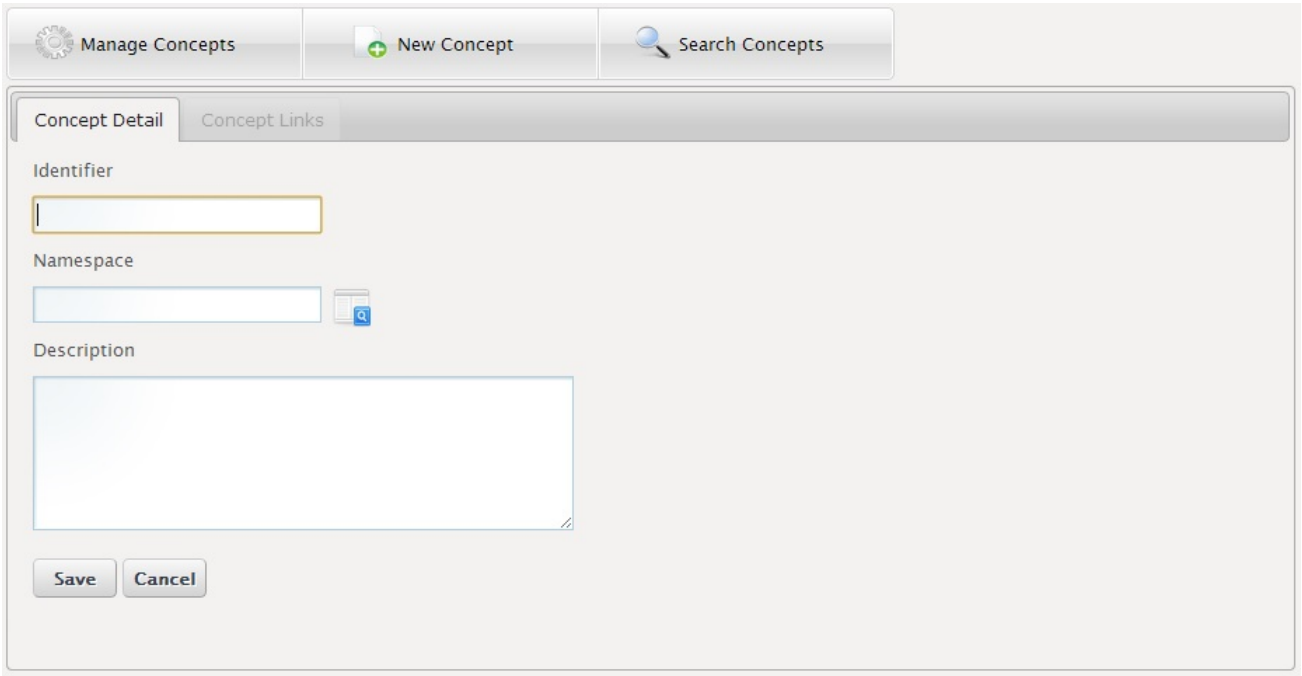
```
  var hash = IUI.resources.concept.createAuth();

  var url = this.connectionURI + "concept";

  $.ajax({
    url : url,
    type : 'POST',
    contentType: "application/json",
    data: concept,
    dataType: "json",
    crossDomain: true,
    beforeSend : function(req) {
      req.setRequestHeader('Authorization', hash);
    },
    error : function(xhr, ajaxOptions, thrownError) {
          Concept_errorUtilities.showErrorMessage(xhr.responseText,
    "Error!","New Concept");
    },
    success : function(data) {
    //show success message
    .....
    }
  });
};
```

Nella Figura 6.1, invece, viene mostrato un esempio di form per l'inserimento di un nuovo contenuto:

Figura 6.1: Esempio form di inserimento

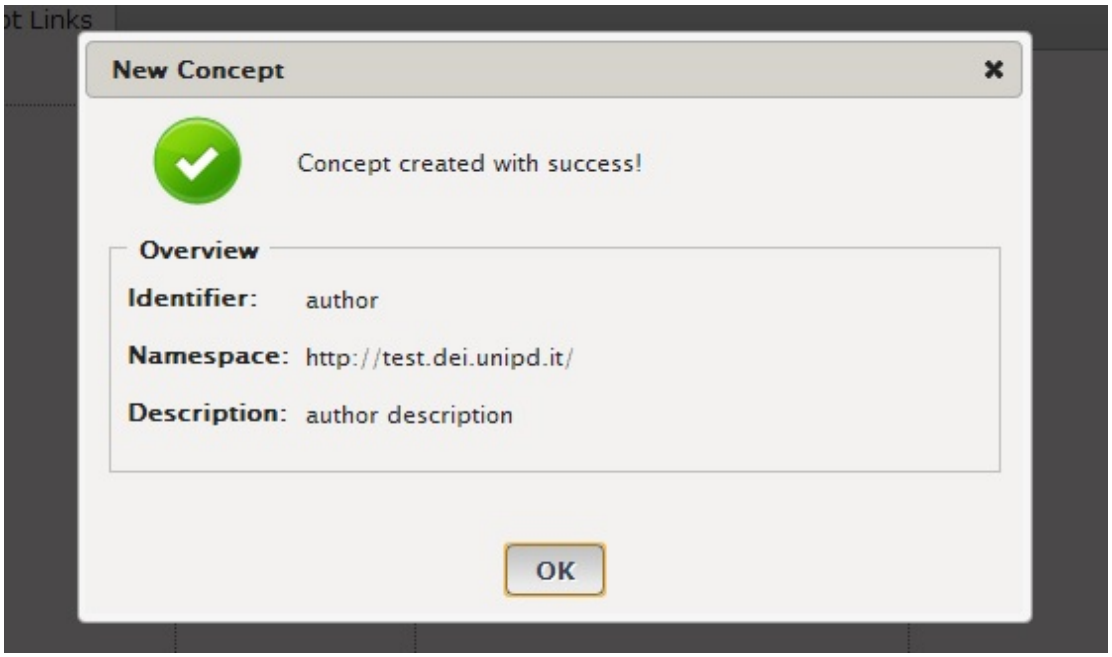


Il codice riportato sopra permette di effettuare una chiamata AJAX verso il Web Service che gestisce i dati ed i metadati del SIAR. Inizialmente vengono recuperati i valori inseriti nei campi del form e successivamente viene costruito un oggetto Javascript che contiene i dati appena recuperati. L'oggetto viene poi convertito, tramite il metodo JSON.stringify(obj), in una stringa che rispetta il formato JSON e, infine, viene costruita la stringa completa che deve essere inviata al Web Service tramite una POST; infatti secondo il paradigma REST una

request di tipo POST corrisponde alla creazione di una nuova risorsa. Il formato JSON è stato scelto per la sua semplicità e per la facilità di manipolazione delle stringhe costruite in questo modo. Il metodo $.ajax, già visto in precedenza, accetta diversi parametri; in questo caso, oltre all'endpoint del Web Service e agli altri valori di base come il tipo di dati, vengono definite anche tre funzioni:

1. beforeSend: permette di modificare la request prima che questa venga inviata al Web Service; in questo caso viene modificato l'Header della request per poter effettuare la Basic Authentication ai Web Services del SIAR;
2. error: viene eseguita nel caso in cui la richiesta non vada a buon fine e consiste nella visualizzazione del messaggio d'errore;
3. success: viene eseguita quanto la richiesta ha successo e di solito viene usata per visualizzare un messaggio di conferma, in modo da notificare all'utente il corretto inserimento dei dati; in questo caso viene visualizzato un messaggio di conferma e un riepilogo dei dati inseriti (vedi Figura 6.2).

Figura 6.2: Messaggio di conferma dell'inserimento



6.2.2 Gestione contenuti

6.2.2.1 Visualizzazione

La parte di gestione dei contenuti è quella più complessa e la complessità varia dal tipo di contenuto che deve essere gestito. I dati da mostrare e da modificare vengono visualizzati tramite una tabella, che verrà descritta di seguito. La tabella è paginata, in modo da poter visualizzare i record su più pagine e non su un'unica pagina; l'utente può scegliere quanti risultati visualizzare per ogni pagina. Il motivo per cui è stato deciso di aggiungere questa funzionalità è dovuta alla cardinalità dei dati da visualizzare, infatti se questa risulta essere elevata è possibile decidere quanti dati visualizzare in una singola pagina. Un'altra funzionalità introdotta è costituita da un filtro per selezionare alcuni record della tabella, effettuando una ricerca semplice sui campi dei contenuti mostrati. Il filtro è costituito da un campo di testo (un tag input HTML) che permette di inserire dei valori specifici, visualizzando così

i record che contengono la stringa inserita. Anche questa che risulta essere particolarmente utile nel caso in cui i dati visualizzati abbiano cardinalità elevata e, soprattutto, perchè può risultare dispendioso selezionare tali dati scorrendo manualmente le pagine della tebella. Di seguito viene riportato il codice di esempio per la costruzione di una tabella con paginazione; il metodo `IUI.resource.concept.diplayConceptTableResults()` permette quindi di creare e visualizzare una tabella con le caratteristiche sopra descritte.

```
//used to display a paginated table with the concepts
this.diplayConceptTableResults = function (arrData,pageI) {
  var columnDefs = [
    {key:"Identifier",sortable: true},
    {key:"Namespace"},
    {key:"Description"},
    {key:"Actions"}];

  var numRecordPage = $('#rowsPerPage').val();
  var initialPage = pageI;

  AUI().use('aui-paginator', function(A) {
    var pg = new A.Paginator({
      containers: '.paginator',
      total: arrData.length,
      maxPageLinks: 5,
      rowsPerPage: numRecordPage,
      page: initialPage,
      alwaysVisible: true,
      circular: false,
      on:{changeRequest: function(event) {
        var instance = this;
        var newState = event.state;
        var page = newState.page;
        var limitMin = (page-1)*numRecordPage;
        var limitMax = page*numRecordPage;
        if (limitMax>arrData.length)
          limitMax=arrData.length;
        var arrDataPag=[];
        for (var i=limitMin;i<limitMax;i++){
          arrDataPag.push(arrData[i]);}

        AUI().use("datatable", 'datatable-sort', function (A) {
          var table = new A.DataTable.Base({
            columnset: columnDefs,
            recordset: arrDataPag
          }).plug(A.Plugin.DataTableSort);
          table.render("#tableConcepts");
        });
        instance.setState(newState);
      }
    }}).render();
 });
};
```

