**“Software Engineering”**

**Course**

**a.a. 2016-2017**

**Template version 2.0**

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

**Planner Path Calculator version v2 Deliverables**

|  |  |
| --- | --- |
| **Date** | 23/12/2016 |
| **Deliverable** | 2 |
| **Team (Name)** | Comaiam |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name & Surname** | **Matriculation Number** | **E-mail address** |
| Davide De Marco | 228737 | davide.demarco@student.univaq.it |
| Francesco Fuschi | 239266 | francesco.fuschi@student.univaq.it |
| Alice Laraspata | 235943 | alice.laraspata@student.univaq.it |
| Nicolò Paoletti | 235800 | nicolo.paoletti@student.univaq.it |
| Mario Vetrini | 238806 | mario.vetrini@student.univaq.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 Tasks](#List_of_challenging_tasks)
* [A. Requirements Collection](#Requirements_collection)
* [A.1 Functional Requirements](#Functional_requirements)
  + [A.1.1 GUI Requirements](#GUI_requirements)
  + [A.1.2 Business Logic Requirements](#BL_requirements)
  + [A.1.3 DB Requirements](#DB_requirements)
* [A.2 Non Functional Requirements](#NF_requirements)
* [A.3 Content](#Content)
* [A.4 Assumptions](#Assumptions)
* [A.5 Prioritization](#Prioritization)
* [B. Analysis Model](#Analysis_model)
* [C. Software Architecture](#Software_architecture)
* [C.1 Component Diagram](#Component_diagram)
* [C.2 Sequence Diagram](#Sequence_diagram)
* [D. ER Design](#ER)
* [E. Class Diagram](#Class_diagram)
* [F. Design Decision](#Design_decisions)
* [G. Requirements Through Design](#Requirements_through_design)
* [H. Effort Recording](#Effort_recording)

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.>*

|  |  |  |  |
| --- | --- | --- | --- |
| **Challenging**  **Task** | **Date the task is identified** | **Date the challenge is resolved** | **Explanation on how the challenge has been managed** |
| Scelta del metodo di calcolo del path | 14/11/2016 | 14/11/2016 | Valutando varie possibilità, si è ritenuto più conveniente (in termini di tempo e complessità) calcolare la somma degli attributi del path direttamente sul DB. |
| Dimenticanza da parte dell’utente del nome dell’albero | 15/11/2016 | 15/11/2016 | Si è deciso di strutturare l’interfaccia in modo tale da permettere all’utente di ritrovare l’albero con facilità (ad esempio mettendo i nomi di tutti gli alberi all’interno di una lista scorrevole e ordinabile). |
| Controllo di concorrenza nella cancellazione di un albero | 13/12/2016 | 13/12/2016 | Si è deciso di inserire un controllo da parte della BL, la quale dovrà verificare se l’albero selezionato dall’utente è ancora presente del DB. |
| Scelta del metodo di creazione dell’albero | 14/12/2016 | 14/12/2016 | Notando che la creazione dell’albero tramite l’utilizzo di una query richiedeva un tempo troppo elevato, si è deciso di utilizzare dei file .CSV creati tramite PHP |
| Gestione di un albero con Split Size = 1 | 21/12/2016 | 21/12/2016 | Notando che le varie operazioni su un albero con Split Size = 1 richiedevano un tempo troppo elevato, si è deciso di mantenere i file .CSV nel server così da effettuare le operazioni direttamente su quest’ultimo, anziché strutturarli nel DB |

