"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	10/01/2017
Deliverable	*Deliverable Fin*
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

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 five 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

INDEX

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	12-18-2016	Abbiamo scelto MySQL come DBMS, PHP come linguaggio di sviluppo. Lunga storia raccontata in design decision.
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 Hangout per la condivisione del progetto e la discussione
Divisione dei ruoli nell'implementazione del progetto	27 Novembre 2016	2 Dicembre 2016	Inizialmente nessuno nel team si occuperà di engine o dbms o gui in particolare, ma si penserà insieme all'infrastruttura per far comunicare le varie componenti del sistema, rimandando la separazione vera e propria dei compiti a quando ognuno saprà quale componente comunicherà in che modo con un altro.
Gestione esecuzioni concorrenti	20 Dicembre 2016	15 Gennaio 2017	Vedere Punto G: come soddisfiamo i requisiti funzionali o non funzionali

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.*

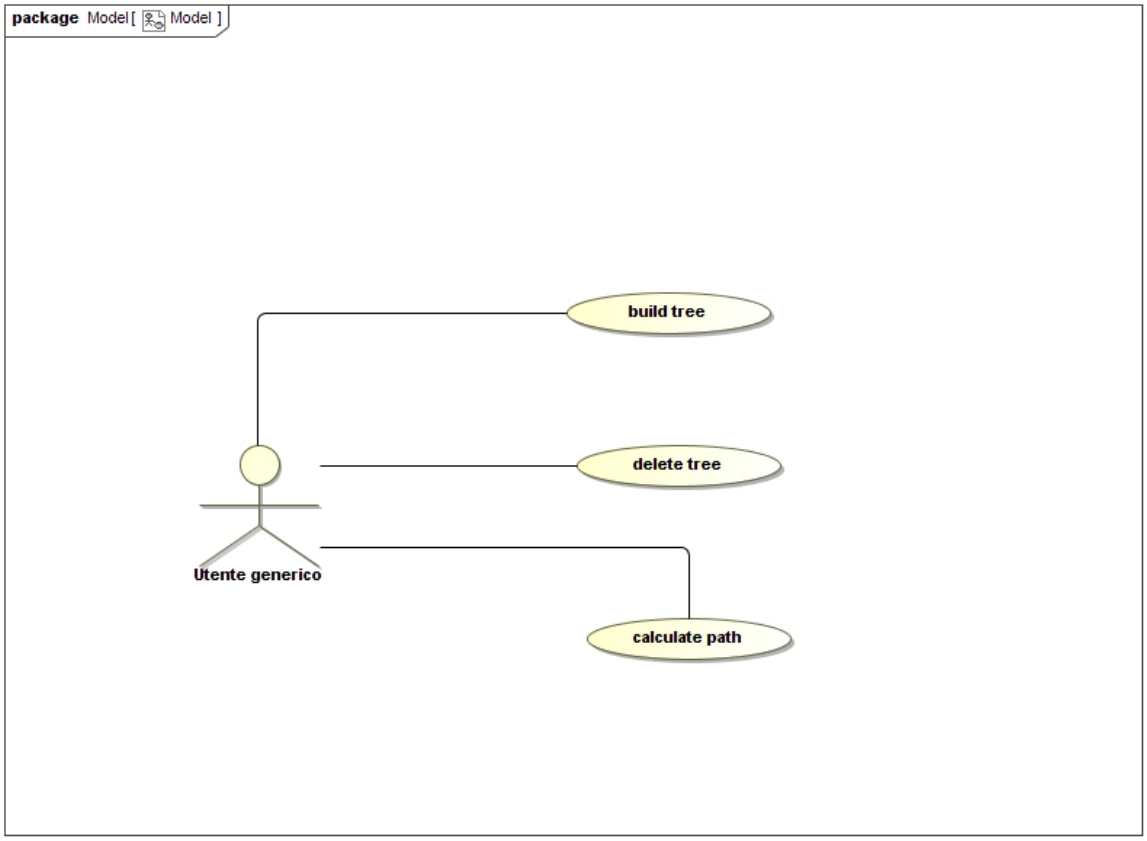
A.1 Functional Requirements

<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).

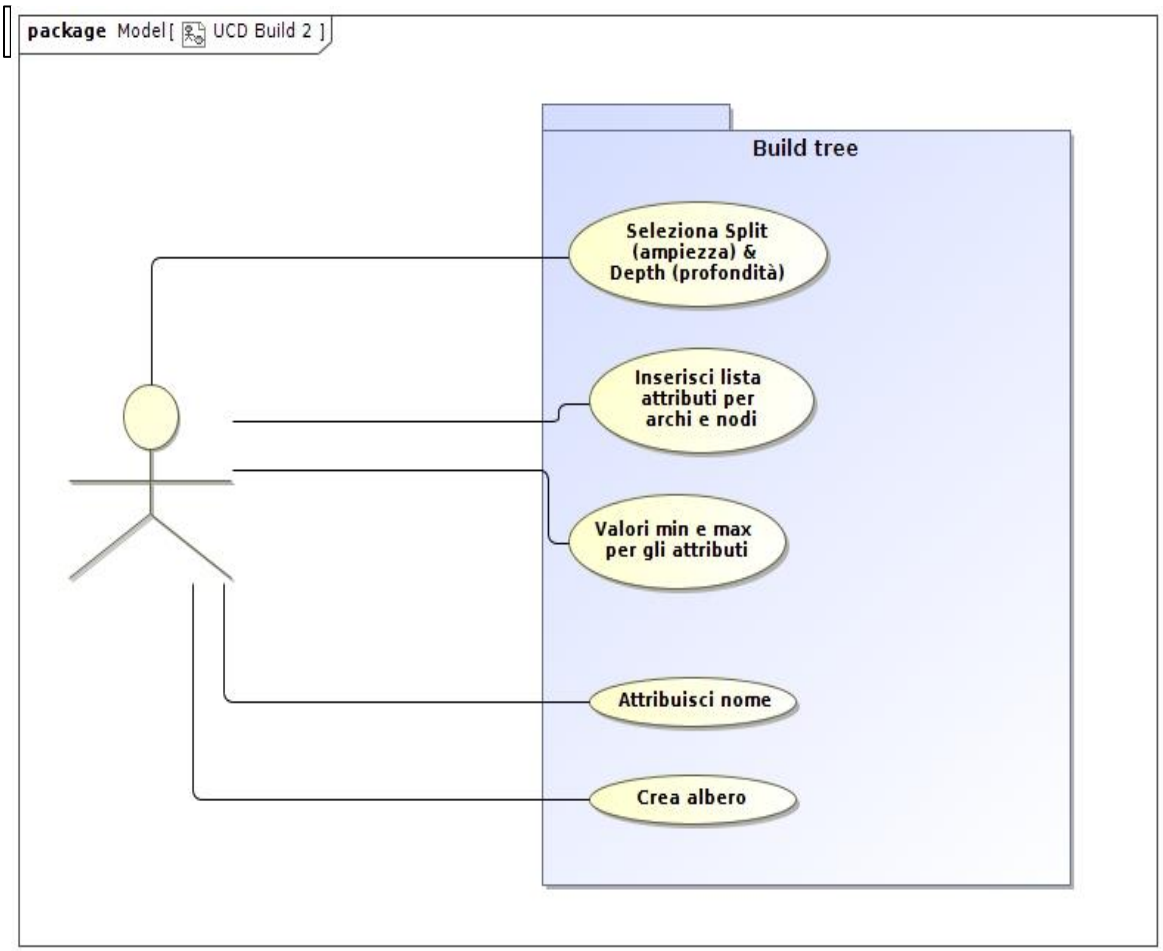
PRIORITIZE THEM FOR EACH DIAGRAM (Use Case D, Analysis Model, Class, Sequence) add a number and a label to it (e.g., Figure 1: Sequence diagram of the xxx scenario)