Per realizzare la tabella con paginazione sono stati conbinati insieme due Widget messi a disposizione da Alloy UI: la datatable e il paginator. La variabile **pg** definisce il paginator e viene costruito in questo modo: `new A.Paginator`. Per creare correttamente l'istanza del paginator è necessario settare alcuni parametri:

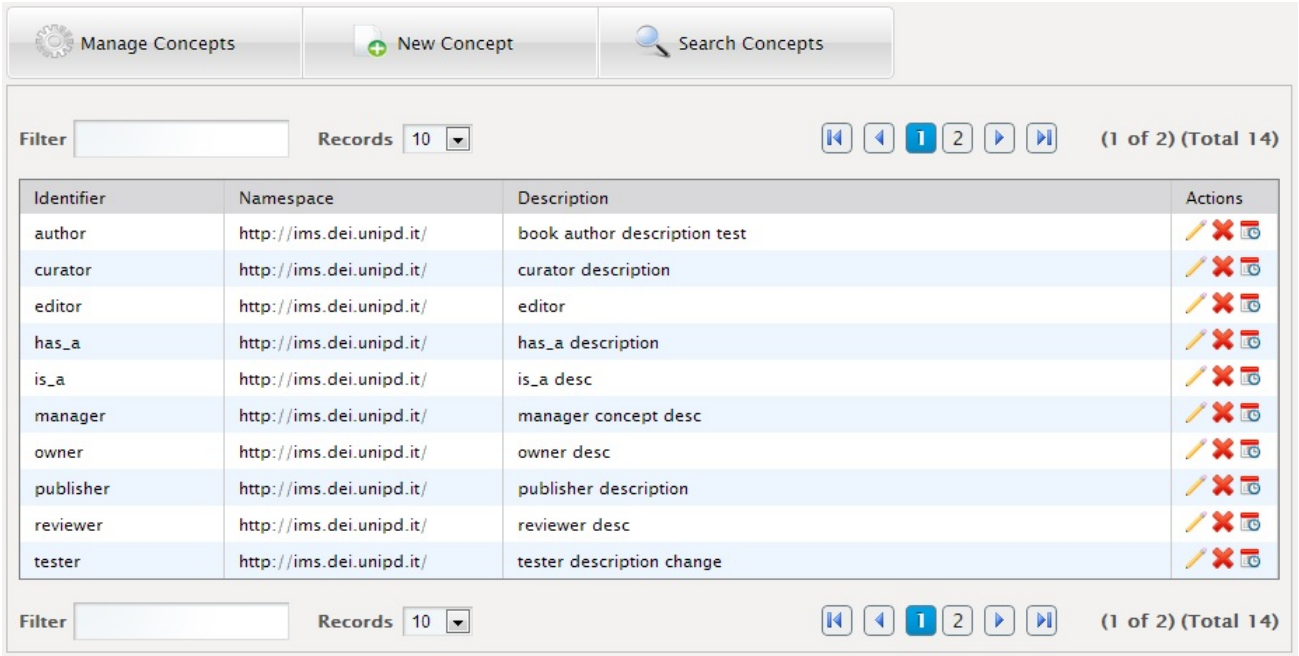
- containers: definisce il contenitore, cioè l'elemento del DOM, all'interno del quale viene poi renderizzato il paginator. Solitamente si usa l'elemento `<div>`. I valori possibili possono essere l'id dell'elemento oppure una classe css associata all'elemento stesso; in

questo caso è stata definita la classe '.paginator', in modo da poter creare più istanze del paginator: la prima di queste viene posizionata in alto, sopra la tabella, mentre la seconda viene messa in basso, sotto la tabella;

- total: questo parametro rappresenta il numero di elementi che vengono paginati, in questo caso viene settato con il numero di elementi dell'array che contiene i record da visualizzare;
- maxPageLinks: questo parametro rappresenta il numero di pagine visibili e cliccabili nel paginator;
- page: indica quale pagina viene visualizzata di volta in volta;
- on: permette di definire delle funzioni personalizzate che vengono eseguite in base al tipo di evento che si verifica. Per le esigenze di questa tabella è stata definita una funzione associata all'evento di changeRequest, cioè all'evento che viene attivato ogni volta che l'utente desidera visualizzare una pagina del paginator. All'interno di questa funzione viene costruita la tabella che visualizza i contenuti. Per la creazione dell'istanza della tabella (già vista in precedenza): `new A.DataTable.Base`. I parametri sono costituiti da un array contenente i record (recordset) che appartengono alla pagina selezionata e un secondo array (columnset) con la definizione delle colonne della tabella stessa. L'array contenente i dati della tabella viene popolato di volta in volta con le risorse relative alla pagina mostrata all'utente.

Il metodo usato per renderizzare sia il paginator che la datatable è `render()` e deve essere invocato dopo la definizione delle istanze dei due Widget in questione.

Figura 6.3: Esempio di tabella con paginazione

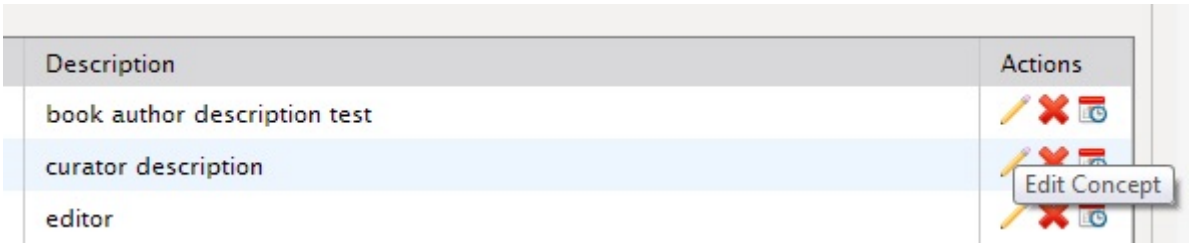


6.2.2.2 Modifica

La funzionalità di modifica di un contenuto già presente all'interno del SIAR può essere attivata cliccando, nella riga oppurtuna, sull'icona che rapprensenta l'edit della risorsa (in questo caso è stato usato come simbolo una matita) nella colonna Actions della tabella (Figura 6.4).

L'azione eseguita permette di aprire una finestra popup, un dialog JQuery che consente all'utente di modificare la risorsa selezionata. L'interfaccia di modifica è equivalente a quella di

Figura 6.4: Funzionalità di modifica



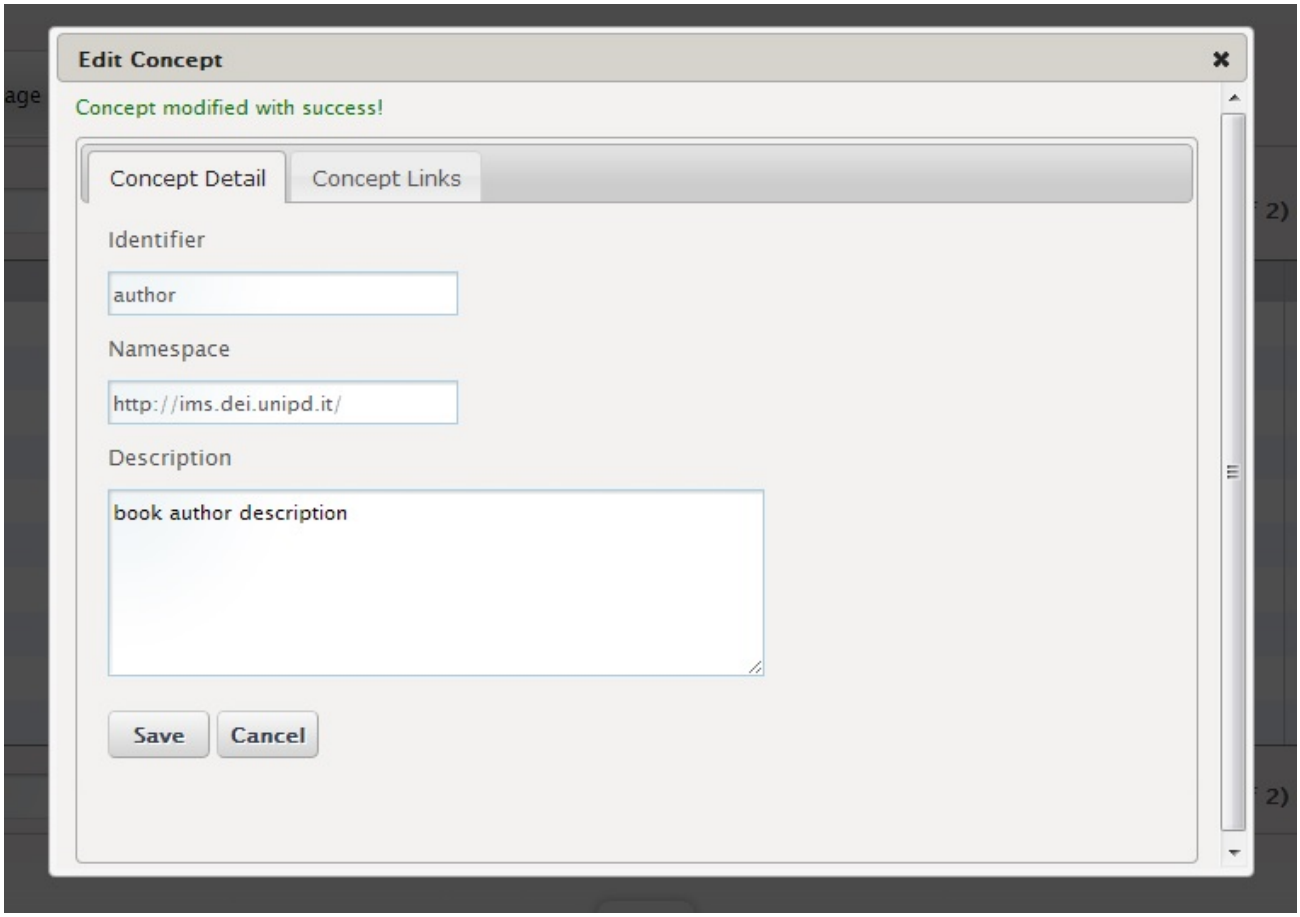
inserimento ed è costituita principalmente da un form con i campi modificabili. Di seguito viene riportato il codice della chiamata AJAX usato per effettuare la modifica di una risorsa (il resto del metodo di modifica è analogo a quello di inserimento):

```
$.ajax({
  url : url,
  type : 'PUT',
  contentType: "application/json",
  data: concept,
  dataType: "json",
  crossDomain: true,
  beforeSend : function(req) {
    req.setRequestHeader('Authorization', hash);
  },
  error : function(xhr, ajaxOptions, thrownError) {
    ...
  },
  success : function(data) {
        ...
    //find the index array of the object which is modified
    var i=0;
    var indexObject=0;
    var arrDataGrep=conceptsArray;
    arrDataGrep = $.grep(arrDataGrep, function (o) {
      i++;
      if(concept['identifier']==o.Identifier
        && concept['namespace']==o.Namespace){
        indexObject=i-1;
        return true;
      }
      else
        return false;
    });
    //replace the object in the array,
    //in order to avoid the ajax call to the Web service
    conceptsArray.splice(indexObject,1,obj);
    ...
  }
});
```

