"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

Date	27/11/2016
Deliverable	*Deliverable 1*
Team (Name)	B-Group

Team Members		
Name & Surname	**Matriculation Number**	**E-mail address**
Francesco Maria Cameli	*238857*	*francescomariacameli@gmail.com*
Tony D'Angelo	*236027*	*tunyx7@gmail.com*
Valerio Crescia	*236107*	*valerio.crescia@live.it*
Kevin Titi	*229587*	*titikevin93@gmail.com*
Cristiano Orsetti	*236425*	*cristiano.orsetti@hotmail,it*

Project Guidelines

[do not remove this page]

This page provides the Guidelines to be followed when preparing the report for the Software Engineering course. You have to submit the following information:

- *This Report*
  - *Diagrams (Analysis Model, Component Diagrams, Sequence Diagrams, Entity Relationships Diagrams)*
- *Effort Recording (Excel file)*

Important:

- *document risky/difficult/complex/highly discussed requirements*
- *document decisions taken by the team*
- *iterate: do not spend more than 1-2 full days for each iteration*
- *prioritize requirements, scenarios, users, etc. etc.*

Project Rules and Evaluation Criteria

General information:

- *This homework will cover the 80% of your final grade (20% will come from the oral examination).*
- *The complete and final version of this document shall be not longer than 40 pages (excluding this page and the Appendix).*
- *Groups composed of seven students (preferably).*

I expect the groups to submit their work through GitHub

Use the same file to document the various deliverable.
Document in this file how Deliverable "i+1" improves over Deliverable "i".

Project evaluation:

Evaluation is not based on "quantity" but on "quality" where quality means:

- *Completeness of delivered Diagrams*
- *(Semantic and syntactic) Correctness of the delivered Diagrams*
- *Quality of the design decisions taken*
- *Quality of the produced code*

Table of Contents of this deliverable

List of Challenging/Risky Requirements or Tasks

<In this section, you should describe using the table below the most challenging or discussed or risky design tasks, requirements, or activities related to this project. Please describe when the risk arised, when and how it has been solved.>

PLEASE FILL IN THIS TABLE AT EACH DELIVERABLE

Challenging Task	Date the task is identified	Date the challenge is resolved	Explanation on how the challenge has been managed
Scelta delle tecnologie	Il giorno della consegna	11-08-2016	Abbiamo scelto di usare Java come linguaggio di programmazione (motivi alla fine) e MongoDB come DBMS.
Organizzazione del team geograficamente distante	Il giorno della consegna	Circa una settimana dopo	Darsi appuntamento in ateneo ed di tanto in tanto in casa di un membro del team per fare il punto della situazione, organizzarsi con Dropbox ed skype per la condivisione del progetto e la discussione

A. Requirements Collection

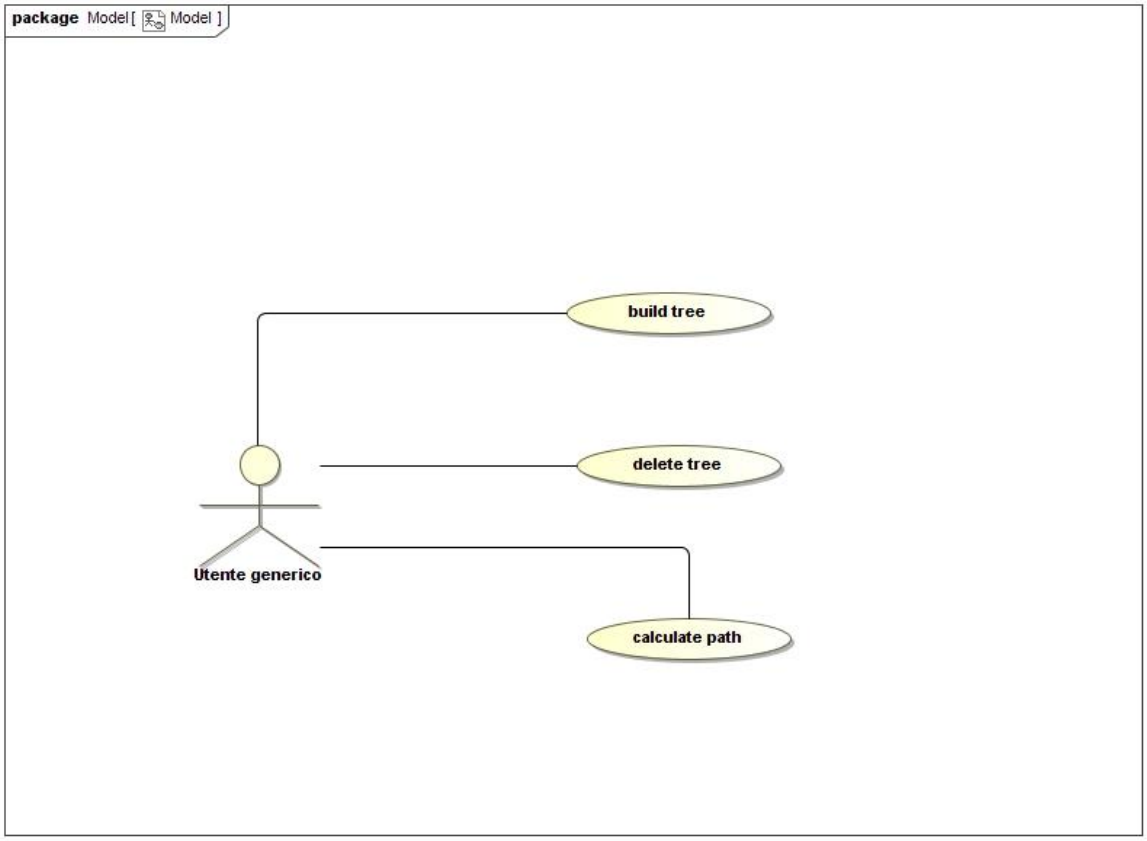
In this section, you should describe both the application ***features/functional*** *requirements as well as the* ***non functional*** *ones. You shall also document* ***constraints*** *and* **rules**, *if they apply.*