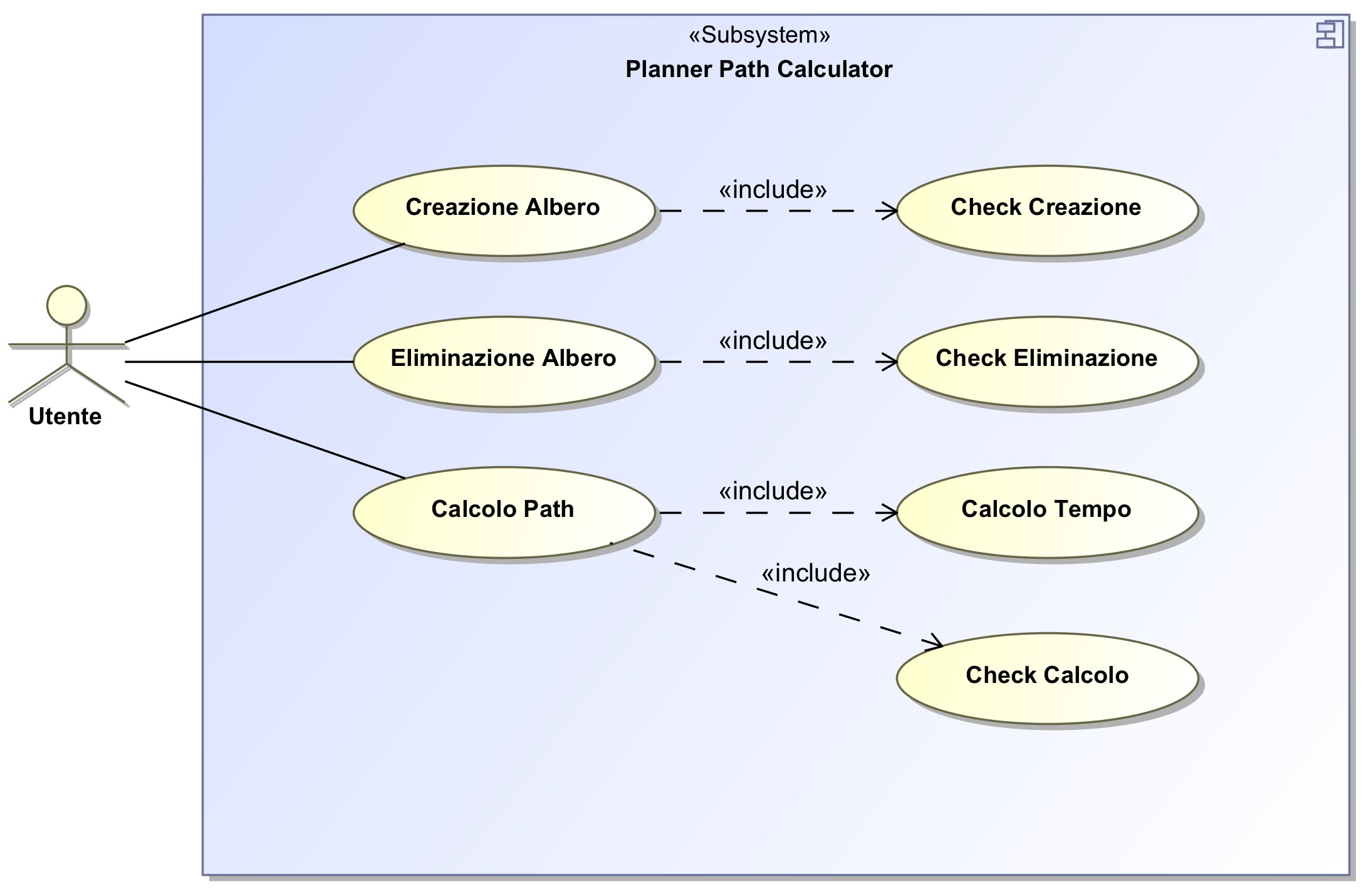
A. Requirements Collection

*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>*

**FIGURA 1: USE CASE DIAGRAM**



Il Planner Path Calculator (PPC) è un sottosistema che permette di generare e/o rimuovere alberi che soddisfino determinate caratteristiche e condizioni ed effettuare su di essi calcoli e operazioni.