Il metodo HTTP per modificare una risorsa, secondo il paradigma REST, è il PUT. All'interno del metodo `success` della chiamata AJAX è presente il frammento di codice che permette di evitare di effettuare la richiesta al per recuperare i dati da visualizzare, se la chiamata è andata a buon fine. Infatti, dopo aver effettuato una modifica l'applicazione ritorna alla situazione precedente e ricostruisce la tabella; i dati da visualizzare nella tabella vengono memorizzati in un array di oggetti, dichiarato come variabile globale, e la richiesta al Web Service viene effettuata solo quando la pagina della Portlet viene caricata dal Browser per la prima volta. L'array, successivamente, viene manipolato sostituendo l'oggetto relativo alla risorsa modificata,

evitando così di peggiorare le prestazioni dell'applicazione. Questo è reso possibile grazie al fatto che il Web Service restituisce la nuova risorsa modificata, in risposta alla richeista di modifica effettuata con il metodo PUT.

Figura 6.5: Esempio di modifica di una risorsa



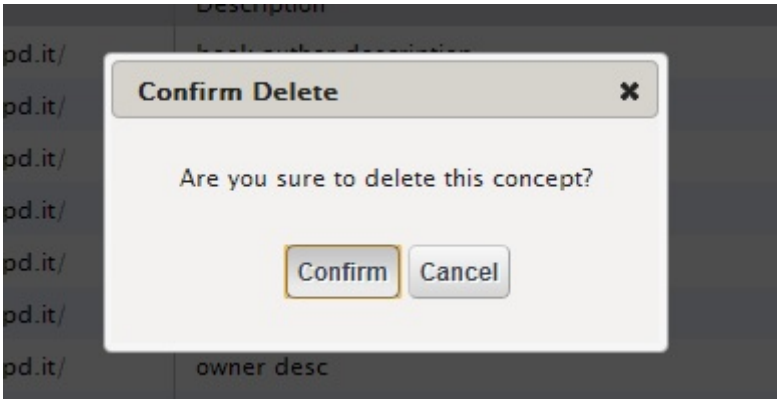
6.2.2.3 Cancellazione

La funzionalità di cancellazione viene attivata in maniera analoga a quella di modifica. L'unica differenza con la precedente sta nel fatto che in questo caso viene aperta solo una finestra popup per richiedere all'utente la conferma dell'eliminazione della risorsa (Figura 6.6) e successivamente viene visualizzato un messaggio per notificare l'avvenuta eliminazione della risorsa oppure un messaggio d'errore, se l'operazione riscontra delle problematiche.

Esempio del codice per la cancellazione:

```
$.ajax({
  url : url,
  type : 'DELETE',
  contentType: "application/json; charset=utf-8",
  dataType: "json",
  crossDomain: true,
  beforeSend : function(req) {
    req.setRequestHeader('Authorization', hash);
  },
  error : function(xhr, ajaxOptions, thrownError) {
    ...
  },
  success : function(data) {
```

Figura 6.6: Conferma di cancellazione




```
    ...
    //find the index array of the object which is modified
    var i=0;
    var indexObject=0;
    var arrDataGrep=conceptsArray;

    arrDataGrep = $.grep(arrDataGrep, function (o) {
      i++;
      if (concept['identifier']==o.Identifier
           && concept['namespace']==o.Namespace){
        indexObject=i-1;
        return true;
      }
      else
        return false;
    });

    //replace the object in the array,
    //in order to avoid the ajax call to the Web service
    conceptsArray.splice(indexObject,1);
    var numRecordPage = 5;
    var pageI=Math.ceil((conceptsArray.length/numRecordPage));
    if (pageI < objectConceptPage)
      objectConceptPage=pageI;
    Concept_tableUtilities.
      diplayConceptTableResults(conceptsArray,objectConceptPage);
    ...
  }
});
```

Anche in questo caso per ricostruire la tabella con i dati da mostrare all'utente non viene effettuata una nuova chiamata al Web Service ma viene manipolato l'array globale, eliminando l'oggetto relativo alla risorsa appena eliminata.

6.2.2.4 Gestione di contenuti con una struttura ad albero

Come è stato descritto nei capitoli precedenti, molte componenti di un sistema informativo archivistico vengono rappresentate tramite una struttura gerarchica ad albero. Infatti, per gestire correttamente alcune risorse del SIAR si è dovuto provvedere a sviluppare una struttura ad albero con alcune funzionalità specifiche, tra cui il Drag & Drop; il Drag & Drop permette di copiare i nodi dell'albero all'interno dell'albero stesso, oppure all'interno di alberi diversi, cliccando sul nodo da muovere e trascinandolo nella posizione desiderata. Per costruire un

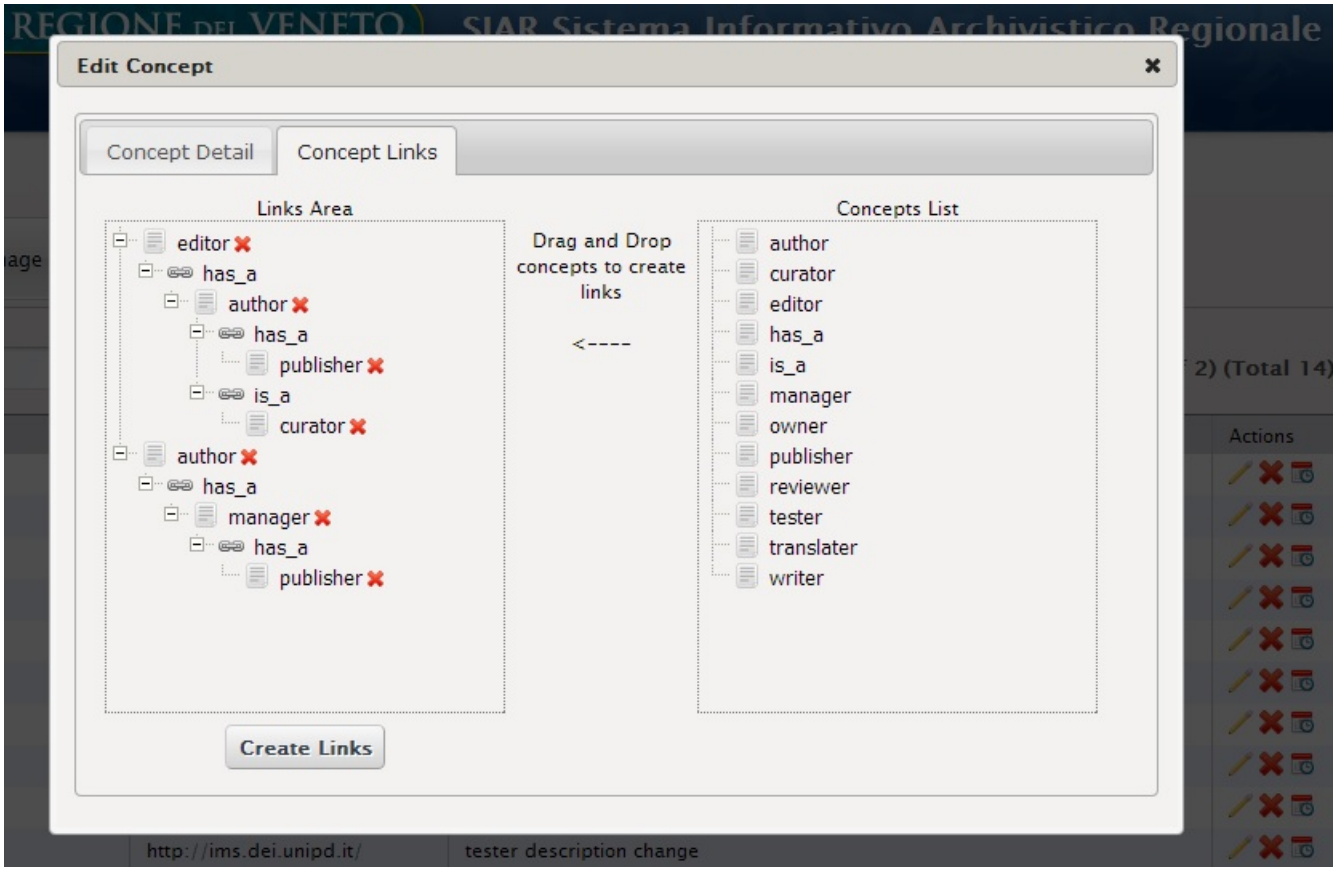
albero di questo tipo è stato utilizzato JQuery.dynatree.js, un plugin di JQuery sviluppato da Martin Wendt; è stata fatta questa scelta allo scopo di evitare di progettare e sviluppare ex novo una componente già presente. Il plugin sopra citato è flessibile e dispone di molte opzioni che permettono di personalizzare l'albero creato. Di seguito viene riportato il codice di un albero costruito nell'applicazione e nella figura 6.7 viene mostrato il risultato.

```
$("#conceptsLinksTree").dynatree({
  persist: false,
  imagePath: "/Concept-portlet/img/",
  children: children,
  onClick: function(node,event) {
  ...
  },
  dnd: {
    onDragStart: function(node) {
      //This function MUST be defined to enable dragging for the tree.
      //Return false to cancel dragging of node.
      if(node.data.addClass == 'conceptList' || node.data.addClass == 'concept')
        return true;
      else
      return false;
    },
    onDragEnter: function(node, sourceNode) {
      //sourceNode may be null for non-dynatree droppables.
      //Return false to disallow dropping on node. In this case
      //onDragOver and onDragLeave are not called.
      //Return 'over', 'before, or 'after' to force a hitMode.
      //Any other return value will calc the hitMode from the cursor position
      if(node.data.addClass == 'concept' && !node.isDescendantOf(sourceNode))
        return true;
      else
        return false;
    },
    onDrop: function(node, sourceNode, hitMode, ui, draggable) {
    //This function MUST be defined to enable dropping of items on the tree.
    //sourceNode may be null, if it is a non-Dynatree droppable.
    ...
    ...
    }
  }
});
```

