UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Facoltà di Scienze
Matematiche, Fisiche e Naturali

Corso di Laurea in Informatica

Una Web Application su infrastruttura GRID per il progetto DAME

Tesi di Laurea Sperimentale

Tutor Accademico

Dott.ssa Anna Corazza

Candidato

Francesco Manna
matr. 566/1898

Tutor Aziendale

Dott. Massimo Brescia

Anno Accademico 2008/2009

Alla mia Famiglia
e a tutti coloro che hanno
sempre creduto in me.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

La presente tesi illustra il lavoro di progettazione e realizzazione del componente Front End all'interno del progetto DAME. Il modulo software è stato da me sviluppato durante il tirocinio esterno presso l' Osservatorio Astronomico di Capodimonte. Tale modulo consiste in una Rich Internet Application (RIA) che permette di effettuare data mining di carattere astronomico e non solo, sfruttando la potenza di calcolo del GRID computing (S.Co.P.E).
Il software sviluppato permette di gestire le funzionalità disponibili in modo intuitivo, consentendo così all'utente, senza specifiche conoscenze informatiche pregresse, di eseguire esperimenti di carettere scientifico sfruttando solo una connessione ad internet e un browser. L'interattività e la facilità d'uso dell'applicazione derivano dalle tecnologie utilizzate, ad esempio Ajax, mediante i framework Google Web Toolkit, e SmartGwt, che la rendono simile ad una desktop application. La tesi è organizzata come segue.

Nel primo capitolo viene fatta un'introduzione all'intero progetto, in particolare vengono descritti i singoli componenti.

Nel secondo capitolo è fornita una panoramica sull'ambiente di sviluppo dell'applicazione, dalle tecnologie utilizzate (prevalentemente tecnologie web) ai framework. Viene dato particolare risalto a ciò che è stato scelto per la realizzazione e dunque appreso durante il periodo di tirocinio.

Nel terzo capitolo viene trattata la parte di studio dei requisiti, progettazione e implementazione del componente Front End.

Infine il quarto capitolo contiene una serie di considerazioni personali sul lavoro svolto durante questi mesi e sui possibili sviluppi futuri del componente Front End.

Capitolo 1

Il progetto DAME

Negli ultimi decenni, il data mining è divenuta una delle attività fondamentali per la comprensione, la navigazione e lo sfruttamento dei dati nella nuova era digitale. Per data mining si intende un processo automatico che permette di estrarre informazioni significative da grandi moli di dati tramite l'applicazione di particolari tecniche e algoritmi. Grazie al continuo progredire delle tecnologie, avere dati a disposizione non è più un grande problema, si sa esattamente dove reperirli o dove depositarli, ma il problema che si deve affrontare, man mano che essi aumentano è di come cercare di utilizzarli per estrarne informazioni. Spesso i dati, nella maggior parte dei casi si presentano in forma eterogenea, ridondante e non strutturata e senza strumenti di data mining sarebbe impossibile estrarne informazioni. Da quanto detto sono evidenti i numerosi ambiti applicativi del data mining, da comuni operazioni di marketing ad ambiti di ricerca scientifica. Il progetto DAME[1] si propone di fare data mining ovviamente in ambito di ricerca scientifica, in particolare di tipo astronomico. Il dominio astronomico è caratterizzato da insiemi estremamente grandi di dati, quindi il tempo di elaborazione per estrarre informazioni da questi insiemi puó avere costi computazionali molto alti, e come ben sappiamo pone grossi limiti. Nel seguente paragrafo vengono esaminate le principali caratteristiche del progetto.

1.1 Caratteristiche del progetto DAME

Il progetto DAME è un'evoluzione di due vecchi progetti Astroneural e VO-Neural e consiste nella progettazione e nello sviluppo di una suite di data mining orientata al web per la comunità astronomica conforme ai requisiti

[1] **DA**ta **M**ining & **E**xploration

e agli standard IVOA[2]. Come già accennato precedentemente la grandezza degli insiemi di dati pone dei limiti computazionali che all'interno del progetto vengono abbattuti mediante l'uso di una potente infrastruttura di supercomputing general purpose, basata sul paradigma GRID, realizzata nel progetto S.C.o.P.E[3]. DAME è finanziato principalmente dalla Comunità Europea e dal MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca), e comprende vari partnership:

- Dipartimento di Scienze Fisiche (sezione di Astrofisica) - Università degli Studi di Napoli Federico II
- INAF - Osservatorio Astronomico di Capodimonte
- California Institute of Technology, Pasadena - USA

e collaborazioni:

- VOTECH (Virtual Observatory Technological Infrastructures)
- S.C.o.P.E
- INAF - Osservatorio Astronomico di Trieste (VO-AIDA)
- Sezione Informatica Dipartimento Scienze Fisiche - Università degli Studi di Napoli Federico II
- Dipartimento di Ingegneria Informatica Università degli Studi di Napoli Federico II
- MIUR (Italian Ministry of Research)
- EURO-VO (The European Virtual Observatory)
- IVOA (International Virtual Observatory Alliance)

1.2 La suite

Come è possibile notare nella figura di seguito, la suite é divisa in cinque componenti:

- **Front End** ha il compito di interagire con l'utente e con il componente Framework.

[2]International Virtual Observatory Alliance `http://www.ivoa.net/`

[3]Sistema Cooperativo per Esperimenti Scientifici ad alte Prestazioni `http://www.scope.unina.it`

- **Framework** è il cuore della suite, comunica con tutti i componenti.
- **Registry e Database** rappresenta il deposito di tutte le informazioni della suite.
- **Driver** ha il compito di eseguire le operazioni richieste dal Framework.
- **Data Mining Models** sono le librerie che implementano gli algoritmi di data mining.

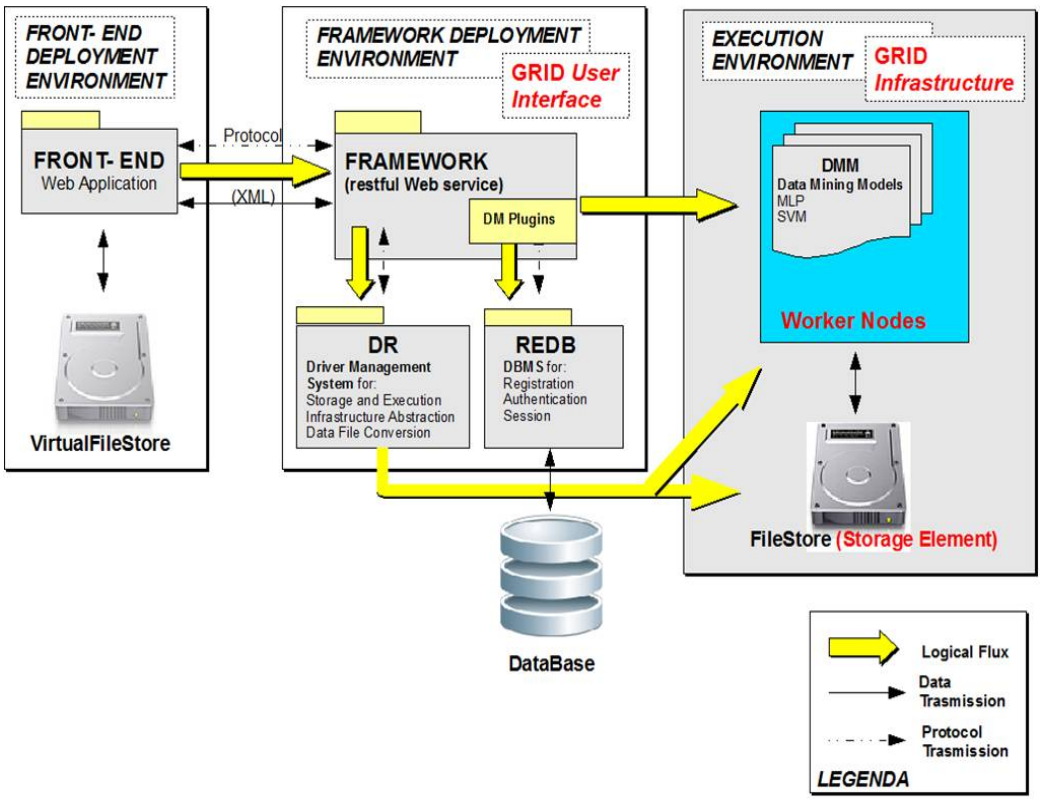

Figura 1.1: Diagramma dei Componenti della suite

La divisione in componenti, è stata più che una scelta una costrizione, vista la complessità del progetto. Ogni componente è stato affidato a uno/due membri del team (per lo più composto da informatici, fisici, ingegneri). Nei seguenti paragrafi verranno esaminati in dettaglio i singoli componenti rendendo sicuramente più esplicativa la Figura 1.1.

1.2.1 IL Framework (FW)

Il componente Framework realizzato da Michelangelo Fiore con supporto di Omar Laurino, è il nucleo centrale dell'intera suite. Il suo compito principale è quello di gestire la comunicazione tra Front End (in particolare l'utente finale) e il resto dei componenti. Grazie all'interazione tra FE e FW, l'utente può registrarsi alla suite, navigare nella sua sessione di lavoro, configurare e eseguire esperimenti e ovviamente visualizzare i risultati.
Dal punto di vista dell'architettura il FW è un *Restful Web Service*. Un Restful Web Service è una particolare architettura client-server, nella quale le risorse offerte sono accessibili ad un URL. Un client che vuole accedere ad una risorsa può usare i comuni metodi HTTP (get,post etc.) all' URL che identifica la risorsa desiderata. Un ambiente Restful è completamente privo di stato, dunque questo fa sì che ogni operazione risulti essere atomica. La scelta di realizzare il FW come un web service restful è scaturita in fase di analisi quando identificando la sequenza di interazioni con il componente FE si è notato che le singole operazioni non erano estremamente complesse e la scelta di rendere il FW restful avrebbe garantito tempi di sviluppo notevoli e interazioni tra i componenti molto più semplici. Compiti fondamentali del FW sono:

- rispondere ad ogni tipo di richiesta da parte del FE, e interagire con tutti gli altri componenti per elaborare una risposta.
- offrire un'interfaccia di amministrazione, che consenta all'amministratore di installare nuove funzionalità ed effettuare operazioni statistiche sul sistema.

Strutturalmente è diviso in tre parti fondamentali Fig.1.2 :

- **Web Service** Interfaccia lato server implementata da Servlet che gestiscono le interazioni con il FE.
- **Data Mining Plugin (DMPlugin)** indicano le funzionalità di data mining utilizzate per effettuare un esperimento. Alcuni DMplugin saranno forniti direttamente poiché sviluppati dal team DAME altri saranno implementati da sviluppatori esterni a cui sará data la possibilitá di integrarli all'interno della suite.
- **Admin Interface** è un interfaccia utilizzata dall'amministratore per aggiungere o rimuovere funzionalità o effettuare statistiche sulla suite.

Gli altri package in Fig.1.2 rispettivamente *XML Generator, Data Types, Security, e Error Management* indicano rispettivamente

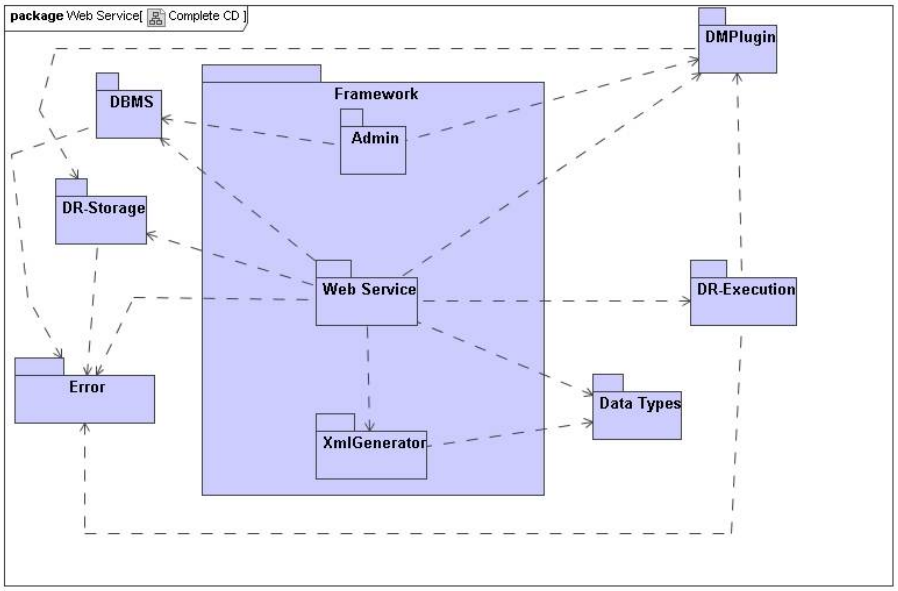

Figura 1.2: Diagramma Architettura componente Framework

- generatore di file XML , mediante i quali avviene la comunicazione con il FE
- tipi di oggetto che rappresentano le entitá fondamentali della suite
- package di classi utilizzate per salvaguardare la sicurezza della suite
- package inerente alla error policy.

L'intero FW è sviluppato in linguaggio Java. Per l'ulteriori informazioni si consulti l'elaborato di tesi di Fiore Michelangelo[4]

1.2.2 Registry e Database (REDB)

Il componente REDB, sviluppato da Alfonso Nocella è il deposito di tutte le informazioni manipolate dalla suite. Il FW non conserva informazioni, ma può solo reperirle, modificarle ed aggiungerle mediante un'interfaccia fornita dal componoente REDB. Le informazioni sono inerenti alla registrazione degli utenti, alle sessioni di lavoro e agli esperimenti associati a quest'ultime. Inoltre vengono conservate informazioni sui dati di input/output e temporanei/finali provenienti dall'esecuzione di esperimenti lanciati dagli utenti.

[4]Il componente Framework del Progetto DAME

Il componente come è ben visibile in Fig.1.3 risulta essere suddiviso in due parti:

- REDB package, si tratta di classi Java , che permettono al FW di effettuare letture e scritture sul Database.
- Database Management System (DBMS), contenitore dei dati, implementato seguendo il modello standard entità-relazione(ER).

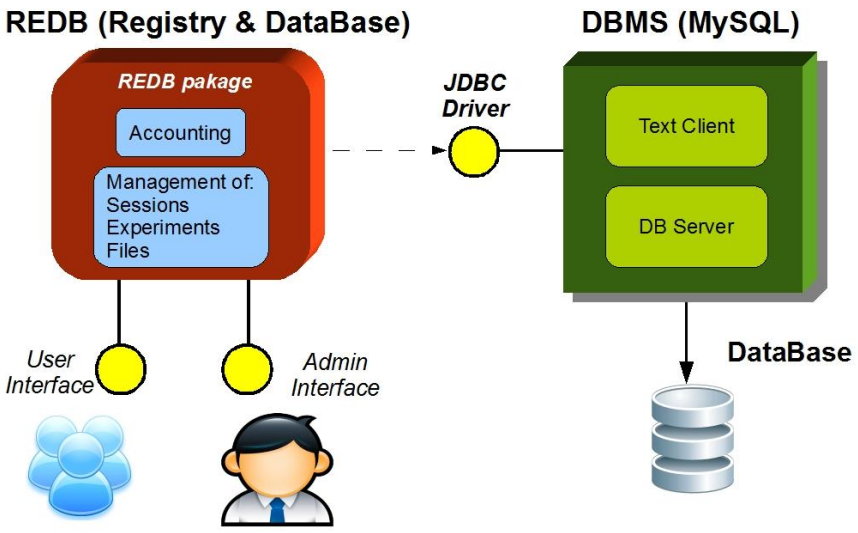

Figura 1.3: Diagramma Architettura componente REDB

Di seguito rispettivamente il diagramma del REDB package Fig.1.4 e il diagramma ER del Database Fig.1.5.