Prioritizzeremo le varie funzionalità in “alta”, “media” e “bassa”.
Il sistema dovrà essere in grado di fare essenzialmente 3 operazioni:
- immessi i dati per creare un albero orientato dalle foglie verso la radice, n-ario di profondità k, in cui ogni nodo e ogni arco abbiano degli attributi scelti dall’utente I cui valori siano randomici secondo un criterio voluto dall’utente, creare l’albero con le caratteristiche desiderate. Dopodiché dovrà poter salvare l’albero creato nel database con il nome deciso dall’utente (MEDIA)
- selezionato un albero qualsiasi dal database, cancellarlo (BASSA)
- scelti due nodi in un albero, se esiste un’orientamento che porta dal primo nodo scelto al secondo, calcolare la somma degli attributi di questi due nodi insieme a quelli degli altri nodi di passaggio e restituire ogni somma di ogni attributo + il percorso percorso + il tempo impiegato per l’operazione(ALTA)

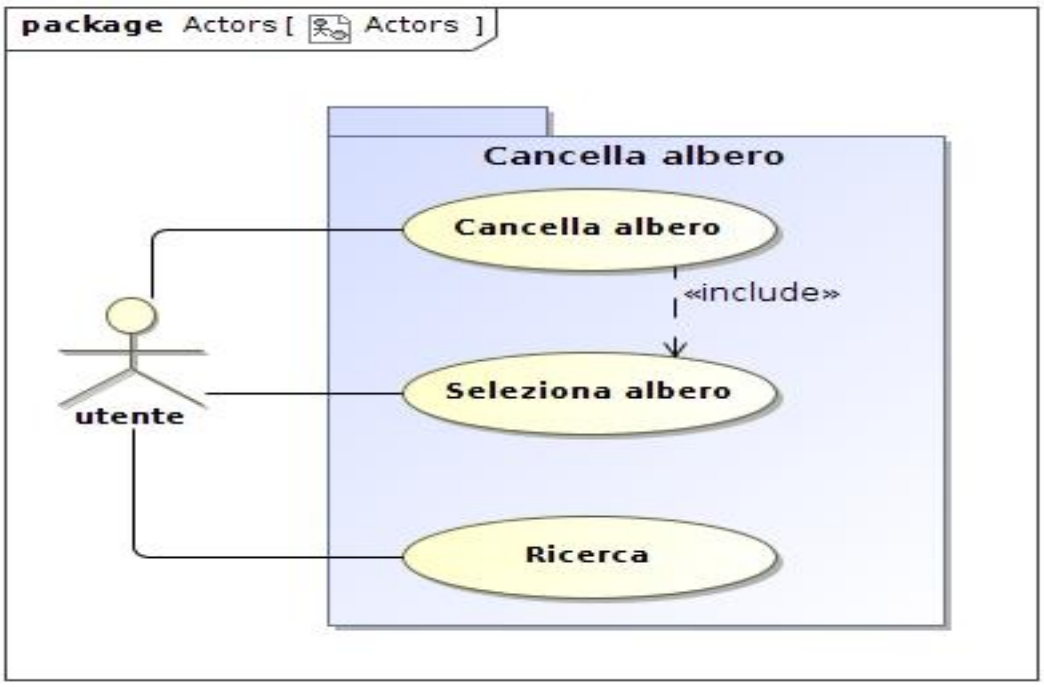
Queste 3 operazioni a loro volta si dividono in sottoperazioni, che talvolta possono essere in comune (come per esempio ricerca e seleziona in comune con cancellazione e esecuzione del calcolo). Le approfondiremo negli specifici campi di GUI, business e DB requirements.



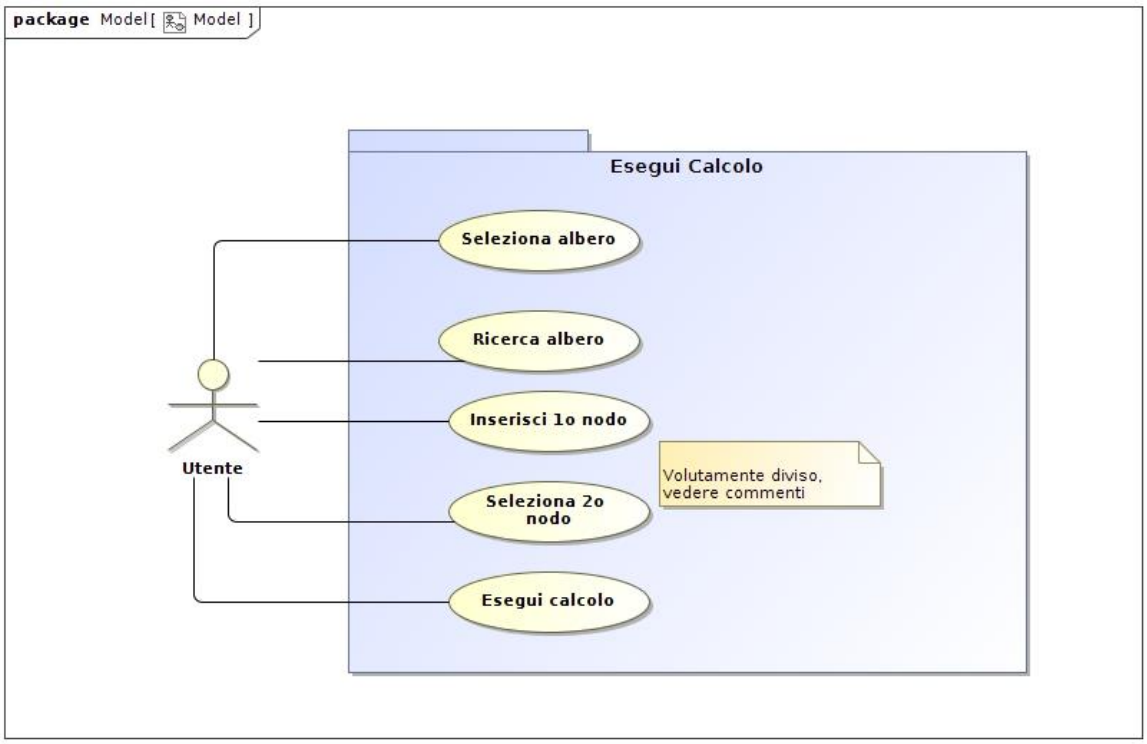
Basandoci sulle specifiche del progetto e sui chiarimenti che ci sono stati dati ai dubbi, abbiamo deciso che le operazioni principali da fare con questo sistema saranno:
costruzione dell'albero
cancellazione dell'albero
esecuzione dell'operazione
È stato deciso di fare un sistema completamente distribuito, l'unica operazione che avverrà in locale sarà la creazione della rappresentazione dell'albero, che userà la memoria volatile della macchina creatrice ma che quando dovrà salvare non salverà nel proprio disco fisso ma sul server.