Per creare il l'albero è necessario utilizzare la funzione `$(selector).dynatree(options)`. Il metodo accetta in input un oggetto contenente le varie opzioni per la costruzione dell'albero; tra quelle disponibili, sono state utilizzate le seguenti :

- imagePath: rappresenta il percorso dove sono memorizzate le immagini utilizzate nell'albero;
- children: è l'array contenente i nodi dell'albero. Ogni nodo (DynaTreeNode) è rappresentato da un oggetto Javascript e i suoi principali attributi sono:
  - title: rappresenta l'etichetta del nodo dell'albero, può essere testo semplice oppure codice HTML;
  - tooltip: il tooltip del nodo;
  - isFolder: indica se il nodo rappresenta una cartella oppure un documento semplice;
  - expand: indica se il nodo è espanso o meno quando l'albero viene caricato la prima volta;

Figura 6.7: Esempio di albero



  - icon: il nome dell'immagine del nodo;
  - addClass: la classe css per associare al nodo gli stili;
  - children: un array contenente i sottonodi;

- onClick: una funzione di callback che viene eseguita quando un nodo viene cliccato;
- dnd: è l'oggetto che definisce le caratteristiche del Drag & Drop; è costituito da una serie di funzioni, tra cui:
  - onDragStart: è la funzione che permette di abilitare o meno il Drag di un nodo o di un gruppo di nodi e deve essere definita obbligatoriamente per consentire il corretto funzionamento di questa funzionalità;
  - onDragEnter: questa funzione permette ad un nodo di accettare o meno il Drop, con la possibilità di forza la posizione in cui mettere il nodo che viene spostato o copiato ('over', 'before, or 'after');
  - onDrop: questa funzione è obbligatoria e deve essere definita per abilitare il Dropping su un nodo e su un gruppo di nodi.

Il plugin in esame fornisce, inoltre, una serie di metodi per accedere e gestire sia l'intero albero che un singolo nodo. Di seguito ne vengono analizzati alcuni. Alcuni tra i metodi messi a disposizione dal plugin in esame:

- `$("#tree").dynatree("option", )`: permette di modificare le opzioni. Ad esempio:
  - `$("#tree").dynatree("option", "autoCollapse", true);`

  - `$("#tree").dynatree("option", "fx", { height: "toggle", duration: 200 });`

- `$("#tree").dynatree("getTree")`: restituisce l'oggetto DynaTree, che rapprensenta l'albero.
- `$("#tree").dynatree("getRoot")`: restituisce il nodo radice, rappresentato da un oggetto DynaTreeNode.
- `$("#tree").dynatree("getActiveNode")`: restituisce l'oggetto DynaTreeNode che rappresenta il nodo attivo.
- `$("#tree").dynatree("getSelectedNodes")`: restituisce un array di oggetti DynaTreeNode che rapprensentano i nodi selezionati.

Di seguito, invece, vengono elencati alcuni metodi che permettono di manipolare l'albero, tramite l'oggetto DynaTree:

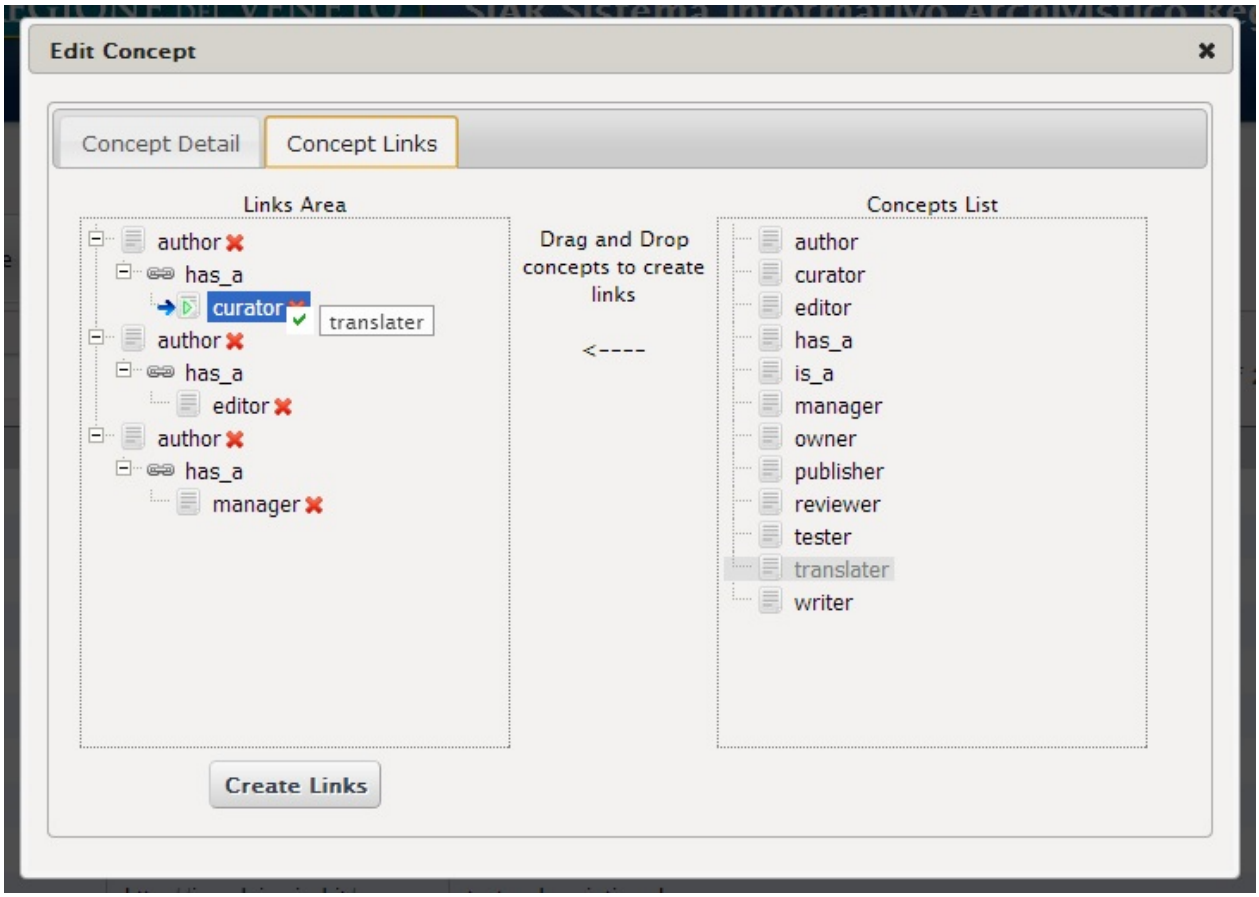
- `tree.getNodeByKey(key)`: restituisce un nodo dell'albero data la sua chiave; se non viene trovato alcun nodo relativo alla chiave il metodo resituisce null.
- `tree.reload()`: ricarica l'intero albero.
- `tree.toDict()`: converte l'albero in un oggetto Javascript.
- `tree.visit(fn, includeRoot)`: consente di visitare i nodi dell'albero. Deve essere definita la funzione di callbak `fn(node)`. L'iterazione viene fermata se `fn()` restituisce false, mentre viene fermata nel ramo corrente se `fn()` resituisce 'skip'.

Di seguito, invece, vengono elencati alcuni metodi che permettono di manipolare i singoli nodi, rappresentati da oggetti DynaTreeNode:

- `node.addChild(nodeData[, beforeNode])`: consente di aggiungere un nuovo nodo.
- `node.getChildren()`: resituisce un lista di nodi figli del nodo corrente, se esistono, altrimenti resituisce null.
- `node.makeVisible()`: rende il nodo corrente visibile, eventualmente espandendo tutti i suoi genitori.
- `node.move(targetNode, mode)`: permette di spostare un nodo in una determinata posizione, come figlio. Il parametro mode rappresenta la modalità con la quale il nodo da spostare verrà legato con il nodo destinazione:
  - child: il nodo da muovere viene aggiunto come figlio del nodo target;
  - before: il nodo da muovere viene aggiunto come fratello sinistro del nodo target;
  - after: il nodo da muovere viene aggiunto come fratello destro del nodo target.
- `node.remove()`: rimuove il nodo corrente e tutti i suoi discendenti.

