ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

FACOLTA' DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

TESI DI LAUREA
in
ROBOTICA INDUSTRIALE M

SVILUPPO DI UNA PIATTAFORMA WEB PER UN LABORATORIO REMOTO DI ROBOTICA MOBILE

CANDIDATO
Stefano Zanotti

RELATORE
Chiar.mo Prof. Claudio Melchiorri

CORRELATORI
Dott. Ing. Riccardo Falconi
Ing. Raffaele Grandi

Anno Accademico 2010/2011
Sessione I

Sviluppo di una piattaforma web per un laboratorio remoto di robotica mobile

Stefano Zanotti

20-07-2011

You can try the best you can,
if you try the best you can,
the best you can is good enough.
(Radiohead, Optimistic)

Parole chiave:

Laboratorio remoto
Robotica mobile
Model Driven Software Architecture
Web Applications

L.A.R., Laboratorio di Automazione e Robotica.
D.E.I.S., Dipartimento di Elettronica, Informatica e Sistemistica.
Università di Bologna.
Software utilizzati:
Eclipse - http://www.eclipse.org/
Enterprise Architect - Sparx System
Motion - Kenneth Jahn Lavrsen
Drupal - http://drupal.org/
TexMaker - http://www.xm1math.net/texmaker/
Tesi scritta in LaTeX 2_ε, la stampa in Postscript.
Le immagini sono create con Adobe Photoshop.
Adobe Photoshop un marchio registrato di Adobe System, Inc.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Prefazione

La possibilità di effettuare esercitazioni controllando dispositivi da remoto è al giorno d'oggi una realtà concreta e realizzabile,in grado di permettere agli studenti di effettuare un'esperienza pratica senza essere fisicamente presenti all'interno di un laboratorio.
WebLab si propone come una piattaforma per permettere ai docenti di gestire un vero e proprio laboratorio virtuale, completo di esercitazioni e valutazioni, in grado di controllare i setup di un laboratorio reale a cui gli studenti potranno accedere, esaminando gli esercizi, caricando le loro soluzioni e osservando immediatamente le conseguenze delle loro azioni sui dispositivi reali.
Esamineremo in questa tesi il percorso che ha portato alla realizzazione della piattaforma WebLab e illustreremo i risultati ottenuti.
Nel Capitolo 2 ci soffermeremo sul concetto di laboratorio remoto, citando e illustrando alcuni esempi esistenti e soffermandoci su quello che offrono e sulle loro mancanze.
Il Capitolo 3 introduce la piattaforma WebLab, illustrandone a grandi linee gli scopi e le parti che la compongono.
Nei Capitoli 4 e 5 incomincia la trattazione dello sviluppo dell'applicazione WebLab, partendo dalla fase di analisi, per poi ampliarla e svilupparla nella fase di progettazione. In questi capitoli si analizzeranno gran parte delle scelte progettuali effettuate durante lo sviluppo dell'applicazione.
Col Capitolo 6 si chiude la trattazione della creazione della piattaforma WebLab evidenziando alcune scelte implementative e analizzando il

deployment dei vari sottosistemi.
In conclusione nei Capitoli 7 e 8 si effettueranno alcune considerazioni sulla questione della sicurezza dell'informazione all'interno della piattaforma e su alcuni esempi applicativi incentrati sulla gestione di laboratori remoti di robotica mobile.
In Appendice A infine verranno mostrate, attraverso esempi illustrati, alcune delle funzioni che la piattaforma WebLab mette a disposizione di docenti e studenti.

Capitolo 2

Il concetto di laboratorio remoto

2.1 Introduzione

Internet e le nuove tecnologie Web hanno avuto un impatto determinante e hanno modificato il modo in cui l'automazione e la robotica possono essere insegnate. Negli ultimi anni sono stati numerosi gli esempi di *remote education* e fra questi i laboratori remoti rivestono un ruolo di primaria importanza.
Lo scopo che si pone lo studio della robotica è quello di rendere gli studenti, al termine del loro percorso di studi, capaci di progettare, sviluppare e programmare elementi robotici. La pratica costituisce un elemento fondamentale nello studio della robotica. Lo studente dovrebbe dedicare gran parte del suo tempo ad attività pratiche di laboratorio per accumulare familiriatà ed esperienza con i dispositivi che si troverà ad utilizzare. Per raggiungere tali obiettivi è richiesta una gran disponibilità da parte dello studente a trascorrere una significativa quantità di tempo in laboratorio in attività pratiche che a volte possono richiedere materiale molto costoso e delicato. La costruzione e il mantenimento di un laboratorio che possa ospitare una quantità sufficiente di studenti richiede somme ingenti per l'acquisto e la manutenzione di robot di vario tipi, sensori, arene e ambienti dove muovere i robot e computer a disposizione degli utenti. La costruzione di una infrastruttura hardware-software che possa permettere di effettuare esercitazioni a distanza per tutti gli studenti su di un unico laboratorio automatizzato, può aiutare a ridurre i costi e mettere a disposizione una

struttura che consenta agli studenti di fare pratica 24 ore su 24 e in orari a loro più comodi.
Si possono distingue due tipi diversi di laboratori per educazione a distanza:

- laboratori virtuali: gli utenti remoti effettuano le loro esperienze utilizzando un ambiente software di simulazione.
- laboratori remoti: gli utenti interagiscono a distanza con apparecchiature e processi reali.

Implementare un laboratorio remoto richiede normalmente sforzi maggiori, in quanto l'infrastruttura software deve rapportarsi con apparecchiature reali e con le loro problematiche. I laboratori remoti costituiscono quindi una sfida maggiore ma sono allo stesso tempo più stimolanti perchè mettono l'utente in condizioni di controllare e interagire con dispositivi realmente esistenti rendendo tutta l'esperienza più coinvolgente e stimolante.

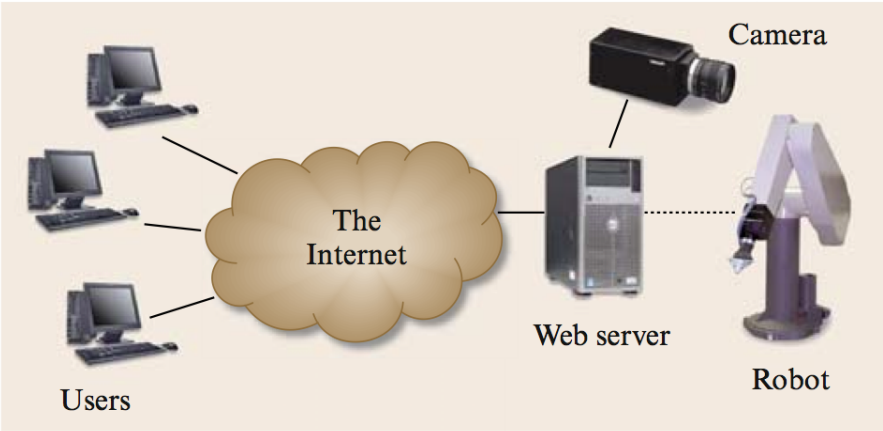

Figura 2.1: La stuttura classica di un laboratorio remoto.

2.2 Robotica remota: cenni storici e panoramica

È possibile ricondurre l'inizio degli studi sulla telerobotica agli anni 40 [1] con la ricerca e i primi veri esperimenti motivati dalla necessità di

maneggiare materiali radioattivi. Con lo sviluppo delle tecnologie Web e l'avvento delle prime webcam, nel 1993 incominciarono i primi esperimenti di robot comandati attraverso la rete. Nella decade fra il 1995 e il 2005 gli esperimenti e gli sviluppi sullo studio della robotica remota accelerarono decisamente. Furono introdotti nuovi sistemi, nuove sperimentazioni e nuove applicazioni che allargarono i campi di studio oltre le applicazioni (militari, spaziali e nucleari) che avevano motivato l'inizio della ricerca negli anni 50. È in questo momento che fanno la loro comparsa le prime applicazioni legate all'educazione, ma anche all'industria, alla sanità, alla geologia e al monitoraggio ambientale. Controllare un robot da remoto fornisce all'utente un mezzo per interagire con un ambiente che altrimenti non potrebbero raggiungere, fornendo interattività e coinvolgimento. Tutto questo si adatta perfettamente al concetto di educazione e istruzione remota. I robot controllati da remoto permettono agli studenti di avere accesso e di imparare a conoscere ed utilizzare robot spesso confinati in laboratori non accessibili al pubblico o comunque molto costosi: in quest'ottica un laboratorio remoto può migliorare notevolmente l'esperienza didattica poichè fornisce allo studente un'esperienza interattiva e diretta che altrimenti potrebbe non ricevere.
Come illustrato in Figura 2.1, un tipico laboratorio remoto include tre componenti:

- gli utenti, forniti di connessione alla rete
- un web server
- uno o più robot o un dispositivo che possa modificare il suo stato

Gli utenti accedono al sistema tipicamente attraverso un web browser connettendosi con il web server che gestisce il controllo del robot ed invia all'utente una rappresentazione visiva di quello che sta accadendo normalmente utilizzando una webcam.
Alcuni punti chiavi riguardanti la progettazione di un laboratorio remoto riguardano, ad esempio, la decisione su come rendere visivamente più appagante l'esperienza per l'utente (interfaccia, rappresentazione del risultato)

o su come collegare l'applicazione software a dei veri e propri comandi eseguiti su robot reali o ancora su come rapportarsi in merito alla compresenza o meno di più utenti in contemporanea operanti sui robot dei laboratori. Effettuare una scelta piuttosto che un'altra diversifica notevolmente i vari laboratori remoti che possono essere progettati e realizzati.

2.3 Alcuni esempi di laboratori remoti

Andremo ora ad analizzare due esempi di laboratori remoti e virtuali realizzati noti in letteratura in modo da fornire una panoramica sulla situazione attuale reale della educazione remota.

2.3.1 ACT: Automatic Control Telelab

Sviluppato a Siena [2], è usato fin dal 1999 nei corsi di controllo all'Università di Siena. Utilizza l'ambiente MATLAB/Simulink per implementare i movimenti dei vari esperimenti. Permette all'utente di utilizzare un controllore già predefinito e vederlo all'opera, o di crearne uno nuovo e personale utilizzando MATLAB/Simulink. I risultati dell'esperimento sono visibili attraverso una interfaccia che mostra un grafico e un video ripreso da una webcam. Lo scopo del progetto è quello di permettere agli studenti di applicare le proprie nozioni di controllistica implementando un controllore e vedendolo in azione senza restrizioni su orari del laboratorio o disponibilità delle apparecchiature. ACT è accessibile 24 ore su 24 da ogni computer collegato ad Internet.

Tra le caratteristiche principali di ATC troviamo:

- una interfaccia facile da usare, costruita su pagine HTML e Applet java (Figura 2.2)

- la possibilità di effettuare modifiche runtime al controllore (ad esempio modificare i parametri di un controllore PID) anche se queste potrebbe soffrire del ritardo dovuto alla trasmissione via Internet

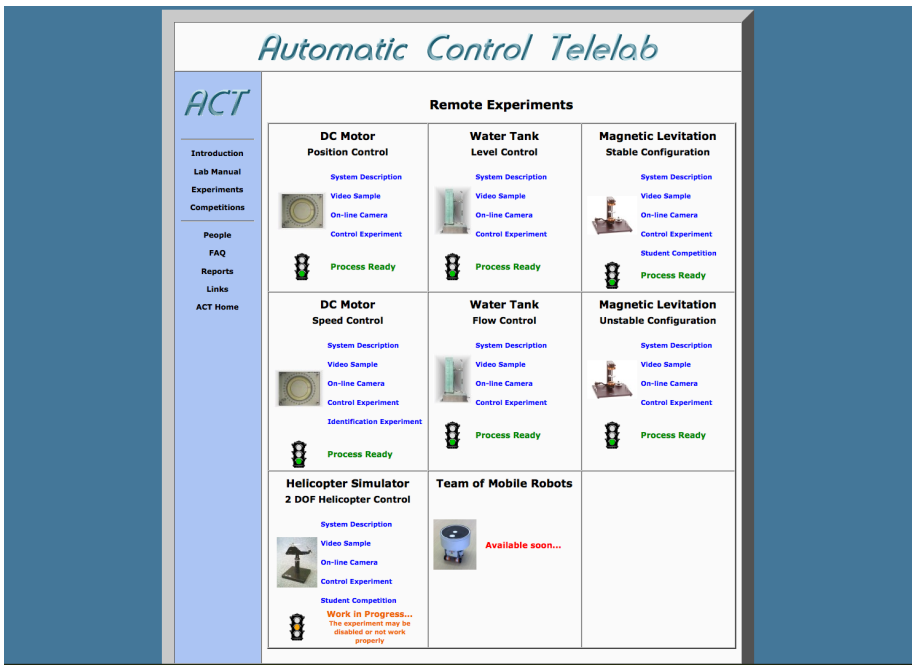

Figura 2.2: ACT: Automatic Control Telelab.

- la possibilità di vedere il risultato dell'esperimento attraverso una webcam
- un utente alla volta può prendere possesso di un esperimento per un tempo massimo di 5-10 minuti.

L'architettura del sistema è composta da due parti: un server che gestisce il processo fisico e un client che presenta una interfaccia all'utente. Il server si basa e utilizza l'ambiente MATLAB/Simulink per generare i controllori che vengono poi compilati in C ed eseguiti sui vari dispositivi. La parte client ,come detto, è invece realizzata tramite pagine HTML e applet Java. ATC presenta sicuramente alcune ottime soluzioni, come ad esempio la possibilità di effettuare diversi tipi di esperimenti diversi, o la buona gestione delle webcam e l'interessante possibilità per l'utente di creare il proprio controllore in MATLAB e vederlo in azione. D'altro canto però questa può anche essere vista come una limitazione: i tipi di esperimenti in fin dei conti sono tutti piuttosto simili tra loro. Si tratta di apparecchiature che

eseguono un compito sempre uguale per cui lo studente può risultare poco invogliato a ripetere un esperimento una volta effettuato e superato con successo. L'ambiente in generale sembra piuttosto statico, gli esperimenti e le apparecchiature sono predefinite e costanti, non prevedendo grosse varianti o possibilità.

2.3.2 The Telelabs Project

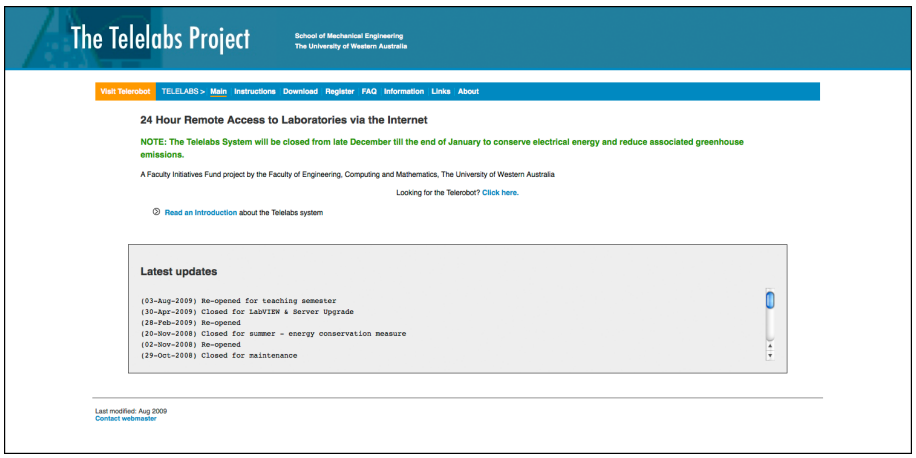

Figura 2.3: The Telelabs Project: homepage.

Sviluppato dall'University of Western Australia e frutto di un investimento economico ingente (250 mila dollari Australiani in 3 anni) il progetto Telelabs [1] (Figura 2.3) si propone di combinare tecniche avanzate di monitoraggio e di controllo con strutture di laboratorio raggiungibili via Internet per migliorare l'esperienza educativa degli studenti. Significativo è il fatto che Telelabs Project si proponga come un progetto globale e non limitato ad un singolo laboratorio virtuale, ma come una infrastruttura software applicabile a vari progetti di educazione remota sparsi per il mondo. Il software per sviluppare tale infrastruttura si basa sul linguaggio LabVIEW di National Instrument [2] . La piattaforma software sviluppata permette ad

[1] http://telerobot.mech.uwa.edu.au/
[2] http://sine.ni.com/np/app/flex/p/ap/global/lang/en/pg/1/docid/nav-77/

apparecchiature di laboratorio di interfacciarsi attraverso controllori realizzati in LabVIEW con internet, permettendo agli studenti di utilizzare una comoda interfaccia utente per controllare i dispositivi. Il risultato dell'esperimento viene poi trasmesso agli studenti attraverso il plotting di grafici significativi all'interno dell'interfaccia utente (in Figura 2.4).

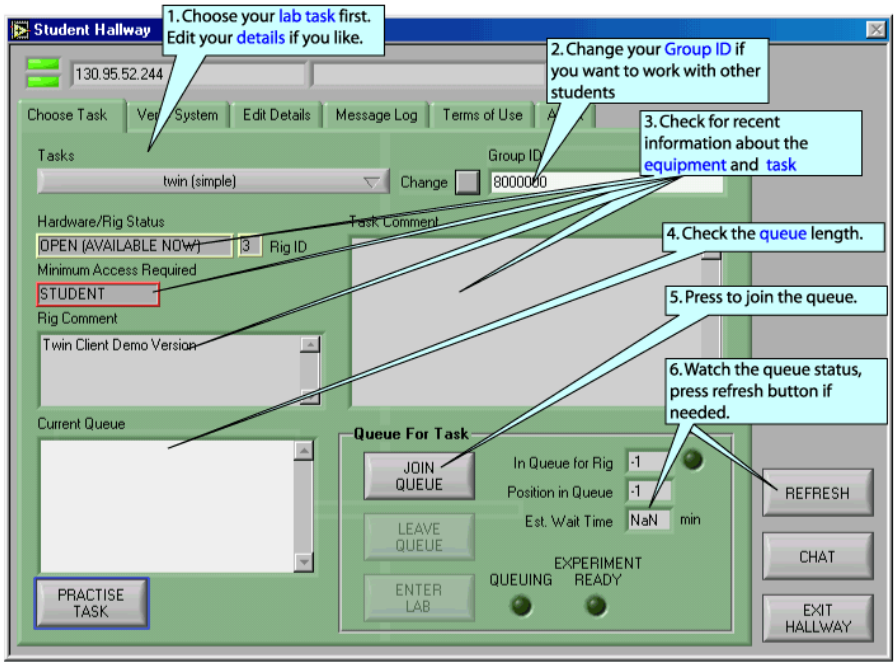

Figura 2.4: The Telelabs Project: interfaccia.

Molti sono gli aspetti positivi legati a questo progetto, in particolare l'idea di creare non una soluzione ad hoc, ma un contenitore il più generale possibile che possa adattarsi a realtà di laboratorio molto diverse tra loro. Molti sono però anche i limiti. In particolare:

- la scelta di LabVIEW come linguaggi implementativo della piattaforma. LabVIEW è un linguaggio proprietario, decisamente non comune e costoso. A chi fosse interessato ad utilizzare il software Telelabs nel proprio laboratorio remoto viene richiesta una licenza full development di LabVIEW per poter installare e creare i propri esperimenti virtuali di laboratorio

- come segnalato sul sito, i costi di mantenimento per chi fosse interessato ad utilizzare Telelabs nel proprio laboratorio sono abbastanza alti e inoltre si richiede un periodo di formazione di tre mesi da effettuare in loco per apprendere come utilizzare la piattaforma, gestire gli studenti e creare esercizi
- per accedere al laboratorio gli studenti sono obbligati a scaricare sul proprio sistema una applicazione eseguibile oltre che il LabVIEW Runtime Engine. L'applicazione è per ora disponibile esclusivamente per ambiente WINDOWS, escludendo così una buona fetta di sistemi operativi dalla possibilità di accedere al laboratorio remoto.

Quindi se da un lato l'approccio molto generico e modulabile di Telelabs Project rappresenta un punto a suo favore, la non immediatezza e difficoltà, sia economica che implementativa, nel realizzare una propria versione del laboratorio remoto utilizzando l'infrastruttura software concessa gioca decisamente a suo sfavore. Così come anche il fatto che l'utente non possa accedere facilmente al laboratorio remoto, ma che sia invece costretto a scaricare ed installare software dedicato sulla propria macchina compatibile solo con il sistema operativo Windows.

Capitolo 3

WebLab come piattaforma web

3.1 Introduzione

All'interno del contesto esposto nel Capitolo 2, avendo esaminato pregi e difetti dei laboratori remoti esistenti, si è sentita la necessità di sviluppare un piattaforma in grado di dare la possibilità agli studenti, in questo caso dei corsi inerenti la robotica, di effettuare esercitazioni da remoto direttamente da casa loro attraverso l'utilizzo di una semplice interfaccia web. Il lavoro si é dunque focalizzato sulla realizzazione di una infrastruttura software, la più generica e modulare possibile, che potesse permettere questo, in particolare:

- è stato realizzato e caricato su un server un portale web per consentire agli studenti di avere accesso a tutte le informazioni ed al laboratorio virtuale vero e proprio;
- è stata predisposta e configurata una webcam con relativo webcam-server per poter permettere agli studenti, attraverso il portale web appena ricordato, di avere una visione istantanea di quello che accade nel laboratorio reale;
- è stata infine progettata e sviluppata una applicazione client-server in grado di gestire la connessione di più utenti, la gestione di esercizi da parte dei docenti e l'accesso al laboratorio virtuale vero e proprio.

Nelle prossime sezioni analizzeremo brevemente queste tre componenti.

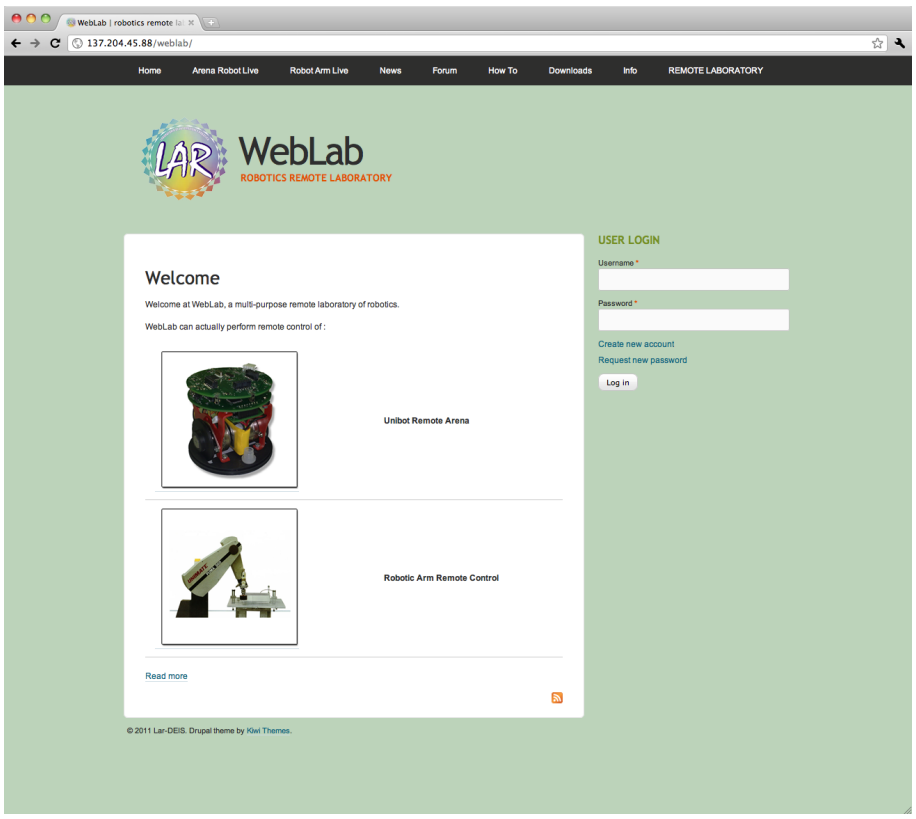

Figura 3.1: Home page del portale WebLab

3.2 Il laboratorio sul web

Effettuare esercitazioni da remoto deve essere per lo studente il più intuitivo e semplice possibile, in modo che possa concentrarsi completamente nella soluzione dell'esercizio evitando di perdere tempo ed energie preziosi in questioni organizzative e pratiche che dovrebbero essere immediate. Inoltre lo studente deve avere la possibilità di poter accedere a tutte le informazioni necessarie in maniera semplice, chiara e veloce.

Per questo motivo la realizzazione di un sito o portale web risulta essere necessaria, in particolar modo se si rende disponibile la possibilità di accedere al laboratorio virtuale direttamente dalle pagine del sito web, evitando così allo studente l'obbligo di scaricare un applicativo sulla propria macchina con tutti i problemi connessi riguardanti download, aggiornamenti,

configurazioni varie e incompatibilità con i vari sistemi operativi.

Tutto quello che lo studente dovrà fare sarà dunque connettersi ad un portale che automaticamente presenterà, all'interno di una semplice pagina web, una interfaccia per interagire col laboratorio virtuale. Tutta l'esperienza riguardante l'esercitazione e la partecipazione al laboratorio virtuale verrà svolta all'interno di un normale browser senza necessità di nessuna applicazione esterna. Il mantenimento e l'aggiornamento di tale interfaccia web sarà poi a carico dello sviluppatore in maniera totalmente trasparente all'utente.
Il portale web deve inoltre essere fonte di informazioni per l'utente, sia riguardanti l'utilizzo dell'applicazione sia riguardanti argomenti più generici come la materia di studio, altri programmi o link inerenti. A tal proposito il sito potrà contenere ad esempio un forum dove gli utenti possano lasciare domande e la possibilità per i docenti di inserire messaggi e avvisi inerenti il corso e la piattaforma.

3.3 La gestione della webcam

L'utente deve inoltre avere la possibilità di poter vedere in azione il laboratorio durante l'esecuzione dell'esercizio, per poter avere un riscontro tangibile della propria esercitazione, e poter anche osservare l'attività del laboratorio durante il resto del giorno. Questo può essere reso possibile attraverso una webcam configurata per poter raccogliere immagini dal laboratorio e renderle disponibili all'utente. L'utente avrà dunque la possibilità di vedere quello che accade in tempo reale nel laboratorio fisico sia durante l'esecuzione del proprio esercizio, sia durante ogni momento della giornata, rendendo così l'esperienza ancora più reale e interessante.

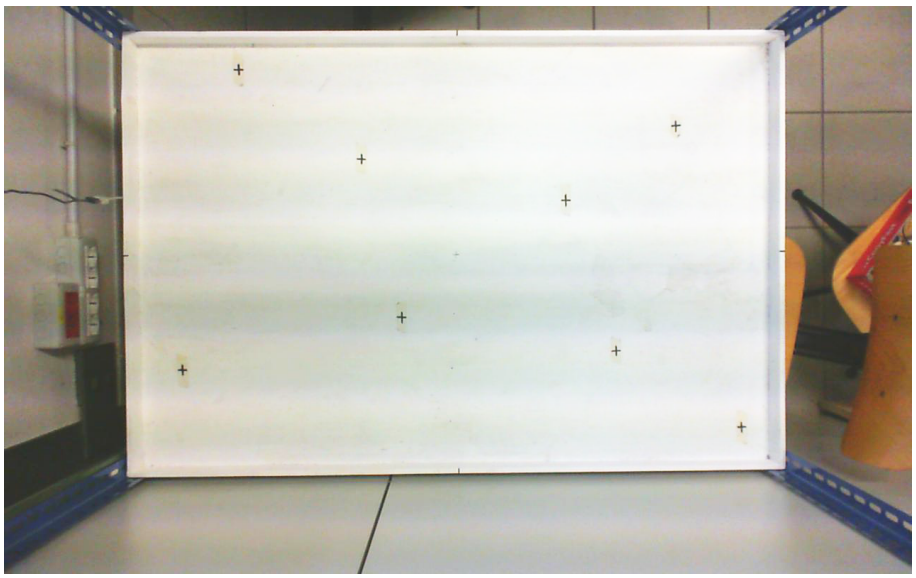

Figura 3.2: La visuale ripresa dalla webcam nel laboratorio UniBot

3.4 Applicazione WebLab

Il centro e il cuore della piattaforma web sarà però costituito dall'applicazione WebLab che avrà il compito di gestire e coordinare tutta l'attività del laboratorio virtuale. L'applicazione avrà una struttura tipicamente *client-server* e attraverso di essa studenti e docenti potranno avere accesso alle funzionalità del laboratorio remoto: gli studenti accedendo uno alla volta per poter eseguire le esercitazioni e i docenti gestendo attraverso una apposita interfaccia tutto il *background*, ovvero creazione e gestione di esercizi, visione dei risultati dello studente e soprattutto definizione e comportamento delle strutture che il laboratorio reale mette a disposizione. Nei prossimi capitoli verranno analizzati nel dettaglio la progettazione e lo sviluppo di tale applicazione, in particolare si discuterà la fase di analisi (Cap. 4), la fase di progettazione (Cap. 5) e infine la fase di realizzazione e deployment (Cap. 6).
In tutti questi capitoli si farà uso intensivo di diagrammi UML[1] per il-

[1]Unified Modeling Language,linguaggio di modellazione e specifica basato sul paradigma object-oriented. http://www.uml.org

lustrare e spiegare in maniera chiara e univoca le scelte architettoniche e progettuali fatte prima e durante la realizzazione dell'applicazione.

Capitolo 4

WebLab - Analisi

4.1 Introduzione

Lo sviluppo del sistema software alla base del laboratorio remoto prende il via con la fase di analisi [3], nella quale si cerca di mettere a fuoco e di perfezionare ciò che l'applicazione dovrà fare (requisiti) e di dare una visione astratta dei componenti del sistema software e delle relazioni fra loro. Dopo vari incontri e interazioni con i responsabili del progetto si è arrivati alla stesura di un documento contente i requisiti richiesti all'applicazione.