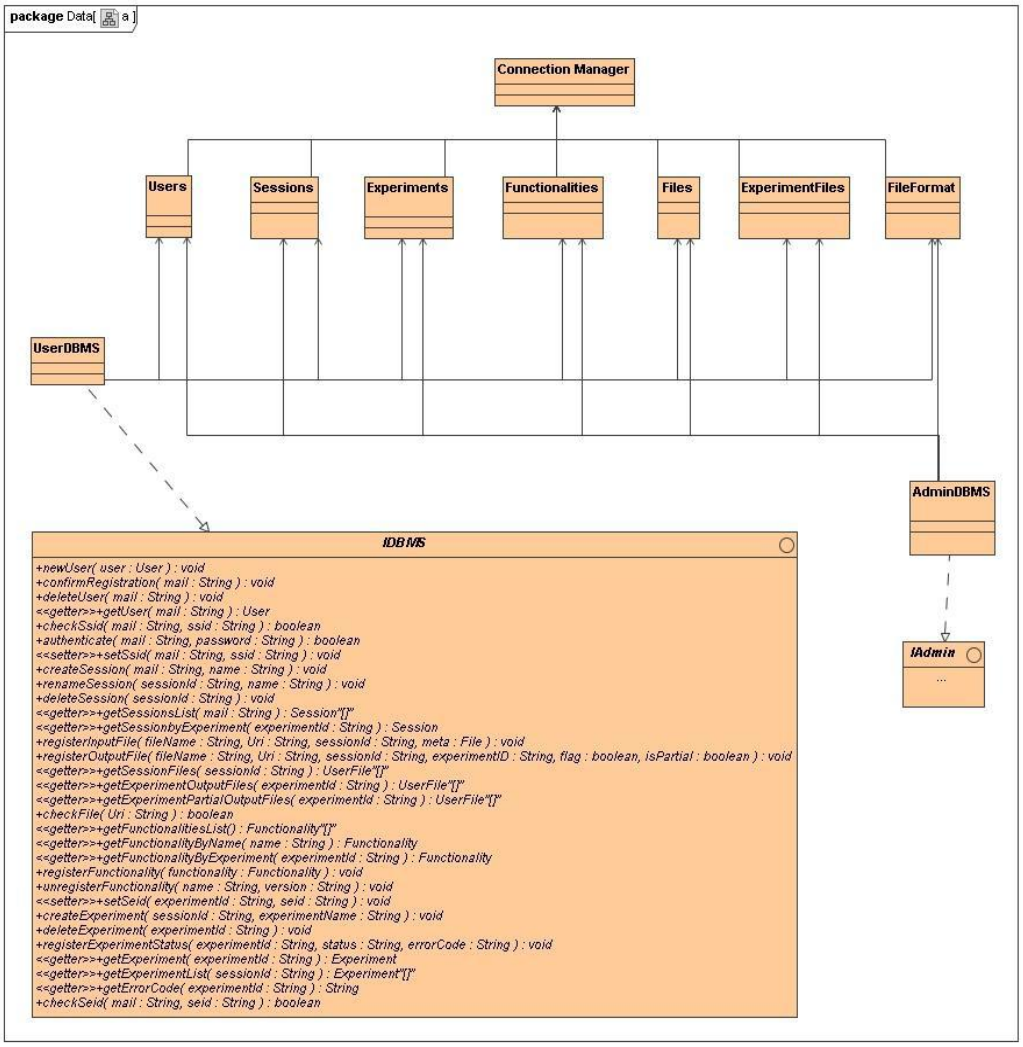

Figura 1.4: Diagramma REDB package

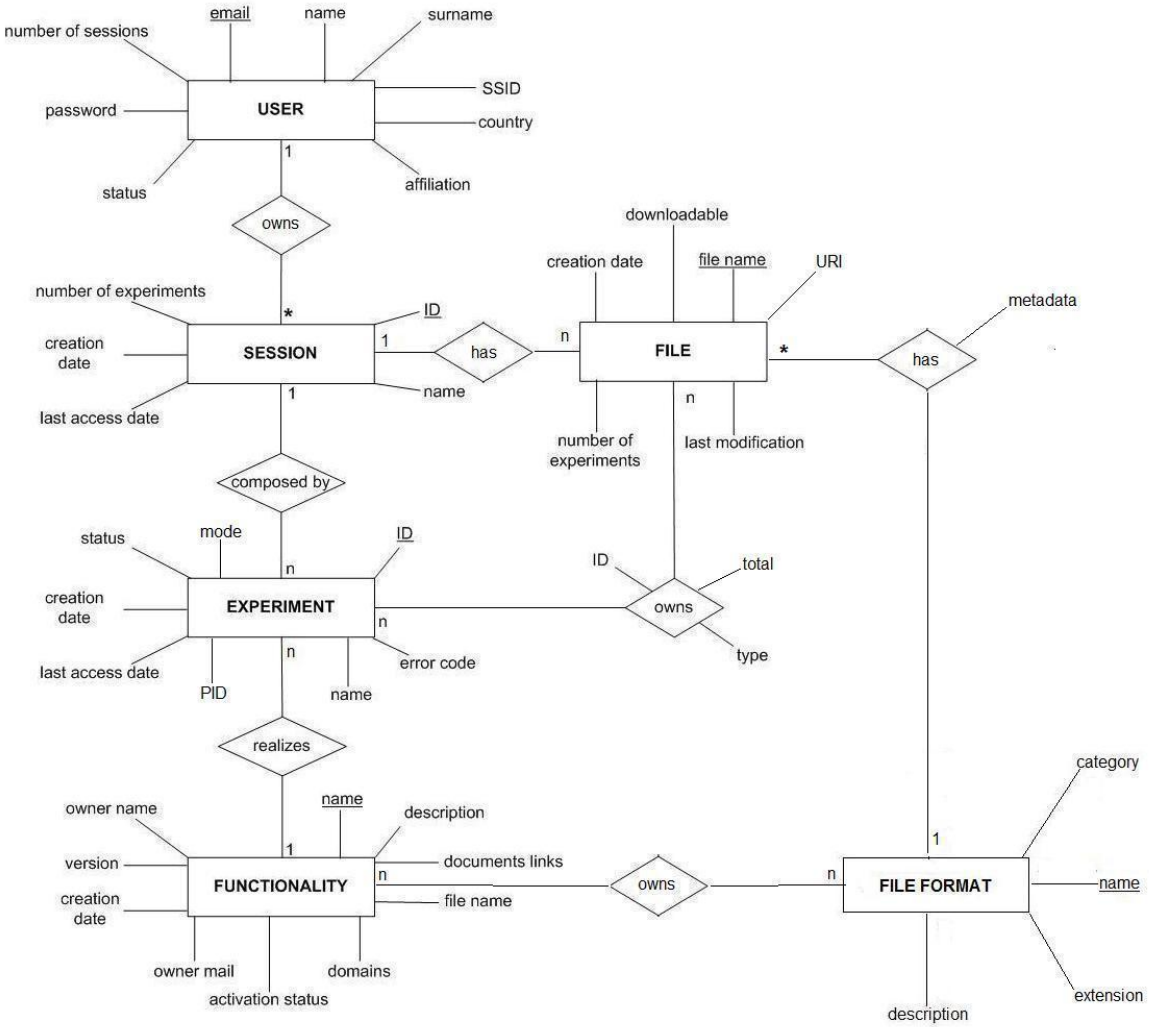

Figura 1.5: Diagramma ER Database

1.2.3 Driver Management System (DRMS)

Il DRMS è il componente usato dal FW per gestire l'ambiente di esecuzione degli esperimenti, storage dei file e processing dei dati. Sviluppato da Giovanni D'Angelo e Mauro Garofalo, il DRMS realizza un'interfaccia di basso livello con l'ambiente di esecuzione del FW quale *Stand Alone (SA), GRID*.

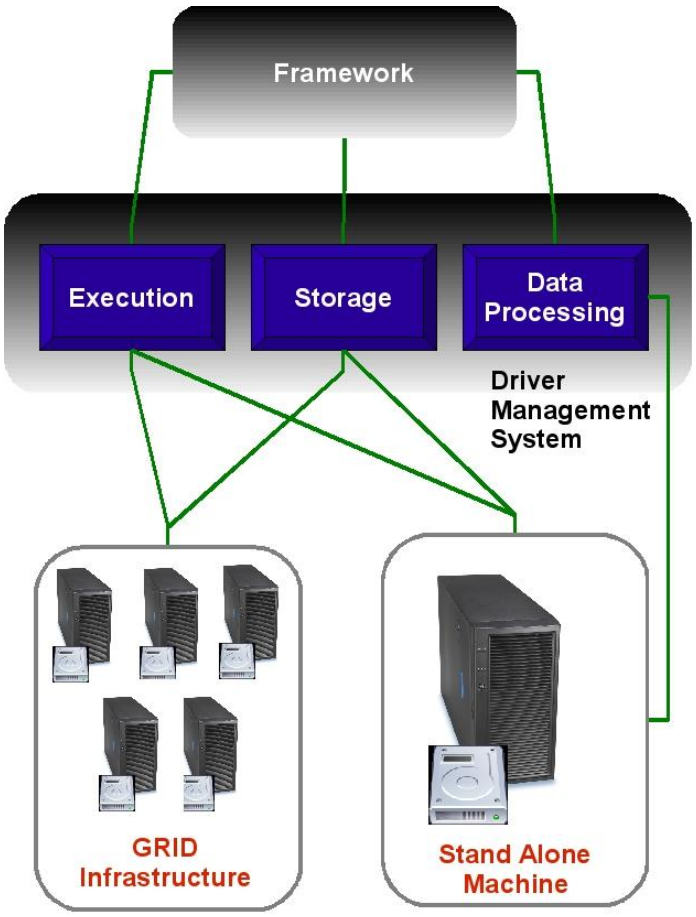

Figura 1.6: Driver Management System

Le operazioni che effettua il DRMS sono le seguenti:

- gestione dei file fisici (download, upload, copia ed eliminazione);
- operazioni di traduzione dei file in un diverso formato. Ogni funzionalità della suite puó avere delle necessità di traduzione del file prima di lanciare un esperimento.

- esecuzione di un esperimento. Ogni esperimento come è ben visibile in Fig.1.6 può essere eseguito Stand Alone, dunque sulla macchina dove si trova il DRMS senza alcuna complicazione oppure su GRID. Nel caso di GRID, il DRMS si deve occupare di spostare l'esecuzione dell'esperimento sull'infrastruttura GRID.

1.2.4 Data Mining Models (DMM)

Il componente DMM, identifica un package che implementa tutti i modelli di data mining previsti dalla suite. Il componente è stato sviluppato da Stefano Cavuoti, Omar Laurino, Michelangelo Fiore e Alessandro Di Guido. I modelli di solito sono scritti in diversi linguaggi di programmazione e strutturati secondo differenti paradigmi (in alcuni casi si riducono a degli eseguibili da riga di comando). Le principali caratteristiche del DMM sono:

- Implementazione orientata alle funzionalità per esempio Classificazione e Regressione.
- Possibilitá di utilizzare funzionalità con più di un modello senza duplicazione di codice.
- Separazione tra modelli supervisionati e non supervisionati.
- Interfaccia comune tra tutti i modelli, mediante una specifica classe rendendo i parametri di data mining auto-adattivi.

Nella prima release, il DMM implementa i modelli Multi Layer Perceptron (MLP) e Support Vector Machine (SVM). Segue il class diagram del componente Fig.1.7

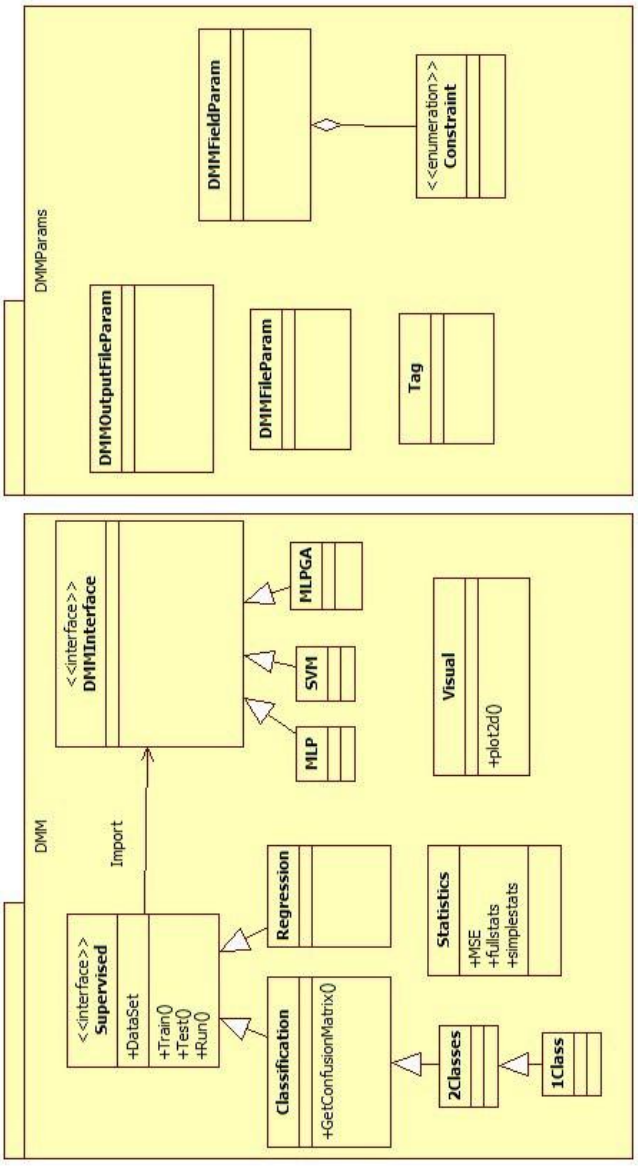

Figura 1.7: Diagramma Architettura Data Mining Models

1.2.5 Front End (FE)

Realizzato da me e principale oggetto di questa tesi il FE è il componente della suite che interagisce direttamente con l'utente e con il FW. I compiti del FE sono:

- fornire un'interfaccia utente semplice e intuitiva mediante la quale l'utente può gestire le sue sessioni di lavoro, creare e lanciare esperimenti, fare upload e download di file.

- comunicare con il FW ricevere le risposte e saper interpretarle in maniera corretta.

Le principali caratteristiche, le tecnologie utilizzate, ulteriori dettagli e lo sviluppo del FE saranno trattati dettagliatamente nei capitoli successivi.

Capitolo 2

Scelte Progettuali e Tecnologie Utilizzate

Nel primo capitolo è stata fatta un'introduzione al progetto DAME. In questo secondo capitolo entreremo in dettaglio sulle tecnologie utilizzate e su parte delle scelte progettuali che interessano il componente oggetto di questa tesi.

Lo sviluppo di internet degli ultimi anni ha alimentato in maniera esponenziale la diffusione delle *web-application* o nel gergo informatico *webapp* dove con tale termine si indica un'applicazione accessibile via web. La diffusione delle web-application è avvenuta soprattutto grazie alla possibilità offerta ai client di accedere alle applicazioni mediante comuni browser. Un'evoluzione delle web-application sono le *Rich Internet Application* (RIA), cioè applicazioni web che possiedono le interattività e le sembianze dal punto di vista grafico delle tradizionali applicazioni per computer senza necessitare di alcuna installazione su disco fisso. Le RIA si caratterizzano per la dimensione interattiva e per la velocità d'esecuzione. Infatti la parte dell'applicazione che elabora i dati è trasferita a livello client e fornisce una pronta risposta all'interfaccia utente, mentre la gran parte dei dati e dell'applicazione rimane sul server remoto, con notevole alleggerimento per il computer utente. Anche l'interazione con una RIA avviene in remoto, tramite un comune browser. In un certo senso le RIA rappresentano una generazione di applicazioni che permette un'interazione totalmente rinnovata, fondata sul meglio delle caratteristiche funzionali e progettuali, che finora erano prerogativa del web o delle applicazioni desktop. Da questa breve introduzione è facile comprendere che il componente FE del progetto DAME si tratta di una RIA a tutti gli effetti. L'intento del progetto come già precisato in precedenza è quel-

lo di abbattere i limiti imposti dall'hardware/software dei client, poichè per effettuare esperimenti di data mining, il tipico utente dovrà essere in possesso solamente di una connessione ad internet (oggigiorno presente anche in aeroporti,piazze ect) e di un browser. La scelta di sviluppare il FE come una RIA è scaturita dal fatto di voler un'applicazione interattiva al punto che l'utente non si accorga che sta eseguendo esperimenti di data mining mediante un browser. Per rendere tutto ciò possibile sono state esaminate le varie tecnologie per lo sviluppo di RIA, tra queste sono state scelte le migliori in termini di tempo di sviluppo, manutenebilità ed estendibilitá del codice.

2.1 Asynchronous JavaScript and XML (AJAX)

Ajax è una delle tecniche più famose per lo sviluppo di applicazioni web interattive. Il concetto alla base di Ajax è in parte espresso nell'acronimo scelto, cioè un utilizzo asincrono di Javascript che attraverso l'interfacciamento con XML, può permettere ad un client di richiamare informazioni lato server in modo veloce e trasparente. Ajax ha via via sempre di più portato a far mutare la natura di quelle che erano delle buone applicazioni di tipo interattivo a delle applicazioni per il web. Tutte le nuove applicazioni sono destinate ad essere online, anche perchè in tal modo vi è possibilitá di sfruttare potenza di calcolo che va al di là di quella di un personal computer di comune uso domestico. Il componente FE, ad esempio, se fosse stato sviluppato come una desktop application, sicuramente non avrebbe permesso di effettuare esperimenti di data mining in tempi ragionevoli, poichè non avrebbe avuto l'opportunità di sfruttare la potenza di calcolo dell'infrastruttura di GRID computing di S.Co.P.E.

Tornando ad Ajax, sostanzialmente è un approccio basato su tecnologie ben note fin dai tempi del web 1.0 come HTML, CSS, DOM e Javascript, con l'aggiunta di linguaggi di tipo server-side come ad esempio le servlet. Da ciò é facile intuire che Ajax quindi non è una tecnologia ma un insieme di più tecnologie in particolare include:

- XHTML e CSS: per la presentazione della pagina in un formato standard
- DOM[1]: per il layout dinamico e l'interazione con la pagina
- XML e XSL[2]: scambio e manipolazione dati.