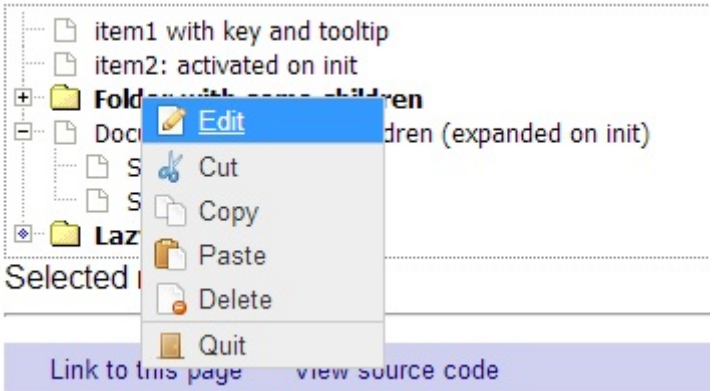
La funzionalità di Drag and Drop tra due alberi diversi viene mostrata nella figura 6.8, dove viene copiato un nodo dall'albero di destra all'interno di quello di sinistra. Quando viene effettuata un'operazione di questo tipo, è possibile copiare il nodo in tre diverse posizioni: prima, dopo oppure come figlio del nodo dell'albero di destinazione.

Figura 6.8: Drag and Drop



E' disponibile inoltre anche la funzionalità di menù contestuale, attivabile cliccando con il tasto destro del mouse sul nodo che deve essere modificato (figura 6.9).

Figura 6.9: Menù contestuale



Ogni nodo dell'albero può contenere un qualsiasi tipo di contenuto, come visibile dalla figura 6.10. Inoltre i nodi possono anche essere associati a radiobuttun o a delle checkbox, in modo da poter selezionare quelli di interesse (figura 6.11).

Figura 6.10: Contenuti personalizzabili

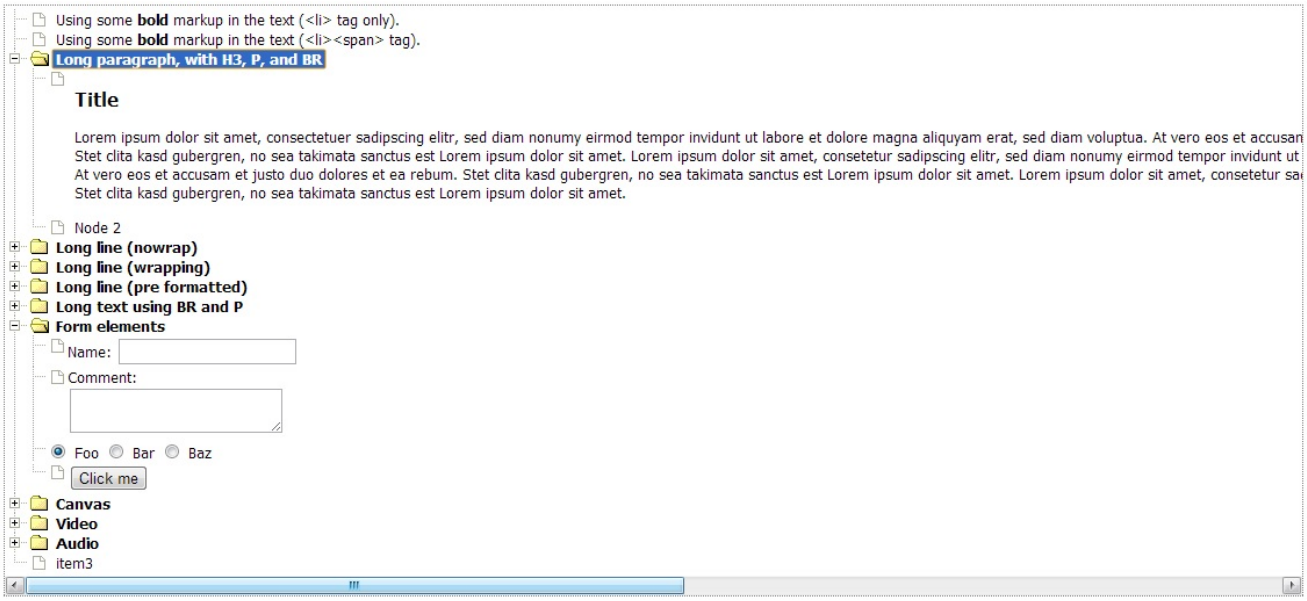
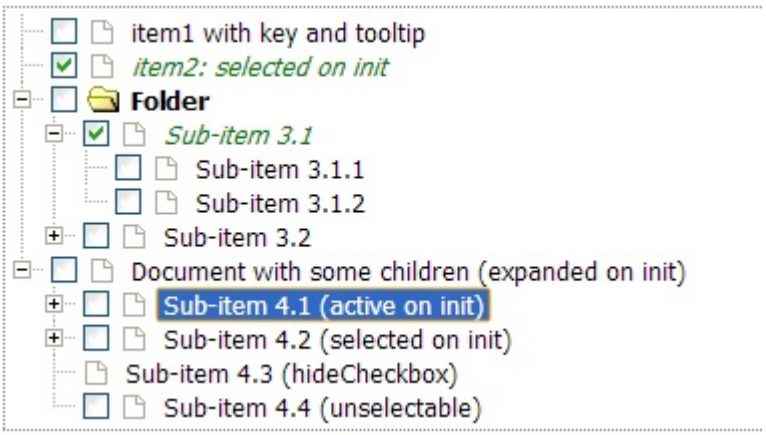


Figura 6.11: Checkbox nei nodi



Come visto in precedenza, i metodi e le opzioni disponibili sono numerosi e permettono di ottenere un albero completamente personalizzato.[4]

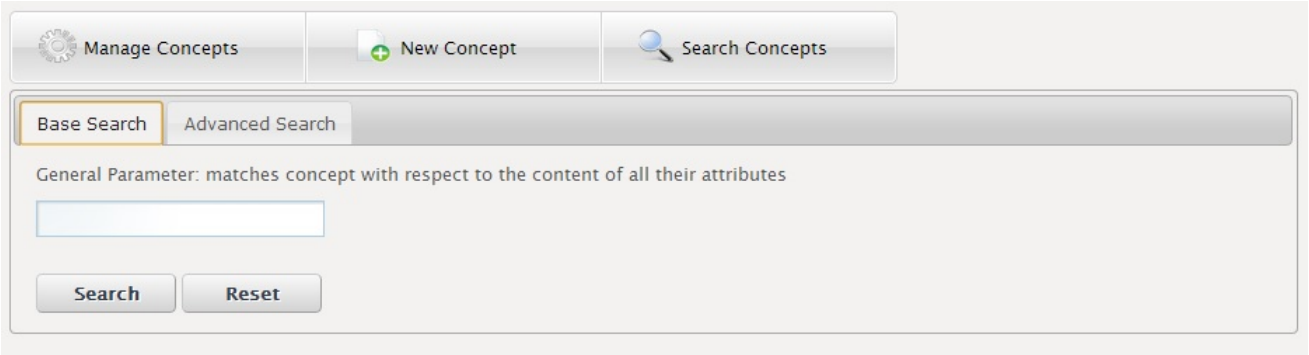
[4]L'elenco completo delle funzionalità e la guida sono disponibili all'indirizzo *http://wwwendt.de/tech/dynatree/doc/dynatree-doc.html.*

6.2.3 Ricerca contenuti

La terza componente di ogni portlet permette di usufruire di una funzionalità di ricerca avanzata. Sono stati sviluppati due tipologie di ricerca:

- ricerca generale: permette di effettuare una ricerca generica su tutti i gli attributi della risorsa o delle risorse che si vogliono trovare; di seguito viene riportato il frammento di codice per effettuare questo tipo di ricerca:

Figura 6.12: Ricerca generale




```
//general search
this.searchGeneral = function(value){

  var hash = IUI.resources.concept.createAuth();
  var url = this.connectionURI + 'list/ici.concept.general='+value;

  return $.ajax({
    url : url,
    type : 'GET',
    contentType: "application/json;
    charset=utf-8",
    dataType: "json",
    crossDomain: true,
    beforeSend : function(req) {
      req.setRequestHeader('Authorization', hash);
    },
    error : function(xhr, ajaxOptions, thrownError) {
      ...
    }
  });
};
```

- ricerca avanzata: permette di effettuare una ricerca più specifica; l'utente può costruire la query da effettuare aggiungendo i parametri sui quali vuole effettuare la ricerca. Di seguito viene riportato il frammento di codice per effettuare questo tipo di ricerca; la query viene costruita concatenando i vari parametri recuperati dal form e usando il linguaggio messo a disposizione dal Web Service:

