

# **“Software Engineering”**

## **Course**

### **a.a. 2016-2017**

#### **Template version 1.0**

**Lecturer: Prof. Henry Muccini (henry.muccini@univaq.it)**

# **Planner Path Calculator**

## **version v2**

### **Deliverables**

<b>Date</b>	23/12/2016
<b>Deliverable</b>	<i>Deliverable 2</i>
<b>Team (Name)</b>	Indifferente

<b>Team Members</b>		
<b>Name &amp; Surname</b>	<b>Matriculation Number</b>	<b>E-mail address</b>
Tommaso Di Salle	236202	<a href="mailto:Tommasodisalle@gmail.com">Tommasodisalle@gmail.com</a>
Luca D’Orazio	227635	<a href="mailto:Lucaadorazio@gmail.com">Lucaadorazio@gmail.com</a>
Stefano Corsetti	227288	<a href="mailto:s.corsetti@hotmail.it">s.corsetti@hotmail.it</a>
Francesco Di Cato	203356	<a href="mailto:dicatofrancesco@gmail.com">dicatofrancesco@gmail.com</a>
Eugenio Mancini	230024	<a href="mailto:Emancini1992@libero.it">Emancini1992@libero.it</a>

# Index

- 1. List of Challenging/ Risk requirements or Task (pag.4)**
- A. Requirements Collection & Description (pag. 5)**
  - A.1 List Functional Requirements (pag.5-9 )**
    - A1.1 GUI Requirements**
    - A1.2 Business Logic Requirements**
    - A1.3 DB Requirements**
  - A.2 Non Functional Requirements (pag.10 )**
  - A.3 Content (pag.10 )**
  - A.4 Assumption (pag. 10)**
  - A.5 Prioritization (pag. 11)**
  - A.6 UseCase Diagram (pag.12 )**
  - A.7 Scenari (pag. 13 )**
  - A.8 Descrizione UseCase (pag.14-19 )**
- B. Analysis Model (pag.20-25 )**
  - B.1 Robustness Diagram**
- C. Software Architecture (pag. 26)**
  - C.1 Component Diagram (pag.26-27 )**
  - C.2 Sequence Diagram (pag. 28-35)**
- D. ER Design (pag. 36)**
- E. Class Diagram of the implemented System (pag.37-38 )**
- F. Design Decision (pag.39-40 )**
- G. FRs and the NFRs (pag.41-42 )**
- H. Effort recording (pag. 43 )**

## Table of Contents of this deliverable

-Nella Deliverable v1 abbiamo affrontato i punti A,B,C,F e H.

In questi punti si chiedeva di riportare la collezione dei requisiti (requisiti funzionali, scenari, assunzioni e prioritizzazioni) , l'Analisi dei modelli con il diagramma Robustness, l'Architettura software con il relativo diagramma di Componenti, le Design Decision ed infine una tabella con le ore spese per questo primo step d progetto.

-Nella Deliverable v2 abbiamo affrontato i punti A,B,C,D,E,F,G,H.

In questi punti, invece, si chiedeva di riportare, oltre alla revisione dei punti svolti nello step precedente, anche i requisiti non funzionali, ER Design, Class Diagram, Sequence Diagram e un prototipo della nostra applicazione

## 2.List of Challenging/Risky Requirements or Tasks

Challenging Task	Date the task is identified	Date the challenge is resolved	Explanation on how the challenge has been managed
Utilizzo Software MagicDraw	8/11/2016	8/11/2016	Consultazione manuale online poichè non trattato nello specifico a lezione
Dove eseguire calcolo: database o server?	2/12/2016	6/12/2016	Discussione in gruppo
Linguaggio di programmazione da utilizzare	5/12/2016	17/12/2016	Ricerca da fonti varie su web
Tipo di Database da utilizzare	5/12/2016	5/12/2016	Discussione di gruppo e ricerche
Connessione Client-Server ed invio dati	5/12/2016	23/12/2016	Ricerca da fonti varie su web
Struttura dati Albero	2/12/2016	2/12/2016	Il team ha operato con struttura ad array
Architettura SW	22/11/2016	26/11/2016	Approfondimenti su libro ti testo, slide ed internet su Component Diagram.
Document/ Documentazione	24/11/2016	23/12/2016	Il team ha documentato tutte le scelte fatte in linguaggio natural

# A. Requirements Collection

## *A.1 Functional Requirements*

---

### **-Lista Requisiti funzionali**

- 1- Creazione Albero
  - 1.1-inserimento parametri
  - 1.2- Salva Albero
- 2- Seleziona Albero
  - 2.1- Carica Albero
  - 2.2- Cancellazione
- 3- Calcolo Somma
  - 3.1- Scelta due vertici
  - 3.2somma tra i 2 vertici
- 4- tempo di calcolo

### **-Descrizione Requisiti funzionali**

I numeri (\*) rispettano l'ordine della lista dei requisiti funzionali.

Dall'analisi dei requisiti e dal file condiviso abbiamo estrapolato i seguenti R.F. che possono essere raggruppati in :

Creazione, caricamento, cancellazione, selezione e calcolo.

Creazione(1) dedotto da **“The PPC must have a GUI that generates a tree structure using the following parameters”**.

SalvaAlbero(1.2) :L'albero verrà generato e salvato cliccando sul bottone “buildtree”. Requisito dedotto da “The GUI can be used to generate a tree when the “Build Tree” button is pressed. The result is saved in the database.”

Carica Albero(2.1): l'utente può scegliere un albero precedentemente salvato. Dedotto da “The GUI can be also used to retrieve from the database a Tree previously stored”

Cancellazione(2.2) dedotto dal file id domanda 6 con risposta “L'albero può essere inserito, cancellato e visualizzato. Non serve modificarlo.”

Selezione(2): dedotta dal file id domanda 6(“L'albero può essere inserito, cancellato e visualizzato. Non serve modificarlo.”) e da id domanda 22(“Non è necessario visualizzare l'albero in quanto stiamo parlando di alberi potenzialmente grandissimi. Se trovate un modo efficiente di farlo va benissimo, ma consideratela una cosa opzionale. Quello che è richiesto invece è visualizzare il risultato dell'operazione di calcolo con la lista dei vertici attraversati (es: Vertice3, Vertice4, Vertice 5. Ptime=50 Cost=210)”). Abbiamo quindi deciso di inserire una funzionalità antecedente quella di caricamento e cancellazione in cui l'utente avrà una descrizione generale dell'albero(altezza, splitsize, numero totale nodi etc,) e da questa potrà scegliere se cancellare o modificare l'albero.

CalcoloSomma(3): che raggruppa Scelta di 2 vertici (3.1) e somma tra i 2 vertici(3.2) dedotto da “users can select **2 Vertices A and B, and the system must return the list of vertices from A to B among with the SUM of each attribute**

Tempo di calcolo(4) dedotto da “The GUI will also display the time the system took to make this calculation” \*\*.