4.2 Requisiti

Si vuole costruire una piattaforma software che permetta agli studenti di effettuare esercitazioni di robotica da remoto utilizzando dei robot presenti in laboratorio.
La piattaforma deve essere composta da:

- un portale web in cui gli studenti possono trovare tutte le informazioni e a cui gli studenti devono registrarsi per poter accedere al laboratorio remoto
- una webcam che deve restituire agli studenti una visione in diretta dello stato del laboratorio

- una applicazione che deve permettere agli studenti ed ai professori di gestire una serie di esercitazioni

In particolare tale applicazione deve:

- permettere ai docenti di caricare e cancellare esercitazioni dal server
- permettere ai docenti di vedere quali studenti hanno eseguito quali esercizi e con che valutazione
- permettere agli studenti di vedere gli esercizi disponibili, quelli già sostenuti con relativa valutazione e di sostenere quelli ancora non sostenuti
- deve gestire la concorrenza: solo uno studente alla volta può avere accesso al laboratorio fisico e per un tempo determinato
- deve permettere allo studente di caricare la propria soluzione e vederla in azione e di ricevere una ricevuta di avvenuta esercitazione
- deve essere a conoscenza di quanti e quali robot sono disponibili e dello stato del laboratorio

Gli esercizi devono essere cosi composti:

- numero identificativo
- titolo
- testo dell'esercizio
- difficoltà
- tipo e numero di robot coinvolti
- immagine esemplificativa

Il tutto deve poi essere immaginato e implementato in uno scenario completamente distribuito, ovvero la piattaforma deve poter gestire léventualità di vari laboratori remoti all interno della stessa architettura, ognuno indipendente dall'altro. L'applicazione deve essere in grado di riconoscere quanti e quali laboratori sono attivi e di indirizzare l'utente verso il laboratorio scelto per svolgere l'esercitazione remota. Ogni laboratorio deve essere a conoscenza degli esercizi disponibili e degli studenti che hanno preso parte al laboratorio e dei loro risultati.

4.2.1 Analisi dei requisiti

Da una prima lettura dei requisiti si evince che l'applicazione deve essere pensata e progettata con in mente la sua natura distribuita, ogni istanza dell'applicazione deve essere indipendente dalle altre. Inoltre l'applicazione dovrà soddisfare in maniera diversa le richieste di due classi di utenti differenti: docenti e studenti. Particolare importanza rivestono inoltre gli esercizi che andranno progettati in maniera chiara e precisa in base all'indicazione dei requisiti.
Ogni laboratorio virtuale dovrà gestire i propri esercizi ed i propri studenti, inoltre dovrà amministrare l'accesso degli studenti al laboratorio reale, permettendo l'accesso ad uno studente alla volta e gestendo nella maniera più opportuna il tentativo di accesso a laboratorio occupato. Si dovrà quindi pensare sul come gestire questa eventualità, ad esempio attraverso meccanismi di accesso ritardato o di code di accesso.

4.3 Glossario

Al fine di evitare incomprensioni dovute ad ambiguità del linguaggio in Tab. 4.1 è riportato un glossario dei termini presenti nei requisiti.

TERMINE	SIGNIFICATO
Piattaforma Software	Insieme di componenti software che si relazionano fra loro per creare un ambiente che permetta di effettuare determinate operazioni
Studenti	particolare classe di utenti costituita da studenti del corso opportunamente registrati
Esercitazione (esercizio)	prova da superare creata dai docenti per gli studenti
Portale Web	sito o pagina web che consenta il reperimento di informazioni da parte degli utenti
Webcam	telecamera collegata al computer che consente la ripresa di immagini e la loro diffusione sul web
Laboratorio (reale e virtuale)	zona fisica o software in cui si eseguono le esercitazioni
Professori (docenti)	particolare classe di utenti costituita dai docenti dei corsi responsabili dei laboratori
Server	macchina all'interno della quale risiede l'applicazione
Valutazione	valore numerico compreso tra 1 e 10 che rappresenta la qualità della soluzione dell'esercizio
Concorrenza	capacità di gestire più utenti in contemporanea , gestendo i tempi e mantenendo consistenti i dati

Tabella 4.1: Glossario dei termini.

4.4 Casi d'uso

I diagrammi UML per i casi d'uso sono stati proposti per la prima volta da Ivar Jacobson in [6] come un metodo sistematico e naturale per chiarire, specificare ed esplicitare i requisiti attraverso la rappresentazioni di interazioni fra il sistema ed agenti esterni (detti attori) ad esso interessati. Attraverso i casi d'uso si cerca di esplicitare in modo chiaro chi userà il sistema e quali sono gli obiettivi che si intendono conseguire usando il sistema.

Attraverso i casi d'uso si può successivamente passare a definire degli scenari applicativi, in cui saranno esplicitate le modalità d'uso, chiarendo il modo in cui inizia ogni caso d'uso, le risposte che l'utilizzatore si attende dal sistema, la sequenza di passi con cui l'interazione si svolge, eventuali altri soggetti (esterni al sistema) coinvolti.
Illustreremo di seguiti alcuni primi ed esemplificativi casi d'uso del sistema ed i relativi scenari.

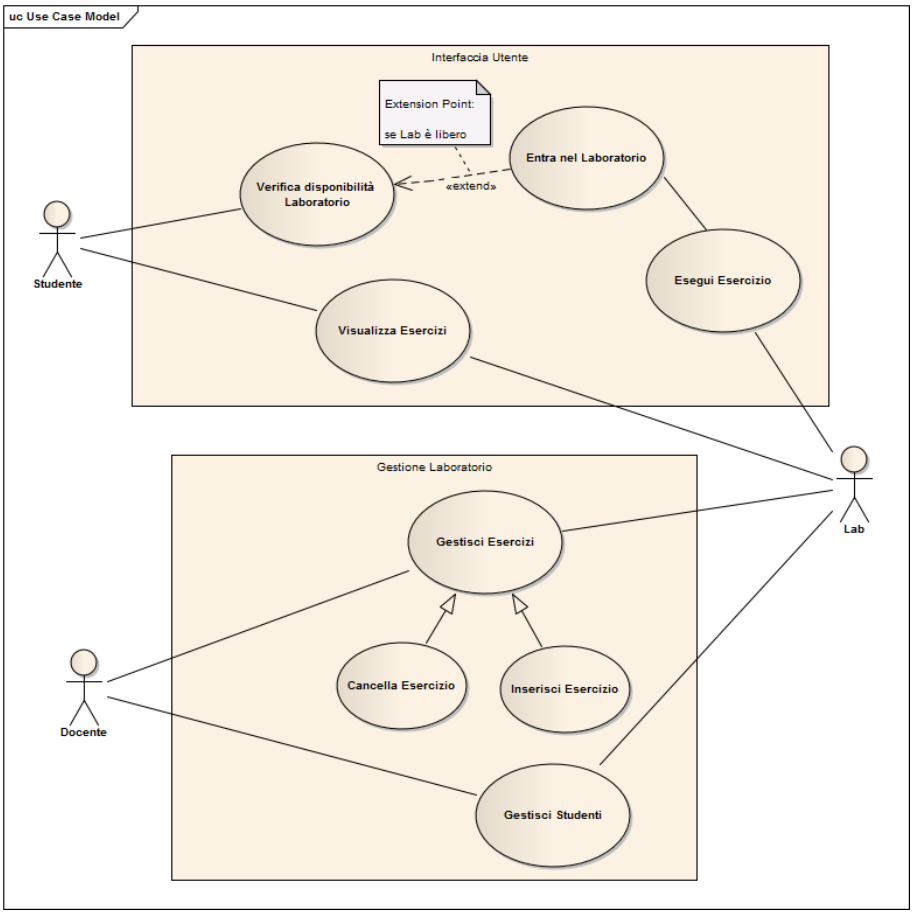

Figura 4.1: Casi d'uso.

Analizziamo ora i vari scenari derivanti dai casi d'uso.

CAMPO	DESCRIZIONE
Nome	UC1- Visualizza Esercizi
Descrizione	Lo studente esamina gli esercizi disponibili nel laboratorio
Attori	Studente, Laboratorio
Precondizioni	Laboratorio online e studente connesso
Scenario Principale	Lo studente è connesso e esamina gli esercizi disponibili. La lista di esercizi disponibili viene inviata all'utente dal laboratorio virtuale
Scenari Alternativi	Non sono presenti sul server esercizi disponibili
Postcondizioni	L'interfaccia visualizzerà l'esercizio voluto
Nome	UC2- Verifica disponibilità laboratorio
Descrizione	Lo studente fa richiesta di accesso al laboratorio virtuale
Attori	Studente
Precondizioni	Laboratorio online e studente connesso
Scenario Principale	Lo studente tenta di accedere al laboratorio per svolgere un esercizio
Scenari Alternativi	nessuno
Postcondizioni	Lo studente à entrato nel laboratorio virtuale o è stato messo in attesa
Nome	UC3- Entra nel laboratorio
Descrizione	Lo studente è riuscito ad entrare nel laboratorio
Attori	Studente
Precondizioni	Laboratorio online e libero e studente connesso
Scenario Principale	Lo studente riesce ad entrare nel laboratorio virtuale avendolo trovato libero o essendo arrivato il proprio turno
Scenari Alternativi	nessuno
Postcondizioni	L'interfaccia mostra il laboratorio virtuale
Nome	UC4- Esegui esercizio
Descrizione	Lo studente fa eseguire al laboratorio la soluzione
Attori	Studente e Laboratorio
Precondizioni	Laboratorio online, studente all'interno del laboratorio
Scenario Principale	Lo studente, ora all'interno de laboratorio virtuale, fa eseguire al laboratorio reale la soluzione di un esercizio, *uploadando* un file contenente la propria soluzione. Lo studente rimane all'interno del laboratorio virtuale fino alla fine dell'esecuzione e riceve al termine un file di report
Scenari Alternativi	L'esecuzione dell'esercizio potrebbe essere rifiutata o bloccata dal laboratorio in caso di errore nella soluzione o bloccata dall'utente stesso per qualsiasi emergenza
Postcondizioni	Lo studente al termine dell'esercitazione riceverà una valutazione e un file di report di avvenuta valutazione.

Tabella 4.2: Scenari.

Nome	UC5- Gestisci esercizi
Descrizione	Il docente gestisce gli esercizi sul laboratorio
Attori	Docente, Laboratorio
Precondizioni	Laboratorio online e docente connesso
Scenario Principale	Il docente, attraverso una interfaccia, gestisce gli esercizi presenti nel laboratorio virtuale, creandoli, cancellandoli o rendendoli invisibili allo studente.
Scenari Alternativi	nessuno
Postcondizioni	Ogni modifica agli esercizi resterà permanente nel laboratorio
Nome	UC6- Cancella esercizio
Descrizione	Il docente cancella un esercizio
Attori	Docente, Laboratorio
Precondizioni	Laboratorio online e docente connesso
Scenario Principale	Il docente attraverso una interfaccia utente può cancellare un esercizio dal laboratorio virtuale. Oltre a cancellare un esercizio il docente può anche solo nasconderlo e renderlo invisibile agli studenti.
Scenari Alternativi	nessuno
Postcondizioni	Ogni esercizio rimosso non comparirà più nel laboratorio
Nome	UC7- Inserisci esercizio
Descrizione	Il docente crea un nuovo esercizio
Attori	Docente, Laboratorio
Precondizioni	Laboratorio online e docente connesso
Scenario Principale	Il docente attraverso una interfaccia può inserire un esercizio nuovo nel laboratorio . L'esercizio nuovo deve rispettare tutti gli standard degli altri esercizi e verrà inserito attraverso una comoda interfaccia grafica
Scenari Alternativi	nessuno
Postcondizioni	Ogni esercizio inserito rimarrà permanente nel laboratorio
Nome	UC8- Gestisci studente
Descrizione	Il docente può esaminare gli studenti del laboratorio
Attori	Docente, Laboratorio
Precondizioni	Laboratorio online e docente connesso
Scenario Principale	Il docente attraverso una interfaccia grafica può controllare quali studenti sono entrati nel laboratorio , quali esercizi hanno provato a risolvere e con quali risultati
Scenari Alternativi	nessuno
Postcondizioni	nessuna

Tabella 4.3: Scenari.

4.5 Analisi del problema

Dall'analisi dei requisiti e dai relativi casi d'uso si nota che saranno da affrontare e risolvere i seguenti problemi relativi alla funzionalità del sistema:

- Configurazione e gestione degli esercizi da parte del docente (UC6 e UC7)
- Possibilità di gestire il laboratorio e gli studenti da parte del docente (UC8)
- Controllo e gestione dell'interazione dello studente col laboratorio
- Problematiche relative all'accesso dello studente al laboratorio, gestione della concorrenza (UC2)
- Gestione e impostazione dell'esperienza di laboratorio dello studente (UC4)

La problematica maggiore sarà però costituita dalla natura distribuita del sistema richiesta dai requisiti. Ogni laboratorio deve essere indipendente, con la possibilità di avere più laboratori connessi e resi disponibile allo studente. Diventa quindi necessario organizzare la struttura della soluzione in più sottosistemi software tra loro indipendenti e comunicanti attraverso messaggi remoti. Questo sarà chiarito e perfezionato meglio in sede di definizione di una architettura logica, ma possiamo comunque già anticipare una prima esemplificazione di divisione.
Il nostro sistema sarà costituito indicativamente dalle seguenti parti:

- Un sottosistema responsabile dell'autenticazione, sia di studenti che di laboratori attivi, in grado di verificare le credenziali degli studenti e di inviare ad ogni utente regolarmente identificato la lista di laboratori attivi
- Un sottosistema laboratorio, in grado di identificarsi con il sistema di autenticazione e munito di una opportuna interfaccia che permetta ai

docenti di inserire, rimuovere e cancellare esercitazioni e di verificare in ogni momento quale studente risulti connesso e cosa sta succedendo nel laboratorio. Tale sottosistema dovrà inoltre essere in grado di comunicare con gli studenti connessi, ricevendo e rispondendo alle loro richieste. Dovrà inoltre essere equipaggiato per gestire la concorrenza fra studenti, permettendo ad uno studente alla volta di accedere al laboratorio reale

- Un sottosistema per gli studenti, un *client* a tutti gli effetti, che possa identificarsi col sottosistema di autenticazione e, una volta ricevuto da questi l'elenco dei laboratori attivi, connettersi col laboratorio prescelto. Tale sottosistema, una volta connesso col sottosistema laboratorio, deve poter essere in grado di leggere gli esercizi disponibili in quel determinato laboratorio e provare ad accedere al laboratorio reale. Tale sottosistema non farà altro che inviare comandi al sottosistema laboratorio che gestirà tutte le richieste e invierà all'interfaccia utente del richiedente i risultati.

Siamo quindi in presenza di un sistema organizzato a blocchi che comunicano tra loro. L'architettura del sistema può essere impostata abbastanza naturalmente secondo una dimensione *BCE: Boundary, Control, Entity.*

La parte *Boundary* sarà costituita dal sottosistema *client* a disposizione degli studenti, che non farà altro che interagire con il resto del sistema e presentare i risultati.

La parte *Control* sarà invece costituita dal sottosistema laboratorio che realizzerà la *business logic* dell'applicazione, controllando tutte le operazioni e gestendo le richieste del *client.*

La parte *Entity* sarà invece costituita da tutte le entità in gioco e che verranno specificate meglio nel prossimo paragrafo sotto il nome di dominio dell'applicazione.

4.6 Modello del domino

La definizione di un modello del dominio costituisce un passo fondamentale nell'analisi di un sistema software poichè ci permette di rilevare e descrivere quelle entità relativamente autonome, reali o astratte, che operano in accordo a precise regole e vincoli e che costituiscono le fondamenta sulle quali costruire il nostro sistema.
In questo caso il dominio del problema è quello dei laboratori virtuali e non. Gli elementi fondamentali del dominio sono quindi costituiti dagli esercizi, dagli studenti e dai laboratori.
Andremo ora a descriverli e analizzarli tenendo conto dei tre punti di vista relativi a struttura, interazione e comportamento. Questi punti di vista costituiscono tre indispensabili dimensioni in cui articolare la descrizione di un sistema, indipendentemente dal linguaggio di programmazione usato. Nei prossimi capitoli espliciteremo ogni dimensione per ogni elemento del dominio dell'applicazione attraverso anche l'utilizzo di diagrammi UML.

4.6.1 Struttura

L'analisi della struttura di una applicazione, in questo caso del suo dominio applicativo, esamina in primo luogo l'architettura complessiva del sistema per poi approfondire ricorsivamente quella di ogni sottosistema.

Esercizi, studenti e laboratori sono rappresentati dalle classi Exercise, WLStudent e Lab che realizzano le relative interfacce IExercise,IWLStudent e ILab. Il diagramma UML in figura 4.2 mette in evidenza come le tre classi siano anche relazionate fra loro, in particolare un oggetto di tipo Lab potrà contenere al proprio interno un numero illimitato di oggetti Exercise e WLStudent.

Esercizio

La struttura degli esercizi è descritta dalla interfaccia UML IExercise e dalla classe UML Exercise che implementa tale interfaccia. Ogni esercizio, come da requisiti, è costituita da un titolo, un testo, una difficoltà, il tipo

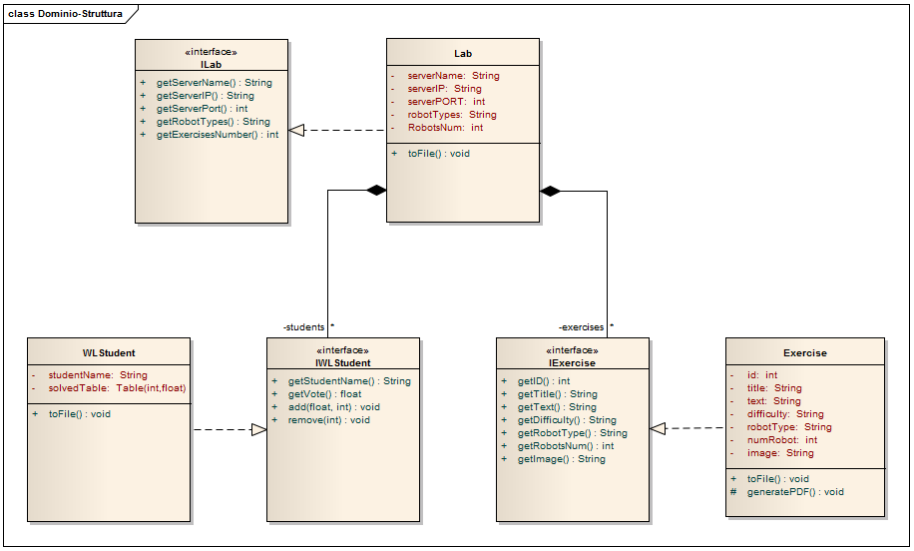

Figura 4.2: Dominio - Struttura.

e il numero di robot utilizzati e una immagine identificativa. Si è aggiunto anche un campo *id* per poter numerare in maniera progressiva gli esercizi inseriti. Le operazioni definite sono tutte operazioni di selezione (*get*) ad eccezione di una *toFile()* pensata per poter dare la possibilità di salvare gli esercizi su disco per renderli permanenti e una operazione *generatePDF()* per poter creare eventualmente un file PDF da ogni esercizio.

Studente

La struttura degli studenti è descritta dalla interfaccia UML IWLStudent e dalla classe UML WLStudent che implementa tale interfaccia. Lo studente è costituito da due attributi: il nome e una tabella in cui inserire gli esercizi risolti e la relativa valutazione. Le operazioni definite sono le classiche operazioni di selezione (tra cui anche una particolare che estrae la valutazione ottenuta in un dato esercizio) più un'operazione per aggiungere un elemento alla tabella e una per toglierlo. Anche la struttura della classe WLStudent prevede un'operazione *toFile()* per poter salvare i dati degli studenti su disco.

Laboratorio

La struttura del laboratorio è descritta dalla interfaccia UML ILab e dalla classe UML Lab che implementa tale interfaccia. Ogni laboratorio è identificato da un nome, dal proprio indirizzo IP e dalla porta usati per comunicare con l'esterno e dai tipi e numero di robot disponibili all'interno del laboratorio. Le operazioni disponibili sono tutte operazioni di selezione e anche per i laboratori si prevede una operazione *toFile()* per poter pensare di salvare i dettagli del laboratorio su disco.

4.6.2 Interazione

L'analisi dell'interazione in un sistema software si occupa di esplicitare, prima tramite una visione di alto livello poi scendendo sempre più nel dettaglio, come le varie entità del sistema comunichino fra loro. Questa dimensione gioca un ruolo sempre più importante per il corretto sviluppo del sistema.
Essendo questi elementi del dominio la loro interazione è molto semplice e limitata, essa avviene infatti invocando una operazione resa disponibile dalla interfaccia attraverso il meccanismo di chiamata di funzione con trasferimento del controllo ed eventuali argomenti da parte del chiamante e valori di ritorno.
Si può interagire con la classe Exercise, e allo stesso modo con le classi WLStudent e Lab, semplicemente invocando uno dei metodi messo a disposizione dalla interfaccia.

4.6.3 Comportamento

Il comportamento è solitamente correlato alle operazioni associate ad ogni entità nel suo specifico. Riveste un ruolo molto importante nella descrizione della logica dell'applicazione poichè specifica cosa ogni entità sia in grado di fare e come la faccia.

Esercizio

Un Exercise è un oggetto atomico il cui stato non è modificabile. Il comportamento di ogni operazione di un oggetto Exercise è assimilabile al comportamento di una funzione priva di effetti collaterali che non fa altro che restituire il valore di una specifica proprietà. Si parla in pratica di tutte operazioni di tipo *get* per poter ottenere i valori di specifici attributi dell'oggetto.

Studente

Un WLStudent è un oggetto atomico contenente al proprio interno il resoconto degli esercizi superati e la loro valutazione. Tale stato è modificabile e ciò avviene attraverso l'utilizzo di due funzioni di tipo set, ovvero la funzione *add(int,float)*, che permette di aggiungere un esercizio risolto e la rispettiva valutazione, e *remove(int)* che permette di rimuovere un esercizio dal resoconto. Le restanti operazioni su WLStudent, in maniera simile ad Exercise, si comportano come funzioni prive di effetti collaterali che restituiscono il valore di uno specifico attributo dell'oggetto.

Laboratorio

Un oggetto Lab, in maniera analoga ad Exercise, è un oggetto atomico il cui stato non è modificabile. Il comportamento anche in questo caso può essere riconducibile a quello di una funzione priva di effetti collaterali che restituisce i valori di specifici attributi dell'oggetto.
Da notare come tutti e tre gli oggetti abbiano anche a disposizione un'operazione *toFile()* che viene predisposta con in mente l'idea di poter avere la possibilità di rendere persistenti gli oggetti memorizzandoli in File.

4.7 Architettura logica

Una prima architettura logica può scaturire dall'organizzazione dei sistemi fatta da Jacobson: organizzeremo il sistema in sottosistemi che si occuperanno di gestire rispettivamente l'interazione con l'utente (*boundary*), il controllo del sistema (*control*) e il dominio del problema (*entity*).

4.7.1 Struttura

Per delineare una prima struttura dell'architettura logica del problema utlizzeremo i digrammi UML package.

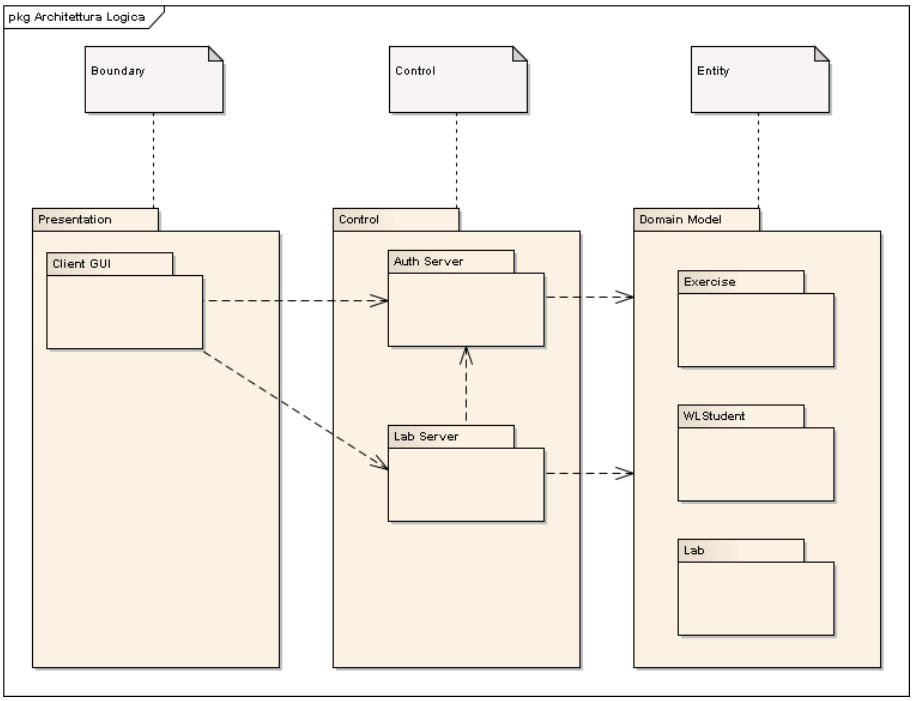

Figura 4.3: Architettura Logica.

Come già anticipato in fase di analisi del problema (Cfr 4.5) i sottosistemi in gioco nell'applicazione sono tre: una interfaccia utente che permetta agli studenti di collegarsi (denominata Client GUI) , un server per l'autenticazione (nel diagramma Auth Server) e una applicazione che rappresenti e gestisca il laboratorio virtuale (Lab Server). L'architettura logica qui

proposta mira a mettere in evidenza i rapporti e le dipendenze logiche tra queste parti del sistema e il modello del dominio appena descritto.
La suddivisione BCE(boundary, control, entity) permette di non contaminare le entità del dominio con funzionalità di una particolare applicazione o con la gestione del mondo esterno e dell'interazione con l'utente.
Dal diagramma UML in figura 4.3 si evince che la parte che realizza l'interazione con l'utente è costituita dal sottosistema *Client GUI*, mentre la parte che realizza la logica dell'applicazione, il *controllo*, è costituita dai sottosistemi *Lab Server* e *Auth Server*. La parte che realizza l'*entity* del problema è costituita dal modello del dominio visto in precedenza.
In questo schema architetturale sono state indicate alcune dipendenze; in particolare:

- il package che realizza il Controllo dell'applicazione (Lab Server e Auth Server) dipende dal modello del dominio descritto in precedenza.
- all'interno del package Controllo, Lab Server dipende da Auth Server, in quanto il laboratorio deve identificarsi presso il server di autenticazione.
- l'interfaccia utente Client GUI dipende sia da Lab Server che da Auth Server in quanto deve loggarsi prima sul server di autenticazione e quindi sul laboaratorio desiderato.

4.7.2 Interazione

Andiamo ora ad analizzare l'interazione fra le varie parti del sistema attraverso l'uso di diagrammi UML di sequenza.

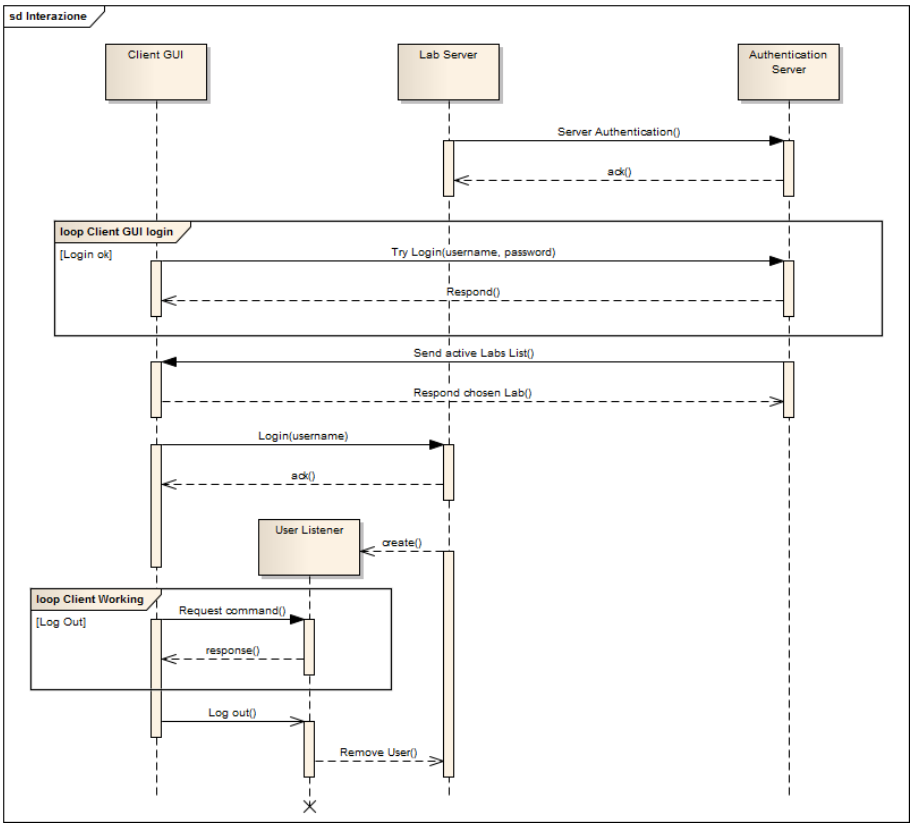

Figura 4.4: Interazione.

In Figura 4.4 sono rappresentate le interazioni tra i sottosistemi componenti l'architettura logica. Da notare come, dopo le due fasi di autenticazione (quella del laboratorio e quella dello studente col server di autenticazione), il server di autenticazione invii allo studente la lista dei laboratori attivi e lo studente invii in risposta al server di autenticazione il laboratorio virtuale prescelto.
A questo punto lo studente (Client GUI) è pronto per loggarsi al laboratorio virtuale inviando il proprio username. Il laboratorio, una volta ricevuta la richiesta dallo studente, crearà una nuova entità che avrà il compito di

dialogare con Client GUI interpretando e rispondendo alle richieste dell'utente, fino al log out dello stesso. Al termine dell'esecuzione User Listener avviserà il laboratorio che lo studente non è più attivo.

4.7.3 Comportamento

Andiamo ora ad analizzare il comportamento dei singoli sottosistemi attraverso l'utilizzo di diagrammi di stato UML.