[1]Document Object Model
[2]eXtensible Stylesheet Language

- XMLHttpRequest: recupero asincrono dei dati.
- Javascript: collante per tutte le tecnologie precedenti, eseguito lato client.

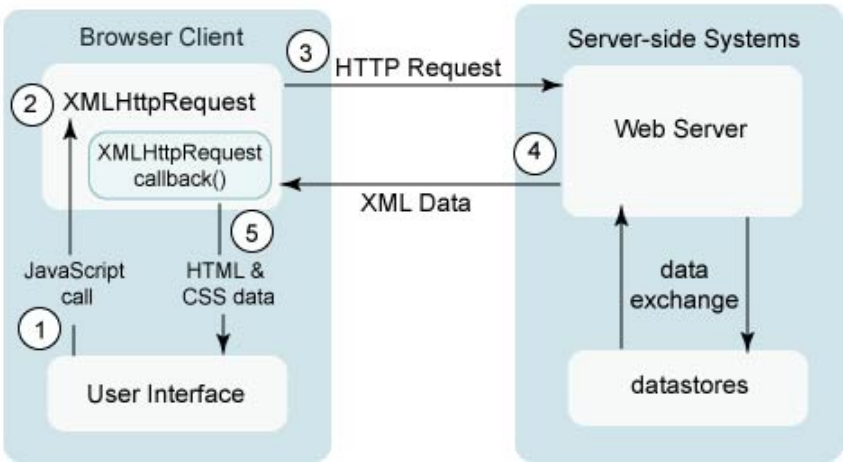

Figura 2.1: Sequenza generale di una richiesta Ajax

La Fig.2.1 illustra i passi fondamentali di una richiesta Ajax:

1. L'utente genera un evento, ad esempio clicca su un bottone, e viene eseguito del codice Javascript.
2. Viene creato e configurato un oggetto XMLHttpRequest, con i rispettivi parametri, i quali includono l'ID del componente che ha generato l'evento e ciascun valore che l'utente ha inserito.
3. L'oggetto XMLHttpRequest effettua una richiesta (HTTP) asincrona al webserver. Un oggetto (ad esempio una servlet) riceve la richiesta, la elabora, e memorizza ciascun dato passato come parametro alla richiesta nel datastore.
4. L'oggetto che ha elaborato la richiesta ritorna un documento XML che contiene il risultato dell'elaborazione (informazioni che devono essere ricevute dal client).
5. Infine, l'oggetto XMLHttpRequest riceve i dati sotto forma di XML, li elabora , e aggiorna la pagina HTML con i nuovi dati.

Le più comuni applicazioni web vengono sviluppate utilizzando le richieste HTTP inoltrate attraverso l'oggetto XmlHTTPRequest al web server. Per le operazioni il server effettua l'elaborazione della richiesta che può essere una semplice raccolta o manipolazione dei dati e restituisce una response al client sotto forma di pagina HTML, tutto ciò senza che l'utente debba aspettare il completamento dell'operazione richiesta grazie all'utilizzo di una comunicazione asincrona. Ajax dunque è una grande innovazione degli ultimi anni rivolta ad estendersi ancora per molto eppure si basa su tecnologie già in uso da tempo. Occorre però sottolineare che Ajax non è una tecnologia innovativa, ma piuttosto un approccio innovativo. Infatti nel web 2.0 si ha una nuova visione della struttura della rete e delle applicazioni su essa costruite che sfrutta la programmazione lato server e introduce una grandissima innovazione: l'elaborazione lato client. Questa è un grande è sostanziale differenza perché si ha la possibilità di sfruttare due piattaforme di elaborazione, una lato client e una lato server, costruendo cosi la propria applicazione ampliando i confini delle potenzialità delle applicazioni web. Dalla Fig.2.2 possiamo notare come il modello tradizionale delle applicazioni web e il modello Ajax siano differenti. Infatti col modello Ajax mentre il server sta elaborando, l'utente continua la sua navigazione.

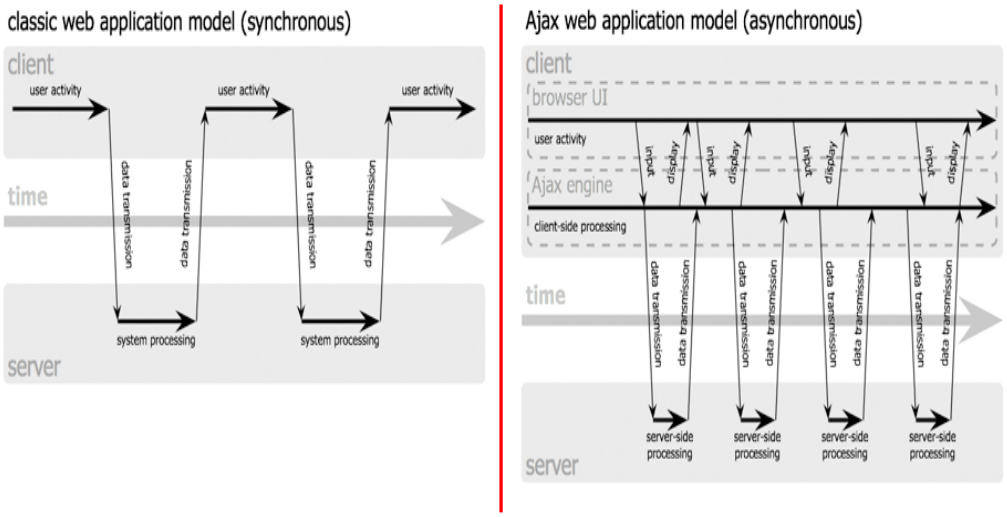

Figura 2.2: Interazione sincrona delle tradizionali web application vs interazione asincrona di un web application con Ajax

Da come è stato descritto fin ora può sembrare che Ajax abbia solo aspetti positivi, ma non è così. Sviluppare applicazioni con Ajax è davvero difficile. Questo risulta vero per varie ragioni:

- richiede un'approfondita conoscenza di Javascript;
- bisogna gestire i diversi tipi di browser;
- i tool di sviluppo sono prematuri, e il debugging in ambienti multipli è problematico.

Tutti questi fattori fanno si che lo sviluppatore se decide di realizzare una grossa applicazione con Ajax, debba avere una vasta conoscenza delle caratteristiche dei differenti browser. Per fronteggiare questi problemi e per far si che lo sviluppatore non debba preoccuparsi di questi aspetti esistono vari toolkit e librerie ad Esempio Dojo, Ext js, Script.aculo.us ect. I vari toolkit elencati sono tutti validi approcci, ma *Google Web Toolkit* (GWT) rappresenta qualcosa di davvero differente, rende lo sviluppo con Ajax davvero semplice e soprattutto molto più veloce.

2.2 Google Web Toolkit

GWT è uno tra i tanti toolkit/framework per scrivere applicazioni Ajax. L'approccio di GWT è quello di permettere di scrivere le applicazioni utilizzando il linguaggio Java e solo alla fine convertire (tramite il compilatore GWT) da classi Java a codice Javascript e HTML browser compliant (Internet Explorer, Firefox, Mozilla ect). Il ciclo di sviluppo di un'applicazione con GWT si può riassumere nel seguente modo:

1. Utilizzo di un IDE Java (Eclipse, Netbeans, IntelliJ, Jprofiler, ecc.) per scrivere l'applicazione scritta in linguaggio Java, con l'ausilio di alcune librerie messe a disposizione da GWT.
2. Test dell'applicazione (Hosted Mode) come codice Java compilato che gira all'interno della JVM.
3. Utilizzo del compilatore GWT per convertire le classi Java in un'applicazione web composta da files JavaScript e HTML.
4. Verifica dell'applicazione (Web Mode) con i browser che si vuole supportare.

2.2.1 Architettura GWT

In Fig.2.3 è rappresentata l'architettura di GWT, principalmente composta da quattro componenti: compilatore Java-to-JavaScript, un web browser - hosted e 2 librerie Java. In dettaglio:

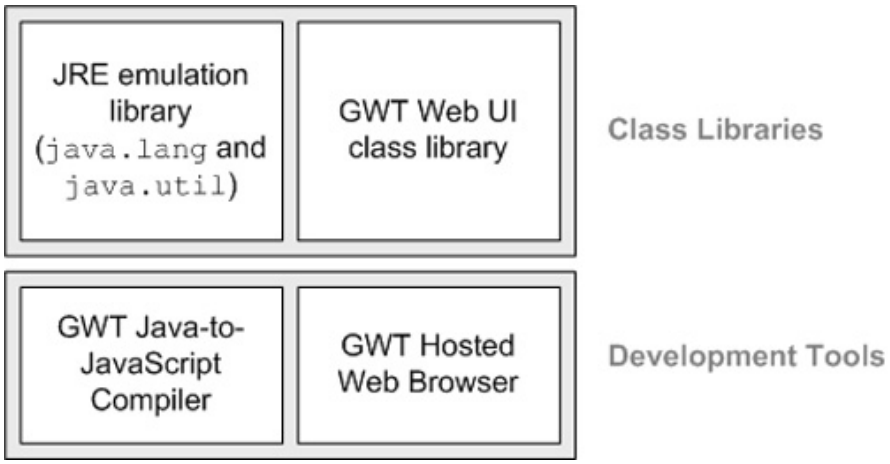

Figura 2.3: Architettura Google Web Toolkit

- Il **compilatore GWT Java-to-JavaScript** converte il codice Java in codice Javascript. Deve essere utilizzato quando si vuole eseguire l'applicazione in modalità web mode. È considerato il fulcro di GWT. L'approccio del compilatore di GWT è quello di prendere e incapsulare le differenze tra i vari browser e compilare codice Javascript da codice Java. La cosa importante da tenere in considerazione è che il compilatore GWT non compila il codice Java nella stessa maniera di javac, ma *traduce* il codice Java in codice Javascript. Ovviamente, dovendo il compilatore fare una conversione da Java a Javascript, non è supportato tutto il linguaggio Java ma solo una parte[3].

- Il **GWT Hosted Web Browser** permette di eseguire le applicazioni in modalità *hosted mode*, ossia di eseguire il codice Java nella JVM senza convertire il codice in linguaggio JavaScript/HTML. Per fare questo, il browser GWT include speciali librerie per il browser in uso.

- **JRE emulation library** contiene le implementazioni in linguaggio Javascript delle librerie Java standard maggiormente utilizzate (package java.lang.* e java.util.*). Tutti gli altri package (ad es. java.io.*, java.net.*, ecc.) non sono supportati nativamente da GWT.

- **GWT Web UI class library** è una libreria User Interface contenente un insieme di interfacce e classi che permettono di disegnare le pagine web (ad es. bottoni, text boxes, immagini, ecc.) . Questa è la libreria standard principale per creare applicazioni web-based basate su GWT.

[3]Per maggiori dettagli `http://code.google.com/intl/it-IT/webtoolkit/`

2.2.2 Perchè GWT

La scelta che spinge qualsiasi utente che vuole sviluppare una RIA ad usare GWT è che scrivendo codice in Java si hanno a disposizione tutti i concetti propri del paradigma orientato agli oggetti per la strutturazione del codice, per la riusabilità e per la sua manutenibilità. Inoltre codice generato dalla compilazione Java-to-Javascript di GWT sarà compatibile con tutti i più comuni browser. Altri vantaggi che hanno portato il team DAME alla scelta di questo strumento sono:

- La continua evoluzione di GWT, i numerosi tool di sviluppo, e il supporto on line (tutorial e altro), e la predominanza di Google nel campo web sono stati i fattori principali.

- Il risultato di un'applicazione sviluppata con GWT è davvero molto vicina ad una desktop application (scopo principale di DAME), infatti GWT offre una molteplicitá di UI component e da la possibilità di svilupparne propri a seconda delle esigenze.

- La comunicazione client-server mediante *Remote Procedure Call* (RPC).

- La rapida curva di apprendimento, ha giocato anch'essa un ruolo importante. Infatti si può quasi dire che per iniziare a sviluppare un'applicazione non bisogna altro che conoscere il linguaggio Java.

2.2.3 Comunicazione Client-Server in GWT

É noto che quando si sviluppa una RIA, cosa fondamentale è la comunicazione tra il browser(client) e il server. Oggi tutti i più noti browser, includono uno speciale oggetto Javascript chiamato *XMLHttpRequest*, che permette la comunicazione tra il browser e il server senza fare alcun refresh della pagina. L'oggetto XMLHttpRequest è alla base delle browser-based Remote Procedure Calls. GWT prevede due particolari tool che fanno da strato superiore all'oggetto XMLHttpRequest:

- **RPC** permette di mandare e ricevere oggetti Java tra il client e il server.

- **Request Builder** è una classe che fa essenzialmente da wrapper a questo oggetto.

Remote Procedure Call

È cosa risaputa che un server offre servizi. Richiedere questi servizi mediante GWT RPC, risulta come chiamare un metodo in locale ma con qualche riga di codice in più.

Il meccanismo GWT RPC fondamentalmente può essere diviso in tre parti:

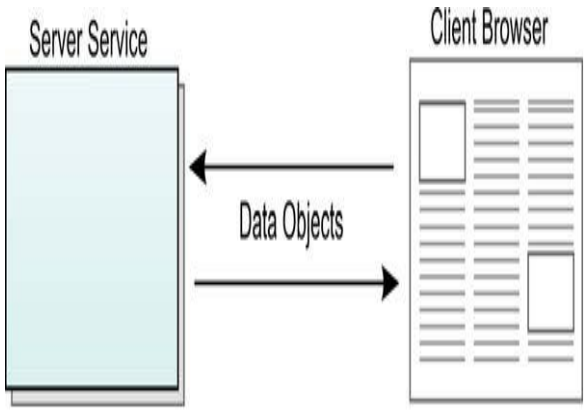

Figura 2.4: GWT RPC : interazione tra client e server mediante data object

- Il servizio, che viene eseguito sul server come una servlet.
- Il browser che funge da client e richiede il servizio.
- Gli oggetti che vengono trasmessi dal client al server e viceversa. Sia il client che il server hanno la possibilitá di serializzare e deserializzare questi oggetti in modo che il client e il server possono scambiarseli.

I tipi di dato di scambio tra server e client devono essere innanzi tutto serializzabili e possono essere sostanzialmente dei seguenti tipi:

- Tipi primitivi Java: *boolean, byte, char, double, float, int, long, short.*
- I wrapper dei tipi primitivi.
- Un sottoinsieme degli oggetti Java Runtime Environemnt (JRE)

- Qualsiasi tipo definito dall'utente a patto che implementino l'interfaccia *Serializable*[4]

Specificate queste nozioni vediamo come si implementa un servizio utilizzando GWT-RPC. Gli step da seguire sono i seguenti Fig.2.5 :

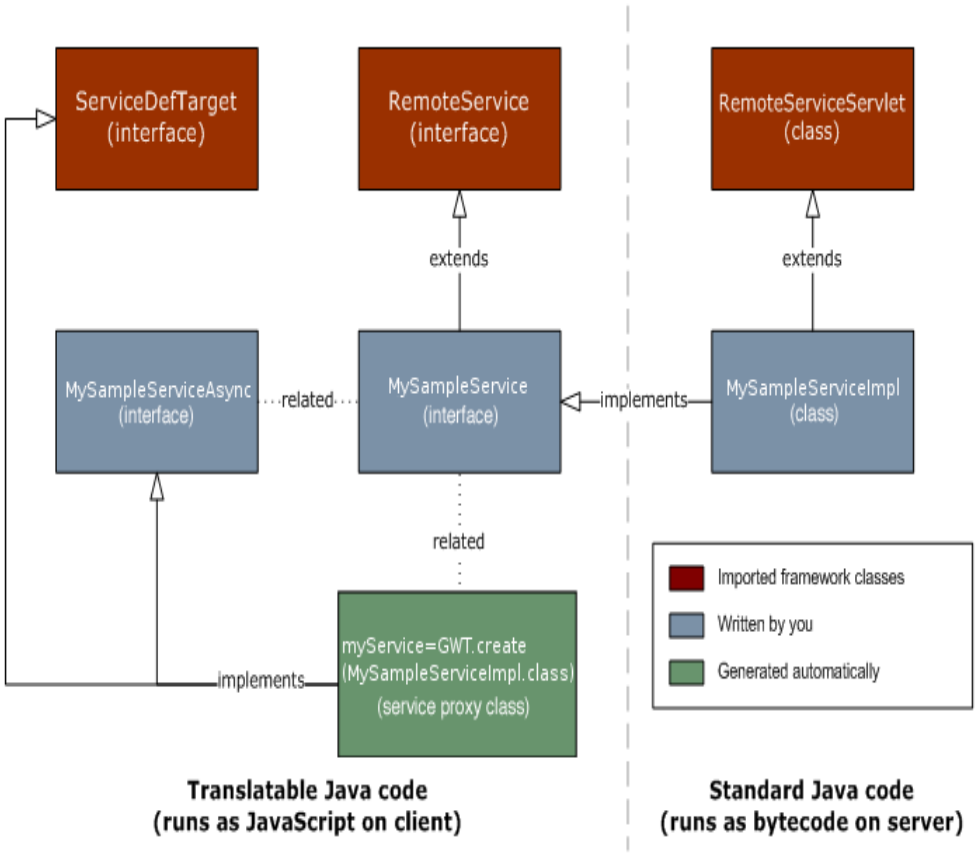

Figura 2.5: Il meccanismo GWT RPC

1. Definire un'interfaccia (*MySampleService*) che implementa *RemoteService* e definisce al suo interno tutti i metodi RPC che si intendono implementare.