\*\* per tempo di calcolo non intendiamo che il Sistema deve essere veloce a ritornare il risultato(NFR) ma che come funzionalità deve riportare a video il tempo impiegato

---

### *A1.1 GUI Requirements*

La gui dovrà provvedere a fornire le seguenti funzionalità:

- Generazione albero premendo il bottone "build tree";
- Inserimento parametri su nodi e archi premendo rispettivamente i bottoni "AggiungiAttrNodo" e " AggiungiAttrArco" ;
- Scelta dei due nodi (Nodo1 e Nodo2) con due input;
- Calcolo Somma premendo il pulsante "Calcola";
- Cancellazione Albero premendo il bottone "Cancella";
- Bottone "Lista" per visualizzare l'elenco degli alberi presenti nel database
- Bottone "Seleziona" per effettuare l'operazione di calcolo sull'albero selezionato
- Definito dall'utente un percorso (Nodo(i) e Nodo(j)) mi fa vedere la lista dei nodi coinvolti nell'operazione

### *A1.2 Business Logic Requirements*

---



La Business Logic svolgerà le funzioni di:

- Creare un albero secondo i parametri inseriti in input dall'utente;
- Cancellare un albero presente nel db;
- selezionare un albero tra quelli presenti
- effettuare il calcolo della somma degli attributi selezionati dal nodo di partenza al nodo di destinazione e riportare l'elenco dei nodi percorsi

### *A1.3 DB Requirements*

Il DB deve essere sempre accessibile e mantenere le informazioni precedentemente memorizzate

### *A.2 Non Functional Requirements*

- Il Sistema deve essere veloce a ritornare i risultati;
- Gestire 10/100 utenti senza subire cali di prestazioni;
- Gestire alberi fino a 2000000 di nodi;
- Gui in html5;
- La sicurezza non deve essere garantita;

### *A.3 Content*

---

- 1) I dati verranno immessi dall' utente per creare un nuovo albero.
- 2) Per recuperare un albero, i dati verranno estratti da un database.

### *A.4 Assumptions*

---

- 1) Assumiamo che il Sistema sia collegato in rete.
- 2) Assumiamo che il Sistema disponga di risorse di calcolo adeguate.
- 3) Si assume che il calcolo venga effettuato tra due nodi di cui uno deve essere antenato del secondo

### *A.5 Prioritization*

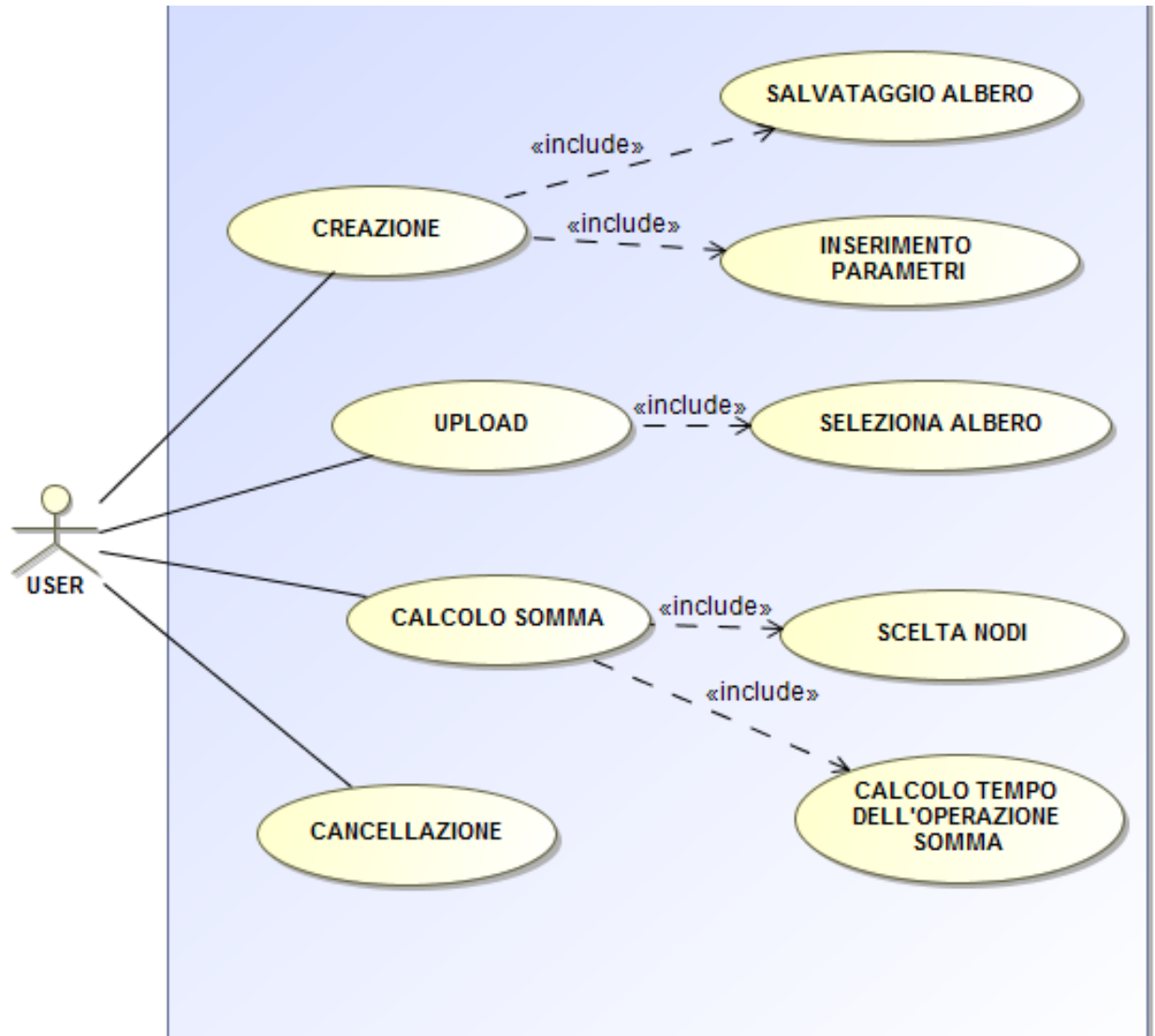
*Requisiti listati secondo ordine di priorità.*

Ordine Priorità	Requirement	Priorità	Motivazione
1	CalcoloSomma	High	Funzione indispensabile del sistema.
2	Creazione Albero	High	Senza questa funzionalità, si creerebbero alberi con poca utilità per l'utente.
3	InserimentoParametri	Medium	Funzionalità indispensabile richiesta dal cliente.
2	Caricamento	Medium	È necessario per eseguire altre funzionalità. Permette di caricare alberi già presenti nel sistema,
4	Cancellazione	Low	È una funzionalità opzionale, poiché senza di essa è comunque possibile effettuare le altre funzionalità (operazioni).

#### A.6 USE-CASE Diagram

Abbiamo deciso di fermarci ad un alto livello di astrazione. Abbiamo identificato solo i 4 use-case principali che sono emersi dai vari scenari

ipotizzati e alcuni di questi sono stati esplosi con le funzionalità che riteniamo più importanti.



#### A.7 Scenari :

Di seguito alcuni possibili scenari:

1- CREA:

l'utente inserisce i parametri dell'albero ( nomi, split-size e altezza ).

Sceglie la lista di attributi da inserire a nodi ed archi ed il range di valori da assegnare ad ogni attributo. Cliccando sul pulsante BUILD-TREE verrà generato l'albero con tali proprietà e salvato.

