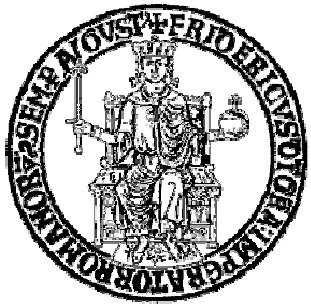
UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II



FACOLTÀ DI SCIENZE
MATEMATICHE, FISICHE E NATURALI

Corso di Laurea in Informatica

Sviluppo di strumenti grafici su una Web Application per il data mining.

Tesi sperimentale di Laurea Triennale

Tutor Accademico
Prof. Marco Faella

Tutor Aziendale
Dr. Massimo Brescia

Candidato
Giovanni Albano
matr. N86/54

ANNO ACCADEMICO 2011/2012

Indice

Elenco delle figure

INTRODUZIONE

I settori della ricerca legati all'ICT (Information & Communication Technology) hanno rapidamente consolidato la loro funzione di strumento essenziale per molteplici ambiti di speculazione scientifica, al punto da identificare la scienza informatica come un elemento comune e integrante per la moderna ricerca multi-disciplinare. In molti settori scientifici di base, fra cui l'astronomia e l'astrofisica, la ricerca si basa ormai sulla necessità di esplorare enormi quantità di dati, spesso eterogenei e derivanti sia da osservazioni reali che da simulazioni complesse e computazionalmente onerose. In breve, molte scienze sono ormai considerate dato-centriche, ponendo cioè l'analisi dei dati al centro della speculazione teorica e sperimentale. Il problema odierno è dunque imperniato sull'eterogeneità e interoperabilità tra grandi archivi di dati. Gli obiettivi di questo nuovo paradigma della scienza moderna sono dunque focalizzati sulla creazione di strumenti informatici in grado di permettere agli scienziati di esplorare, in modo semplice, affidabile ed efficiente i dati, rendendo al contempo trasparente e asincrono l'uso di infrastrutture di calcolo geograficamente delocalizzate. Le moderne infrastrutture computazionali richiedono dunque l'esigenza, da parte degli scienziati, di superare qualunque barriera tra il calcolo e l'acquisizione di conoscenza sui dati (Fabbiano et al. 2010).

I dati possono essere di molti tipi: tabelle, immagini, grafici, osservati o simulati, risultato di analisi statistiche o acquisiti da sensori di vario genere. Inoltre la recente introduzione del concetto di metadato, unitamente al proliferare di risorse computazionali ad alte prestazioni e a costo contenuto, hanno rapidamente contribuito alla proliferazione di data warehouse basate su tecniche OLAP (On-Line Analytical Processing), in grado di sfruttare sistemi efficienti di analisi e visualizzazione dati. La grande quantità di dati multi-dimensionali da gestire pone l'accento sulla reale capacità di interpretarli e utilizzarli in modo efficiente e comprensibile dalla mente umana, di cui è ben nota la limitazione naturale della visione a 3 dimensioni. D'altra parte, nel mondo reale, spesso i dati acquisiti non sono direttamente interpretabili e comprensibili. O perché sono oscurati da alcune informazioni ridondanti o fonti di rumore, oppure perché hanno bisogno di essere fusi con altri dati.

Il "quarto paradigma della scienza" pone in primis, ancora prima della rappresentazione o della strategia di memorizzazione, il problema della comprensione dei dati. Questo infatti pone in termini scientifici una metodologia per ricavarsi informazioni utili a partire da una conoscenza senza pregiudizi e da datasets di qualsiasi tipo (Brescia & Longo, 2012). Di norma questo paradigma impone l'uso di strumenti informatici

efficaci e versatili, in grado di colmare il divario tra le limitate capacità umane (in termini di tempo di elaborazione) ed una crescita costante della quantità e della complessità dei dati oggi in nostro possesso.

Il presente lavoro di tesi va proprio in questa direzione. L'argomento discusso riguarda la progettazione e sviluppo di strumenti software specializzati nella rappresentazione efficiente di grandi volumi di dati. Rappresentazione intesa in termini di creazione di grafici (istogrammi, scatter plot 2D e 3D) e di visualizzazione avanzata di immagini ad alta risoluzione (in particolare immagini astronomiche ricavate da speciali sensori a milioni di pixel). Grandi volumi di dati poiché si tratta di esplorare graficamente cataloghi relativi a centinaia di migliaia di pattern, ciascuno formato da centinaia di elementi (dimensioni dello spazio dei parametri). In questo caso tools grafici d'indagine analitica permettono di individuare correlazioni nascoste tra parametri, classificare oggetti simili ma sparsi in un grande iperspazio e/o realizzare analisi di trend temporali o spaziali. Infine efficiente in termini di scalabilità rispetto alle dimensioni dei dati, pur rispettando vincoli progettuali imposti dall'infrastruttura entro cui era previsto l'innesto di questi tools grafici. Infatti gli strumenti realizzati con questo lavoro di tesi sono stati integrati in una web application dedicata al data mining in ambito astrofisico con tecniche di machine learning (DAMEWARE, [1]). Tale infrastruttura ha imposto in fase di progettazione il rispetto di alcuni vincoli tecnici: interazione utente-applicazione basata su browser Web, senza alcuna installazione lato utente; flusso di dati asincrono e basato sul meccanismo servlet e codifica XML delle informazioni; interfaccia grafica basata su SmartGWT.

Il valore intrinseco del lavoro di tesi, nonché il suo elemento di originalità, consistono dunque nell'introduzione di efficienti e versatili strumenti grafici di analisi visuale e prospettica all'interno di un'infrastruttura di analisi ed esplorazione di dati complessi, creando un unico sistema completo di indagine scientifica. I vari tools grafici consentono infatti alla web application di fornire all'utente, tutti i principali strumenti di pre- e post-processing, potendo analizzare e manipolare i dati con piena cognizione. In ambito prettamente astrofisico e legato ai sistemi di data mining, l'integrazione di tali tools hanno consentito al progetto DAMEWARE di poter competere con i principali sistemi di esplorazione dati disponibili sul Web. Servizi come Topcat, [2], o Aladin, [3], piuttosto che Knime, [4], Orange, [5], o Weka, [6], nella maggior parte dei casi presentano caratteristiche parzialmente deficitarie in termini di tool di rappresentazione o di elaborazione dei dati. DAMEWARE, attraverso i tools grafici integrati, completa e coniuga entrambe le tipologie di esigenze, permettendo la realizzazione di esperimenti e workflow scientifici completi in un unico servizio.

1. La Web application DAMEWARE

L'esplosione del progresso tecnologico nei campi dell'elaborazione digitale, dell'informatica, del calcolo ad alte prestazioni , del calcolo distribuito, dei telescopi astronomici e degli strumenti di piano focale, impone un nuovo approccio nel fare scienza, in grado di fronteggiare in maniera efficiente l'arrivo di uno "tsunami" di petabyte di dati raccolti in tutto il mondo negli archivi e nei data center.

Queste considerazioni hanno spinto il gruppo di collaborazione DAME (I cui membri ufficiali sono il Dipartimento di Fisica dell'Università Federico II di Napoli, l'INAF Osservatorio Astronomico di Capodimonte di Napoli e l'istituto CALTECH della California), a perseguire i loro obbiettivi da una nuova, più organizzata, coerente ed efficiente prospettiva. L'idea consisteva, dunque, nel creare un'unica infrastruttura di servizi di data mining, in grado di coniugare in modo omogeneo e multi-disciplinare le esigenze di sperimentazione scientifica in campo astrofisico con lo stato dell'arte delle tecnologie informatiche.

Per di più, la conseguenza immediata è stata la consapevolezza che tale infrastruttura potrebbe rappresentare un mezzo per realizzare il "quarto paradigma della scienza" (Hey et al., 2009), per le scoperte future nel campo della scienza, in particolare nell'astrofisica. In altre parole, un prodotto da condividere con l'intera comunità scientifica in maniera "facile e aperta".

"Aperta" significa facilmente estendibile in termini di funzionalità e di modelli di data mining, in grado di essere impiegati nella ricerca astrofisica e nell'esplorazione dei dati in generale.

Mentre il termine "facile" si riferisce alle caratteristiche dei servizi offerti agli utenti della comunità, in termini di alta potenza di calcolo e applicazioni scientifiche di facile utilizzo, disponibili "in un click", tramite un semplice web browser.

In alter parole, questo prodotto eredita gli aspetti tecnologicamente avanzati messi a disposizione degli utenti in modo assolutamente trasparente, lasciando che essi concentrino le loro energie mentali per organizzare ed eseguire esperimenti scientifici e flussi di lavoro in generale.

1.1 Caratteristiche della web-app

Il progetto si basa su cinque componenti software principali: *Front End (FE), Framework (FW), Registry&DB (REDB), Driver (DR) e Data Mining Models (DMM)*. Lo schema seguente in Figura 1 mostra il diagramma dei componenti dell'intera Web App con la loro principale interfaccia/layout di scambio informazioni.

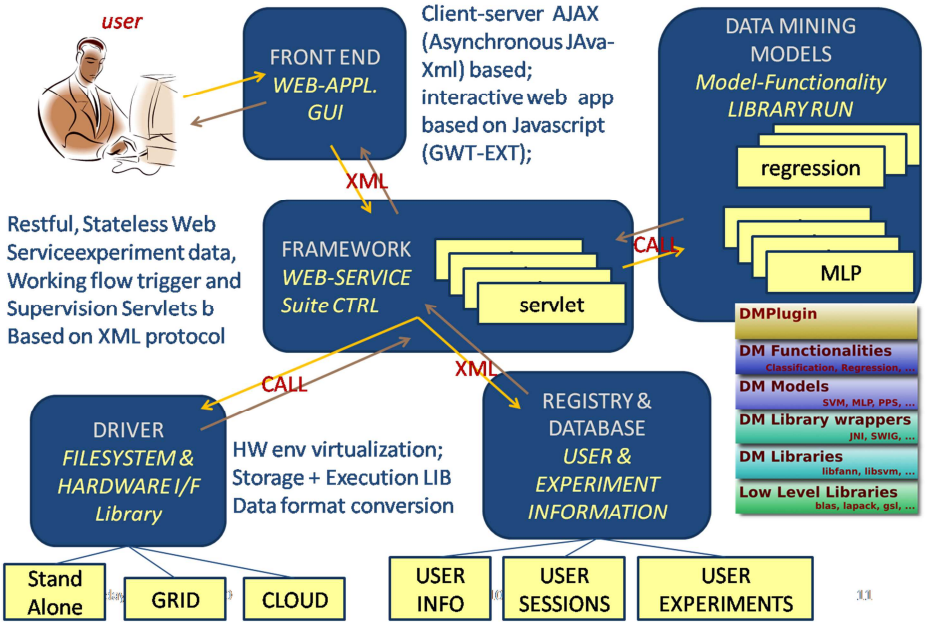


Figura 1: Diagramma dei Componenti della suite

In DAMEWARE viene adottata la seguente terminologia:

- **DM model**: uno dei modelli di data mining integrati nella web app. Può essere un algoritmo di machine learning supervisionato o non supervisionato, a seconda se abbia a disposizione una "Base di conoscenza" (Base of Knowledge o BoK), o meno e in base all'obbiettivo scientifico dell'esperimento dell'utente;
- **Functionality**: uno dei domini funzionali in cui l'utente vuole esaminare I dati disponibili (ad esempio, regressione, classificazione, estrazione di caratteristiche o clustering). La scelta della funzionalità può limitare la scelta del modello di data mining a cui è associata;
- **Experiment**: è la configurazione e l'esecuzione del workflow scientifico (compresa un pre-elaborazione opzionale o una preparazione del dataset) che segue la scelta del modello di data mining e la combinazione delle funzionalità;

- **Use Case**: vi sono esposti all'utente diversi casi di running del modello scelto, che può essere eseguito singolarmente o in una sequenza prefissata. Essendo modelli derivati da un paradigma di machine learning, ogni modello ha training, test, validazione ed esecuzione degli use case, al fine di effettuare, rispettivamente, le fasi di apprendimento, verifica, validazione ed esecuzione. Nella maggior parte dei casi, c'è anche un caso d'uso "completo", che esegue automaticamente training e test case come una sequenza intera.

Inoltre, circa gli aspetti progettuali, DAMEWARE è una web application basata sul tradizionale paradigma R&D (Research & Development), il cui processo implica la successione di fasi di sviluppo standard, quali l'intervista agli utenti, le specifiche dei requisiti, la descrizione del progetto, nonché le procedure di realizzazione e verifica (la relativa documentazione si basa su regole stabilite dagli standard IEEE).

L'architettura del design di DAMEWARE è stata implementata seguendo la strategia standard LAR (Layered Application Architecture) (Figura 2), che prevede un sistema software basato su una struttura logica a strati, in cui i diversi livelli comunicano tra di loro con regole semplici e ben definite, mediante gli standard dei documenti XML.

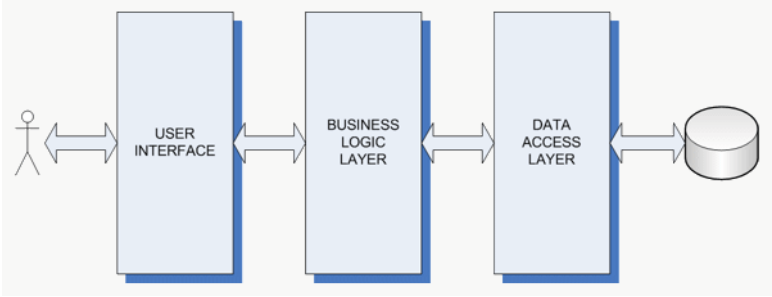


Figura 2: Architettura di un'applicazione a livelli

- ✓ Data Access Layer (DAL): il layer della gestione dei dati persistenti, responsabile del sistema di archiviazione dei dati, compresa la manutenzione della consistenza e dell'affidabilità;
- ✓ Business Logic Layer (BLL): il cuore del sistema, responsabile della gestione di tutti i servizi e delle applicazioni implementate nell'infrastruttura, compreso il controllo e la supervisione del flusso di informazioni;
- ✓ User Interface (UI): responsabile dei meccanismi di interazione tra il BLL e l'utente, compreso l'I/O dei dati e dei comandi ed il rendering dell'interfaccia grafica.

Un altro aspetto importante che deve essere sottolineato è che la web app ha come requisito di progettazione principale l'essere perfettamente compatibile con la tecnologie basate sul web 2.0, che ha portato l'avvento di applicazioni web, cioè applicazioni accessibili tramite web browser.

Una evoluzione diretta di tali strumenti è la Rich Internet Application (RIA), che consiste in applicazioni che possiedono le caratteristiche di interazione e interfacciamento tradizionali delle desktop application, ma usufruibili tramite semplici web browser, e quindi, senza necessità di alcuna installazione sulla macchina locale dell'utente. Le RIA sono particolarmente efficienti in termini di interazione e velocità di esecuzione. Ciò è dovuto al fatto che una parte dell'elaborazione dei dati dell'applicazione associata alla macchina dell'utente è trasferita a livello client, fornendo una reazione istantanea ai comandi dell'utente, mentre i dati e le operazioni di calcolo sono residenti sul lato server remoto, limitando così, anche l'oneroso flusso di comunicazione client-server.

Questi vantaggi evidenti hanno spinto verso la progettazione e lo sviluppo di DAMEWARE come una RIA. Inoltre, il proposito è quello di mantenere e sfruttare queste caratteristiche in tutte le future release.

Le atre line guida alle spalle del design ingegneristico di DAMEWARE sono:

- Modularità;
- Standardizzazione;
- Virtualizzazione dell'hardware;
- Interoperabilità;
- Espandibilità: molte parti dell'infrastruttura dovrebbero richiedere di essere espanse nel corso del tempo. Questo vale in particolar modo per l'architettura di elaborazione, le capacità del framework, le caratteristiche della GUI, le funzionalità e i modelli di data mining (in quest'ultimo caso compresa l'integrazione con gli algoritmi propri dell'utente);
- Interazione asincrona: l'interazione tra utente finale e infrastruttura client-server è basata su meccanismi non sincroni. Da un lato l'utente non è vincolato dal mantenere attiva la connessione con il servizio dopo aver lanciato l'esecuzione di un esperimento, per attenderne il completamento. Dall'altro lato, l'infrastruttura, tramite la tecnologia Ajax (T. Powell, 2008), esegue la gestione delle informazioni relative all'esecuzione di un esperimento senza interferire direttamente con i meccanismi d'interazione con l'utente.
- Indipendenza dal linguaggio di programmazione: si tratta fondamentalmente delle API che

compongono le librerie e i package dei DMM. Sebbene la maggior parte dei modelli ed algoritmi disponibili sono implementati dal gruppo di lavoro DAMEWARE stesso, questo non è considerato vincolante (riuso dei tools e delle librerie già esistenti, integrazione dei tools per l'utente finale, ecc…). Attualmente, la web app mette a disposizione un sistema di wrapping standard basato su Java per realizzare l' interfaccia standard con le API multi-linguaggio;

- Piattaforma orientata al calcolo distribuito ed al composto ibrido delle risorse (GRID) con un'architettura orientata al servizio (Service Oriented Architecture o SOA).

La filosofia principale alle spalle dell'interazione tra l'utente e la web app è la seguente.

La web app è organizzata sotto forma di sessioni di lavoro (da qui in poi chiamate workspace) che possono essere create, modificate e rimosse dall'utente. Si può immaginare l'intera web app come un contenitore di servizi, gerarchicamente strutturato come in Figura 3.

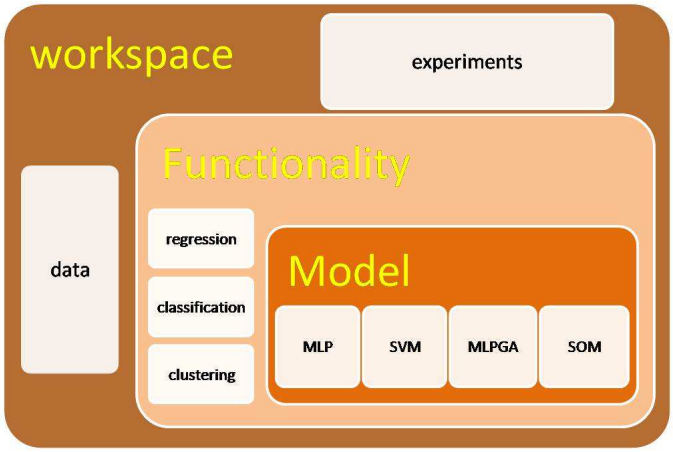


Figura 3: La web app dalla prospettiva dell'utente

Gli aspetti principali sono:

- Un workspace, per contenere tutte le risorse di input/output del lavoro;
- Un editor di dataset, dotato di una serie di funzionalità di pre-processing per modificare e manipolare I dati grezzi caricati dall'utente nel workspace attivo;

- La possibilità di copiare i file in output da un esperimento nel workspace per essere usato come dataset di input per un'esecuzione successiva (l'output della fase di training dovrebbe diventare l'input per la fase di validazione/esecuzione dello stesso esperimento);
- Un insieme di strumenti per il setup di un esperimento, per la selezione del dominio di funzionalità e dei modelli di machine learning da essere configurati ed eseguiti;
- Funzioni per visualizzare risultati grafici e testuali dell'output degli esperimenti;
- Un toolkit per estendere le funzionalità e i modelli di data mining con applicazioni proprie dell'utente;

L'utente può creare tanti workspace quanti desidera. Ogni workspace contiene una lista di file dati e esperimenti, quest'ultimo definito dalla combinazione tra una dominio di funzionalità ed una serie (almeno una) di modelli di data mining. In linea di massima ci dovrebbero essere diversi esperimenti appartenenti ad un singolo workspace, ottenuti fissando il dominio funzionale e variando leggermente la configurazione ed il setup del modello oppure variando il modello associato.

Un workspace è quindi una sessione di lavoro, in cui l'utente può includere risorse relative ad esperimenti di data mining scientifici. Le risorse possono essere file dati, caricati dall'utente, file ottenuti da qualche manipolazione di questi file (dataset), contenenti sottoinsiemi di file, selezionati dall'utente come file in input per i suoi esperimenti, eventualmente normalizzati o riorganizzati in qualche maniera. Le risorse possono anche essere file di output, cioè ottenuti come risultato di uno o più esperimenti configurati de eseguiti nel workspace "attivo" corrente.

L'utente può creare un workspace nuovo oppure selezionarne uno già esistente, selezionando il nome. Dopo aver aperto il workspace, questo diventa automaticamente un workspace "attivo". Questo implica che ogni successiva azione (manipolazione di file, configurazione ed esecuzione di esperimenti, upload/download di file), risulterà effettuata nel workspace attivo, Figura 4. In questa figura viene mostrata anche la sequenza concettuale delle azioni principali da fare per eseguire un esperimento nella maniera corretta.

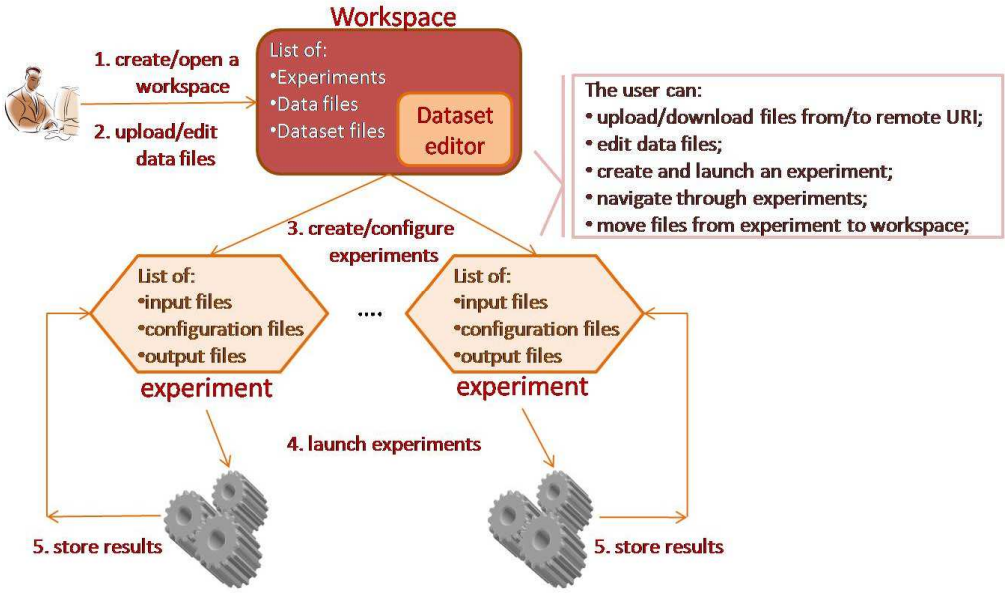


Figura 4: La tipica sequenza per configurare ed eseguire un esperimento

Lo scopo principale di un workspace è rendere facile l'organizzazione degli esperimenti ed i relativi file di input/output all'utente. Per esempio, l'utente può inglobare nello stesso workspace tutti gli esperimenti relativi ad un particolare dominio di funzionalità, pur usando modelli differenti. Inoltre, in ogni momento, è data la possibilità di copiare un file da un esperimento alla lista dei workspace, per riutilizzare lo stesso file dati per più sessioni di esperimenti, cioè per eseguire un workflow.

Tornando ai cinque componenti software introdotti sopra, essi interagiscono tra di loro per configurare ed eseguire esperimenti in maniera descritta nel sequence diagram, mostrato in Figura 5.

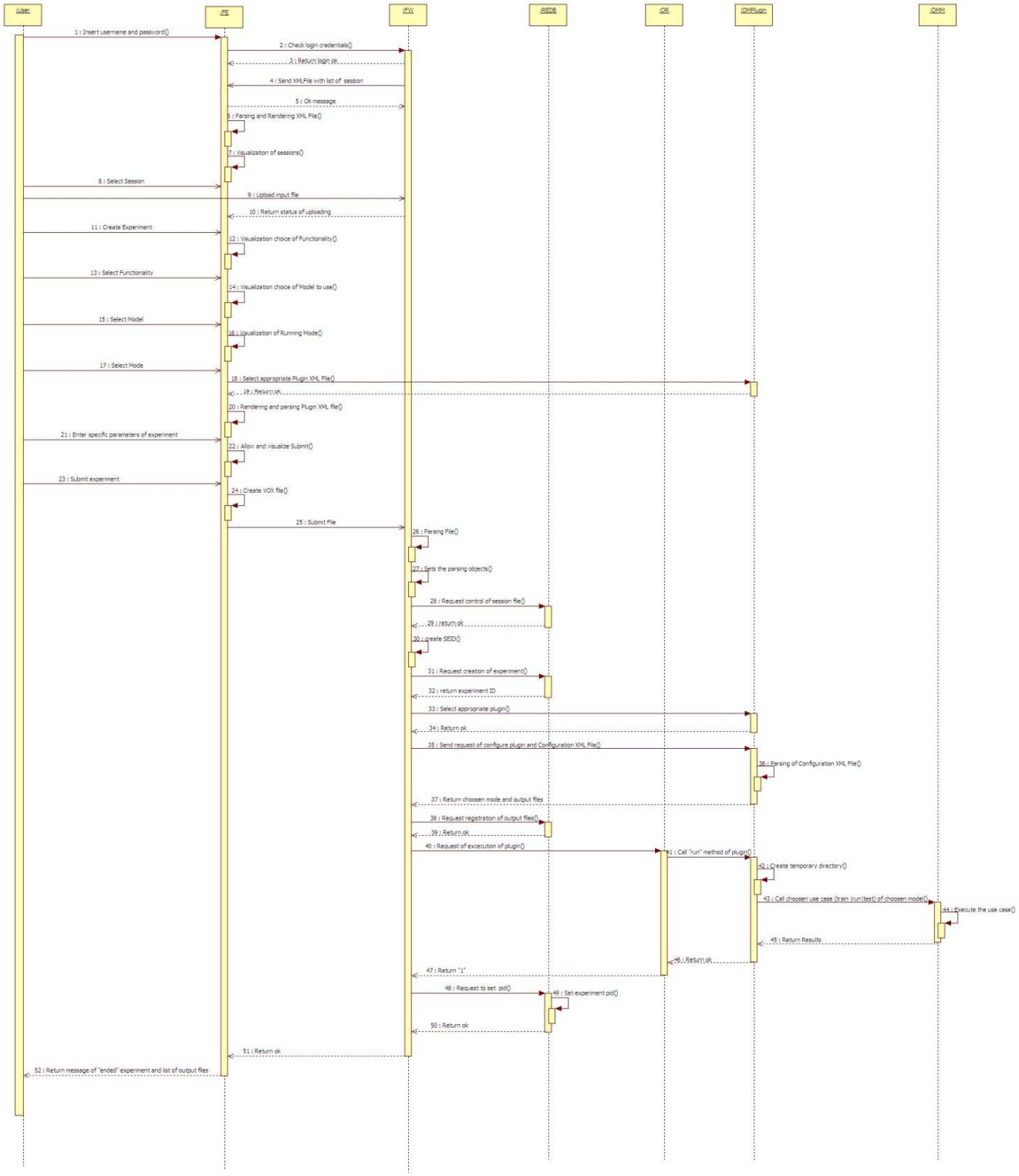


Figura 5: sequence diagram dell'esecuzione di un esperimento

Sono state adottate soluzioni di ingegneria del software specifiche fra i componenti per assicurarsi alcuni comportamenti corretti. Ad esempio, in alcuni casi, sono stati impiegati design pattern specifici. Inoltre, in molti aspetti ingegneristici del progetto della Web App, è stato impiegato un processo di standardizzazione.