[4]La serializzazione è il processo di trasmissione di un oggetto in forma binaria attraverso una connessione di rete. Le chiamate alle GWT-RPC sono tra codice Javascript e Java e GWT prevede la serializzazione come parte del meccanismo RPC. Da notare che la serializzazione GWT è differente dalla serializzazione di Java. Per ulteriori infomrazioni `http://code.google.com/intl/it-IT/webtoolkit/tutorials/1.6/clientserver.html`

2. Creare una classe (*MySampleServiceImpl*) che estende *RemoteService-Servlet* e implementa l'interfaccia creata precedentemente.

3. Definire un'interfaccia (MySampleServiceAsync) che verrà implemententa client-side e permetterà di invocare il/i metodo/i definiti nell'interfaccia che implementa *RemoteService* e ricevere il valore di ritorno (un oggetto).

Infine una cosa importante da notare al di là del discorso tecnico ampiamente discusso in letteratura è che se usate propriamente, le RPC danno l'oppurtunità di spostare tutta la parte di *logic User Interface* (UI) sul client, questo comporta un miglioramento notevole delle performance, una riduzione di occupazione di banda sul webserver, una riduzione di sovraccarico del server, e soprattutto un iterazione più fluida con l'utente. All'interno del FE la comunicazione client-server è stata implementata mediante RPC in modo da sfruttare a pieno questi vantaggi, fatta eccezione per il download e l'upload dei file

Request Builder

La classe *RequestBuilder* permette di creare una richiesta HTTP al server, ed è essenzialmente molto più semplice e lineare rispetto ad una chiamata RPC. Per effettuare una richiesta con RequestBuilder bisogna seguire i seguenti passi:

1. Istanziare un'oggetto della classe RequestBuilder.

2. Inizializzare i parametri della richiesta, come ad esempio l' URL.

3. Inviare la richiesta con il metodo *sendrequest* che ha come parametro un istanza di un oggetto RequestCallback. RequestCallBack ha due metodi:

    - onError, viene invocato in caso di fallimento della richiesta al server
    - onResponseReceived, invocato in caso di successo. Dettagli della risposta del server sono contenuti nell'oggetto Response. Dettagli della risposta del server (status code, HTTP header ect) possono essere ricavati dagli argomenti dell'oggetto Response. Inoltre dato che le richieste HTTP in GWT sono asincrone, comporta che tutto il codice seguente alla chiamata del metodo sendrequest viene subito eseguito, dunque mediante l'oggetto Response bisogna monitorare lo stato della richiesta, e se il caso cancellarla.

2.3 SmartGwt

Fin'ora all'interno del capitolo sono stati affrontati i concetti di comunicazione client-server tralasciando totalmente la parte *Grafical User Interface* (GUI). GWT di suo offre un'ampia gamma di Widget[5] e oltretutto dà la possibilità di estenderli e crearne altri nuovi secondo le proprie esigenze. La possibilità di estendere i widget da parte di Google ha dato il via alla nascita di numerose librerie che offrono widget dall'aspetto molto più accattivante. I piú noti sono:

- **GWT-Ext** (`http://www.gwt-ext.com`): consiste in una libreria GWT Java basata sulla libreria Javascript ExtJS JavaScript(`http://www.extjs.com`). Ext js è davvero ricca di funzionalità e molto performante, oltre ovviamente allo stile elegante. Il modo in cui GWT-EXT utilizza ExtJs è quello di piazzare le librerie ExtJs all'interno della pagina web e poi trasformare il codice Java in Javascript il quale usa a sua volta le librerie ExtJs. Lo stato di Gwt-Ext ha raggiunto livelli avanzati in brevissimo tempo coprendo la maggior parte delle caratteristiche di ExtJs. Purtroppo il gran numero di problemi sollevati da parte degli utenti che utilizzavano GWT-EXT hanno fatto si che il progetto fosse abbandonato e soppiantato da un'altra libreria sviluppata dagli stessi ideatori di GWT-EXT: SmartGwt.

- **SmartGwt** (`http://www.smartclient.com/smartgwt/`): è basato sulla libreria Javascript SmartClient (`http://www.smartclient.com`). Questa libreria funziona in maniera simile a GWT-EXT. Il codice Javascript auto-generato dal compilatore GWT, fa delle chiamate alla libreria SmartClient e il tutto viene visualizzato nel browser. Come accennato in precedenza SmartGwt ha soppiantato GWT-EXT ed è presente una release stabile della libreria. Inoltre SmartGwt risulta tra tutti quello più supportato e in continua evoluzione.

- **GXT** lavora alla stessa maniera di GWT-EXT, la sostanziale differenza è che non neccessita dell'inclusione della libreria ExtJs. GXT è sviluppato secondo l'idea di GWT, infatti genera codice Javascript ottimizzato e include nella pagina web solo il codice Javascript utilizzato. La differenza la si può notare se compilassimo due identiche applicazioni, una in GWT-EXT e una in GXT, il risultato sarebbe che l'applicazione in GXT risulta molto più leggera e veloce. L'unica pecca di GXT, è che necessita di una licenza di uso commerciale.

[5] I Widget sono componenti di un'applicazione GWT visibili all'utente all'interno del browser. La composizione di più widget generano la GUI dell'applicazione.

Ritornando al componente FE, dunque la scelta su come implementare la GUI è stata di utilizzare SmartGwt. La vasta scelta di widget, ha dato la possibilità di creare una GUI davvero intuitiva, adatta a tutti i tipi di utenti. Inoltre l'utilizzo di questa libreria ha permesso di curare ogni minimo dettaglio grafico. Infatti l'utente può effettuare ad esempio il drag and drop, il minimize di alcune finestre; in tal modo può metterle in secondo piano e continuare ad utilizzare l'applicazione proprio come farebbe con un comune software installato su disco. Infine un'altro aspetto importante di SmartGwt è la validazione dei dati, ovviamente questa caratteristica è già presente in GWT, ma nella libreria basata su SmartClient è arricchita con vari tipi di tooltip e animazioni che in caso di inserimento di dati non corretti ad esempio all'interno di un form guidano l'utente alla corretta compilazione.[6]
Ulteriori librerie di minore rilevanza e scelte progettuali più specifiche saranno trattate nel prossimo capitolo.

[6]Per maggiori dettagli `http://www.smartclient.com/smartgwt/`

Capitolo 3

Sviluppo del Front-End

3.1 Descrizione dettagliata e Requisiti

Il FE, è il componente della suite che interagisce con l'utente e con il FW. L'utilizzo del FE è rivolto principalmente ad una figura professionale di carattere scientifico, anche senza approfondite conoscenze informatiche. Lo scopo è quello di creare una GUI user-friendly, che l'utente già al primo uso può utilizzare senza alcuna difficoltà.

Figura 3.1: Interazione tra FE e FW

Il FE in linea di principio deve permettere all'utente di:

- Effettuare una registrazione (in modo da ottenere delle credenziali per poter accedere al sistema).
- Effettuare l'accesso al sistema qualora l'utente sia già registrato.
- Gestire workspace[1]

[1]Un workspace può essere visto come un contenitore di esperimenti appartenenti allo stesso caso scientifico.

- Trasferire i dati di input (dati caricati dall'utente) al componente FW.

- Elaborare i dati di input.

- Mostrare all'utente il risultato delle elaborazioni.

Un utente può usufrire del FE, come già specificato nei capitoli precedenti mediante browser. A questo punto se si è in possesso delle credenziali di accesso (username e password) si puo' effettuare l'accesso al sistema; in caso contrario è necessario che l'utente effettui una registrazione. Le informazioni che l'utente deve inserire per registrarsi sono le seguenti:

- Nome
- Cognome
- E-mail
- Paese
- Istituto Affiliato (Es. Inaf, Caltech, Federico II)
- Password

Dopo aver inserito queste informazioni l'utente può inviare la richiesta di registrazione. L'account da questo momento in poi deve essere attivato per poter garantire l'accesso all'utente. L'utente non appena avrà effettuato la registrazione riceverà un e-mail di conferma con un link di attivazione e solo al momento della validazione del link la sua username e la sua password garantiranno l'accesso alla suite[2]. Una volta attivato l'account, l'utente può accedere al sistema. In prima istanza il componente FE effettua una richiesta al FW, il quale risponde mediante un file XML (VOTABLE compliant) contenente le informazioni inerenti ai workspace, esperimenti e i file di input e output di ciascun esperimento. Se l'utente non possiede workspace, per iniziare ad usufruire delle funzionalità della suite deve provvedere a crearne uno ed effettuare l'upload di uno o più file di input. Una volta effettuati questi semplici passi l'utente può iniziare ad effettuare degli esperimenti. Inoltre il FE deve poter:

- controllare lo stato di upload di un file;

[2]Nella prima release della suite, l'attivazione degli account non sarà gestita con la validazione del link, ma attraverso una procedura manuale effettuata dall'amministratore della suite. Infatti, ogni qual volta un utente si registrerà l'amministratore ricever'a una notifica via e-mail e provvedera mediante un apposita interfaccia web sul componente REDB a rendere attiva la registrazione.

- fornire un sistema di validazione lato client e lato server per garantire la correttezza dei dati, sia inseriti dall'utente quando vuole effettuare un esperimento, sia quelli che il FE invia al componente FW;

- offrire una solida comunicazione con il componente FW, al quale effettua ogni tipo di richiesta;

- controllare lo stato dei file di output, in modo da permettere all'utente qualora siano stati generati dal FW, di poterli scaricare e visualizzare (questo vale anche per i file di output parziale[3], qualora l'esperimento sia stato sottomesso in modalita interattiva).

Specifiche di accesso utente

Il FE offre all'utente l'accesso ai seguenti elementi:

- **Virtual File Store (VFS)**,collegamento logico al reale File Store[4]. Il VFS, dunque permette all'utente di interagire (download, edit, etc) con i file inerenti agli esperimenti o nel caso più generale ai workspace.

- **Session Manager**, con questo termine vengono indicate tutte le operazioni che l'utente può effetture sui workspace.
  Un utente può:

  - Creare un nuovo workspace.
  - Utilizzare un workspace già esistente. Ad esempio per creare, rinominare, eliminare esperimenti al suo interno, oppure per effettuare l'upload di nuovi dataset.
  - Cancellare un workspace.
  - Rinominare un workspace.

  Ogni workspace è caratterizzato da:

  – **Nome**, scelto dall'utente al momento della creazione.
  – **SessionID**, indica un identificativo univoco.
  – **Data di Creazione**
  – **Data di ultimo accesso**

[3] I file parziali, indicano quei file che sono pronti per il download anche se l'esperimento non è terminato del tutto, ad esempio un eventuale file che descrive l'andamento dell'errore

[4] All'interno del File Store sono conservati fisicamente tutti i file. Il FW mediante il componente DRMS effettua al suo interno operazioni di lettura/scrittura.

  - **VFS - Lista dei file**: sono tutti i file di input e output inerenti agli esperimenti del workspace.

  Per quanto riguarda l'uso, la creazione e il rename di un workspace il flusso delle operazioni risulta essere lineare. Occorre viceversa soffermarsi sul comportamento del FE nel caso di cancellazione. L'utente può cancellare:

  1. Un workspace privo di esperimenti.
  2. Un workspace con degli esperimenti a patto che il loro status non sia di pending.

  Nel caso in cui l'utente voglia cancellare un workspace con degli esperimenti in pending, lo può fare utilizzando la funzionalità di cancellazione forzata. Si è ritenuto opportuno inserire questa funzionalità per salvaguardare i dati di un workspace concettualmente ancora valido.

- **Functionality Selector** Quando un utente crea un nuovo workspace, e effettua l'upload di uno o più file, può scegliere una funzionalità (Es. SVM, MLP, MLPGA). Ogni funzionalità ha le seguenti caratteristiche:

  - Nome
  - Descrizione
  - Dominio
  - Data Creazione
  - Versione
  - Link alla documentazione

  Effettuata la scelta di una delle funzionalità l'utente è pronto ad inserire i dati di input per sottomettere un nuovo esperimento.

Specifiche di Input/Output

Ogni tipo di dato, sia esso di input o di output, non risiede fisicamente sulla macchina che ospita il FE ma su quella che ospita il componente FW[5]. Il FE come detto in precedenza permette di ottenere mediante interazioni con il VFS qualsiasi tipo di file, o informazioni su di essi, effettuando richieste HTTP al FW. Identico discorso per quanto riguarda l'output (Paragrafo 3.1).

[5]In particolare fisicamente sono conservati sulla macchina che ospita il FW e dal punto di vista logico nel componente REDB Paragrafo1.2.2

Specifiche Dataset Editor

Il FE dà inoltre la possibilità di costruire interattivamente nuovi dataset mediante manipolazione di meta-dati inerenti a file di input o output gia presenti nel VFS dell'utente. Questa funzionalità permette la creazione di workflow. Il Dataset editor ad esempio viene utilizzato quando in input si ha un dataset grezzo di grandi dimensioni. L'utente che vuole costruire interattivamente il suo dataset, seleziona il numero di colonne di interesse, e inserisce una percentuale che indica la lunghezza in righe del file. Dopo questi semplici passi, l'utente può sottomettere un esperimento con file di input il/i nuovo/i dataset estratto/i. C'e inoltre da sottolineare che le operazioni sul dataset sono elaborazioni che vengono delegate al FW, il quale provvederà ad assegnarle al DRMS 1.2.3, e al FE verrà restituito l'URI del nuovo file.

3.2 Architettura

Il FE, utilizza un'architettura Three-Tier, che è un' estensione del modello client-server. Essa prevede la suddivisione del sistema in tre diversi moduli dedicati rispettivamente all'interfaccia utente, alla logica funzionale (business logic) e alla gestione dei dati persistenti. Tali moduli interagiscono fra loro secondo le linee generali del paradigma client-server (l'interfaccia è client della business logic, quest ultima è client del modulo di gestione dei dati persistenti) e utilizzando interfacce ben definite. In questo modo, ciascuno dei tre moduli può essere modificato o sostituito indipendentemente dagli altri. In particolare la gestione dei dati persistenti nella classica architettura three-tier, è composto da database, nel nostro caso è il componente FW a prendere il suo posto (vd. Fig.3.2).

Figura 3.2: Architettura FE

3.2.1 Server-side

La parte server del FE, è quella che si occupa dell'elaborazione e di restituire i dati al client. Quando riceve una richiesta da parte del client, il server esegue i seguenti passi:

1. controlla la correttezza dei dati ricevuti.
2. Effettua una richiesta HTTP al FW.
3. Riceve una risposta dal FW in un documento XML.
4. Effettua il parsing del file XML ricevuto e il wrapping del contenuto dell' XML in oggetti Java.
5. Restuituisce l'oggetto al client.

In particolare la parte server del FE è composta da:

- due librerie, una di low level e una di high level (Fig.3.4).
- l'implementazione concreta dei servizi.(La figura di seguito ricapitola il funzionamento di una RPC Fig.3.3 già discussa nel Capitolo 2.2.3)
- Il generatore e parser dei file XML (Fig.3.5).