Per eseguire tali operazioni abbiamo bisogno di tre differenti sottosistemi del PPC:

* GUI
* Business Logic (BL)
* Database (DB)

Per ognuna delle tre funzionalità del sistema è incluso un diverso check di verifica della correttezza dei dati inseriti (suddiviso in GUI-check e DB-check). Il GUI-check e il DB-check verranno approfonditi successivamente nei rispettivi requisiti. Per il caso d’uso di “Calcolo Path” è incluso anche il “Calcolo Tempo”; anch’esso verrà approfondito successivamente.

A1.1 GUI Requirements

La GUI è l’interfaccia grafica che permette all’utente la possibilità di scegliere tra i 3 servizi offerti dal PPC. A seconda della scelta, la pagina si aggiornerà in modo dinamico e mostrerà all’utente la rispettiva pagina per l'inserimento dei dati e la visualizzazione dei risultati.

* Per la creazione dell’albero, la GUI prende in input dall’utente i dati riguardanti le dimensioni e il nome dell'albero, il nome degli attributi (di vertici ed archi) e la regola di generazione dei loro valori (la quale può avere un range di valori stabilito dall'utente). La GUI esegue una prima fase di check sui dati inseriti (GUI-check). Nel caso in cui i dati immessi siano incorretti, la GUI mostra un messaggio di errore contenente i dati da dover modificare. In caso contrario, la GUI manda i dati alla BL; quest’ultima può restituire un messaggio di avvenuta creazione o un messaggio di errore. La GUI provvede a visualizzare tale messaggio all’utente. (**priorità alta**)
* Per l’eliminazione dell’albero, la GUI fornisce all’utente la lista ordinata (per nome o per data) degli alberi presenti nel DB, nella quale l’utente può selezionare quale tra questi eliminare. La GUI controlla che sia stato selezionato un nome (GUI-check). Nel caso in cui ciò non avvenga, la GUI mostra un messaggio di errore contenente il dato da dover selezionare. In caso contrario, la GUI manda il dato alla BL; quest’ultima può restituire un messaggio di avvenuta eliminazione o un messaggio di errore. La GUI provvede a visualizzare tale messaggio all’utente. (**priorità media**)
* Per il calcolo del path, la GUI fornisce all’utente la lista ordinata (per nome o per data) degli alberi presenti nel database. L’utente dovrà selezionare un albero e inserire due dei suoi vertici. La GUI esegue una prima fase di check sui dati inseriti (GUI-check). Nel caso in cui i dati immessi siano incorretti, la GUI mostra un messaggio contenente i dati da dover modificare. In caso contrario, la GUI manda i dati alla BL; quest’ultima può restituire un messaggio di avvenuta operazione (contenente inoltre il risultato e il tempo impiegato) o un messaggio di errore. La GUI provvede a visualizzare tale messaggio all’utente. (**priorità media**)

Specifichiamo che la tipologia di check effettuata dalla GUI è limitata alla sola correttezza dei tipi di dati inseriti (es. caselle lasciate vuote, alberi non selezionati, altezza o numero di figli negativo o decimale, ecc…)

**Assunzione**: Abbiamo deciso di attribuire priorità alta alla creazione dell’albero in quanto, senza tale operazione, gli altri due servizi non potrebbero essere eseguiti.

A1.2 Business Logic Requirements (BL)

La Business Logic è il sottosistema che si occupa di effettuare o “lanciare” i servizi offerti dal PPC.

* Per la creazione dell’albero, la BL prende in input dalla GUI le dimensioni e il nome dell'albero, il nome degli attributi (di vertici ed archi) e la regola di generazione dei loro valori (compreso, se disponibile, del valore di range). La BL, in primo luogo, manda al DB il nome dell’albero per controllare che non sia già in uso. In caso non lo sia, la BL crea dei file .CSV da inviare al DB; quest’ultimo può restituire un messaggio di avvenuta creazione o un messaggio di errore. La BL provvede a restituire tale messaggio alla GUI. (**priorità alta**)
* Per l’eliminazione dell’albero, la BL prende in input dalla GUI il nome dell’albero da eliminare all’interno del DB e lo invia a quest’ultimo; il DB può restituire un messaggio di avvenuta cancellazione o un messaggio di errore. (**priorità media**)
* Per il calcolo del path, la BL prende in input il nome dell’albero e due dei suoi vertici e li manda, in primo luogo, al DB per controllare che l’albero esista. In caso non lo sia, la BL restituisce alla GUI un messaggio di errore. In caso di avvenuta operazione, il DB restituisce un messaggio di successo (contenente inoltre il risultato) e la BL lo restituisce alla GUI (insieme al tempo impiegato). Altrimenti, un messaggio di errore. (**priorità media**)