A.1 Functional Requirements

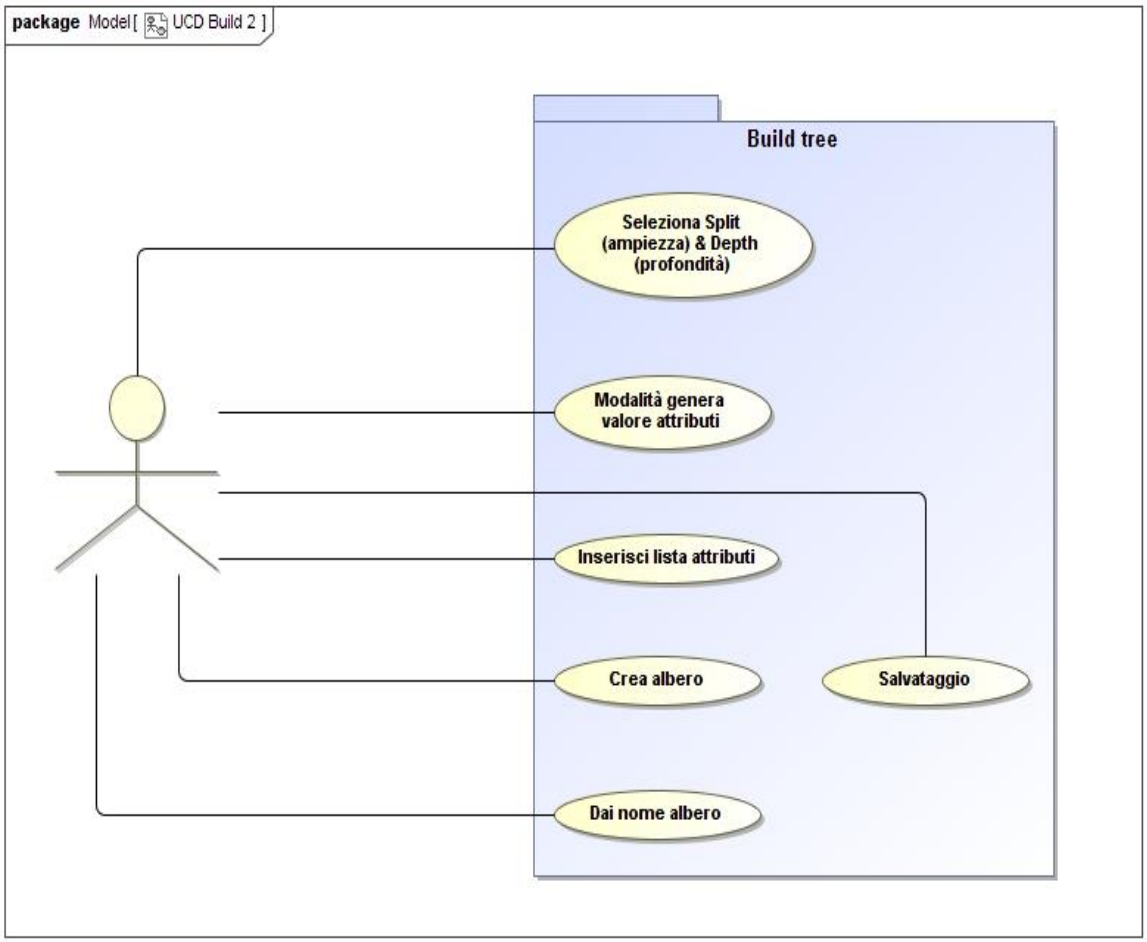
<List and describe functional requirements through Use Case Diagrams. Then, prioritize them, and provide a table-based description of the most important requirements>
PLEASE REPORT: (i) ALWAYS the diagram to be discussed, (ii) the text explaining the DECISIONS taken when creating the diagram (that is, do not spend time in EXPLAINING what the diagram says, but more on the decisions taken to create the diagram).