All'interno dei componenti della Web App è possibile identificare tre tipi principali di dati da manipolare:

- dati utente;
- dati di elaborazione fra componenti;
- dati di I/O degli esperimenti;

I dati utente sono un insieme di informazioni associate alla sua autenticazione, registrazione e sessioni di lavoro.

I dati di elaborazione fra componenti si riferiscono al flusso di scambio informazioni (comandi, risposte, messaggi) tra i componenti della Web App durante l'esecuzione.

I dati di I/O degli esperimenti si riferiscono ai dati di input ed output dei modelli degli esperimenti.

Al fine di evitare una distribuzione molteplice di questi dati sui diversi componenti, è stato fornito un sistema di immagazzinamento file (FileStore), che ospiti i file dati reali, gestito direttamente dai componenti interni. Un Virtual FileStore è un collegamento logico al FileStore reale, che si trova sul componente FE, necessaria per tutte le operazioni di interazione con l'utente. Essa è basata sullo scambio di metadati tra i componenti FE e FW.

1.2 La GUI (Interfaccia Utente Grafica)

La GUI (Graphical User Interface) della suite si basa su pagine web dinamiche, ottenute tramite il toolkit di Google (GWT), capace di interfacciare l'utente finale con i modelli, le applicazioni ed i servizi per lanciare esperimenti scientifici.

Figura 6: La tecnologia GWT

Per maggiori informazioni sul funzionamento di questo toolkit si rimanda al paragrafo 3.2.

Passiamo a descrivere in dettaglio l'uso pratico dell'applicazione dal punto di vista dell'utente finale.

L'interazione fra l'utente e la GUI è basata sulla selezione di icone, che corrispondono alle funzionalità disponibili di base per eseguire le azioni. Qui di seguito si può vedere la loro descrizione, relative ai cerchi rossi, in Figura 7:

1. **Opzioni del menu dell'header**. Quando uno dei menu disponibili è selezionato, viene visualizzato un sottomenu con alcune opzioni;
2. **Pulsante di Logout**. Alla sua pressione, la GUI (e la sessione di lavoro relativa) viene chiusa;
3. **Tab delle operazioni**. La GUI ha l'aspetto di un browser multi-tab. Tab diverse sono aperte automaticamente quando l'utente vuole modificare o caricare i flie oppure configurare e lanciare esperimenti. Tutte le tab possono essere chiuse dall'utente, eccetto quella principale (Resource Manager);
4. **Creazione di nuovi workspace**. Dopo averlo selezionato e assegnatogli un nome, il nuovo workspace compare nella Workspace List Area (finestra al di sotto del pulsante Workspace);
5. **Workspace List Area**: sezione della tab principale Resource Manager dedicata ad ospitare tutti i workspace definiti dall'utente;
6. **Comando Upload**. Quando viene selezionato, l'utente può scegliere un nuovo file da caricare

nell'area denominata Workspace Data Area (finestra al di sotto di File Manager Area). I file possono essere caricati da un URI esterno oppure dall'hard disk della macchina locale;

7. **Creazione di un nuovo esperimento**. Quando viene selezionato, l'utente è in grado di creare un nuovo esperimento (una nuova tab specifica è aperta per configurare e lanciare l'esperimento);
8. **Comando Rename workspace**. Quando viene selezionato l'utente può rinominare il workspace;
9. **Comando Delete Workspace**. Quando selezionato, l'utente può eliminare il workspace associato (solo se non ci sono esperimenti al suo interno, altrimenti il sistema avverte l'utente di svuotare il workspace prima di cancellarlo);
10. **File Manager Area**. la sezione della tab Resource Manager dedicata alla lista dei file appartenenti a diversi workspace. Tutti i file presenti in quest'area sono considerati possibili file di input per ogni tipo di esperimento;
11. **Comando Download**. Quando viene selezionato l'utente può effettuare il download locale (sul suo HD) del file selezionato.
12. **Comando Dataset Editor**. Quando viene selezionato, viene aperta una nuova tab in cui l'utente può creare/modificare i dataset dei file usando tutte le funzionalità di manipolazione dei dataset;
13. **Comando Delete file**. Quando viene selezionato l'utente può eliminare il file selezionato dal workspace corrente;
14. **Experiment List Area**. La sezione della tab di Resource Manager dedicata alla lista degli esperimenti e ai relativi file di output presenti nel workspace selezionato;
15. **Comando Experiment verbose list**. Quando viene selezionato, l'utente può aprire la lista dei file degli esperimenti (per gli esperimenti in stato "ended", cioè terminati) in modalità dettagliata, mostrando tutti i relativi file creati e memorizzati;
16. **Comando Delete Experiment**. Alla sua pressione, l'intero esperimento (tutti I file in lista) viene cancellato;
17. **Comando Download experiment file**. Quando viene selezionato l'utente può effettuare il download in locale (sul proprio HD) del file di output dell'esperimento relativo;
18. **Comando AddinWS**. Quando viene selezionato, il file correlato è automaticamente copiato dall'area "Experiment List" all'area "File Manager" del workspace attivo al momento. Questa funzionalità è utile per riutilizzare un file in output da un precedente esperimento come file di input per un nuovo esperimento (in figura, vedere il file "weights.txt", che, dopo questa operazione, è anch'esso nella lista File Manager). Un file presente in entrambe le aree, può essere usato come input o come output negli esperimenti.

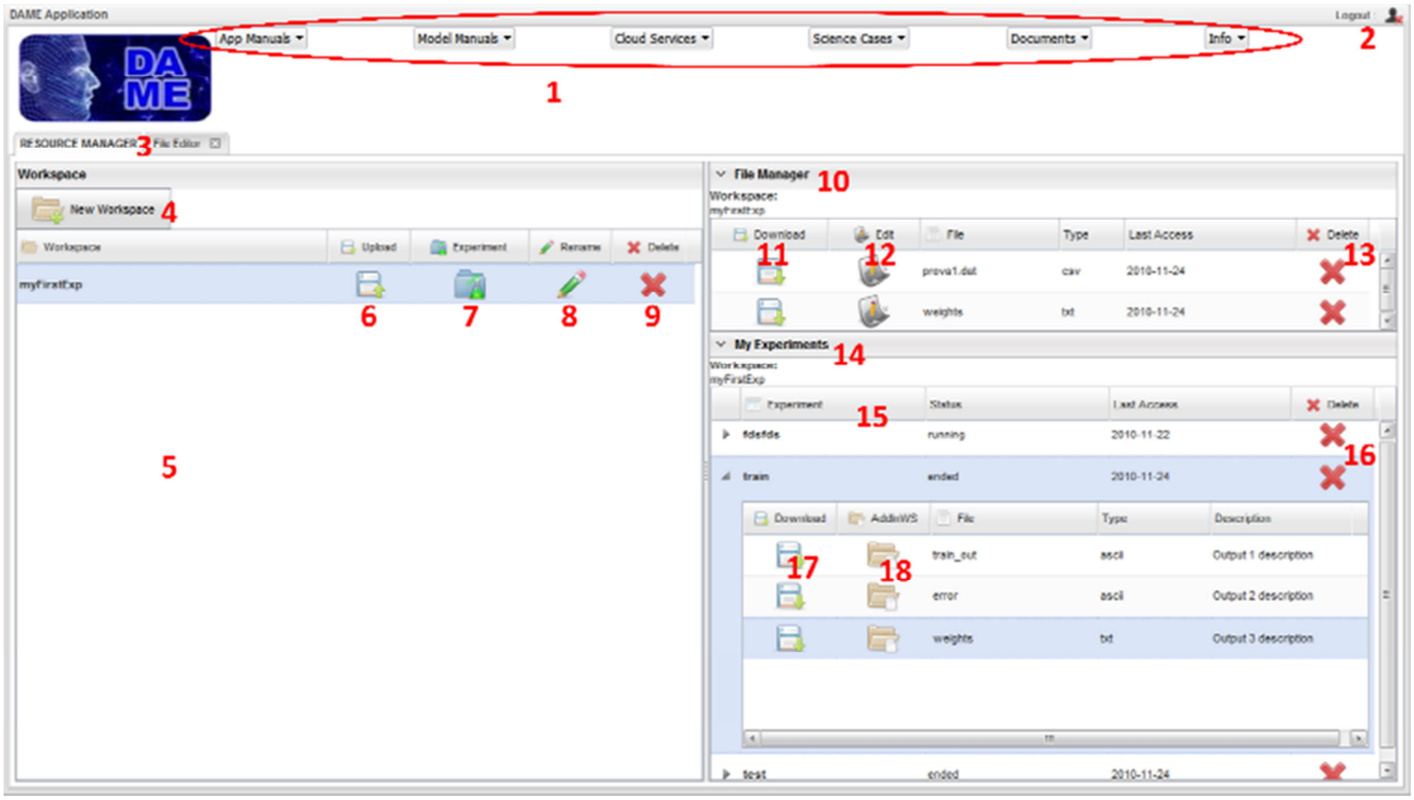


Figura 7: Le principali aree e comandi della Web Application

2. Estensione di DAMEWARE

L'obiettivo del presente lavoro è quello di estendere le funzionalità già esistenti nella web app DAMEWARE, integrandovi specifiche opzioni relative alla creazione di grafici e visualizzazione di immagini.

Il problema della visualizzazione di dati multi-dimensionali rientra tra le principali caratteristiche che permettano all'utente di poter analizzare con piena cognizione i dati di un fenomeno osservativo/simulato, oltre a garantire una facile esplorazione all'interno dello spazio dei dati e di eventuali risultati di esperimenti scientifici.

Dato che la web application è in prima istanza dedicata al data mining in ambito astrofisico, appare ancora più evidente e necessario dotare l'utente di strumenti di ispezione grafica e visualizzazione dati. Gli strumenti integrati sono i seguenti:

- ✓ Grafici
  - o Istogramma: diagramma a barre verticali.
  - o Scatter Plot 2D: chiamato anche grafico di dispersione. Riproduce la dispersione dei punti che rappresentano i dati desiderati.
  - o Scatter Plot 3D: versione 3D del grafico precedente; aggiungendo un asse cartesiano aumentano le caratteristiche dei dati presi in considerazione.
  - o Line Plot: diagramma bi-dimensionale a linee (utile per rappresentare spettri).

- ✓ Visualizzazione Immagini:
  - o viene data la possibilità di consultare cataloghi di immagini provenienti da diverse fonti (plot di grafici, caricamento arbitrario dell'utente, output proveniente da alcuni modelli di data mining), senza dover uscire dal contesto della web app. La consultazione prevede anche operazioni di zoom e panning delle immagini.

L'integrazione di queste funzioni ha richiesto una serie di modifiche e adattamenti del codice preesistente dei componenti della web application. In particolare:

- **Grafiche**:

  - ✓ aggiunta di 2 pulsanti nella sezione **Resource Manager** per effettuare operazioni di creazione di grafici e editing di immagini;
  - ✓ aggiunta di nuove tab per visualizzare e manipolare i grafici prodotti (figura 22) e le immagini (figura 21).

- **Server:**

  - ✓ aggiunta di quattro nuove servlet che rispondano alle seguenti necessità:
    - o invio del nome del file da plottare risponde con i metadati dello stesso.
    - o rispondere alle richieste di plotting richiamando le funzioni delle librerie STILS (restituendo le immagini di plot senza salvarle nel workspace).
    - o salvataggio dei plot nel workspace.
    - o visualizzare i file immagine.

- **Database*:**

  - ✓ aggiunta di **jpg, gif, png** e **eps** ai formati di file supportati.

*Le modifiche al DB sono risultate molto stringate poiché il sistema era già provvisto di una cernita dei tipi di file compatibili; i file sono così divisi:

- supportati (che a loro volta si dividono in):
  - o editable:
    - ▪ fits-table
    - ▪ ascii
    - ▪ csv
    - ▪ votable
  - o non editable (i rimanenti formati accettati ma non modificabili)

- other (tutti i formati che rimangono al di fuori della web app)

precedentemente, tra i formati *other* c'erano anche **jpg, gif, png** e **eps** che ora sono diventati *supportati,* non *editable*.

2.1 Scelte progettuali

Il progetto prevede uno scenario in cui l'utente debba poter visualizzare e manipolare i grafici ottenuti a proprio piacimento partendo dai dati. Per far ciò, è stato necessario accordarsi sul modo in cui il sistema debba produrre i grafici; da un lato c'è la strategia basata sulla produzione lato client, tramite le Google charts API, [7], e dall'altro c'è la strategia che prevede di produrre i grafici esclusivamente lato server. Vediamo in dettaglio i pro e i contro di queste due scelte progettuali:

2.1.1 Strategia orientata al browser

L'idea, in questo caso, è quella di portare l'onere della produzione dei grafici, e quindi anche della loro modifica, interamente sul browser. In questo modo avremmo una notevole miglioria dal punto di vista dell'interattività dell'utente con i grafici, dovuta al fatto che le modifiche di quest'ultimi non necessitano di richieste al server, ma vengono effettuate a spese del client. Di conseguenza, in questo modo, non si andrebbe ad aumentare il carico del server, rispettando anche la natura stessa dell'applicazione (le operazioni di plotting sono considerate secondarie rispetto agli esperimenti sui dati).

D'altro canto, però, i dati utilizzati sarebbero spostati interamente sul client alla prima richiesta, aumentando la latenza di trasmissione dati sulla rete e contravvenendo al paradigma fondamentale di DAME, cioè quello di evitare o quantomeno minimizzare lo spostamento dei dati sulla rete. Oltretutto, il tool per la produzione di grafici di Google che si utilizzerebbe per questa strategia è risultato poco idoneo al nostro scopo; non sono garantite, infatti, tutte le operazioni di manipolazione dei grafici di cui vogliamo dotare l'applicazione, (ad esempio: operazione di linear correlation per Scatter Plot 2D e bin placement per Histogram, come si può vedere dall' editor al link:

https://developers.google.com/chart/image/docs/chart_wizard?hl=it-IT).

2.1.2 Strategia orientata al server

La seconda ipotesi prevede che sia il server ad occuparsi della produzione dei grafici. In questo modo verrebbero evitati i trasferimenti di interi file sulla rete, fornendo al client solo i metadati e rimanendo così in linea con la filosofia attuale della web app. Il browser, infatti, dovrà solo ricevere e visualizzare all'utente i grafici in formato immagine (png), senza doversi assumere l'onere di caricare e manipolare grosse quantità di dati. D'altro canto, però, questo approccio penalizza, e non poco, l'aspetto interattivo rappresentato dalla manipolazione delle opzioni grafiche messe a disposizione dell'utente; ogni cambiamento di queste ultime, infatti, genererebbe una richiesta al server, che si dovrà occupare di ricostruire il grafico ex-novo, seguita dalla risposta contenente il grafico da visualizzare. Dal punto di vista della comunicazione si avrebbero numerosi scambi di messaggi, ma di scarso contenuto (un file immagine non supera mai una certa soglia di grandezza); dal punto di vista del carico di lavoro, il server dovrà produrre grafici di file potenzialmente voluminosi, sottraendo così risorse alle operazioni di esperimenti di data mining. La tecnologia scelta per questo approccio in questo scenario è STILTS (STIL API), descritte nel prossimo capitolo.

2.1.3 Scelta della miglior strategia

In conclusione, la scelta è ricaduta sulla strategia orientata al server. Questa decisione, per nulla facile, è scaturita dalla considerazione che le funzionalità introdotte, sebbene assolutamente importanti, non costituiscono la principale prerogativa della web app. Essa infatti nasce in primis per eseguire data mining sui dati. Inoltre, i benefici che l'utente trarrebbe dalla maggiore interattività data dalla prima strategia, non valgono l'onere di avere e manipolare il file sulla sua macchina. Questo, infatti, unito alla non completezza del tool che si utilizzerebbe, potrebbe spingere gli utenti verso altre applicazioni più complete (es. Topcat).

3. Tecnologie utilizzate

Per soddisfare i requisiti descritti in precedenza, le seguenti tecnologie sono state impiegate considerando le loro caratteristiche principali. Onde rispettare i principali vincoli progettuali della web-app originaria si è proceduto ad utilizzare:

3.1 Asynchronus Javascript and XML (AJAX)

Ajax è una tecnica di sviluppo per la realizzazione di applicazioni web interattive, in particolare di Rich Internet Application. Non si può attribuire una vera e propria paternità a questo strumento, ma colui che ha dato per la prima volta un nome a questa tecnica è Jesse Garrett che, nel 18 Febbraio 2005, ne parlò nel suo Blog. Il concetto alla base di Ajax l'utilizzo asincrono di Javascript che, attraverso l'interfacciamento con XML, permette ad un client di richiamare informazioni lato server in modo veloce e trasparente. Grazie alla potenza di Ajax, le pagine della nostra applicazione possono scambiare piccole quantità di dati con il server senza andare incontro ad un submit di un form ed ad un conseguente ricaricamento dell'intera pagina. Questo rende l'applicazione più performante e più reattiva alle azioni dell'utente. La chiave di questo approccio è l'oggetto XML HttpRequest. Il W3C[1] lo definisce "un'interfaccia esposta da un motore di scripting che consente agli script di eseguire funzionalità sul client HTTP, come il submit di un form di dati o il caricamento di dati da un sito web remoto.".

Ajax ha portato progressivamente a far mutare la natura di buone applicazioni di tipo interattivo verso le applicazioni per il web. Si può prevedere, infatti, che tutte le nuove applicazioni saranno online, anche perché in tal modo vi è possibilità di sfruttare potenza di calcolo ben al di sopra di quella che può generare un semplice personal computer di uso domestico. Nel nostro caso, ad esempio, se si fosse scelto di sviluppare il componente FE come una desktop application, sicuramente non si sarebbero potuti effettuare esperimenti di data mining in tempi ragionevoli, poiché non avremmo avuto l'opportunità di sfruttare la potenza di calcolo dell'infrastruttura di GRID computing di S.Co.P.E.

Tornando ad Ajax, essa è sostanzialmente un approccio basato su tecnologie ben note 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 si evince che Ajax quindi non è una tecnologia ma un insieme di più tecnologie. In particolare include:

[1] World Wide Web Consortium

- XHTML[2] e CSS: per la presentazione della pagina in un formato standard
- DOM[3]: per il layout dinamico e l'interazione con la pagina
- XML e XSL[4]: scambio e manipolazione dati.
- XMLHttpRequest: recupero asincrono dei dati.
- Javascript: collante per tutte le tecnologie precedenti, eseguito lato client.

La figura seguente mostra come queste tecnologie collaborano per effettuare l'aggiornamento di un parte della pagina con nuovi dati provenienti dal server.

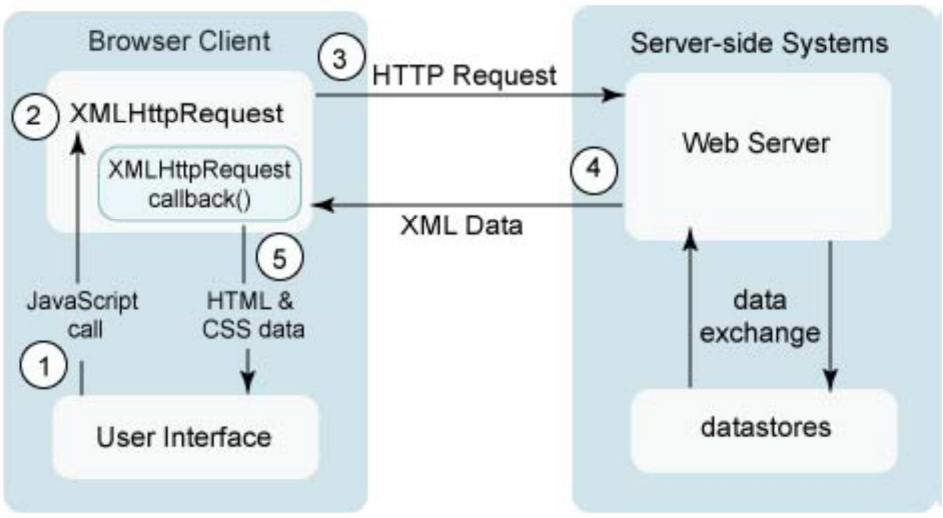


Figura 8 – Sequenza generica di una richiesta Ajax

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.

[2] eXtensible HyperText Markup Language
[3] Document Object Model
[4] eXstensible Stilesheet Language

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.

In generale, le applicazioni web vengono sviluppate utilizzando le richieste HTTP, inoltrate mediante l'oggetto XmlHTTPRequest al web server. Dopo aver effettuato una serie di operazioni, scaturite dall'elaborazione della richiesta, che può essere una semplice raccolta o manipolazione dei dati, viene restituita 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 rappresenta una grande innovazione tecnologica, che, in quanto giovane, prevede ulteriori miglioramenti; eppure essa 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, con l'introduzione del 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 rappresenta una sostanziale differenza perché si ha la possibilità di sfruttare contemporaneamente due piattaforme di elaborazione, una lato client e una lato server, ampliando, così, i confini delle potenzialità delle applicazioni web. Dalla Fig.9 possiamo notare le principali differenze tra il modello tradizionale delle applicazioni web e il modello Ajax. Infatti, col modello Ajax, l'utente procede nella navigazione mentre il server sta elaborando.

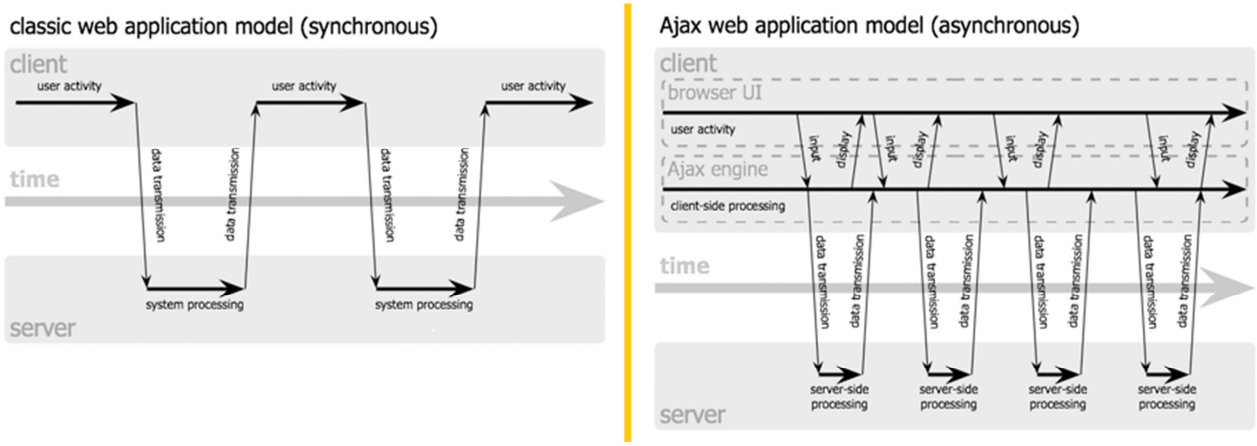


Figura 9 - Interazione sincrona delle tradizionali web application vs Interazione asincrona di un web application con Ajax, [8]

Finora, Ajax è stata descritta come uno strumento che presenta solo aspetti positivi, ma non è così. Sviluppare applicazioni con Ajax non è intuitivo, e richiede al programmatore competenze avanzate. 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, qualora lo sviluppatore decida di realizzare una grossa applicazione con Ajax, debba avere una vasta conoscenza delle caratteristiche dei differenti browser. D'altro canto, per far fronte a questi problemi e per tenere fuori lo sviluppatore da questi aspetti, esistono vari toolkit e librerie come ad esempio Dojo, [9], Sencha Ext js, [10], Script.aculo.us, [11], 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.

3.2 GWT

Google Web Toolkit (GWT) è un toolkit di sviluppo per la costruzione e l'ottimizzazione di complesse applicazioni basate sul browser. Il software e i codici di esempio sono sviluppati da Google e sono rilasciati sotto licenza Apache, v 2.0. Il suo obiettivo è quello di consentire lo sviluppo produttivo delle applicazioni web Ajax-based ad alte prestazioni senza che l'autore debba essere un esperto in browser, XMLHttpRequest, e JavaScript.

Il toolkit GWT, [8], arrivato alla versione 2.5, è risultato essere adatto alla nostra applicazione per le sue caratteristiche intrinseche. La principale caratteristica di questo strumento è quella di permettere di scrivere le applicazioni utilizzando il linguaggio Java e, tramite il compilatore GWT, convertire il codice prodotto in codice Javascript e HTML browser compliant (Internet Explorer, Firefox, Mozilla ect). Lo sviluppo di un'applicazione basata su GWT si può riassumere in questi 4 punti:

1. Utilizzo di un IDE Java (nel mio caso Netbeans) per la scrittura del codice in linguaggio Java, con l'ausilio delle librerie proprie di GWT.

2. Test dell'applicazione (Hosted Mode) come codice Java compilato che gira all'interno della JVM.
3. Conversione di tutte le classi Java tramite il compilatore GWT in un'applicazione web composta da file Javascript e HTML.
4. Verifica dell'applicazione (Web Mode) all'interno del/i browser supportati.

Analizziamo ora in dettaglio l'architettura del framework GWT, rappresentata in Fig. 10.

Come possiamo vedere, quest'ultima è composta principalmente da quattro componenti: compilatore Java-to-JavaScript, un web browser-hosted e 2 librerie Java. In dettaglio:

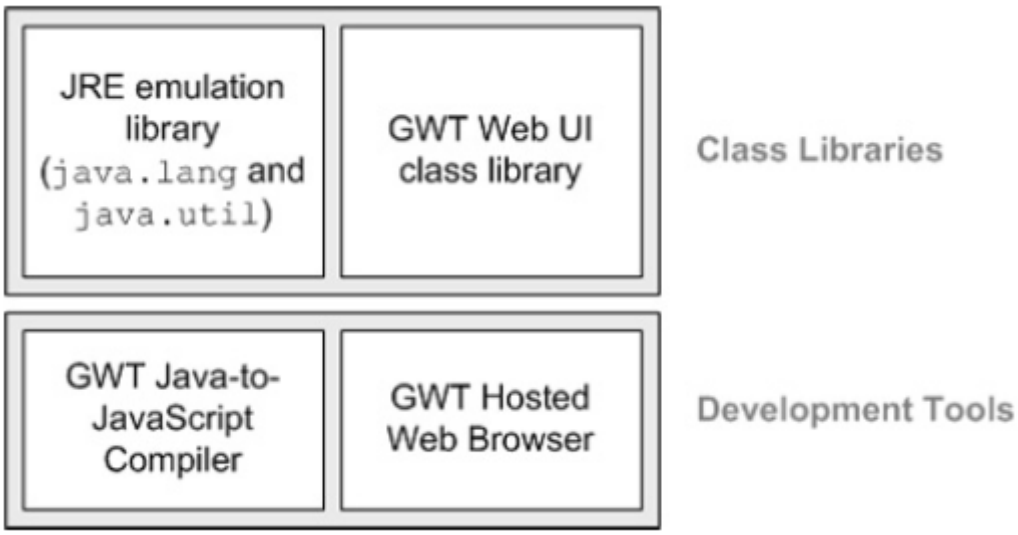


Figura 10 – Architettura GWT

- **Il compilatore GWT Java-to-JavaScript** converte il codice Java in codice Javascript. Viene utilizzato quando si vuole eseguire l'applicazione in modalità web mode. È considerato il cuore di GWT. L'approccio del compilatore di GWT è quello di comprendere ed incapsulare le differenze tra i vari browser e compilare codice Javascript da codice Java. Un aspetto 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.