Authentication Server

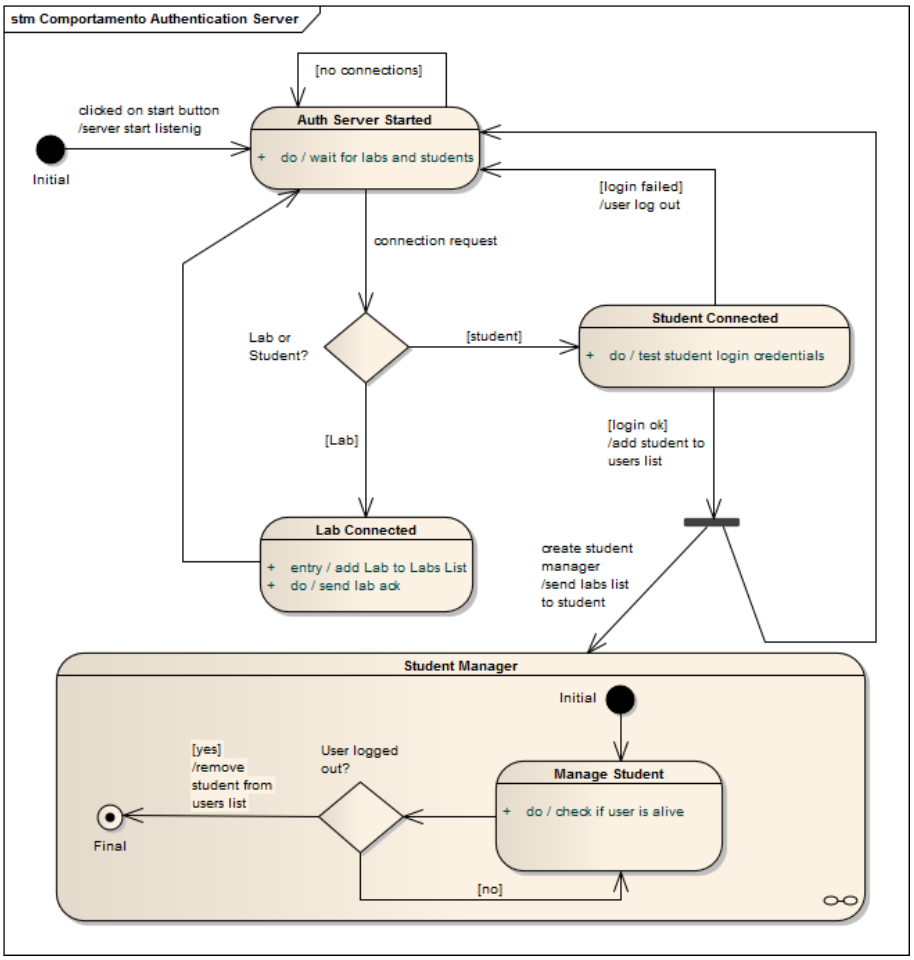

Figura 4.5: Comportamento Authentication Server.

Il diagramma di stato in Figura 4.5 descrive il comportamento del sottosistema Authentication Server, ovvero del sottosistema responsabile della autenticazione di laboratori e studenti. Una volta avviata l'applicazione e messo in ascolto il server, il sottosistema attende connessioni in arrivo.

All'arrivo di ogni connessione si distinguono due comportamenti diversi nel caso si tratti di un laboratorio o di uno studente. Nel primo caso il laboratorio virtuale viene inserito nella lista dei laboratori disponibili e il server ritorna in ascolto. Nel secondo caso si attiva una procedura di verifica delle credenziali di login dello studente (normalmente username e password) che, se superata, porta all'inserimento dell'utente nella lista degli utenti attivi. In caso contrario l'utente viene rifiutato e il sottosistema torna in ascolto. Per ogni utente che supera con successo la procedura di login viene creato un nuovo sottosistema che rimane in ascolto per verificare che la connessione con lo studente sia attiva e non ci siano problemi. Nel caso in cui ciò venga meno, l'utente viene rimosso dalla lista degli utenti attivi e il sistema torna in ascolto.

Lab Server

Il diagramma di stato in Figura 4.6 descrive il comportamento del sottosistema Lab Server ovvero del sottosistema responsabile della gestione del laboratorio virtuale. Una volta avviato Lab Server tenta di loggarsi presso il server di autenticazione per poter inviare ad esso i propri dati. Una volta che questa operazione ha avuto successo, Lab Server si mette in ascolto per le connessioni in arrivo da studenti provenienti dal server di autenticazione. All'arrivo di ogni nuovo studente viene creato un nuovo sottosistema Lab Student Manager che si occuperà di gestire le richieste derivanti dall'interazione dello studente con il laboratorio virtuale. Può essere utile soffermarsi leggermente anche in fase di analisi su di una richiesta in particolare: quella di ingresso nel laboratorio remoto. Questa richiesta, che è poi la parte fondamentale di tutta la business logic dell'applicazione, andrà trattata con molta attenzione; in fase di analisi torna utile sottolineare, come si può vedere in Figura 4.6, che il sottosistema dovrà attraversare due stati diversi: uno in cui gestire la concorrenza (solo uno studente alla volta può entrare nel laboratorio remoto) e uno in cui gestire l'esperienza di laboratorio vera e propria.

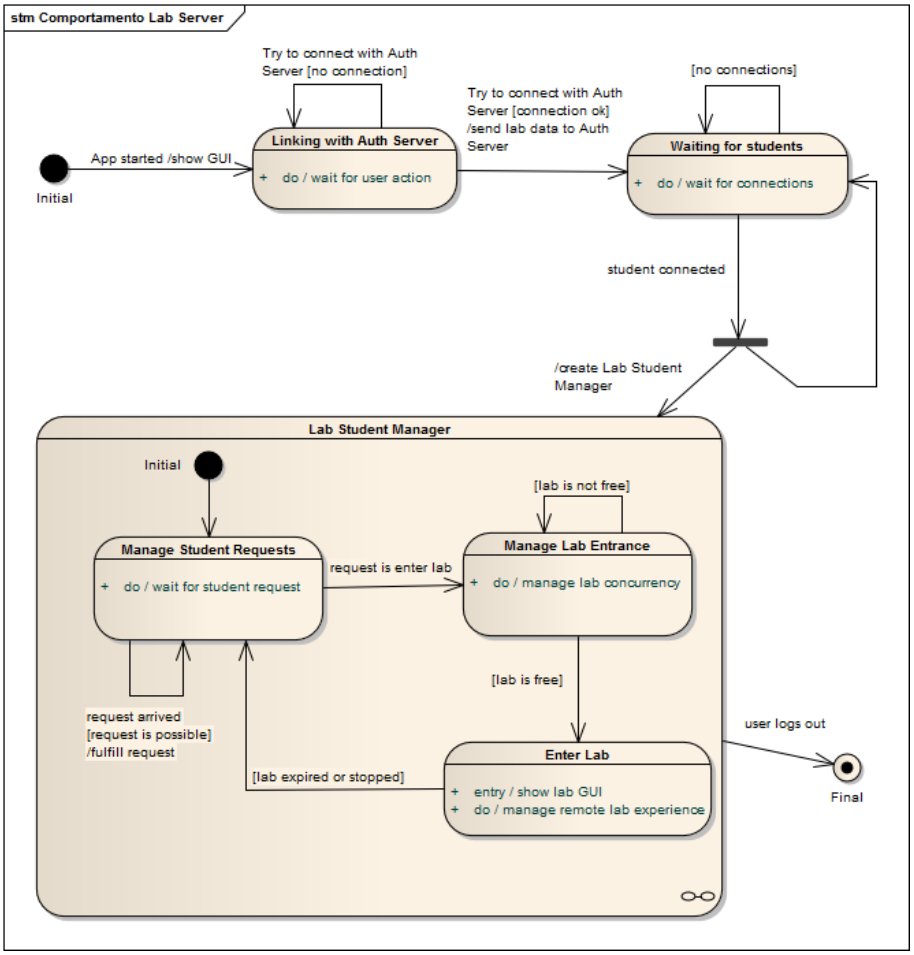

Figura 4.6: Comportamento Lab Server.

ClientGUI

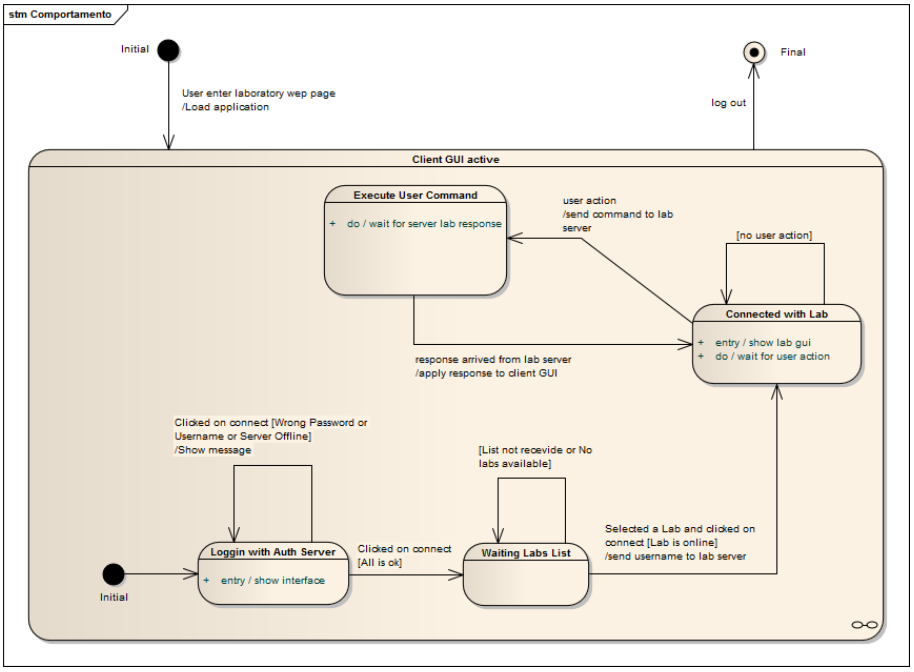

Figura 4.7: Comportamento Client GUI.

Il diagramma di stato in Figura 4.7 descrive il comportamento del sottosistema ClientGUI ovvero dell'interfaccia utente. Osservando la figura si nota che una volta che l'utente è entrato nella pagina web contente l'applicazione e che questa è stata caricata, ClientGUI entra nel suo stato attivo, attende di superare i controlli con il server di autenticazione e quindi di ricevere la lista dei laboratori disponibili. Una volta all'interno del laboratorio l'interfaccia non fa altro che attendere un qualche comando da parte dell'utente e inviarlo al Lab Server mettendosi in attesa della risposta. Questo comando potrà essere o la richiesta di un esercizio da visionare o scaricare, o un tentativo di accedere al laboratorio remoto se questo risulta libero. Essendo ora in fase di analisi di una architettura logica non vogliamo addentrarci particolarmente in dettagli applicativi che rimandiamo alla fase di progettazione in Capitolo 5.

Capitolo 5

WebLab - Progetto

5.1 Introduzione

Con la fase di progettazione ci si pone lo scopo di raffinare l'architettura logica prodotta in fase di analisi considerando tutti gli aspetti che sono stati tralasciati finora. Provvederemo quindi ad analizzare e commentare l'approfondimento dell'architettura logica, evidenziando e motivando le scelte fatte. In questa sezione individueremo anche il linguaggio di programmazione, il sistema operativo e gli strumenti ritenuti utili per la costruzione del sistema.
Adotteremo a tal proposito una progettazione di tipo bottom-up. La necessità di un tale stile di progettazione è dovuta al fatto che risulta importante e vincolante che l'utente possa avere accesso al laboratorio remoto direttamente dall'interno di un pagina web. Questa condizione pone già all'inizio della fase di progettazione il problema della scelta della piattaforma e del linguaggi di programmazione da utilizzare. Uno stile di tipo bottom-up presenta un problema importante a livello progettuale, poichè una volta scelta la piattaforma questa finisce col porre vincoli e a bloccare alcune decisioni di progetto in base alle feature e alle possibilità che tale piattaforma mette a disposizione. Per questo, nonostante risulti in questo caso importante scegliere fin dall'inizio la giusta piattaforma per realizzare una applicazione che soddisfi i requisiti richiesti, si cercherà comunque nella fase di progettazione di ragionare in termini di astrazione senza focalizzarsi direttamente sulle possibilità offerte dalla piattaforma implementativa.

5.2 Scelte

Le scelte riguardano in particolare la scelta del tipo di modello di comunicazione (essendo l'applicazione una applicazione distribuita)e del linguaggio di programmazione e piattaforma operativa. Inoltre verranno illustrati anche i design pattern individuati all'interno della progettazione e il loro ruolo nel funzionamento dell'applicazione.

Il modello di comunicazione tra i sottosistemi sarà un modello del tipo cliente/servitore. Nel dettaglio tale modello C/S avrà le seguenti caratteristiche:

- sarà sincrono e bloccante: il servitore resterà in attesa di una richiesta da parte del cliente e il cliente attenderà la risposta da parte del servitore. In questo modo il server sarà sempre in attesa delle richieste da parte del client e non dovrà attivarsi ogni volta.
- sarà un modello uno a molti: un servitore per molti clienti. Questo vale sia per Auth Server che per Lab Server.
- l'interazione sarà connection-oriented, con la creazione di un canale di comunicazione virtuale di tipo TCP. Questo per sfruttare l'affidabilità e la capacità di mantenere l'ordine dei messaggi di TCP.
- il servitore sarà stateful, ovvero con stato interno relativo all'utente che condizionerà la risposta ad ogni richiesta del cliente. Questo solleva il cliente da ulteriori responsabilità, il cliente invierà semplicemente un messaggio e attenderà la risposta. Il servitore sarà naturalmente più complicato e occuperà più risorse, ma avremo il vantaggio di un minor costo delle operazioni in banda, di maggiore sicurezza e possibilità di ripristino. Lo stato mantenuto dal server (in particolare quello del laboratorio) sarà di tipo permanente: il server dovrà tenere traccia di tutte le interazioni che i clienti avranno fatto (accessi, esercizi risolti e valutazioni) e mantenerla per un tempo indefinito.
- il servitore sarà un servitore di tipo concorrente: dovrà gestire molte richieste di servizio concorrentemente. Questo è essere reso possibile

attraverso la generazione di un processo leggero (thread) all'arrivo di ogni richiesta di servizio.

Con queste premesse la progettazione del cliente sarà molto più semplice, visto che dovrà semplicemente mostrare l'interfaccia ed inviare richieste al servitore aspettando la risposta. La progettazione del server richiederà invece molta più attenzione in quanto il server dovrà gestire più clienti contemporaneamente, ascoltare le loro richieste, modificare il loro stato, mantenere il loro stato e inviare le risposte. Oltre a questo il server (in questo caso i due server: Auth Server e Lab Server) dovranno anche gestire la parte di controllo dell'applicazione, in particolare l'accesso e l'autenticazione per Auth Server e la parte del laboratorio remoto e degli esercizi per Lab Server. Il client sarà quindi un client leggero (thin client) mentre il server si accollerà gran parte di lavoro.
Infine come linguaggio di programmazione e piattaforma operativa si è pensato di fare riferimento a Java e a Java2SE, per garantire piena portabilità su ogni sistema operativo e per non dover affrontare il problema di gestire scambi di messaggi tra piattaforme diverse: i due server saranno infatti realizzati in Java così come il client, evitando così di complicare ulteriormente il problema.

5.3 Architettura

Andremo ora a raffinare e ad arricchire l'architettura logica presentata in fase di analisi. Analizzeremo quindi la struttura dei singoli sottosistemi e relativi componenti.

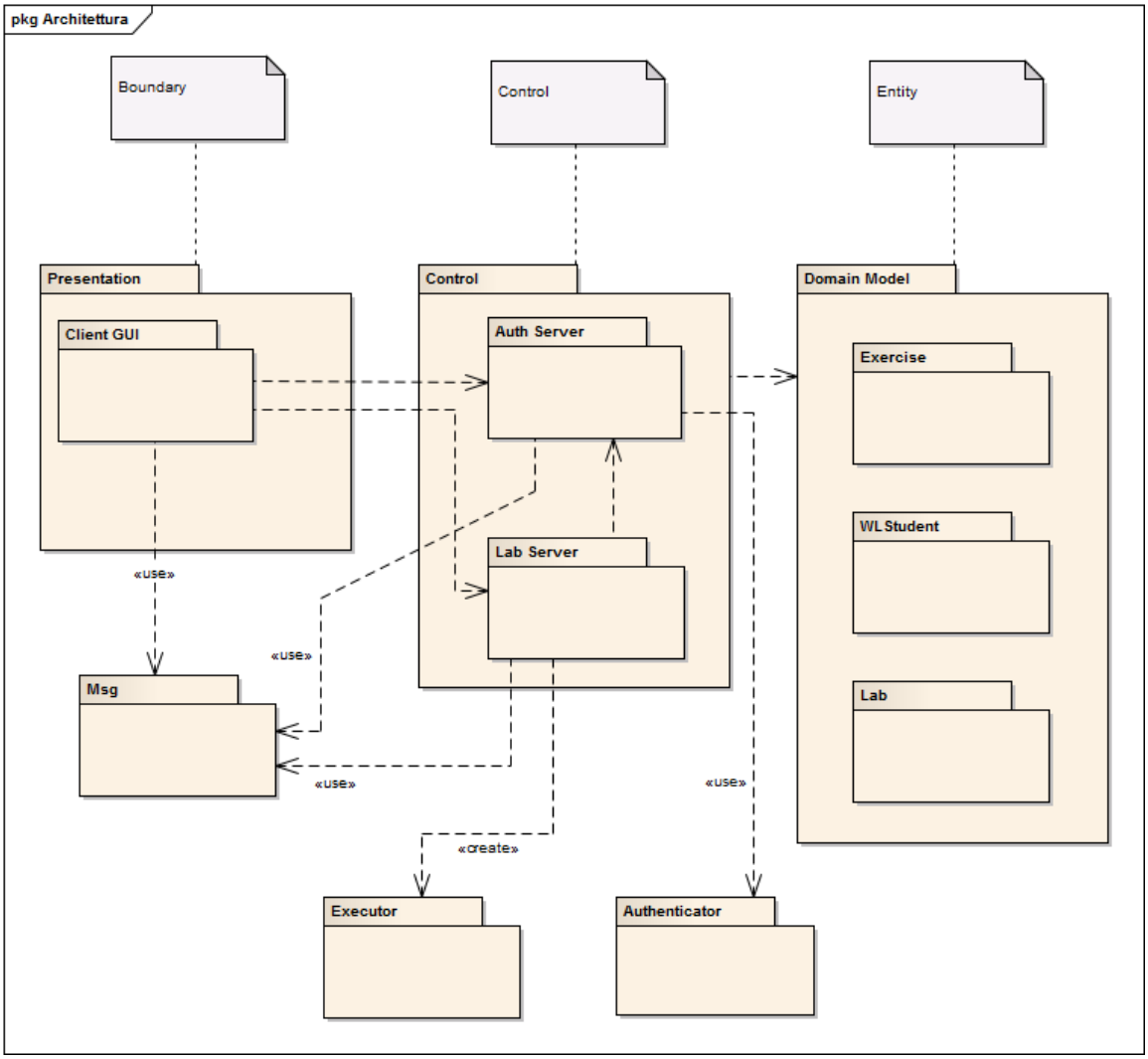

Figura 5.1: Architettura WebLab.

L'architettura logica presente nel Capitolo 4 in Figura 4.3 viene espansa e dettagliata attraverso l'introduzione di alcuni sottosistemi che si affiancano ai principali nella logica dell'applicazione. In particolare si sono fatte

alcune scelte a livello progettuale con l'introduzione dei nuovi sottosistemi che andremo ora a motivare.

5.3.1 Msg

Si è scelto di introdurre il sottosistema Msg per standardizzare le comunicazioni fra client e Lab Server. Ogni richiesta del client verso il server e ogni risposta del server verso il client potrà contenere un comando, un parametro e un oggetto in allegato. Si è deciso di incapsulare il tutto in una struttura che verrà poi creata e spedita e, una volta ricevuta, aperta e decomposta. Si è preso spunto per questo dal Pattern Message [4] con alcune modifiche.

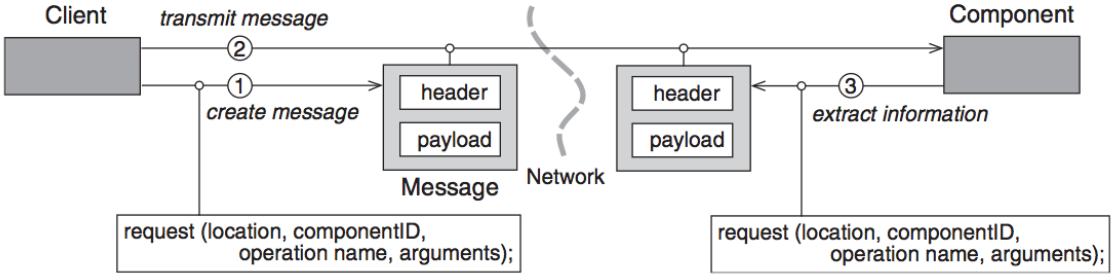

Figura 5.2: Message Pattern.

In particolare, essendo il destinatario noto sia al client che al server (la comunicazione è infatti connection-oriented) non sarà necessario introdurre nell'header del messaggio il destinatario e il mittente. Nel nostro caso l'header del messaggio conterrà il comando richiesto e eventuali paramentri, mentre il payload conterrà oggetti in allegato. Questa soluzione permette di gestire tutte le interazioni richieste fra client e server nel nostro progetto.

Msg - Struttura

La struttura di Msg è molto semplice, è costituto da tre attributi e dalle relative operazioni di recupero.

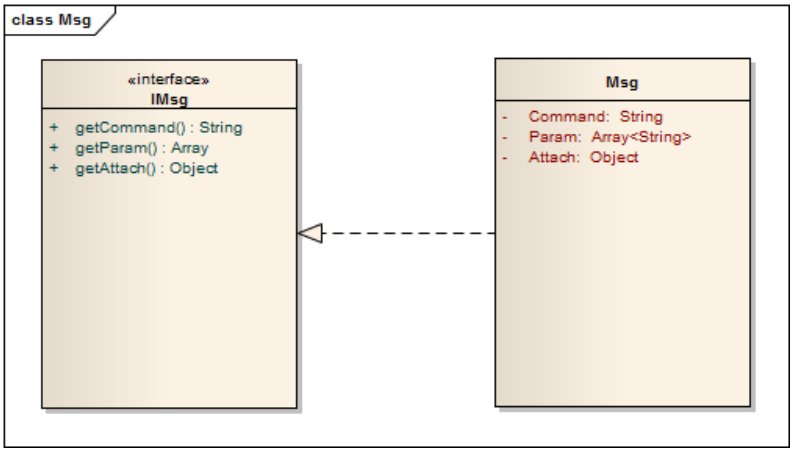

Figura 5.3: Msg - Struttura.

Msg - Interazione

L'interazione con Msg avviene mediante l'invocazione di una operazione resa disponibile dall'interfaccia. Si tratta di una chiamata procedura/funzione con trasferimento del controllo e restituzione di valore.

Msg - Comportamento

Msg è un oggetto composto da tre attributi e dalle relative operazioni di get. All'atto della sua costruzione Msg sarà riempito in ogni suo campo e preparato per essere inviato attraverso il canale di comunicazione. A tale scopo Msg dovrà essere un oggetto serializzabile[1].

5.3.2 Authenticator

Nella gestione dell'accesso dello studente all'infrastruttura WebLab verranno verificate le credenziali dell'utente. Solo gli utenti registrati e con le giuste credenziali potranno accedere alla piattaforma. Per fare questo si rende necessario prevedere una fase di controllo dell'accesso in fase di

[1]Serializzazione è un processo per salvare un oggetto in un supporto di memorizzazione lineare (ad esempio, un file o un'area di memoria), o per trasmetterlo su una connessione di rete.

progettazione. Auth Server avrà quindi una dipendenza dal sottosistema Authenticator. Si è deciso di sfruttare il Pattern Authorization [4].

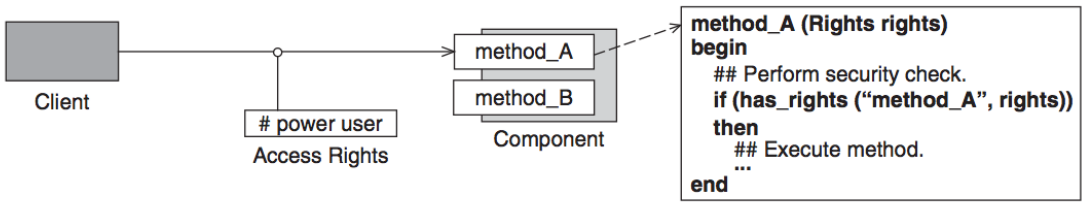

Figura 5.4: Authorization Pattern.

In particolare la fase di autorizzazione avverrà all'inizio, all'atto della connessione fra Client GUI e AuthServer, prima che il server per l'autenticazione invii all'utente la lista dei laboratori attivi.

Authenticator - Struttura

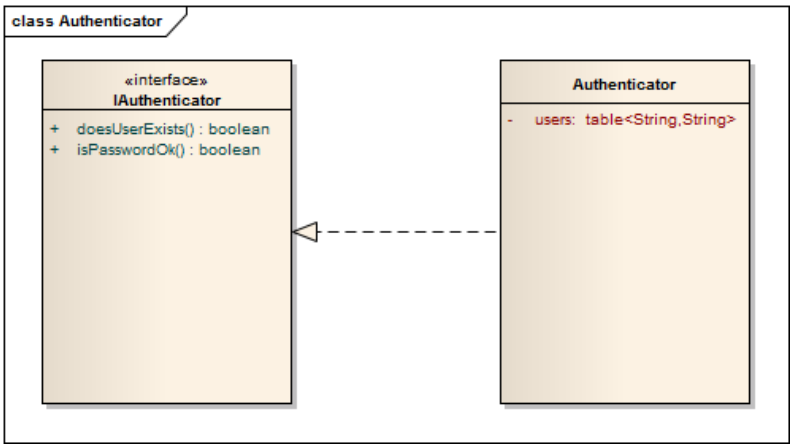

Figura 5.5: Authenticator - Struttura.

La struttura di Authenticator è molto semplice, importanti sono le due operazioni richieste dall'interfaccia che gestiscono l'esistenza dell'utente e la validità della password.

Authenticator - Interazione

Anche in questo caso l'interazione con Authenticator avviene mediante l'invocazione di una operazione resa disponibile dall'interfaccia. Si tratta di una chiamata procedura/funzione con trasferimento del controllo e restituzione di valore.

Authenticator - Comportamento

Authenticator verrà inizializzato e messo in esecuzione insieme a Auth Server. Auth Server richiederà quando necessario ad Authenticator di effettuare un controllo su un utente che richiede la connessione invocando uno dei suoi metodi. Authenticator a sua volta verificherà se l'utente possiede le credenziali giuste; per farlo si può appoggiare ad una lista interna di utenti ammessi o anche a database o strutture dati esterne.

5.3.3 Executor

Una volta che l'utente riesce ad avere accesso e controllo nel laboratorio reale e a caricare la propria soluzione, il sistema deve mettere in atto tutti i meccanismi per poter mettere in esecuzione ciò che lo studente ha caricato. Il sottosistema software che gestisce e muove nella realtà il laboratorio non è oggetto di questa tesi, ma il sistema WebLab deve essere progettato tenendo anche in considerazione che la parte mancante verrà prossimamente costruita e che la progettazione di tale parte deve essere resa il più semplice possibile in modo da permettere una perfetta integrazione col sistema software WebLab. Il sottosistema Executor gioca esattamente questo ruolo, viene istanziato e messo in esecuzione ogni volta che lo studente carica la propria soluzione e si aspetta di vedere eseguito il proprio esercizio.
Executor dovrà svolgere i seguenti compiti:

- essere in grado di muovere ogni robot presente nel laboratorio.
- effettuare una fase di setup fisico del laboratorio (ad esempio spostare alcuni robot in certe posizioni o controllare la carica delle batterie)

- leggere la soluzione proposta dallo studente e verificarne la correttezza sintattica
- mettere in esecuzione la soluzione dell'esercizio proposta dall'utente facendo fisicamente muovere i robot

Per progettare Executor si sono presi in considerazione due Pattern in particolare, sempre presenti in [4]: Activator Pattern e Lifecycle Callback Pattern.

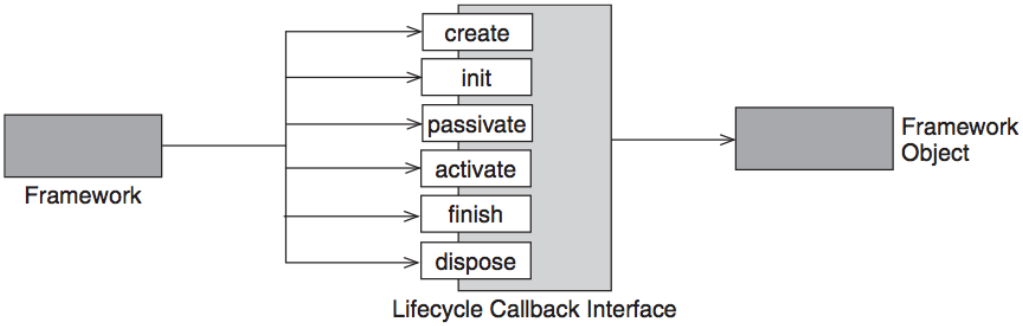

Figura 5.6: Lifecycle Callback Pattern.

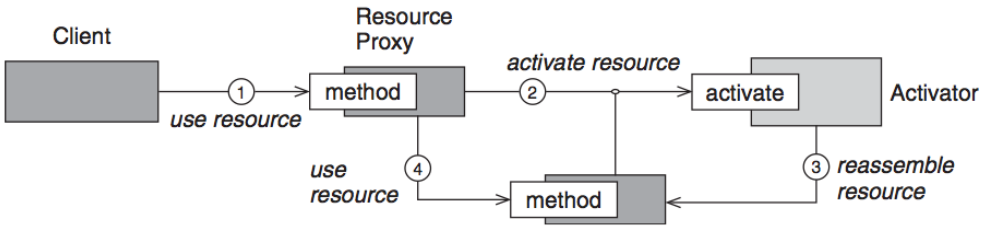

Figura 5.7: Activator Pattern.

Il Pattern Lifecycle viene utilizzato nella progettazione della struttura di Executor, che sarà costruito in maniera che l'applicazione che ne usufrui-

sce (in questo caso Lab Server che ha una dipendenza use da Executor) possa controllare il suo flusso di esecuzione(*lifecycle* appunto) attraverso dei metodi callback. Lab Server utilizza una interfaccia con metodi appropriati, implementati su Executor per soddisfare i compiti elencati sopra e per controllare il ciclo di esecuzione del sottosistema senza conoscerne i dettagli interni.
Questa parte di gestione dell'esecuzione di Executor viene ancora più disaccoppiata dal sottosistema Lab Server attraverso l'uso del Pattern Activator (Figura 5.7): una sottoparte di Lab Server si occuperà di attivare, disattivare e gestire il ciclo di esecuzione di Executor in maniera del tutto trasparente al server ed al client che ha fatto richiesta di accesso.