**Assunzione**: Abbiamo deciso di attribuire priorità alta alla creazione dell’albero in quanto, senza tale operazione, gli altri due servizi non potrebbero essere eseguiti.

A1.3 DB Requirements

Il DB è il sottosistema che si occupa di gestire e strutturare gli alberi creati dal PPC.

* Per la creazione dell’albero, il DB riceve in input dalla BL il nome dell’albero e verifica, in primo luogo, se non sia già in uso. In caso lo sia, Il DB restituisce alla BL un messaggio di errore, altrimenti le comunica che tale nome non è in uso. Il DB riceve così dalla BL i file .CSV contenti l’albero e li struttura al suo interno, restituendo un messaggio di avvenuta creazione. Altrimenti, restituisce alla BL un messaggio di errore. (**priorità alta**)
* Per l’eliminazione, il DB riceve in input dalla BL il nome dell’albero e verifica, in primo luogo, se esiste. In caso non esista, il DB restituisce alla BL un messaggio di errore, altrimenti elimina l’albero e comunica alla BL un messaggio di successo. (**priorità media**)
* Per il calcolo del path, il DB riceve in input dalla BL il nome dell’albero e due dei suoi vertici e verifica, in primo luogo, se l’albero esiste (DB-check). In caso non esista, il DB restituisce alla BL un messaggio di errore. Il DB-check, inoltre, si occupa di verificare la presenza dei vertici all’interno dell’albero selezionato e di verificare la presenza di un path che li colleghi. In caso di avvenuta operazione, il DB restituisce alla BL un messaggio di successo (contenente inoltre il risultato). Altrimenti, un messaggio di errore. (**priorità media**)

Specifichiamo che la tipologia di check effettuata dal DB riguarda l’integrità dei dati presenti al suo interno (es. verificare che l’albero da voler eliminare non sia stato già eliminato, verificare l’esistenza dei vertici in un determinato albero durante il calcolo del path, verificare l’esistenza di un cammino tra i due vertici, ecc…)

A.2 Non Functional Requirements

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

* Il database non ha limiti nella memorizzazione di alberi.
* Il sistema principale si occupa della sicurezza e della visibilità dei dati.
* Il tempo di creazione di un albero non è rilevante data la rarità della sua esecuzione (una volta a settimana).
* Il tempo di calcolo del path su un albero da 2.000.000 di nodi deve essere inferiore ai 60 secondi. Oltre i 2.000.000 è accettabile un andamento esponenziale della velocità di calcolo.
* Il range (K-N) deve essere differente per ogni categoria di attributo. I valori possono essere sia interi che decimali.
* Gli alberi possono essere visualizzati e cancellati da qualsiasi utente, non solo da chi li ha inizialmente generati.
* Gli alberi, una volta salvati nel DB, non possono essere modificati.
* Il DBMS da utilizzare deve essere MSSQL o open-source.

A.3 Content

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

Dopo una prima fase di testing riguardante la creazione di un albero tramite query eseguita direttamente sul DB, ci si è accorti che tale operazione richiedeva un tempo superiore rispetto al previsto. Si è dunque deciso di optare per l’utilizzo di un file .CSV tramite PHP in modo da abbattere drasticamente i tempi. Ciò ha comportato la perdita del dato gerarchico offerto da MSSQL. Di conseguenza, si è deciso di abbandonarne l’utilizzo ed utilizzare MySQL, essendo più conosciuto da parte del team.