- **Il GWT HostedWeb Browser** permette di eseguire le applicazioni in modalità hosted mode, cioè 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 per l'interfaccia utente 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.

Il toolkit, così strutturato, ci offre numerosi vantaggi nello sviluppo della nostra applicazione, tra cui:

- ✓ Compatibilità del codice generato dalla compilazione Java-to-Javascript di GWT con tutti i più comuni browser.
- ✓ La possibilità di scrivere la nostra applicazione Ajax-based interamente in Java, avendo così a disposizione tutti i concetti propri del paradigma orientato agli oggetti per la strutturazione del codice, per la riusabilità e per la sua manutenibilità.
- ✓ Il supporto alla comunicazione Client-Server che possiamo considerare come la versione object oriented della comunicazione asincrona alla base di una applicazione Ajax. Questo ha consentito, soprattutto, di spostare tutta la parte relativa alla logica di UI sul Client, alleggerendo di gran lunga il carico del Server.

É noto che quando si sviluppa una RIA, un aspetto fondamentale da tenere in considerazione 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 dover

fare alcun aggiornamento 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;
- *RequestBuilder* è una classe che fa essenzialmente da wrapper a questo oggetto;

vediamoli nel dettaglio.

3.2.1 Remote Procedure Call

Nella comunicazione sul web, un client ha bisogno di inviare richieste ad un server per accedere alle risorse ed ai 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:

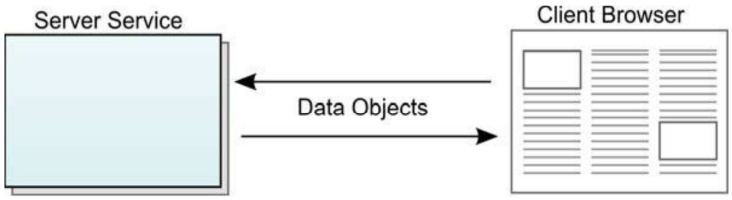


Figura 11- GWT RPC : interazione tra client e server mediante data object

- il servizio, che viene eseguito sul server (es. una servlet o un programma CGI);
- 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 possibilità 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 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[5]

Una volta chiarite queste nozioni vediamo come si implementa un servizio utilizzando GWT-RPC. I passi da seguire sono i seguenti Fig.16 :

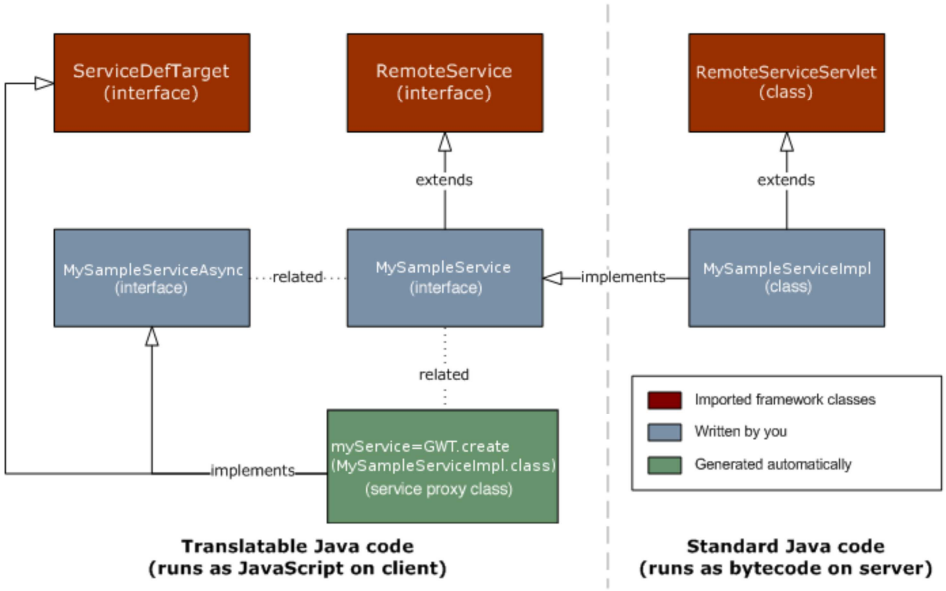


Figura 12 - Il meccanismo GWT RPC

1. Definire un'interfaccia (MySampleService) che implementa RemoteService e definisce al suo interno tutti i metodi RPC che si intendono implementare.
2. Creare una classe (MySampleServiceImpl) che estende RemoteService-Servlet e implementa l'interfaccia creata precedentemente.
3. Definire un'interfaccia (MySampleServiceAsync) che verrà implementata client-side e permetterà di invocare il/i metodo/i definiti nell'interfaccia che implementa RemoteService e ricevere il valore di ritorno (un oggetto).

[5] 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.

3.2.2 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. Alcuni dettagli della risposta del server (status code, HTTP heade, ecc.) possono essere ricavati dagli argomenti dell'oggetto Response. Inoltre, data la natura asincrona delle chiamate http in GWT, l'esecuzione del codice che segue la chiamata del metodo *sendRequest()* è immediata, ed è dunque opportuno monitorare l'oggetto *Response* per verificare la correttezza della risposta.

3.2.3 SmartGWT

Finora abbiamo parlato solo dell'aspetto architetturale e della comunicazione, tralasciando l'aspetto grafico, i.e. la GUI (Grafical User Interface). GWT offre una vasta gamma di widget[6] 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. Quello scelto per i nostri scopi è SmartGWT.

SmartGwt, [9], è basato sulla libreria Javascript SmartClient, [10]. Il codice Javascript auto-generato dal compilatore GWT, effettua delle chiamate alla libreria SmartClient e il risultato viene visualizzato nel browser. La libreria presenta una release stabile. Inoltre SmartGwt risulta tra tutti quello più supportato ed in continua evoluzione. SmartGWT ha permesso quindi di creare una interfaccia grafica molto ricca e valida che rende intuitivo l'utilizzo del software alle sue spalle. Questi, ed altri motivi, hanno portato il team DAME alla scelta di questo widget.

[6] 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.

3.3 Servlet

Per quanto riguarda la programmazione lato server, per il mio lavoro, ho sfruttato le servlet Java. Le servlet sono uno strumento efficiente, flessibile e sicuro per ottenere l'esecuzione di determinate operazioni su una macchina remota oppure per il trattamento e immagazzinamento di dati da client a server.

D'altro parte, però, occorre che il server sia predisposto all'esecuzione delle servlet, e preveda quello che si chiama *servlet engine* o *servlet container*, quale ad esempio Tomcat[7].

Tipicamente, il server deve gestire molte richieste simultaneamente e deve gestirle senza problemi di sincronizzazione e con un occhio alle prestazioni, sia in termini di tempo che di spazio.

Inoltre le servlet permettono di condividere informazioni sia all'interno di un'applicazione che col server.

Nelle servlet, ad ogni richiesta corrisponde un nuovo thread all'interno del servlet engine: thread diversi possono accedere a variabili comuni e questo permette di condividere informazioni tra richieste diverse alla stessa servlet.

Per tutta questa serie di motivi possiamo dire che una servlet estende il server con nuove funzionalità.

Le servlet si presentano come classi Java che possono venir caricate *dinamicamente* per estendere le funzionalità del server. A differenza delle estensioni proprietarie, le servlet vengono eseguite all'interno della Java Virtual Machine sul server, e sono quindi sicure e portabili. Operano solo sul server e quindi non richiedono alcun supporto da parte del browser. Inoltre le servlet offrono un'ottima portabilità sia su sistemi operativi diversi che su server e servlet engines diversi.

Una servlet è in grado di rispondere direttamente alle request HTTP producendo le adeguate response in funzione di parametri che possano essere passati alla Servlet utilizzando i metodi HTTP Get e Post.

Una servlet deve implementare almeno uno dei seguenti metodi:

- **doGet**: per rispondere alle richieste di tipo GET;
- **doPost**: per rispondere alle richieste di tipo POST;
- **init**: il metodo invocato una sola volta prima di ogni altro metodo;
- **destroy**: il metodo invocato prima di distruggere un oggetto di tipo Servlet;
- **getServletInfo**: il metodo usato per recuperare delle informazioni riguardo la servlet;

[7] http://tomcat.apache.org/

- **getLastModified**: il metodo ritorna il tempo dell'ultima modifica fatta.

Le servlet per il Web tipicamente estendono la classe **HttpServlet**. I metodi di questa classe ricevono in ingresso due parametri:

- **HttpServletRequest request:** contiene tutte le informazioni riguardo la richiesta inoltrata dal client. La classe fornisce dei metodi per recuperare sia le informazioni presenti nello header della richiesta che nel corpo.
- **HttpServletResponse response:** tramite questo oggetto è possibile inviare la risposta al client. I metodi messi a disposizione per fare ciò sono il **getWriter()** che restituisce un oggetto di tipo java.io.Writer, ed **il getOutputStream()** che, come lascia intuire il nome, restituisce un oggetto di tipo ServletOutputStream.

Nel nostro caso, le servlet trasmettono il risultato della loro esecuzione al client sotto forma di file XML, che verrà poi analizzato da un parser opportuno (SAX[8]).

Nel caso specifico, la scelta dell'utilizzo delle servlet, aldilà di tutti i vantaggi descritti sopra, è scaturita dal non voler snaturare la web app nelle sue nuove funzionalità da me introdotte. Questo contribuisce a far si che l'applicazione venga vista come un blocco unico e non un agglomerato di funzioni a se stanti.

3.4 STIL/STILTS

Dopo una breve ricerca sulla tecnologia da utilizzare per la produzione di grafici, si può dire che STIL è risultata la più adatta al nostro scopo. Il motivo per preferirla ad altre è la buona copertura dei grafici che vogliamo rappresentare e la facilità di gestione dei dati astronomici.

Più in dettaglio, STIL è una libreria java open source, sviluppata da Mark Beauchamp Taylor, programmatore di software astronomico nel Gruppo di Astrofisica della Scuola di Fisica dell'Università di Bristol, per input e output generici e processing di dati tabulari. Mostra i dati al programmatore dell'applicazione sotto forma di tabelle, che hanno le stesse sembianze indipendentemente dal fatto che provengano da un file FITS, un VOTable, un file di testo ASCII o una query di un database relazionale. In questo modo, l'applicazione non deve preoccuparsi del formato di memorizzazione delle tabelle né in fase

[8] Simple Api for Xml

di lettura né in quella di scrittura, concentrandosi così sull'elaborazione. Il concetto di STIL di tabella include metadati di tabelle e colonne, e tabelle che contengono valori scalari o array mono e multi-dimensionali di dati numerici, stringhe o altri tipi. Per questi motivi, si adatta perfettamente alla gestione di dati astronomici. I formati di input ed output supportati sono VOTable, FITS, SQL, ASCII, CSV e può essere esteso per trattarne altri.

Per il nostro scopo utilizzeremo un tool set di STIL, STILTS. Il pacchetto è stato progettato per tabelle di dati astronomici come cataloghi di oggetti, ma non solo. In qualche modo, STILTS rappresenta la controparte a riga di comando della GUI di Topcat. Il pacchetto è robusto, pienamente documentato, e progettato per essere efficiente, specialmente quando lavora su grandi moli di dati.

La libreria prevede un gran numero di comandi, come quelli per convertire il formato di una tabella, ma noi ci limiteremo ad usare i comandi per il plotting:

- plothist - Histogram plot
- plot2d - 2d scatter plot
- plot3d - 3d scatter plot

Grazie a questi comandi possiamo ottenere la copertura completa nell'ambito dei grafici di cui si voleva dotare la web app.

3.5 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 denire 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 denizione della struttura dei documenti. Lo schema adottato è quello VOTable.

3.6 Design Pattern Singleton

Il Singleton, pattern creazionale, ha l'obiettivo di fornire in tutto il software, una sola ed unica istanza di un determinato oggetto e di fornire un punto di accesso globale ad esso.

Ecco il diagramma UML del Singleton:

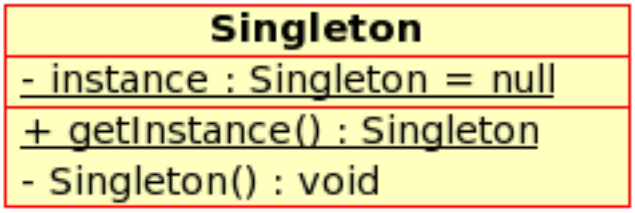


Figura 17 – Diagramma UML del Singleton

Come mostra il diagramma UML, c'è bisogno di una proprietà statica privata, un metodo statico pubblico e

un costruttore privato.

Il codice java per l'implementazione è il seguente:

```
class Singleton {
   private static $_instance = NULL;

   private function __construct() { }

   public static function getInstance() {
      if(is_null(self::$_instance)) {
         self::$_instance = new Singleton();
      }

      return self::$_instance;
   }
}
```

In particolare l'integrazione delle nuove funzionalità ha richiesto una progettazione tale da permettere l'interazione fra le nuove classi ed il pattern che nella web application è usato nella GUI, in particolare nelle seguenti classi:

public class **Application Panel** – si occupa di creare e riempire il pannello della MAIN TAB;

public class **Utility** – fornisce una serie di funzioni di servizio, come ottenere la lista dei file per ogni workspace, oppure il formato di un certo file, ecc.;

public class **WorkspaceList** – si occupa di riempire la lista dei workspace nella MAIN TAB, associandogli i pulsanti per interagirvi ed i relativi ascoltatori.

3.7 Design Pattern MVP

Nello scrivere applicazioni orientate al web si può scegliere fra diversi design pattern architetturali, tra cui Presentation-abstraction-control, Model-view-controller, Model-view-presenter, etc. e sebbene ogni pattern abbia i suoi benefici, si può notare che l'architettura MVP lavora al meglio nello sviluppo di applicazioni GWT per 2 motivi principali. Primo, il modello MVP, così come in molti altri design pattern, disaccoppia lo sviluppo in maniera da consentire a più sviluppatori di lavorare contemporaneamente. In secondo luogo, questo modello ci permette di minimizzare l'uso di **GWTTestCase**[9], che si basa sulla presenza di un browser, favorendo l'uso di test JRE leggeri (e veloci) che non richiedono un browser.

Al centro del pattern c'è la separazione delle funzionalità in componenti che hanno senso logico diverso, ma nel caso di GWT c'è una chiara focalizzazione nel rendere la view più semplice possibile allo scopo di minimizzare la nostra dipendenza da GWTTestCase e ridurre il tempo totale speso per il running dei test.

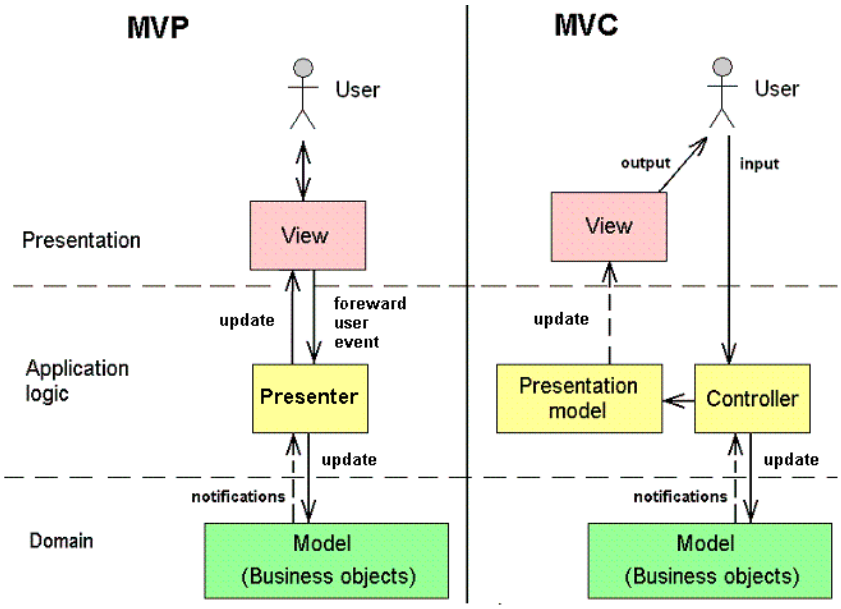


Figura 18 – MVP-MVC

MVP e MVC provano entrambi a risolvere lo stesso problema di separazione di compiti. La differenza principale che si può individuare è che MVC è spesso implementato con qualche accoppiamento tra la view e qualche tipo di model, affidando così alla view il compito specifico di fornire una visualizzazione di un dato oggetto (model).

[9] GWT include una speciale classe base "GWTTestCase" che fornisce l'integrazione JUnit. Mandando in esecuzione una sottoclasse GWTTestCase compilata sotto JUnit viene lanciata un'istanza invisibile del browser GWT hosted mode che serve a emulare il comportamento della nostra applicazione durante l'esecuzione del test.

In MVP invece, generalmente, il presenter si occupa di lavorare con il model, ed è lui che decide quali informazioni è necessario prelevare da quest'ultimo per fornire una sorta di visualizzazione.

All'interno della web application, il pattern si interfaccia con le seguente componenti:

- GUI: coincide con la **View** all'interno del pattern. Il suo unico compito è quello di interfacciarsi con l'utente;
- Framework: coincide con il **Presenter** nel pattern. Contiene tutta la logica dell'applicazione;
- REDB: coincide con il **Model**. Contiene i dati e la logica necessari al presenter.

4. Integrazione tools grafici e visualizzazione immagini

Il lavoro è consistito nell'integrazione di una serie di funzionalità tali da permettere all'utente le principali manipolazioni grafiche dei dati oggetto della web application. In particolare la comunità astrofisica ha necessità di visualizzazione e zoom di immagini, derivate da osservazioni astronomiche, ad alta risoluzione. Inoltre gli utenti hanno l'esigenza di visualizzare plot e istogrammi di cataloghi voluminosi di oggetti astrofisici, onde selezionare e/o ispezionare specifici sottoinsiemi di dati. A tal fine si è proceduto a progettare e sviluppare una serie di funzioni aggiuntive.

Prima di passare alla rappresentazione e alla descrizione di queste funzioni, si vuole tenere presente che l'immersione di nuovi strumenti nella web-app ha richiesto che fosse imposto un requisito base (req. 0), imprescindibile:

Le nuove funzionalità devono essere integrate all'interno della web app esistente, uniformando eventuali scelte implementative alle soluzioni già realizzate.

Il requisito è stato soddisfatto dal momento che per tutta la durata del lavoro sono state seguite le linee guida utilizzate per scrivere la web-app già esistente. All'interno di questo documento si trovano infatti numerosi riferimenti a questo requisito.

Oltre al requisito primario, l'integrazione di nuove funzionalità ha richiesto giocoforza l'analisi approfondita di tutti i componenti e le loro interazioni della web application. Questa analisi ha evidenziato una serie di requisiti progettuali e implementativi, enucleati di seguito:

Requisiti generici:

1. L'utente può accedere alla sezione "Plot maker" dalla main window solo se esiste almeno un file "editable";
2. L'utente può accedere alla sezione "Image Viewer" dalla main solo se esiste almeno un file supportato ma non "editable";

Requisiti della sezione Plot maker:

3. L'utente visualizza le tab relative ai grafici;
4. L'utente può scegliere un workspace ed un file (tra i tipi supportati per questa funzionalità);
5. L'utente può cambiare workspace e file in qualsiasi momento;
6. L'utente può cambiare i parametri per il diagramma "Histogram", che sono: XAxis, Flip, Bin placement, Bar form, Line thickness, Colour, Title, XLabel, YLabel, Grid;

7. L'utente può cambiare i parametri per il diagramma "Scatter Plot 2D", che sono: XAxis, YAxis, XFlip, YFlip, Marker size, Marker shape, color, Title, XLabel, YLabel, Grid, Linear correlation;
8. L'utente può cambiare i parametri per il diagramma "Scatter Plot 3D", che sono: XAxis, YAxis, ZAxis, XFlip, YFlip, ZFlip, Marker size, Marker shape, color, Title, XLabel, YLabel, Grid, Phi, Theta;
9. L'utente può cambiare i parametri per il diagramma "Line Plot", che sono: YAxis, YFlip, Colour, Line Width, Title, XLabel, YLabel, Grid;
10. L'utente può richiedere l'elaborazione dei grafici in qualsiasi momento premendo il pulsante "plot", a patto che tutti i campi obbligatori siano stati settati;
11. Il grafico ottenuto deve poter essere visualizzato;
12. L'utente può selezionare più file per ottenerne un grafico che contenga informazioni su tutto il dataset selezionato;
13. L'utente può richiedere il download del grafico ottenuto in 2 formati diversi: PNG o EPS-GZIP;
14. L'utente può richiedere il salvataggio del grafico all'interno dei workspace della web app;
15. L'utente può ridimensionare i pannelli delle opzioni in ognuna delle tab dei grafici;

Requisiti della sezione Image Viewer:

16. L'utente visualizza la tab relativa alle immagini;
17. L'utente può scegliere un workspace ed un file (tra i tipi supportati per questa funzionalità);
18. L'utente può cambiare workspace e file in qualsiasi momento;
19. L'immagine relativa al file selezionato deve essere visualizzata;
20. L'utente può effettuare lo zoom sull'immagine visualizzata;
21. L'utente può effettuare una selezione (ritaglio) dell'immagine;
22. L'utente può salvare (in formato fits, png o eps-gzip) l'immagine ottenuta fino a quel momento;
23. L'utente può ridimensionare il pannello delle opzioni.

Nel seguito, sarà evidenziato ogni aspetto che dimostra il rispetto di ogni requisito.

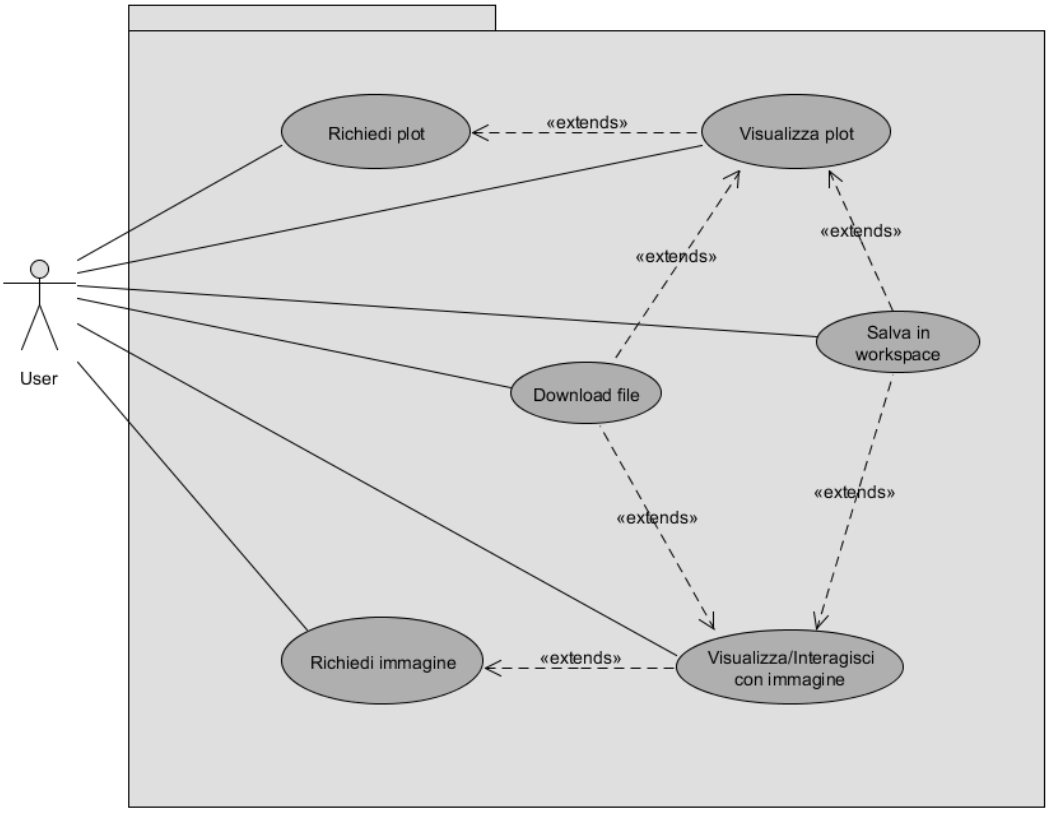


Figura 19 – Diagramma Use Case delle principali funzionalità del sistema

Diagramma dei casi d'uso che riguarda le funzionalità di plotting e visualizzazione di immagini. L'attore primario è l'utente che può compiere le azioni all'interno del package. Le azioni che presentano una freccia uscente con lo stereotipo <<extends>> sono da ritenersi facoltative e dipendenti dalle azioni che estendono. (Ad esempio non posso salvare un file se prima non lo visualizzo). (per i Cockburn vedere paragrafo 7.1)

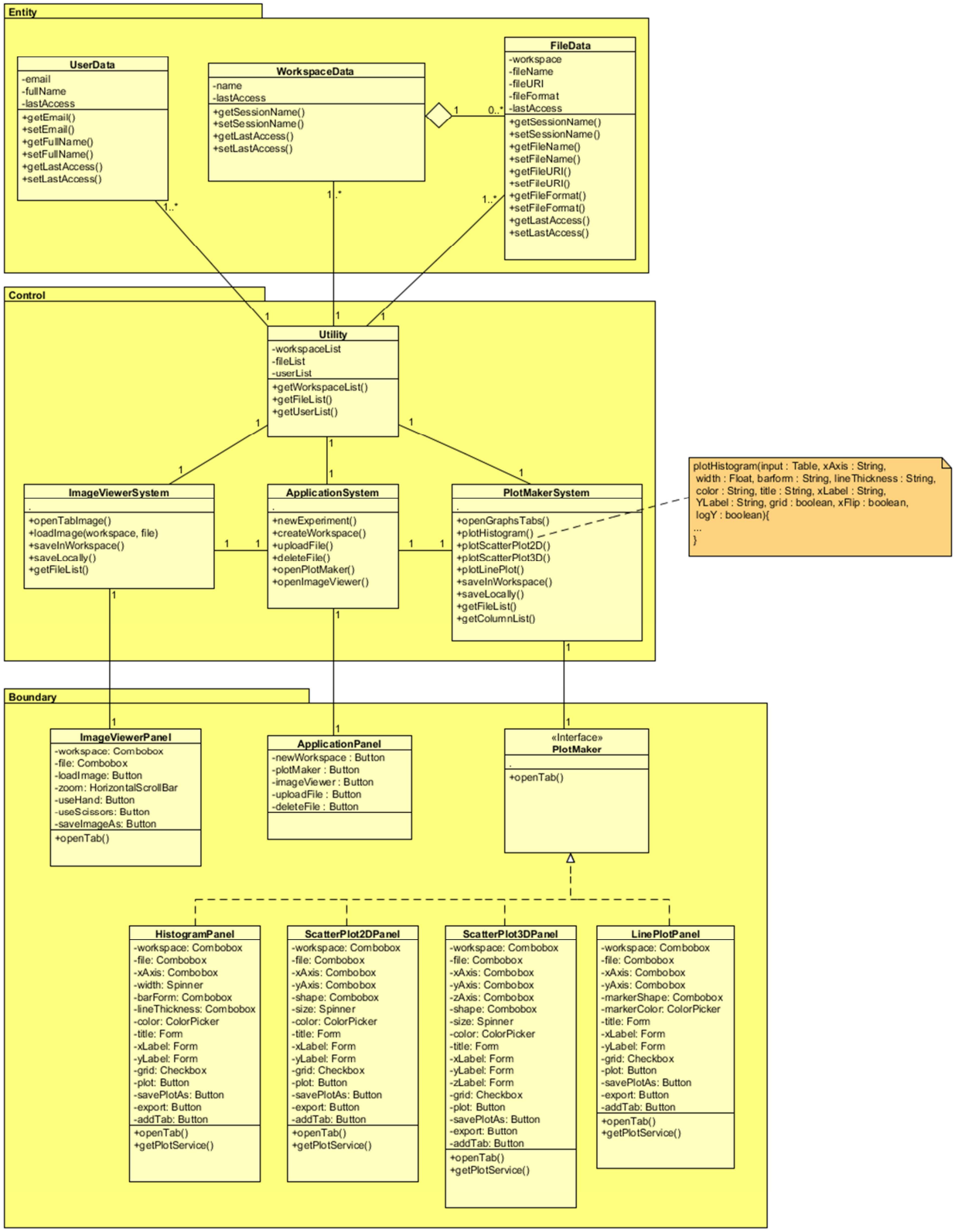


Figura 20 – Class Diagram che riguarda le nuove funzionalità

Come si può vedere dalla figura, il diagramma delle classi che rappresenta le nuove funzionalità è stato saggiamente diviso seguendo il modello "**entity-boundary-control**", chiamato anche Model-View-Controller (nel nostro caso il Controller diventa Presenter).