Executor - Struttura

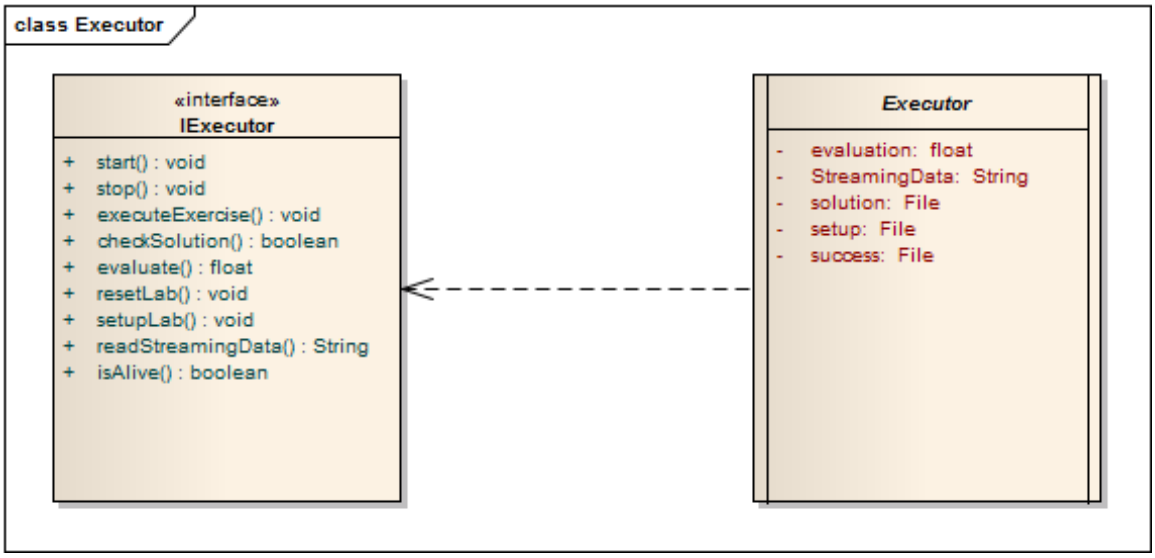

Figura 5.8: Executor - Struttura.

La struttura di Executor rispecchia la necessità di controllare il flusso della sua esecuzione attraverso una serie di metodi di callback che permettono al chiamante di gestire l'esecuzione del sottosistema stesso. Da notare che Executor viene immaginato come una classe astratta e attiva, ad esempio un Thread, quindi con un proprio lifecycle. L'eventuale implementatore

di Executor dovrà rispettare l'interfaccia e la classe astratta implementando i metodi richiesti che verranno poi utilizzati dall'Activator all'interno di Lab Server per manipolare il laboratorio reale. Da notare che Executor ha come attributi anche tre File : il file soluzione caricato dall'utente e altri due file, uno per il setup dell'esercizio e uno contenenti le condizioni di successo per la valutazione. Questi ultimi due sono da intendersi come proprietà intrinseche dell'esercizio e fanno parte della sua definizione, rappresentano in particolare le condizioni iniziali richieste per effettuare l'esercitazione (ad esempio la posizione di ostacoli o la posizione di alcuni robot in particolare) e quelle finali che determinano il superamento o meno dell'esercizio stesso.

Executor - Interazione

Alla ricezione del segnale da parte di LabServer che l'utente ha richiesto l'esecuzione di un esercizio, Activator si attiva, crea e manda in esecuzione una istanza di Executor. Dopo aver controllato che il file di soluzione sia conforme ai requisiti richiesti, Activator comanda, usando i metodi messi a disposizione dalla sua interfaccia, Executor durante tutta l'esecuzione dell'esercizio fino ad ottenere una valutazione e conclude con fermare l'esecuzione di Executor.

Executor - Comportamento

Il comportamento di Executor è dettato dai suoi metodi, nell'ordine di esecuzione:

- Start() : Executor viene messo in esecuzione.
- evaluateSolution() : viene valutata la soluzione e se ritenuta valida viene mandata in esecuzione.
- setupLab() : utilizzando un riferimento al setup dell'esercizio il laboratorio viene preparato all'esecuzione.
- executeExercise() : questo metodo contiene tutta la logica di esecuzione fisica dell'esercizio.

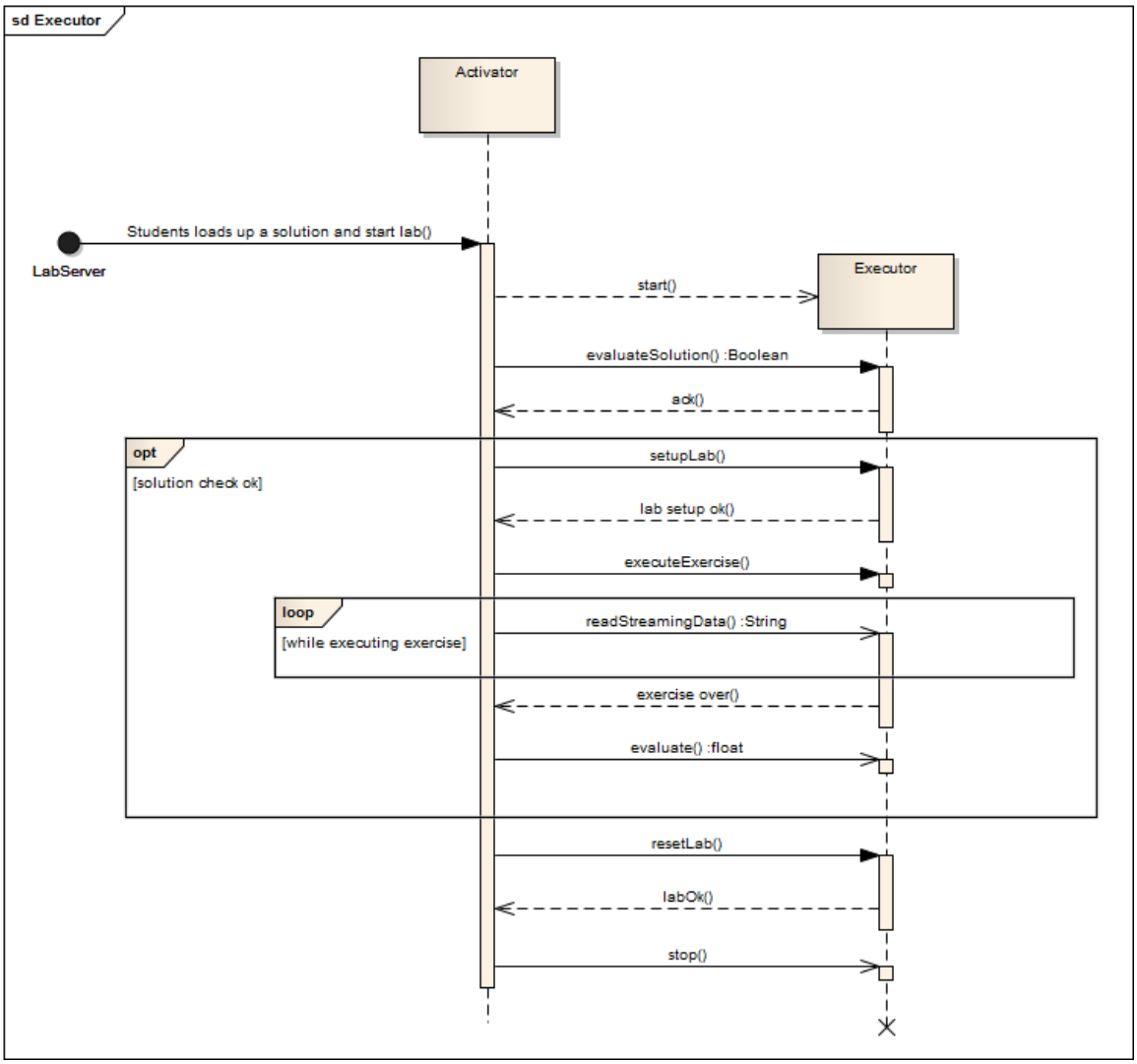

Figura 5.9: Executor - Interazione.

- readStreamingData() : permette ad Activator e quindi a LabServer di leggere una stringa di dati che Executor rende sempre disponibile durante l'esecuzione dell'esercizio.
- evaluate() : stima una valutazione numerica (1-10) dell'esecuzione dell'esercizio.
- resetLab() : a fine esercizio (o nel caso di problemi e di stop dell'esecuzione) questo metodo riporta il laboratorio reale nelle condizioni iniziali.
- stop() : Executor viene fermato.

L'ordine di esecuzione dei metodi viene gestito da Activator. L'implementazione dei metodi di Executor condizionerà il comportamento del laboratorio reale e sarà differente per ogni laboratorio remoto. L'importante vincolo è che l'implementazione rispetti l'interfaccia definita da IExecutor.

Dopo aver esaminato le scelte e i nuovi sottosistemi introdotti, andremo ora ad approfondire le caratteristiche dei quattro sottosistemi principali(ClientGUI, LabServer, AuthServer ed il modello del dominio), definendoli ed esaminandoli sempre più nel dettaglio.

5.3.4 Struttura

Dominio

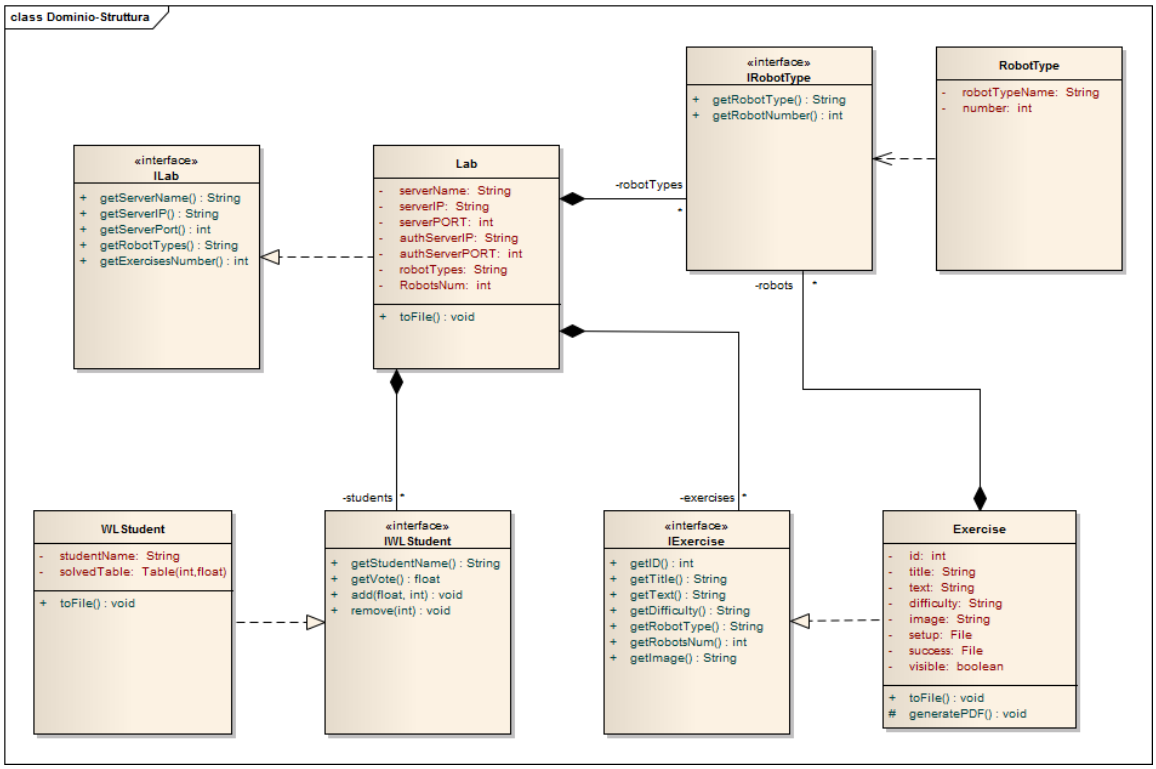

Figura 5.10: Dominio - Struttura.

Il modello del domino è stato leggermente ampliato rispetto alla fase di analisi con l'introduzione dell'entità RobotType aggiunta per modellare il tipo e il numero dei robot sia presenti in un laboratorio, sia necessari per un esercizio. Sono stati inoltre aggiunti gli attributi *success* e *setup* di tipo File in Exercise: questi attributi modellano la necessità dell'esercizio di imporre delle condizioni di inizio e di successo per le esercitazioni. Queste condizioni sono state modellate come generici File (ad esempio di testo) in modo da lasciare ampia libertà di scelta al progettista della parte Executor. Sono stati infine aggiunti gli attributi authServerIP e authServerPORT all'entità Lab per modellare la necessità del laboratorio virtuale di es-

sere a conoscenza del nome e della porta di connessione del server di autenticazione.

Auth Server

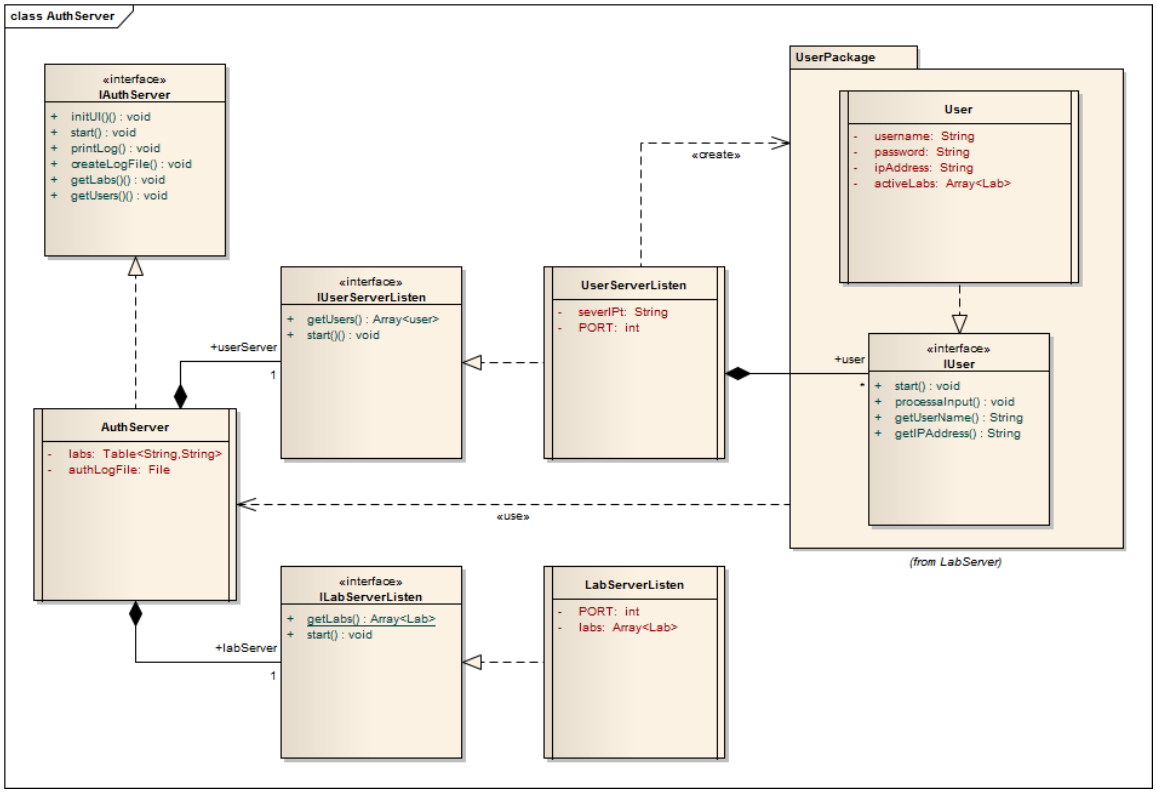

Figura 5.11: Auth Server - Struttura.

Analizzando la struttura di Auth Server si nota come il sottosistema sia composto a sua volta da varie sottoparti ognuna responsabile di parte dell'interazione che Auth Server deve avere con il mondo esterno. Auth Server, in quanto server che gestisce l'autenticazione di studenti e laboratori, dovrà lavorare in maniera diversa e su canali diversi con entrambi. Il compito di questo è affidato a due sottosistemi: UserServerListen e LabServerListen, leggermente diversi tra loro. In particolare UserServerListen instaura un'interazione che va al di la del semplice invio di un singolo messaggio ma può protrarsi nel tempo (l'autenticazione dell'utente può fallire

per vari motivi e inoltre l'utente deve ricevere da Auth Server anche la lista dei laboratori attivi), per questo viene generato e utilizzato un nuovo componente User che si occupa di tenere attiva la connessione fra Auth Server e l'utente. Nel progettare il componente User si è sfruttato il Pattern Server Request Handler [4] (Figura. 5.12), in questo modo avremo un componente che si occuperà di gestire sotto ogni aspetto la connessione con l'utente.

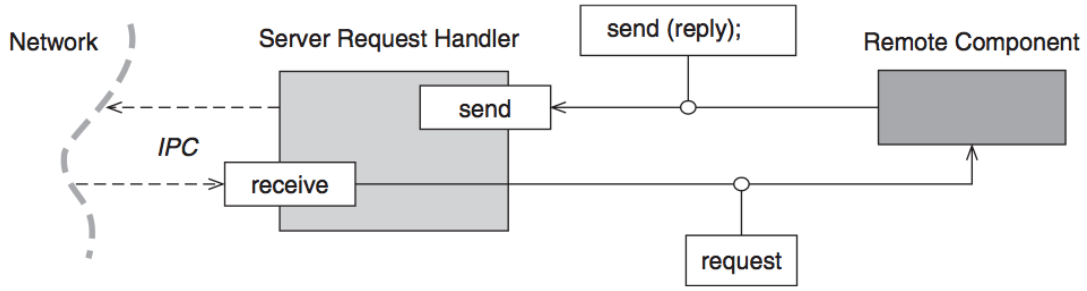

Figura 5.12: Server Request Handler Pattern.

Lab Server

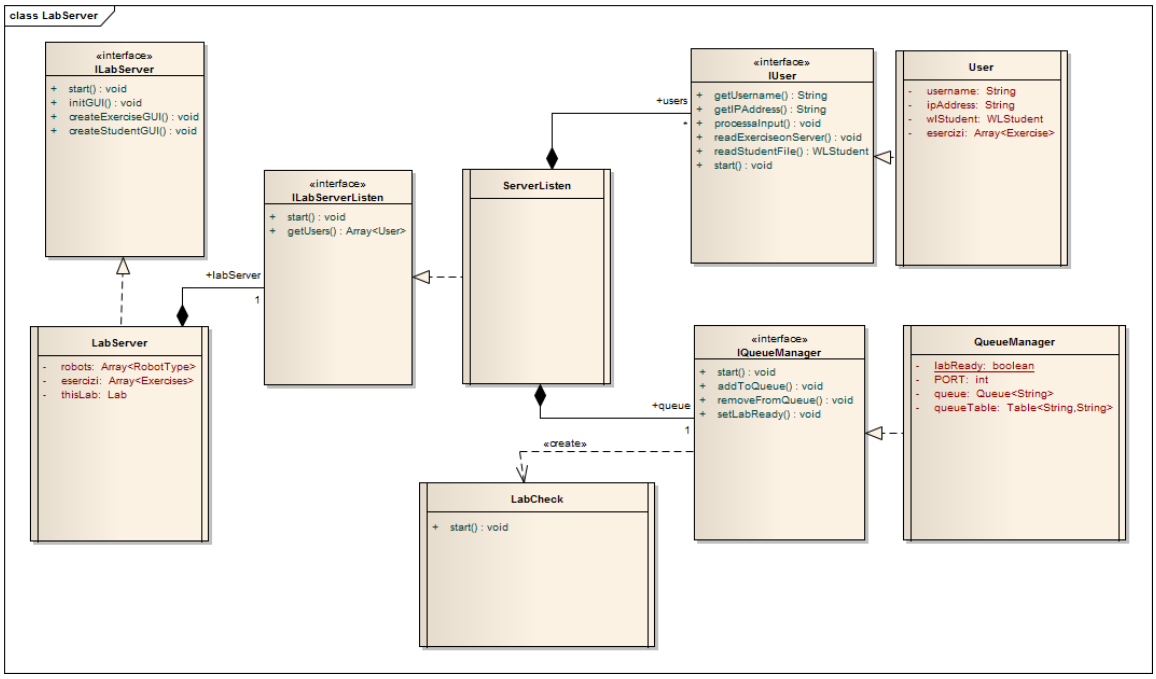

Figura 5.13: Lab Server - Struttura.

La struttura di Lab Server ricalca in parte quella di Auth Server anche se con alcune importanti differenze. In particolare il meccanismo che si occupa dell'interazione con gli utenti è strutturalmente identico a quello di Auth Server e segue anche lui il Server Request Handler Patter (Figura. 5.12). La struttura si diversifica invece per l'aggiunta di un componente responsabile di gestire l'accesso al laboratorio reale.

Si è scelto a tal proposito di modellare l'accesso al laboratorio reale come una coda FIFO (first-in-first-out) gestita dal componente QueueManager. Da notare che il componente QueueManager deve anche essere in grado di avvisare l'utente in cima alla coda quando il laboratorio è libero e disponibile. Per far questo crea e fa uso del componente LabCheck, modellato seguendo il Pattern Client Request Handler [4] (Figura. 5.14).

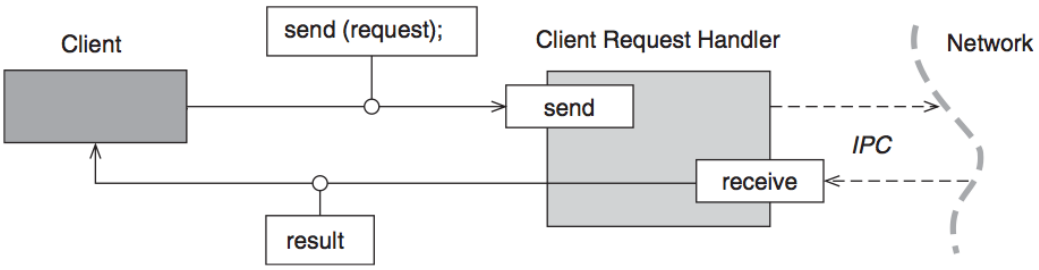

Figura 5.14: Client Request Handler Pattern.

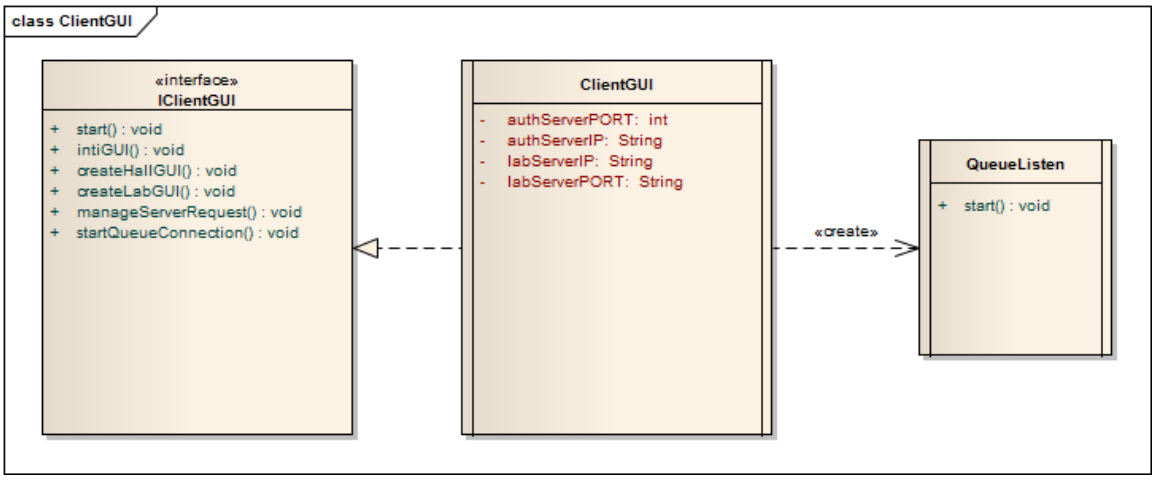

Figura 5.15: Client GUI - Struttura.

Client GUI

La struttura di Client GUI è molto semplice, mantenendo in questo modo una organizzazione di tipo *thin-client*, lasciando il carico computazionale e la gestione dell'applicazione tutta al lato server del sistema. Da notare anche qui il componente QueueListen che si rifà al Pattern Server Request Handle (Figura 5.12) per gestire le comunicazioni con il componente LabCheck di LabServer.
Analizzeremo ora queste interazioni e i comportamenti dei singoli sottosistema entrando nel dettaglio del funzionamento di ognuno dei componenti presenti in WebLab.

5.3.5 Interazione

Analizziamo ora una ad una le interazioni fra i vari sottosistemi.

Interazione Lab Server - Auth Server

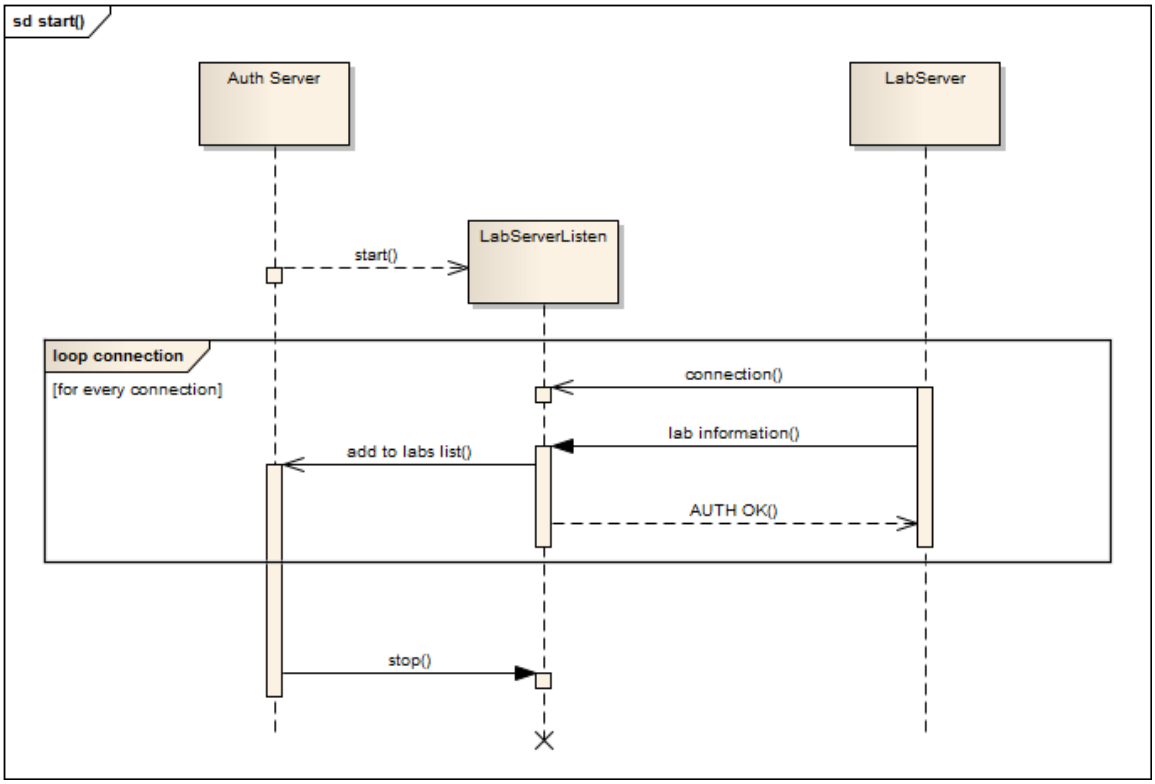

Figura 5.16: Interazione Lab Server - Auth Server.

L'interazione di Lab Server con Auth Server riguarda lo scambio di messaggi necessari ad identificare il laboratorio virtuale presso il server di autenticazione. All'avvio del Auth Server viene creato e eseguito il componente LabServerListen che attende le richieste di connessione dei laboratori. Dopo aver stabilito una connessione LabServer invia a LabServerListen le informazioni riguardanti se stesso (nome,indirizzo,porta di comunicazione). LabServerListen comunica le informazioni a Lab Server che aggiunge il laboratorio ad una lista ed infine comunica l'avvenuto inserimento inviando un messaggio di successo a LabServer. Terminata la transazione

LabServerListen si rimette in ascolto per nuovi laboratori che richiedano connessione.

Interazione Client GUI - Auth Server

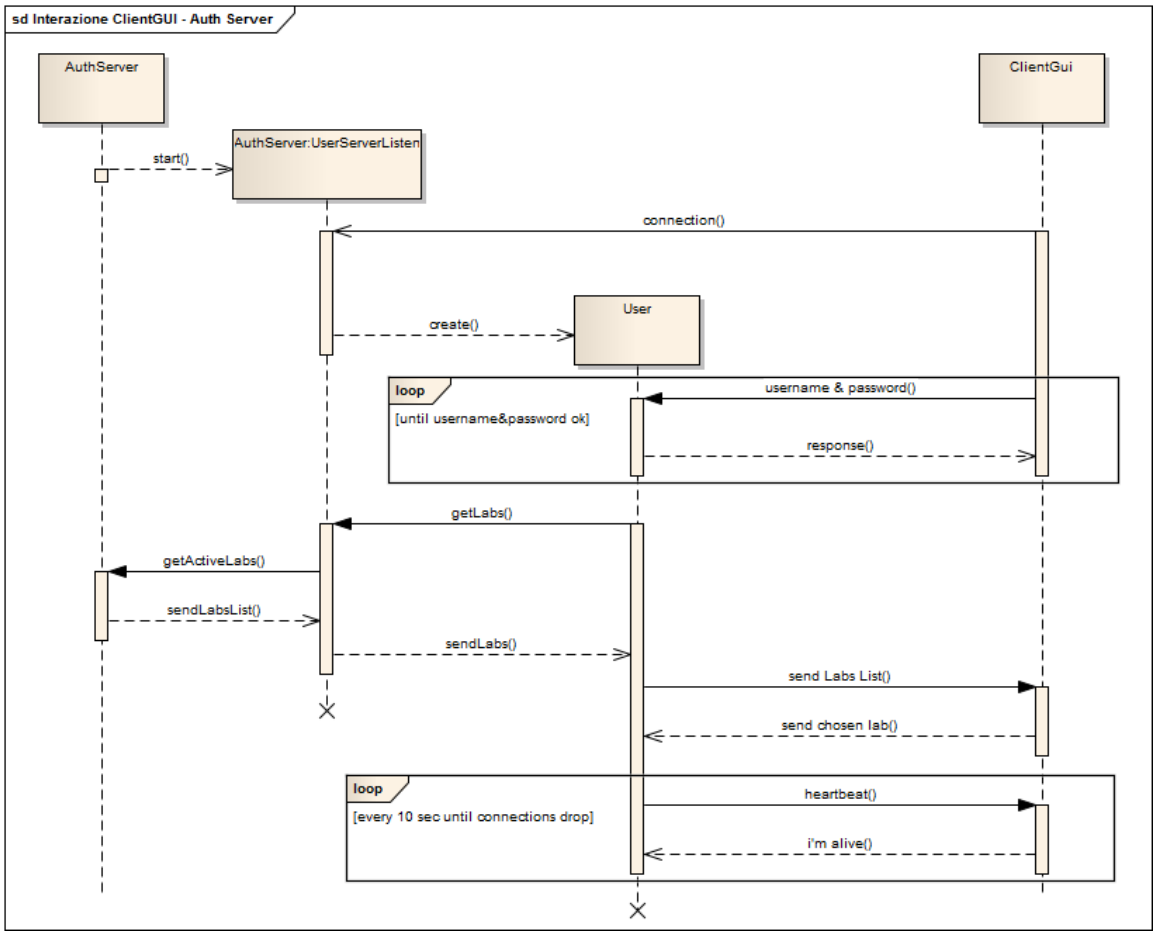

Figura 5.17: Interazione Client GUI - Auth Server.