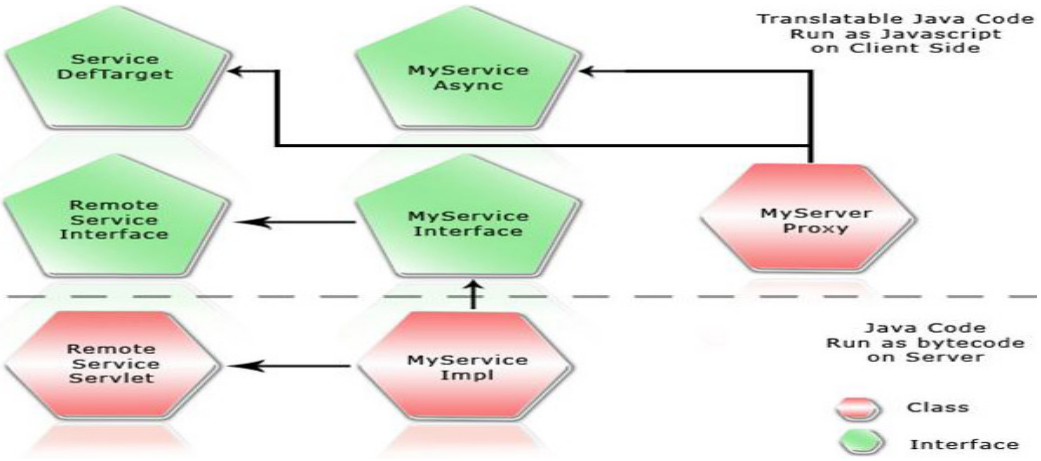

Figura 3.3: Struttura di una RPC con scambio di oggetti tra client e server

High Level e Low Level Library

High Level e Low level Library sono due librerie scritte in codice Java. Lo scopo di queste due librerie è quello di dare la possibilità a chiunque in futuro di realizzare la propria implementazione del FE. Lo sviluppatore non dovrà preoccuparsi dell'interazione con il FW e di effettuare le richieste HTTP, ma dovrà occuparsi solamente di gestire i documenti XML di risposta. Un'altro vantaggio che ha portato il Team Dame alla scelta di questo approccio è la possibilità in caso di cambiamenti all'interno del FW, di qualche interfaccia o simile, di non apportare alcun cambiamento al codice, ma solamente di importare una nuova versione delle librerie.

- **High Level Library**: contiene un unica classe FrameworkUseCase. Ogni metodo della classe restituisce una stringa con l'url di accesso ad un singolo caso d'uso implementato dal FW (Fig.3.4).
- **Low Level Library**: contiene la classe FrontEndUseCase, la quale in ciascun metodo utilizza un oggetto di FrameworkUseCase per recuperare l'url della funzionalità del FW desiderata, effettua una request HTTP sull'url, e una volta recuperati i dati dal FW, restituisce una stringa contenente il documento XML di risposta (Fig.3.4).

Service Implementation

Di seguito Fig.3.5 è raffigurato il diagramma UML, della realizzazione concreta di ogni singolo service. Ciascuna classe implementa l'interfaccia RemoteServiceServlet.

DataType

Questo pacchetto (Fig.3.6) contiene gli oggetti che rappresentano entità fondamentali della suite. Ogni qual volta il FE effettua una richiesta, il FW risponde con XML contenente dati. Il FE a questo punto effettua il parsing del documento XML e crea un oggetto corrispondente che viene restituito al client. Ogni oggetto è di tipo serializzabile (Serializable Java interface), questo perchè deve essere restituito al client come valore di ritorno della chiamata RPC corrispondente. Il Client a questo punto riceve l'oggetto e mostra all'utente i risultati della richiesta.

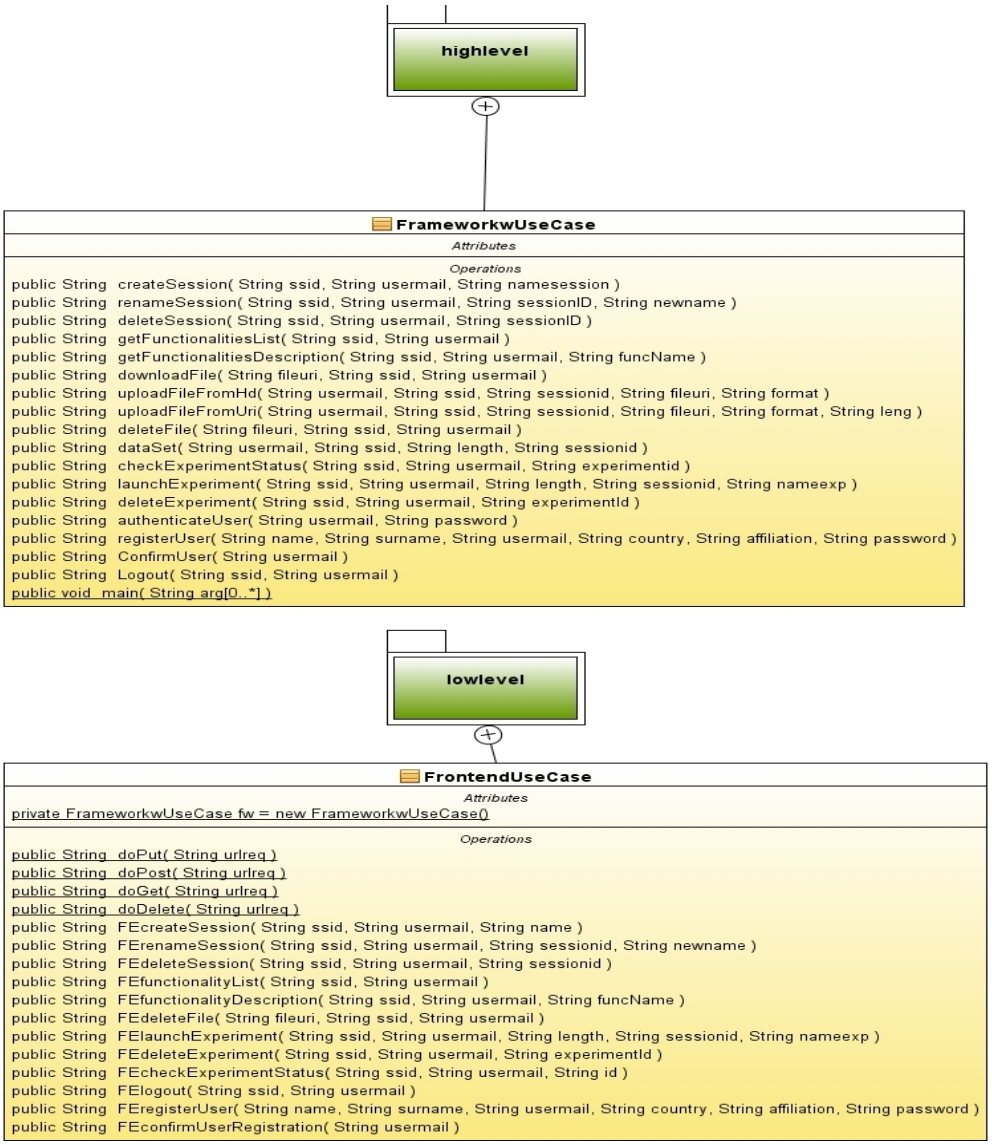

Figura 3.4: Librerie lato Server: highlevel e lowlevel

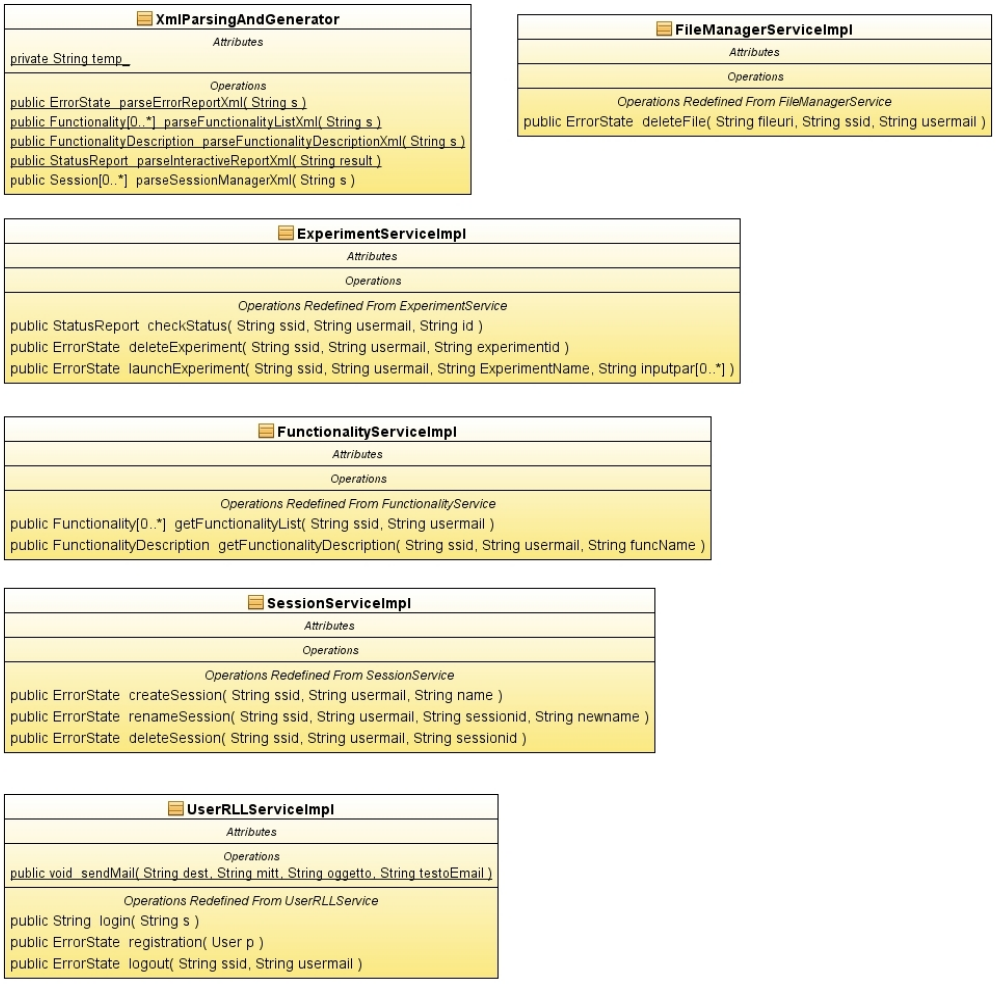

Figura 3.5: Diagramma UML: implementazione concreta Service

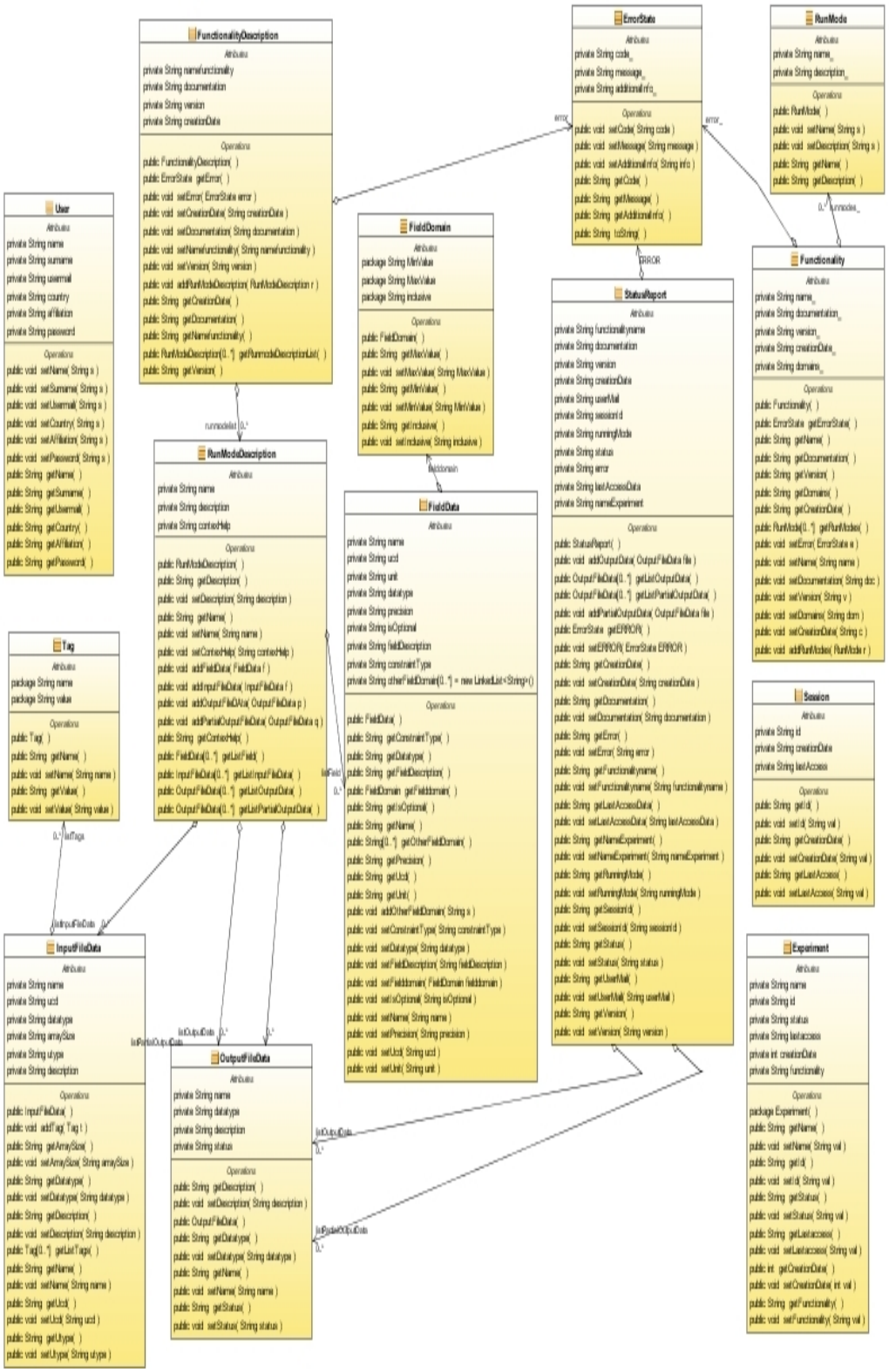

Figura 3.6: DataType FE client-server communication

3.2.2 Client-side

La parte Client è divisa in due parti: la GUI (interfaccia grafica con la quale l'utente interagisce), e le classi che implementano le chiamate alle RPC. Queste classi effettuano le seguenti operazioni:

- la chiamata ad un metodo RPC.
- La deserializzazione dell'oggetto ricevuto dal server.
- L'inserimento dell'oggetto deserializzato nei widget della GUI.
- La validazione dei form,la quale garantisce la correttezza dei dati di input immessi dall'utente. I diagrammi delle classi descritte sono rappresentati in Fig.3.7 e in Fig.3.8