Tutte le classi sono state divise in tre macro categorie:

- Classe **Boundary**: descrive gli oggetti che rappresentano l'interfaccia tra un attore ed il sistema. Rappresenta una parte dello stato del sistema che è presentata all'utente in forma visiva. È chiamata anche View
- Classe **Control**: descrive gli oggetti che percepiscono gli eventi generati dall'utente e controllano l'esecuzione del processo di business. Rappresenta un supporto per le operazioni e le attività. È Chiamato anche Controller.
- Classe **Entity**: descrive gli oggetti che rappresentano la semantica delle entità nel dominio applicativo. Corrisponde ad una struttura dati nel database del sistema. È chiamato anche Model.

Più nel dettaglio, vediamo che, mentre **ImageViewerPanel** e la classe **ImageViewerSystem** comunicano direttamente, il sistema **PlotMakerSystem** non ha un'interazione diretta con le classi *Boundary* responsabili della creazione delle tab dei grafici, ma vi comunica tramite un'interfaccia astratta. Quest'ultima ha un solo metodo astratto *openTab()* che viene implementato opportunamente dalle singole classi sottostanti. In questo modo si astrae in parte la logica di apertura delle tab per la classe **PlotMakerSystem**, in quanto quest'ultima, non dovrà sapere quale tab sta creando, ma semplicemente richiamare in sequenza il metodo di creazione delle tab sugli oggetti di tipo *PlotMaker*. Inoltre, grazie a questo artifizio, risulta estremamente facile l'aggiunta di nuovi (futuri) tipi di grafici. L'integrazione di questi, infatti, non necessiterebbe di alcuna modifica sostanziale sui componenti software pre-esistenti, rispettando così la caratteristica architetturale Plug&Play di cui è dotata l'intera applicazione.
Si può notare anche che il sistema è dotato di una classe **Utility**, (implementata seguendo il design pattern *Singleton*), i cui metodi si interfacciano direttamente con gli oggetti **Entity**. In questo modo si riduce di molto l'accoppiamento tra le classi **Entity** e **Control**. I dettagli di queste ed altre interazioni fra le classi sopra rappresentate, sono descritte nei sequence diagram nel paragrafo 7.2.

4.1 Sistema GUI

Il sistema GUI ha subito consistenti modifiche per accogliere tutte le funzionalità descritte. Nella schermata principale sono stati introdotti due pulsanti, (per le due "macro-funzionalità") , che portano l'utente a visualizzare le ricche interfacce GUI che permettono di sfruttare i nuovi servizi. Le tab prodotte contengono una serie di elementi grafici quali pulsanti, scrollbar, combobox, slider, ecc. che sono stati saggiamente posizionati in seguito ad attente prove e analisi per fornire al sistema GUI un aspetto di applicazione orientata all'utente. Risulta facile, infatti, anche per un non addetto ai lavori, intuire immediatamente come sfruttare le nuove funzionalità introdotte.

Nonostante il notevole ventaglio di opzioni fornite dal tool grafico utilizzato (SmartGWT), le nuove schermate del sistema GUI hanno mantenuto, di fatto, tutte caratteristiche dettate dal sistema GUI preesistente, in termini di stile, flessibilità e contenuti. L'esempio lampante deriva dall'immersione delle schermate per la produzione dei grafici all'interno di tab specifiche, aperte opportunamente. L'obbiettivo, infatti, è quello di vedere le nuove parti della GUI come parte integrante dell'applicazione, e non come un'appendice della stessa.

Per la progettazione della GUI si è fatto uso di layout appositi che contengono tutti gli elementi per interazione con utente. Le tab permettono all'utente di conservare le informazioni relative a diversi tipi di grafici ottenuti, visualizzando contemporaneamente anche un'immagine.

4.2 Visualizzazione immagini

Uno dei requisiti è quello di poter visualizzare un'immagine (file formato **fits-image**, **gif**, **png** e **jpeg**). In questo caso il sistema dovrà riconoscere tutti i file che hanno al loro interno un immagine che sia possibile consultare.

Questi file in particolare possono derivare da:

- output proveniente da alcuni modelli di data mining;
- caricamento arbitrario dell'utente;
- salvataggio di grafici creati mediante l'opzione "Plot maker", in formato PNG;

Per questa funzionalità è stata dedicata una tab apposita in cui sarà data la possibilità agli utenti di effettuare zoom e panning delle immagini. La tab è costituita da due sezioni:

- sezione opzioni: include la scelta del workspace e del file displayable. Questa sezione potrà essere ridimensionata, in modo da permettere la visione dell'immagine a tutta tab;
- sezione immagine: la restante parte della tab in cui apparirà l'immagine. L'immagine viene visualizzata nella sua grandezza originale e l'utente può scorrerla in larghezza e lunghezza tramite apposite scrollbar.

Nella sezione opzioni l'utente selezionerà per prima cosa il workspace di riferimento e un immagine al suo interno, dopodiché l'utente potrà visualizzarla premendo l'apposito pulsante.

L'utente, quindi, avrà a disposizione in qualsiasi momento tutte le immagini presenti nel suo account sulla web app. Inoltre, in qualsiasi momento si può richiedere il download dell'immagine che si sta visualizzando rimanendo all'interno della tab dedicata.

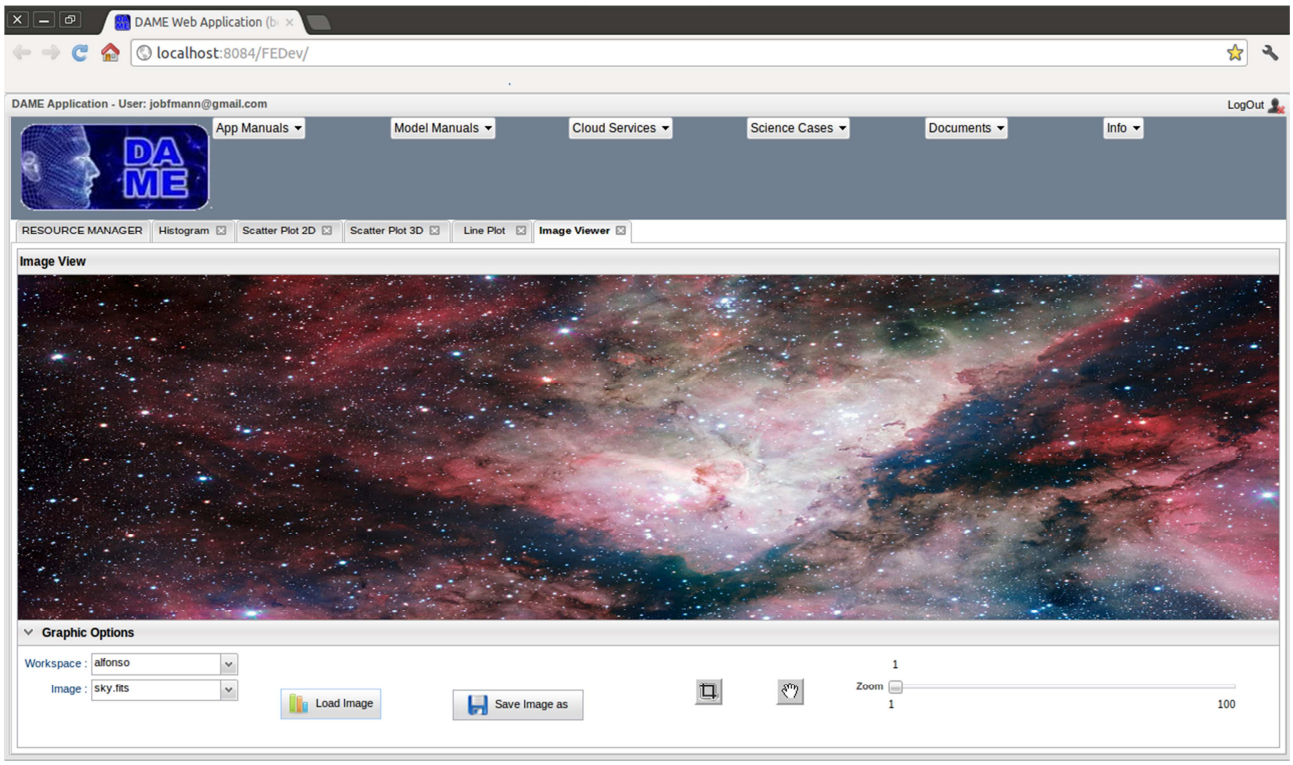


Figura 21 – Esempio di visualizzazione immagine

Vale la pena di accennare una soluzione utilizzata per i file di tipo Fits-image:

essendo questi ultimi dotati di una struttura che contiene informazioni di tipo astronomico (ma non di una tabella da cui poter estrarre un grafico) oltre all'immagine vera e propria, si è ricorsi ad un pacchetto di nome JFits per la loro visualizzazione. Il pacchetto **eap.fits** (reperibile all'indirizzo: http://heasarc.nasa.gov/docs/swift/sdc/software/java/) include una applet per visualizzare file FITS. Nel nostro caso è stata usata per visualizzare il contenuto dei file FITS che non sono provvisti di tabelle di dati rappresentabili mediante plot.

4.3 Creazione di grafici

Questa funzionalità viene attivata dal bottone "Plot Maker" della main window della GUI, la cui pressione causerà l'apertura automatica di una tab per ogni tipo di grafico previsto (4 nel caso specifico). Queste verranno aperte tutte sullo stesso livello della main tab "*RESOURCE MANAGER*" (figura 22).

L'utente potrà scegliere quale tab usare in base al tipo di grafico che vorrà realizzare. Ogni tab avrà due sezioni:

- sezione opzioni: include una sotto-tab incrementale con cui poter selezionare e sovrapporre nello stesso grafico più diagrammi, eventualmente relativi a file diversi, provenienti anche da workspace differenti. In ogni sotto-tab sarà possibile selezionare il workspace, il file e la/le colonna/e del file da rappresentare (a seconda del tipo di grafico). A fianco ogni sotto-tab vi dovranno essere il bottone "*Plot*", che comanderà la creazione del grafico, nonché una serie di ulteriori opzioni, tutte derivate dall'impiego di STILTS come tecnologia per la creazione dei grafici. Eventualmente l'utente potrà scegliere di scaricare direttamente in locale, oppure salvare in uno dei suoi workpace, il file del grafico prodotto, scegliendo la risoluzione dello stesso a proprio piacimento. Questa sezione potrà essere ridimensionata, in modo da permettere la visione del grafico a tutta tab;
- sezione display: include l'area della tab dedicata alla visualizzazione del grafico. La risoluzione di default (modificabile dall'utente) è 1250x250.

Dal punto di vista implementativo, la strategia della creazione delle tab seguirà lo stile e le modalità utilizzate per le altre tab all'interno della web app (per il requisito 0).

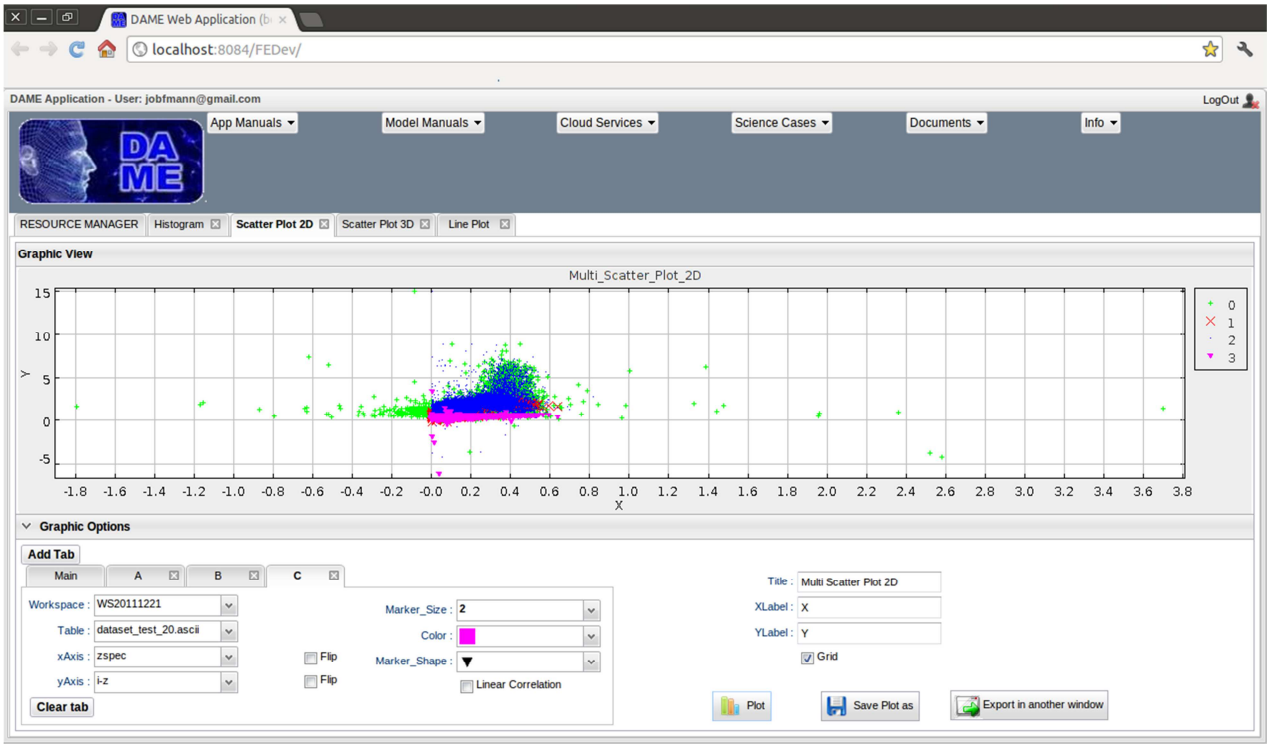


Figura 22 – Aspetto delle tab di creazione grafici

L'utente può scegliere il/i file da rappresentare mediante grafico sfogliando prima i vari workspace e poi i file. Per come è stata progettata l'applicazione, i workspace e i file visualizzati all'utente in questa fase, vengono filtrati in base, rispettivamente, al contenuto ed al tipo. In altri termini, un workspace viene visualizzato all'interno della combobox dedicata nella tab dei grafici solo se contiene almeno un file di tipo "editable", mentre un file viene visualizzato all'interno della combobox dedicata solo se è di tipo "editable". In questo modo non c'è possibilità di selezionare file non adeguati al plotting.

Dopo aver riempito tutti i campi obbligatori, l'utente potrà richiedere il plotting da lui settato. Il pulsante "Plot" risulta sempre abilitato, e la verifica del riempimento dei campi obbligatori avviene solo in seguito alla sua pressione. In caso di errore si visualizza un *warning* che mostra i campi che devono essere ancora riempiti.

Ogni richiesta di plotting richiede l'esecuzione della servlet associata (una per ogni tipo di grafico, fatta eccezione per il *Line Plot* che sfrutta la servlet dello *Scatter Plot 2D*). Le immagini prodotte non saranno salvate finché l'utente non lo richiede esplicitamente premendo l'apposito pulsante "*Save Plot As*". Quest'ultima operazione richiede una nuova creazione del grafico nel nuovo formato, scelto dall'utente. (vedere USE CASE #3, paragrafo 7.1).

Di seguito sono passati in rassegna i vari tipi di grafici con le relative caratteristiche di progettazione.

4.3.1 Istogramma

Una delle tab previste per la creazione di grafici permette all'utente di ottenere plot di tipo Histogram partendo sempre dai dati presenti sulla web app. Il tipo di grafico, conosciuto anche come diagramma a barre verticali (o colonne), prevede la scelta da parte dell'utente del valore da rappresentare sulle ascisse, mentre sulle ordinate saranno automaticamente rappresentate le densità di frequenza relative.

Il comando della libreria STILTS è "**plothist**", [11], e prevede che gli vengano passati, tra gli altri, i parametri di forma e dimensione delle barre.

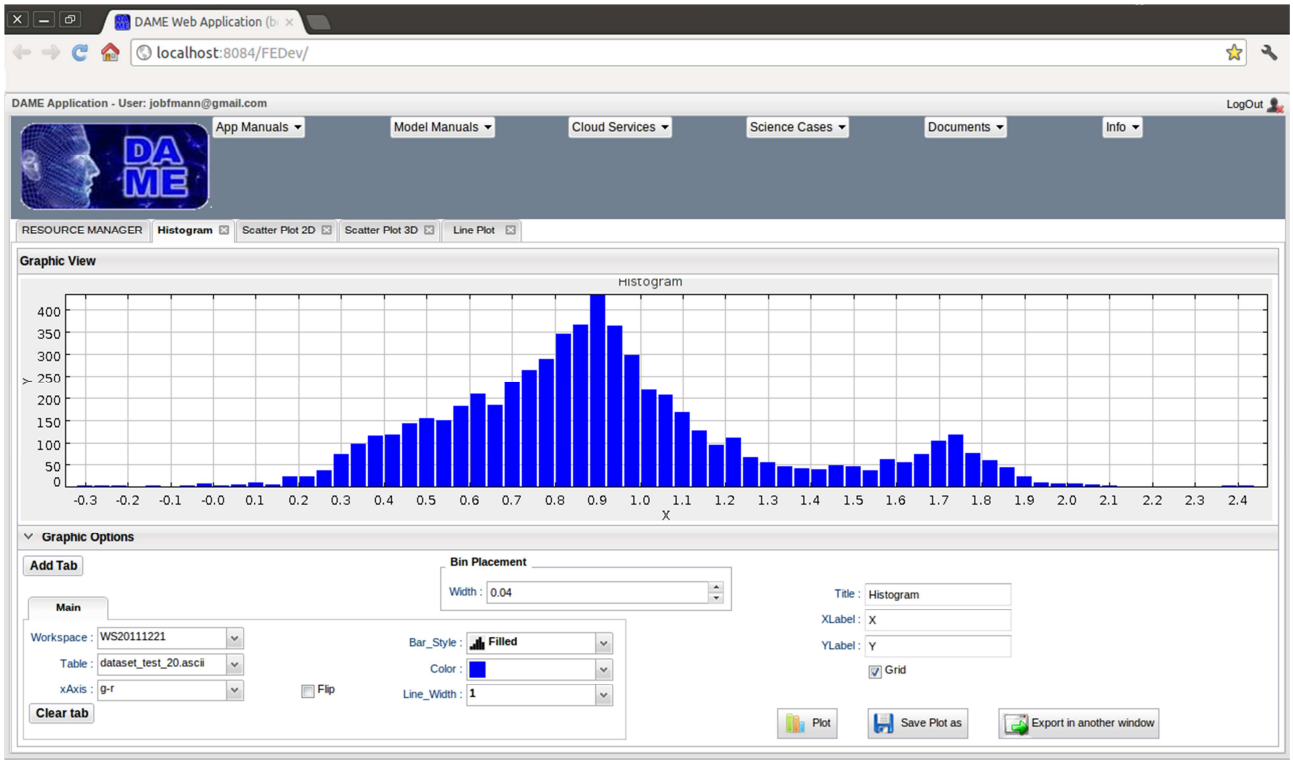


Figura 23 – Esempio di Histogram

Un particolare della progettazione, utilizzato solo per questo tipo di grafico, e derivante dalle caratteristiche del diagramma, è la scelta dello spessore dei bin delle ascisse, che avviene tramite uno "spinnerItem", [12]. In pratica si tratta della scelta della quantità di dati da rappresentare all'interno in un certo range.

Qui di seguito viene esaminata questa funzionalità molto di più nel dettaglio della implementazione, al fine di chiarire al lettore come interagiscono i vari componenti presi in considerazione e come sono stati sfruttati gli strumenti descritti in precedenza per questo scopo.

Per prima cosa vediamo come è composta la tab; come già detto in precedenza, la tab è divisa in due sezioni. L'oggetto che viene utilizzato per far ciò è il "SectionStack", [13], che permette di formare una pila di sezioni (SectionStackSection', [14]) che possono essere adeguatamente riposizionate. All'interno di esse saranno inseriti dei layout orizzontali o verticali a seconda dell'esigenza (HLayout, [15] e VLayout, [16]); i layout possono contenere a loro volta altri layout oppure elementi quali menù a tendina, pulsanti, e selettori. Tutti gli elementi grafici sono stati posizionati in maniera consona in seguito ad attente valutazioni.

Passiamo ora ad analizzare l'aspetto funzionale. Per far funzionare il meccanismo di selezione di un file è stato necessario effettuare delle chiamate asincrone al momento del riempimento a cascata delle varie combobox; le combobox sono opportunamente svuotate prima di ogni chiamata asincrona per evitare che queste siano riempite in maniera errata. Tutti gli altri elementi del pannello delle opzioni, non hanno bisogno di essere riempiti dinamicamente, ed inoltre sono dotati di un valore di default che permette la rapida creazione dei grafici.

In basso a destra sono presenti i pulsanti che azionano rispettivamente la richiesta un plot con i dati attualmente settati, il salvataggio dell'ultimo file ottenuto e la visualizzazione del grafico in una finestra a parte in formato maggiore. Il widget utilizzato per i pulsanti è **Button**, [17]; ognuno di essi ha un ascoltatore che gestisce l'evento click (costrutto: *onClick(ClickEvent event)*). La pressione del pulsante "Plot" fa partire dei controlli sui campi del pannello delle opzioni. In caso ci sia qualche dato mancante ai fini del plot, viene visualizzato un pop-up di warning con i campi necessari per richiedere un plot. Quando invece le informazioni sono complete, parte una chiamata asincrona verso la servlet di produzione grafico:

getPlotService().Plot(param1,param2,…..,callback);

Il parametro *callback* verrà analizzato in seguito.

La chiamata asincrona, mostrata nel dettaglio nel paragrafo 3.2.1, passa il controllo alla servlet opportuna, che estrapola tutti i dati passati all'interno dell'URL della HttpServletRequest (vedere paragrafo 3.3). A questo punto, viene creata una stringa ad hoc che rappresenta il comando shell che richiamerà l'esecuzione

dello script di STILTS. Un esempio di chiamata a questa libreria è:

/some/where/stilts plothist in=file.ascii format=ascii xdata=col1 ydata=col3 grid=true title=ScatterPlot2D

Dove:

- "in=": precede il file che si vuole dare in input (nel caso di file multipli si può indicare un numero ad ogni input, ad es.: in1=…. in2=…);
- "format=": precede il formato del file (vale il discorso fatto sopra per i file multipli);
- "xdata=": indica la colonna da rappresentare sulle ascisse (come sopra);
- "ydata=": indica la colonna da rappresentare sulle ordinate (come sopra);
- "grid=": indica la possibilità di avere o meno una griglia sullo sfondo;
- "title=": precede la stringa che verrà utilizzata come titolo.

Da qui possiamo evincere che alcuni parametri che formato il comando sono proprietà generiche che si vogliono ottenere nel grafico, (ad esempio **grid** e **title**). Questi parametri vengono inseriti nella stringa del comando per primi, poiché rappresentano la parte statica della chiamata, mentre gli altri parametri possono variare in numero. E' possibile, infatti, dare in pasto alle api STILTS più di un file in input con i relativi dati a contorno (formato, colore, ecc.). Essendo quindi il loro numero variabile, è necessario dotare il codice di un costrutto iterativo che non dipenda dal numero di dataset, ma che funzioni per ognuno di essi:

```
for(i=1; i<=multi; i++){
    String fileURI = request.getParameter("fileURI"+i);
    String format = request.getParameter("format"+i);
    …
    …

    comand = comand + " in"+i+"=" + fileURI + " xdata"+i+"=" + xAxis + " ifmt"+i+"=" + format+ "…

}
```

i parametri vengono man mano letti dall'URL della richiesta e vengono aggiunti alla stringa. La variabile "multi" rappresenta il numero di file selezionati, mentre la variabile "i", rappresenta il numero attuale del file che si sta aggiungendo. Una volta composto il comando per intero, è necessario effettuare una chiamata al sistema per eseguire il comando bash.

Process proc = Runtime.getRuntime().exec(comand);

proc.waitFor();

Il file generato, salvato all'interno di un file temporaneo all'interno del database, verrà visualizzato nella sezione apposita una volta che il controllo verrà ritornato alla GUI. A questo punto ci sono due possibilità:

- La chiamata asincrona è andata a buon fine, il grafico è stato prodotto ed è pronto per essere visualizzato.
- La chiamata asincrona è fallita, è stato riscontrato un problema nel server.

Ciò si trascrive nella seguente maniera:

```
final AsyncCallback <ErrorState> callback = new AsyncCallback<ErrorState>() {

    public void onSuccess(ErrorState result) {

        …

    }

    public void onFailure(Throwable caught) {

        SC.warn(result.getMessage());

    }
```

L'oggetto callback è un'istanza dell'interfaccia principale (AsyncCallback) che un chiamante deve implementare per ricevere una risposta da una RPC (vedere paragrafo 3.2.1).