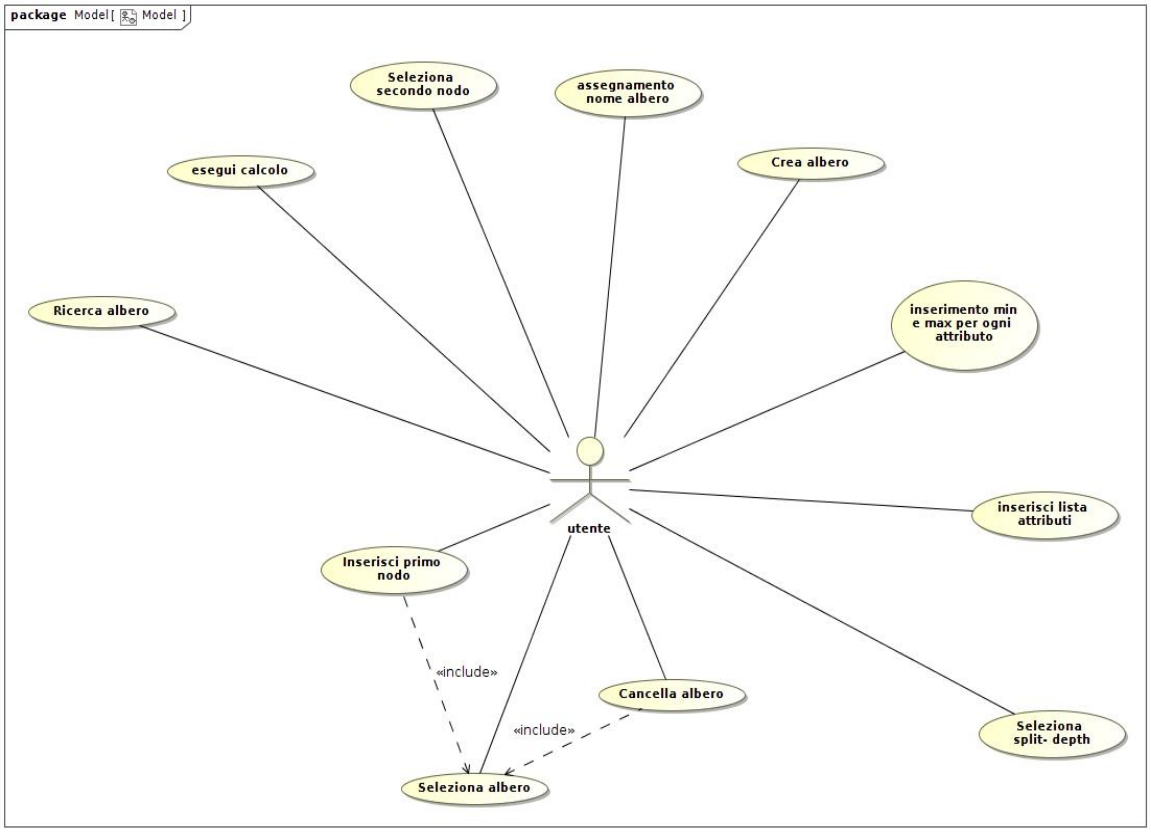
L'operazione di Build Tree è stata esplosa in 5 operazioni:
Selezionare split e size; inserimento degli attributi che si vogliono dare a ogni nodo e ogni arco dell'albero; decisione di minimo e massimo valori per attributo; attribuzione del nome; creazione dell'albero.
Rispetto al primo deliverable, abbiamo cancellato l'operazione di salvataggio in quanto abbiamo automatizzato tutto a un solo tasto.



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 per selezionare un albero si può fare la ricerca. Abbiamo tolto l'extend da ricerca a seleziona per il secondo deliverable perché, dopo aver ristudiato la sintassi di extend, abbiamo capito che l'operazione di ricerca, che avverrebbe prima del seleziona, il quale a sua volta NON ha come "possibile opzione" il ricercare, non estende la selezione.



Per l'esecuzione del calcolo abbiamo pensato, forse pure anzitempo, a un dettaglio implementativo: il numero di nodi dell'albero cresce in maniera esponenziale con la serie geometrica (k^(n+1)-1)/(k-1) 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 in un albero di split=2 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 di potenziali secondi nodi nodi tra cui scegliere. Quindi ricapitolando si selezionerà l'albero dopo un'eventuale ricerca, si inserirà il primo nodo, si selezionerà dalla nostra rosa il secondo e infine si farà eseguire il calcolo.



Una volta unite le esplosioni studiate una a una, lo Use Case Diagram esce così.

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	Salva struttura con nome
	4	Crea l'albero e riempi la struttura

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)

Attraverso l'interfaccia l'utente deve essere abilitato a decidere a quale macrofunzionalità accedere: di creazione, di cancellazione o di calcolo.
Dopo aver deciso la funzionalità, sempre attraverso l'interfaccia, l'utente può:

- inserire split and depth (alta)
- scegliere quali attributi deve avere ogni arco e ogni nodo (alta)
- scegliere la modalità di generazione di ogni attributo (media)
- creare effettivamente l'albero(alta)
- salvare l'albero con nome nel database(alta)

- Avere una lista di tutti alberi salvati nel database (bassa)
- Cercare un albero secondo un criterio (che può essere il nome quanto un ID, se queste due cose saranno diverse) (bassa)
- selezionare l'albero in sé (fare click sull'albero scelto) (alta)
- cancellare l'albero selezionato (bassa)

- scegliere il primo nodo nel calcolo (alta)
- selezionare il secondo nodo nel calcolo (alta)
- ordinare al programma di iniziare il calcolo (alta)

- Ovviamente deve poter visualizzare il risultato, altrimenti tutto il software sarà inutile

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

(Creazione)
- dopo che l'utente ha dato i parametri, la Business Logic crea l'albero vero e proprio seguendo le direttive date. (media)

(Calcolo)
- dopo aver selezionato un albero si sceglie un nodo di esso; se quel nodo esiste (la GUI NON fornisce un elenco di 2'000'000 di nodi tra cui scegliere e l'utente è libero, anche di sbagliare) la Business Logic deve fornire un insieme di vertici che possono creare un path se accoppiati al primo nodo; dopo che l'utente ha selezionato il secondo nodo, la Business Logic deve eseguire il calcolo degli attributi nodo per nodo, arco per arco, calcolare il tempo che viene impiegato e il path seguito. Infine ritornare il risultato che verrà visualizzato nella GUI(ALTA)

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

- Il database deve memorizzare tutti gli alberi che gli vengono richiesti di salvare (media)
- fornire una lista di tutti gli alberi che possiede quando gli viene chiesto (per poter fare l'operazione di seleziona) (Bassa)

- si devono poter cancellare alberi dal database(Bassa)
- deve interfacciarsi con la Business Logic (ALTA)

A1.4 Assumptions (da riempire a partire dalla Versione 2)

-Il database verrà toccato pochissimo. Sarà quasi tutto fatto dalla business logic. La creazione dell'albero come struttura vera e propria avverrà dalla business logic, nel database verrà inserito l'albero dopo la creazione.

-Tutti gli attributi saranno dello stesso tipo: real.

-Non dobbiamo occuparci di sicurezza.

-La creazione avviene una volta a settimana per utente

-Il calcolo avviene 3 volte al giorno per utente

-Non dobbiamo occuparci di differenziare I permessi perché tutti hanno stessi pemessi

- I permessi sono di cancellazione e calcolo per tutti gli alberi nel db; di creazione.

-Gli alberi non sono modificabili.

-La rappresentazione astratta dell'albero avrà una numerazione con etichetta del nodo radice = 1, e poi crescente da sinistra verso destra, dall'alto verso il basso, tutti INTERI.

Esempio: se split = 2, depth = 3:

lvl 0: nodo 1.

lvl 1: nodi 2 3

lvl 2: nodi 4 5 6 7

A1.5 Priorization (da riempire a partire dalla Versione 2)

Non inseriremo qui la prioritizzazione perché è già intrinsecamente presente sopra.

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

Development requirement

L'interfaccia deve essere sviluppata per forza di cose in html5, ed essere accessible via web

Product requirement

- Disponibilità

La micron è un'azienda che ha succursali in tutto il mondo e in tutte queste succursali il software dev'essere sempre disponibile e utilizzabile.

-Usabilità

Il nostro target di cliente è rappresentato da un tecnico micron, un utente con delle conoscenze medio- alte informatiche; in caso di necessità possiamo rendere il sistema poco user friendly, anche se sarebbe meglio evitare.

-Performance

Le performance sono il punto cardine del nostro software, ci sono stati posti vincoli stringenti, il calcolo dev'essere in massimo 30 secondi per 1 milione di nodi, mentre 60 secondi per 2 milione di nodi. 100 utenti devono poter eseguire calcoli in parallelo senza risentire dell'utilizzo condiviso del sistema.