In Figura 5.17 è rappresentato lo scambio di messaggi che avviene fra ClientGUI e Auth Server al momento del login di uno studente. All'avvio di AuthServer viene ed eseguito il componente UserServerListen che attende le richieste di connessione degli utenti. Per ogni richiesta di connessione accettata viene creato un componente User che si occupa di gestire la conversazione con ClientGUI. In questo modo è possibile sviluppare un sistema

Client-Server concorrente, lasciando libero UserServerListen di accogliere nuove richieste di connessione.
Una volta stabilita la connessione ClientGUI invia ad User il proprio username e password. User, attraverso l'uso del sottosistema Authenticator, valuta la correttezza dei dati e risponde a Client GUI. Terminata con successo la fase di autenticazione User richiede a UserServerListen che richiede a Auth Server la lista dei laboratori che si sono identificati finora con Auth Server e sono attivi. Ricevuta questa lista, User la invia a ClientGUI il quale risponde comunicando a User il laboratorio scelto.
User ha ora esaurito il suo compito, ma continua comunque ad inviare un heartbeat a ClientGUI ad intervalli regolari per controllare che il client sia ancora connesso.

Interazione Client GUI - Lab Server

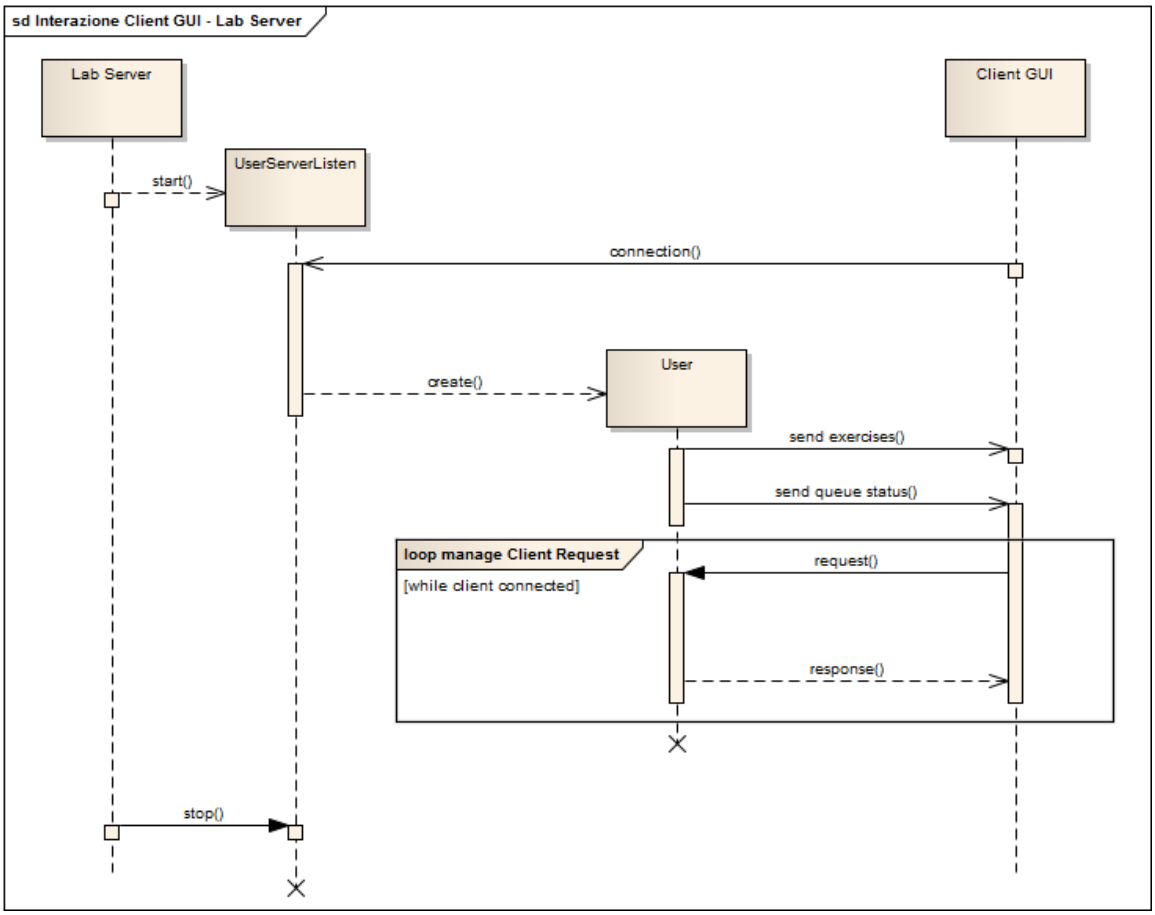

Figura 5.18: Interazione Client GUI - Lab Server.

In Figura 5.18 è rappresentato lo scambio di messaggi fra ClientGUI e Lab Server. Una volta avviato, LabServer crea e mette in esecuzione User-ServerListen che si mette in attesa di connessioni provenienti dagli studenti. Similmente a quanto accade per l'interazione fra ClientGUI e Auth Server, anche qui per ogni connessione entrante viene creato un componente User che si occupa di gestire tutte le comunicazioni fra ClientGUI e il server. In questo modo anche Lab Server può essere un server concorrente. Appena creato User trasmette a ClientGUI la lista degli esercizi e lo stato della coda che gestisce il laboratorio. Terminata questa fase di setup, finchè il

client rimane connesso, User rimane in ascolto e accetta e gestisce tutte le richieste che arrivano da ClientGUI.

Interazione Client GUI - Lab Server QueueManager

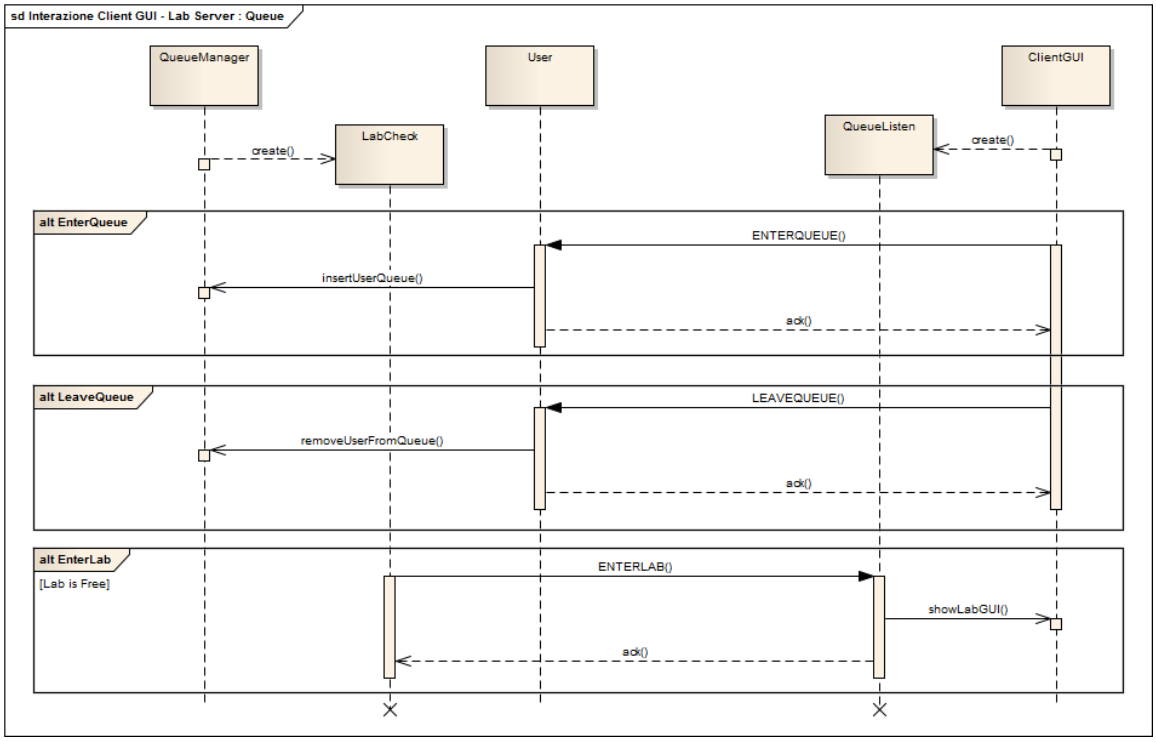

Figura 5.19: Interazione Client GUI - Lab Server QueueManager.

In Figura 5.19 abbiamo un esempio di tali richieste; un esempio molto significativo perchè queste richieste coinvolgono altri componenti, ovvero QueueManager e LabCheck. Questi componenti regolano l'accesso al laboratorio reale attraverso l'uso di una coda di priorità. QueueManager viene creato e avviato insieme a UserServerListen. QueueManager a sua volta crea e attiva un ulteriore processo responsabile di monitorare lo stato del laboratorio. A livello client, invece, il componente QueueListen viene creato e avviato direttamente da ClientGUI e ha la responsabilità di rimanere in ascolto per eventuali messaggi riguardanti lo stato del laboratorio.
In particolare i messaggi qui evidenziati sono tre:

- ENTERQUEUE: ClientGUI fa richiesta a Lab Server per entrare nella coda di attesa. User invia il messagio a QueueManager che gestisce la coda.
- LEAVEQUEUE: ClientGUI fa richiesta a Lab Server di abbandonare la coda di attesa. User invia la richiesta a QueueManager.
- ENTERLAB: il laboratorio è libero. LabCheck preleva il primo utente in coda e invia a QueueListen un messaggio ENTERLAB per avvisarlo che può entrare nel laboratorio.

5.3.6 Comportamento

Andiamo ora ad approfondire il comportamento dei vari sottosistemi visto in fase di analisi.

Auth Server

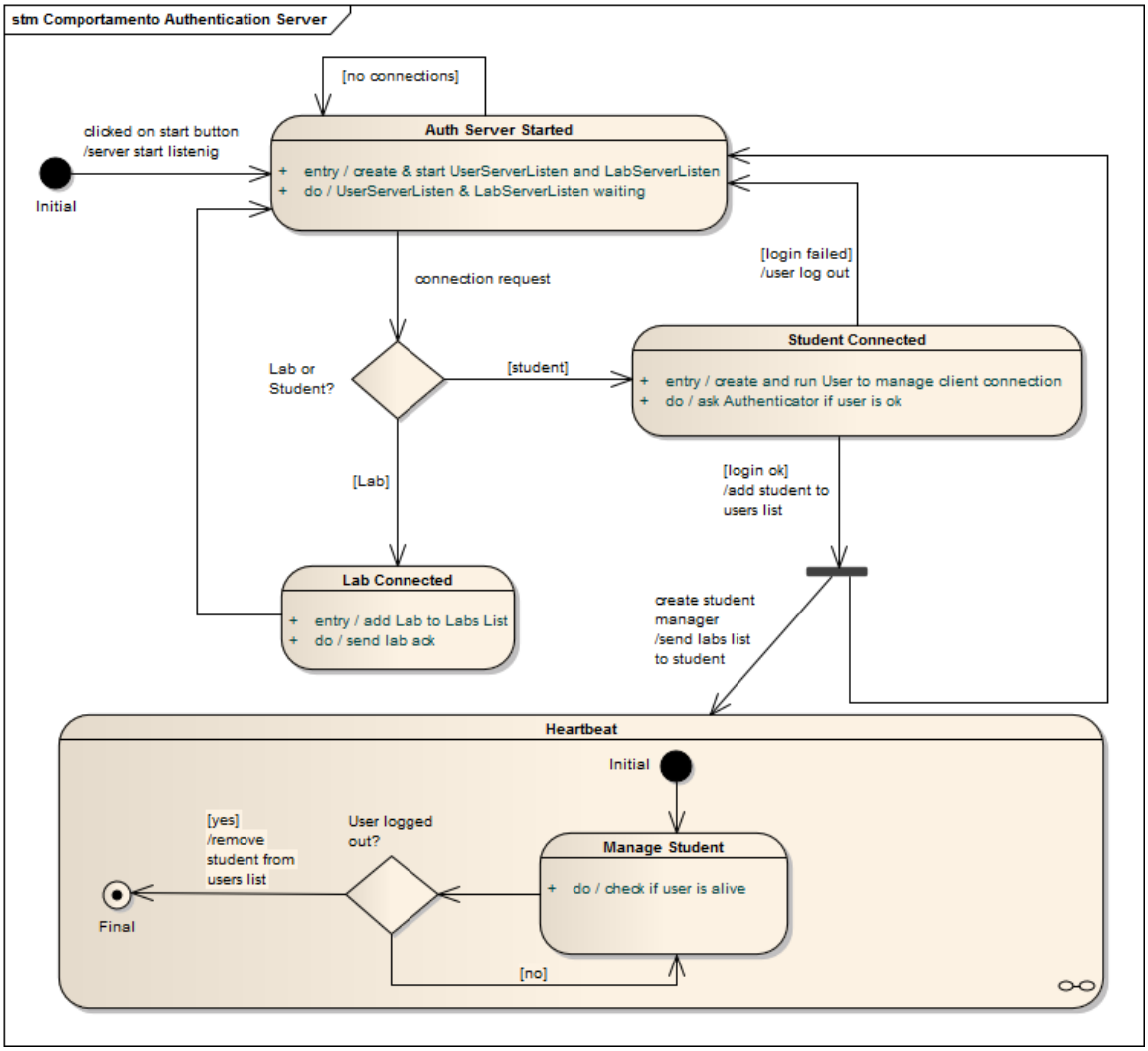

Figura 5.20: Comportamento Auth Server.

Il comportamento rappresentato dal diagramma UML in Figura 5.20 non si discosta molto da quello visto in Figura 4.5. L'unica differenza è

data dall'introduzione dei nomi dei sottosistemi responsabili dell'autenticazione (Authenticator) e della gestione delle connessioni (LabServerListen,UserServerListen e User) all'interno dei rispettivi eventi.

Lab Server

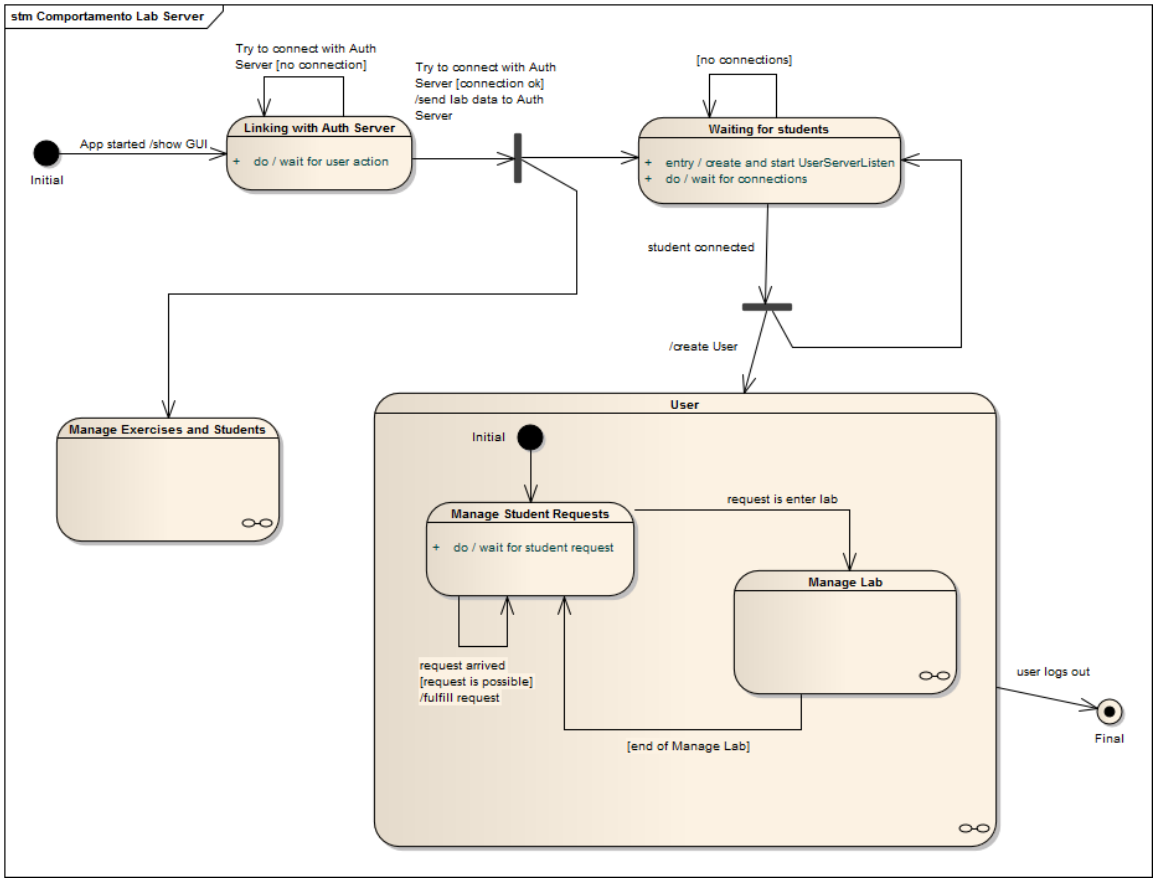

Figura 5.21: Comportamento Lab Server.

Il comportamento rappresentato dal diagramma UML in Figura 5.21 si discosta leggermente da quello visto in Figura 4.6 per l'inserimento di due State Machine che rappresentano il comportamento di due parti particolari del sistema: la parte che gestisce esercizi e studenti e la parte che gestisce l'ingresso e lo sfruttamento del laboratorio remoto. Andremo

ora ad analizzare nel dettaglio, con altri due diagrammi UML, questi due sottostati.

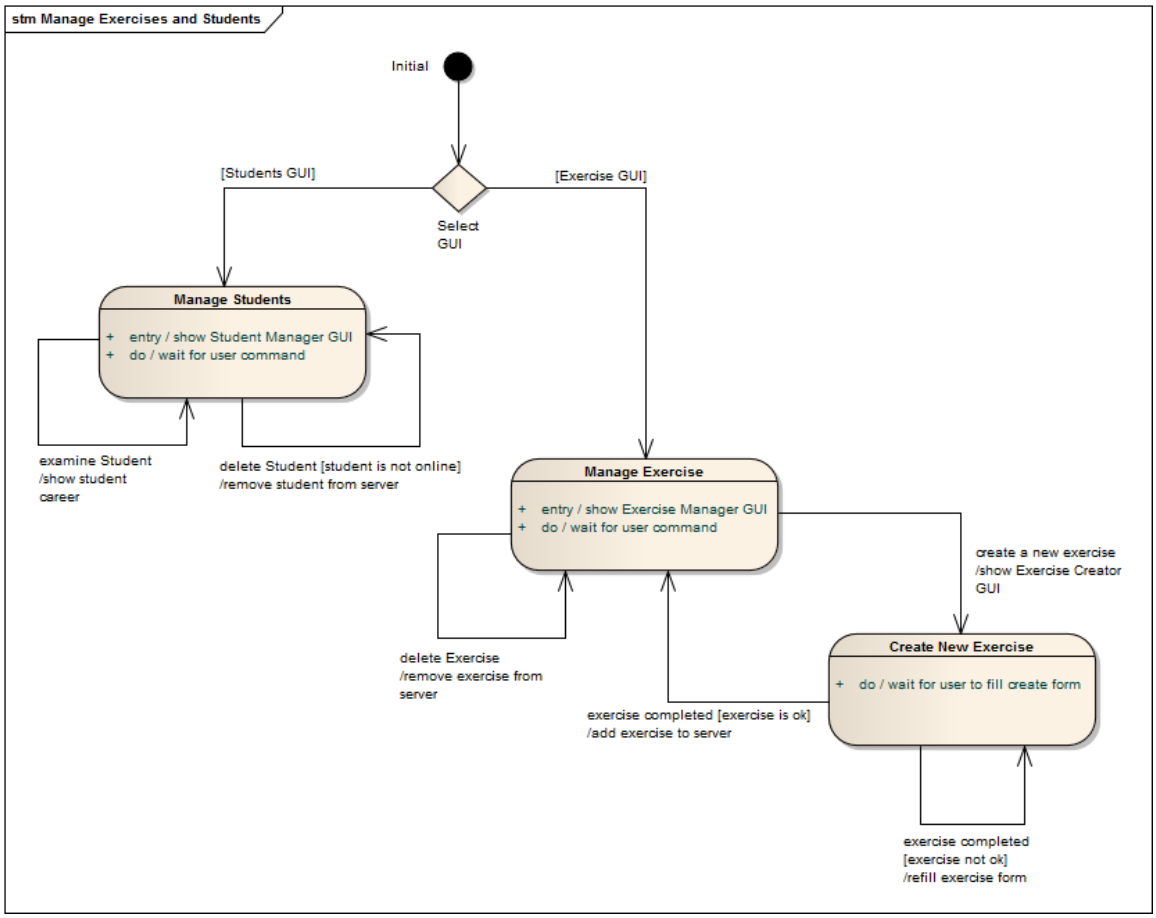

Figura 5.22: Comportamento Manage Exercise and Students.

In Figura 5.22 è rappresentato il diagramma degli stati di Manage Exercise and Students. Come si può vedere è organizzato in due sottostati, uno che gestisce gli studenti e uno che gestisce gli esercizi. La gestione degli studenti permette di visionare la carriera dello studente all'interno del laboratorio e di cancellarla. La gestione degli esercizi invece permette di eliminare un esercizio dal laboratorio o di crearne uno nuovo. La creazione di un nuovo esercizio consiste nel riempimento di una griglia in una apposita interfaccia. Nel caso i vincoli della griglia siano stati rispetti l'esercizio viene inserito nel laboratorio, altrimenti no. In Figura 5.23 è

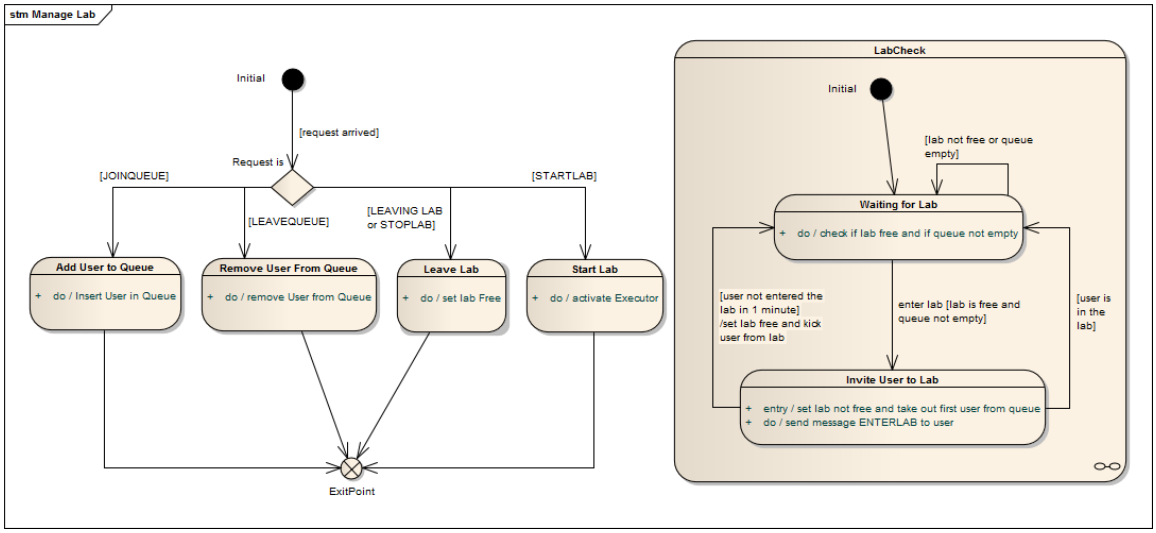

Figura 5.23: Comportamento Manage Lab.

invece rappresentato il comportamento della parte che si occupa di gestire l'accesso al laboratorio remoto. Una parte si occupa di gestire le richieste dell'utente, in particolare:

- JOINQUEUE: l'utente viene inserito nella coda di accesso al laboratorio.
- LEAVEQUEUE: l'utente viene tolto dalla coda di accesso al laboratorio.
- LEAVINGLAB o STOPLAB: l'utente è uscito dal laboratorio o ha stoppato l'esercitazione. Il laboratorio viene messo nello stato Free.
- STARTLAB: l'utente è all'interno del laboratorio e ha richiesto l'esecuzione di un esercizio. Viene allora attivato il componente Executor.

La parte LabCheck invece non fa altro che controllare continuamente se si verifica la condizione Laboratorio libero e coda di attesa non vuota. Al verificarsi di questa condizione preleva il primo utente in coda di attesa, mette il laboratorio nello stato Occupato e invia all'utente il messaggio ENTERLAB. Se l'utente non entra nel laboratorio entro 60 secondi, LabCheck mette il laboratorio nello stato Free e prosegue con l'esecuzione.

Client GUI

La Figura 4.7 nel Capitolo 4 riflette ancora sufficientemente bene il comportamento che dovrà tenere il client. In fase di progettazione risulta importante però definire i comandi che il client potrà inviare al server e che il server dovrà gestire ed eseguire (in Figura rappresentato dallo stato Execute User Command). Ricordiamo che le funzionalità che il client deve permettere all'utente sono: il poter esaminare gli esercizi disponibili, richiedere di entrare nel laboratorio remoto e se possibile entrarci, una volta all'interno del laboratorio remoto lo studente deve potere caricare una soluzione, vedere l'esecuzione dell'esercizio e ricevere una valutazione, in caso di necessità il cliente deve avere la possibilità di interrompere l'esercitazione. I comandi principali in questo scenario sono:

- GETEXERCISES: il client richiede al server gli esercizi presenti nel laboratorio.
- GETSOLVED: il client richiede al server di sapere se lo studente collegato ha risolto o meno un dato esercizio.
- JOINQUEUE: il client richiede di essere messo in coda per entrare nel laboratorio
- LEAVEQUEUE: il client richiede di lasciare la coda di attesa.
- ENTERINGLAB: quando è il proprio turno il client chiede di entrare nel laboratorio.
- LOADINGSOL: il client sta caricando la propria soluzione.
- STARTLAB: il client fa partire l'esecuzione dell'esercitazione.
- STOPLAB: il client interrompe l'esecuzione del laboratorio.
- READINGDATA: il client legge dati che vengono emessi da Executor riguardanti l'esecuzione dell'esercizio.
- GETEVALUATION: il client richiede la valutazione del proprio esercizio.

- LEAVINGLAB: il client esce dal laboratorio remoto che ora diventa libero.

Capitolo 6

WebLab - Implementazione e Deployment

6.1 Introduzione

Al termine della fase di progettazione, ottenuta una visione più chiara sia dell'insieme che delle singole parti, si passa alla fase di implementazione e di deployment, ovvero sia la distribuzione fisica dei componenti del sistema. In questo capitolo ci occuperemo appunto di illustrare alcune caratteristiche fondamentali sia dell'implementazione che della distribuzione dei vari sottosistemi. Analizzeremo anche lo sviluppo delle altre due parti che compongono la piattaforma WebLab, ovvero sia il sito web e la gestione della telecamera.

6.2 Il sito web

Il sito web per WebLab è stato pensato fin dall'inizio come punto d'accesso e riferimento per lo studente e come mezzo di comunicazione da parte del docente. Rivestendo questo doppio ruolo, il sito deve essere in grado sia di fornire accesso allo studente alla piattaforma responsabile del laboratorio remoto sia di permettere la creazione e la gestione dei contenuti da parte del docente e la fruizione degli stessi da parte dello studente.

Implementazione

Come punto di partenza ed esempio in questo senso, si è scelto di utilizzare nella realizzazione del sito uno dei CMS [1] disponibili al momento in ambito open-source.
La scelta è ricaduta su Drupal [2], nella sua versione 7, per la sua stabilità e modularità. La nuova versione inoltre prevede l'introduzione di alcune norme di sicurezza migliorate (ad esempio l'utilizzo di tecniche avanzate di hashing per lo storage delle password) e una interfaccia completamente rinnovata che rende tutte le operazioni molto intuitive e immediate.

Figura 6.1: Il logo di Drupal.

[1] Content Management System: è uno strumento software installato su un server web studiato per facilitare la gestione dei contenuti di siti web, svincolando l'amministratore da conoscenze tecniche di programmazione Web
[2] http://drupal.org/

Il sito così realizzato permette dunque le seguenti operazioni:

- STUDENTE:
  - Accesso ai contenuti del sito, tra cui ad esempio informazioni sull'utilizzo della piattaforma.
  - Download di materiale di riferimento (Figura 6.3).
  - Visione live dei laboratori remoti attraverso webcam (Figura 6.2).
  - Lettura di avvisi lasciati da parte del docente.
  - Accesso ad un forum di riferimento dove scambiare opinioni con altri studenti e docenti (Figura 6.4).
  - Accesso alla piattaforma WebLab tramite interfaccia grafica (Figura A.3).
- DOCENTE:
  - Visione e gestione degli studenti iscritti.
  - Possibilità di inserire avvisi e contenuti (Figura 6.5).
  - Gestire e moderare il forum.