2- SELEZIONA:

l'utente seleziona l'albero da un elenco presente nella lista e avrà informazioni sull' albero

3- CALCOLO SOMMA:

l'utente sceglierà 2 vertici. Tra questi 2 vertici verrà calcolata la somma di tutti gli attributi su archi e nodi e il risultato verrà stampato a video insieme al tempo impiegato per effettuare l'operazione

NOTA :

Qui è stata messa come funzionalità anche il tempo dove intendiamo il calcolo del tempo che il sistema impiega per calcolare la somma

4- CANCELLAZIONE

L'utente cliccando sul bottone "cancella" elimina l'albero selezionato dall'elenco

### *A.8 UseCase Description*

USE CASE	Creazione
----------	-----------

<b>Goal in Context</b>	Generazione e Salvataggio Albero	
<b>Scope &amp; Level</b>	L'utente sceglie i valori da assegnare all'albero e quest'ultimo viene salvato	
<b>Preconditions</b>	Inserimento Parametri corretti	
<b>Success End Condition</b>	L'albero viene generato	
<b>Failed End Condition</b>	(Notifica di errore)	
<b>Primary Actor</b>	USER	
<b>Trigger</b>	Click sul pulsante "CREA"	
<b>DESCRIPTION</b>	<b>STEP</b>	<b>ACTION</b>
	1	L'utente inserisce i parametri (split-size, altezza, nome, lista attributi nodi/archi e range valori per ogni attributo)
	2	Salvataggio albero generato
<b>RELATED INFORMATION</b>	Creazione	

<b>Priority</b>	High
<b>Performance</b>	< 2 ore
<b>Frequency</b>	1 volta a settimana per utente
<b>OPEN ISSUES</b>	<ul style="list-style-type: none"> <li>• Metodologia salvataggio albero</li> <li>• Metodologia rappresentazione albero</li> </ul>

<b>USE CASE</b>	Seleziona Albero
<b>Goal in Context</b>	Selezione di un albero tra quelli presenti

<b>Scope &amp; Level</b>	L'utente cliccando su un albero tra la lista di quelli presenti recupererà le informazioni ad esso associate (numero nodi, altezza, lista attribute, etc..)	
<b>Preconditions</b>	L'utente ha cliccato su "Lista"	
<b>Success End Condition</b>	è stato selezionato un albero	
<b>Failed End Condition</b>	(notifica di errore)	
<b>Primary Actor</b>	USER	
<b>Trigger</b>	Click su albero	
<b>DESCRIPTION</b>	<b>STEP</b>	<b>ACTION</b>
	1	Click per la selezione di un albero

<b>RELATED INFORMATION</b>	Seleziona albero
<b>Priority</b>	Low



<b>Performance</b>	< 5 sec
<b>Frequency</b>	(caricamento) 3 volte al giorno per utente (cancellazione) 1 volta a settimana per utente
<b>Superordinates</b>	<ul style="list-style-type: none"> <li>• Caricamento</li> <li>• Cancellazione</li> </ul>

<b>USE CASE</b>	Calcolo Somma
<b>Goal in Context</b>	Somma sugli attributi dei nodi selezionati
<b>Scope &amp; Level</b>	il Sistema fornirà la lista dei vertici tra i nodi selezionati e la somma dei valori degli attributi su nodi/archi di tale lista

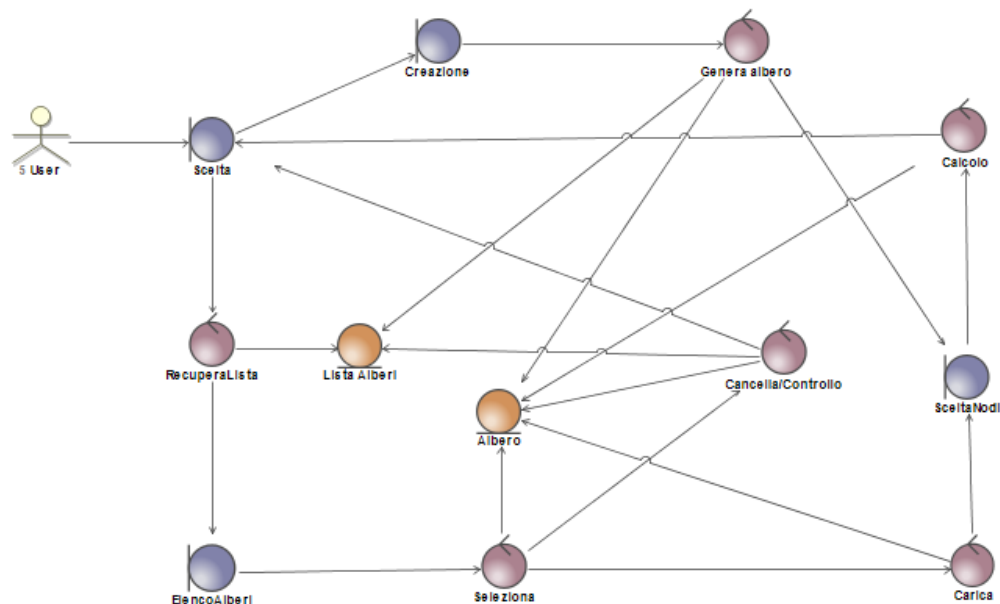
<b>Preconditions</b>	L'utente ha cliccato sul bottone "carica"	
<b>Success End Condition</b>	Stampa a video il risultato della somma e il tempo impiegato per calcolarlo	
<b>Failed End Condition</b>	Notifica Errore	
<b>Primary Actor</b>	USER	
<b>Trigger</b>	Click sul pulsante "SOMMA"	
<b>DESCRIPTION</b>	<b>STEP</b>	<b>ACTION</b>
	1	L'utente sceglie due nodi
	2	Clicca sul bottone calcola
	3	Il sistema ritorna a video la lista dei vertici, il risultato della somma ed il tempo impiegato a calcolarla

<b>RELATED INFORMATION</b>	Calcolo somma
<b>Priority</b>	High

<b>Performance</b>	< 30 sec    per 1 milione di nodi  <60 sec    per 2 milioni di nodi
<b>Frequency</b>	3 volte al giorno per utente
<b>OPEN ISSUES</b>	Le operazioni verranno fatte sul DB o sul server
<b>Subordinates</b>	<ul style="list-style-type: none"><li>• Scelta nodi</li><li>• Tempo</li></ul>

## B. Analysis Model

### B.1 Robustness Diagram



Il team ha prodotto il diagramma di Analysis Model affrontando diverse decisioni.

Come boundary abbiamo identificato i seguenti oggetti:  
Scelta, form Creazione, Elenco Alberi, Albero selezionato, Scelta nodi, Risultato.

Come controller, abbiamo identificato i seguenti oggetti: Recupera lista, Seleziona, Carica, Cancella/Controllo, Calcolo, Genera albero.

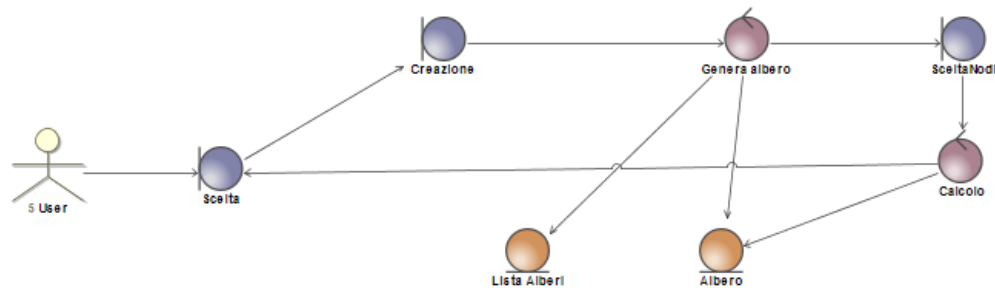
Come entity abbiamo identificato i seguenti oggetti: Lista alberi ed Albero.

Come primo step l'utente si interfaccia con una schermata, identificata dal boundary "Scelta", che rappresenta la pagina iniziale della nostra applicazione. L'utente avrà due opzioni: creare un nuovo albero cliccando sul bottone "Crea", oppure visualizzare la lista degli alberi presenti nel Database, cliccando sul bottone "Lista".

Ramifichiamo la spiegazione in base al bottone premuto e riportando il relativo diagramma.

### Caso1

Bottone Crea:



-Form creazione: Boundary Object dove l'utente inserisce i parametri per la generazione dell'albero.