PRIORITIZE THEM

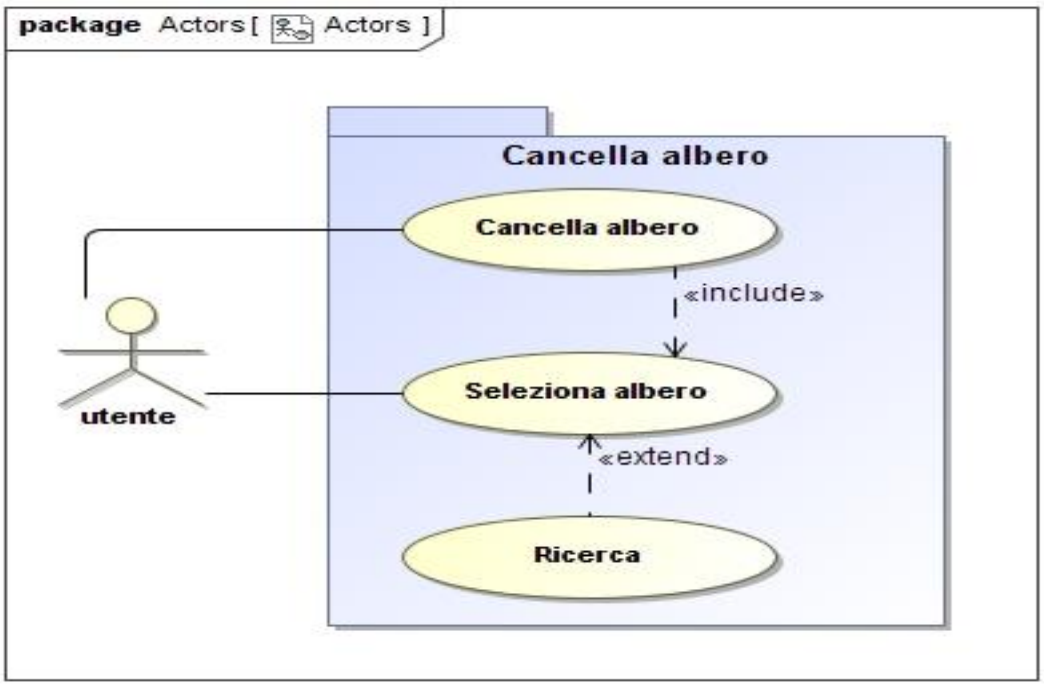
FOR EACH DIAGRAM (Use Case D, Analysis Model, Class, Sequence) add a number and a label to it (e.g., Figure 1: Sequence diagram of the xxx scenario)