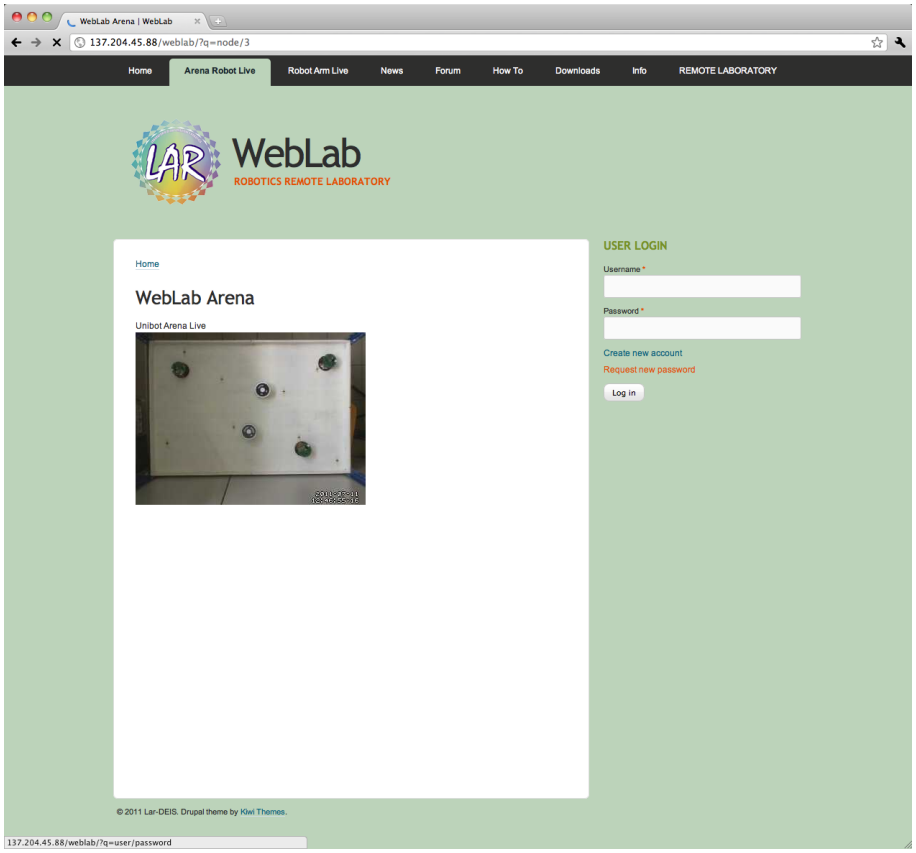

Figura 6.2: La webcam di WebLab.

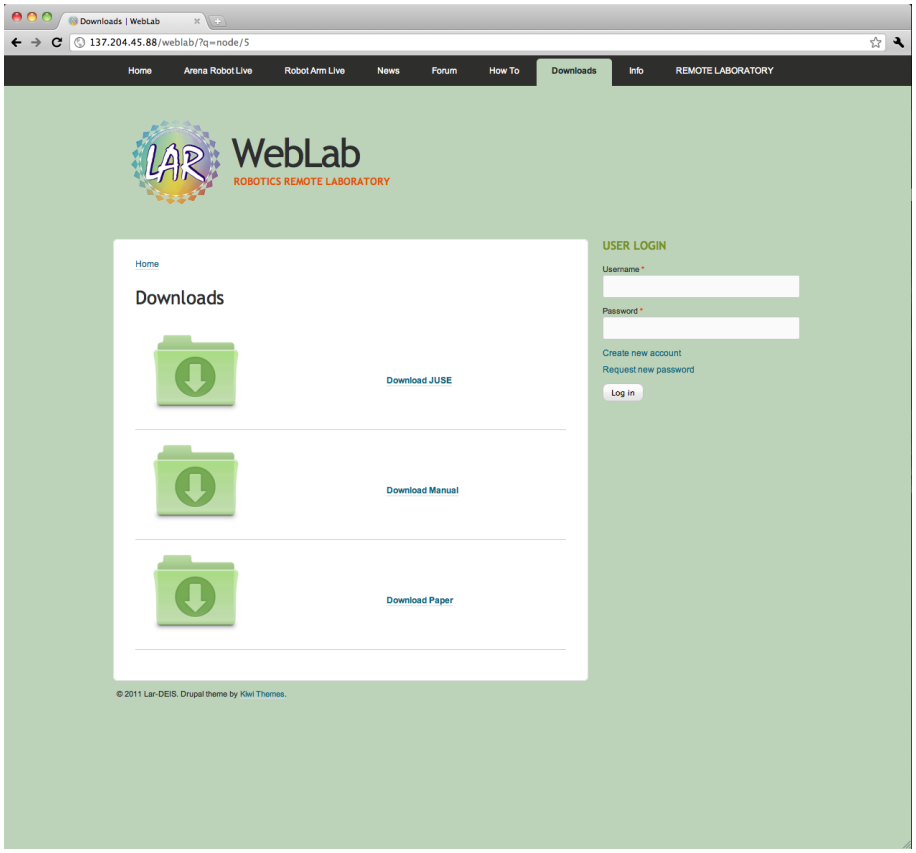

Figura 6.3: La pagina di download del materiale di WebLab.

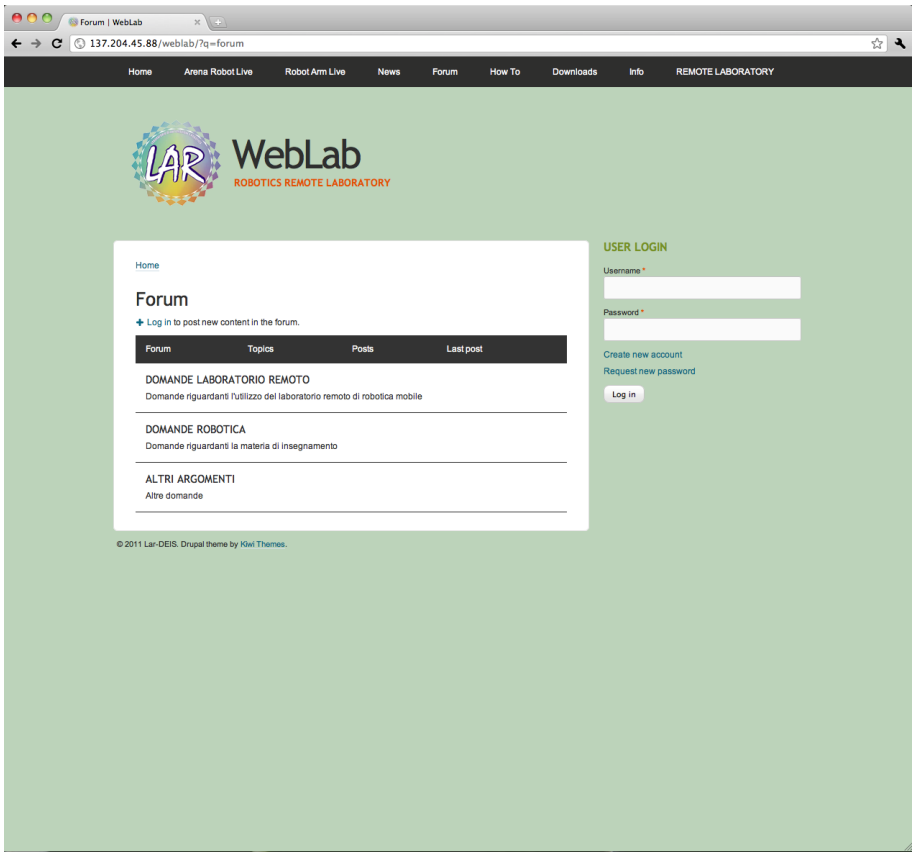

Figura 6.4: Il Forum di WebLab.

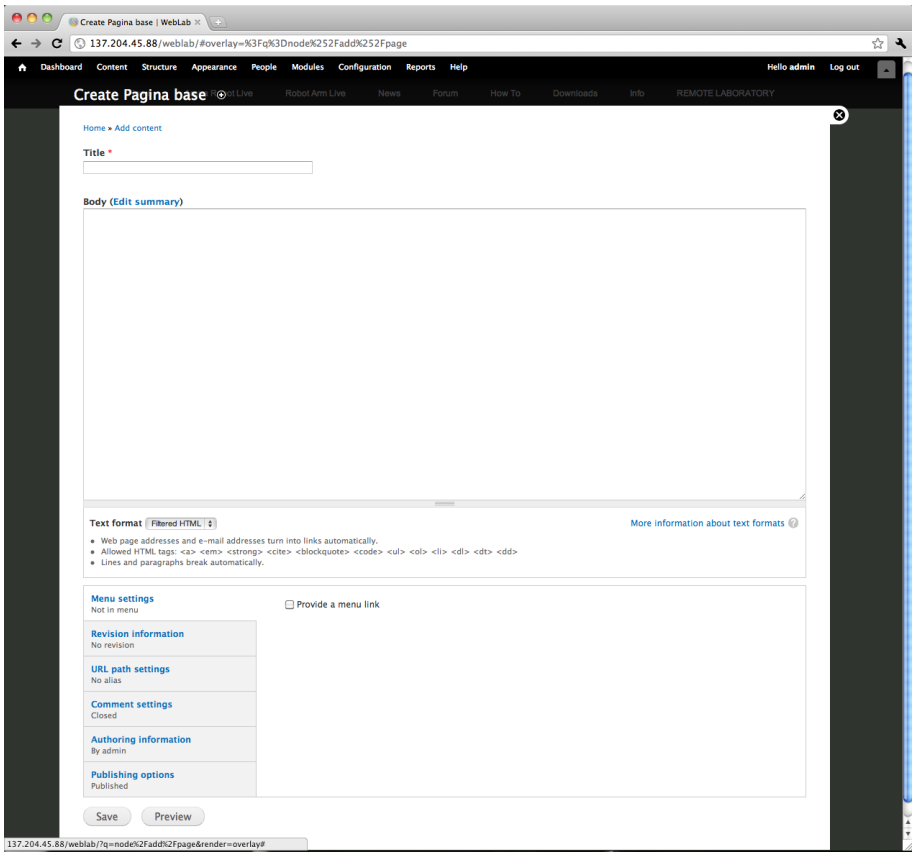

Figura 6.5: L'interfaccia di Drupal 7 per inserire contenuti in Weblab.

Deployment

Il sito è stato caricato su di un Web Server Apache2[3] installato su di una macchina con sistema operativo Linux. Questa macchina è stata opportunamente configurata per consentire connessioni dall'esterno e funge da centro dell'intero sistema WebLab. Su questa macchina risiede e si esegue anche l'applicazione Authentication Server, come vedremo tra poco.

6.3 La webcam

La gestione della webcam è un altro punto molto importante nell'insieme del funzionamento del sistema WebLab. La webcam infatti deve essere in grado di riprendere l'intero laboratorio remoto e permettere così all'utente di osservare ciò che accade in diretta. L'utente deve poter accedere alla visione sia tramite il sito, sia all'interno dell'applicazione vera e propria mentre viene eseguita la soluzione da lui proposta per l'esercitazione.

Implementazione

La gestione della webcam è stata affidata ad un semplice programma open-source di nome Motion [4]. Tale applicazione è disponibile per i sistemi Linux e si occupa principalmente di videosorveglianza: gestisce l'acquisizione di immagini e flussi video da sorgenti quali webcam ed è configurabile per attivare la registrazione in caso di presenza di movimento. In questo caso il programma è stato configurato per essere eseguito come daemon sotto Linux e riprendere in maniera continuativa 50 frame ad intervalli di 200ms dalla webcam collegata alla macchina. Motion inoltre può installare ed eseguire un webcam server sulla macchina in modo da permettere la visione delle riprese anche da pagina web: questa opzione è stata utilizzata per mostrare la visione della webcam all'interno del sito WebLab (Figura 6.2).

[3] http://www.apache.org/

[4] http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome

Deployment

La gestione della webcam deve essere eseguita sulla macchina in cui è installato ed eseguito Lab Server. Se Lab Server è eseguito su una macchina Linux la soluzione precedentemente esposta è valida ed applicabile. Nel caso in cui Lab Server sia eseguito su macchine con sistemi operativi diversi da Linux (Windows, OSX o altro) si dovranno cercare strade alternative. L'importante è riuscire a catturare un certo numero di frame continuativi dalla webcam e salvarli in una precisa directory del sistema. Questo perchè anche l'applicazione ClientGUI dovrà accedere a tale immagini e quindi la directory contenente i frame del laboratorio dovrà essere passata a ClientGUI all'inizio come parametro di configurazione.
Nel caso di sistema operativo Windows, a titolo puramente esemplificativo, può risultare abbastanza semplice ed intuitivo realizzare una piccola applicazione Java, utilizzando le librerie JMF [5], che esegua la cattura di un numero fisso di frame da una webcam collegata al sistema.

[5]JAVA MEDIA FRAMEWORK: http://www.oracle.com/technetwork/java/javase/documentation/index-135173.html

6.4 WebLab Application

Andremo ora ad analizzare nel l'implementazione e il deployment di tutti i componenti dell'applicazione WebLab.

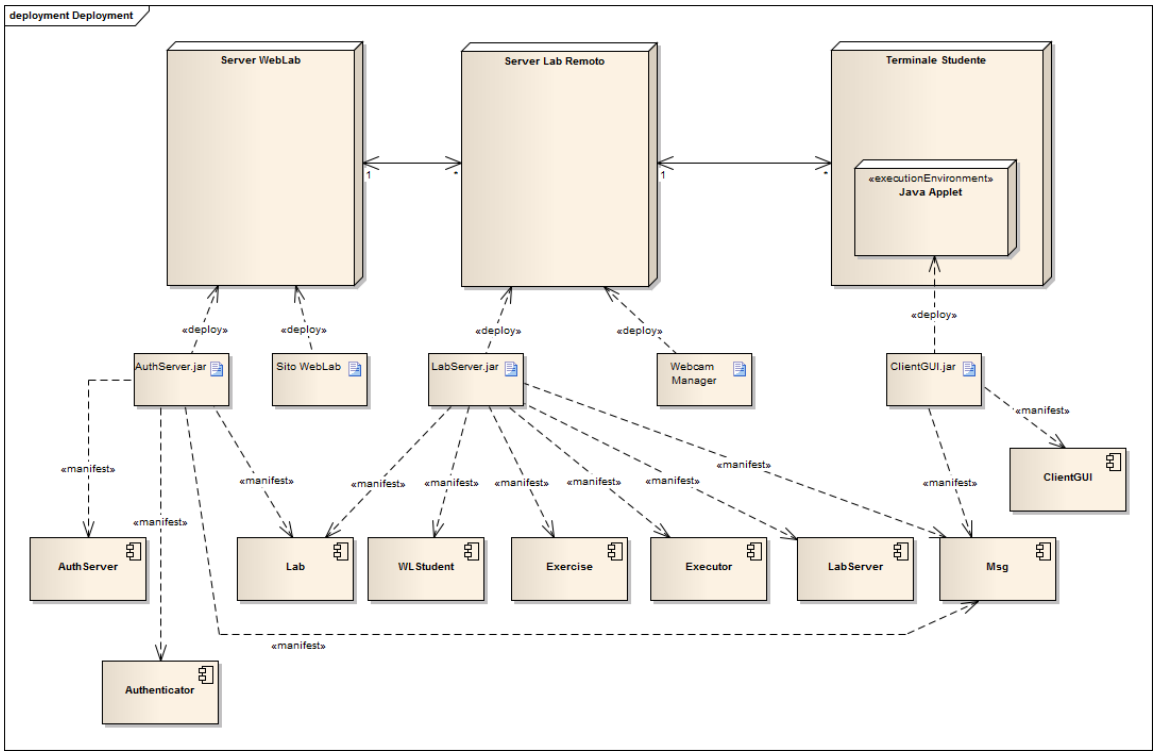

Figura 6.6: Deployment WebLab.

6.4.1 Authentication Server

Implementazione

AuthServer è stato realizzato in Java su piattaforma operativa JavaSE-1.6. Alcuni dettagli implementativi includono:

- Le comunicazioni con laboratori (LabServer) e studenti (ClientGUI) sono gestite attraverso socket java su canali di tipo TCP. Auth server rimane in ascolto per LabServer sulla porta TCP 30490 e per ClientGUI sulla porta TCP 30480. Tali porte sono state opportunamente sbloccate nella configurazione del firewall della macchina che ospita AuthServer.

- L'interfaccia utente dell'applicazione è stata realizzata usando Java SWING. È composta da due JPanel, uno che visualizza il log delle operazioni avvenute sul server e permette di avviare e stoppare l'applicazione, l'altro che permette di visualizzare gli utenti connessi e i laboratori attivi (Figura A.2).

- AuthServer mantiene un log di tutte le operazioni che compie e questo viene salvato su un file di testo nella directory /logs/ . Il file di testo viene creato ad ogni avvio di AuthServer e ha un nome indicante data e ora della creazione.

- AuthServer richiama il componente Authenticator nel momento in cui deve verificare le credenziali dell'utente. Authenticator può essere implementato in varie maniere: gestendo una lista di utenti permessi e relative password o nel caso interagendo direttamente con il database del sito drupal, attraverso le API JDBC, e verificando le credenziali in base all'iscrizione effettuata dall'utente sul sito WebLab.

Deployment

Il deployment di AuthServer viene effettuato attraverso un file .jar di nome AuthServer.jar che contiene al suo interno i componenti AuthServer,Lab,Authenticator e Msg. Il file AuthServer.jar sarà poi depositato

e mandato in esecuzione sul server centrale dell'architettura WebLab. Su tale server sarà inoltre presente il sito Drupal di WebLab con relativo database MySql, col quale, nel caso servisse per l'autenticazione, il componente Authenticator potrà dialogare. La struttura della directory di AtuhServer sarà la seguente:

```
AuthServer.jar      // Jar Eseguibile
/logs/              // Directory in cui saranno contenuti
                    // i file di Log
```

6.4.2 Lab Server

Implementazione

LabServer è stato realizzato in Java su piattaforma operativa JavaSE-1.6. Alcuni dettagli implementativi includono:

- Al primo avvio di LabServer, l'applicazione richiederà all'utente di compilare una griglia inserendo i dati del laboratorio (nome,indirizzo ip,porta e robot disponibili) e i dati del server di autenticazione (indirizzo ip e porta di comunicazione). Una volta inseriti questi dati saranno mantenuti persistenti attraverso un file di nome labconf salvato nella directory /confs/ .

- Le comunicazioni tra LabServer e AuthServer e ClientGUI avvengono attraverso socket java su canale di tipo TCP. In particolare AuthServer rimane in ascolto per ClientGUI sulla porta TCP 30500.

- Il componente QueueManager genera il Thread LabCheck che stabilisce anche lui a sua volta una connessione con ClientGUI di tipo TCP sulla porta 30502 per poter inviare se necessario a ClientGUI il segnale che il laboratorio è pronto per accoglierlo.

- L'interfaccia grafica è stata realizzata usando Java SWING. In particolare è composta da 6 JPanel: uno per il login, uno per la configurazione del laboratorio, uno di monitoraggio delle operazioni avvenute

sul server, uno per la gestione degli esercizi, uno per la creazione degli esercizi e uno per la supervisione degli studenti.

- Le informazioni riguardanti gli studenti, ovvero gli esercizi superati e le rispettive valutazioni, vengono rese persistenti memorizzandole in file con nome *nomestudente*.wls nella directory /students/. Ogni volta che uno studente accede al laboratorio il sistema controlla se nella directory in questione è presente un associato allo studente: se il file è presente ne recupera le informazioni, altrimenti ne crea uno nuovo vuoto.

- Gli esercizi vengono creati attraverso l'interfaccia messa a disposizione dall'applicazione. Ogni esercizio viene reso persistente memorizzandolo come un file con nome *Esercizioxxx*.wle nella directory /exercises/. Tale directory contiene anche le sottodirectory /setup/ , /success/, /images/, che contengono a loro volta rispettivamente i file di setup,successo e immagine rappresentativa di ogni esercizio. Gli esercizi vengono numerati sequenzialmente attraverso il loro campo ID utilizzando un indice interno al laboratorio presente nel file labconf che viene incrementato ogni volta che viene creato un esercizio.

- Ogni volta che uno studente entra nel laboratorio e carica la propria esecuzione, Activator mette in esecuzione Executor che si occupa di fare muovere il laboratorio. Nel caso l'esercizio venga superato con successo, l'ID dell'esercizio con il rispettivo voto vengono memorizzati nel file nomestudente.wls . La soluzione caricata dallo studente viene memorizzata nella cartella /solutions/ fintanto che l'esercizio è in esecuzione e quindi, in caso di completamento dell'esercitazione, viene copiata nella sottocartella /solutions/backups/ con il nome *esercizio-studente-valutazione*.sol come backup in caso di bisogno per controlli a posteriori.

Deployment

Il deployment di LabServer viene effettuato attraverso un file .jar di nome LabServer.jar che contiene al suo interno i componenti Labserver, Lab, Exercise, WLStudent, Executor e Msg. Il file LabServer.jar sarà poi depositato e mandato in esecuzione sul server che gestisce il laboratorio. A tale server sarà connessa anche la webcam che riprende l'evoluzione del laboratorio remoto con relativo software di acquisizione. La directory contenente le immagini catturate farà parte dei parametri del laboratorio e andrà inviata a AuthServer che poi provvederà ad inviarla insieme agli altri dati a ClientGUI. La struttura della directory di LabServer sarà la seguente:

```
LabServer.jar          // Jar Eseguibile
/logs/                 // Directory per i file di Log
/students/             // Directory per i file studenti.wls
/exercises/            // Directory per gli esercizi.wle
     /images/          // Directory per i file immagine degli
                       //esercizi
     /setup/           // Directory per i file di setup degli
                       // esercizi
    /success/          // Directory per i file successo degli
                       //esercizi
/solutions/            // Directory per l'upload delle
                       //soluzioni
     /backups/         // Directory di backup per le soluzioni
```

6.4.3 ClientGUI

Implementazione

ClientGUI è stato realizzato in Java su piattaforma operativa JavaSE-1.6. Alcuni dettagli implementativi includono:

- L'interfaccia utente è stata realizzata con Java SWING. È composta da 3 JPanel: uno per il login, uno per la parte di attesa (hall) e uno per il laboratorio.

- Si è cercato di alleggerire il client da quanto più lavoro possibile, lasciano il compito di svolgere le operazioni completamente al server. Il client può solo inviare comandi al server ed attendere le risposte. I comandi vengono inviati attraverso una connessione TCP usando socket java.

- ClientGUI utilizza due Timer Java per temporizzare due esecuzioni: l'entrata nel laboratorio e il tempo massimo che un utente può trascorrere all'interno del laboratorio. La durata di questi due eventi è un parametro del laboratorio e viene inviata a ClientGUI all'atto dello scambio di informazioni iniziali.

Deployment

Il deployment di ClientGUI viene effettuato attraverso un file .jar di nome ClientGUI.jar che contiene al suo interno i componenti ClientGUI e Msg. Il deployment di ClientGUI è sicuramente una delle parti più delicata dell'intero sistema.
Un requisito fondamentale dell'intera piattaforma era che l'utente fosse in grado di accedere al laboratorio remoto direttamente da una pagina web del sito, senza la necessità di scaricare applicazioni esterne. Per fare questo l'applicazione ClientGUI deve essere in grado di andare in esecuzione all'interno di una pagina web. Si è quindi pensato di realizzarla come Applet e avere così la possibilità di integrarla all'interno del sito WebLab.

Le Applet non hanno gli stessi diritti di una applicazione, vengono eseguite all'interno di una SandBox, sorvegliate dall'AppletSecurityManager. Le Applet comuni hanno molte limitazioni:

- Non posso accedere al file system locale.
- Non possono accedere alla rete tranne che alla macchina da cui sono state scaricate.
- Non hanno accesso a molte proprietà del sistema.

Queste enormi limitazioni si scontrano con le necessità dell'applicazione ClientGUI, in particolare perchè questa ha necessità di instaurare più connessioni anche con entità diverse (i vari laboratori remoti) e ha inoltre bisogno di accedere la file system dell'utente (ad esempio per poter scaricare il file report di avvenuta esercitazione).
Per superare questi limiti si è deciso di firmare digitalmente l'Applet ClientGUI. In questa maniera ClientGUI potrà operare come una applicazione normale, con tutti i permessi e senza restrizioni imposte dal modello Sandbox. All'utente verrà richiesto di accettare l'esecuzione dell'Applet presentandogli le credenziali del certificato digitale.

Capitolo 7

WebLab - Protocollo di sicurezza

7.1 Introduzione

La sicurezza all'interno di una applicazione distribuita gioca un ruolo fondamentale. Molti sono i messaggi scambiati fra le varie parti dell'applicazione e su tali messaggi bisogna imporre chiare e precise regole di sicurezza [5]. Tali regole andranno quindi a formare un protocollo che ci permetterà di articolare lo scambio di messaggi fra le entità in gioco per garantire le proprietà fondamentali della sicurezza informatica.
Ad un primo sguardo si nota che in particolare due frangenti saranno fondamentali: l'autenticazione dell'utente e dei laboratori con Authentication Server e il riconoscimento da parte di Lab Server dell'autenticità di un utente quando questo si connette al laboratorio virtuale. In questo capitolo ragioneremo appunto su tali argomenti, dando prima una panoramica delle proprietà che ci interessa garantire e degli attacchi dai quali prevediamo di doverci difendere, per poi passare ad illustrare un possibile protocollo di sicurezza prendendo spunto da alcuni famosi protocolli attualmente in uso e in circolazione.

7.2 I possibili attacchi e le proprietà da garantire

La sicurezza informatica è di fondamentale importanza all'interno di ogni applicazione, particolarmente per quelle che utilizzano la rete (sia essa locale o intranet) per comunicare e scambiarsi messaggi. Ogni sistema può presentare vulnerabilità che al presentarsi di alcune situazioni, che possiamo definire minacce, possono indurre comportamenti non voluti e quindi danni. Un esempio di punti deboli possono essere comportamenti non voluti da parte di utenti o particolari configurazioni hardware o software o ancora particolari forme di trasmissione di dati. Le tecnologie per la sicurezza definiscono delle contromisure, ovvero azioni, procedure, tecniche o dispositivi che possono rimuovere o ridurre le vulnerabilità garantendo al sistema determinate proprietà, ad esempio:

- riservatezza: regole sulla conoscenza alle risorse
- disponibilità: regole sull'accesso alle risorse
- integrità: regole per permettere di verificare se un dato è stato modificato
- autenticazione: regolamentare l'identificazione dell'utente o di due entità fra loro all'interno di una comunicazione
- non ripudio: regole per attribuire in maniera univoca la paternità di un dato

All'interno della struttura WebLab i requisti per la sicurezza richiedono sicuramente che siano soddisfatte tre proprietà in particolare:

- riservatezza: per tutto quello che riguarda i dati degli utenti, in particolare le password che andranno mantenute memorizzate in maniera sicura
- integrità: per quel che riguarda il passaggio di dati tra ClientGUI e Lab Server, in particolare per quel che riguarda il caricamento di una soluzione di un esercizio

- autenticazione: per garantire una corretta identificazione sia degli utenti con Auth Server, ma anche dei singoli laboratori con Auth Server e di ogni ClientGUI con Lab Server

7.2.1 Possibili attacchi

Tali proprietà devono essere garantite per impedire che utenti malintenzionati possano effettuare degli attacchi minacciando la sicurezza del sistema. Analizzando la struttura dell'applicazione si possono immaginare vari tipi di attacchi che potrebbero sfruttare le vulnerabilità del sistema per generare dei danni.

Attacco dell'uomo in mezzo (MITM)

Proprietà minata: autenticazione

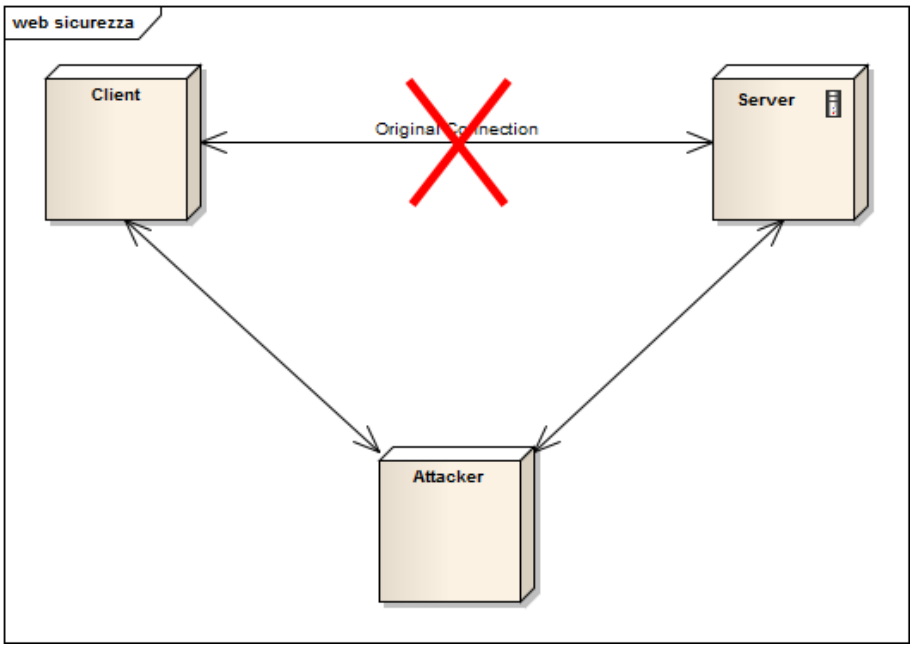

Figura 7.1: Attacco dell'uomo in mezzo.

L'attacco dell'uomo in mezzo è un tipico attacco in cui un malintenzionato rimane in ascolto dei messaggi fra due interlocutori e falsifica gli

scambi dei messaggi in modo da farsi passare per uno dei due. I due interlocutori non si accorgono di nulla. Per eliminare questo tipo di vulnerabilità bisogna evitare la simmetria dei messaggi, ad esempio inserendo timestamp o numeri di sequenza.

Attacco di reflection

Proprietà minata: autenticazione
Utilizzando più sessioni di uno stesso protocollo di identificazione con messaggi simmetrici e intercettando i messaggi si può riuscire a concludere il protocollo di autenticazione. Per proteggersi da questo tipo di attacco si deve ancora una volta eliminare la simmetria, inserendo nonce [1] e l'identificatore del ricevente.