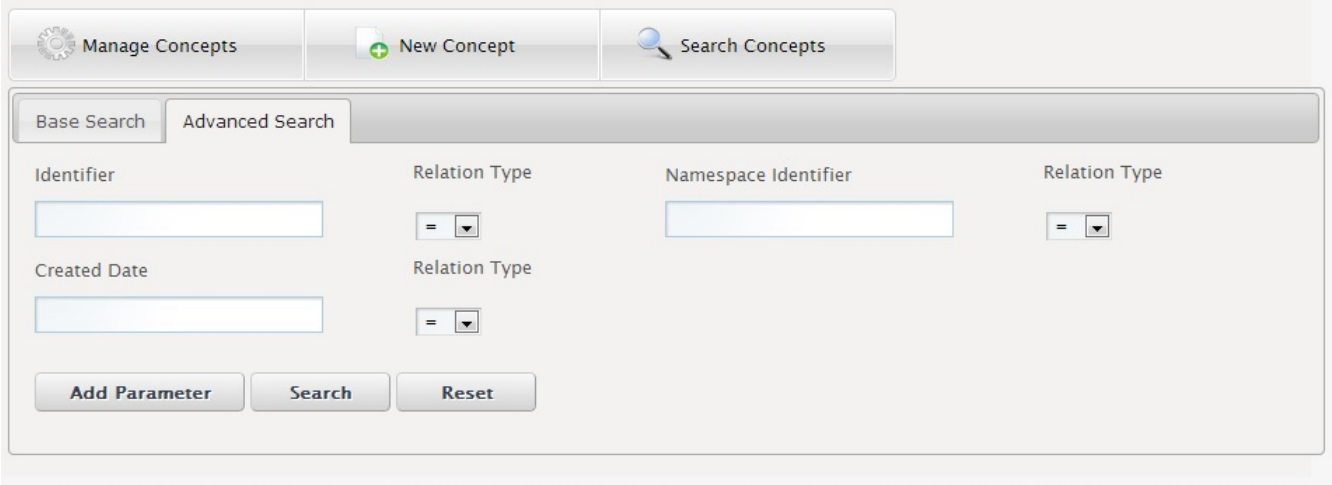
```
//search by a parameter (identifier, prefix, etc)
this.searchByParameters = function(){

  var hash = IUI.resources.concept.createAuth();
  var url = this.connectionURI + 'list/';

  if (paramCount > 0){
    for(var i=1;i<=paramCount;i++){
```

```
      if ($('#param'+i+'Concept').val().length > 0){
      if (i > 1)
        url += ' OR ';
      url += $('#paramHidden'+i+'Concept').val();
      url += $('#relTypeSearch'+i+'Concept').val();
      if ($('#paramHidden'+i+'Concept').val() != 'ici.concept.description')
        url += '*';
      url += $('#param'+i+'Concept').val();
      if ($('#paramHidden'+i+'Concept').val() != 'ici.concept.description')
        url += '*';
      }
    }
  }
  return $.ajax({
    url : url,
    type : 'GET',
    contentType: "application/json",
    dataType: "json",
    crossDomain: true,
    beforeSend : function(req) {
      req.setRequestHeader('Authorization', hash);
    },
    error : function(xhr, ajaxOptions, thrownError) {
      ...
    }
  });
};
```

Figura 6.13: Ricerca avanzata



Di seguito vengono riportate un paio di esempi di query.

In figura 6.14 viene mostrato un esempio di query generica, dove request che contiene la query è:

`http://svrims.dei.unipd.it:8080/deboni/list/ici.concept.general=author`.

In figura 6.15 è riportato un esempio di ricerca avanzata, utilizzando quattro parametri diversi tra quelli disponibili. La request contente la query è così composta:

`http://svrims.dei.unipd.it:8080/deboni/list/ici.concept.identifier=*author*%20OR%20ici.concept.namespace.identifier=*test*%20OR%20ici.concept.source.identifier=*curator*%20OR%20ici.concept.target.relation.identifier=*has_a*`

Questa query permette di effettuare una ricerca avanzata, recuperando tutti i concept che contengono la stringa *author* nell'attributo ***identifier***, oppure la stringa *test* nell'attributo ***namespace identifier***, oppure la string *curator* come ***source identifier***, o, infine, la stringa *has_a* come ***target link identifier***. Ogni risorsa gestita dal SIAR è composta da determinati attributi e il Web Service che gestisce tali dati mette a disposizione un linguaggio per effettuare le query sugli attributi, come è possibile notare dalla query precedente.

Figura 6.14: Esempio di query generica

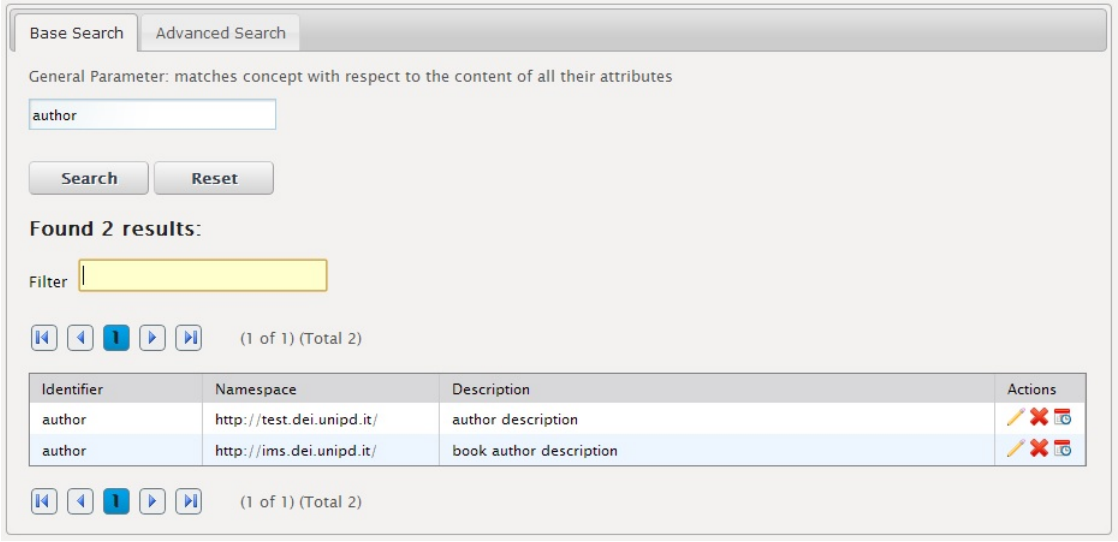
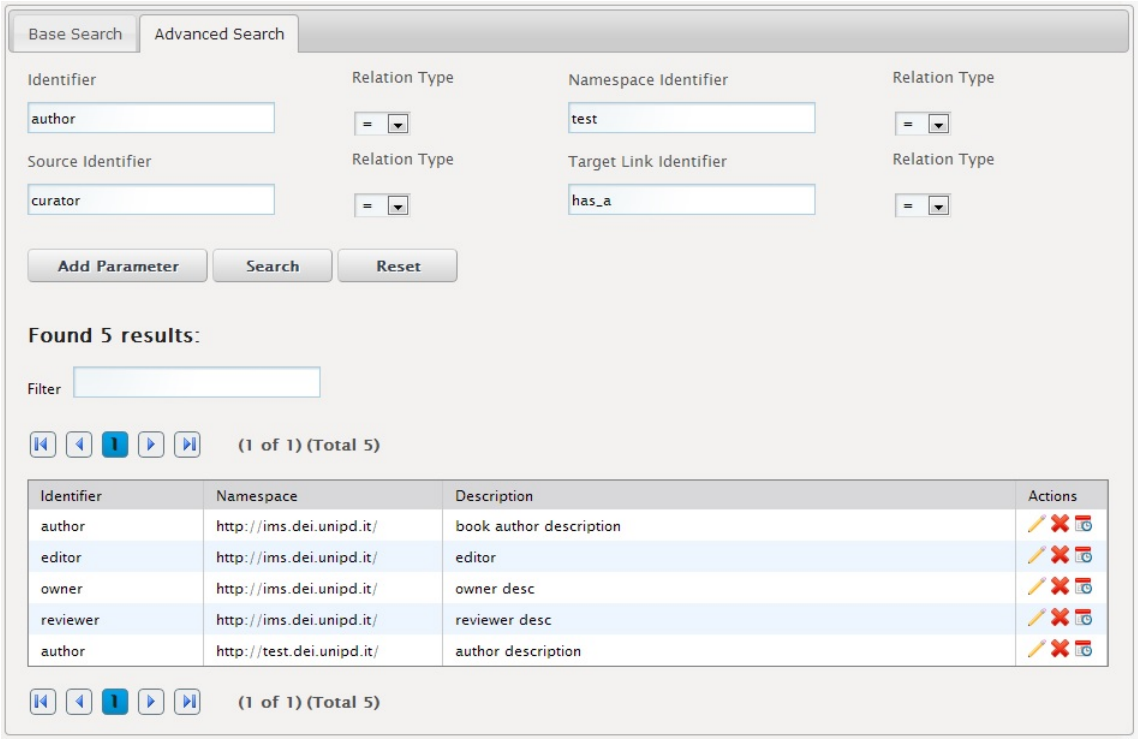