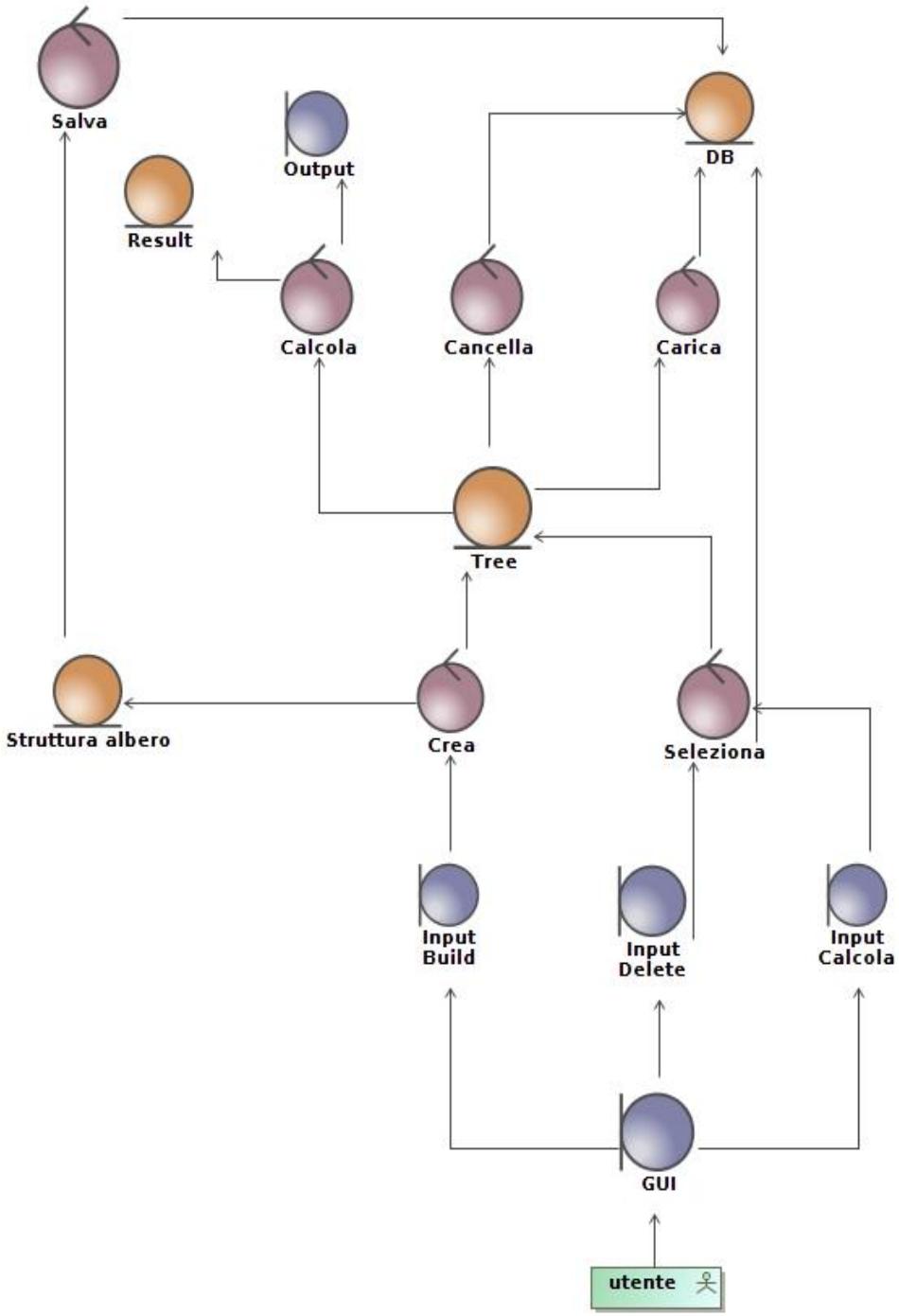
- Affidabilità

Quando l'utente avvia il software, deve essere sicuro che non ci sarà un business failure, evitando il rischio di avere forti perdite economiche

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 seleziona l'albero dal database e poi lo cancella; nell'ultimo caso seleziona l'albero dal database, il controller "calcola" esegue l'operazione e trova il risultato.

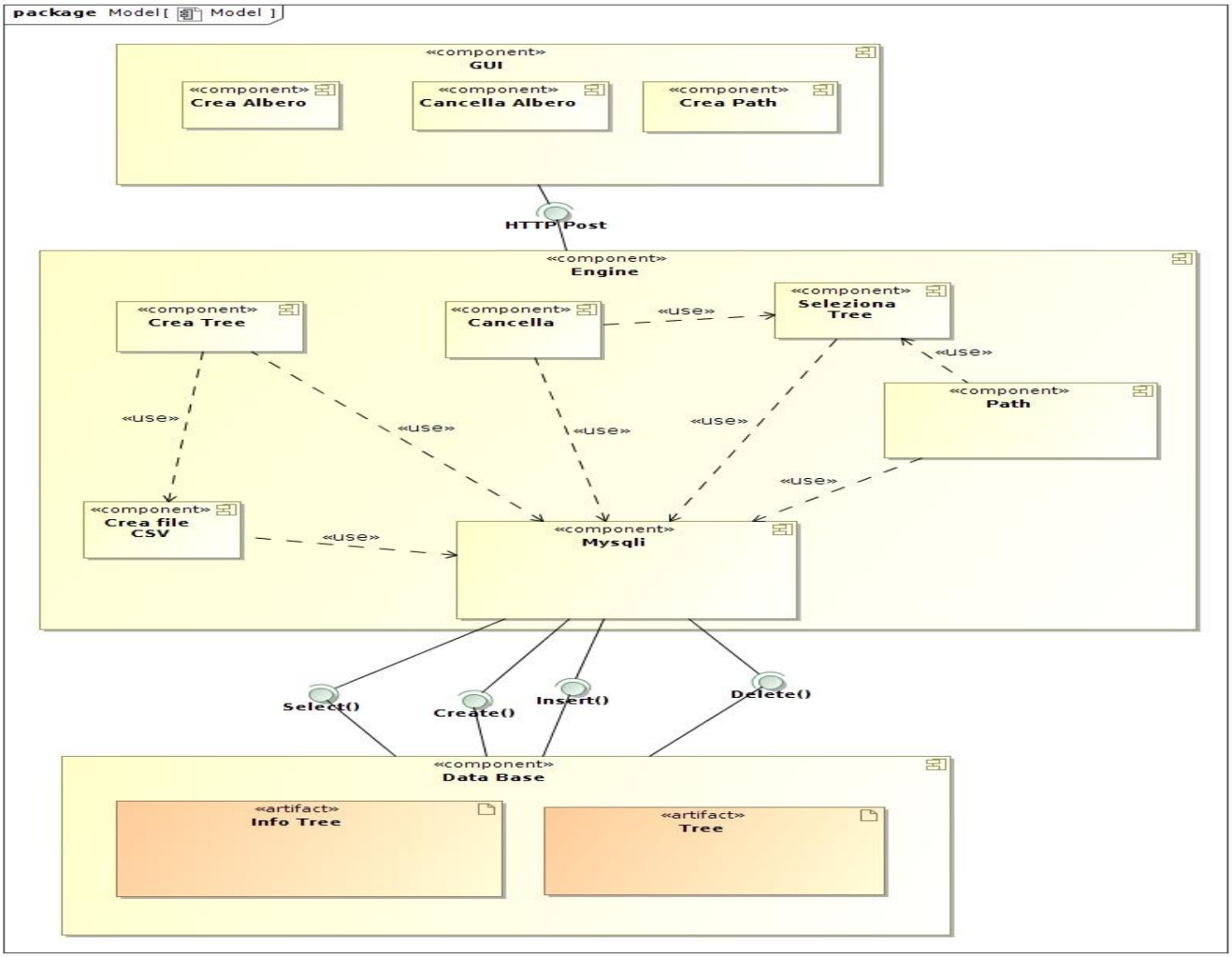
Rispetto alla versione precedente, abbiamo collegato il cancella al database perché esplicitamente consigliato dal professore; per quanto riguarda l'utilizzo del database manager, l'abbiamo reso controller di SELECT dal database, in modo tale che, insieme a salva, carica e delete formi il database manager che questa volta non è esplicitamente dichiarato.
Abbiamo anche messo il boundary output per indicare che il calcolo dovrà calcolare un result, ma questo result dovrà apparire in un output
Infine, abbiamo cambiato la struttura in modo che il salvataggio della struttura nel DB sarà una cosa diversa dal caricamento del contenuto in tale struttura; in particolare, la struttura dell'albero rappresenta la creazione di una tabella vuota con 1 colonna ID, N colonne che rappresentano gli attributi di ogni nodo, M colonne che rappresentano di attributi di ogni arco.
Per comodità nel modello, abbiamo considerato il file da caricare nel database come lo stesso albero che può essere selezionato, cancellato e sul quale si possono fare calcoli.