Intercettazione

Proprietà minata: riservatezza Un utente malintenzionato rimane in ascolto dei messaggi tra due interlocutori. Per evitare questo tipo di attacco si deve rendere sicuro il canale di comunicazione, attraverso l'uso di cifratura o di servizi sicuri già esistenti (SSL,IPsec).

Modifica dei pacchetti

Proprietà minata: integrità Un utente malintenzionato può intercettare e modificare i valori dei messaggi scambiati tra due interlocutori. Questo attacco non può essere prevenuto ma può essere rilevato trasmettendo oltre al messaggio la sua impronta, in modo che il destinatario abbia modo di rilevare se ci sono state manomissioni.

[1] un numero, generalmente casuale o pseudo-casuale, che ha un utilizzo unico.

7.3 Soluzioni e protocolli di sicurezza

Analizziamo ora le soluzioni e i protocolli attuati per garantire le proprietà di sicurezza precedentemente elencate.

7.3.1 Riservatezza

All'interno della struttura di un laboratorio remoto dobbiamo garantire la riservatezza in un caso in particolare: nella memorizzazione delle password degli utenti all'interno del server di autenticazione. Le password degli utenti andranno memorizzate non in chiaro, ma andrà memorizzata esclusivamente la loro impronta. Questo meccanismo può a sua volta essere reso più sicuro aggiungendo alla password dei salt ed effettuando tecniche di *stretching* per rendere l'impronta generata ancora più sicura.

7.3.2 Autenticazione e integrità

Per gestire l'autenticazione possiamo pensare di implementare un protocollo di scambio di messaggi che si ispira al protocollo di autenticazione di Kerberos [2], visto che la struttura di WebLab ricorda per certi aspetti la struttura di un sistema Kerberos.

Kerberos

Kerberos è un servizio di autenticazione per un ambiente client/server sviluppato agli inizi degli anni 80 al MIT. Compito del sistema è controllare l'accesso ad una serie di server da parte di una comunità di utenti. È costituito da un protocollo di scambio di messaggi per garantire autenticazione tramite crittografia. Kerberos previene attacchi di intercettazione, MITM e assicura anche integrità dei dati. Fornisce mutua identificazione e elimina la comunicazione della password. Il sistema è composta da due server:

[2] http://web.mit.edu/kerberos/

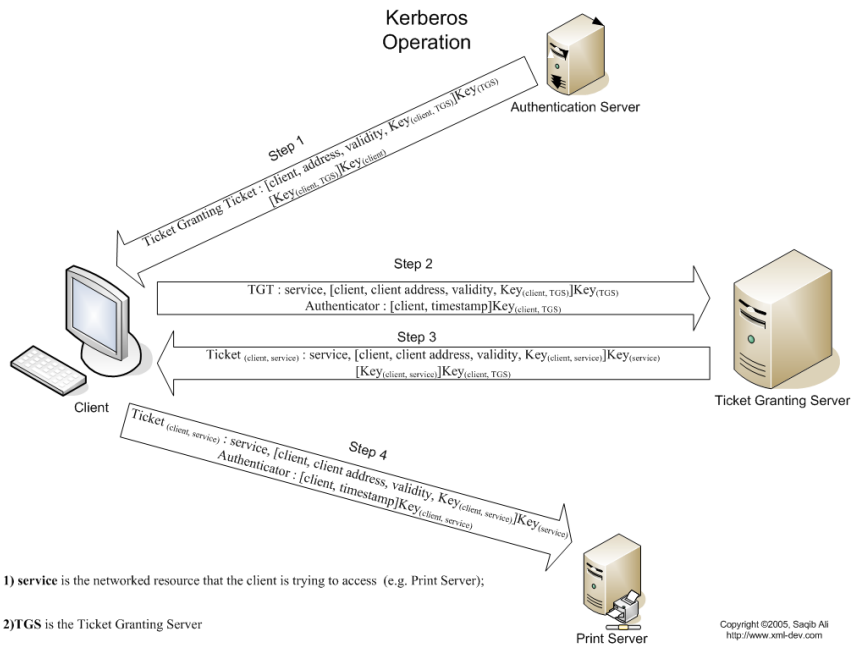

Figura 7.2: Kerberos: il protocollo.

- AS (Authentication Server): memorizza password cifrate e lancia una sfida di identificazione a chi inizia una sessione di lavoro e gli restituisce un documento cifrato che dovrà presentare a TGS al fine di ottenere le specifiche autorizzazioni
- TGS (Ticket Granting Server): condivide una chiave segreta con AS ed una con ogni altro server del sistema. Riceve il documento di AS dall'utente, lo decifra e lo verifica. Esamina la correttezza della risposta alla sfida lanciata da AS e, nel caso sia tutto corretto, invia all'utente un diritto di accesso (ticket) per il server a cui vuole accedere, valido per tutta la sessione di lavoro.

Il protocollo di autenticazione di Kerberos prevede 6 fasi ed utilizza timestamping e invio dell'identificatore del ricevente per eliminare la simmetria dei messaggi e garantire autenticazione ed integrità.

7.3.3 Un protocollo di sicurezza per WebLab

Esaminando il modello proposto da Kerberos si possono trovare molte analogie con la struttura e il funzionamento di WebLab. Entrambi hanno un Authentication Server (AS) che gestisce la prima connessione di un utente. Kerberos ha in più un Ticket-Granting Server (TGS) che permette di organizzare la struttura di Kerberos in sottosistemi chiamati *reami* che non troviamo in WebLab. Si può pensare e progettare un protocollo di sicurezza che si ispiri al modello proposto da Kerberos ma accentrando AS e TGS in un unico server, nel nostro caso per Weblab in Auth Server, responsabile di gestire l'accesso e la distribuzione dei ticket per accedere ai vari Lab Server.
Ispirandoci al modello di Kerberos, progettiamo e descriviamo ora un protocollo di autenticazione per WebLab, esaminando e descrivendo passo a passo le varie fasi.

Fase 1

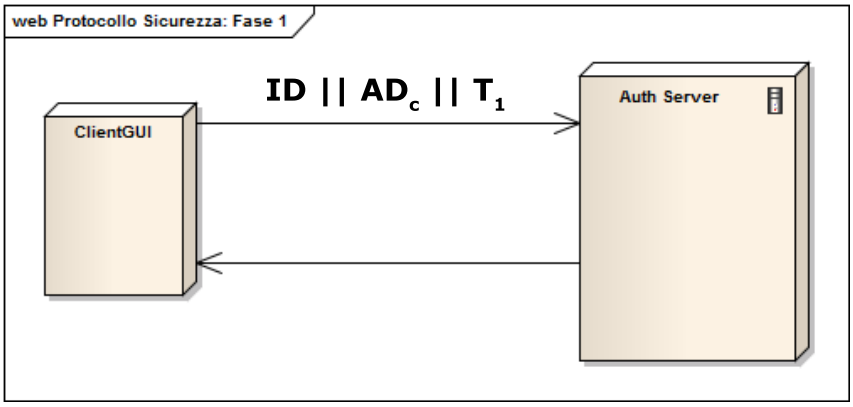

Figura 7.3: WebLab: il protocollo, Fase 1.

Quando ClientGUI si collega per la prima volta fornisce il proprio identificativo (ID, ad esempio username), il proprio indirizzo di rete (AD_C) e una marca temporale di log-on (T_1).

Fase 2

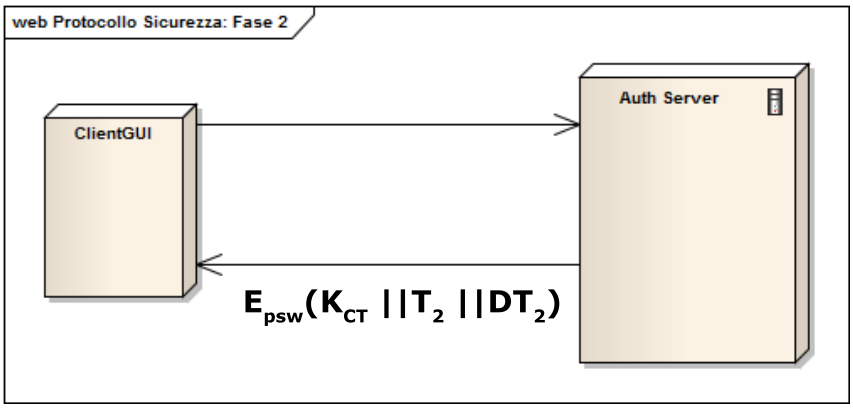

Figura 7.4: WebLab: il protocollo, Fase 2.

AS controlla T_1, preleva dalla propria memoria l'impronta della password corrisponde all'ID contenuto nel messaggio e la usa per cifrare il messaggio in Figura 7.4 sfidando così ClientGUI a decifrarlo. Il messaggio contiene la chiave di sessione nel dialogo con Auth Server (K_{CT}), l'inizio e la durata massima della sessione di lavoro (T_2,DT_2).

Fase 3

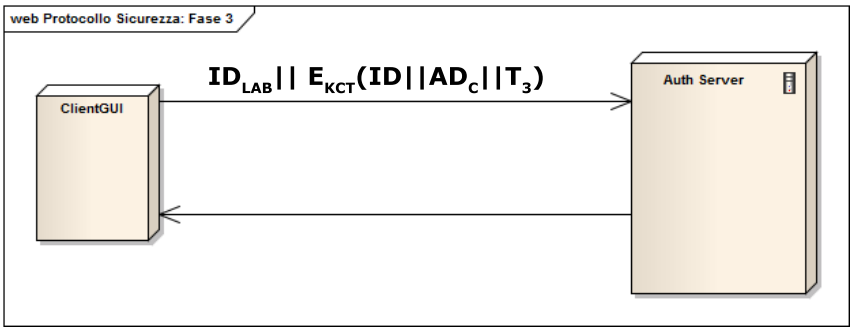

Figura 7.5: WebLab: il protocollo, Fase 3.

ClientGUI calcola l'impronta della password e la usa per decifrare il messaggio di AS. Se riesce ricava K_{CT} ed invia ad AS un ulteriore messag-

gio contente l'ID del Lab Server scelto, unito ad un autenticatore cifrato con K_{CT} per dimostrare di aver superato la sfida portata da AS. T_3 è un timestamp che consente ad AS di controllare se la sessione è ancora valida.

Fase 4

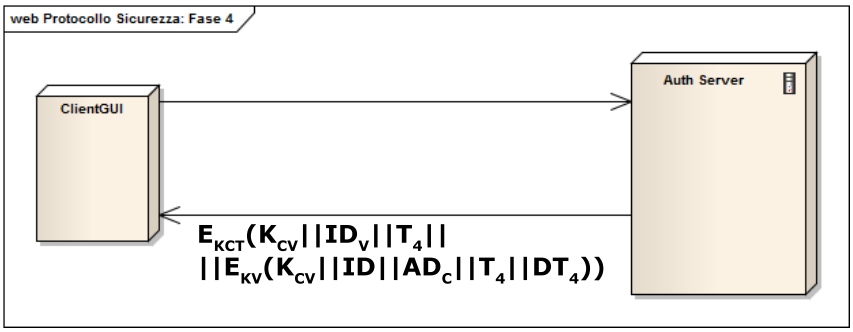

Figura 7.6: WebLab: il protocollo, Fase 4.

AS utilizzando K_{CT} decifra l'autenticatore e completa il protocollo sfida-risposta con ClientGUI. A questo punto genera una chiave di sessione per la comunicazione fra ClientGUI e LabServer K_{CV} e la invia all'interno di un messaggio cifrato con la chiave di sessione K_{CT}. All'interno di questo messaggio cifrato vi sarà anche il ticket per comunicare con il laboratorio prescelto cifrato con una chiave K_V che AuthServer e LabServer avevano condiviso in precedenza.

Fase 5

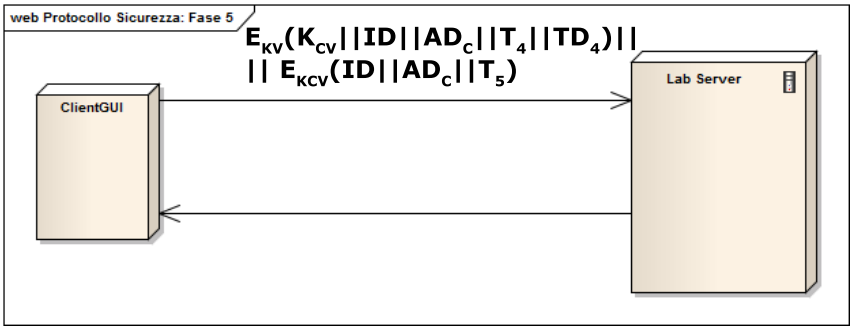

Figura 7.7: WebLab: il protocollo, Fase 5.

ClientGUI decifra il messaggio di AS con la chiave di sessione e ottiene la chiave di sessione per comunicare con LabServer K_{CV} e il ticket preparato da AS per LabServer. Ora ClientGUI inoltra a LabServer il ticket generato da AS ed un autenticatore con cui da un lato si fa identificare e dall'altro sfida Lab Server a dimostrare la propria identità. La marca temporale T_5 indica il momento in cui ClientGUI contatta LabServer, la sessione di autenticazione scadrà all'istante T_5+DT_4.

Fase 5

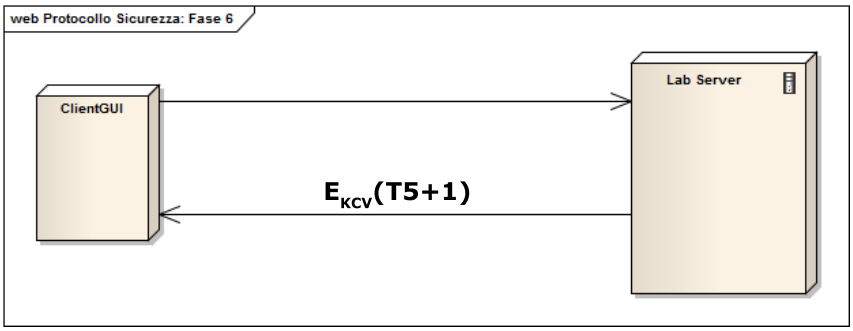

Figura 7.8: WebLab: il protocollo, Fase 6.

Siamo giunti al termine del protocollo di identificazione. Ora LabServer dimostra la propria identità usando K_V per decifrare il ticket preparatogli

da AS ottenendo così tutti i dati su ClientGUI e la chiave di sessione K_{CV}. Con questa chiave può decifrare l'autenticatore di ClientGUI ed essere certo della sua identità. Per concludere il protocollo LabServer invia a ClientGUI un messaggio con cui dimostra di aver decifrato tutto e con questo da conferma a ClientGUI della propria identità. A questo punto ClientGUI è certo di stare comunicando proprio con il LabServer prescelto ed inizia ad avvalersi dei suoi servizi.

Capitolo 8

Esempi Applicativi

8.1 Introduzione

Siamo quindi giunti al termine del processo di sviluppo della piattaforma WebLab, partendo dai requisiti, attraverso l'analisi, fino alla progettazione, all'implementazione e al deployment. La piattaforma WebLab fin qui analizzata, attraverso l'integrazione di varie componenti, è quindi in grado di realizzare ciò che era richiesto dai requisiti. In questo capitolo approfondiremo e esemplificheremo alcuni esempi applicativi di tale piattaforma, siano essi hardware o software, in modo da evidenziare alcuni scenari d'uso per WebLab.

8.2 Un esempio d'uso

La piattaforma WebLab permette agli studenti di registrarsi utilizzando il sito e, sempre tramite il sito, di poter utilizzare l'applicazione per usufruire del laboratorio remoto. La possibilità di personalizzazione concessa da questa struttura risiede in due componenti in particolare tra quelli visti finora: Authenticator e Executor. Attraverso implementazioni diverse di questi due componenti è possibile far eseguire e supportare a WebLab cose molto diverse tra loro.
Authenticator gestisce e amministra le politiche di accesso. Una implementazione immediata e facilmente realizzabile consiste nell'integrare Au-

thenticator con il database del sito in Drupal per gestire l'autenticazione usando le credenziali che gli studenti hanno comunicato al momento della iscrizione al portale WebLab. Ma si potrebbe anche scegliere di fare in modi diversi, utilizzando ad esempio tabelle di gruppi di utenti con password preimpostate o scegliere ed impostare manualmente nomi utenti e password per gli studenti voluti.
Executor gestisce invece l'esecuzione del laboratorio remoto. Questo componente è quello che più di qualunque altro permette di personalizzare il comportamento dell'intero laboratorio. L'implementazione dei suoi metodi comanda ciò che accade nel laboratorio, sia esso hardware o sia esso software, e il tipo di risposta del laboratorio stesso.

8.3 Applicazioni hardware

Un esempio di applicazione hardware può essere quella di utilizzare Executor per far eseguire esercitazioni di movimento di robot fisicamente presenti ad esempio in una piccola arena, come il caso dei robot UniBot in Figura 8.1. Si presuppone che vi sia un sottosistema esistente che permetta a questi robot di muoversi rispondendo a determinati comandi in un determinato linguaggio.
Il docente preparerà quindi esercitazioni riguardanti questi robot, specificherà nel linguaggio usato dal sottosistema le impostazioni di setup e di successo di ogni esercizio, e metterà in esecuzione Lab Server. Ogni laboratorio a questo punto, prima di diventare utilizzabile, dovrà implementare i metodi di Executor per interfacciarli con il sottosistema che permette di muovere i robot.

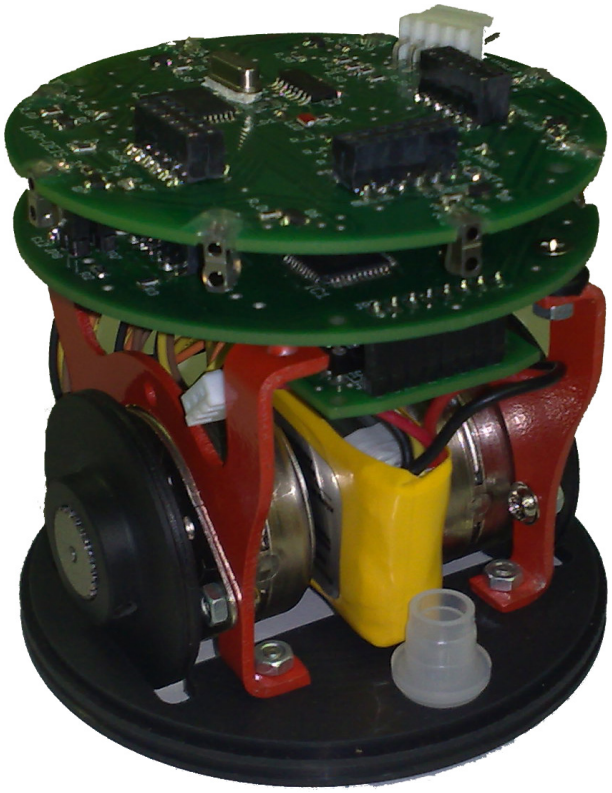

Figura 8.1: Il robot UniBot.

I metodi principali di Executor che andranno implementati sono:

- executeExercise() : legge il file soluzione postato dallo studente, in un linguaggio concordato, e lo traduce in istruzioni comprensibili al robot. Carica queste istruzioni sul robot e lo mette in esecuzione. Il robot si muoverà e la webcam presente invierà allo studente le immagini di quello che sta avvenendo nel laboratorio.
- evaluate() : al termine dell'esecuzione del codice inviato dallo studente, in base alle posizioni finali del robot o ad altri parametri (es il tempo impiegato o la velocità raggiunta), Executor valuterà il raggiungimento o meno delle condizioni di successo ed esprimerà un punteggio.
- resetLab(): in questo metodo andranno implementate quelle istruzioni che inviate al robot permettono di riportare il laboratorio nelle condizioni iniziali. Questo metodo viene richiamato al termine dell'esercitazione o in caso di interruzione da parte dell'utente.
- setupLab(): ogni esercizio può richiedere condizioni diverse dagli altri, presenti nel file setup definito nell'esercizio stesso. Il file setup conterrà le istruzioni per predisporre il robot nel modo corretto per l'esercizio.
- readStreamingData(): ogni fase del laboratorio può emettere una stringa di valori che verrà visualizzata nell'interfaccia di ClientGUI (ad esempio la posizione del robot o la sua velocità).
- generateReport(): questo metodo deve creare un file di report che verrà poi scaricato dallo studente ad esercitazione terminata. Implementando questo metodo si può decidere il tipo di informazioni che si vogliono inserire nel file di report.

Implementando questi metodi in maniera diversa si può quindi sfruttare l'ambiente messo a disposizione da WebLab per far accedere gli studenti e farli interagire con ogni tipo di robot o dispositivo possibile. Basta avere a disposizione un sottosistema software che comandi il robot e integrarlo in Executor.

8.4 Applicazioni software

Lo stesso procedimento appena descritto si può applicare in un altro contesto, diverso ma non per questo meno affascinante. Si può pensare di implementare i metodi appena descritti di Executor non per far muovere robot reali ma per esempio per comandare un software di simulazione robotica (ad esempio JUSE [1]) (Figura 8.2) o per far valutare, ad esempio, la soluzione di esercitazioni MatLab©.
La differenza rispetto al caso precedente, in cui Executor doveva interfacciarsi con un sottosistema software per muovere i robot, sta nel fatto che ora il sottosistema software non dovrà far muovere nessun dispositivo ma passare il file soluzione dell'utente ad un altra applicazione (MatLab©, JUSE o qualsiasi voglia), che elaborerà la soluzione, e valutarne il risultato. Nulla cambierà per l'utente che continuerà ad utilizzare WebLab per esaminare gli esercizi, caricare le proprie soluzioni e ricevere una valutazione.

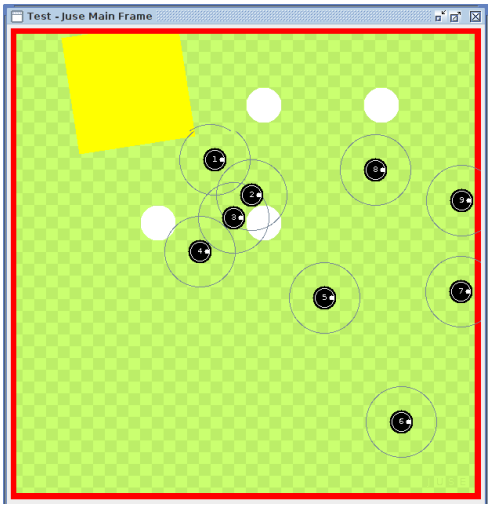

Figura 8.2: JUSE.

[1] JUSE: Java Unibot Simulation Environment di Raffaele Grandi. http://www-lar.deis.unibo.it/people/rgrandi/

Capitolo 9

Conlcusioni

Nel presente lavoro è stata presentata WebLab, una piattaforma web progettata e costruita per creare, gestire e controllare laboratori remoti o virtuali, ad esempio di robotica mobile. La progettazione e l'implementazione di WebLab è stata pensata ed organizzata per cercare di colmare una mancanza in ambito di educazione remota di robotica mobile: quella di un sistema il quanto più possibile aperto e modulabile che sia pronto e predisposto ad interfacciarsi con qualsiasi tipo di laboratorio (remoto o virtuale) in modo da poter permettere agli studenti di effettuare esercitazioni senza dover per forza essere presenti in laboratorio o aver installato sul proprio elaboratore il software necessario.
Si è organizzato e creato un ambiente in grado di gestire le connessioni di studenti attraverso una intuitiva interfaccia web senza bisogno di applicazioni esterne, in grado di creare e gestire una collezione di esercizi da parte dei docenti. Il tutto presentato all'interno di un portale web con la possibilità di osservare attraverso una webcam istante per istante quello che accade nel laboratorio remoto.
Il prossimo passo, già in cantiere, è quello di collegare e rendere attivo il laboratorio (remoto o virtuale che sia) implementando la parte di middleware che unirà la struttura WebLab all'infrastruttura software dedicata alla gestione del laboratorio. Le possibilità in questo campo sono molte visto che WebLab è progettato per controllare qualsiasi apparecchiatura o simulatore che si possa interfacciare con il linguaggio Java.

Appendice A

APPENDICE

A.1 Introduzione

In questa appendice verranno esaminati, attraverso alcune immagini esemplificative, alcuni esempi di uso dei vari sottosistemi dell'applicazione.

A.2 Authentication Sever

Authentication Server è, come detto, quella parte del sistema che coordina i restanti sottosistemi garantendo l'autenticazione di utenti e laboratori.

La sua interfaccia è molto semplice e intuitiva. Attraverso il pulsante START è possibile avviare il server che in questo modo si mette in ascolto per eventuali studenti o laboratori. Il pulsante STOP serve intuitivamente invece per interrompere l'ascolto del server.

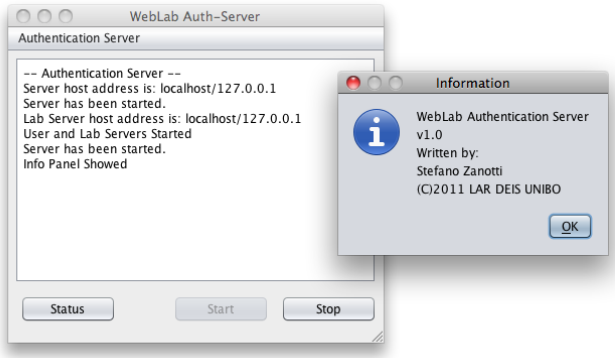

Figura A.1: Authentication Server.

Attraverso il pulsante STATUS si accede invece ad un pannello di controllo che permette di visualizzare in qualsiasi istante quanti e quali laboratori siano connessi e quanti e quali studenti si siano autenticati con successo, il loro indirizzo ip e il laboratorio a cui hanno scelto di connettersi. Authentication Server deve essere sempre online per permettere il corretto funzionamento del sistema e deve risiedere nella macchina che contiene le informazioni di autenticazione, per quel che riguarda il nostro caso deve risiedere e essere in esecuzione nella macchina sulla quale è installato il sito del laboratorio in modo da poter aver accesso ai dati degli studenti per poter verificare correttamente le loro credenziali.

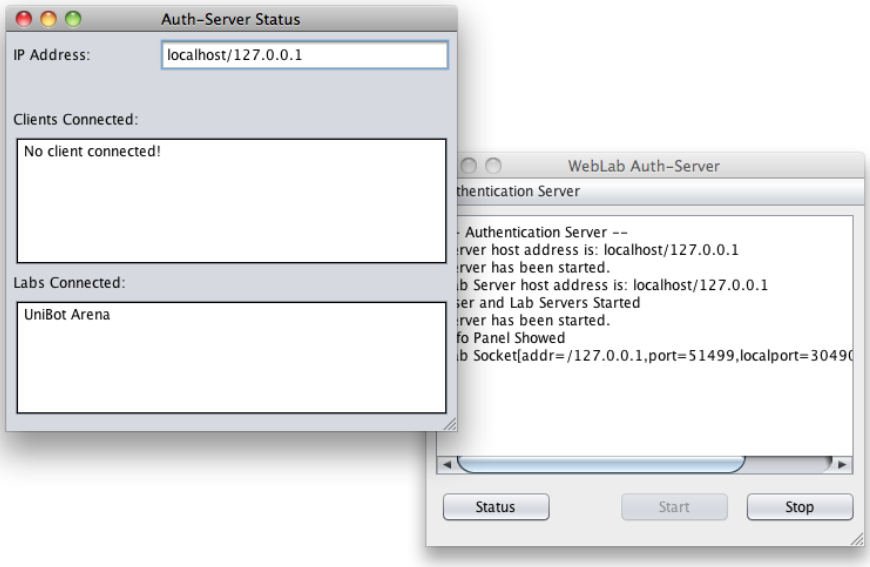

Figura A.2: Authentication Server: status.

A.3 Lab Sever

L'applicazione Lab Server è in realtà il cuore del sistema. Essa, come spiegato nei precedenti capitoli, svolge tutta la business logic interfacciandosi correttamente con gli studenti, ricevendo e gestendo i loro messaggi, permettendo la gestione degli esercizi da parte dei docenti e soprattutto gestendo tutta la concorrenza per l'accesso al laboratorio da parte degli studenti. Andiamo ora ad esaminare nel dettaglio alcuni aspetti pratici del suo funzionamento.

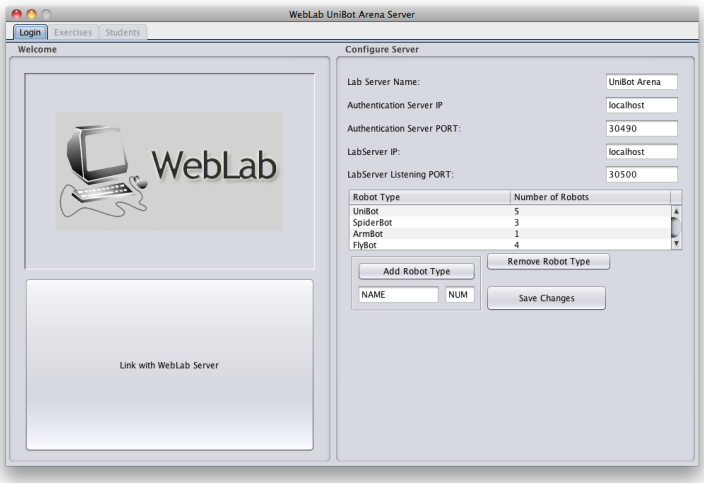

Figura A.3: Lab Server: login.

In Figura A.3 è riportata la schermata che si presenta all'avvio dell'applicazione. Nel pannello di sinistra è a disposizione dell'utente un pulsante LINK WITH WEBLAB SERVER che permette al laboratorio di mettersi in contatto con il server di autenticazione per registrarsi come laboratorio attivo. Da notare che nel caso si esegua per la prima volta l'applicazione tale pulsante non sarà attivo, poichè mancano le informazioni riguardanti il laboratorio che l'applicazione invierà poi al server di autenticazione.

A tal proposito il pannello di destra permette quindi di inserire tali informazioni, in particolare:

- Il nome del laboratorio
- L'indirizzo IP del laboratorio
- La porta di comunicazione del laboratorio
- L'indirizzo IP del server di comunicazione
- La porta di comunicazione del server di autenticazione(dati che andranno forniti dal responsabile di WebLab a chi chiede di poter mettere in esecuzione un laboratorio remoto)
- I tipi e i numeri di robot disponibili nel laboratorio

Cliccando infine sul pulsante SAVE CHANGES i dati del laboratorio saranno memorizzati su un file residente su disco in modo da essere recuperati al momento di una nuova esecuzione dell'applicazione.