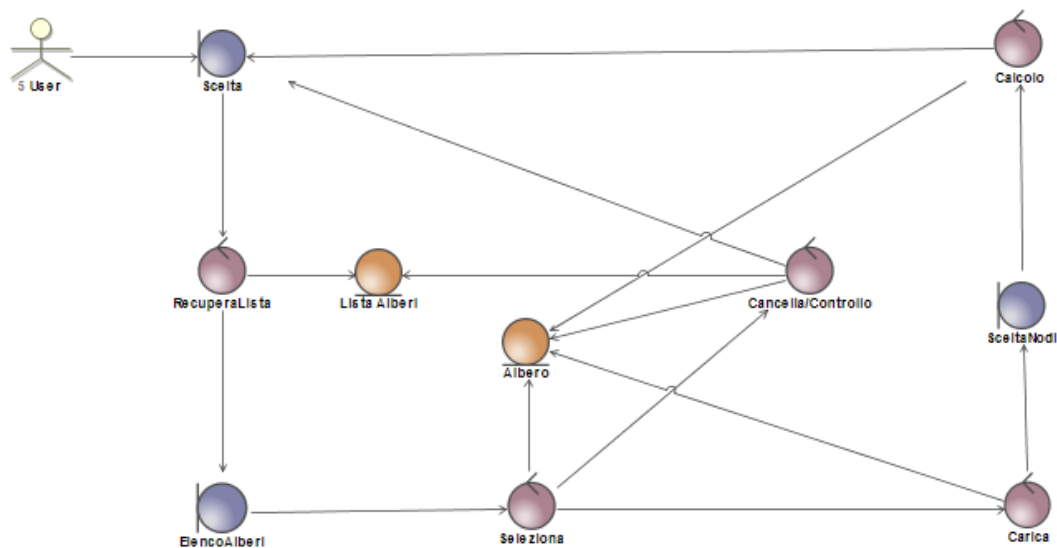
-GeneraAlbero: Controller che prende i parametri inseriti in FormCreazione e crea l'albero

#### • descrizione funzionalità :

L'utente una volta generato l'albero può scegliere i nodi per il calcolo della somma. "SceltaNodi" va nel controller "Calcolo" per questo passaggio, per poi arrivare nel boundary "Risultato" dove potrà visualizzare il risultato atteso. Adesso, da questa schermata (dove compare il risultato della somma) si può tornare nella scelta dei nodi (vedi collegamento tra i due boundary "Risultato" e "SceltaNodi") oppure tornare alla pagina iniziale (vedi collegamento tra Boundary "Risultato" con "Scelta" )

## Caso2

Bottone Lista:



-RecuperaLista: l'utente, precedentemente ha cliccato su "Lista", questo Controller ha la funzionalità di recuperare le informazioni presenti in "ListaAlberi"

-ElencoAlberi: Entity che mostra l'elenco degli alberi presenti in "ListaAlberi". Cliccando su un albero presente nella lista entra in funzione il controller "Selezione"

-Selezione: Controller che recupera informazioni sull'albero

-Albero Selezionato: Entity che mostra le informazioni dell'albero precedentemente selezionato. Avrà due bottoni, di cui uno "Carica" e uno "Cancella".

-Carica: Controller precedentemente citato, che ha la funzione di caricare l'albero

-Cancella: Controller che ha la funzione di eliminare definitivamente l'albero precedentemente selezionato

Object in comune tra i 2 percorsi:

-ListaAlberi: Entity Object in cui è inserita la lista dei nomi degli alberi presenti.

-Albero: Entity Object in cui è inserita la lista dei nodi ed il valore degli attributi

---

-SceltaNodi: Entity Object dove l'utente sceglie i due vertici e gli attributi su cui effettuare la somma

-Calcolo: Controller Object che ha la funzione di effettuare il calcolo della somma e l'elenco dei nodi tra i due selezionati

-Risultato: Boundary Object che mostra a video il risultato calcolato dal controller "Calcolo" ed il tempo impiegato per effettuare questa operazione.

Qui ci sono due bottoni "Pagina iniziale" e "Ricalcola".

Cliccando su "PaginaIniziale" l'utente viene portato nell'Entity "Scelta".

Cliccando su "Ricalcola" l'utente viene portato nell'Entity "SceltaNodi" ed effettuare nuovamente calcoli sull'albero precedentemente selezionato

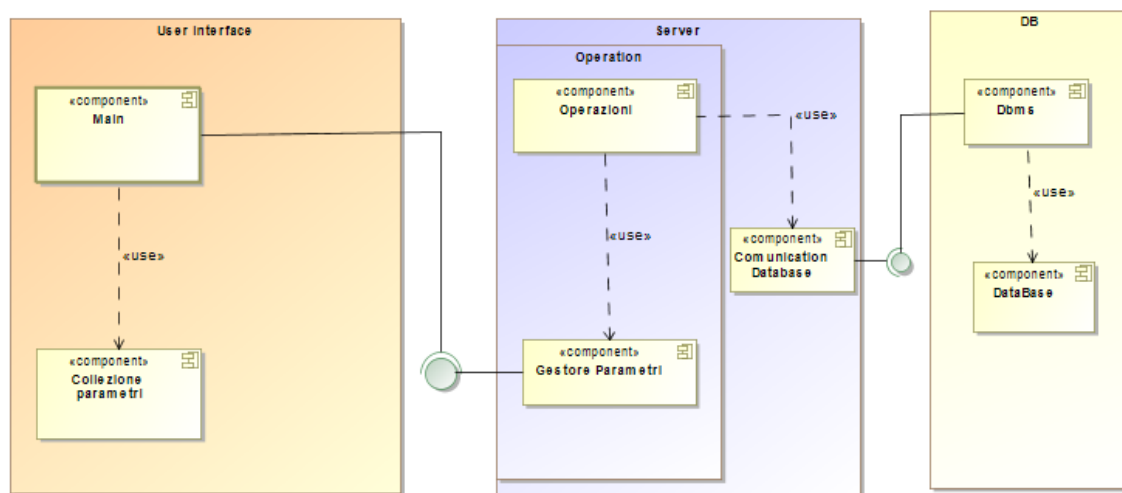


- descrizione funzionalità poco chiara:

Una volta selezionato l'albero ("Albero Selezionato") abbiamo due strade: una è la cancellazione IMMEDIATA di tale albero (recuperando l'albero che abbiamo identificato con l'entità "Albero" e quindi tornare alla schermata principale), mentre la seconda è caricarlo visualizzando le sue proprietà, per poi tornare alla sceltaNodi (vedi collegamento tra controller "Carica" e boundary "SceltaNodi") quindi effettuare il calcolo della somma su i nodi scelti.

## C. Software Architecture

### C.1 *The static view of the system: Component Diagram*



#### C1.1 *Documentazione component diagram*

Nel design del sistema abbiamo utilizzato un architettura client/server. La componente UserInterface(U.I.) richiede dei servizi alla componente Server che risponderà alle richieste interrogando la componente DB. Quindi ci sono 3 macro: U.I., Server e DB.

##### 1) User Interface:

la component U.I. è l'interfaccia con cui comunica l'utente.

A sua volta è stata divisa in 2 component:

La prima, chiamata browser, si occuperà di creare/gestire le finestre e da interfaccia con il Server.

A sua volta il Browser gestisce un sottosistema Main il quale si occupa di far visualizzare le pagine in cui l'utente non può inserire/ricevere dati e le pagine in cui l'utente inserisce/riceve dati dalla componente Server.