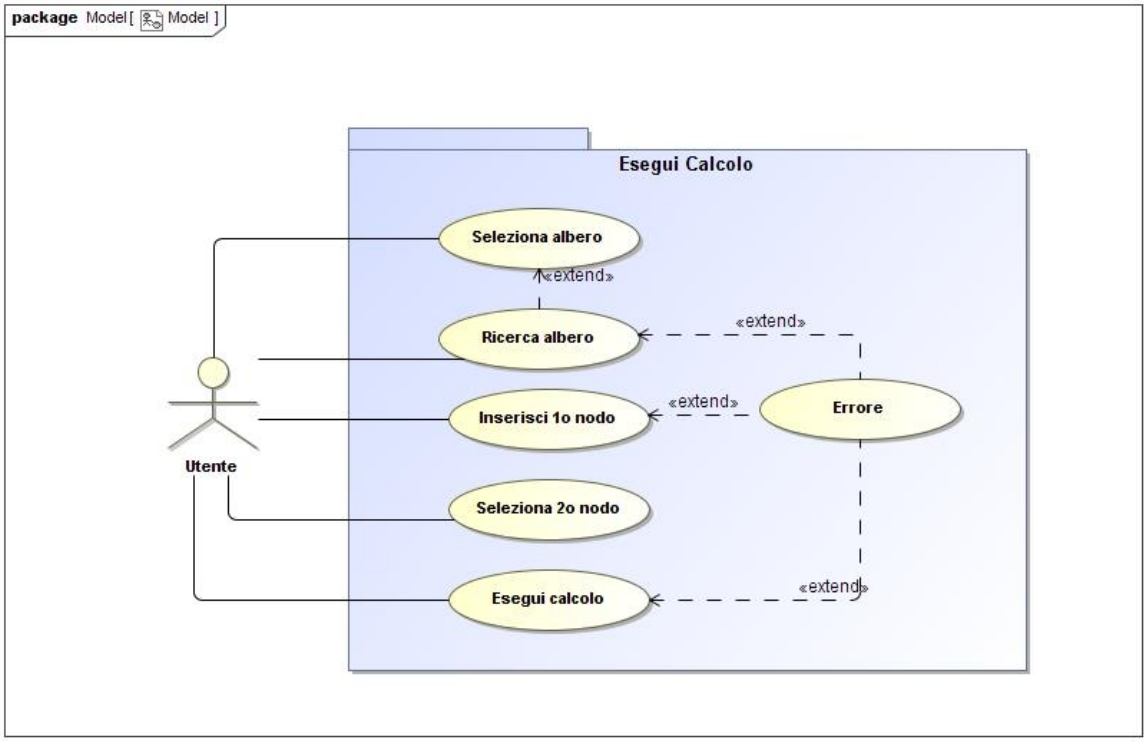
Basandoci sulle specifiche del progetto e sui chiarimenti che ci sono stati dati ai dubbi, abbiamo deciso che le operazioni principali da fare con questo sistema saranno:
costruzione dell'albero
cancellazione dell'albero
esecuzione dell'operazione
È stato deciso di fare un sistema completamente distribuito, quindi non ci saranno differenze tra salvare su un file system o salvare in remoto.



L'operazione di Build Tree è stata esplosa in 5 operazioni:
Selezionare split e size; inserimento degli attributi che si vogliono dare a ogni nodo e ogni arco dell'albero; decisione della modalità con cui questi attributi vengono generati; l'azione stessa della creazione dell'albero una volta che sono state decise le altre cose; l'assegnazione di un nome all'albero; il salvataggio dell'albero nel database.