Se la RPC è andata a buon fine, il metodo *onSuccess(Object)* è chiamato, altrimenti verrà chiamato *onFailure(Throwable).* In caso di successo viene visualizzata l'immagine prodotta, in caso contrario viene avvertito l'utente con un pop-up di warning contenente il messaggio d'errore riscontrato.

Questa è, in buona sostanza, l'implementazione adottata per questa funzionalità.

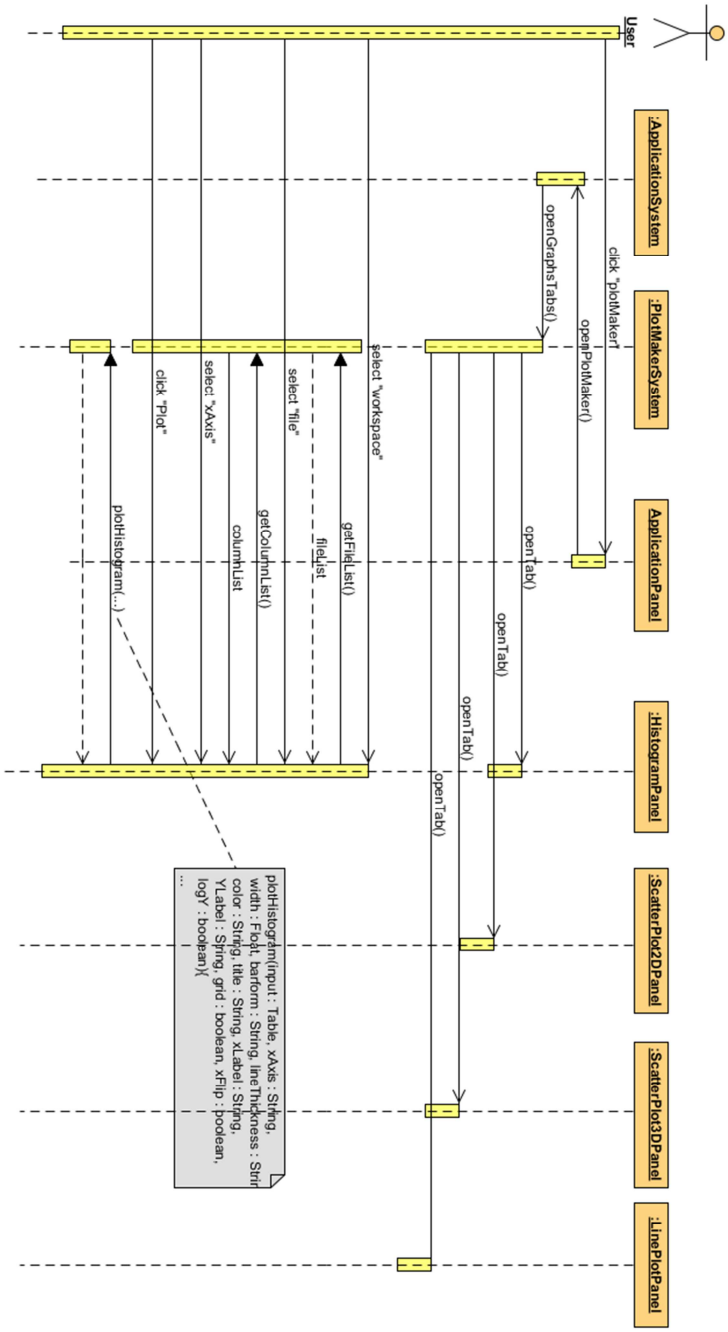


Figura 24 – Sequence Diagram relativo alla creazione del grafico Histogram

Per le altre funzionalità si è operato in maniera simile, ma, per questioni di prolissità e ridondanza, esse non sono state enucleate così nel dettaglio.

4.3.2 Scatter plot 2D

Questo tipo di grafico, chiamato anche grafico di dispersione, viene spesso usato quando una delle variabili è sotto controllo dello sperimentatore. Può essere utile per visualizzare il grado di correlazione (cioè di dipendenza lineare) tra le due variabili rappresentate sugli assi. E' molto utile anche quando vogliamo vedere quanto corrispondono due set di dati comparabili.

Il comando della libreria STILTS è "**plot2d**", [18], e prevede che gli vengano passati, tra gli altri, i parametri di forma e dimensione dei marker.

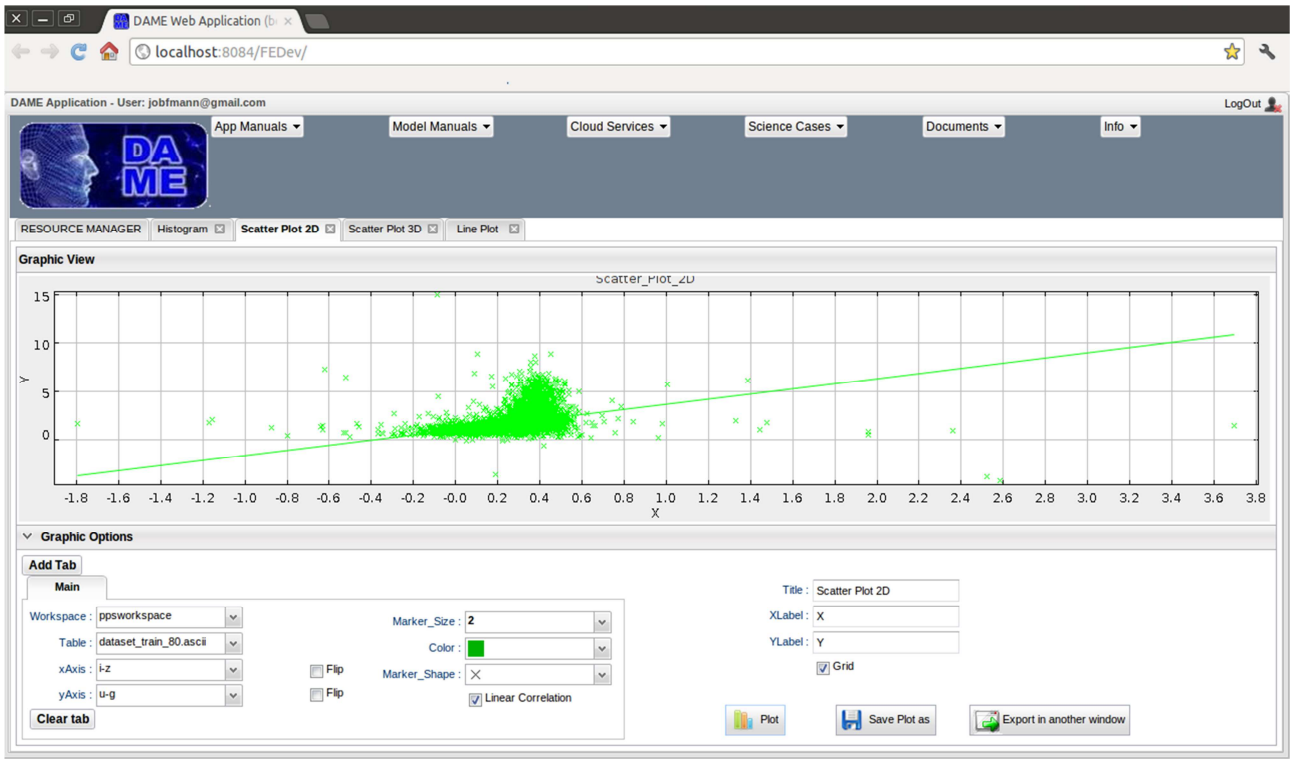


Figura 25 – Esempio di Scatter Plot 2D

Una caratteristica di questo tipo di plot è la possibilità di ottenere la rappresentazione della retta di regressione lineare basata sui punti visibili nel grafico.

4.3.3 Scatter plot 3D

Un'ulteriore grafico che si può ottenere dalla web-app è lo scatter plot 3D, che altro non è, che la versione tridimensionale dello scatter plot 2D appena visto. L'introduzione di un terzo asse permette di avere informazioni, a colpo d'occhio, su tre diverse caratteristiche dei dati analizzati.

Il comando della libreria STILTS è "**plot3d**", [19], e prevede che gli vengano passati, tra gli altri, i parametri di phi e theta, cioè degli angoli rispetto all'asse Z e rispetto alla prospettiva (verticale) dell'utente.

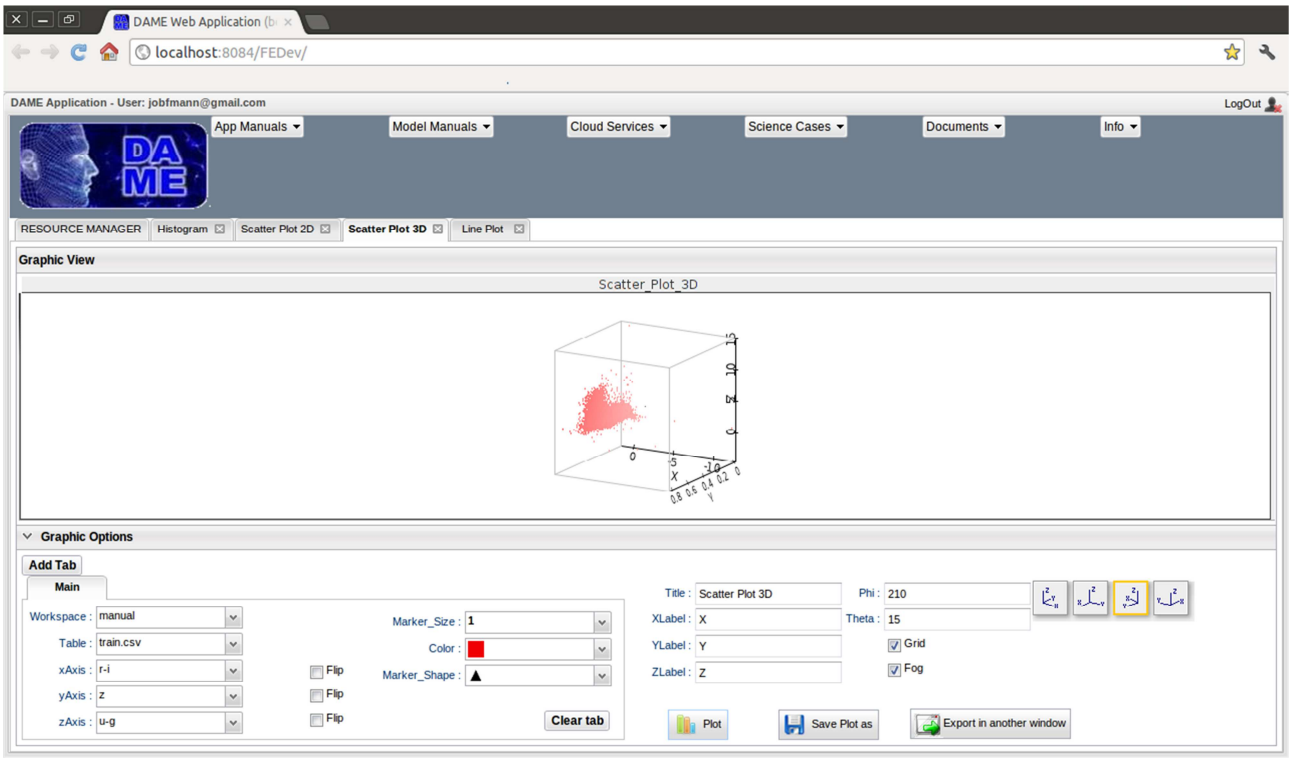


Figura 26 – Esempio di Scatter Plot 3D

Una caratteristica di questo tipo di plot è la possibilità di visualizzare, o meno, la "nebbia" cioè un'indicazione visuale della profondità dei punti rappresentati. In caso venga settata, gli oggetti tracciati in lontananza rispetto allo spettatore appariranno più sbiaditi.

4.3.4 Line plot

Il line plot, o diagramma bi-dimensionale a linee è ottenibile accedendo alla tab relativa nella sezione Plot Maker. Questo tipo di grafico è molto utile per rappresentare spettri.

I grafici di questo tipo sono ottenuti tramite lo stesso comando della libreria STILTS dello scatter plot 2D ("**plot2d**") impostando alcuni parametri ad hoc:

- *line=DotToDot* – Ogni punto è collegato al successivo in sequenza tramite una linea retta.
- *size=0* – La grandezza dei marker è 0. Questo perché non devono comparire in questo tipo di grafico.
- *xdata=index* – Sulle ascisse deve essere plottato il numero di righe del file in ingresso.

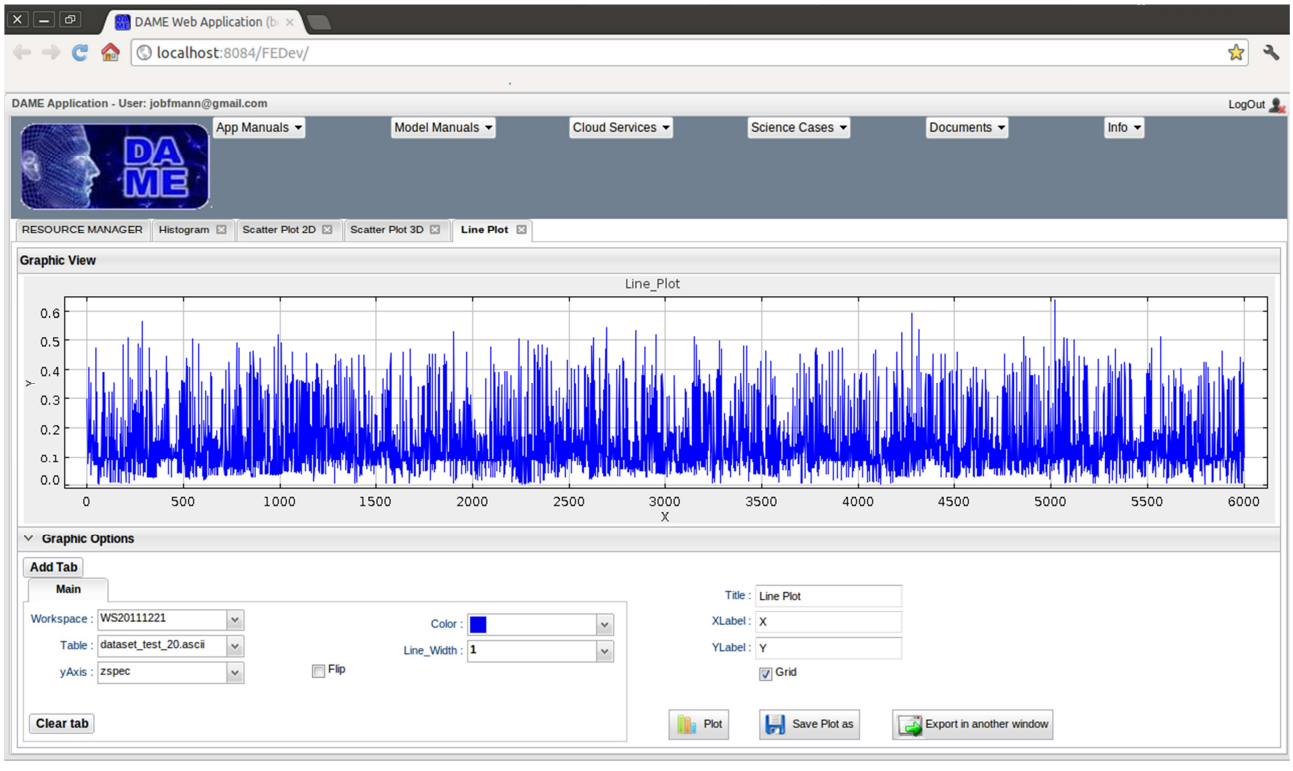


Figura 27 – Esempio di Line Plot

4.4 Analisi dei requisiti

Con riferimento alla lista di requisiti stilata all'inizio di questo capitolo, verranno ora descritte le soluzioni adottate per il loro soddisfacimento:

Requisiti generici:

Req 1, 2

L'utente può accedere alla sezione "Plot maker" della main window solo se esiste almeno un file "editable"

L'utente può accedere alla sezione "Image editor" della main window solo se esiste almeno un file supportato ma non "editable"

Questi due requisiti possono essere valutati insieme, in quanto entrambi richiedono che si possa selezionare un pulsante sulla GUI della main window sotto una certa condizione. I pulsanti verranno inseriti seguendo lo stile della GUI già esistente (per il req 0). I pulsanti saranno disattivati nel caso in cui non esistano file che soddisfino le condizioni. Questa è la scelta più intuitiva e dal punto di vista implementativo non richiede eccessive risorse.

Req 3

L'utente visualizza le tab relative ai grafici.

Questo requisito può essere soddisfatto in più modi, a seconda di come si decide di sistemare sulla GUI le diverse tab relative ai grafici. Il primo scenario prevedeva che le tab (nel nostro caso 5, una per ogni tipologia di grafico), venissero aperte tutte sullo stesso livello della main tab "RESOURCE MANAGER".

Nel secondo scenario si prevedeva un'unica tab generica dei grafici che si aprisse al livello della main tab, al cui interno vi si aprivano, ad un livello sottostante, le 5 tab relative ai vari tipi di grafici. Questo approccio, seppur fosse più giusto da un punto di vista logico/gerarchico, avrebbe introdotto lo svantaggio del consumo di spazio, in termini di pixel, a disposizione dell'utente per la consultazione i grafici. Considerando poi, che la web app è stata creata anche per essere utilizzata su dispositivi mobili, con schermi ovviamente

ridotti, la scelta è ricaduta immediatamente sul primo scenario.

Dal punto di vista implementativo, la strategia della creazione di tab ha seguito le modalità utilizzate per le tab già esistenti all'interno della web app (req 0).

Req 4

L'utente può scegliere un workspace ed un file (tra i tipi supportati per questa funzionalità)

All'interno della sezione plot maker, l'utente deve poter scegliere il file da rappresentare sfogliando prima i vari workspace e poi i file. Per come è stata progettata l'applicazione, i workspace e i file visualizzati all'utente in questa fase, vengono filtrati in base al contenuto e al tipo rispettivamente. In altri termini, un workspace viene visualizzato all'interno della combobox dedicata nella tab dei grafici solo se contiene almeno un file di tipo "editable", mentre un file viene visualizzato all'interno della combobox dedicata solo se è di tipo "editable".

Questo approccio disambigua definitivamente l'implementazione del requisito 4.

Req 5

L'utente può cambiare workspace e file in qualsiasi momento

Questo requisito potrebbe essere semplicemente implementato lasciando la possibilità di interagire con i menù a tendina relativi a workspace e file descritti nel precedente requisito.

Req 6, 7, 8, 9

L'utente può cambiare i parametri per il diagramma "Histogram", che sono: XAxis, Flip, Bin placement, Bar form, Line thickness, color, Title, XLabel, YLabel, Grid, Legend, XLimit, YLimit

Con questo requisito si intende dire che l'utente è libero di interagire con i vari oggetti di interfaccia che si trovano sulle tab dei grafici "Histogram", "Scatter Plot 2D", "Scatter Plot 3D", "Line Plot". Gli oggetti saranno diversi a seconda del tipo di grafico, a causa della natura stessa di questi. Gli oggetti

potranno essere sotto forma di:

- combobox
- button
- form
- checkbox
- spinner

L'associazione tra i parametri del grafico e gli oggetti grafici è descritta nel class diagram in figura 20 a pagina 61 . Questa analisi di requisito vale anche per i requisiti **7, 8** e **9.**

Req 10

L'utente può richiedere l'elaborazione dei grafici in qualsiasi momento premendo il pulsante "Plot", a patto che tutti i campi obbligatori siano stati settati

Questo requisito può essere soddisfatto intuitivamente in 2 modi:

1. nascondere il pulsante "Plot" fino a quando tutti i campi obbligatori per ottenere il grafico siano stati settati; ciò comporta un impiego notevole di risorse in termini di linee di codice, in quanto si dovrebbe aggiungere un ascoltatore per ogni campo obbligatorio, con all'interno un controllo che porterebbe eventualmente all'attivazione del pulsante "Plot".
2. lasciare il pulsante in questione sempre attivo, e verificare SOLO alla sua pressione il soddisfacimento del requisito, cioè il riempimento di tutti i campi obbligatori. In caso di errore si visualizzerebbe un messaggio di errore che mostri i campi che devono essere ancora riempiti.

Questo secondo approccio, seppur meno elegante, porta un notevole risparmio di righe di codice, ed è per questo che è stato impiegato.

Req 11

Il grafico ottenuto deve poter essere visualizzato

Questo requisito è soddisfatto semplicemente visualizzando, nell'area predisposta, l'immagine che viene prodotta in output dalla servlet tramite le chiamate alle librerie grafiche.

Req 12

L'utente può selezionare più file per ottenerne un grafico che contenga informazioni su tutto il dataset selezionato

Questo requisito si può ottenere predisponendo un area del pannello delle opzioni , sotto forma di insieme di tab, in modo tale che permetta di selezionare (con le dovute limitazioni), nuovi file da rappresentare. Una volta terminata la selezione, e richiesto il plot, verranno inviati alla servlet opportuna i metadati che permetteranno di generare il grafico contenente tutti i file selezionati.

Req 13

L'utente può richiedere il download del grafico ottenuto in 2 formati diversi: PNG o EPS-GZIP

Questo requisito è stato soddisfatto generando una finestra di dialogo, conseguente alla pressione del pulsante "*Save Plot as*" da parte dell'utente, dove si richiede il formato desiderato per il salvataggio. La generazione del grafico da parte delle librerie STILTS, infatti, prevede che venga specificato il parametro "*ofmt*", cioè il formato del file in output. E' possibile scegliere tra:

- **png**: image/png format
- **gif**: image/gif format
- **jpeg**: image/jpeg format
- **pdf**: application/pdf format
- **eps**: application/postscript format
- **eps-gzip**: application/postscript (gzip) format

da cui abbiamo scelto i due sopra citati per questioni di utilità. In particolare il formato EPS è un formato molto utile qualora un grafico debba essere inserito in una pubblicazione scientifica; la scelta di questo formato da parte dell'utente, a differenza di quanto accade per il formato **png**, richiede che il grafico venga prodotto ex-novo lato server.

Req 14

L'utente può richiedere il salvataggio del grafico all'interno dei workspace della web app

Questo requisito richiede che venga data la possibilità all'utente di scegliere la modalità di salvataggio del file immagine prodotto contenente il grafico. In questo modo, il file in questione andrebbe ad arricchire la collezione di dati a disposizione dell'utente nella sua sezione di lavoro. L'unica complicazione è rappresentata dal caso in cui il grafico venga generato unendo diversi dataset provenienti da workspace diversi (vedere soddisfacimento requisito 12). In questo caso si è deciso di lasciare la scelta all'utente di dove voglia che il file ottenuto in output venga salvato.

Req 15

L'utente può ridimensionare i pannelli delle opzioni in ognuna delle tab dei grafici

Questo requisito richiede di dotare le tab di visualizzazione dei grafici con sezioni mobili in grado di scomparire all'occorrenza. Le librerie di SmartGwt sono dotate di strumenti per far ciò:

- **SectionStack**: contenitore che gestisce un elenco di sezioni di widgets, ognuna con un'intestazione;
- **SectionStackSections**: descrittore di sezione usato da *SectionStack* per descrivere una sezione di oggetti che sono visualizzati o nascosti insieme, e la loro intestazione associata.

Settando adeguatamente i parametri di questi oggetti si può ottenere l'effetto desiderato. In particolare questo approccio è lo stesso usato già allo stesso scopo nella MAIN tab della GUI, attenendoci così al requisito 0.

Requisiti della sezione Image Viewer:

Req 16

L'utente visualizza la tab relativa alle immagini

Questo requisito è stato soddisfatto semplicemente adeguandosi allo stile della web app esistente. La tab relativa alle immagini altro non è che una nuova tab di lavoro, aperta al livello della MAIN TAB. Questa scelta è giustificata in maniera uguale a quella fatta per il requisito 3, con la differenza che in questo caso la tab è unica.

Req17

L'utente può scegliere un workspace ed un file (tra i tipi supportati per questa funzionalità)

Per soddisfare questo requisito è stato necessario introdurre un filtro sulla selezione del workspace e file nella tab di visualizzazione di immagini; a differenza di quanto fatto per la sezione "Plot Maker" (requisito 4), però, qui dovranno comparire solo file supportati ma non di tipo "editable", cioè i restanti. In particolare, ricordiamo che sono ammessi in questa sezione i seguenti formati: **fits-image**, **png**, **jpeg** e **gif**.

Req 18

L'utente può cambiare workspace e file in qualsiasi momento

L'utente è in grado di interagire liberamente con gli oggetti *combobox* nella tab in questione. I criteri di selezione sono sicuri e non generano errori di consistenza.

Req 19

L'immagine relativa al file selezionato deve essere visualizzata

L'immagine, che si trova sul framework, deve essere visualizzata nella sezione apposita della tab.
Per fare ciò si è deciso di usare indirizzi URI temporanei, criptati in base al session ID della sessione corrente. In questo modo, viene negata la possibilità ad un utente generico di accedere liberamente ai file immagine, anche degli altri utenti, che si trovano sul server.

Req 20

L'utente può effettuare lo zoom sull'immagine visualizzata

Questo requisito è stato soddisfatto introducendo uno *SpinnerItem* all'interno della tab di Image Viewer. Grazie a questo strumento, l'utente può effettuare lo zoom in avanti ed indietro sull'immagine. Questo è ottenuto aumentando la grandezza dell'immagine visualizzata nell'apposita sezione, e spostandola progressivamente in alto e a sinistra. In questo modo sarà simulato l'effetto dello zoom al centro.

Req 21

L'utente può effettuare una selezione (ritaglio) dell'immagine

Questo requisito è stato soddisfatto introducendo uno strumento di selezione, e quindi ritaglio, all'interno del pannello delle opzioni nella tab di visualizzazione immagine. Selezionando l'apposito pulsante, infatti, è possibile utilizzare il mouse per selezionare la sezione dell'immagine interessata. Il ritaglio costituirà a tutti gli effetti una nuova immagine.

Req 22

L'utente può salvare (in formato fits, png o eps-gzip) l'immagine ottenuta fino a quel momento

Il requisito è stato soddisfatto convertendo, ove necessario, il file selezionato dall'utente nel formato desiderato. È prevista la scelta di scaricare il file in locale, oppure salvarlo in uno dei workspace dell'utente.

Req 23

L'utente può ridimensionare il pannello delle opzioni

Questo requisito è stato soddisfatto alla stessa maniera del requisito 15.

5. Test e verifica

Questo capitolo è dedicato alla fase di test e verifica delle funzionalità descritte, sulla base dei requisiti già descritti nel capitolo precedente.

Dato che l'ambiente di sviluppo riguarda un'applicazione in ambito astrofisico, i risultati dei test sono stati organizzati prendendo spunto da un problema reale. Nel seguito, oltre ad una introduzione al problema specifico, saranno descritte tutte le fasi di verifica con relativi esempi.