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

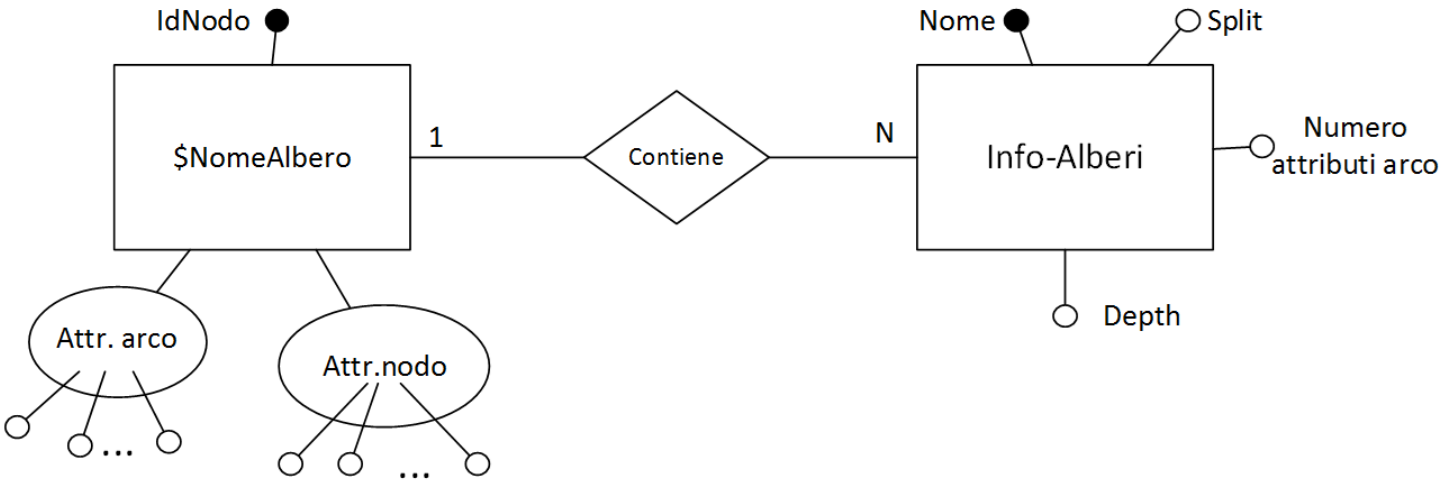
AVOID TO MAKE IT TOO COMPLEX AND FINE GRAINED. FOCUS MORE ON "HOW" THE COMPONENTS SHALL INTERACT IN ORDER TO SATISFY THE REQUIREMENTS, ADD INTERFACES and their parameters



WARNING: Il Sequence diagram si trova dopo il Class diagram e NON dopo il component.

D. ER Design

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

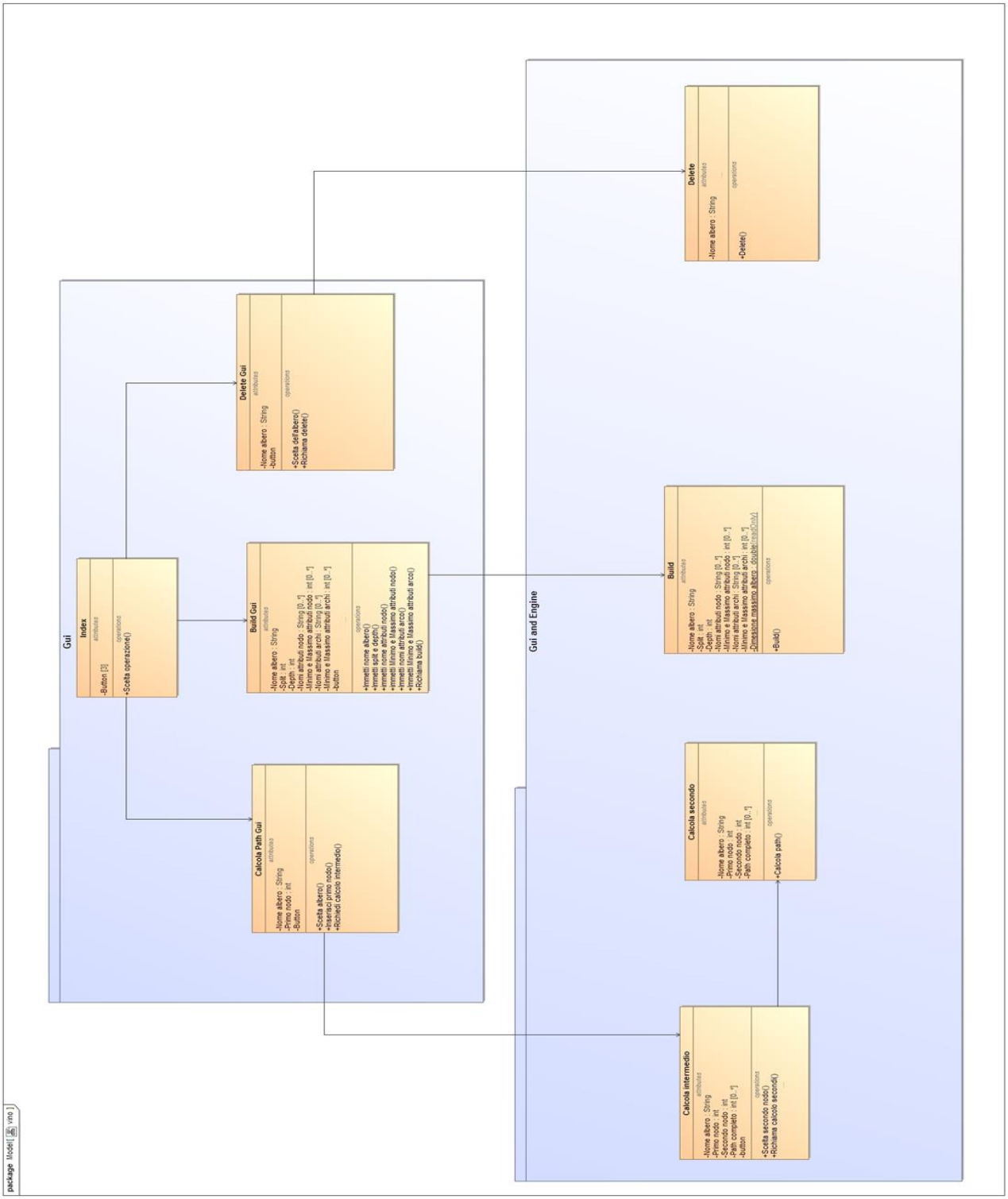


Il nostro db è composto da n+1 tabelle, dove n è il numero di alberi creati. Una che contiene tutte le info degli alberi presenti nel db, le altre n tabelle verranno rappresentate da invece una sola, generica.

Nella tabella info-alberi sono presenti: il nome dell'albero le cui informazioni sono presenti nella tupla che etichetta; l'informazione split di tale albero, la depth di tale albero; il numero di attributi che sono presenti nell'arco (per esempio, se un albero ha 7 attributi per tupla e nel suo corrispettivo infoalberi→numeroattributi c'è scritto 4, vorrà dire che I primi 3 attributi sono del nodo, gli ultimi 4 dell'arco verso il padre)

Nella tabella $NomeAlbero ci sono tante tuple quanti sono I nodi, ognuno etichettata da un intero crescente. Nella tupla etichettata dal questo intero ci sono N attributi del nodo che rappresenta, M attributi dell'arco di tale nodo diretto verso il padre.