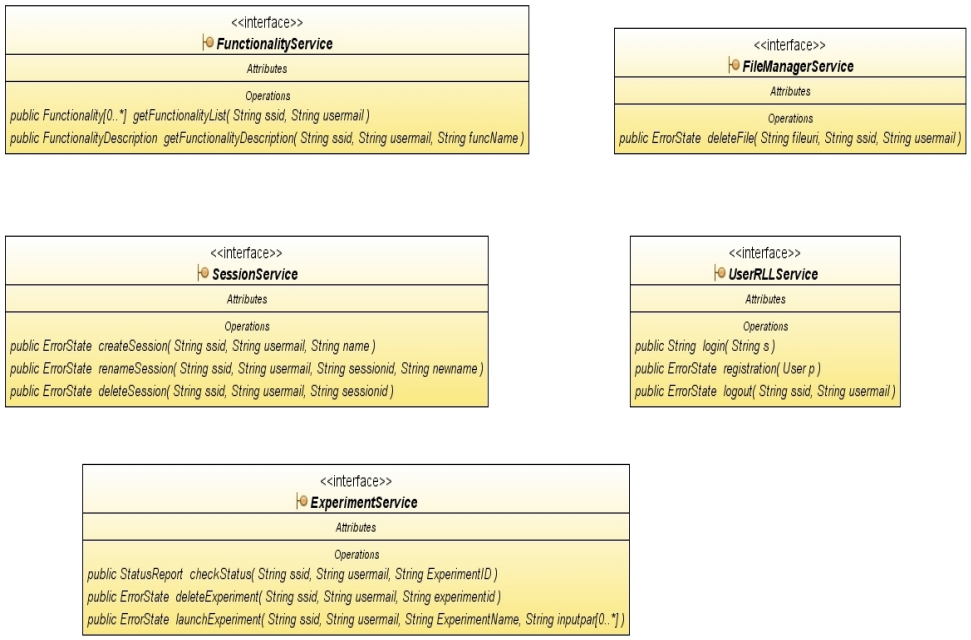

Figura 3.7: Service interface : implementano RemoteService

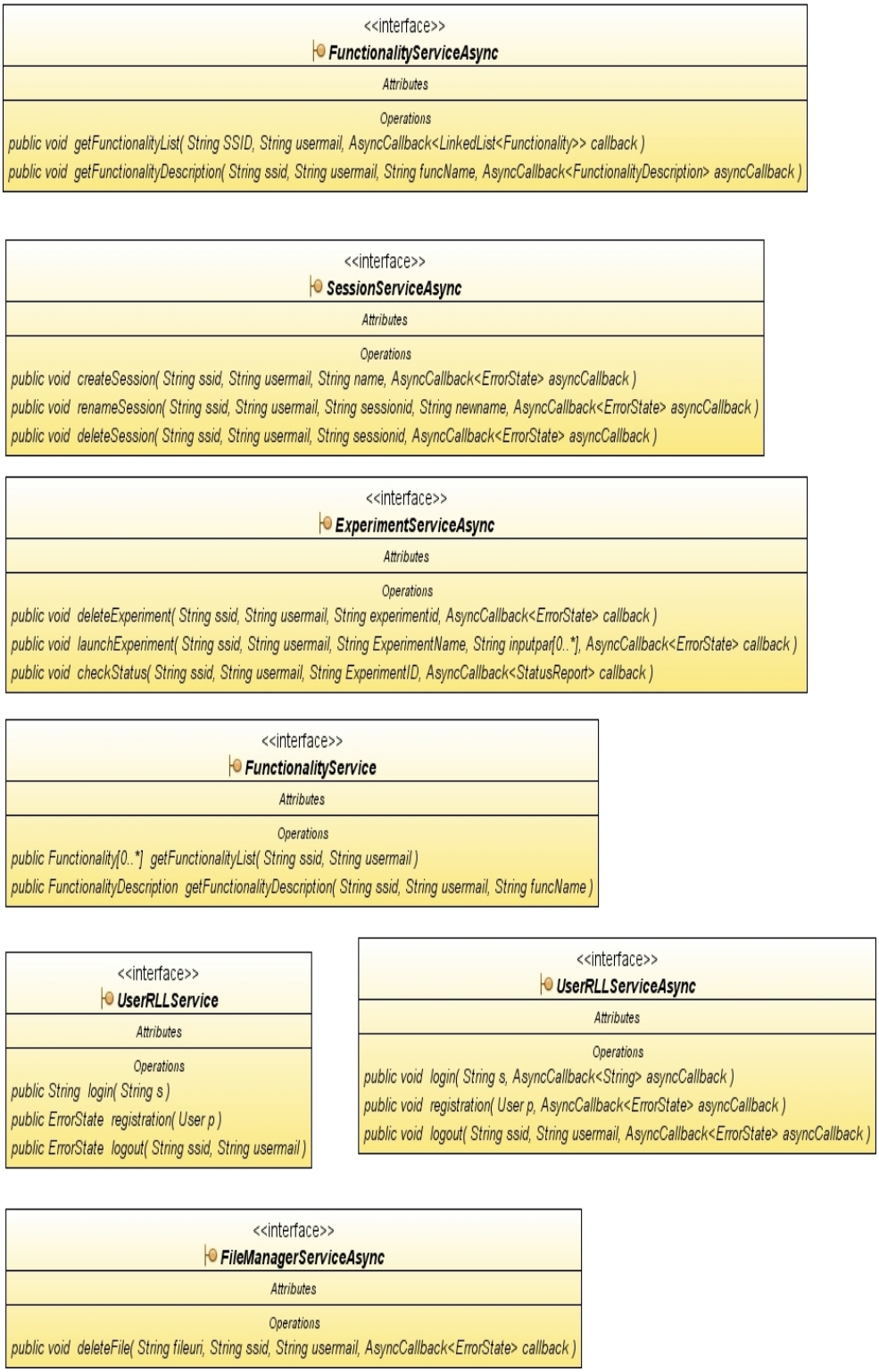

Figura 3.8: Interfacce Service Async: si occupano della serializzazione e deserializzazione degli oggetti Datatype

3.3 Comunicazione con il Componente FW

3.3.1 Interfacce di comunicazione

La comunicazione tra FE e FW avviene mediante documenti XML (discussi nel paragrafo successivo). Ogni qual volta il FE vuole recuperare delle informazioni, effettua delle richieste HTTP al FW. La parte del FW che interagisce con il FE è costituita dal webserver. Il FE dunque possiede le specifiche per effettuare le richieste al FW ed ogni richiesta rappresenta un particolare caso d'uso. Sono elencate di seguito Fig.3.9.

Use Case	Url
Crea Sessione	POST /sessions? ssid=x&userMail=y&name=z
Rinomina Sessione	PUT /sessions? ssid=x&userMail=y&sessionId=z
Cancella Sessione	DELETE /sessions? ssid=x&userMail=y&sessionId=z
Lista Funzionalità	GET /functionalities/list? ssid=x&userMail=y
Descrizione Funzionalità	GET /functionalities/description? ssid=x&userMail=y
Download File	GET /files? fileUri=x&ssid=y&userMail=z
Delete File	DELETE /files? fileUri=x&ssid=y&userMail=z
Upload da HD	POST /files/uploadFromHd? userMail=x&ssid=y&sessionId=z& fileUri=r&format=t
Upload da Uri	POST /files/uploadFromUri? userMail=x& ssid=y&sessionId=z& fileUri=r&format=t&length=u
Dataset Editor	POST /Dataset? userMail=x&ssid=y&length=z& sessionId=t
Crea Esperimento	POST/experiments/newExperiment ? ssid=x&userMail=y&length=z& sessionId=r&experimentName=t
Cancella Esperimento	POST /experiments/delete? ssid=x&userMail=y&experimentId= z
Controlla Stato Esperimento	GET /esperiments/status? ssid=x&userMail=y&experimentId= z
Registra Utente	PUT /users/newUser? name=u&surname=v&userMail=w &country=x&affiliation=y& password=z
Conferma Registrazione Utente	POST /users/newUser ?userMail=x
Login	GET /users/authentication? userMail=x&password=y
Logout	GET /Logout?ssid=x&userMail=y

Figura 3.9: Use case e url FE-FW

3.3.2 XML

XML è un meta-linguaggio per definire la struttura di documenti e dati. Concretamente, è un file di testo che contiene una serie di tag, attributi e testo secondo regole sintattiche ben definite. In un documento XML un elemento è la porzione di testo racchiusa fra $< >$. Gli elementi possono avere associate alcune informazioni che ne descrivono le proprietà. Queste informazioni sono chiamate attributi. L'organizzazione degli elementi segue un ordine gerarchico che prevede un elemento principale, chiamato root o radice. La radice contiene l'insieme degli altri elementi del documento. La struttura logica di un documento XML dipende dalle scelte progettuali, infatti è lo sviluppatore a decidere la maniera più opportuna per organizzare gli elementi al suo interno, dunque non esistono regole universali. Un XML deve avere due caratteristiche principiali, ossia deve essere ben formato e valido. Un documento XML è ben formato quando segue le regole sintattiche proprie del linguaggio XML, ed è definito valido se segue ulteriori regole semantiche che permettono ad uno sviluppatore di definire il linguaggio secondo sue particolari necessità. Queste regole semantiche sono contenute in uno schema, che non è altro che una descrizione dettagliata del contenuto del documento XML. Due esempi di schema sono Document Type Definition (DTD) e XML Schema. Il DTD risulta essere il più semplice e supportato schema per XML. I principali limiti di DTD sono: una certa rigidità nella definizione del documento e la mancanza di estendibilità. Questa ultima preclude l'utilizzo delle nuove versioni di XML aventi nuove caratteristiche, quali ad esempio i namespaces, i quali consentono di risolvere ambiguità nell'assegnazione dei nomi agli elementi, separandoli in diversi domini. Un vantaggio di DTD è però la sua semplicità. XML Schema è stato creato come successore di DTD. Il suo vantaggio principale è quello di essere più versatile, consentendo di imporre ai documenti vincoli più complessi. Ad esempio si possono imporre vincoli sui tipi di dati del documento, cosa non possibile in DTD. I principali vantaggi derivanti dall'utilizzo di XML, e che hanno spinto il team di sviluppo ad utilizzarlo come standard di comunicazione, sono:

- Alta portatilità, essendo un semplice documento di testo.
- Flessibilità, infatti permette di definire con semplicità i propri dati.
- Automazione di Validazione, questo perchè XML possiede delle regole sintattiche ben precise.
- Basato su standard internazionali.

In particolare all'interno del progetto DAME si è scelto di adottare XML Schema per la definizione della struttura dei documenti. Lo schema adottato è quello VOTable.

3.3.3 Lo Schema VOTable

VOTable è uno schema definito dal Virtual Observatory[6] per rappresentare in maniera efficiente dati tabulari di tipo astrofisico. Principali caratteristiche di un documento XML con schema VOTable sono elencati di seguito:

- **Struttura gerarchica.** Gli elementi di una VOTable possono essere combinati in maniera ricorsiva, in modo da rappresentare concetti più complessi.

- **Separazione di dati e meta-dati.** I meta-dati, ovvero i nomi delle colonne della tabella, vengono definiti in una sezione precedente ai loro valori effettivi. VOTable pone una grossa enfasi sui meta-dati, permettendo di organizzare le colonne in gruppi, e di specificarle in maniera precisa, definendo parametri come il loro tipo, la loro grandezza e la precisione dei loro valori. Questa separazione inoltre può consentire un parsing più efficiente di un documento. Infatti se è necessario conoscere solo la struttura della tabella, non è necessario scorrere tutto il documento, ma basta fermarsi alla fine della sezione dei meta-dati. Nel caso in cui sia necessario costruire una propria rappresentazione della tabella del documento, è possibile creare la struttura mentre si scorrono i meta-dati per poi assegnare, in maniera semplice i valori quando si giunge alla sezione dati.

- **Utilizzo degli Unified Content Descriptors (UCD).** A ogni colonna di una tabella viene associato un UCD, che rappresenta il suo tipo di dato. Un client può fare quindi il parsing di un XML senza sapere il nome o la posizione effettiva delle colonne, ma limitandosi a considerare gli UCD a cui è interessato.

All'inizio di ogni schema VO-Table, dopo un header contenente informazioni sulla versione di XML utilizzata, viene posto l'elemento VOTABLE. Tra i figli di VOTABLE ci sono, solitamente,DESCRIPTION e INFO, che permettono di definire parametri generici, riguardanti il documento. Gli elementi più importanti contenuti in una VOTABLE sono però le RESOURCE.

[6]Il Virtual Observatory è un sistema di virtualizzazione di database astrofisici, derivanti da osservazioni/simulazioni, per la federazione dei dati secondo standard di rappresentazione definiti ad hoc.

Una risorsa è un insieme di tabelle correlate. Oltre ad alcuni parametri descrittivi della risorsa, questa contiene uno o più elementi TABLE, rappresentanti le tabelle del documento. Tra gli elementi più importanti di una tabella ci sono i campi FIELD e PARAM. Un campo FIELD rappresenta una colonna della tabella. Un PARAM è invece una costante. Questi campi possono avere vari attributi, come un UCD che definisca meglio il tipo del campo. Ogni tabella contiene una sezione TABLEDATA, dove sono inseriti i dati delle colonne corrispondenti. Dato che la suite deve trattare prevalentemente dati astrofisici, VOTable rappresenta la soluzione ideale, essendo strutturato proprio per questo scopo. Si è preferito utilizzare un singolo schema, in modo da semplificare la generazione e la validazione dei documenti XML[7]. Il FE comunicherà con il FW mediante cinque documenti XML[8]:

1. **Error Report XML:** contiene informazioni riguardante messaggi di errore, in particolare il codice di errore, il messaggio e un campo di informazioni addizionali.

2. **FunctionalitiesList XML:** contiene informazioni sulle funzionalità supportate dal FW, in particolare la descrizione dettagliata e il dominio.

3. **Interactive Report XML:** contiene informazioni riguardanti un esperimento e il suo stato, compresi i suoi file di output.

4. **Session Manager XML:** contiene informazioni riguardanti i workspace, gli esperimenti, e i file di input/output.

5. **Functionality Description XML:** rappresenta il documento di descrizione dettagliato di una funzionalità, contiene i suoi dati di input, output e informazioni sui parametri.

3.3.4 Generazione e parsing XML

La generazione e il parsing dell' XML sono operazioni che verranno eseguite lato server. Effettuare il parsing di un documento XML, significa leggerne il contenuto ed estrapolarne i dati ritenuti necessari. Esistono due tipologie di Parser:

1. **Simple API for XML (SAX)**, segue uno standard basato sugli eventi: il documento viene elaborato carattere per carattere dal parser, che

[7]Per ulteriori dettagli `http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/IvoaVOTable`

[8]Appendice A A.2

genererà un determinato evento ogni qualvolta un particolare comando viene trovato. Il grande vantaggio di SAX è che non bisogna conoscere la struttura del documento,

2. **Document Object Model (DOM)**, segue uno standard più sofisticato e di conseguenza più dispendioso in termini di risorse di sistema, in quanto l'intero documento XML viene caricato in memoria e viene rappresentato in una struttura ad albero di oggetti-nodo, la quale può essere modificata tramite semplici chiamate API. Questa tipologia di parser segue lo standard W3C.

Entrambi gli approcci presentano pro e contro.
La scelta del parser, in base alle esigenze è ricaduta su JDOM[9]. JDOM è una libreria open source, progettata per elaborare e creare un documento XML in modo semplice e diretto. L'idea di JDOM è quella di fornire uno strumento per la gestione di documenti XML con i meccanismi che si aspetterebbe uno sviluppatore Java. Principali vantaggi che hanno portato alla scelta di questo strumento sono:

- il suo modello a oggetti è molto più pratico di DOM. Ad esempio il poter creare gli oggetti direttamente con il costruttore è sicuramente più immediato che usare metodi factory;
- l'utilizzo delle collection semplifica inoltre notevolmente il codice necessario ed è un idioma di programmazione molto vicino al programmatore Java.
- Il futuro di JDOM è molto promettente anche perchè il progetto è stato proposto come Java Specification Request e potrebbe essere incluso in una delle prossime versioni di Java.
- Si propone di incorporare le caratteristiche migliori di DOM e SAX. Infatti si tratta di una libreria leggera con lo scopo di elaborare gli XML in maniera veloce e con poco spreco di memoria.

3.4 Standard Error Handling Policy

Per uniformare i codici di errore è stato definito uno standard composto da una stringa di 13 caratteri nella maniera seguente. Come è possibile vedere in Fig.3.10, ogni stringa di errore porta con se una notevole quantità di informazioni. La scelta di realizzare una gestione degli errori in questa

[9] `http://www.jdom.org`

Figura 3.10: Stringa di Errore comune a tutti i componenti della suite.

maniera ha permesso durante la fase di implementazione e di testing la facile individuazione e correzione degli errori. Un esempio di stringa di errore generato dal componente FE può essere la seguente: **aa01010201002**.[10]

3.5 Sicurezza della Suite

Nella prima release della suite, visti gli scopi di quest'ultima (ricerca scientifica), il team di sviluppo non ha ritenuto opportuno introdurre meccanismi di sicurezza eccessivamente complessi. La suite è stata progettata seguendo il modello iterativo per lo sviluppo del software, dunque ha rimandato a versioni successive l'approfondimento della sicurezza. Nella prima release infatti ci si è concentrati sulle funzionalità e l'usabilità. I meccanismi di sicurezza introdotti alla fine di questo primo ciclo di sviluppo sono i seguenti:

- **Verifica degli IP**: ogni volta che il FE contatta il Framework quest'ultimo effettua un controllo sull'indirizzo IP del oggetto chiamante per accertarsi che sia uno dei Front-End autorizzati. In questo modo si limita la possibilità che un qualunque utente possa accedere direttamente al Framework, eventualmente effettuando operazioni dannose.

- **Security Session Id (SSID)**: dopo che un utente effettua un log-in, il Framework genera un SSID: una stringa di 8 caratteri, criptati in md-5. A ogni interazione di un utente il Front-End dovrà inviare il suo

[10]Per ulteriori dettagli si rimanda al documento di Specifica dei Requisiti: frontend-VONEURAL-SRS-NA-0001-Rel1.3

SSID, che verrà confrontato con quello generato dal FW per capire se l'utente è autorizzato ad effettuare delle richieste.

3.6 Testing

The software is done. We are just trying to get it to work...
Da questa citazione è chiaro che ogni qual volta che si sviluppa del software, l'ultima fase di un processo di sviluppo è proprio quella di verificare se tutto funziona come ci si aspetta. Questa fase prende il nome di Testing. Esistono vari modi per testare il corretto funzionamento di un prodotto software. Un modo è ad esempio quello di effettuare l'ispezione manuale del codice, cioè controllare tutto il codice sorgente. Questo per ovvi motivi comporta un grande dispendio di tempo. Una modalità di testing che è quella che ho attuato per testare il componente FE , è quello dell'ispezione dinamica. Questa modalità prevede due approcci:

- Black-box, il sistema è trattato come una scatola nera.
- White-box, consiste invece nel testare il codice delle unità, forzando i diversi percorsi al suo interno nei vari casi di test.

L'unica strategia di testing che è stata utilizzata ad oggi per il FE è la black-box. Dato che la suite non è completa nella sua totalità è stato possibile fare solo l'unit testing del componente FE e non l'integrazione con il FW (unico componente con il quale interagisce). Infatti tutte le richieste che il FE effettua al FW sono state simulate con delle servlet implementate ad hoc.

In seguito sicuramente verranno anche adottate la metodologia white-box per un testing più approfondito del sistema, e possibilmente a seconda del man-power anche metodologie statiche.