5.1 Il problema astrofisico: Classificazione di Ammassi Globulari

Nella nostra e nelle altre galassie sono presenti ammassi stellari che trovano la loro origine in episodi collettivi di formazione stellare. Gli ammassi globulari (Castellani 1985), sono un insieme sferoidale di stelle che orbita come un satellite intorno al centro di una galassia e sono sorretti al loro interno da una forte gravità; ciò conferisce loro il tipico aspetto sferico e trattiene, al loro centro, una densità di stelle relativamente molto elevata. Il nome di questa categoria di oggetti deriva dal latino globus, che significa "globo" o "sfera"; talvolta ci si riferisce a tali oggetti semplicemente con l'appellativo di "ammasso globulare".

Figura 28 - Esempio di ammasso globulare

Gli ammassi globulari sono piuttosto numerosi: se ne conoscono attualmente poco più di un centinaio intorno alla Via Lattea, con forse altri 10-20 da scoprire, essendo nascosti all'osservazione da Terra dalle polveri interstellari che oscurano la vista in direzione del centro galattico; si ipotizza che galassie più grandi possano averne un numero nettamente superiore (la Galassia di Andromeda potrebbe averne fino a 500), mentre alcune galassie ellittiche giganti (come M87) ne contano fino a 10.000.

Gli ammassi globulari sono composti generalmente da centinaia di migliaia di stelle vecchie a bassa "metallicità"[10] ; alcuni sono visibili a occhio nudo e si presentano come delle piccole macchie chiare e dai contorni sfumati. I più luminosi sono Omega Centauri e 47 Tucanae, visibili solo dall'emisfero australe, e l'Ammasso Globulare di Ercole visibile dall'emisfero boreale. Gli studiosi non sono sicuri che le stelle si siano formate in una singola generazione, [20], o se si estendano per diverse generazioni in periodi di varie centinaia di milioni di anni. Questo periodo di formazione stellare è tuttavia relativamente breve se paragonato all'età di molti ammassi. Le osservazioni mostrano che la formazione delle stelle degli ammassi globulari avviene innanzitutto in regioni dove questo fenomeno è molto elevato e dove il mezzo interstellare ha una densità maggiore rispetto alle regioni normali di formazione stellare. La formazione dei suddetti ammassi globulari avviene principalmente nelle regioni dette starburst[11] .

Nel 1771 l'astronomo francese Charles Messier creò un catalogo, [21], in cui gli ammassi erano indicati dalla lettera M seguita dal numero del catalogo. Una più completa classificazione degli ammassi delle Galassie è quella fornita nel 1888 dal New General Catalogue, [22], di galassie, ammassi e nebulose. In essa sono anche riportati numerosi ammassi appartenenti alle due vicine galassie irregolari note come Piccola e Grande Nube di Magellano. Per fare riferimento agli oggetti di questo catalogo si utilizza la sigla NGC seguita dal numero di catalogo. A seguito delle numerose identificazioni, molti oggetti celesti, e in particolare molti ammassi stellari, hanno una molteplicità di nomi variamente e alternativamente utilizzata nella letteratura scientifica.

[10] Per **metallicità** s'intende l'abbondanza di elementi più pesanti dell'idrogeno e dell'elio presente negli oggetti celesti, ovvero la chimica superficiale delle stelle.
[11] Le **starburst** sono regioni in cui vi è un alto tasso di formazione stellare.

5.2 Descrizione dei dati per il problema reale

Il problema astrofisico attraverso cui procederemo per i test delle funzionalità implementate e relativa verifica funzionale, riguarda un set di osservazioni astronomiche di una regione di cielo, caratterizzata dalla presenza di un ammasso di oggetti intorno ad una galassia centrale, denominata NGC1399. Tale regione, mostrata in Fig.30, è stata osservata mediante utilizzo di diversi strumenti osservativi, sia da Terra (telescopio Keck), sia dallo Spazio (Hubble Space Telescope e Spitzer), in diverse bande dello spettro elettromagnetico. Attraverso metodi di analisi fisica è possibile cross-correlare le diverse immagini spettrali, al fine di ricavare una varietà di informazioni, sia ottiche che strutturali, degli oggetti presenti nella regione di cielo.

L'obiettivo primario del problema è predisporre un campione di oggetti, con relativi parametri osservati, con cui compiere una serie di esperimenti di data mining, in modo da riuscire a classificare correttamente la natura di ciascun oggetto (se sia o meno un candidato ammasso globulare).

Con riferimento ai vari tools grafici implementati, risulta di estrema utilità per l'utente poter visualizzare immagini della regione osservata, correlare graficamente tra loro diversi parametri di ogni oggetto, analizzare la loro distribuzione, in modo da poter scegliere con cognizione eventuali sottoinsiemi di parametri, significativi in termini di input a successivi esperimenti di data mining.

Dunque in ogni problema reale, che riguardi l'esplorazione e l'analisi di dati (immagini o tabelle), i tools realizzati in questo lavoro, rappresentano la base fondamentale per il pre-processing e la preparazione dei dati.

Riguardo ai tools di visualizzazione immagini, nelle figure 29, 30 e 31 è mostrata la regione di cielo osservata, sotto forma di diversi punti di vista. L'immagine di tipo jpg rappresenta una schematizzazione della regione di cielo osservata, con il dettaglio del mosaico di osservazione (Fig.29); l'immagine di tipo png è una vista ottenuta mediante sovrapposizione di tutte le bande di osservazione (Fig.30); infine l'immagine di tipo FITS rappresenta la stessa regione di cielo in cui siano stati isolati astrometricamente tutti gli oggetti in essa presenti (Fig.31).

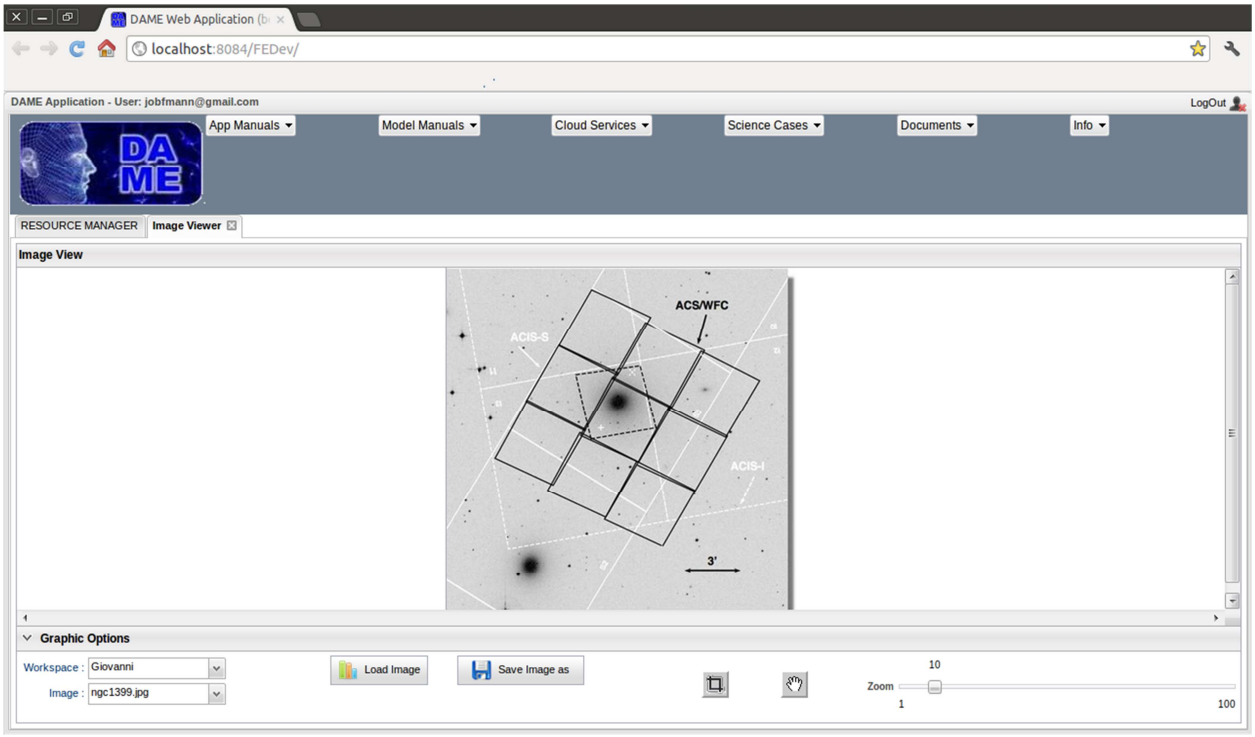


Figura 29 – Campo di vista coperto dal mosaico 3x3 osservato da Hubble Space Telescope. Il campo centrale, con differente orientamento, mostra la regione osservata dagli altri strumenti.

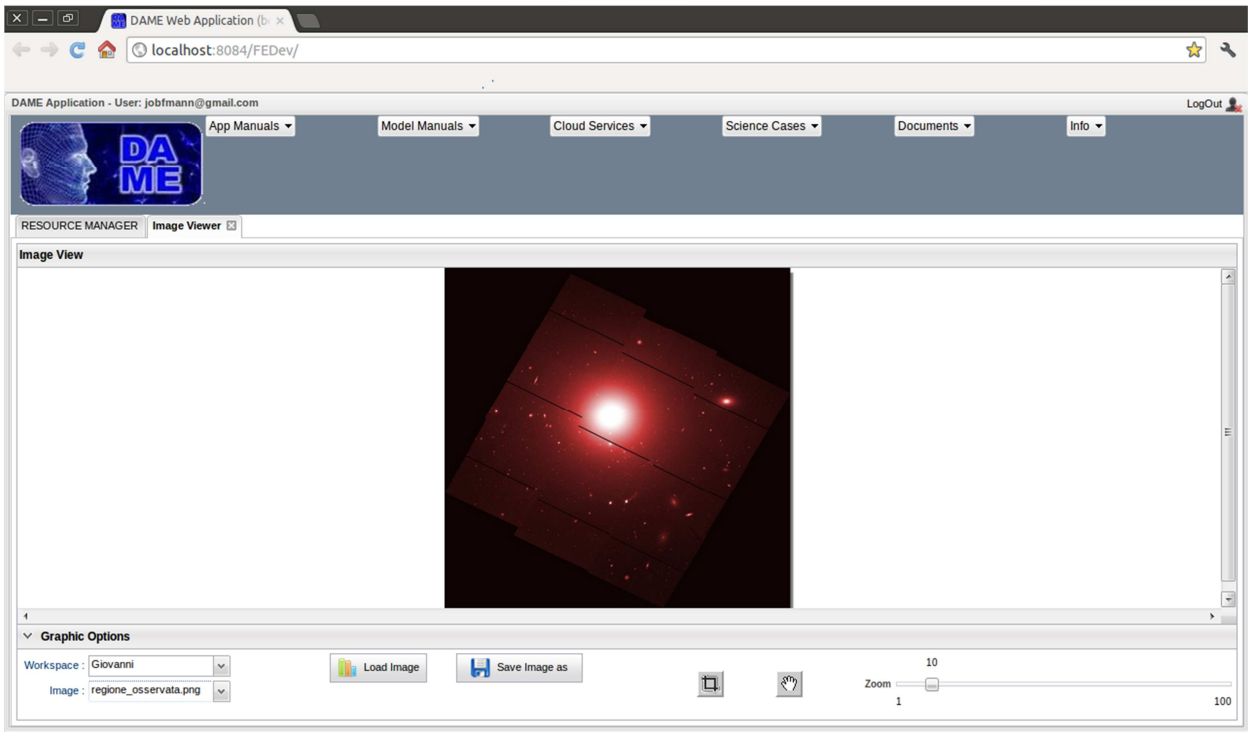


Figura 30 – Analoga immagine della figura 29, in cui sono stati applicati i filtri di correlazione tra le diverse bande di osservazione.

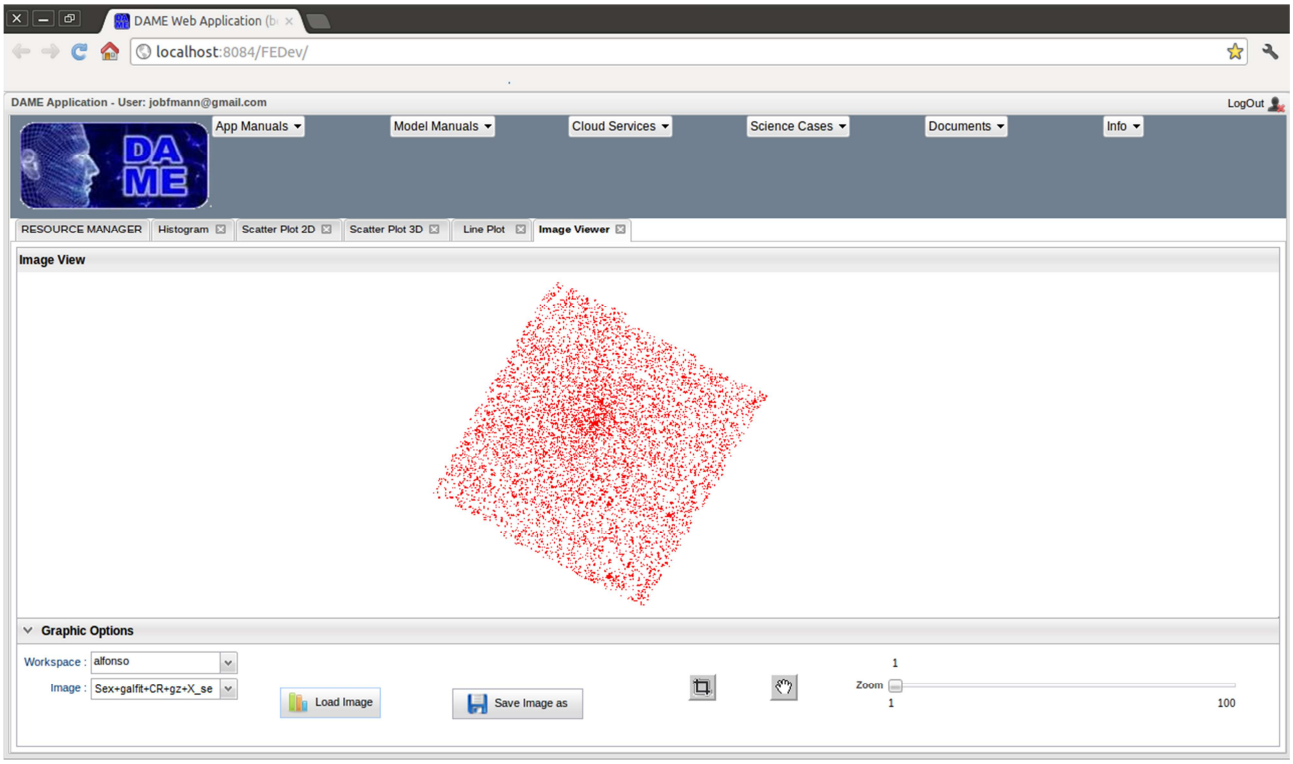


Figura 31– Analoga immagine mostrata nelle figure 29 e 30, con dettaglio sugli oggetti estrapolati attraverso calcoli astrometrici.

In queste immagini sono evidenziate le opzioni di visualizzazione progettate e implementate, seguendo i requisiti funzionali. In particolare, con riferimento all'elenco di requisiti riportato nel capitolo 4, il tool di visualizzazione di immagini include:

- visualizzazione della tab apposita (requisito nr. 16);
- scelta Workspace e file (requisito nr. 17, 18);
- visualizzazione dell'immagine selezionata (requisito nr. 19);
- zoom slider (requisito nr. 20);
- possibilità di effettuare il ritaglio dell'immagine (requisito nr.21);
- possibilità di salvataggio del file ottenuto fino a quel momento (requisito nr. 22);
- pannello delle opzioni ridimensionabile (requisito nr. 23).

Per quanto riguarda i tools grafici relativi alle tabelle di dati, la figura 32 mostra una porzione del dataset estrapolato dalle immagini precedentemente descritte, in cui ogni riga rappresenta un oggetto contenuto nell'immagine.

Table Browser for 1: opt_and_struct_full.txt

	MAG_ISO	MAG_APER1	MAG_APER2	MAG_APER3	KRON_RADIUS	ELLIPTICITY	FWHM_IMAGE	mu0	calr_c	calr_h	calr_t	target
1	24,4753	26,7468	24,3789	0,0205	3,72	0,067	4,12	16,25	-0,1139	1,822	51,29	0
2	24,2342	26,5263	24,1632	0,0196	3,5	0,027	4,01	16,61	0,1321	1,856	35,38	0
3	23,1554	25,5964	23,1654	0,016	3,5	0,032	4,09	14,47	-0,3295	2,638	129,2	1
4	22,6316	25,3519	22,6808	0,0151	3,5	0,039	4,69	16,33	0,8065	5,002	80,45	1
5	22,4708	24,4951	22,4699	0,0216	3,5	0,066	3,45	12,81	-0,3912	-7,425	5,66	0
6	23,9033	27,5896	23,9168	0,0255	4,49	0,272	9,63	19,99	8,397	14,79	88,5	1
7	24,1972	26,4219	24,0978	0,0192	3,7	0,079	4,04	15,72	-0,1447	1,514	44,77	0
8	20,2423	22,1866	20,2963	0,017	3,5	0,03	3,23	6,68	-0,6999	-0,1492	1,899	0
9	23,5134	26,0983	23,511	0,0167	3,76	0,05	4,55	16,6	0,3777	4,75	105,8	1
10	22,5967	25,1807	22,6182	0,0147	3,5	0,021	4,45	15,98	0,6535	3,615	52,13	1
11	21,8602	23,8834	21,8737	0,0171	3,5	0,077	3,25	12,2	-0,2371	-3,39	-0,2887	0
12	24,4292	26,6224	24,3097	0,0201	4,33	0,116	3,96	16,22	-0,1139	2,635	81,01	0
13	20,3466	22,2583	20,3962	0,0161	3,5	0,047	2,89	8,9	-0,5763	0,07634	-48,54	1
14	23,8686	26,4629	23,8326	0,0179	3,92	0,144	4,53	17,16	0,5617	4,345	76,83	0
15	24,1944	26,5159	24,0998	0,0188	4,26	0,109	4,2	15,52	-0,2987	5,187	322,2	0
16	23,3385	25,6679	23,3182	0,0166	3,5	0,042	4,04	15,12	-0,1447	1,514	44,77	1
17	24,8669	27,3614	24,7216	0,0247	4,49	0,075	4,79	18,6	1,204	5,504	73,48	0
18	23,6528	26,1355	23,609	0,0177	3,68	0,091	4,36	16,69	0,4391	3,505	60,31	1
19	23,1038	26,0645	23,1542	0,016	3,95	0,056	5,66	17,97	2,176	7,619	84,62	1
20	23,435	25,2643	23,3951	0,0162	3,5	0,129	3,05	13,35	-0,3604	0,3709	-34,75	0
21	25,1723	27,781	24,9976	0,0308	4,6	0,061	5,39	19,58	2,146	6,942	71,81	0
22	24,7851	27,2661	24,6492	0,0251	4,22	0,345	5,33	17,54	1,6	6,105	71,81	0
23	22,3268	24,3309	22,3327	0,0345	3,5	0,06	3,27	14,63	0,5004	0,5712	5,914	0
24	21,8194	24,2154	21,8633	0,0139	3,5	0,063	3,97	13,76	-0,08314	2,483	70,97	1
25	19,9421	23,1152	20,3493	0,0136	3,5	0,17	4,57	14,33	1,234	20,81	755,	0

Figura 32 – Sottoinsieme del dataset relativo al campione di oggetti correttamente classificati, presenti nella regione di cielo mostrata nell'immagine della figura 30. Ogni riga rappresenta un oggetto, caratterizzato da 11 parametri più la 12ª colonna (classe di attribuzione)

Per ogni oggetto vi sono elencati i relativi parametri osservati (prime 11 colonne) e relativa classe di attribuzione (colonna nr. 12). Le prime 11 colonne contengono in particolare le seguenti tipologie di parametri:

- **magnitudine isofotale** (1ª caratteristica);
- **3 magnitudini di apertura** (caratteristiche 2-4) ottenute tramite aperture circolari di raggio 2, 6 e 20 arcsec, rispettivamente;
- **Kron radius, ellipticity** e la **FWHM** dell'immagine (caratteristiche 5-7);
- **4 parametri strutturali** (caratteristiche 8-11) che sono, rispettivamente, **central surface brightness**, **core radius**, **effective radius** e il **tidal radius**;
- un valore di **target** utilizzato solo per il training set: assume valore 1 se risulta essere un AG[12], 0 altrimenti;

di cui i primi 7 sono parametri ottici e i restanti 4 sono parametri strutturali.

[12] Ammasso Globulare

Nella fase di pre-processing di un qualunque esperimento di data mining con tale dataset, risulta molto utile dotare l'utente di strumenti grafici per l'analisi delle varie correlazioni tra tutti i parametri, al fine di individuare il migliore raggruppamento che permetta di eseguire un esperimento di data mining su questi dati.

A tal fine, sono stati implementati i seguenti tools.

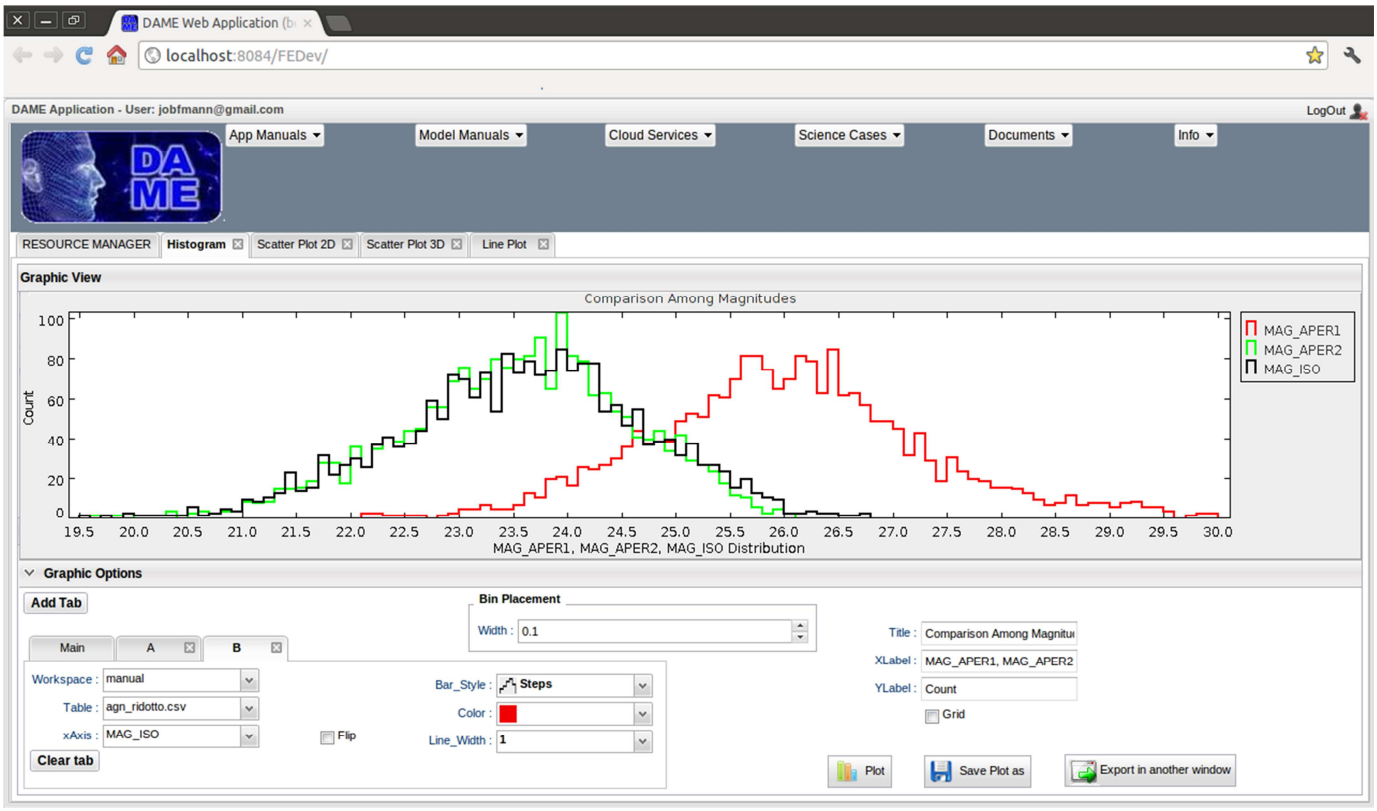


Figura 33 – Distribuzione di luminosità del campione di oggetti correttamente classificati. Il confronto è eseguito tra 2 magnitudini di apertura (rosso: MAG_APER1, verde: MAG_APER2) e la magnitudine isofotale (nero: MAG_ISO)

In Fig. 33 è mostrato un istogramma multiplo, in cui è possibile confrontare le distribuzioni degli oggetti in funzione di diversi parametri (ad esempio correlando diverse magnitudini fra loro).