E. *Class Diagram of the implemented System*



Premessa

Il paradigma usato per la scrittura del codice è procedurale non orientato agli oggetti. Per avere comunque una visione statica di tutto il sistema abbiamo adattato

la semantica per la costruzione del Class diagram, assumendo ogni script come classe: le proprietà rappresenteranno i principali dati usati in quel determinato file, il nome dei metodi rappresenta una descrizione delle azioni che intraprende il codice durante la sua esecuzione.

Classi

Index: in essa sono presenti 3 pulsanti che permettono di scegliere l'operazione da fare, e quindi di cambiare interfaccia.

Calcola Path Gui: esegue un'interrogazione sulla tabella Tree nel DB ed estrae i nomi di tutti gli alberi presenti; permette all'utente di sceglierne uno. Come altro input richiede il nodo dal quale partire per il calcolo del path nell'albero scelto. Usando il bottone richiama l'altra classe calcolo intermedio.

Calcolo intermedio: lo script **calcola path GUI** passa il nome dell'albero e il primo nodo del path. Dopodiché fa una query su Tree per sapere split e depth dell'albero in questione; usando queste informazioni calcola gli ID di tutti i nodi padri e offre all'utente una rosa dalla quale scegliere il secondo. L'utente premendo sul bottone richiamerà uno script che eseguirà la somma vera e propria.

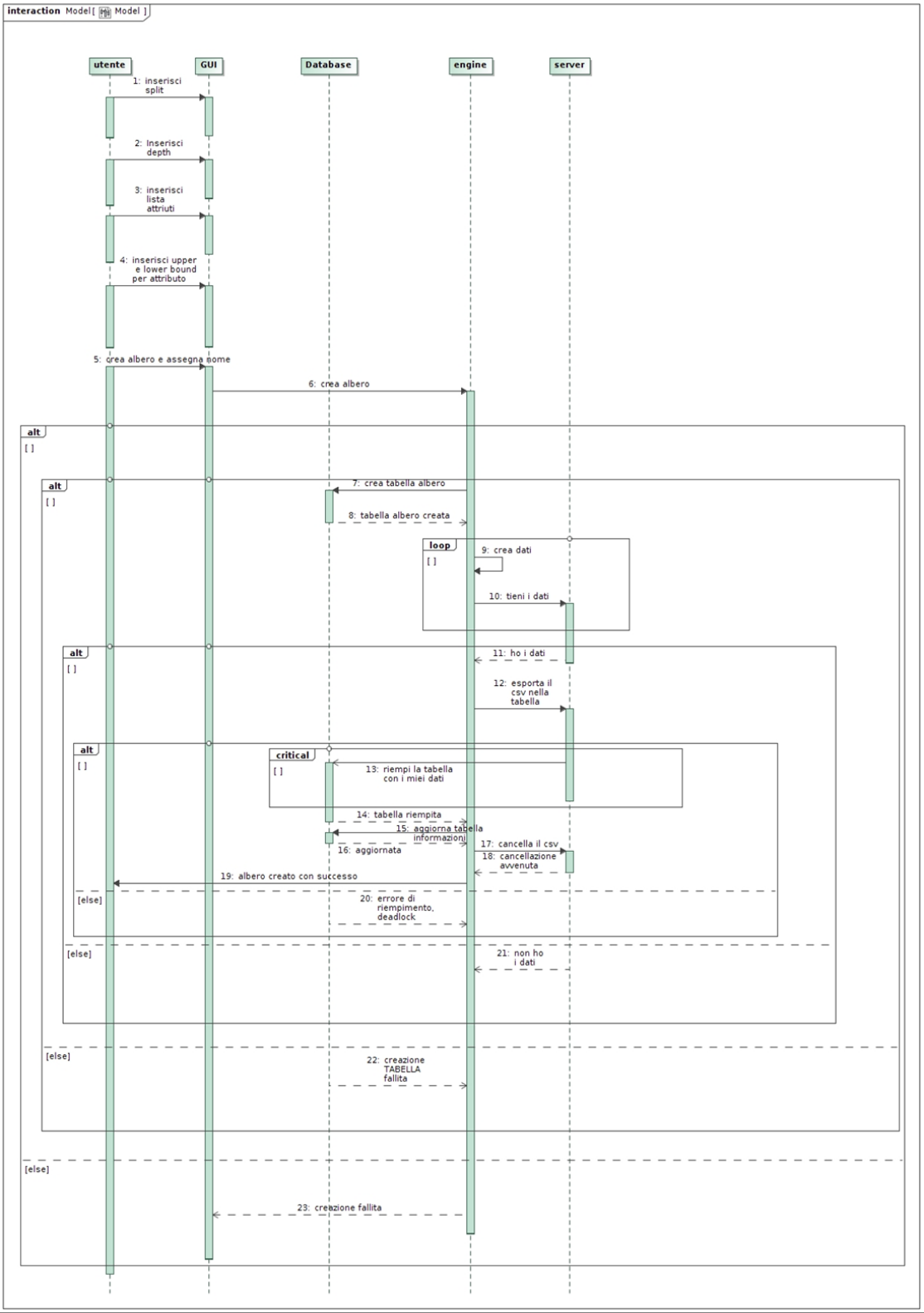
Calcola secondo: rappresenta **calcola effettivamente;** fa una query sulle tuple (nodi e archi) che hanno ID minore del primo vertice e maggiore del secondo vertice e che facciano parte del path. Calcola la somma di tutti gli attributi, eccezion fatta per gli attributi arco del nodo finale del path, e restituisce queste quantità all'utente.

Delete gui: descrive il file **Delete_inter**; interroga l'entità Tree e mostra all'utente tutti gli alberi presenti nel database. l'utente ne potrà scegliere uno e passarlo a delete.php premendo il bottone.
Delete: cancella sia l'albero passato da **delete gui** sia la tupla rappresentante quell'albero dalla tabella Tree
Crea Tree gui: L'utente pu decidere il nome dell'albero, le sue dimensioni (split e depth) , gli attributi presenti nei nodi o negli archi con i rispettivi valori di massimo e minimo. Premendo il bottone si richiama la creazione dell'albero vera e propria.
Crea Tree: Prende le informazioni passate da **Crea Tree Gui,** verifica che l'albero richiesto non sia troppo grande; crea la tabella albero e inserisce la tupla che lo rappresenta in Tree; crea un file CSV con dati coerenti alle volontà dell'utente; inserisce il contenuto del CSV dentro la tabella; cancella il file CSV.

Package

Gui: Racchiude tutte le classi (file) che si occupano di richiedere le informazioni all'utente.
Gui and Engine: Presenta le classi che rielaborano i dati da immagazzinare nell'server o da proporre all'utente.