2) Il Server è la componente che si occupa di servire le richieste della U.I. E' la parte principale del nostro sistema. E' stata divisa in 2 sottosistemi:

2.1) Operation,

a sua volta diviso in:

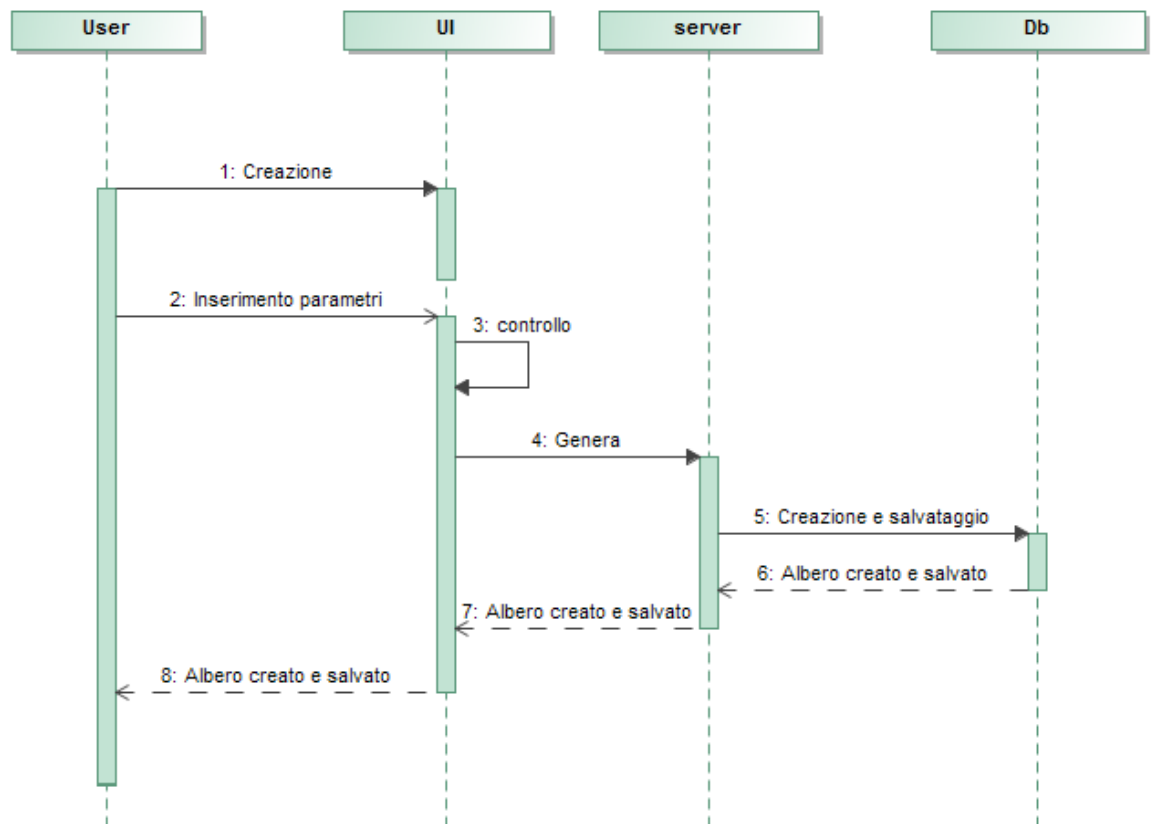
- Gestore Parametri: sottosistema che si occuperà di smistare le richieste;
- Operazioni: sottosistema che si occuperà di generare, salvare, cancellare, recuperare lista e informazioni sugli Alberi ;

2.2) CommunicationDatabase: è il sottosistema che interroga il DB. Esso deve gestire eventuali race-condition nella creazione e cancellazione di un albero.

3) L'ultima componente DB sarà composta da un sottosistema dbms che si occuperà di gestire e recuperare i dati e una parte Database che sarà quella dove saranno realmente memorizzati.

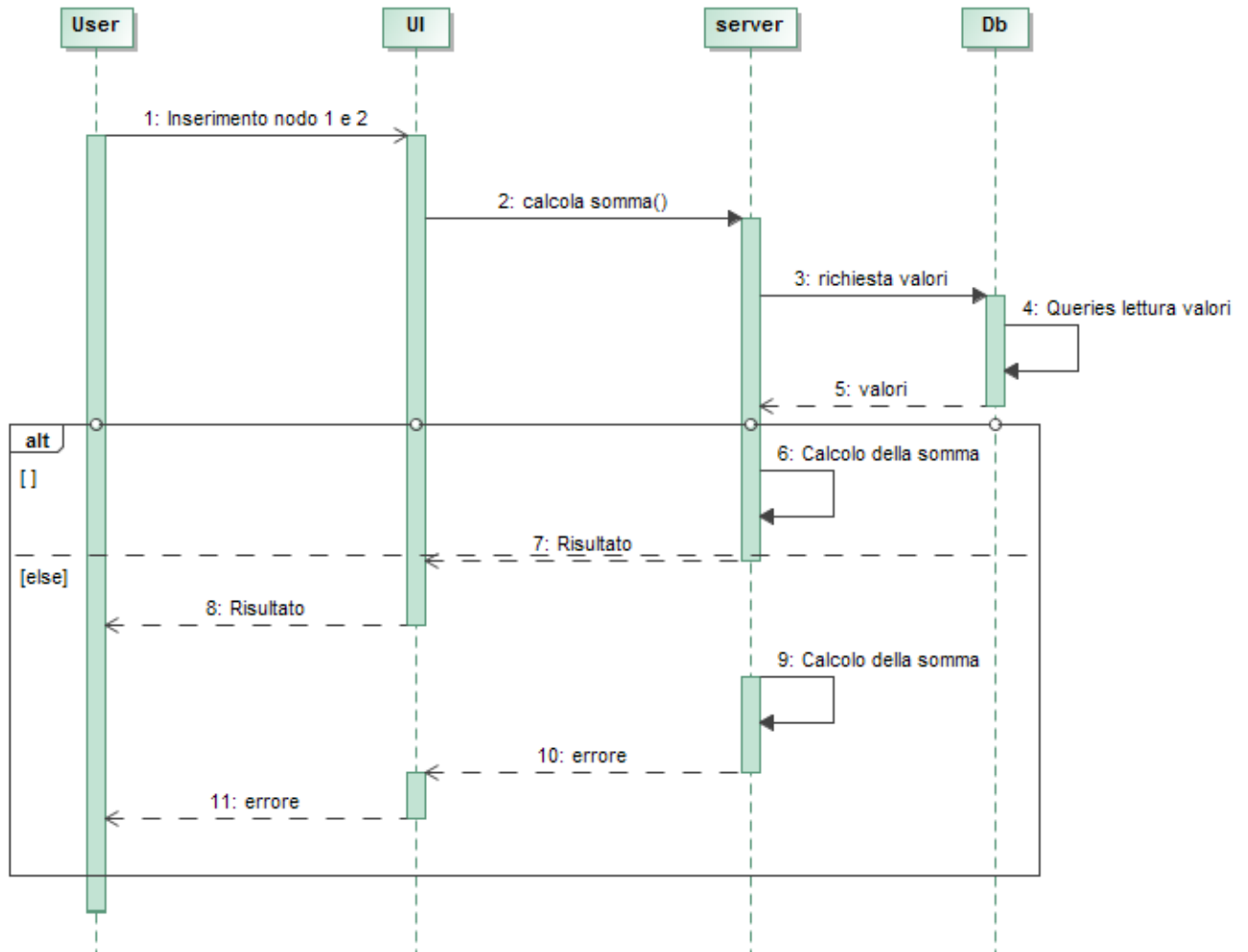
*C.2 The dynamic view of the software architecture: Sequence Diagram*