In questo tool si nota come sia possibile:

- visualizzare le tab relative ai grafici, che verifica il requisito 3;
- scegliere in qualsiasi momento un workspace ed un file (combobox ***Workspace*** e ***Table*** in figura), che verifica il requisito 4 e 5;

- cambiare i parametri per il diagramma “Histogram”, che sono: **XAxis**, **Flip**, **Width**, **Bar Style**, **Colour**, **Line Width**, **Title**, **XLabel**, **YLabel**, **Grid**. Il che soddisfa il requisito 6;
- richiedere l’elaborazione dei grafici in qualsiasi momento premendo il pulsante “Plot”, che soddisfa il requisito 10;
- visualizzare Il grafico di cui si è richiesta l’elaborazione (pannello ***Graphic View*** in figura), che soddisfa il requisito 11;
- aggiungere nuove colonne per ottenerne un grafico che contenga informazioni su tutto il dataset personale selezionato (pulsante ***Add Tab*** in figura), che soddisfa il requisito 12;
- richiedere il download del grafico ottenuto in 2 formati diversi: **PNG** o **EPS-GZIP** (pulsante ***Save Plot as***), che soddisfa il requisito 13;

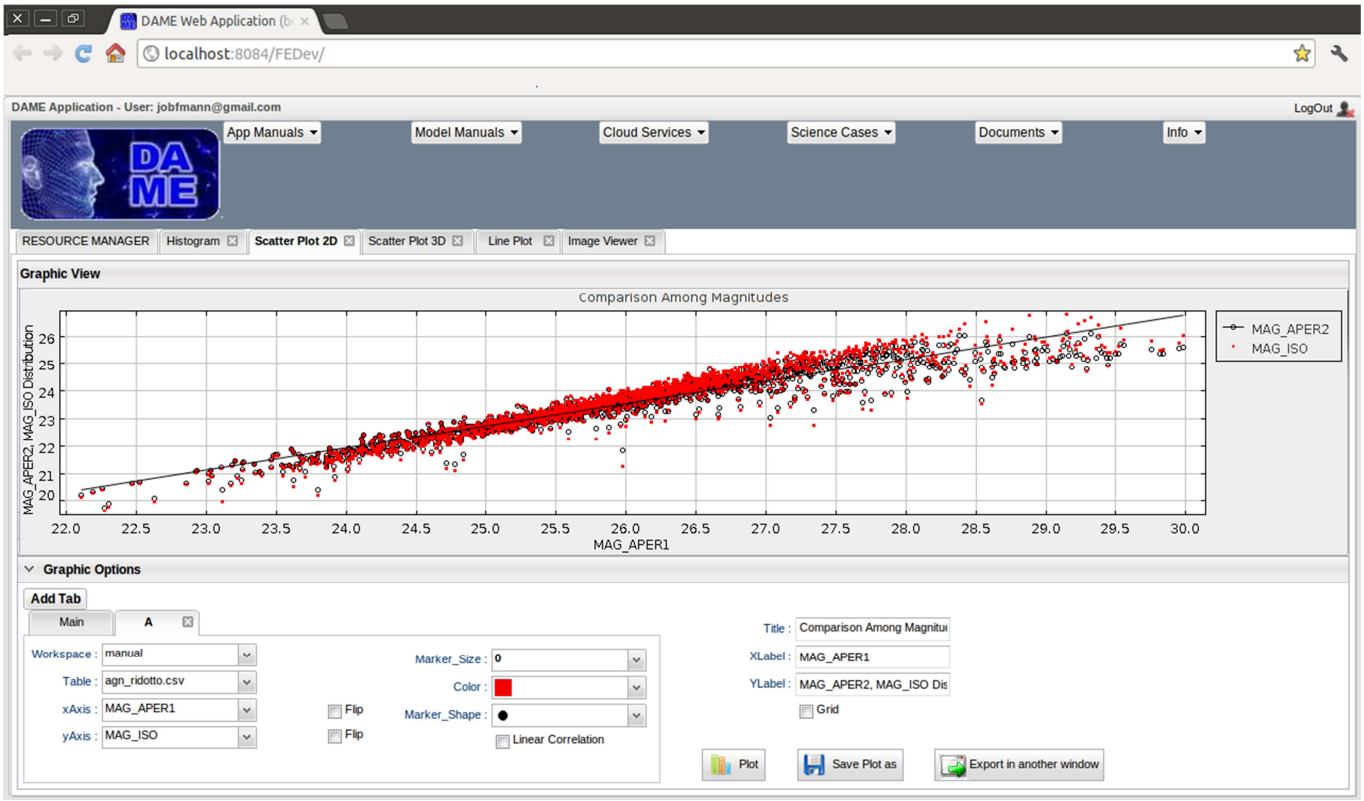


Figura 34 – Confronto tra magnitudini di apertura e isofotale per il campione di oggetti correttamente classificati

In Fig. 34 si può vedere un diagramma di tipo Scatter Plot bidimensionale, in cui è possibile confrontare le distribuzioni delle magnitudini MAG_APER2 e MAG_ISO in funzione della magnitudine MAG_APER1.

La diversa colorazione e forma dei marker utilizzati evidenzia la corrispondenza fra i valori assunti dalle magnitudini messe a confronto.

In questo tool si nota come sia possibile cambiare i parametri per il diagramma "Scatter Plot 2D", che sono: **XAxis, YAxis, XFlip, YFlip, Marker size, Marker shape, Colour, Title, XLabel, YLabel, Grid, Linear correlation.** Il che soddisfa il requisito 7;

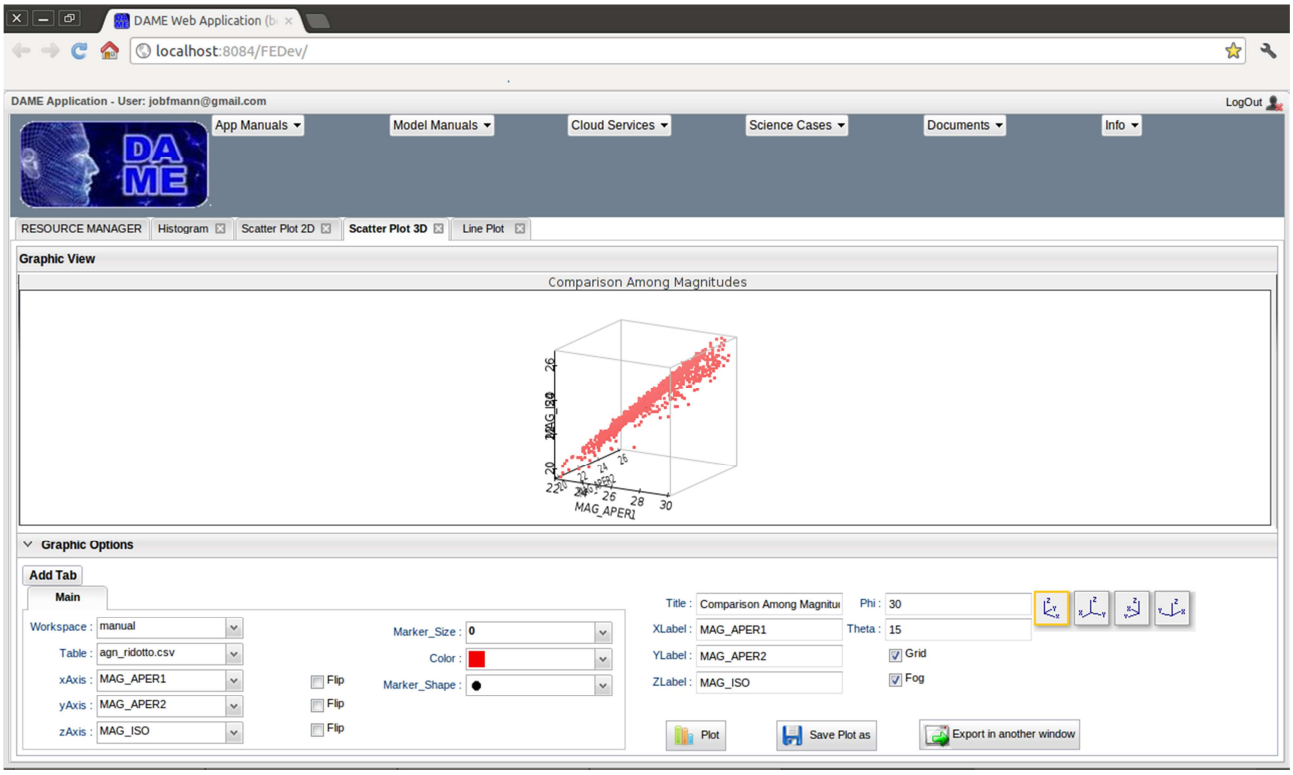


Figura 35 – Confronto 3D tra magnitudini di apertura e isofotale per il campione di oggetti correttamente classificati.

In Fig. 35 si può vedere un diagramma di tipo Scatter Plot 3D, in cui sono state plottate sui tre assi cartesiani le due magnitudini di apertura e quella isofotale, rispettivamente. Nelle figure 36, 37 e 38 sono evidenziate le correlazioni fra i valori plottati in full screen e con angolazioni diverse.

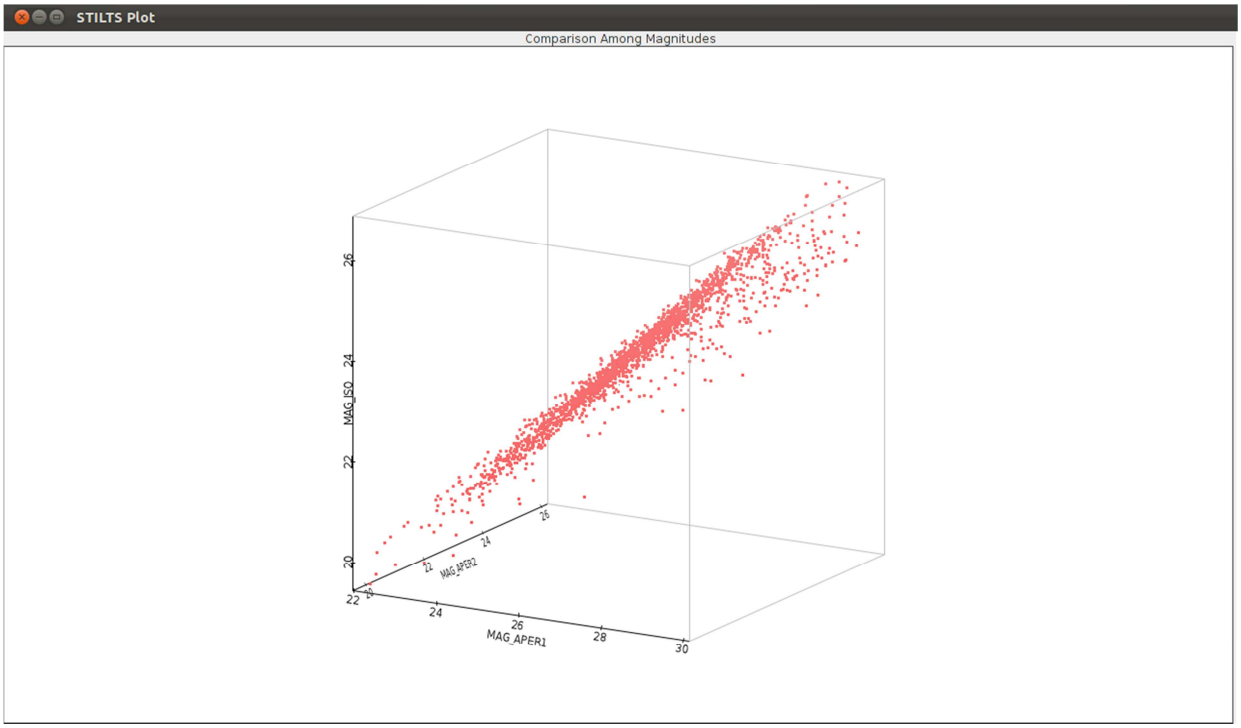


Figura 36 – Confronto 3D tra magnitudini di apertura e isofotale per il campione di oggetti correttamente classificati. Vista per il confronto diretto fra MAG_APER1 e MAG_ISO.

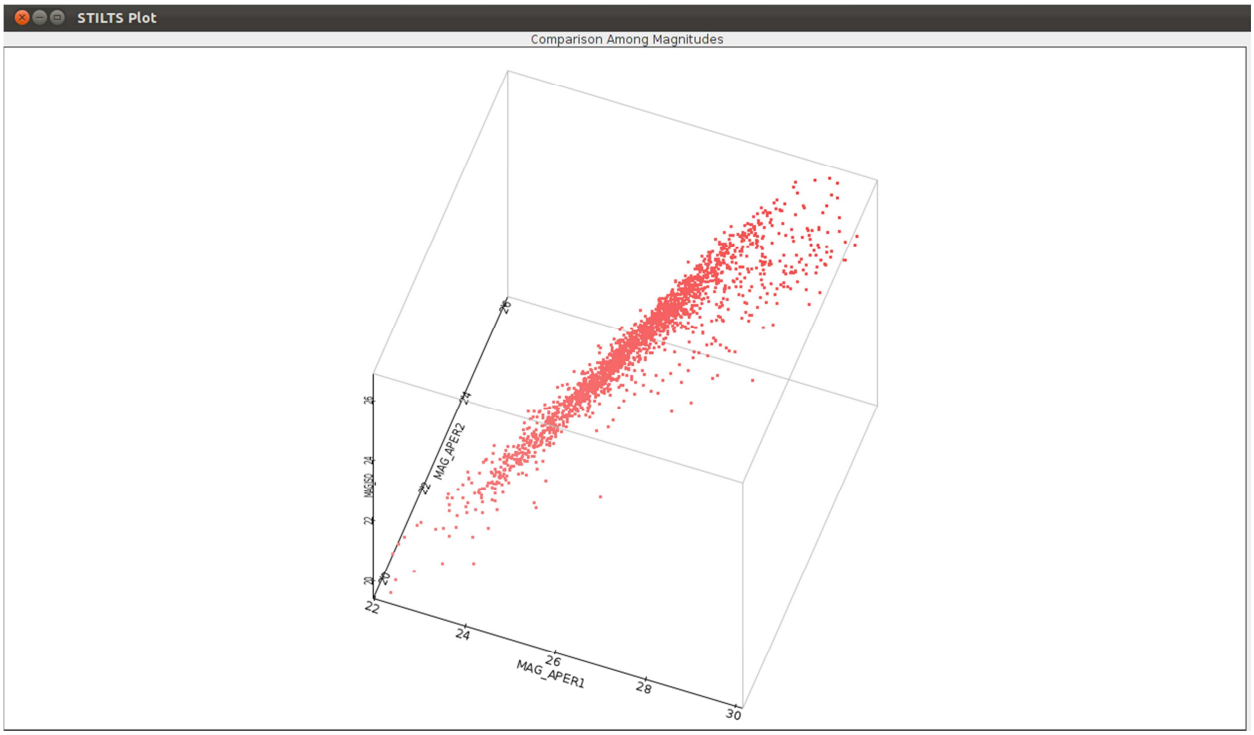


Figura 37 – Confronto 3D tra magnitudini di apertura e isofotale per il campione di oggetti correttamente classificati. Vista per il confronto diretto fra MAG_APER1 e MAG_APER2.

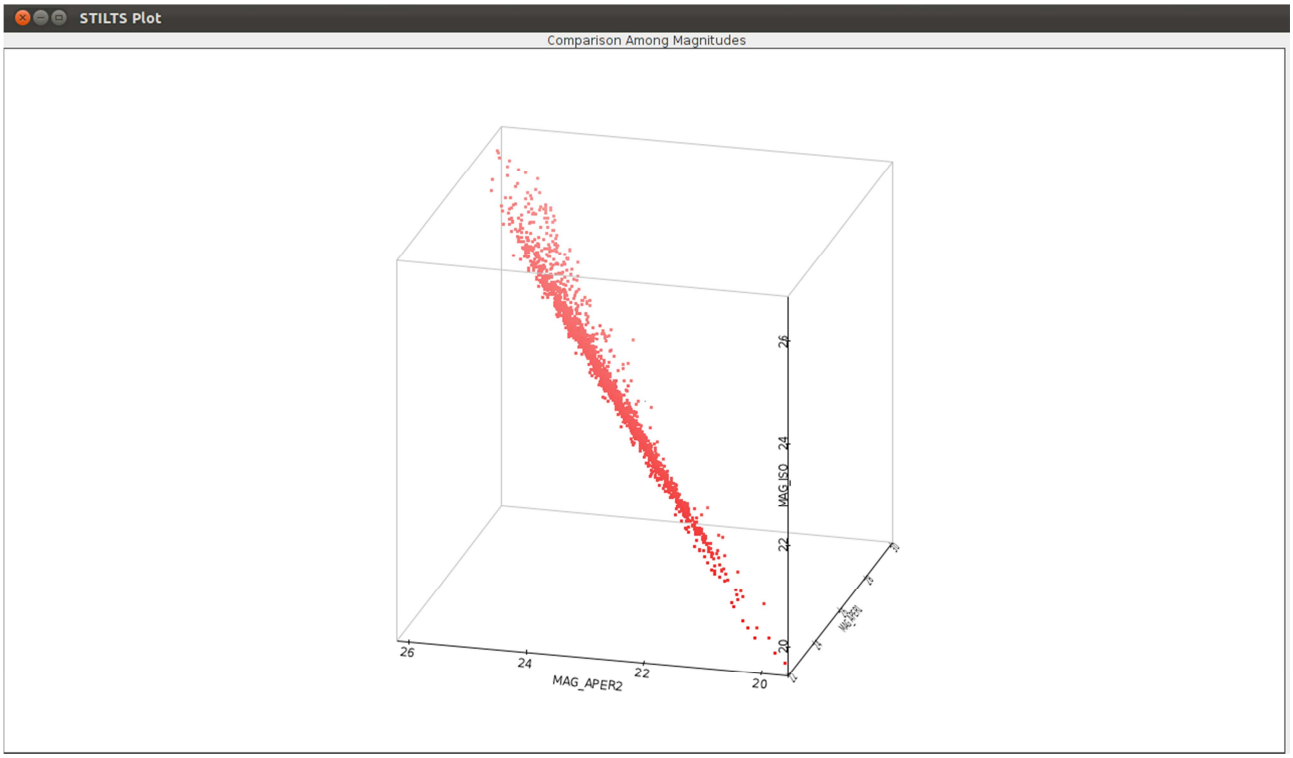


Figura 38 – Confronto 3D tra magnitudini di apertura e isofotale per il campione di oggetti correttamente classificati. Vista per il confronto diretto fra MAG_APER2 e MAG_ISO.

In questo tool si nota come sia possibile cambiare i parametri per il diagramma “Scatter Plot 3D”, che sono: **XAxis, YAxis, ZAxis, XFlip, YFlip, ZFlip, Marker size, Marker shape, Colour, Title, XLabel, YLabel, Grid, Phi, Theta.** Il che soddisfa il requisito 8;

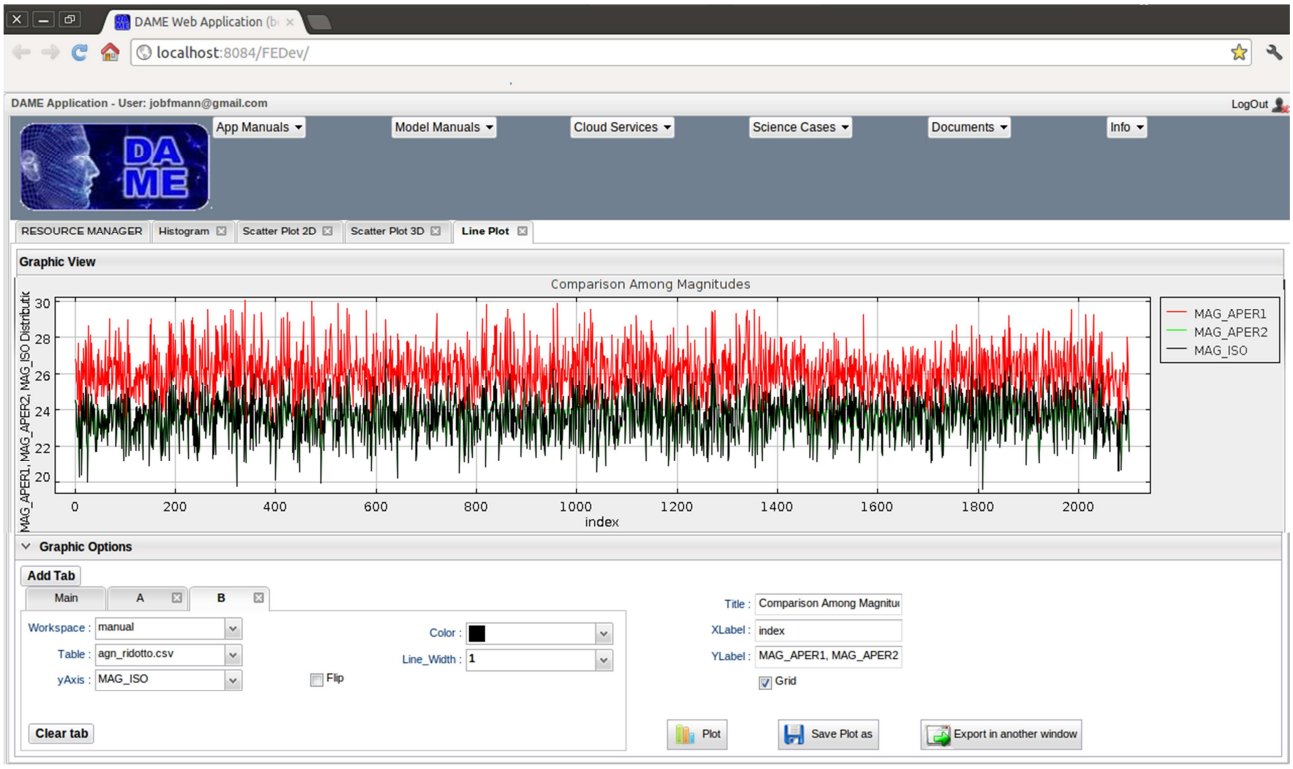


Figura 39 – Confronto mediante grafico di tipo Line Plot tra diverse magnitudini per il campione di oggetti correttamente classificati. Il confronto è eseguito tra 2 magnitudini di apertura (rosso: MAG_APER1, verde: MAG_APER2) e la magnitudine isofotale (nero: MAG_ISO)

In questo tool si nota come sia possibile cambiare i parametri per il diagramma “Line Plot”, che sono: **YAxis**, **YFlip**, **Colour**, **Line Width**, **Title**, **XLabel**, **YLabel**, **Grid**. Il che soddisfa il requisito 9;

Dunque, quest’insieme di tools verificano i requisiti funzionali enucleati nel capitolo 4.

6. Conclusioni e prospettive future

L'infrastruttura di esplorazione dati DAMEWARE, sin dalle prime fasi della sua progettazione, aveva tra gli obiettivi principali la necessità di dotarsi di un sistema d'integrazione di nuovi modelli di *data mining* e funzionalità di trattamento dati, basato sulla tecnica *plug-and-play*. Ciò al fine di renderla capace di venire incontro alla molteplicità delle esigenze dei potenziali *stakeholders*, permettendone il facile ed intuitivo uso anche da parte di utenti non esperti in tecnologie informatiche.

Tale obiettivo finora era stato solo parzialmente raggiunto, mancando di strumenti efficienti di analisi grafica e visualizzazione dati, che permettessero le principali operazioni di pre- e post-processing sui dati, complementari rispetto alla loro elaborazione.

Il presente lavoro di tesi riguarda la progettazione e sviluppo di strumenti software specializzati nella rappresentazione efficiente di grandi volumi di dati, integrati nella web application attraverso il meccanismo plug&play. Per rappresentazione s'intende la creazione e personalizzazione di grafici (istogrammi, scatter plot 2D e 3D) e di visualizzazione avanzata di immagini ad alta risoluzione (in particolare immagini astronomiche ricavate da speciali sensori a milioni di pixel).
Il valore intrinseco del presente lavoro di tesi consiste nell'introduzione di efficienti e versatili strumenti grafici di analisi visuale e prospettica all'interno di un'infrastruttura di analisi ed esplorazione di dati complessi, creando un unico sistema completo di indagine scientifica. I vari tools grafici consentono infatti alla web application di fornire all'utente la possibilità di effettuare, in modo completo, workflow di indagine su grandi volumi di dati. In ambito prettamente astrofisico e legato ai sistemi di data mining, l'integrazione di tali tools hanno consentito al progetto DAMEWARE di poter competere con i principali sistemi di esplorazione dati disponibili sul Web. I principali servizi esistenti sono, in diversa misura, parzialmente carenti, in termini di rappresentazione o di elaborazione dei dati. DAMEWARE, attraverso i tools grafici sviluppati con questo lavoro di tesi, completa e coniuga entrambe le tipologie di esigenze, permettendo la realizzazione di esperimenti e workflow scientifici completi in un unico servizio.

In termini di future attività legate ai servizi e tools grafici e di visualizzazione dati, la tecnica del plug&play con cui sono stati progettati e sviluppati, permette di estendere tali strumenti nella web application con estrema rapidità e semplicità, senza necessità alcuna di modifiche sui componenti software pre-esistenti.

7. Appendice

Questa sezione del documento è dedicata a tutti i contenuti che non sono stati inseriti nei precedenti paragrafi per evitare di occupare eccessivo spazio. In particolare sono presenti tabelle di Cockburn relative allo Use Case nel capitolo 4 e i Sequence Diagram relativi al Class Diagram, sempre nel capitolo 4.