Figura 6.15: Esempio di query avanzata



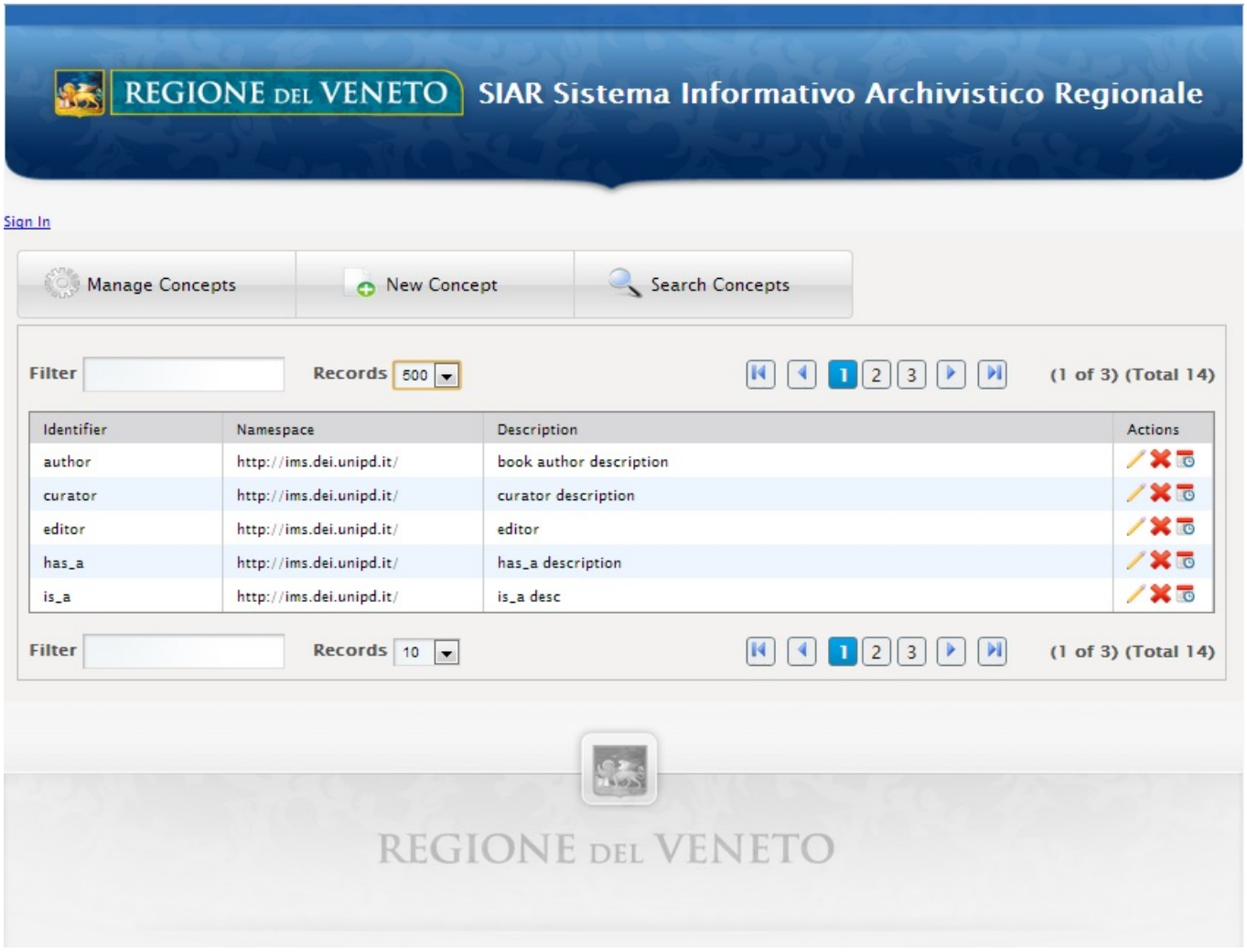
6.3 Tema grafico del portale

I temi Liferay sono organizzati in una struttura modulare che consente allo sviluppatore di modificare rapidamente l'aspetto grafico di una singola portlet o di un qualsiasi oggetto presente su una qualsiasi pagine del portale. Non è necessario personalizzareo modificare ogni singolo aspetto del tema in quanto esso eredita le immagini, gli stili e i template dai temi di default di Liferay. I temi personalizzati si basano sulle differenze rispetto a temi di default di Liferay.

Il tema creato per il SIAR è stato sviluppato a partire da un tema di base già presente: il tema base per i portali della Regione Veneto. Il problema con questo tema sta nel fatto che è stato costruito per Liferay 5.x e quindi non risulta compatibile con le versioni 6.x di Liferay, in quanto con le nuove versioni del portale sono cambiate alcune cose.

Nella figura 6.16 è mostrata la schermata iniziale della nuova interfaccia grafica del SIAR. L'header della pagina è composto dal logo della Regione Veneto e dal nome esteso del SIAR. Nella parte centrale vengono posizionate le Portlet, nel caso in esame è presente solo una Portlet. Come già detto in precedenza, ogni Portlet che gestisce una risorsa del SIAR è costituita da tre funzionalità: gestione, nuovo contenuto e ricerca. Anche il footer è costituito dal logo regionale. L'header e il footer sono stati modificati rispetto al tema originale riducendo gli spazi per rendere più compatta la pagina.

Figura 6.16: Schermata del SIAR



La struttura del tema sviluppato è la seguente:

```
/ TemaRegioneVeneto6.1-theme /
  / Docroot /
    / WEB-INF /
      liferay-plugin-package.properties
      liferay-look-and-feel.xml
    / _diffs /
    / Css /
      / Images /
      application.css
      base.css
      custom.css
      dockbar.css
      dropdown.css
      extras.css
      forms.css
      jquery-ui-1.9.1.custom.css
      JQuery.loadmask.css
      layout.css
      main.css
      navigation.css
      portlet.css
    / js /
    / Images /
      ...
    / Img /
    / Tema /
      ...
    / Templates /
      init_custom.vm
      footer-info.vm
      footer-nav.vm
      navigation.vm
      portal_normal.vm
      portal_pop_up.vm
      portlet.vm
      search_header.vm
```

All'interno della cartella docroot del tema è presente la cartella _diffs dove vanno inserite tutte le personalizzazioni, modificando esclusivamente le parti del tema che differiscono dal tema principale. Per fare questo è necessario copiare la struttura della directory del tema principale dentro la cartella _diffs, ponendo solo le cartelle e i file necessari alla personalizzazione. Per esempio, se si vuole sostituire un'icona di default (ad esempio l'immagine search.png) è necessario ricreare il percorso relativo dentro la cartella _diffs, cioè /images/common/search.png.

All'interno della cartella css si trovano gli stili .css del tema. I file di default sono i seguenti:

- application.css: contiene lo stile predefinito relativi ai componenti delle applicazioni, tra cui le Tabs, gli alberi, i Dialag ecc.;
- base.css: contiene lo stile predefinito per i principali tag HTML, come i paragrafi, i titoli, le tabelle ecc.. Questo file contiene anche lo stile per gli elementi specifici di Liferay, come errori, avvisi, ecc.;

- dockbar.css: contiene lo stile predefinito per la Dockbar, che si trova nella parte superiore della pagina quando l'utente è loggato;
- custum.css: per personalizzare il tema è sufficiente modificare il file custom.css aggiungendo nuovi stili o sovrascrivendo quelli del tema principale. Si raccomanda di non modificare gli altri stili specifici del portale (application.css, dockbar.css, ecc); questo rende più facile l'aggiornamento del portale in quanto basta sostituire solamente questo file;
- forms.css: contiene lo stile predefinito per gli elementi dei form;
- layout.css: contiene lo stile predefinito utilizzato dai modelli di layout;
- main.css: non contiene codice css, ma viene usato per importare tutti gli altri file;
- navigation.css: contiene lo stile predefinito per i principali elementi di navigazione;
- portlet.css: contiene lo stile predefinito per il layout delle Portlet.

All'interno della cartella Templates ci sono i file Velocity Macro, con estensione .vm. Apache Velocity[5] è strumento di sviluppo utilizzato per generare codice ed è utilizzato in ambiente Java, perchè sviluppato nello stesso linguaggio e quindi integrabile all'interno delle applicazioni Java. Un generatore di codice è un strumento che consente di definire delle regole (un template) per la generazione finale di un documento (in questo caso le pagine HTML del portale) con informazioni che possono cambiare dinamicamente. I template presenti di default:

- init_custom.vm: consente di aggiungere variabili personalizzate di Velocity;
- navigation.vm: implementa la pagina di navigazione all'interno del tema.
- portal_normal.vm: il template generale che viene applicato a tutte le pagine del portale;
- portal_pop_up.vm: il modello per le Portlet che usano le finestre pop-up personalizzate di Liferay.
- portlet.vm: il template per costruire le finestre delle Portlet.

Di seguito viene riportato come esempio un frammento di codice del file portal_normal.vm:

```
<!DOCTYPE html>
#parse ($init)
<html class="#language("lang.dir")"
   dir="#language("lang.dir")" lang="$w3c_language_id">
  <head>
    <title>$the_title - $company_name</title>
    <link href="/Tema6.1-theme/css/jquery-ui-1.9.1.custom.css"
      rel="stylesheet" type="text/css" />
      ...
    <script type="text/Javascript"
      src="$Javascript_folder/jquery-1.8.2.js"></script>
      ...
    $theme.include($top_head_include) </head>
  <body class="$css_class">
  $theme.include($body_top_include)
  #if ($is_signed_in) #dockbar() #end
  <div id="wrapper">
    <div id="head-topbar">
      ...
    </div>
    <div id="head-topbar-title-siar">
      <h1>SIAR Sistema Informativo Archivistico Regionale</h1>
      ...
```

[5] http://velocity.apache.org/

```
    </div>
    #if ($has_navigation)
    <div id="head-menu">
    #parse ("$full_templates_path/navigation.vm")
    </div>
    #end
    ...
    #if($permissionChecker.isCompanyAdmin()
      || $permissionChecker.isCommunityAdmin( $portletGroupId ))
   <div id="content" class="clearfix">
   #else
   <div id="content" class="clearfix not-admin">
   #end
   #if ($selectable)
   $theme.include($content_include)
   #else
   $portletDisplay.recycle()
   $portletDisplay.setTitle($the_title)
   $theme.wrapPortlet("portlet.vm", $content_include)
   #end
   ...
...
```

Ad esempio la riga di codice `#if ($is_signed_in) #dockbar() #end` permette di visualizzare nelle pagine del portale la dockbar, se l'utente ha effettuato il login. Invece, la riga `#parse ("$full_templates_path/navigation.vm")` permette di includere il template navigation.vm.

7 Conclusioni

Lo scopo della tesi riguardava la reingegnerizzazione di una parte dell'attuale sistema informativo archivistico regionale del Veneto; in particolare l'attività svolta si è focalizzata sulla progettazione, sull'analisi ed infine sullo sviluppo della nuova interfaccia Web 2.0 del SIAR.

La prima fondamentale attività portata avanti è stata quella di analisi ed ha riguardato le tematiche e i concetti che hanno interessato questa tesi. Sono stati introdotti le Digital Libraries e gli archivi, attraverso un breve panoramica sulla letteratura degli argomenti. Infatti, il SIAR costituisce sia un archivio che una tipologia particolare di Digital Library, un sistema informativo archivistico. L'analisi è continuata con la descrizione dei principali sistemi informativi archivistici sviluppati ed utilizzati in Italia e con l'introduzione di alcuni parametri di confronto, utili per mettere in evidenza i punti di forza ed i limiti di ogni sistema. Il passo successivo è stato quello di introdurre brevemente il vecchio sistema SIAR, evidenziandone le caratteristiche principali e l'architettura.