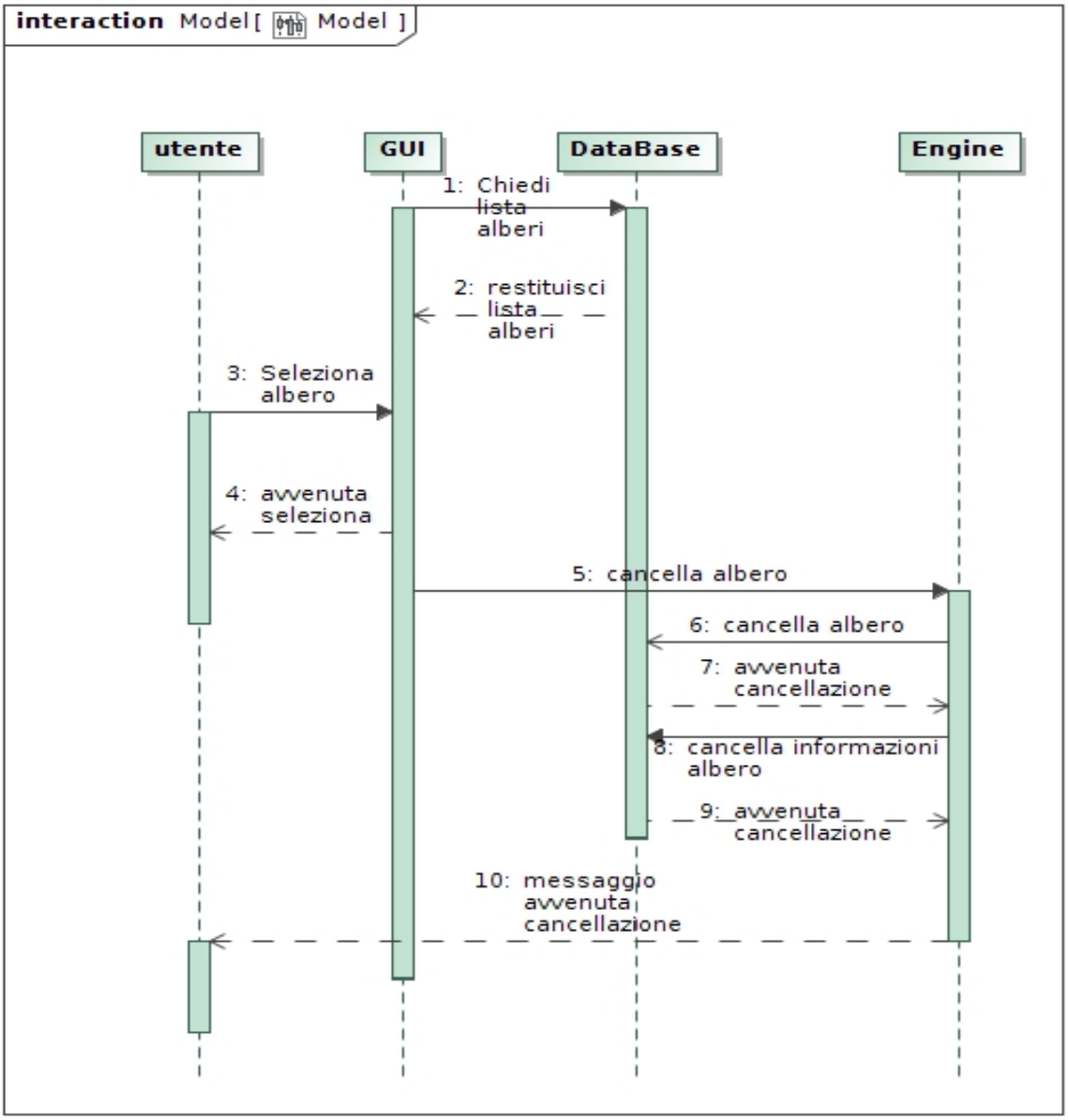
C.2 The dynamic view of the software architecture: Sequence



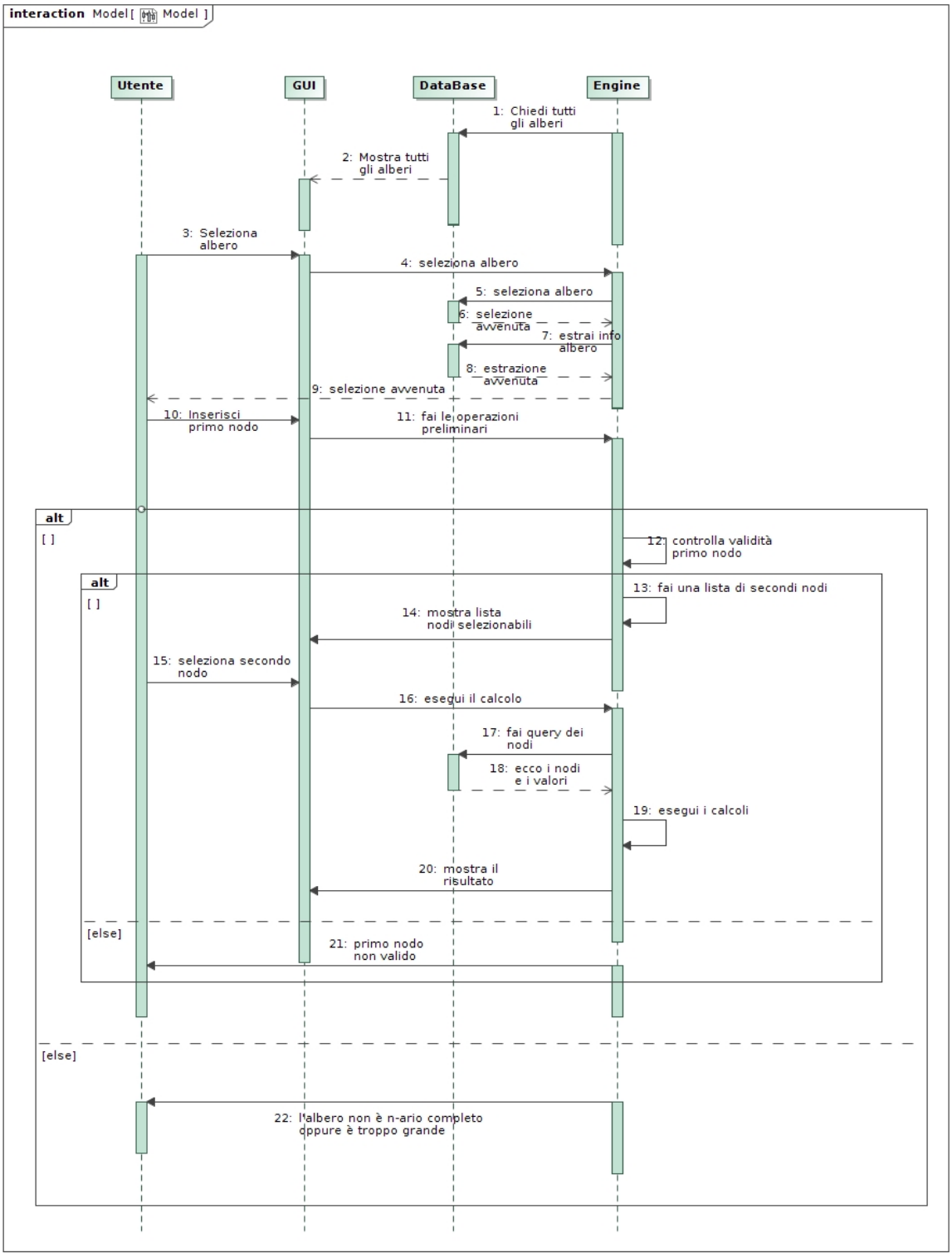
Diagram

BE SURE THAT THE STRUCTURE IS SYNCHRONIZED WITH THE DYNAMIC VIEW

L'utente inserisce split, depth, nomi degli attributi, valori lower e upperbound per ogni parametro e il nome dell'albero e preme "crea albero". Da lì sarà tutto più o meno automatizzato.
- l'engine prende I nomi degli attributi e crea la tabella sql con questi attributi più quello iniziale, ID, che sarà un intero crescente da 1 a seriegeometrica(k,n). Questa creazione può dare errore perché il nome dell'albero può già essere utilizzato da un altro.
- l'engine crea il file csv che sarà droppato dentro la tabella e lo salva nel server. Può dare errore.
- il csv così creato si vedrà inserito nella tabella precedente creata. Qui è una sezione critica perché si accede effettivamente al database a inserire, nel caso peggiore, 2000000 di tuple da 100 utenti.
- altra sezione critica è l'aggiornamento della tabella che contiene le informazioni sull'albero, in quanto lì c'è un vero rischio di deadlock in quanto condivisa
- infine l'engine cancella il csv dal server perché non serve(r) più.