Una volta registratasi correttamente con il server di autenticazione l'applicazione presenterà all'utente la schermata seguente, riportata in Figura A.4. Il funzionamento di questa parte dell'applicazione è in tutto e per tutto simile a quello di Authentication Server: cliccando sul pulsante START il Lab Server si metterà in ascolto per eventuali studenti mentre cliccando su STOP il server si fermerà. Da notare che una volta avviato il server lancia e mette in esecuzione anche il processo responsabile di amministrare la gestione della coda di acceso al laboratorio. Questo e qualsiasi altra cosa accada nel laboratorio è visualizzabile nell'ampia finestra di log, ad esempio in Figura A.4 la dicitura *** QUEUE SERVER STARTED *** indica appunto che il server ha avviato anche il processo che si occupa di gestire la coda.

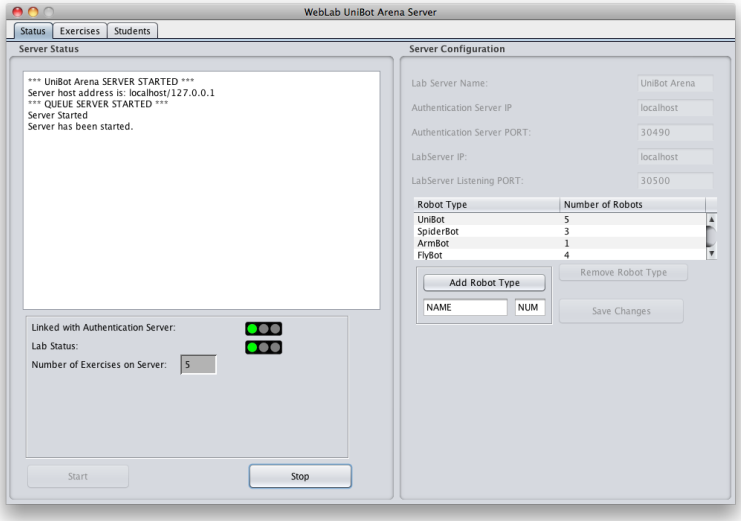

Figura A.4: Lab Server.

Il pannello sulla destra, denominato Server Configuration, permette di avere sotto occhio i dati con cui si è configurato il server in precedenza, mentre le due icone a forma di semaforo danno indicazioni sullo stato dei server e il numero subito sotto indica quanti esercizi sono al momento presenti nel laboratorio.
Infine le due etichette nella parte alta dell'interfaccia danno la possibilità all'utente di avere accesso alla gestione di studenti ed esercizi che ora andremo ad analizzare.

Gestione Esercizi

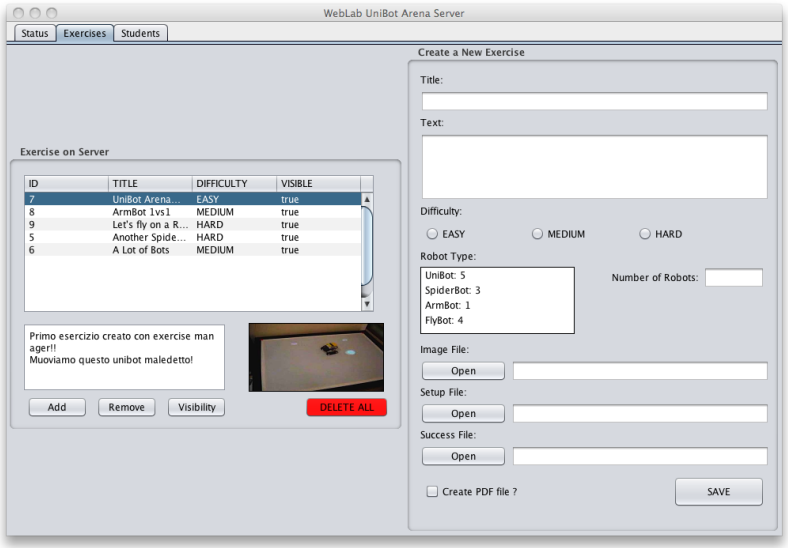

Figura A.5: Lab Server : esercizi.

Questa interfaccia permette al docente di gestire le esercitazioni presenti sul server. In particolare il pannello a sinistra permette di visionare tutti gli esercizi presenti nel laboratorio, di cancellarne uno col pulsante REMOVE, di cancellarli tutti col pulsante DELETE ALL o di occultarne alcuni utilizzando il pulsante CHANGE VISIBILITY in modo da renderli invisibili agli studenti. La parte destra dell'interfaccia permette invece di creare ed inserire nuovi esercizi nel server. L'interfaccia è abbastanza intuitiva e una volta completati tutti i campi l'esercizio viene inserito alla pressione del pulsante SAVE.

Da notare che attivando l'opzione CREATE PDF FILE al momento della pressione del pulsante SAVE verrà creato e salvato sul server anche un file PDF dell'esercizio.

Gestione Studenti

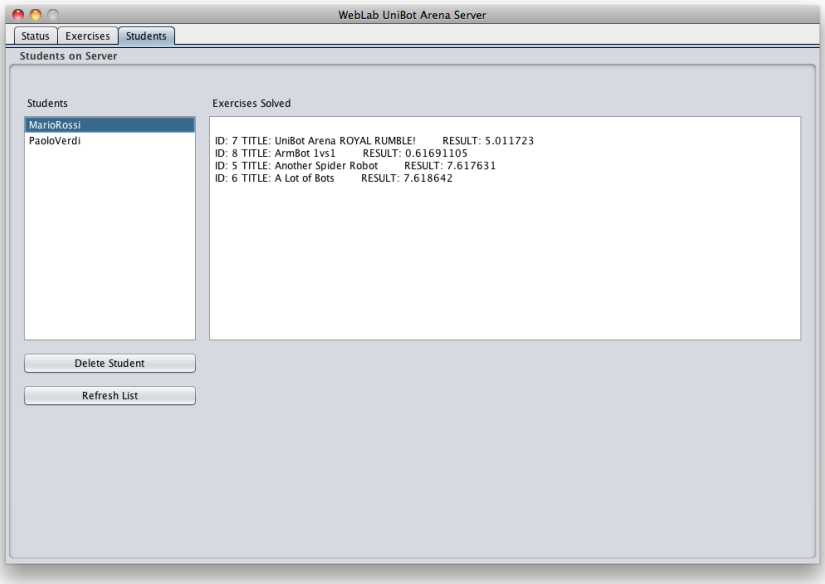

Figura A.6: Lab Server : studenti.

Questa parte dell'interfaccia permette ai docenti di esaminare quanti e quali studenti hanno partecipato al laboratorio virtuale, quali esercizi hanno svolto e quale è stata la loro valutazione. Attraverso l'interfaccia il docente può anche rimuovere la *carriera* di ogni studente attraverso il pulsante DELETE STUDENT.

A.4 Student Client GUI

Client GUI costituisce l'interfaccia utente che permetterà agli studenti di entrare nel laboratorio remoto. Essendo sviluppata come Applet Java, l'applicazione partirà non appena lo studente caricherà la relativa pagina sul sito.

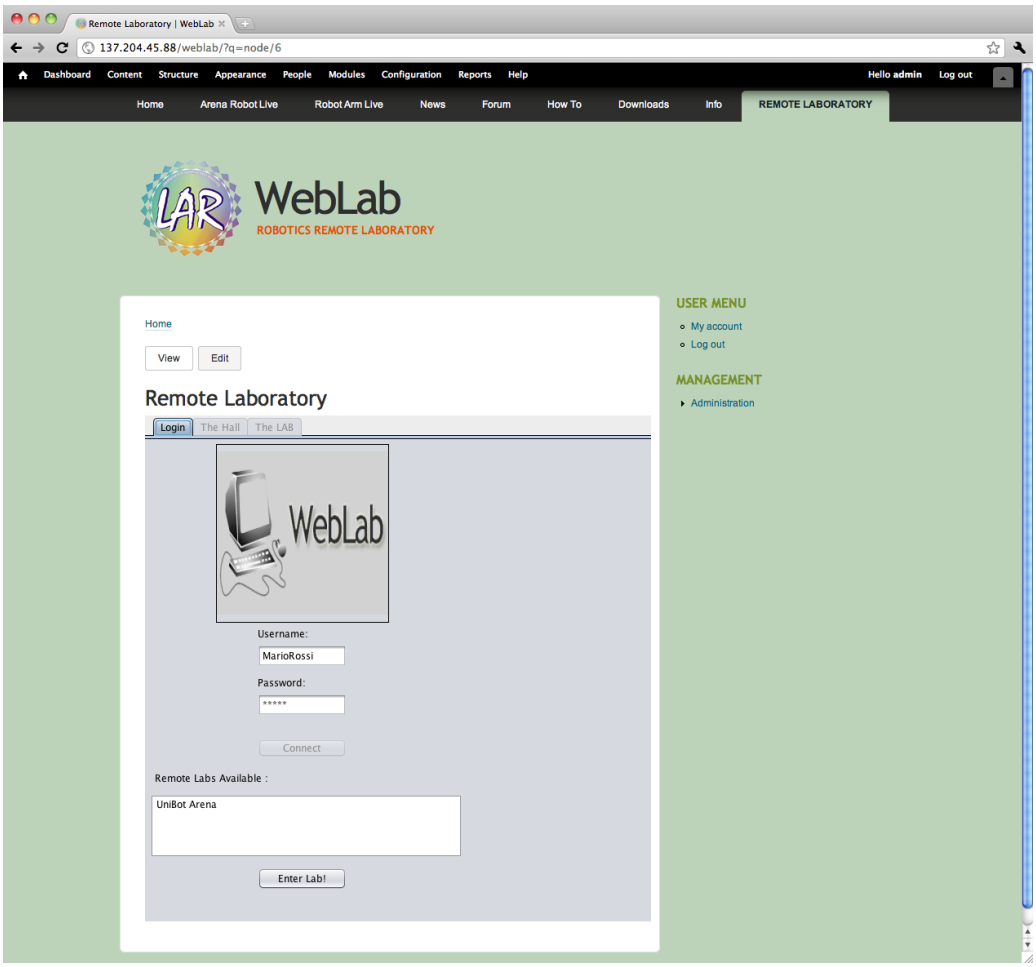

Figura A.7: Client GUI : login.

In Figura A.7 è rappresentata la schermata iniziale che lo studente troverà appena lanciata l'applicazione. Tale parte dell'interfaccia gestisce il login dello studente prima al server di autenticazione e poi al laboratorio virtuale prescelto. Dopo aver inserito i dati (username e password) e premuto CONNECT il sistema esaminerà le credenziali dell'utente e, nel caso venga correttamente riconosciuto, elencherà nello spazio sottostante i laboratori virtuali disponibili. Dopo aver cliccato su ENTER LAB l'utente verrà quindi messo in connessione con il laboratorio prescelto.

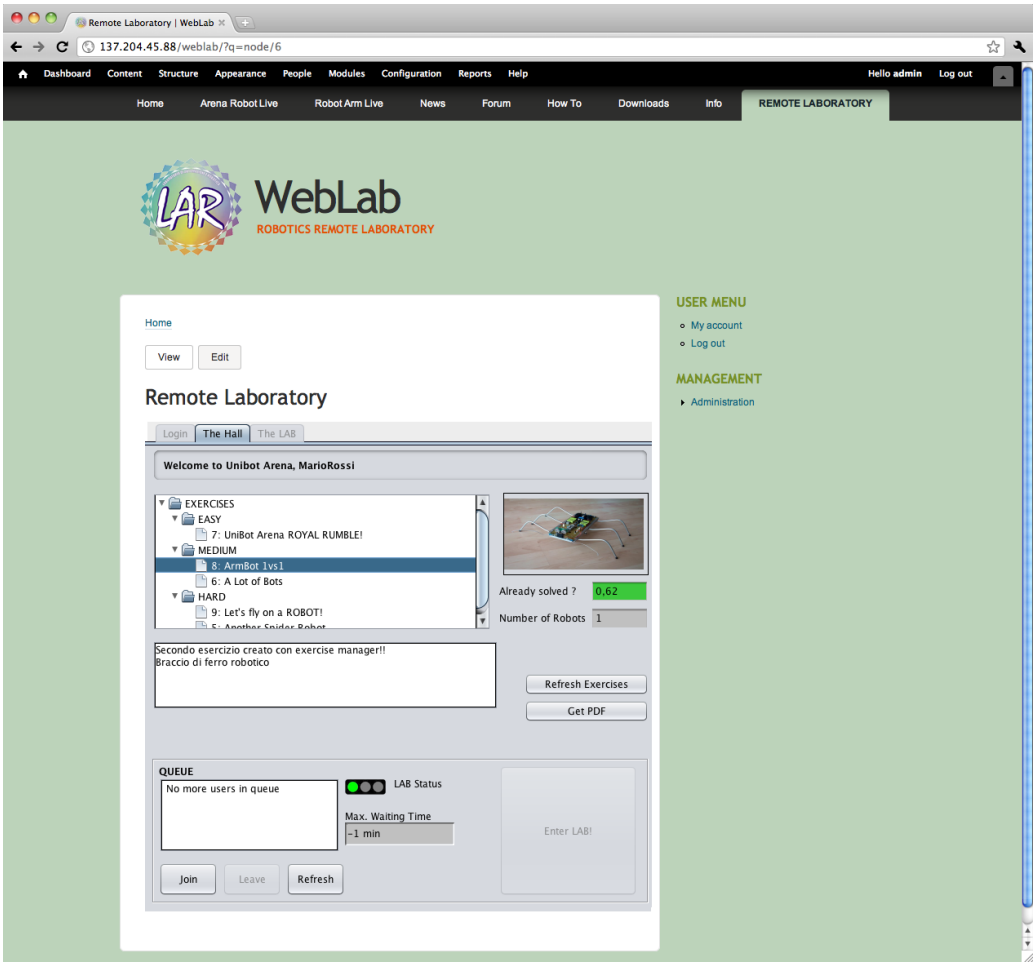

Figura A.8: Client GUI : hall.

Una volta connessa con il laboratorio remoto prescelto Client GUI visualizzerà l'interfaccia Hall del laboratorio (Figura A.8). La Hall del laboratorio rappresenta la camera d'aspetto: qui lo studente può esaminare gli esercizi disponibili, scaricare il PDF relativo cliccando sul bottone GET PDF o provare ad entrare nel laboratorio. Il laboratorio è gestito con una coda d'attesa. Cliccando su JOIN nel pannello QUEUE lo studente si mette in attesa. Cliccando su LEAVE lo studente decide di lasciare la coda lasciando spazio agli utenti in coda dopo di lui. L'icona a forma di semaforo indica lo stato del laboratorio remoto, in caso di semaforo rosso il laboratorio remoto risulta non accessibile (ad esempio per una manutenzione od un guasto tecnico). Premendo REFRESH lo studente può aggiornare lo stato della coda per controllare quanti utenti lo precedono o lo seguono in attesa.

Nel caso in cui dopo aver cliccato su JOIN la coda risulti vuota e il laboratorio libero o sia semplicemente giunto il suo turno per entrare nel laboratorio, lo studente riceve un avvertimento attraverso una finestra di dialogo che annuncia la necessità di effettuare l'accesso al laboratorio entro 60 secondi, pena la perdita della priorità (Figura A.9). Cliccando su OK e poi sul bottone ENTER LAB, che nel frattempo scandisce il count down, lo studente guadagna finalmente l'accesso al laboratorio remoto.

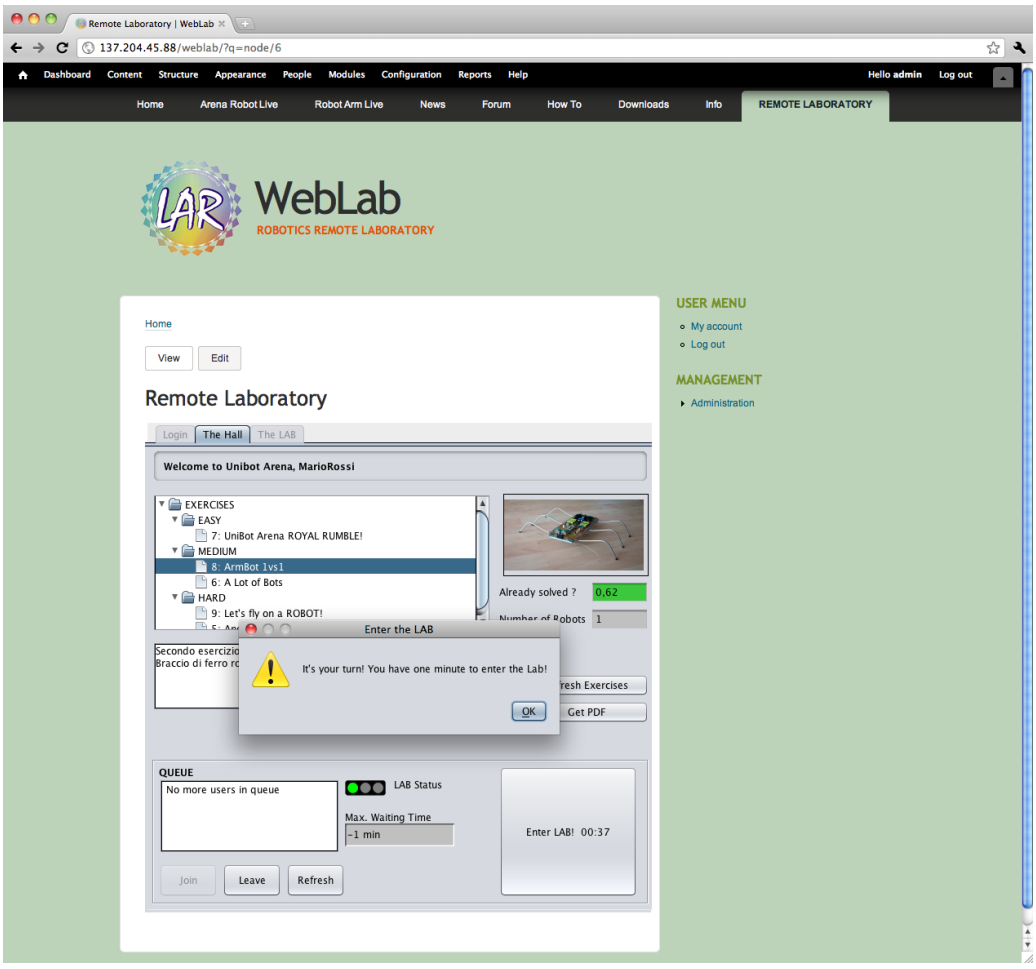

Figura A.9: Client GUI : entrando nel laboratorio.

L'interfaccia che permette all'utente di usufruire del laboratorio reale è rappresentata in Figura A.10 . In alto a sinistra trova spazio una rappresentazione in tempo reale del laboratorio fisico ottenuta tramite una webcam che inquadra la scena. Alla destra di questa immagine è presente un orologio che gestisce un count down prima che l'esperienza di laboratorio dello studente sia conclusa, riportando lo studente nella Hall e predisponendo il laboratorio per l'ingresso di un nuovo studente. Sotto all'orologio invece vi è un'area di testo che sarà utilizzata dall'applicazione per lo streaming live di eventuali dati inerenti l'esercitazione.

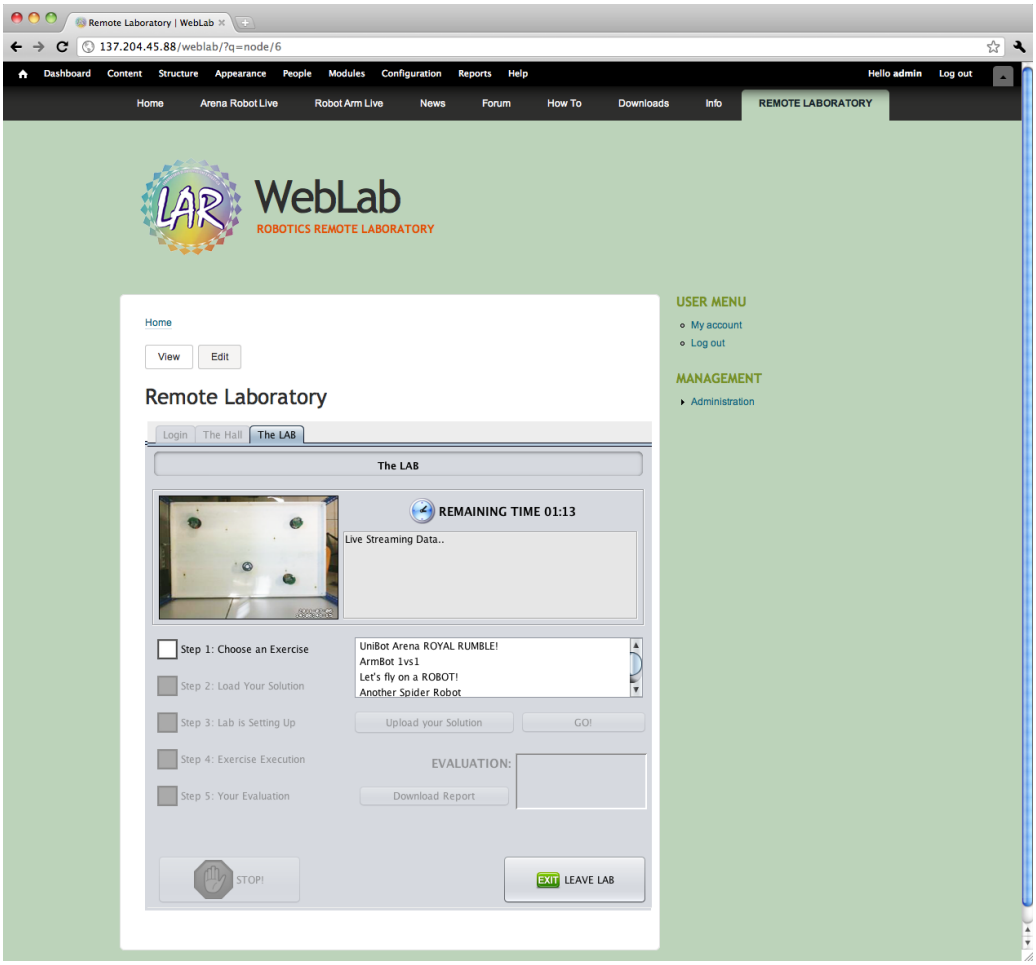

Figura A.10: Client GUI : lab.

Tutto questo rappresenta la parte che permette all'utente di visionare quello che accade nel laboratorio reale. Al di sotto si trova la parte dell'interfaccia che si occupa della gestione da parte dello studente della sua esperienza in laboratorio. Tale esperienza è divisa in due momenti distinti: un primo momento in cui lo studente sceglie l'esercizio da risolvere da una lista di esercizi disponibili e carica la sua soluzione, e un secondo momento in cui lo studente assiste allo svolgimento del proprio esercizio fino al suo completamento ricevendo dal sistema una valutazione e un file di report.

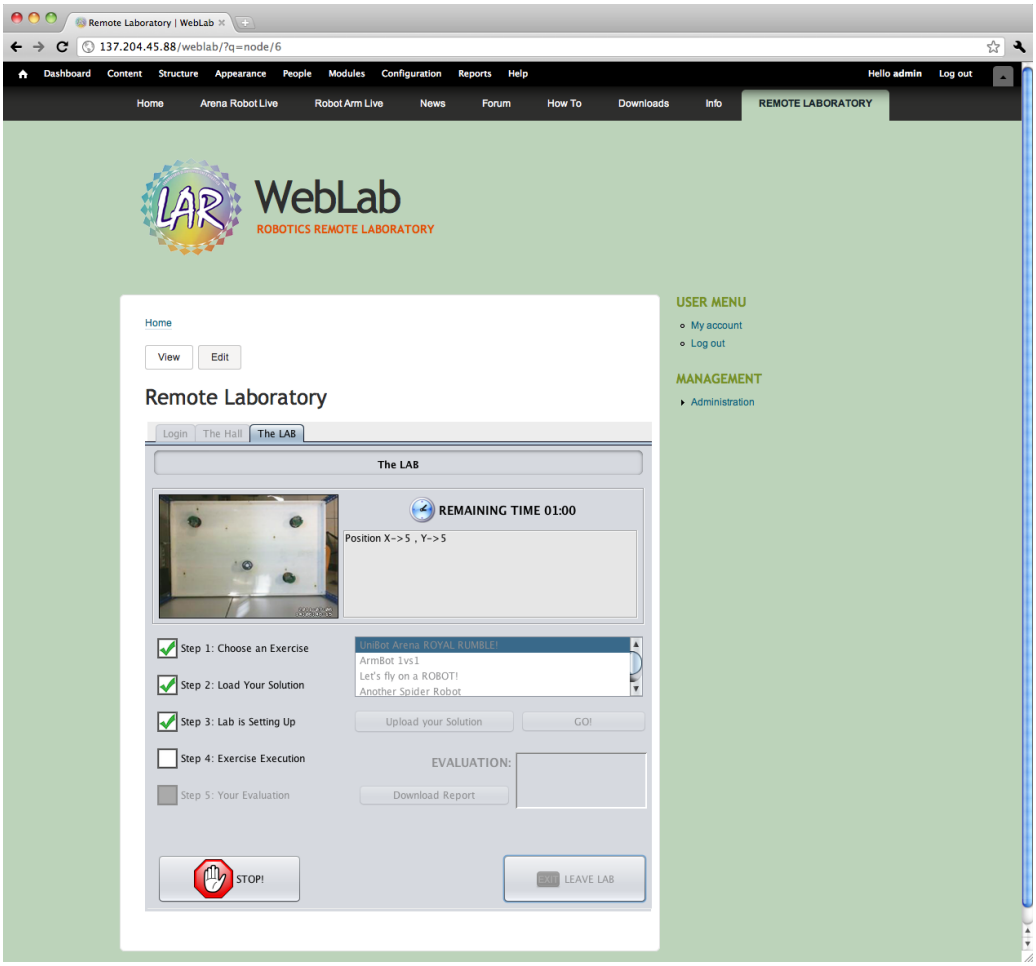

Figura A.11: Client GUI : il laboratorio in funzione.

La prima parte viene attivata nel momento in cui lo studente sceglie, evidenziandolo cliccandoci col mouse, un esercizio dalla lista. A questo punto il pulsante UPLOAD SOLUTION diventa attivo e lo studente può caricare una soluzione per l'esercizio scelto. Una volta caricata la soluzione il pulsante GO diventa attivo e spingendolo l'utente da definitivamente il via alla esercitazione.

A questo punto nella parte sinistra dell'interfaccia verranno completati ad uno ad uno i punti evidenziati mentre lo studente potrà osservare il progredire dell'esperienza grazie alla webcam e all'area di testo soprastante. Da notare che durante tutto lo svolgimento dell'esercitazione lo studente potrà interrompere il tutto in qualsiasi momento attraverso il bottone STOP in basso a sinistra.

Ad esercitazione terminata comparirà nello spazio EVALUATION una valutazione numerica dell'esercitazione compresa tra 1 e 10 e sarà possibile per lo studente scaricare un report cliccando il pulsante DOWNLOAD REPORT. A questo punto col pulsante LEAVE LAB lo studente può abbandonare il laboratorio e tornare nella Hall.

Bibliografia

[1] Oussmane Khatib Bruno Siciliano. *Springer Handbook of Robotics*. Springer, 2008.

[2] M. Casini, D. Prattichizzo, and A. Vicino. The automatic control telelab. *Control Systems, IEEE*, 24(3):36 – 44, jun 2004.

[3] A. Molesini A. Natali. *Costruire sistemi software: dai modelli al codice*. Progetto Leonardo Bologna, 2009.

[4] Douglas C. Shmidt Frank Buschmann, Kevlin Henney. *Pattern Oriented Software Architecture - Volume 4*. John Wiley and Sons, 2007.

[5] A.Riccioni R. Laschi, R.Montanari. *Sicurezza dell'informazione*. Progetto Leonardo Bologna, 2008.

[6] Ivar Jacobson. *Object-Oriented Software Engineering. A Use Case Driven Approach*. Addison-Wesley, 1992.

Ringraziamenti

Think about the good times
And never look back
Never look back
What would I do?
What would I do?
If I did not have you?"
Radiohead, *I Might be Wrong.*

Gino ed Ines.
I miei genitori, non riesco a pensare a nessun altro prima di loro che io possa e debba ringraziare per tutto quello che hanno fatto per me, per la pazienza che hanno avuto nell'aspettarmi e per non aver mai smesso, nemmeno per un solo secondo, di credere in me, anche in questa lunghissima e interminabile odissea universitaria. Questa laurea è vostra quanto mia. Spero di avervi reso orgogliosi in questo giorno, anche se con anni e anni di ritardo, perchè io sono sempre stato orgoglioso di essere vostro figlio.

Francesca.
Non so se sarei mai riuscito ad arrivare alla fine senza di te. Una grossa fetta di questa laurea è anche tua. Sei stata la molla che mi ha fatto scattare, mi hai dato, senza saperlo, una ragione vera per finire. Mi hai sempre spronato, hai sempre creduto in me e mi hai ripetuto in continuazione che ce la potevo fare. Sei stata la mia motivazione in ogni momento. Ti amo.
Riccardo.
Non posso non ringraziare anche te, Ric, mio compagno di studi da tempi ormai lontani. È incredibile come si sia alla fine conclusa questa mia

carriera universitaria, quasi con una morale, avendo te come correlatore. Ma ti posso assicurare che, ripensandoci ora e pensando agli anni passati insieme a Bologna, non riesco ad immaginare maniera migliore di chiudere il tutto. Ti ringrazio per come mi hai aiutato e sostenuto negli ultimi anni e in generale per tutti gli anni bellissimi passati insieme in appartamento. Li ricorderò per sempre con tanta nostalgia.

Un ringraziamento particolare al Professor Claudio Melchiorri per la sua diponibilità e cortesia. Un grazie anche a Raffaele Grandi che mi ha seguito ed aiutato a risolvere alcune difficoltà e problemi tecnici dimostrandosi sempre presente e disponibile.

E poi tutti gli altri, amici e parenti, che mi sono sempre stati vicini e che mi supportano e incoraggiano ogni giorno con il loro affetto e la loro simpatia. Vi voglio bene. Grazie di tutto.