-Creazione albero



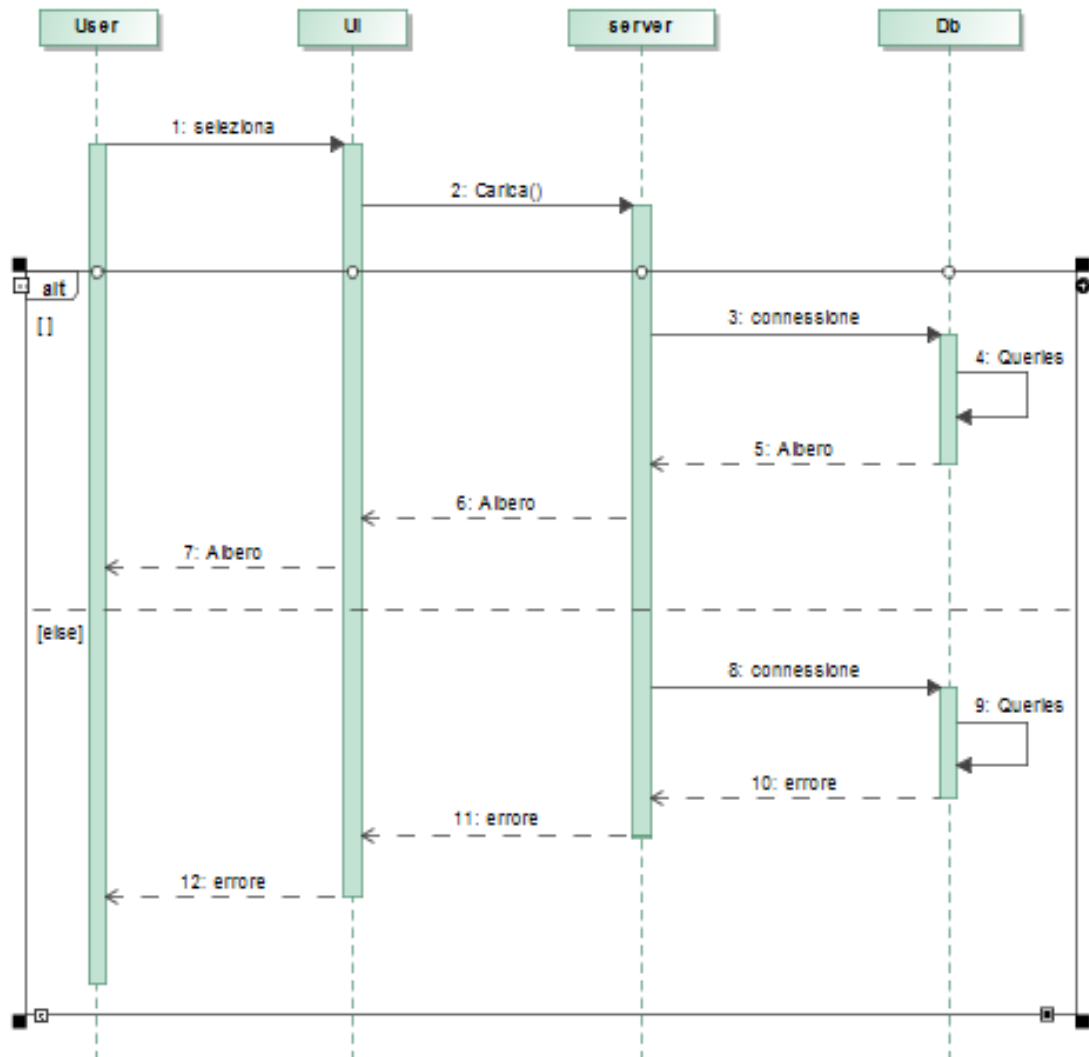
1. La creazione di un nuovo albero, generato a partire da una serie di dati inseriti dall'utente, mediante l'utilizzo della GUI e del suo codice interno.
2. Immissione dei parametri
3. Controllo correttezza
4. UI comunica al Server che è andato tutto a buon fine, quindi può generare l'albero
5. Server comunica al Database che è andato tutto a buon fine, quindi può creare e salvare l'albero
- 6,7,8. Creazione andata a buon fine quindi si può andare alla pagina di calcolo somma

-Calcolo su albero



1. L'utente inserisce i parametri (Nodo1, Nodo2)
2. Pulsante di Calcolo della somma
- 3,4. Richiesta dei valori dei due nodi
5. Il server riceve le informazioni richieste
- 6,7 (if) Se i valori sono corretti il server effettua il calcolo della somma sui due nodi restituendo il risultato a schermo
8. UI rende disponibili il risultato dell'utente
- 9,10 (else) altrimenti restituisce un segnale di errore
- 11 UI rende disponibile il segnale di errore quindi dovrà ripetere la procedura di calcolo