Prima di effettuare la cancellazione, la gui chiede al database la lista degli alberi (delle tabelle) tra cui scegliere quale cancellare. L'utente seleziona l'albero e la GUI dice all'engine eseguire una query di cancellazione. Questa cancellazione rispecchierà la cancellazione della tupla dalle informazioni



Quando si accede alla pagina di calcolo l'engine effettua una query per restituire alla gui la lista di alberi presenti nella base di dati. l'utente sceglie l'albero. L'utente inserisce il primo nodo, e qui ci si diverte.

l'engine controlla se il numero inserito è minore del più grande ID presente nella tabella (o albero). Se è minore ok, se è maggiore c'è errore. Quindi si ricava split e depth dalla tabella con le info sugli alberi. Nella stessa tabella ci sta pure l'informazione per distinguere gli attributi dei nodi dagli attributi degli archi

Fatto ciò, mettiamo che non ci sia stato nessun errore. Il numero inserito dall'utente è un nodo presente nell'albero. L'engine trova il path totale senza toccare la base di dati. Prende il numero, aggiunge k (lo split) e sottrae 2, divide la quantità ottenuta per k e prende la parte intera del risultato; fa questa cosa ricorsivamente fino a 1, così ottiene. Abbiamo dimostrato che questa formula è corretta e, se richiesto, verrà fatta la dimostrazione. Questi numeri verranno proiettati a video all'utente che ne selezionerà uno. l'engine eseguirà la query sui nodi compresi tra i due selezionati dall'utente farà la sommatoria.

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:PHP

È stato scelto di usare PHP come linguaggio di sviluppo oltre che per fare la connessione con il database perché una volta creato il software in pseudocodice è stato notato che le operazioni da fare dall'engine sono molto leggere e molto generiche; l'operazione più dispendiosa sarà la creazione di un file di testo .csv con 2000000 di righe usando una funzione random, cosa che impiega pochi secondi con qualsiasi linguaggio. Non essendoci bisogno di costrutti complessi che un linguaggio potrebbe implementare meglio di un altro, abbiamo optato per usare direttamente php.

DBMS: MySQL

La scelta del DBMS è stata la più sofferta, che ci ha portato a una fase di over engineering. La precisazione che nessun albero dovrà essere modificato è stata una semplificazione che ha reso qualsiasi dbms potenzialmente adatto ai nostri scopi. Abbiamo inizialmente deciso MongoDB perché da come ne si parla nei forum, pare potentissimo. Però il learning è risultato oltremodo difficile e abbiamo abbandonato l'idea di usarlo. A questo punto ci siamo orientati verso OrientDB; confrontandoci con gli altri gruppi e notando che questi altri gruppi hanno scelto neo4j, abbiamo splittato il gruppo in 2: una parte a studiare orient, un'altra neo4j. Si è aggiunto come terzo database da studiare anche mySQL perché osservando la funzione matematica che crea i collegamenti tra i vari nodi, si è notato che effettivamente non c'è alcun bisogno di creare un albero se nell'engine ci sta un'astrazione per l'utente che gli faccia credere di stare lavorando su un albero che in realtà è solo una tabella (insomma, tipo tabella HASH). Il gruppo si è quindi diviso in 3 studi sui 3 database da usare. OrientDB è stato scartato perché troppo poco documentato. MySQL ha creato dei problemi con i tempi di inserimento (non richiesti, ma comunque un inserimento di oltre un'ora crea problemi all'utente, alla macchina e rischia pure, oltre al danno la beffa, di non venire caricato), ma una volta bypassati i controlli, cambiato lo storage engine e ottimizzato l'utilizzo dei csv, si è dimostrato perfettamente in grado di gestire 2000000 di nodi. Anche neo4j ha passato il "controllo qualità", ma visto che mySQL non ha bisogno di tanto learning in quanto stiamo usando phpmyadmin in un altro corso e visto che abbiamo pure i template per connetterci ai database già pronti e utilizzabili, abbiamo finalmente optato per MySQL.

RAPPRESENTAZIONE DELL'ALBERO

L'albero non sarà rappresentato da archi e nodi propriamente detti ma da un'unica tabella che avrà N+M+1 colonne: N colonne attributi nodo, M colonne attributi dell'arco da nodo che contiene l'informazione a suo padre, 1 per l'ID; ci siamo potuti permettere di accorpare l'arco nel nodo perché gli archi sono orientati e da ogni nodo esce UN SOLO arco verso il padre. Quindi gli attributi di ogni singolo nodo saranno quelli propri del nodo e quelli dell'arco diretto verso il padre

RECUPERO DELLE INFORMAZIONI

Il database è formato da una tabella generica (che poi nella realtà saranno N, una per ogni albero creato) e da infonodi. Nella tabella infonodi ci sono nome dell'albero, split, depth e numero di attributi dell'arco verso il padre (vedere commenti su database).

FUNZIONI MATEMATICHE

Il nostro sistema si basa su due funzioni matematiche:
- la prima è la funzione geometrica e ci serve per la CREAZIONE. Grazie ad essa possiamo ricavare in un'operazione il numero totale di nodi che avrà un albero utilizzando split e depth; numereremo quindi ogni nodo in modo crescente da 1 a (funzione geometrica(split, depth)).
- la seconda serve per il CALCOLO, non sappiamo se ha un nome proprio, l'abbiamo ricavata empiricamente e poi abbiamo dimostrato che, seguendo l'assunzione che nella nostra astrazione i nodi degli alberi siano numerati da sinistra verso destra, dall'alto verso il basso, funziona.

Inserito il numero-etichetta (v) del nodo, si può risalire a suo padre aggiungendo lo split (k), sottraendo 2 e dividendo il numero così ottenuto di nuovo per lo split (k); del numero così ottenuto si prende la parte intera.
Parteinteradi: (v+k-2)/k;
Utilizzandola ricorsivamente si può risalire di padre in figlio fino alla radice. Maggiori dettagli nel sequence diagram.

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>

Le funzionalità richieste dall'utente sono creazione dell'albero, cancellazione e calcolo del path. L'utente vuole poter decider gli attributi di nodi e archi dell'albero, i valori minimi e massimi, lo split e la depth. Il sistema deve creare un albero completo di split e depth scelti, attributi con valori randomici entro il delta deciso dall'utente. Effettivamente noi non creiamo un albero in quanto tale, ma non è un problema perché virtualmente l'utente crede di lavorare con un albero.
Tutti i requisiti funzionali richiesti dall'utente sono soddisfatti.
I requisiti non funzionali sono

- Disponibilità

È soddisfatta perché ogni albero è situato nella propria tabella, quindi non ci sono problemi di aggiornamento, inserimento di nuovi nodi in tabelle condivise. O meglio, l'unica tabella condivisa è soggetta di un centinaio d'inserimenti a settimana (scrittura) e la lettura è fatta su una sola tupla.

-Usabilità

Nonostante l'abbiamo messo come meno importante tra i requisiti non funzionali, in quanto la micron è un'azienda in cui bene o male tutti sono avvezzi all'uso di tecnologie informatiche, siamo riusciti a soddisfarla automatizzando gran parte del processo.

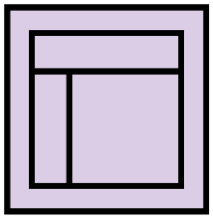
-Performance

Abbiamo cambiato 4 dbms prima di riuscire a capire che i limiti di performance in realtà non esistevano e che le performance richieste non sono niente di particolarmente difficile da raggiungere. abbiamo soddisfatto questo vincolo non funzionale, e abbiamo reso performante non solo il calcolo, ma anche la creazione, attraverso questa velocità riusciamo a gestire 100 utenti .

- Affidabilità

Il software è affidabile quanto disponibile. Ha solide basi su funzioni matematiche formalmente dimostrate. È affdabile perché ogni tabella/albero è fine a se stessa e non deve dipendere da nessun sistema esterno. L'unica dipendenza che potrebbe rendere inaffidabile il sistema è quella dell'albero dalla tabella infonodi, senza la quale non si può calcolare il path, ma quello è più un problema di robustezza. E il problema non esiste veramente perché se non si riesce ad accedere a quell'informazione il sistema non può neanche avere un errore che dia valori sbagliati in quanto non può eseguire il calcolo del path preventivo e necessario per la query.

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.

141h Doing; 67h 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>

Guida installazione:

000AAA Web Planner Path Calculator 3.0\Guida e Dump Database\guida.txt

Dump del database:

000AAA Web Planner Path Calculator 3.0\Guida e Dump Database\Progetto.sql

Viene fornito a correlo il codice completamente commentato.