Conclusioni

In futuro oltre a del testing più approfondito, non vi sono ulteriori funzionalità previste e non implementate. Sicuramente in seguito si presenterà la necessità di modificare parti della GUI, per adempire le preferenze dell'utente che utilizzerà la suite, ma questo non porterà alcun tipo di problema. Il punto di forza dell'intera suite è la modularità, infatti gran parte della lunga fase di studio dei requisiti e progettazione è stata incentrata proprio su questa caratteristica. Essendo comunque una prima release della suite rimangono alcune cose da approfondire riguardanti l'intero progetto DAME:

- Sviluppo del DRMS. Al momento ogni Framework possiede un solo Driver, associato a GRID o ad una macchina Stand-Alone. In futuro è prevista la possibilità di inserire più driver su uno stesso Framework, che comunicherà quindi con un DRMS, che sceglierà il driver più appropriato per la determinata situazione.
- Implementazione di nuove politiche di sicurezza. La suite al momento utilizza un meccanismo di sicurezza estremamente semplice. Nonostante non necessiti di meccanismi di sicurezza avanzati, sarà comunque necessario approfondire questo aspetto.

Appendice A

Appendice

A.1 Diagramma Use Case e Tabelle di Cockburn

In Fig.A.1 è rappresentato il diagramma Use Case del sistema[1]. Le funzionalità del sistema sono state suddivise in tre package:

- Access Management
- Session Management
- Task and Functionalities Management
- File Management

Ciascuna funzionalità è spiegata dettagliatamente nella relativa tabella di Cockburn.

[1] Un diagramma use case è una rappresentazione grafica e compatta di tutte le funzionalità o servizi offerti così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso

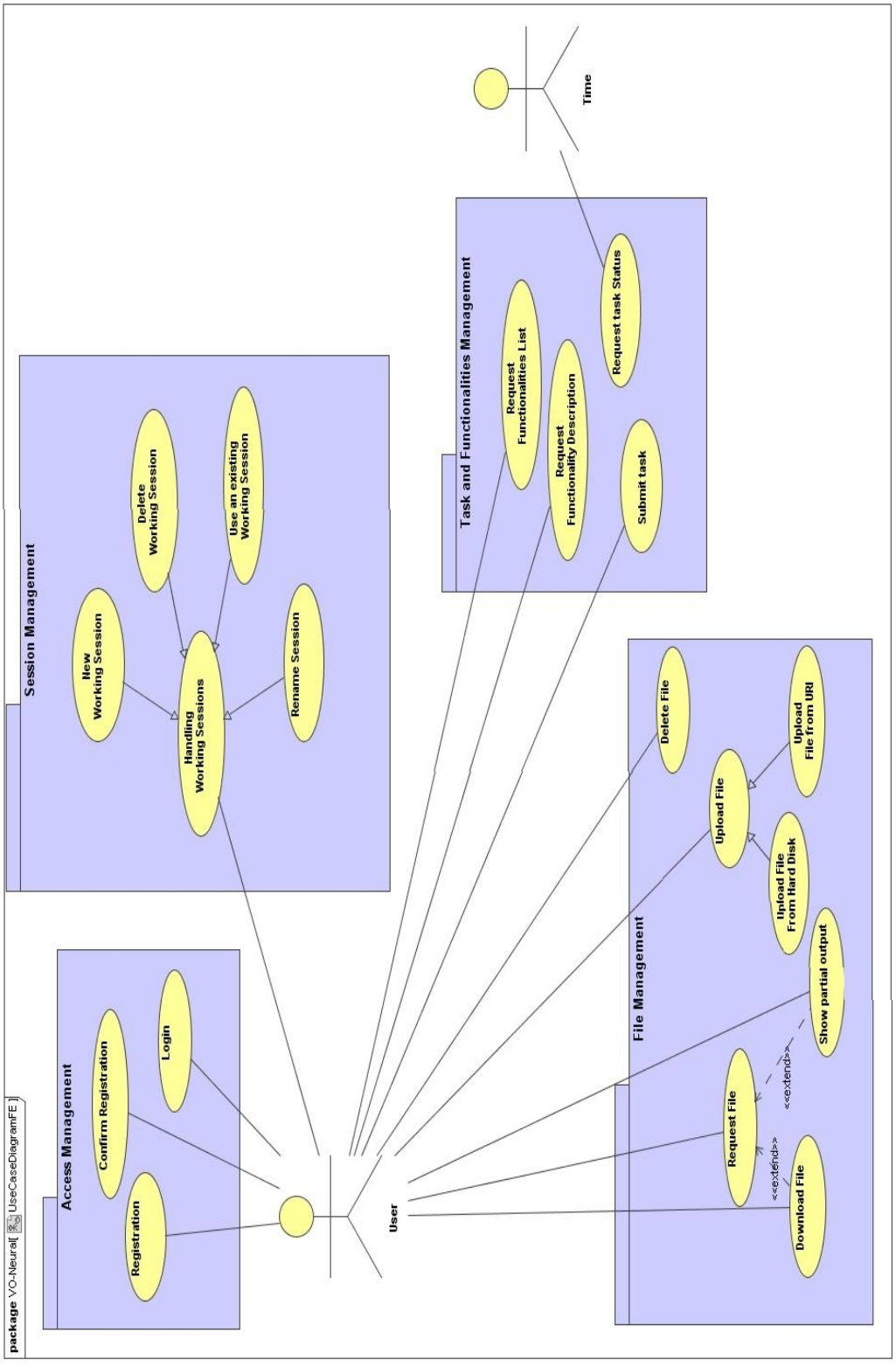

Figura A.1: Diagramma Use Case FE

USE CASE 1	Registrazione		
Goal in Context	L'Utente vuole ottenere le credenziali per accedere al sistema		
Preconditions			
Success End Condition	Registrazione avvenuta con successo		
Failed End Condition	Registrazione non avvenuta con successo		
Primary, Secondary Actor	Utente		
Trigger	L'utente accede al form di Registrazione		
DESCRIPTION	Step	User	FE
	1	Accede al form di Registrazione	
	2		Mostra i campi da inserire per la registrazione
	3	Inserisce le informazioni richieste	
	4	Clicca sul bottone Registra	
	5		Manda al FW una richiesta di registrazione
	6		Riceve un messaggio di OK dal FW
	7		Manda una e-mail di conferma dell'avvenuta registrazione
EXTENSIONS A:			
Utente giá registrato	6a		Riceve un messaggio di KO dal FW
	7a		Mostra un messaggio di errore

Tabella A.1: Registrazione

USE CASE 2	Conferma Registrazione		
Goal in Context	L'Utente vuole attivare il suo account		
Preconditions	Use case Registrazione, e-mail di attivazione ricevuta		
Success End Condition	Attivazione avvenuta con successo		
Failed End Condition	Attivazione non avvenuta con successo		
Primary Actor	Utente		
Trigger	L'Utente clicca sul link di attivazione		
DESCRIPTION	Step	User	FE
	1	Clicca sul link di attivazione	
	2		Effettua una richiesta di conferma registrazione al FW
	3		Riceve un messaggio di OK dal FW
	4		Mostra messaggio di attivazione avvenuta
EXTENSIONS A:			
Utente già attivato	3a		Riceve un messaggio KO dal FW
	4a		Mostra il messaggio: Utente gia registrato .

Tabella A.2: Conferma Registrazione

USE CASE 3	Login		
Goal in Context	L'Utente vuole loggarsi al sistema		
Preconditions	Utente non loggato		
Success End Condition	Utente loggato		
Failed End Condition	Utente non loggato		
Primary, Secondary Actor	Utente		
Trigger	Inserisce le credenziali di accesso		
DESCRIPTION	Step	User	FE
	1	Inserisce le credenziali di accesso	
	2		Effettua una richiesta al FW con le credenziali
	3		Riceve un messaggio di OK dal FW contenente anche il codice SSID
	4		Crea un file cookie contenente il codice SSID.
	5		Mostra le sessioni di lavoro dell'utente
EXTENSIONS A:			
Credenziali non valide	3a		Riceve un messaggio di KO dal FW
	4a		Mostra un messaggio di errore

Tabella A.3: Login

USE CASE 4	Crea Nuova Sessione		
Goal in Context	L'Utente vuole creare una nuova sessione di lavoro		
Preconditions	Utente loggato		
Success End Condition	Sessione di lavoro creata		
Failed End Condition	Sessione di lavoro non creata		
Primary, Secondary Actor	Utente		
Trigger	L'Utente seleziona Crea nuova Sessione		
DESCRIPTION	Step	User	FE
	1	Seleziona Crea Nuova sessione	
	2		Manda una richiesta di nuova sessione al FW
	3		Riceve un messaggio di OK e l' ID della sessione dal FW
	4		Mostra la nuova sessione
EXTENSIONS A:			
Sessione non creata	4a		Riceve un messaggio di KO dal FW
	5a		Mostra un messaggio di errrore

Tabella A.4: Crea Nuova Sessione

USE CASE 5	Cancella Sessione		
Goal in Context	L'Utente vuole cancellare una sessione di lavoro		
Preconditions	Utente loggato, sessione di lavoro esistente e aperta		
Success End Condition	Sessione cancellata		
Failed End Condition	Session non cancellata		
Primary, Secondary Actor	Utente		
Trigger	L'Utente seleziona cancella sessione		
DESCRIPTION	Step	User	FE
	1	Seleziona cancella sessione	
	2		Manda una richiesta al FW di cancella sessione
	3		Riceve un messaggio di OK dal FW
	4		Mostra la lista delle sessioni risultanti
EXTENSIONS A:			
Sessione non cancellata	3a		Riceve un messaggio di KO dal FW
	4a		Mostra un messaggio di errore

Tabella A.5: Cancella Sessione

USE CASE 6	Utilizza sessione esistente		
Goal in Context	L'Utente vuole utilizzare una sessione di lavoro giá esistente		
Preconditions	Utente loggato, almeno una sessione esistente		
Success End Condition	Sessione selezionata e pronta all'utilizzo		
Failed End Condition	Sessione non selezionata		
Primary Actor	Utente		
Trigger	L'utente seleziona una sessione dalla lista		
DESCRIPTION	Step	User	FE
	1	Seleziona una sessione dalla lista	
	2		Manda al FW una richiesta per recuperare i dati della sessione selezionata
	3		Riceve un messaggio di OK le informazioni dal FW
	4		Mostra il contenuto della sessione di lavoro
EXTENSIONS A:			
Sessione non selezionata	4a		Non riceve l'uri del file dal FW
	5a		Mostra un messaggio di errore

Tabella A.6: Usa Sessione già esistente

USE CASE 7	Rinomina Sessione		
Goal in Context	L'Utente vuole rinominare una sessione di lavoro		
Preconditions	Utente loggato, almeno una sessione esistente		
Success End Condition	Sessione rinominata		
Failed End Condition	Sessione non rinominata		
Primary,Secondary Actor	Utente		
Trigger	L'utente seleziona una sessione dalla lista		
DESCRIPTION	Step	User	FE
	1	Seleziona una sessione dalla lista	
	2	Seleziona l'opzione rinomina	
	3	Inserisce il nuovo nome della sessione e conferma.	
	4		Manda una richiesta di rinomina sessiona al FW
	5		Riceve un messaggio di OK dal FW
	6		Mostra la sessione Rinominata
EXTENSIONS A:			
Sessione non rinominata	4a		Receve un messaggio di KO dal FW.
	5a		Mostra un messaggio di errore

Tabella A.7: Rinomina Sessione

USE CASE 8	Richiedi lista funzionalità		
Goal in Context	L'Utente vuole richiedere la lista delle funzionalità		
Preconditions	Utente loggato		
Success End Condition	Lista funzionalità ricevuta		
Failed End Condition	Lista funzionalità non ricevuta		
Primary Actor	Utente		
Trigger	L'Utente richiede la lista delle funzionalità		
DESCRIPTION	Step	User	FE
	1	Richiede la lista delle funzionalità	
	2		Manda una richiesta di funzionalità al FW
	3		Riceve le informazioni inerenti alla funzionalità
	4		Mostra le informazioni ricevute dal FW
EXTENSIONS A:			
Lista non ricevuta	3a		Riceve un messaggio di KO dal FW
	4a		Mostra un messaggio di errore

Tabella A.8: Richiedi Lista funzionalità

USE CASE 9	Richiesta Descrizione Funzionalità		
Goal in Context	L'Utente vuole richiedere la descrizione di una funzionalità		
Preconditions	Utente loggato		
Success End Condition	Descrizione funzionalità ricevuta		
Failed End Condition	Descrizione funzionalità non ricevuta		
Primary Actor	Utente		
Trigger	L'Utente richiede la descrizione di una funzionalità		
DESCRIPTION	Step	User	FE
	1	Requests a Functionality Description.	
	2		Sends a request to FW.
	3		Riceve la descrizione della funzionalità dal FW
	4		Mostra la descrizione della funzionalità richiesta
EXTENSIONS A:			
Descrizione non ricevuta	3a		Riceve KO message From FW.
	4a		Mostra un messaggio di errore.

Tabella A.9: Richiesta Descrizione Funzionalità

USE CASE 10	Crea Esperimento		
Goal in Context	L'Utente vuole creare un esperimento		
Preconditions	Utente loggato, almeno una sessione di lavoro esistente con all'interno dei file di input		
Success End Condition	Esperimento creato		
Failed End Condition	Esperimento non creato		
Primary Actor	Utente		
Trigger	L'Utente inserisce i parametri e seleziona crea esperimento		
DESCRIPTION	Step	User	FE
	1	Inserisce i parametri e seleziona crea esperimento	
	2		Crea il file di configurazione esperimento (XML)
	3		Manda il file xml al FW
	4		Riceve un messaggio di OK dal FW
	5		Mostra il messaggio di avvenuta sottomissione del task
EXTENSIONS A:			
Esperimento non creato	4a		Riceve un messaggio di KO dal FW
	5a		Mostra un messaggio di errore

Tabella A.10: Crea Esperimento

USE CASE 11	Rinomina Esperimento		
Goal in Context	L'Utente vuole rinominare un esperimento		
Preconditions	Utente loggato, almeno una sessione e un esperimento esistenti		
Success End Condition	Esperimento rinominato		
Failed End Condition	Esperimento non rinominato		
Primary Actor	Utente		
Trigger	L'Utente seleziona rinomina esperimento		
DESCRIPTION	Step	User	FE
	1	Seleziona rinomina esperimento	
	2	Inserisce il nuovo nome dell'esperimento	
	3		Manda una richiesta di rinomina esperimento
	4		Riceve un messaggio di OK dal FW
	5		Mostra l'esperimento rinominato
EXTENSIONS A:			
Esperimento non rinominato	4a		Riceve un msg di KO dal FW.
	5a		Mostra un messaggio di errore

Tabella A.11: Rinomina Esperimento

USE CASE 12	Cancella Esperimento		
Goal in Context	L'Utente vuole cancellare un esperimento		
Preconditions	Utente loggato,almeno un esperimento esistente		
Success End Condition	Esperimento cancellato		
Failed End Condition	Esperimento non cancellato		
Primary Actor	Utente		
Trigger	L'Utente seleziona cancella esperimento		
DESCRIPTION	Step	User	FE
	1	Seleziona cancella esperimento	
	2		Manda una richiesta di cancella esperimento al FW
	3		Riceve un messaggio di OK dal FW
	4		Mostra il messaggio di avvenuta cancellazione dell'esperimento
EXTENSIONS A:			
Esperimento non cancellato	3a		Riceve un messaggio di KO dal FW
	4a		Mostra un messaggio di errore

Tabella A.12: Cancella Esperimento

USE CASE 13	Richiesta stato esperimento		
Goal in Context	Visualizzare informazioni sullo stato di un esperimento		
Preconditions	Esperimento creato		
Success End Condition	Status dell'esperimento ricevuto		
Failed End Condition	Status dell'esperimento non ricevuto		
Primary Actor	Time		
Trigger	Time richiede lo stato dell' esperimento		
DESCRIPTION	Step	Time	FE
	1	Richiede lo stato dell' esperimento	
	2		Effettua la richiesta al FW
	3		Riceve le informazioni sull'esperimento e gli uri dei file di output
EXTENSIONS A:			
Status esperimento non ricevuto	3a		Riceve un messaggio di KO dal FW
	4a	ritorna al Passo 2	

Tabella A.13: Richiesta Stato Esperimento

USE CASE 14	Upload File da URI		
Goal in Context	L'Utente vuole effettuare un Upload da Uri		
Preconditions	Sessione di lavoro esistente		
Success End Condition	Upload del file avvenuto con successo		
Failed End Condition	Upload del file non avvenuto		
Primary Actor	Utente		
Trigger	L'Utente seleziona Upload file from Uri		
DESCRIPTION	Step	User	FE
	1	Seleziona Upload file from Uri	
	2	Inserisce l'uri del file	
	3		Manda una richiesta di upload al FW
	4		Riceve dal FW l'uri del file uploadato
	5		Mostra l'uri del file
EXTENSIONS A:			
Upload non effettuato	4a		Riceve un KO message
	5a		Mostra un messaggio di errore