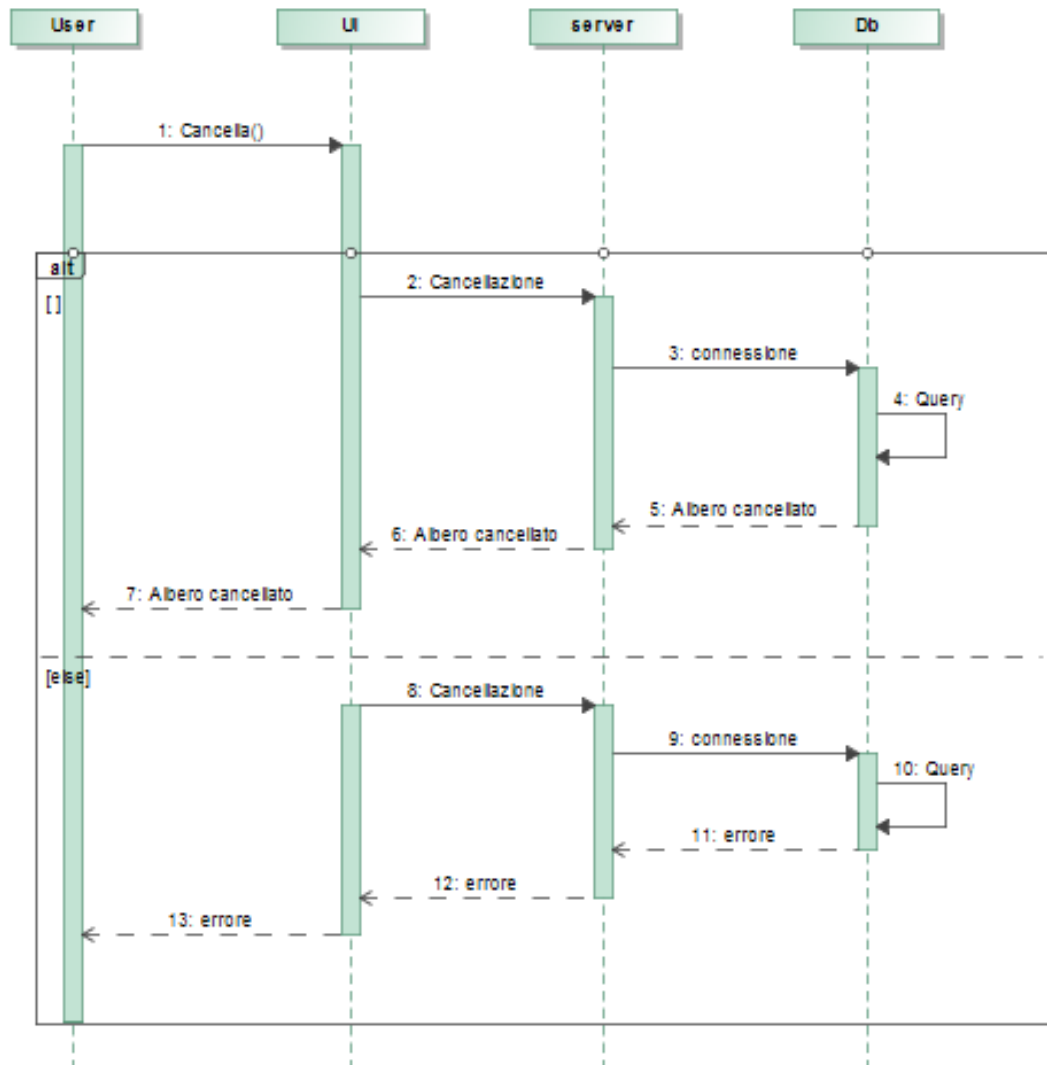
-Caricamento Albero





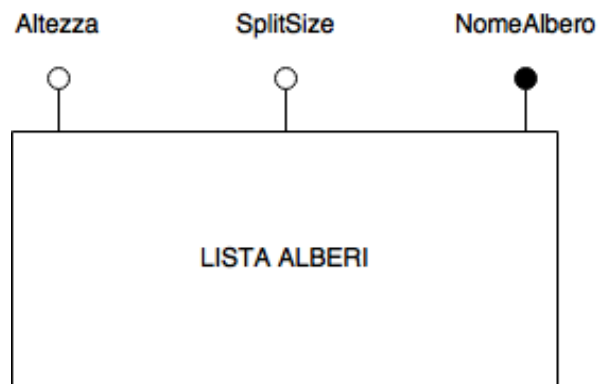
1. L'utente seleziona un albero attraverso la UI
2. La UI avvia la procedura di caricamento di questo albero scelto nel database
3. (if) Il server si connette al database
4. Effettuando una query sul database è possibile riprendere le informazioni dell'albero
5. L'interfaccia con il database notifica il server dell'esito dell'operazione
6. Il server comunica l'esito alla UI
7. UI mostra l'esito dell'operazione all'utente
8. (else) il server non riesce a connettersi al database
- 9, 10, 11. Segnali di errori quindi dovrà ripetere la procedura di caricamento.

-Cancellazione



1. Si ha intenzione di cancellare un albero già presente
2. (if) UI comunica al Server l'intenzione di cancellare un albero
3. Il server si connette e quindi comunica con il database.....
4. ....che si preoccupa di eseguire una query relativa all'operazione di cancellazione
5. Database notifica al Server di aver cancellato l'albero
- 6,7. UI notifica l'utente la cancellazione dell'albero riuscita
- 8,9. (else) il server non si connette con il database
10. Query che non va a buon fine
- 11,12,13. Segnali di errori , quindi l'utente dovrà ripetere la procedura di cancellazione

## D. ER Design

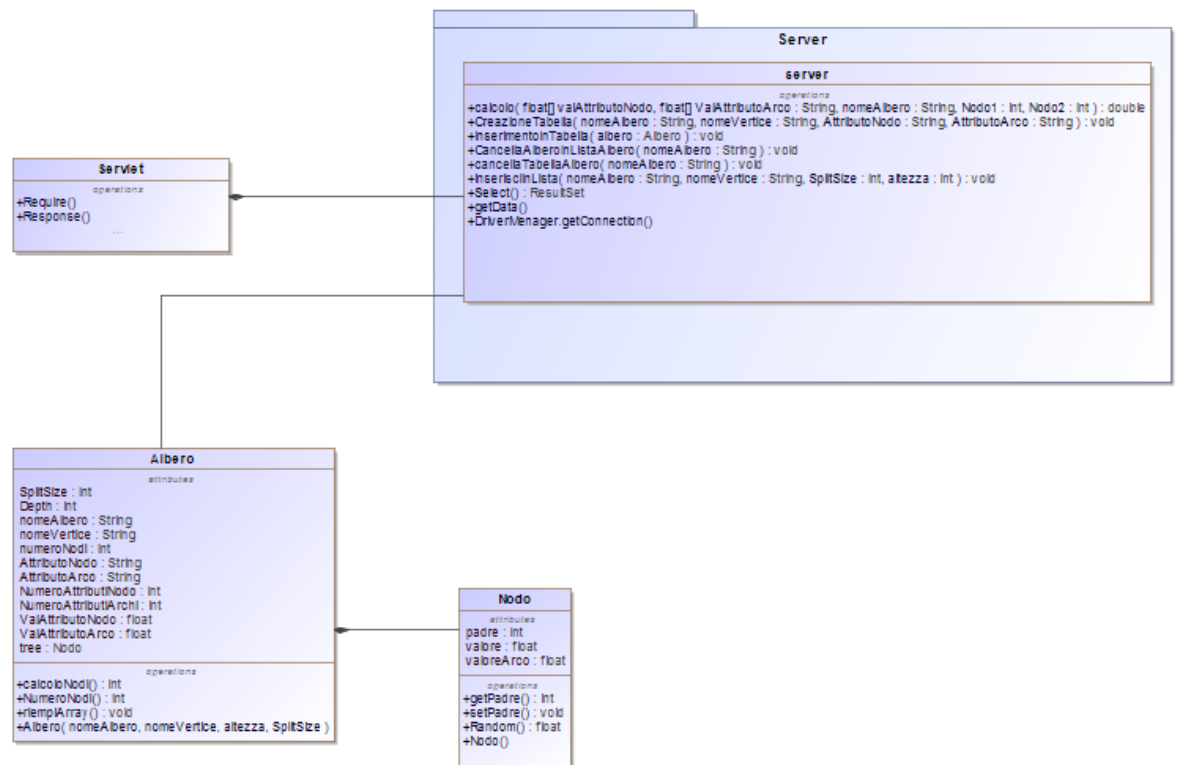


Il Database ha una sola tabella chiamata ListaAlberi.

LISTALBERI ( NomeAlbero VARCHAR(50) Primary key,  
SplitSize INT,  
altezza INT )

Non vi sono relazioni tra tabelle. Ogni Volta che viene creato un nuovo albero viene create una nuova tabella

NomeAlbero (NumeroNodo INT ,  
Padre INT,  
AttributoNodo FLOAT [...],  
AttributoArco FLOAT [...]);



### **-Descrizione class diagram**

#### **Class ALBERO:**

Il team ha deciso di inserire questa classe nel diagramma poichè ritenuta fondamentale al funzionamento del server, in quanto quest'ultimo lavorerà proprio su essa.

La classe albero è composta dall'oggetto albero il quale a sua volta è caratterizzato da attributi (dove AttributoNodo, AttributoArco, ValAttributoNodo e ValAttributoArco sono array) quali specificano la struttura dell'albero in considerazione.

#### **Class Server:**

La classe Server è la parte più importante del nostro sistema, poichè contiene il codice che ci permetterà di effettuare il calcolo effettivo tra due vertici. La classe è caratterizzata da operazioni che indicano ciò che occorre per il corretto funzionamento del sistema.

#### **Class Servlet:**

La classe Servlet è una componente essenziale del sistema, in quanto grazie alle sue funzioni request() e response() ci permette di scambiare dati con la User Interface. La funzione request ci permette di prendere dati input, mentre response ci permette di restituire il risultato.

## F. Design Decisions

---

Lo sviluppo della web application è stato pensato in PHP per la facilità nel reperire informazioni e la facilità di comunicazione con pagine html

Essendo Php un linguaggio interpretato e dovendo gestire una grande quantità di dati non siamo riusciti a creare una struttura dati che ci permettesse di creare un albero delle dimensioni massime. Per aggirare il problema abbiamo provato a memorizzare solo una stringa contenente i valori ottenuti randomicamente e successivamente inserirla nel DB.

I tempi di creazione e salvataggio si attestavano intorno al minuto, tempo ritenuto comunque eccessivo. Da qui la scelta di cambiare linguaggio e utilizzare JAVA.