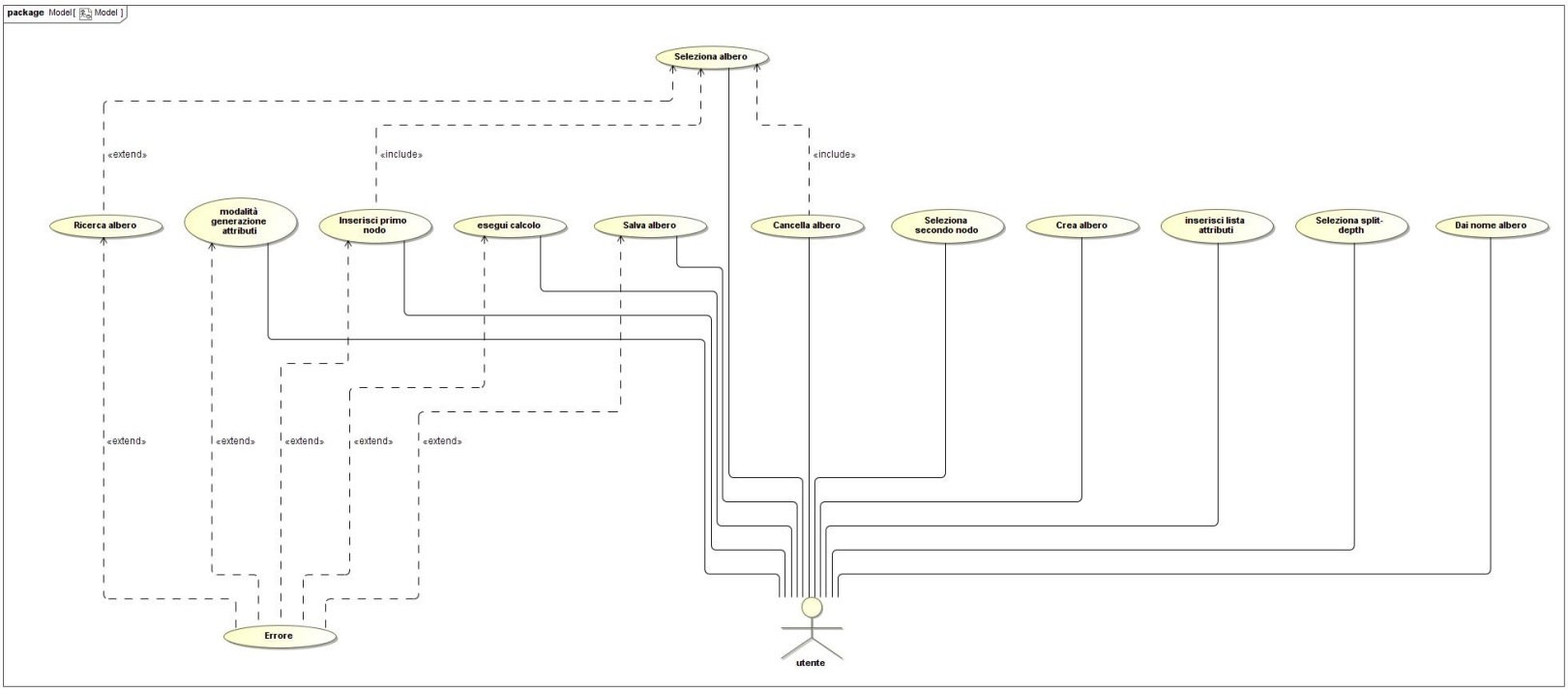


L'operazione di cancellazione dell'albero è piuttosto semplice: consiste nel selezionare l'albero e cancellarlo. Ovviamente la selezione è obbligatoria per la cancellazione, e la selezione può essere estesa tramite la ricerca dell'albero.



Per l'esecuzione del calcolo abbiamo pensato, forse pure anzitempo, a un dettaglio implementativo: il numero di nodi dell'albero cresce in maniera esponenziale *k^n,* dove *k* è lo split, *n* è la size; il numero di percorsi una volta scelto un nodo cresce in maniera

lineare con *n.* Per esempio se ci sono 2'000'000 di nodi e si sceglie un nodo all'ultimo livello come primo nodo, essendo 2'000'000 = circa a 2^21, ci saranno 21 nodi che costruiranno un path, 1'999'979 che daranno errore. Quindi abbiamo considerato più conveniente dare la possibilità all'utente di scegliere il primo nodo e poi far si che il sistema offra una rosa dei nodi tra cui scegliere. Quindi si selezionerà l'albero dopo un'eventuale ricerca, si inserirà il primo nodo, si selezionerà dalla nostra rosa il secondo e infine si eseguirà il calcolo.



Una volta unite le esplosioni studiate una a una, lo Use Case Diagram esce così. Gli errori possibili sono dovuti alla ricerca di un albero che non esiste, al salvataggio di un albero con il nome già usato, all'esecuzione del calcolo che non si sa mai, all'inserimento del primo nodo assente nell'albero selezionato e alla modalità di generazione attributi non valida.

USE CASE #	Genera Albero	
Preconditions	null	
Success End Condition	Porta alla creazione e al salvataggio di un albero	
Failed End Condition	Si genera un alert e porta alla rimmissione dei dati	
Primary, Secondary Actors	Utente null	
Trigger	Immissione dati	
DESCRIPTION	Step	Action
	0	Inserisci split & size

	1	Inserisci lista attributi
	2	Modalita generazione valori attributi
	3	Creazione albero
	4	Dai un nome all'albero
	5	Salva albero

RELATED INFORMATION	Generazione Albero
Frequency	100 a settimana

USE CASE #	Cancellazione albero

Preconditions	Ci sia almeno un albero nel sistema	
Success End Condition	Viene cancellato l'albero, il suo nome, il suo ID dal database	
Failed End Condition	L'albero selezionato viene mutilato o non cancellato affatto; viene selezionato un ID associato a nessun albero	
Actor	Utente	
DESCRIPTION	Step	L'attore seleziona l'albero da eliminare tramite il suo ID e questo viene eliminato dal database

USE CASE #	Esecuzione calcolo
Goal in Context	Sommare I valori degli attributi del path che porta da un nodo a un altro.
Preconditions	Presenza di almeno un albero.
Success End Condition	vengono restituiti path, somma di attributo per attributo, tempo per il calcolo

Failed End Condition	Non viene restituito niente	
Actor	Utente	
DESCRIPTION	Step	L'utente seleziona un albero, seleziona due nodi di quest'albero e il sistema esegue I calcoli facendo la somma degli attributi e restituendo il path che porta dal nodo di partenza a quello di destinazione, il tempo impiegato per questo calcolo.