7.1 Tabelle di Cockburn

USE CASE #1	Richiedi plot		
Goal in Context	L'utente vuole richiedere la creazione di un grafico		
Preconditions	L'utente è loggato ed esiste almeno un file supportato ma non "editable"		
Success end condition	Richiesta di creazione grafico effettuata		
Failed End Condition	Richiesta di creazione grafico non effettuata		
Primary ,Secondary Actors	Utente		
Trigger	L'utente seleziona 'Plot Maker'		
DESCRIPTION	Step	Utente	FE
	1	Seleziona il pulsante 'Plot Maker'	
	2		Apre 5 nuove tab, una per ogni tipologia di grafico
	3	Sceglie la tab del tipo di grafico che gli interessa	
	4	Seleziona un workspace ed un file usando le combo box	

	5		Tramite chiamate asincrone riempie man mano le combobox.
	6	Seleziona gli assi e le altre caratteristiche del grafico che vuole ottenere tramite il pannello delle opzioni	
	7	Seleziona il pulsante 'Plot'	
	8		Manda una richiesta al FW che include l'URI del file e tutti i parametri per quel tipo di grafico
	9		Riceve un messaggio di OK dal FW insieme al grafico richiesto in formato immagine
Extensions A	Non sono stati riempiti tutti i campi obbligatori		
	8a		Visualizza un messaggio di errore segnalando i campi da riempire

USE CASE #2	Visualizza Plot		
Goal in Context	L'utente consulta il grafico prodotto		
Preconditions	L'utente è loggato ed ha richiesto la produzione di un grafico		
Success end condition	Plot visualizzato		
Failed End Condition	Plot non visualizzato		
Primary ,Secondary Actors	Utente		
Trigger	USE CASE #1		
DESCRIPTION	Step	Utente	FE
	1		Visualizza il grafico richiesto nella sezione apposita della tab
	2	Consulta il grafico ottenuto all'interno della tab	
Subvariation A			
	2A	Seleziona il pulsante 'Open in a new tab'	
			Visualizza un pop-up per chiedere il format desiderato
		Seleziona il formato desiderato	
	3A		Manda una richiesta al FW per la produzione del grafico indicando il nuovo formato

	4A		Riceve un messaggio di OK dal FW insieme al grafico richiesto nel nuovo formato
	5A		Apre una nuova finestra del browser e vi inserisce il file appena ottenuto
	6A	Consulta il grafico ottenuto nella nuova finestra	

USE CASE #3	Visualizza/Interagisci con immagine		
Goal in Context	L'utente vuole visualizzare ed interagire con un'immagine		
Preconditions	L'utente è loggato ed esiste almeno un file supportato ma non "editable"		
Success end condition	Immagine visualizzata		
Failed End Condition	Immagine non visualizzata		
Primary ,Secondary Actors	Utente		
Trigger	L'utente seleziona 'Image viewer'		
DESCRIPTION	Step	Utente	FE
	1	Seleziona il pulsante 'Image viewer'	

	2		Apre una nuova tab per la scelta e visualizzazione dell'immagine
	3	Seleziona un workspace ed un'immagine usando le combo box apposite	
	4		Rende visibile il pulsante 'Load Image'
	5	Seleziona il pulsante 'Load Image'	
	6		Manda una richiesta al FW che include l'URI del file
	7		Riceve un messaggio di OK dal FW insieme al file richiesto
	8		Visualizza il file nella sezione apposita della tab
	9	Interagisce con l'immagine tramite lo zoom e le scrollbar	
Extensions A	Il file non è disponibile		
	7a		Riceve un messaggio di KO dal FW

	8a		Visualizza un messaggio con il codice dell'errore
Subvariation A			
	9A	Seleziona il pulsante 'Save Image as'	
	10A		Visualizza un pop-up con la scelta del tipo di salvataggio
	11A	Sceglie la modalità salvataggio su HD	
	12A	USE CASE #4	

USE CASE #4	Download file		
Goal in Context	L'utente vuole scaricare sulla propria macchina un file		
Preconditions	L'utente è loggato		
Success end condition	Download effettuato		
Failed End Condition	Download non effettuato		
Primary ,Secondary Actors	Utente		
Trigger	L'utente seleziona la modalità 'Download on HD'		
DESCRIPTION	Step	Utente	FE
	1	Sceglie la modalità salvataggio su HD	
	2		Visualizza un pop-up per la scelta del formato di salvataggio

	3	Sceglie il formato desiderato	
	4		Esegue le operazioni per configurare il trasferimento del file
	5		Trasferisce il file al Client

USE CASE #5	Salva in workspace		
Goal in Context	L'utente vuole salvare un file all'interno del workspace		
Preconditions	L'utente è loggato		
Success end condition	File salvato		
Failed End Condition	File non salvato		
Primary ,Secondary Actors	Utente		
Trigger	L'utente seleziona la modalità 'Save in workspace'		
DESCRIPTION	Step	Utente	FE
	1	Sceglie la modalità salvataggio in workspace	
	2		Visualizza un pop-up per la scelta del formato di salvataggio e del workspace

	3	Sceglie il formato e il workspace desiderato	
	4		Aggiunge il file al workspace dell'utente
	5		Visualizza un messaggio di successo

7.2 Diagrammi di Sequenza

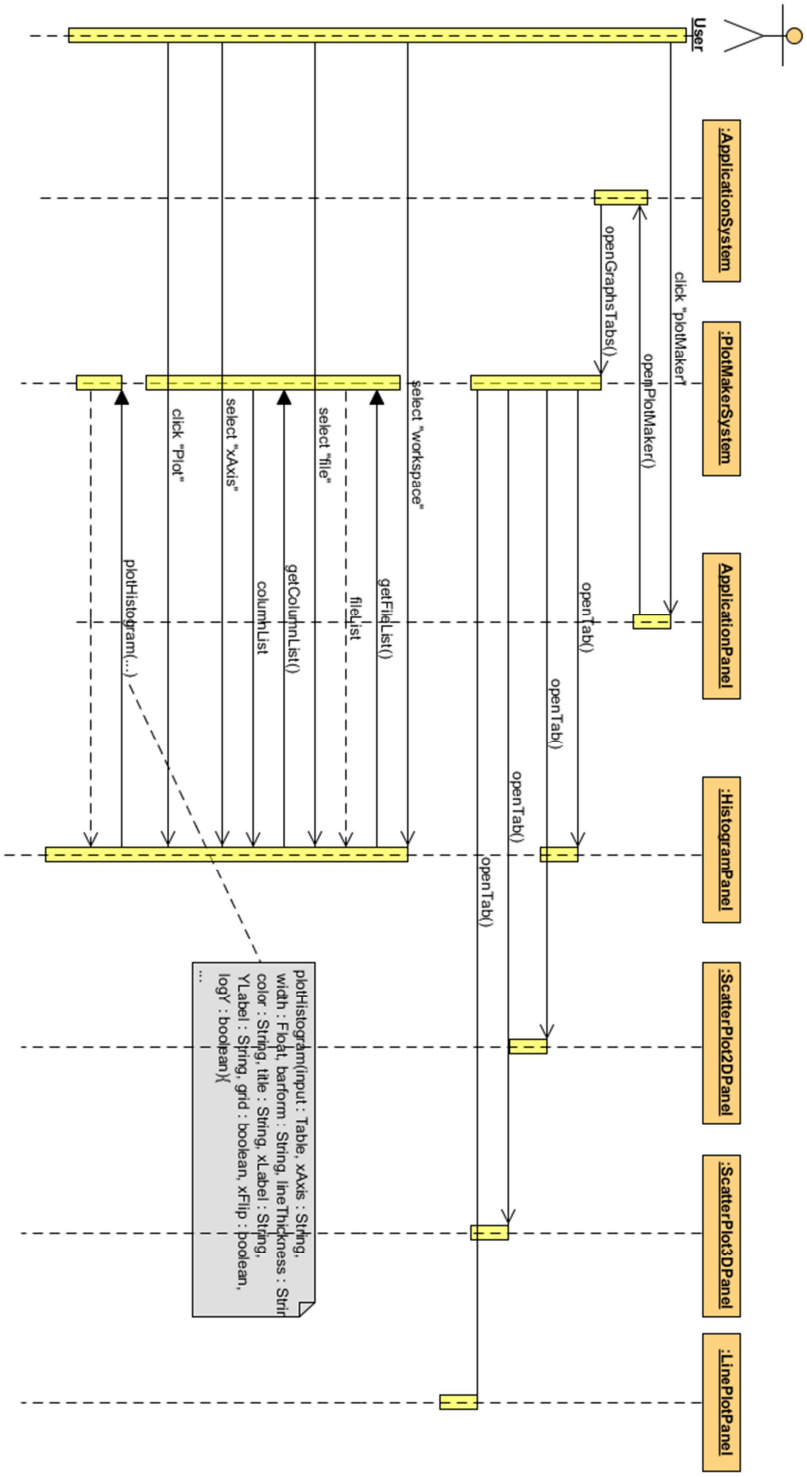


Figura 40 – Sequence della creazione grafico Histogram

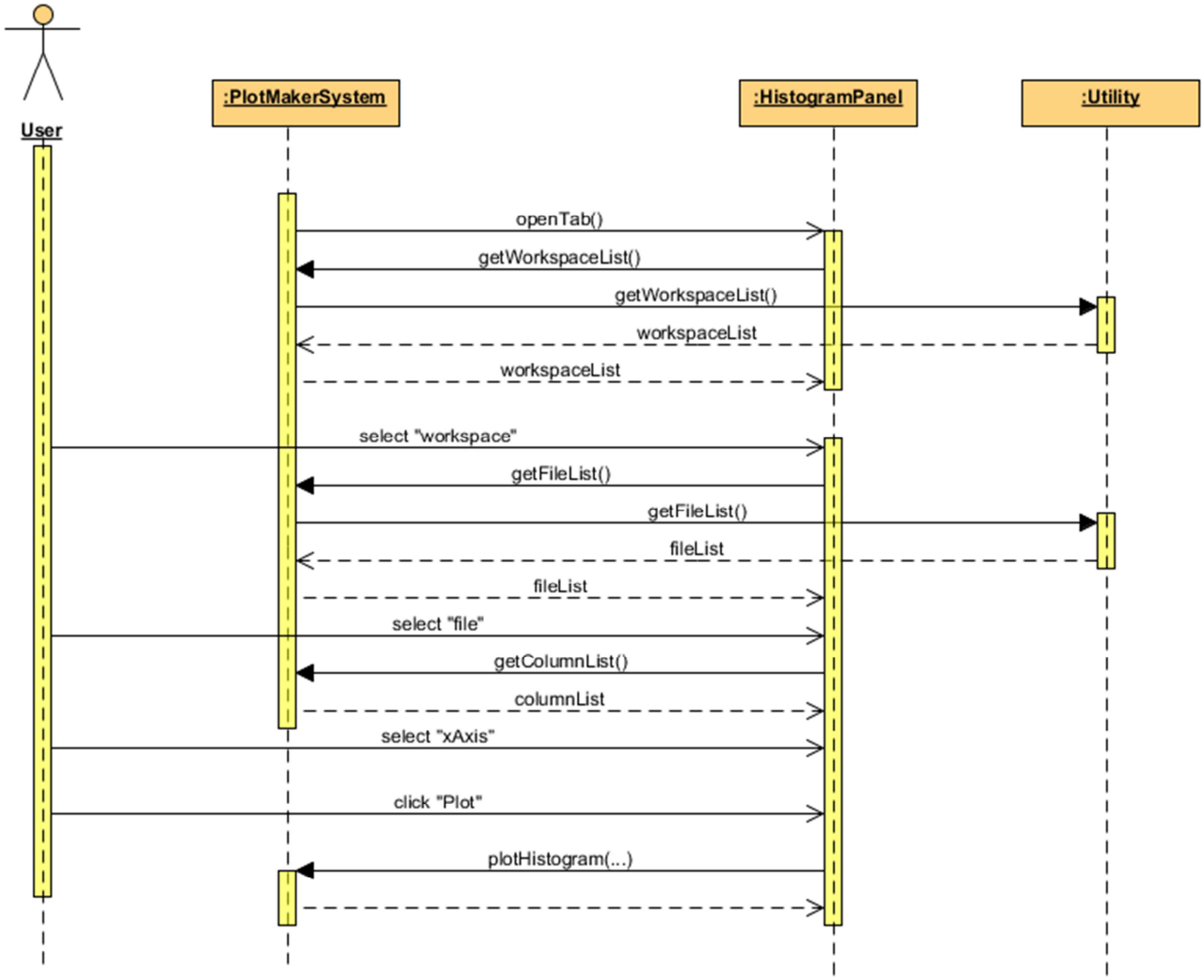


Figura 41 - Sequence del dettaglio apertura tab e riempimento combobox

In questo Sequence è rappresentato in dettaglio come avviene l'apertura in una delle tab dei grafici e di come vengono inserite le informazioni all'interno delle varie combobox interfacciandosi con la classe Utility. Il diagramma è esemplificativo ma vale per ogni tipo di grafico.

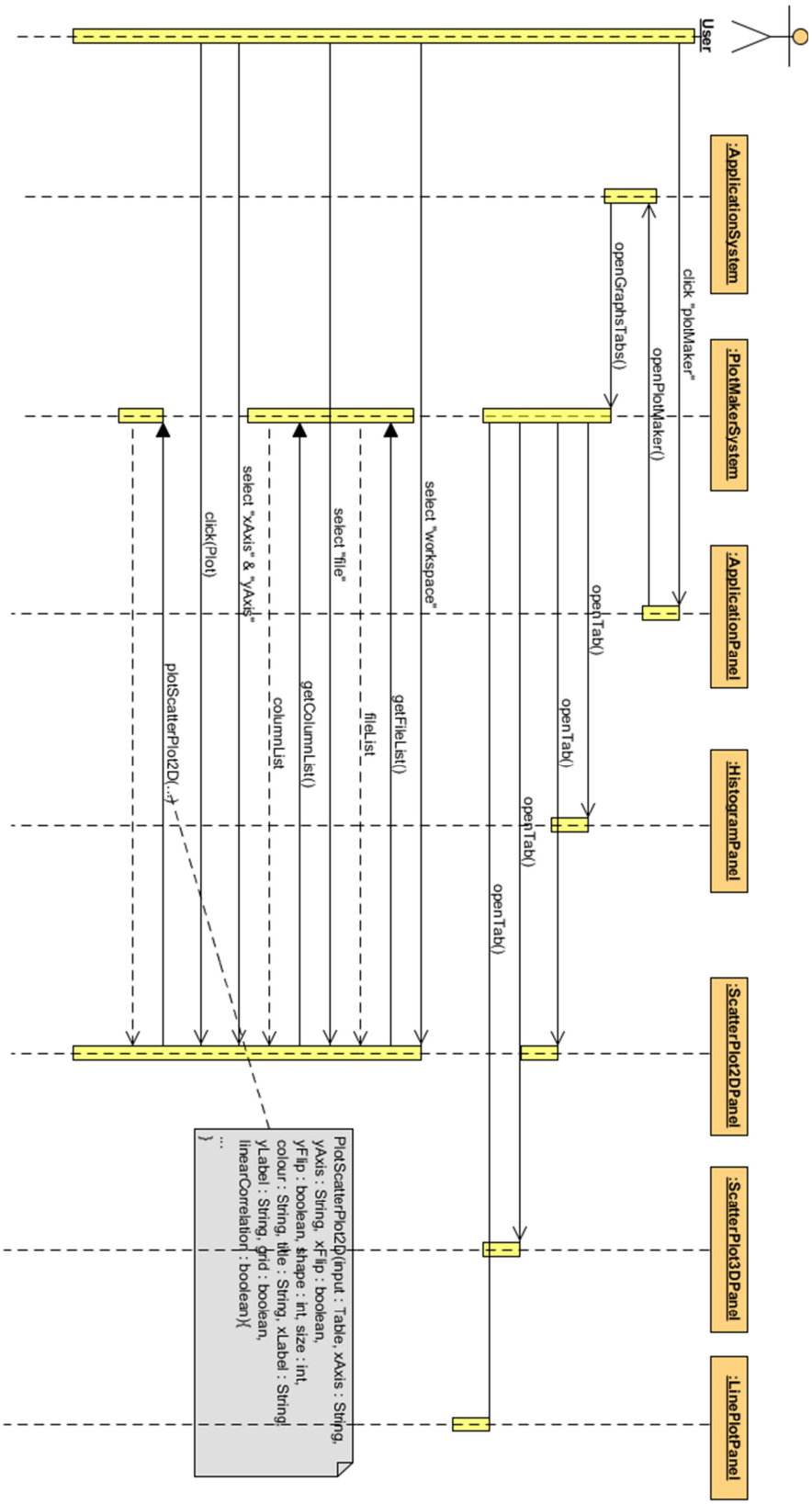


Figura 42 – Sequence della creazione grafico Scatter Plot 2D

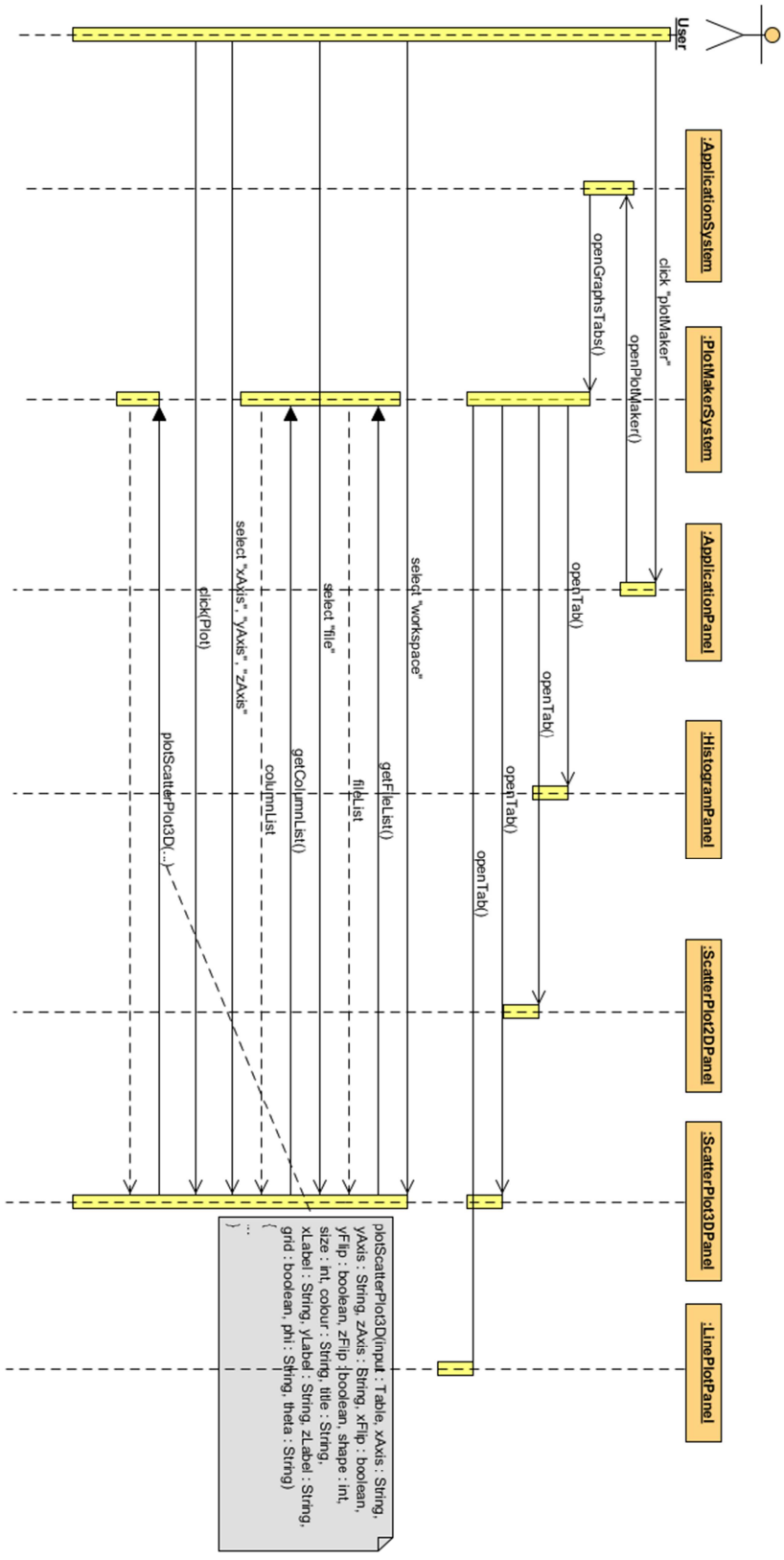


Figura 43 – Sequence della creazione grafico Scatter Plot 3D

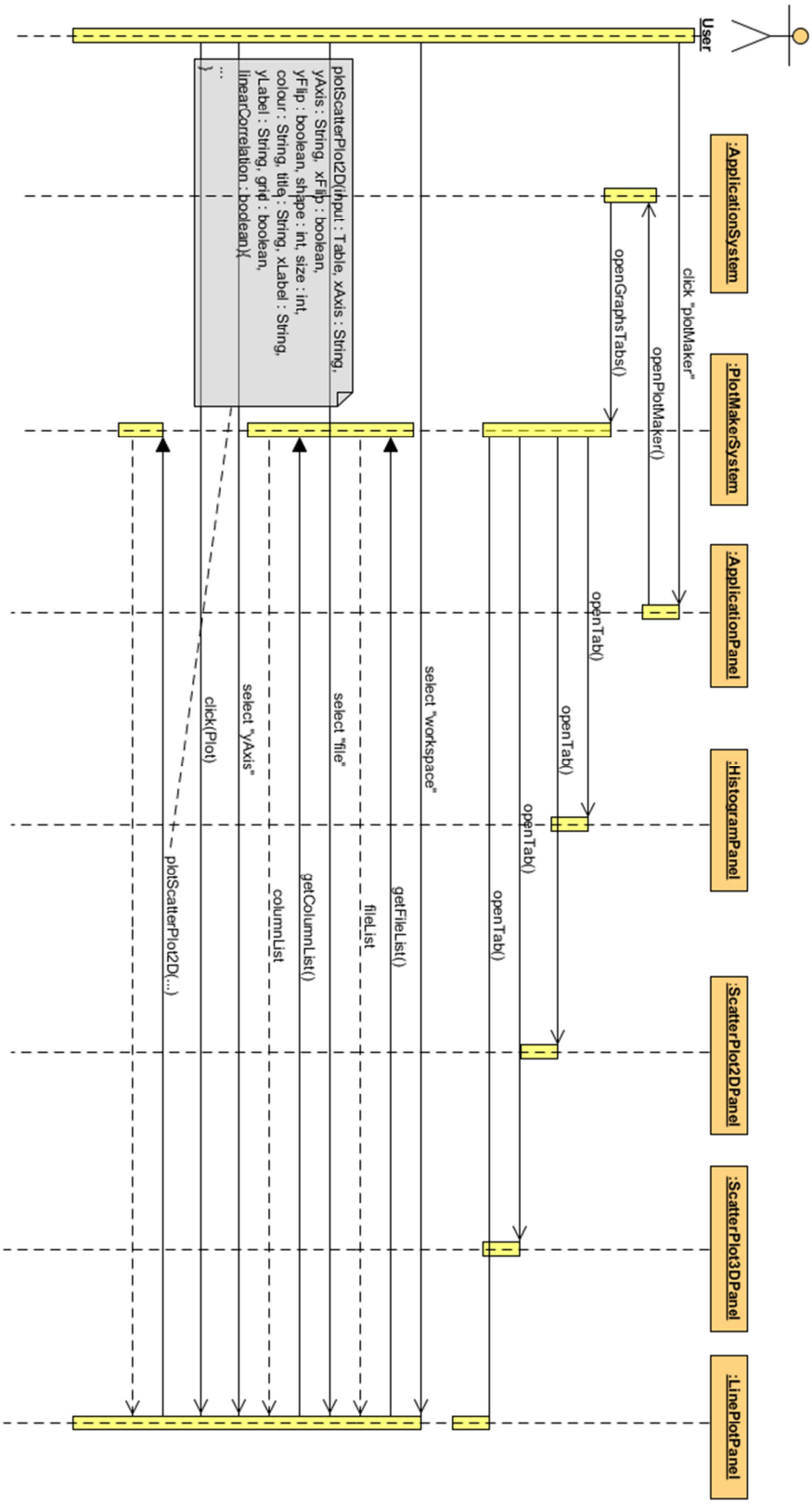


Figura 44 – Sequence della creazione grafico Line Plot

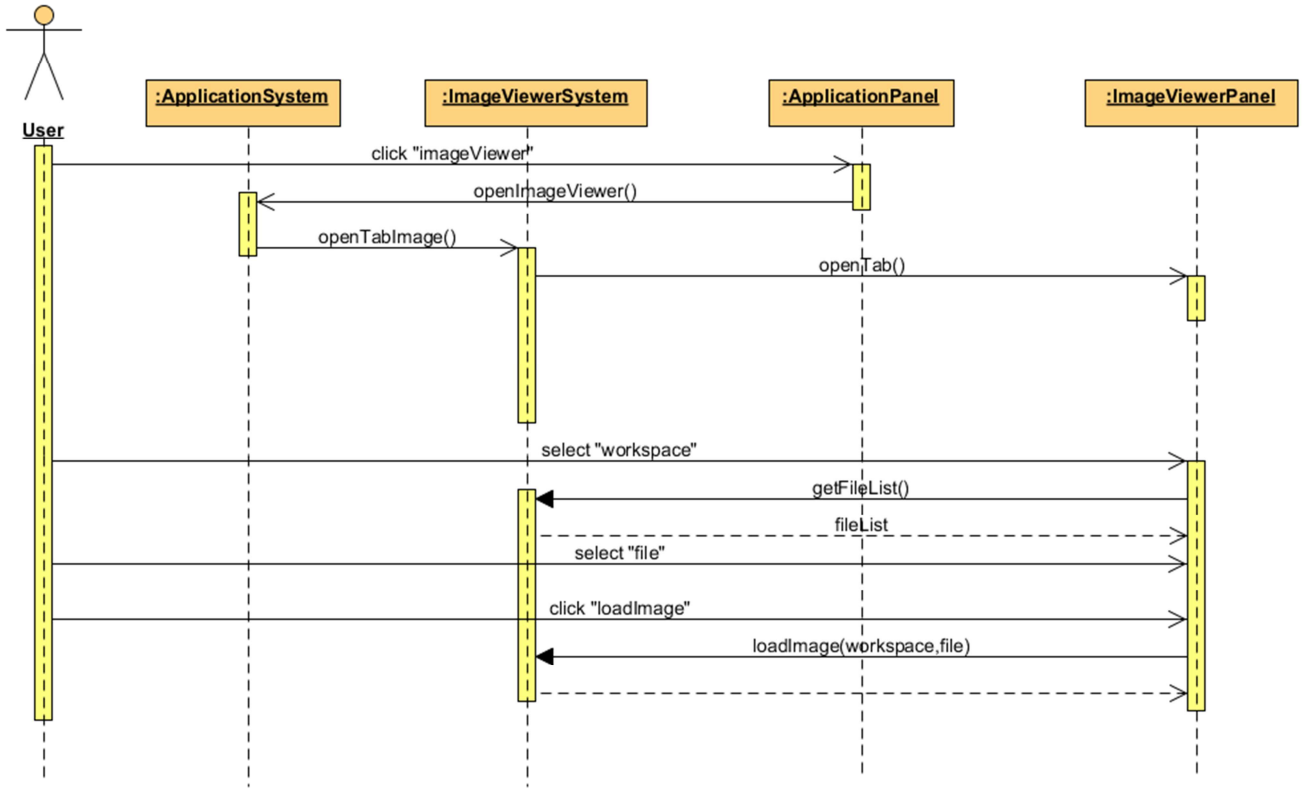


Figura 45 – Sequence visualizzazione immagine in Image Viewer

BIBLIOGRAFIA E SITOGRAFIA

Brescia, M., Longo, G., 2012. *Astroinformatics, data mining and the future of astronomical research*, Nuclear Instruments and Methods in Physics Research A, NIMA Elsevier Journal, arXiv:1201.1867

Castellani, V., 1985. *Fondamenti di Astrofisica Stellare.* Zanichelli.

Fabbiano, G., Calzetti, D., Carilli, C., Djorgovski, S.G., 2010. *Recommendations of the VAO Science Council*. arXiv:1006.2168v1

Hey, T.; Tansley, S.; Tolle, K., 2009. *The Fourth Paradigm: Data-Intensive Scientific Discovery;* ISBN-10: 0982544200, 2009; Microsoft Research, Redmond Washington, USA

Powell T., 2008. *Ajax: The Complete Reference (Paperback);* The McGraw-Hill Companies

[1] DAME sito, http://voneural.na.infn.it

[2] Topcat sito, http://www.star.bris.ac.uk/~mbt/topcat/

[3] Aladin sito, http://aladin.u-strasbg.fr/

[4] Knime sito, http://www.knime.org/

[5] Orange sito, http://orange.biolab.si/

[6] Weka sito, http://www.cs.waikato.ac.nz/ml/weka/

[7] Google chart Api, https://developers.google.com/chart/

[8] Manuale Ajax, : http://manuals.its.virginia.edu/training/webcert/beyondhtml/ajax.html

[9] Dojo sito, http://dojotoolkit.org/

[10] Sencha Ext JS sito, http://www.sencha.com/products/extjs

[11] Script.aculo.us sito, http://script.aculo.us/

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

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

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

[11] Comando “plothist”, http://www.star.bris.ac.uk/~mbt/stilts/sun256/plothist.html

[12] SpinnerItem, http://www.smartclient.com/smartgwt/javadoc/com/smartgwt/client/widgets/form/fields/SpinnerItem.html

[13] SectionStack, http://www.smartclient.com/smartgwt/javadoc/com/smartgwt/client/widgets/layout/SectionStack.html

[14] SectionStackSection, http://www.smartclient.com/smartgwt/javadoc/com/smartgwt/client/widgets /layout/SectionStackSection.html

[15] HLayout, http://www.smartclient.com/smartgwtee/javadoc/com/smartgwt/client/widgets/layout/HLayout.html

[16] VLayout, http://www.smartclient.com/smartgwtee/javadoc/com/smartgwt/client/widgets/layout/VLayout.html

[17] Oggetto “Button”, http://www.smartclient.com/smartgwt/javadoc/com/smartgwt/client/

widgets/Button.html

[18] Comando “plot2d”, http://www.star.bris.ac.uk/~mbt/stilts/sun256/plot2d.html

[19] Comando “plot3d”, http://www.star.bris.ac.uk/~mbt/stilts/sun256/plot3d.html

[20] Generazioni di stelle, http://it.wikipedia.org/wiki/Popolazioni_stellari.

[21] Il catalogo di Messier, http://messier.seds.org/

[22] New General Catalogue, http://spider.seds.org/ngc/ngc.html