Java non crea problemi nel creare una struttura dati capace di memorizzare un albero di 2milioni di nodi.

Con java i tempi di inserimento sono ancora maggiori ma avendo l'albero in memoria centrale possiamo far effettuare i calcoli immediatamente.

La struttura dati scelta è l'array per i seguenti motivi

- accesso in  $O(1)$  a qualsiasi elemento
- minore spreco di memoria

Ogni cella dell array è formata da un istanza della classe nodo,

Formata da un riferimento al nodo padre e un array per la memorizzazione dei valori generati randomicamente

Per occupare la minor memoria possibile il team ha scelto di non utilizzare la classe arco. Considerando che un nodo è collegato solo ad un altro nodo gli attributi del arco verranno aggiunti al nodo corrispondente

L interfaccia grafica comunicherà con il server attraverso le servlet.

Il database scelto è MYSQL. La motivazioni che ci ha portato a questa tecnologia è la pregressa conoscenza di tale db.

Ogni albero crea una nuova tabella. Questa scelta è stata fatta per evitare di avere tabelle di dimensioni troppo grandi per effettuare calcoli in tempo breve

-



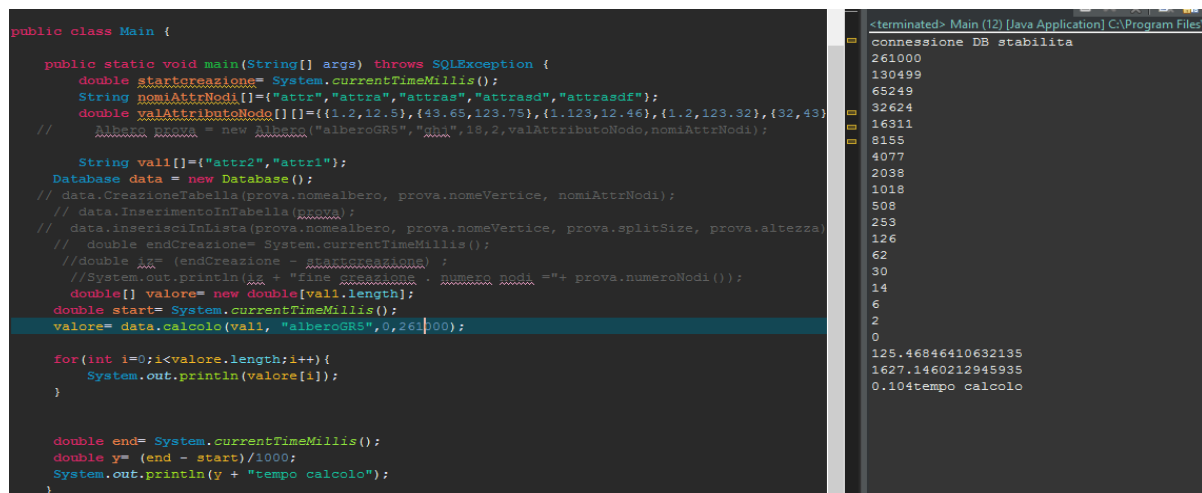
## G. Explain how the FRs and the NFRs are satisfied by design

La UI sarà completamente sviluppata in html5.

Il prototipo presentato riesce a creare alberi di piccole dimensioni(20000/50000 nodi) quasi istantaneamente. Per alberi della dimensione massima con 7/10 attributi il tempo sale intorno ai 10/20 minuti.

Per l'inserimento di grandi quantità di dati il manuale mysql descrive 2 tecniche : utilizzare uno storage engine diverso o eseguire insert multiple. Il team non ha deciso di copiare l'albero in un file esterno e successivamente inserire il tutto nel database ma ha scelto di utilizzare insert multiple .quindi abbiamo scritto le query adHoc per garantire l'inserimento dell'albero.

Per il calcolo invece come avevamo ipotizzato nella 1° deliverable i tempi sono irrisori. Il calcolo non supera il secondo per alberi di dimensiona massima.(Si consideri che il percorso più lungo è di 21 nodi con splitsize=2) che nel implementazione della funzione calcolo() corrispondono a 21 select;



```
public class Main {  
    public static void main(String[] args) throws SQLException {  
        double startCreazione= System.currentTimeMillis();  
        String nomiAttrNodi[]={"attr", "attra", "attras", "attrasd", "attrasdf"};  
        double valAttributoNodo[][]={{1.2,12.5},{43.65,123.75},{1.123,12.46},{1.2,123.32},{32,43}}  
        // Albero prova = new Albero("alberoGR5", "gna", 18, 2, valAttributoNodo, nomiAttrNodi);  
  
        String val1[]={"attr2", "attr1"};  
        Database data = new Database();  
        // data.CreazioneTabella(prova.nomealbero, prova.nomeVertice, nomiAttrNodi);  
        // data.InserimentoInTabella(pppwa);  
        // data.inserisciInLista(prova.nomealbero, prova.nomeVertice, prova.splitSize, prova.altezza);  
        // double endCreazione= System.currentTimeMillis();  
        //double iz= (endCreazione - startCreazione) ;  
        //System.out.println(iz + "fine creazione - numero nodi =" + prova.numeroNodi());  
        double[] valore= new double[val1.length];  
        double start= System.currentTimeMillis();  
        valore= data.calcolo(val1, "alberoGR5", 0, 2, 100);  
  
        for(int i=0;i<valore.length;i++){  
            System.out.println(valore[i]);  
        }  
  
        double end= System.currentTimeMillis();  
        double y= (end - start)/1000;  
        System.out.println(y + "tempo calcolo");  
    }  
}
```

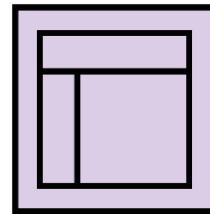
```
<terminated> Main (12) [Java Application] C:\Program Files\...  
= connessione DB stabilita  
261000  
130499  
65249  
32624  
16311  
8155  
4077  
2038  
1018  
508  
253  
126  
62  
30  
14  
6  
2  
0  
125.46846410632135  
1627.1460212945935  
0.104tempo calcolo
```

I tempi di calcolo sono sulla sinistra

il requisito della multiutenza è garantito da l'utilizzo di webserver esterni e le servlet. Questo ci evita di dover utilizzare direttamente le socket.

Il requisito delle 100 è soddisfatto se gli utenti creano alberi di piccole/medie dimensioni. Non riusciamo ancora a gestire la multiutenza con alberi grandi.

# H. Effort Recording



## GANTT

Indifferente					
Corsetti Stefano, Di Salle Tommaso, D'Orazio Luca, Mancini Eugenio, Di Cato Francesco					
5					
s.corsetti@hotmail.it , tommasodisalle@gmail.com , lucaadorazio@gmail.com , Emancini1992@libero.it , dicatofrancesco@gmail.com					
When (Month/Day)	Time spent	Partners (please report how many people have been working)	Brief Description of the performed task	Category	Sub-Category
12   10	3	2	Collegamento da server a database	doing	Code-implementation
12   19	5	3	Collegamento effettivo con gui	doing	code-implementation
12   20	1	2	Class diagram	doing	Diagram
12   15	5	1	esecuzione calcolo	doing	code-implementation
12   20	1	2	Sequence Diagram	doing	diagram
12   9	2	2	Revisione diagrammi primo step	doing	diagram
12   23	3	5	Documentazione e revisione	doing	documentation
12   8	2	3	Decisioni su implemetazione	doing	code-implementation
12   14	5	3	creazione albero	doing	code-implementation
12   12	15	3	implementazione prototipo	doing	code-implementation
TOTALE ORE SPESE :		115			