A1.1 GUI Requirements (da riempire a partire dalla Versione 2)

A1.2 Business Logic Requirements (da riempire a partire dalla Versione 2)

A1.3 DB Requirements (da riempire a partire dalla Versione 2)

A.2 Non Functional Requirements

<List and describe here the most important non functional requirements.>

BE CAREFUL NOT TO MAKE CONFUSION AMONG DIFFERENT NON FUNCTIONAL REQUIREMENTS

A.3 Content

*<Describe the **data provenance** (use of external API, web service, DB ...)>*

A.4 Assumptions

<Briefly document, in this section, the most relevant requirement assumptions/decisions you had to made during your project>

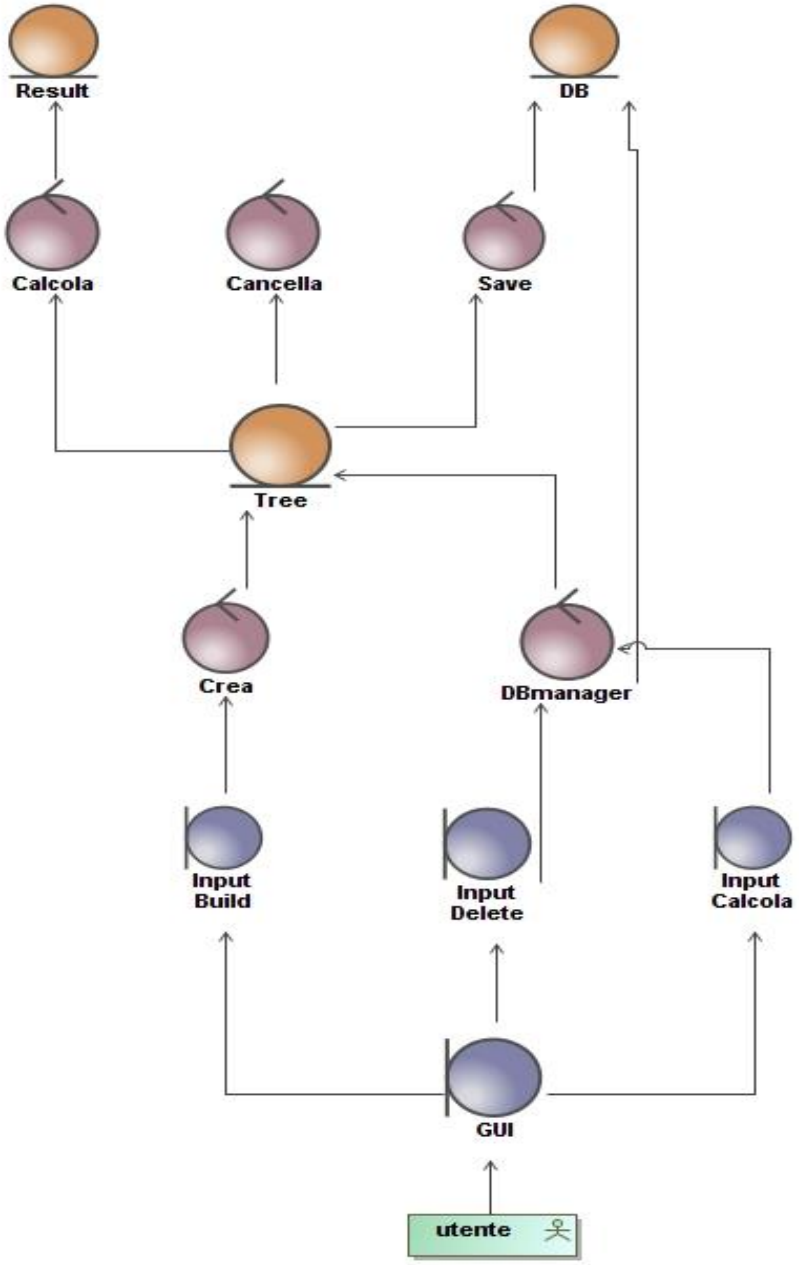
A.5 Prioritization

Requisiti listati secondo ordine di priorità.