Tabella A.14: Upload File da Uri

USE CASE 15	Upload File da Hard Disk		
Goal in Context	L'Utente vuole fare l'upload di un file dal proprio Hard Disk		
Preconditions	Sessione di Lavoro creata		
Success End Condition	File uploadato con successo		
Failed End Condition	File non uploadato		
Primary Actor	Utente		
Trigger	L'Utente seleziona Upload file.		
DESCRIPTION	Step	User	FE
	1	Seleziona upload file	
	2	Seleziona il file e clicca Submit	
	3		Manda una richiesta al FW che include il file dati
	4		Riceve dal FW l'uri del file
	5		Mostra l'uri del file
EXTENSIONS A:			
File non uploadato	4a		Non riceve l'uri del file dal FW
	5a		Mostra un messaggio di errore

Tabella A.15: Upload da Hard Disk

USE CASE 16	Richiedi File		
Goal in Context	L'Utente vuole effettuare la richiesta di un file		
Preconditions	Sessione di lavoro presente, e esperimento creato		
Success End Condition	File richiesto con successo		
Failed End Condition	File non richiesto		
Primary Actor	User		
Trigger	L'Utente clicca sul file per visualizzare i dettagli del file		
DESCRIPTION	Step 1	User	FE Clicca sul file
	2		Manda una richiesta al FW che include l'uri del file
	3		Riceve un messaggio di OK dal FW che ha controllato se il file è disponibile
EXTENSIONS A:			
Download File	3a	Use Case 17: Downwnload File	
EXTENSIONS B:			
Show Partial output	3b	Use Case 18: Mostra output parziale	
EXTENSIONS C:			
File non disponibile	4c		Riceve un messaggio di KO dal FW
	5c		Mostra un messaggio di errore

Tabella A.16: Richiedi File

USE CASE 17	Download File		
Goal in Context	L'utente vuole effettuare il download un file		
Preconditions			
Success End Condition	Download del file effettuato		
Failed End Condition	Download del file non effettuato		
Primary,Secondary Actor	User		
Trigger	L'utente seleziona l'uri del file da scaricare		
DESCRIPTION	Step	User	FE
	1	Seleziona l'uri del file da scaricare	
	2		Trasferisce il file al client

Tabella A.17: Download File

USE CASE 18	Mostra output parziale		
Goal in Context	Il FE mostra all'utente l'output parziale di un esperimento		
Preconditions			
Success End Condition	Output parziale mostrato		
Failed End Condition	Output parziale non mostrato		
Primary,Secondary Actor	Utente		
Trigger	L'utente seleziona la voce output parziale		
DESCRIPTION	Step	User	FE
	1	Seleziona la voce output parziale	
	2		Effettua una richiesta al componente FW
	3		Mostra l'output parziale

Tabella A.18: Mostra Output Parziale

A.2 XML di Comunicazione

Di seguito il codice degli XML di comunicazione (Descritti nel Capitolo3) tra il FE e il FW:

XML A.1: Error Report

```
<?xml version="1.0" ?>
<VOTABLE version="1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.1
 http://www.ivoa.net/xml/VOTable/v1.1"
 xmlns="http://www.ivoa.net/xml/VOTable/v1.1">

 <DESCRIPTION>Error Report</DESCRIPTION>
  <INFO name="errorCode" value="{errorCode}"/>
  <INFO name="errorMessage" value="{errorMessage}"/>
  <INFO name="additionalInfo" value="{info}"/>
</VOTABLE>
```

XML A.2: Functionalities list

```
<?xml version="1.0" ?>
<VOTABLE version="1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.1
 http://www.ivoa.net/xml/VOTable/v1.1"
 xmlns="http://www.ivoa.net/xml/VOTable/v1.1">

<DESCRIPTION>Functionality List Description</DESCRIPTION>
 <#list func as func>
  <RESOURCE name="{func.name}">
   <DESCRIPTION> a functionality </DESCRIPTION>
    <INFO name="functionality" value="{func.name}"/>
    <INFO name="documentation" value="{func.doc}"/>
    <INFO name="version" value="{func.version}" />
    <INFO name="creationDate" value="{func.Date}"/>
    <INFO name="domains" value="{func.domains}"/>
      <TABLE>
          <GROUP name="RunModes">
            <DESCRIPTION>This Functionaly
          supports different run modes</DESCRIPTION>
            <FIELD datatype="char" name="Train">
                <DESCRIPTION></DESCRIPTION>
            </FIELD>
```

```
            <FIELD datatype="char" name="Test">
                <DESCRIPTION></DESCRIPTION>
            </FIELD>
            <FIELD datatype="char" name="Run">
                <DESCRIPTION></DESCRIPTION>
            </FIELD>
            <FIELD datatype="char" name="Full">
                <DESCRIPTION></DESCRIPTION>
            </FIELD>
          </GROUP>
          <DATA>
            <TABLEDATA>
               <tr><td>{func.trainvalue}</td> </tr>
               <tr><td>{func.testvalue} </td> </tr>
               <tr><td>{func.runvalue}  </td> </tr>
               <tr><td>{func.fullvalue} </td> </tr>
            </TABLEDATA>
         </DATA>
      </TABLE>
    </RESOURCE>
    </#list>
</VOTABLE>
```

XML A.3: Interactive Report

```
<?xml version="1.0" ?>
<VOTABLE version="1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.1
 http://www.ivoa.net/xml/VOTable/v1.1"
 xmlns="http://www.ivoa.net/xml/VOTable/v1.1">

<DESCRIPTION>Functionality Description</DESCRIPTION>
<INFO name="functionality" value="{func.name}" />
<INFO name="documentation" value="{func.doc}"/>
<INFO name="version" value="{func.version}" />
<INFO name="creationDate" value="{func.Date}"/>
<INFO name="user mail" value="{userMail}" />
<INFO name="sessionid" value="{sessionId}" />
<INFO name="runningMode" value="{runningMode}" />
<INFO name="status" value="{status}" />
<INFO name="error" value="{error}" />
<INFO name="lastAccessData" value="{lastAccess}" />
<INFO name="name" value="{expName}" />
```

```
<RESOURCE name="{runningMode}">
<DESCRIPTION>Running Mode Descr</DESCRIPTION>
<TABLE>
 <GROUP name="outputData">
<DESCRIPTION>Output Data </DESCRIPTION>
<#assign i = 1>
<#list outputFiles as of>
<FIELD name="{of.name}" datatype="boolean">
<DESCRIPTION></DESCRIPTION>
/FIELD>
<#assign i = i+1>
                </#list>
<GROUP>
<GROUP name="partialOutput">
<DESCRIPTION>Partial Output</DESCRIPTION>
  <#assign i = 1>
<#list partialOutputFiles as pof>
<FIELD name="{pof.name}" datatype="boolean">
<DESCRIPTION>Partial Output {i} description</DESCRIPTION>
</FIELD>
   <#assign i = i+1>
</#list>
        </GROUP>
          <DATA>
        <TABLEDATA>
                <#if (nOutputFiles>0 )>
                <TR>
                <#list outputFiles as of>
                <TD> of.value </TD>
                </#list>
                <#list partialOutputFiles as pof>
                <TD> pof.value </TD>
                </#list>
                </TR>
                </#if>
        </TABLEDATA>
    </DATA>
 </TABLE>
</RESOURCE>
</VOTABLE>
```

XML A.4: Session Manager

```
<?xml version="1.0" ?>
```

```
<VOTABLE version="1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.1
 http://www.ivoa.net/xml/VOTable/v1.1"
 xmlns="http://www.ivoa.net/xml/VOTable/v1.1">
<DESCRIPTION>Session List</DESCRIPTION>
<INFO name="user mail" value="{userMail}" />
<INFO name="fullname" value="{name}"/>
<INFO name="country" value="{country}" />
<INFO name="affiliation" value="{affiliation}" />
<INFO name="creationDate" value="{creationDate}"/>
<INFO name="ssid" value="{ssid}"/>
<#list sessions as s>
<RESOURCE name="{s.name}" ID="{s.id}">
<DESCRIPTION>A Session</DESCRIPTION>
<INFO name="creationDate" value="{s.creationDate}" />
<INFO name="lastAccess" value="{s.lastAccess}" />
        <TABLE ID="files">
<DESCRIPTION>Session Files
 Uploaded by the user himself</DESCRIPTION>
                          <#list s.files as f>
<GROUP name="{f.name}">
<DESCRIPTION>A filename</DESCRIPTION>
<PARAM name="creationDate" datatype="char"
 arraysize="*" value="{f.creationDate}" />
<PARAM name="lastModification" datatype="char"
 arraysize="*" value="{f.lastMod}" />
<PARAM name="uri" datatype="char"
 arraysize="*" value="{f.uri}" />
</GROUP>
</#list>
</TABLE>
<TABLE ID="Experiments">
<DESCRIPTION>Session Experiments</DESCRIPTION>
<#list s.experiments as e>
<GROUP name="{e.name}">
<DESCRIPTION>An Experiment</DESCRIPTION>
<PARAM name="id" datatype="long" value="{e.id}" />
<PARAM name="status"
datatype="char" arraysize="*" value="{e.status}" />
<PARAM name="lastAccess"
 datatype="char" arraysize="*" value="{e.lastAccess}" />
<PARAM name="errorString"
```

```
datatype="char" arraysize="*" value="{e.error}" />
<PARAM name="functionality"
 datatype="char" arraysize="*" value="{e.func}" />
<PARAM name="creationDate"
 datatype="char" arraysize="*" value="{e.creationDate}" />
</GROUP>
</#list>
</TABLE>
</RESOURCE>
</#list>
</VOTABLE>
```

XML A.5: Functionality Description

```
<?xml version="1.0" ?>
<VOTABLE version="1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.1
 http://www.ivoa.net/xml/VOTable/v1.1"
 xmlns="http://www.ivoa.net/xml/VOTable/v1.1">
<DESCRIPTION>Functionality </DESCRIPTION>
  <INFO name="functionality" value=" {name}" />
  <INFO name="documentation" value=" {documentation}" />
  <INFO name="version" value=" {version}" />
  <INFO name="creationDate" value=" {creationDate}" />
  <!--  Only for submission -->
  <INFO name="userid" value="" />
  <INFO name="sessionid" value="" />
  <!--  Only for submission -->
  <#list mode as mode>
  <RESOURCE name=" {mode.name}">
 <DESCRIPTION>Running Mode Description</DESCRIPTION>
 <INFO name="contextHelp" value=" {mode.documentation}" />
  <TABLE>
   <GROUP name="fieldsData">
  <DESCRIPTION>Input Fields Description</DESCRIPTION>
  <#list mode.inputFields as inputFields>
  <FIELD name=" {inputFields.name}"
  ucd=" {inputFields.ucd}"
  unit=" {inputFields.unit}"
  datatype=" {inputFields.type}"
  precision=" {inputFields.precision}"
   isOptional=" {inputFields.isOptional}">
    <DESCRIPTION> {inputFields.description}</DESCRIPTION>
```

```
31      <#if inputFields.constraintType="RANGE">
32      <VALUES ID="fieldDomain">
33     <MIN value=" {inputFields.constraint[0]}" />
34     <MAX value=" {inputFields.constraint[1]}"
35      inclusive="yes" />
36      </VALUES>
37      </#if>
38      <#if inputFields.constraintType="VALUES">
39      <VALUES ID="otherFieldDomain">
40     <#list inputFields.constraint as constraint>
41     <OPTION value=" {constraint}"/>
42     </#list>
43      </VALUES>
44      </#if>
45    </FIELD>
46    </#list>
47     </GROUP>
48      <GROUP name="inputFilesData">
49
50    <#list mode.inputFiles as inputFiles>
51    <FIELD name=" {inputFiles.name}"
52    ucd="meta.ref.uri;meta.file"
53    datatype="char" arraysize="200" utype="">
54
55      <DESCRIPTION> {inputFiles.description}</DESCRIPTION>
56      <VALUES>
57
58     <#list inputFiles.tags as tags>
59     <OPTION name=" {tags.name}" value="" />
60     </#list>
61      </VALUES>
62    </FIELD>
63    </#list>
64     </GROUP>
65     <GROUP name="outputData">
66    <DESCRIPTION>Output Data Description</DESCRIPTION>
67    <#list mode.outputFiles as outputFiles>
68    <FIELD name=" {outputFiles.name}" datatype="boolean">
69      <DESCRIPTION> {outputFiles.description}</DESCRIPTION>
70    </FIELD>
71    </#list>
72     </GROUP>
73     <GROUP name="partialOutput">
```

```
74    <DESCRIPTION></DESCRIPTION>
75    <#list mode.partialOutputFiles as partialOutputFiles>
76    <FIELD name=" {partialOutputFiles.name}"
77     datatype="boolean">
78      <DESCRIPTION></DESCRIPTION>
79    </FIELD>
80    </#list>
81     </GROUP>
82     <!--  Only For Submission-->
83     <DATA>
84    <TABLEDATA>
85    </TABLEDATA>
86     </DATA>
87     <!--  End Data (for submission only)   -->
88   </TABLE>
89    </RESOURCE>
90    </#list>
91    </VOTABLE>
```

Bibliografia

[1] Ryan Dewsbury: *Google Web Toolkit Applications*, Prentice Hall, (2008)

[2] Robert Cooper, Adam Tacy: *GWT in Action: Easy Ajax with the Google Web Toolkit*, Manning Publications, (2007)

[3] Robert Cooper, Charles Collins: *GWT in Practice*, Manning Publications, (2008)

[4] Vipul Gupta: *Accelerated GWT: Building Enterprise Google Web Toolkit Applications*, Apress, (2008)

[5] Prabhakar Chaganti: *Google Web Toolkit: Gwt Java Ajax Programming*, Packt Publishing, (2007)

[6] Jeff Dwyer: *Pro Web 2.0 Application Development with GWT* , Apress, (2007)

[7] Ian Sommerville: *Ingegneria del Software*, Addison Wesley, (2005)

[8] Cay S. Horstmann, Gary Cornell: *Core Java 2: I Fondamenti*, Apress, (2007)

[9] Cay S. Horstmann, Gary Cornell: *Core Java 2: Tecniche Avanzate*, Apress, (2007)

[10] DAME site, `http://voneural.na.infn.it`

[11] SmartGwt, `http://code.google.com/p/smartgwt/`

[12] SmartGwt forum, `http://www.smartclient.com/`

[13] Google Web Toolkit, `http://code.google.com/intl/it-IT/webtoolkit/`

Ringraziamenti

In quest'ultima pagina desidero ringraziare ed esprimere la mia riconoscenza nei confronti di tutte le persone che, in modi diversi, mi sono state vicine e hanno contribuito al conseguimento di questo traguardo.
In primis voglio ringraziare l' Osservatorio Astronomico di Capodimonte per avermi dato la possibilità di svolgere un tirocinio su nuove tematiche, ambienti e strumenti che durante il corso di studi non ho avuto modo di approfondire.
In particolare un immenso grazie va al Dott. Massimo Brescia, il mio tutor aziendale, e a tutto il team DAME (Prof. Giuseppe Longo, Alfonso, Alessandro, Civita, Giovanni, Mauro, Michelangelo, Omar, Sabrina, Stefano) per aver contribuito alla mia formazione e crescita nel campo professionale per la prima volta in una realtà aziendale. Non dimenticherò la continua disponibilità e pazienza che ognuno di voi mi ha dedicato, gli incoraggiamenti, i consigli e al tempo stesso gli apprezzamenti per quanto realizzato durante l'intera esperienza di tirocinio.
Ringrazio la mia famiglia senza il cui appoggio non avrei mai raggiunto questo risultato: i miei genitori, per il costante sostegno economico che mi ha permesso di condurre una vita priva di altri pensieri se non quello per lo studio, ma soprattutto di quell'aiuto e di quei consigli saggi e di cuore ogni qual volta mi hanno visto pensieroso, stanco, arrabbiato per qualsiasi cosa anche non inerente al mondo universitario; mio fratello Marco, che pur essendo più piccolo, in qualche occasione ha fatto anche da fratello maggiore. Non dimenticherò mai quello che hai fatto per me specialmente in quest'ultimo periodo.
Un grazie va ad una persona che pur non essendo parte della mia famiglia si è dimostrato sempre come tale in qualsiasi circostanza: grazie Vincenzo, per ogni consiglio, per ogni telefonata prima e dopo un esame, per ogni serata trascorsa insieme e per quelle che verranno, e soprattutto per avermi fatto

riscoprire il vero significato della parola *amicizia*.
Infine voglio ringraziare i miei amici conosciuti all'Università. Ringrazio Davide, con cui ho condiviso la maggior parte degli studi e affrontato diversi esami (*non mollando mai!!!*), ma soprattutto perché insieme abbiamo inseguito passioni e aspirazioni in comune e mi auguro che continueremo a farlo. Ringrazio Daniele, anzi il Dott. Daniele, per tutti i dubbi risolti e per le sue perle di saggezza durante le pause pranzo. Ringrazio Nino (anche se ha il brutto vizio di bloccare su Skype :P) per avermi sopportato durante i progetti di vari esami.

Concludo ringraziando tutte le altre persone che nel bene e nel male hanno contribuito a fare di me quello che sono.

Grazie a tutti di cuore.

Dicembre 2009
Francesco