L'attività successiva svolta è stata quella di progettazione. Il lavoro di questa tesi, come già accennato, si occupato di rivedere l'attuale interfaccia utente del SIAR, basata su pagine HTML, verso un nuovo sistema basato sulla piattaforma Open Source Liferay (portale Web) e sul paradigma delle Portlet. La prima fase dell'attività di progettazione si è svolta effettuando una panoramica sulle tecnologie utilizzate allo scopo. E' stato descritto e analizzato in dettaglio l'ambiente di lavoro utilizzato per la progettazione e lo sviluppo della nuova interfaccia utente del SIAR. Sono stati introdotti i concetti di portale Web e di REST Web Service, le due componenti fondamentali della nuova architettura dell'interfaccia Web 2.0. Successivamente sono stati spiegati in dettaglio il portale Web Liferay e le Portlet. La prima fase si è conclusa descrivendo i linguaggi lato Client (Javascript, JQuery, JQuery UI, Alloy UI) usati durante lo sviluppo del codice. La seconda fase, invece, si è focalizzata sull'analisi dei motivi che hanno portato alle scelte effettuate e sui vantaggi che ne sono derivati. Per concludere l'attività è stata effettuata un ulteriore analisi del nuovo SIAR 2.0, basata sui parametri di confronto introdotti nel capitolo 2.

L'ultima attivita svolta è stata quella di sviluppo dell'interfaccia Web 2.0 del SIAR. Il prototipo dell'interfaccia del SIAR, realizzata attraverso lo sviluppo di alcune Portlet. Viene introdotta la struttura di base che viene condivisa da ogni Portlet e vengono illustrati i principali dettagli implementativi per giustificare le scelte fatte in precedenza. Ogni Portlet è basata su tre componenti principali: **Nuovo contenuto/risorsa**, che permette di inserire all'interno del sistema SIAR un nuovo contenuto (che varia in base al tipo di Portlet), la **Gestione contenuti/risorse**, che permette di visualizzare e modificare le risorse già presenti nel sistema e la parte di **Ricerca contenuti/risorse**, che fornisce le funzionalità di ricerca e recupero delle risorse memorizzate e mantenute dal SIAR, con la possibilità di effettuare query personalizzate e selettive. Nella sezione finale del capitolo viene descritta la procedura usata per la progettazione e lo sviluppo del tema usato nel portale.

I lavori futuri comprendono lo sviluppo di tutte le componenti del SIAR rimanenti, come le Portlet che devono gestire e visualizzare i produttori, i consumatori ecc. Inoltre, potrebbero presentarsi nuove richieste od esigenze e quindi anche l'interfaccia grafica potrebbe subire nuove evoluzioni.

Bibliografia

Maristella Agosti. Digital Libraries. In Melucci and Baeza-Yates, editors, *Advanced Topics in Information Retrieval*, pages 1–27. Springer, 2009.

Maristella Agosti, Stefano Berretti, Gert Brettlecker, Alberto Bimbo, Nicola Ferro, Norbert Fuhr, Daniel Keim, Claus-Peter Klas, Thomas Lidy, Diego Milano, Moira Norrie, Paola Ranaldi, Andreas Rauber, Hans-Jorg Schek, Tobias Schreck, Heiko Schuldt, Beat Signer, and Michael Springmann. DelosDLMS - The Integrated DELOS Digital Library Management System. In Costantino Thanos, Francesca Borri, and Leonardo Candela, editors, *Digital Libraries: Research and Development*, volume 4877 of *Lecture Notes in Computer Science*, pages 36–45. Springer Berlin Heidelberg, 2007.

Maristella Agosti, Nicola Ferro, Andreina Rigon, Gianmaria Silvello, Erilde Terenzoni, and Cristina Tommasi. SIAR: A User-Centric Digital Archive System. In Maristella Agosti, Floriana Esposito, Carlo Meghini, and Nicola Orio, editors, *Digital Libraries and Archives*, volume 249 of *Communications in Computer and Information Science*, pages 87–99. Springer Berlin Heidelberg, 2011.

William Y. Arms. *Digital Libraries*. MIT Press, 2000.

Ilaria Barbanti. *Appunti su EAD*. Archivio di Stato di Bologna, Scuola di Archivistica Paleografia e Diplomatica, 2006.

Giorgetta Bonfiglio-Dosio. *Primi passi nel mondo degli archivi*. CLEUP, 2003.

Christine L. Borgman. Social Aspects of Digital Libraries. *UCLA, Los Angeles, NSF Workshop Report, Feb*, 1996.

Christine L. Borgman. What are Digital Libraries? Competing Visions. *Information Precessing and Management 35*, pages 227–243, 1999.

Adolf Brenneke. *Archivistica. Contributo alla teoria ed alla storia archivistica europea*. Giuffre', Milano, 1968.

Leonardo Candela, Donatella Castelli, Nicola Ferro, Yannis Ioannidis, Georgia Koutrika, Carlo Meghini, Pasquale Pagano, Seamus Ross, Dagobert Soergel, Maristella Agosti, Milena Dobreva, Vivi Katifori, and Heiko Schuldt. *The Digital Library Reference Model*. Funded under the Seventh Framework Programme, ICT Programme, 2009.

Eugenio Casanova. *Archivistica*. Siena Stabilimento Arti Grafiche Lazzeri, 1929.

Giorgio Cencetti. *Archivi*. Vatican. Scuola pontificia di paleografia, diplomatica e archivistica, 1937.

CRIBC. *SIUSA, Sistema Informativo Unificato per le Soprintendenze Archivistiche Genesi e sviluppi di un progetto*. Centro di Ricerche Informatiche per i Beni Cultural, Scuola Normale Superiore di Pisa, 2001.

Nicola Ferro and Gianmaria Silvello. The NESTOR Framework: How to Handle Hierarchical Data Structures. In Maristella Agosti, JosÃ© Luis Borbinha, Sarantos Kapidakis, Christos Papatheodorou, and Giannis Tsakonas, editors, *Proceedings of the 13th European conference on Research and advanced technology for digital libraries*, ECDL'09, pages 215–226. Springer, Berlin, Heidelberg, 2009.

Nicola Ferro and Gianmaria Silvello. FAST and NESTOR: How to Exploit Annotation Hierarchies. In Maristella Agosti, Floriana Esposito, and Costantino Thanos, editors, *Digital Libraries - 6th Italian Research Conference, IRCDL 2010, Padua, Italy, January 28-29, 2010. Revised Selected Papers*, volume 91 of *Communications in Computer and Information Science*, pages 55–66. Springer Berlin Heidelberg, 2010.

Edward A. Fox. 5S and the Reference Model. *DELOS Reference Model Workshop Frascati-Rome, June 1-2*, 2006.

Vincenzo Freda. *Elementi di archivistica*. Dispensa, 2008.

Marcos Andre Goncalves. Streams, Structures, Spaces, Scenarios, and Societies (5S):A Formal Digital Library Framework and Its Applications. *Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University*, 2004.

ICA. *ISAD (G): General International Standard Archival Description*. International Council on Archives, 1999.

Yannis Ioannidis, Diego Milano, Hans-Jorg Schek, and Heiko Schuldt. DelosDLMS: From the DELOS vision to the implementation of a future digital library management system. *Int. J. Digit. Libr.*, 9(2):101–114, November 2008.

Carl Lagoze and Sandy Payette. *Flexible and Extensible Digital Object and Repository Architecture (FEDORA)*. Springer Berlin Heidelberg, 1998.

Charles Oppenheim and Daniel Smithson. What is the Hybrid Library? *Journal of Information Science*, 25:97–112, 1999.

Marco Piraccini and Stefano Rossini. Service Oriented Architecture: dalla teoria alla pratica. *Mokabyte*, 2005. URL `http://www2.mokabyte.it/cms/article.run?articleId=QLA-M6K-VAO-VRD_7f000001_13985019_673b54fd`.

Daniel V. Pitti. Encoded Archival Description: An Introduction and Overview. *D-Lib Magazine*, 5(11), 1999.

Unni Ravindranathan, Rao Shen, Marcos Andre Goncalves, Weiguo Fan, Edward A. Fox, and James W. Flanagan. Etana-dl: a digital library for integrated handling of heterogeneous archaeological data. In Hsinchun Chen, Howard D. Wactlar, Ching chih Chen, Ee-Peng Lim, and Michael G. Christel, editors, *JCDL*, pages 76–77. ACM, 2004. URL `http://dblp.uni-trier.de/db/conf/jcdl/jcdl2004.html#RavindranathanSGFFF04`.

Jr Rich Sezov. *Liferay in Action. The Official Guide to Liferay Portal Development*. In Action Series. Manning Publications Company, 2012.

Chris Rusbridge. Towards the Hybrid Library. *D-Lib Magazine*, July 1998.

Ashish Sarin. *Portelts in Action*. Manning, 2012.

Gianmaria Silvello. *A Set-Based Approach to Deal with Hierarchical Structures* . Tesi di Dottorato - Universita' degli Studi di Padova, 2011.

MacKenzie Smith, Mary R. Barton, Margret Branschofsky, Greg McClellan, Julie Harford Walker, Michael J. Bass, David Stuve, and Robert Tansley. DSpace: An Open Source Dynamic Digital Repository. *D-Lib Magazine*, 9(1), 2003.

Federico Valacchi. *ARCHIVISTICA GENERALE I*. Universita' degli studi di Macerata, 2009.

Stefano Vitali. Archival information systems in italy and the national archival portal. In Maristella Agosti, Floriana Esposito, and Costantino Thanos, editors, *IRCDL*, volume 91 of *Communications in Computer and Information Science*, pages 5–11. Springer, 2010. ISBN 978-3-642-15849-0.

Peter Walne. *Dictionary of Archival Terminology Dictionnaire de Terminologie Archivistique English and French. With Equivalents in Dutch, German, Italian, Russian and Spanish*. ICA Handbooks Series, 1988.