B. Analysis Model

<In this section, you shall report the Analysis Model produced for your system>

PLEASE REPORT: (i) ALWAYS the diagram to be discussed, (ii) the text explaining the decisions taken to create the diagram (that is, why did you create a certain analysis model design?)



L'utente trova un'interfaccia che gli permette di decidere se l'operazione che sta andando a fare sarà di costruzione, cancellazione o calcolo. Nel primo caso crea l'albero e lo salva nel database; nel secondo caso usa il DB manager per selezionare l'albero dal database (quindi il controller DB manager funge da "seleziona") e poi lo cancella; nell'ultimo caso si usa il DB manager per selezionare l'albero dal database, il controller "calcola" esegue l'operazione e trova il risultato.

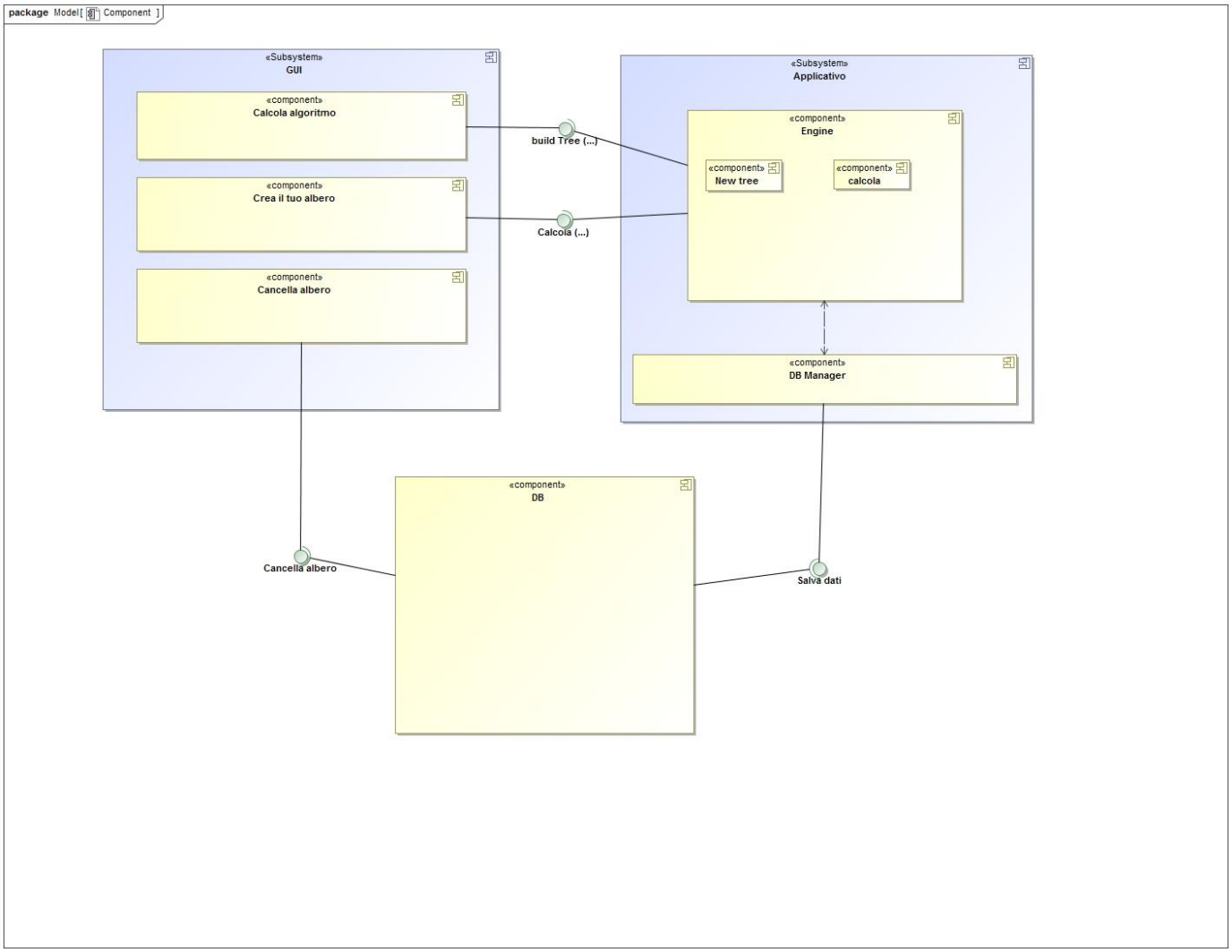
C. Software Architecture

<Report here both the static and the dynamic view of your system design, in terms of a Component Diagram, Class Diagrams and their related Sequence Diagrams >

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

AVOID TO MAKE IT TOO COMPLEX AND FINE GRAINED. FOCUS MORE ON "HOW" THE COMPONENTS SHALL INTERACT IN ORDER TO SATISFY THE REQUIREMENTS

ADD INTERFACES and their parameters



Il component diagram si compone di tre sottosistemi:

- La GUI (interfaccia grafica) che si occupa di ottenere in input tutti i dati necessari all'engine (per la creazione dell'albero ed il calcolo)
- L'applicativo che esegue i calcoli viene diviso in engine e DB manager per gestire il database
- Infine il database che immagazzina tutti i dati necessari

L'operazione di cancellazione albero si interfaccia direttamente con il db poiché basta una semplice query per la cancellazione; le altre due operazioni data la loro complessità necessitano di supporto da parte dell'applicativo che eseguirà gran parte dei calcoli per poi memorizzare il tutto nel DB attraverso un manager che ci strutturerà i dati

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

BE SURE THAT THE STRUCTURE IS SYNCHRONIZED WITH THE DYNAMIC VIEW

D. ER Design

<Report here the Entity Relationship Diagram of the system DB>

E. *Class Diagram of the implemented System*

F. Design Decisions

<Document here the **5** most important design decisions you had to take. You can use both a textual or a diagrammatic specification.>

THIS IS A VERY IMPORTANT PART. BE SURE TO DOCUMENT THE 5 MOST IMPORTANT DECISIONS (related to your requirements and design) YOU MADE

Linguaggio di sviluppo

Abbiamo scelto di usare Java come linguaggio di sviluppo perché tutto il gruppo ne ha un'infarinatura generale; abbiamo previsto che la lentezza di tale linguaggio sarà compensata dal tempo guadagnato per il testing a discapito del learning.

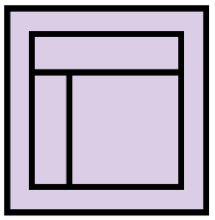
DBMS

Abbiamo discusso parecchio se usare un database relazionale o meno; appena saputo che gli alberi, una volta creati, non saranno modificati, uno svantaggio dei database non relazionali, ovvero la mancanza d'integrità, ha perso il suo peso nella lista dei contro dei database non relazionali. È stato in particolare scelto MongoDB dopo aver confrontato le prestazioni dei vari NoSQL e aver interpretato le sue caratteristiche come quelle che più si confaranno al nostro sistema.

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

<Report in this section how the design you produced satisfies the FRs and the NFRs>

H. Effort Recording



GANTT
Make a GANTT documenting the tasks and timing you expect to spend on the deliverable. Try to be as precise as possible. Check, after the deliverable deadline, if and how you satisfied (or not) the deadlines.

Logging
As you are working on the assignment, record what you are doing and how long you spent. As a rule of thumb, you should add a log entry every time you switch tasks. For example, if you do something for two hours straight, that can be one log entry. However, if you do two or three things in half an hour, you must have a log entry for each of them. You do not need to include time for logging, but should include the time spent answering the other parts of this question.

For this purpose, please use the ***LogTemplate.xls*** *file.*

Categorization
*When logging the time spent on the project, please create different sub- categories. Specifically, it is important to clearly distinguish between two main categories: the time spent for "****learning****" (the modeling languages, the tools, etc.) from the time needed for "****doing****" (creating the models, taking the decisions, ...). Learning tasks are in fact costs to be paid only once, while doing costs are those that will be repeated through the project.*
For each category, please define sub-categories. Examples follow. You may add other sub-categories you find useful.

Learning
- ***Requirements Engineering***
- ***Non functional Requirements***
- ***Use Case Diagrams***
- ***Tool study***

Doing:
- ***Requirements discovery***
- ***Requirements Modeling (UC diagrams)***

Summary Statistics
Based on the attributes defined above, calculate the summary statistics of the time spent for "learning", the time spent for "doing", and the total time.

13h Doing; 10h Learning;

Note: this Deliverable report shall document only the Summary Statistics for the different deliverables (D1, D2, and Final). Detailed information shall be reported in the Excel file.

COPY HERE (computed from the spreadsheet): i) the total number of hours spent by the group (that is, hours per task X number of people working on that task), ii) the time spent for LEARNING and for DOING

Appendix. Code

<Report in this section a **documented** version of the produced code>