A.4 Assumptions

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

* Assumiamo che non sia necessario fornire le credenziali di accesso al sistema in quanto è un sottosistema di un software in cui è già previsto un sistema di autenticazione.
* Assumiamo che il PPC giri su un server Micron.
* Assumiamo che il server vada in timeout dopo 10 minuti
* Assumiamo che la GUI non permetta all’utente di bloccare un processo una volta avviato

***A.5 Prioritization***

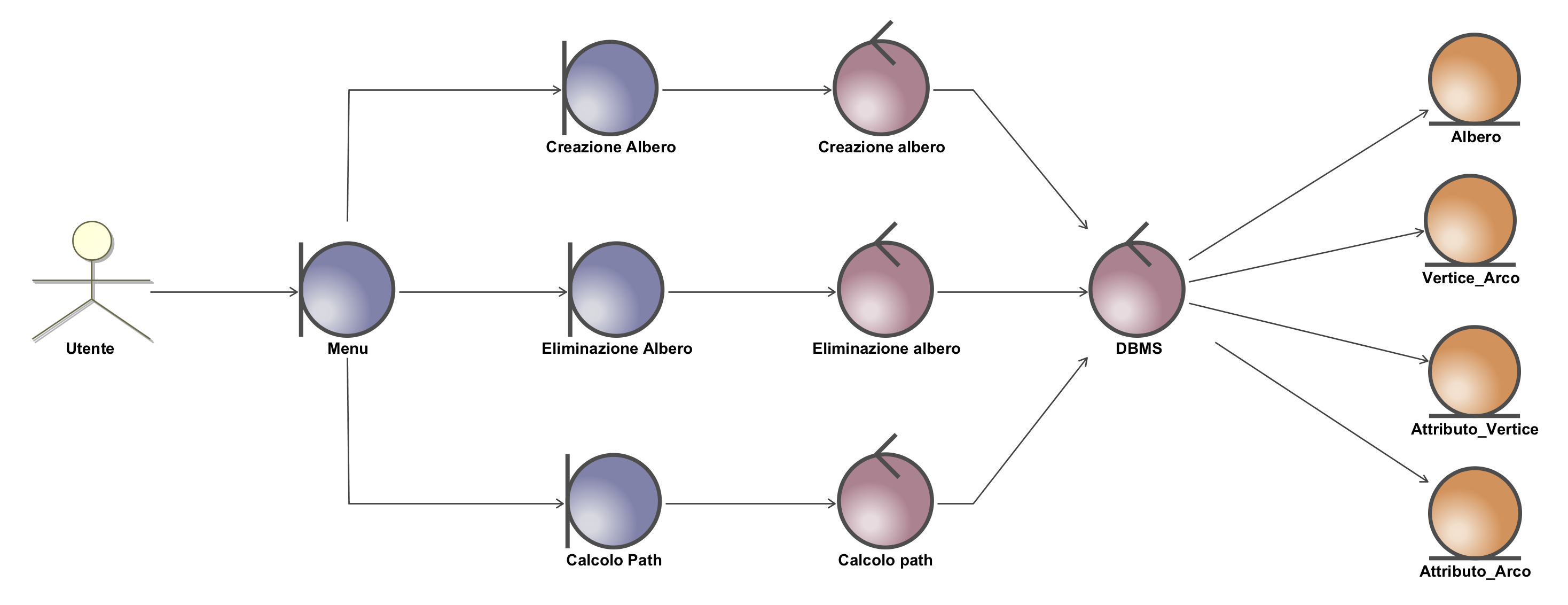
Requisiti listati secondo ordine di priorità:

* Programma di “Creazione Albero”
* Programma di “Calcolo Path”
* Programma di “Eliminazione Albero”
* DB
* GUI

B. Analysis Model

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

**FIGURA 2: ROBUSTNESS**



L’utente si interfaccia con il *boundary* del menù, dal quale può selezionare uno dei tre *boundaries* presenti:

* Creazione Albero
* Eliminazione Albero
* Calcolo Path

Il sistema prosegue nell’esecuzione di uno dei tre *controllers,* rispettivamente:

* Creazione Albero
* Eliminazione Albero
* Calcolo Path

Il controllo passa poi al DBMS che esegue l’operazione richiesta su una o più *entities*:

* Albero
* Vertice\_Arco
* Attributo\_Vertice
* Attributo\_Arco

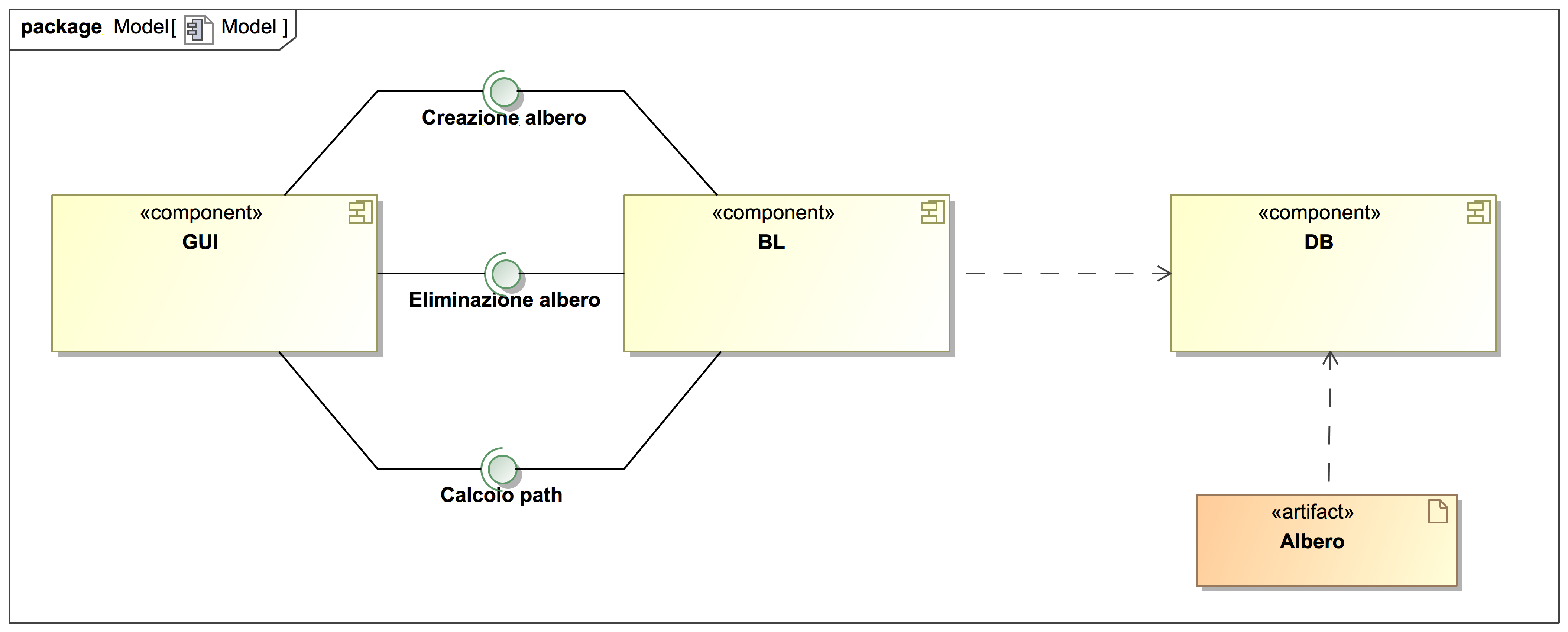
Nel Robustness Diagram non sono presenti i 3 *controllers* riguardanti il check degli errori poiché tali features sono gestite direttamente dai *controllers*. In più, l’entità “Albero” ha al suo interno le business rules necessarie a controllarne l’integrità e la correttezza.

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.1The static view of the system: Component Diagram

**FIGURA 3: COMPONENT DIAGRAM**

****

Il Component Diagram mostra la divisione dell’intero sistema in tre componenti:

* GUI
* BL
* DB

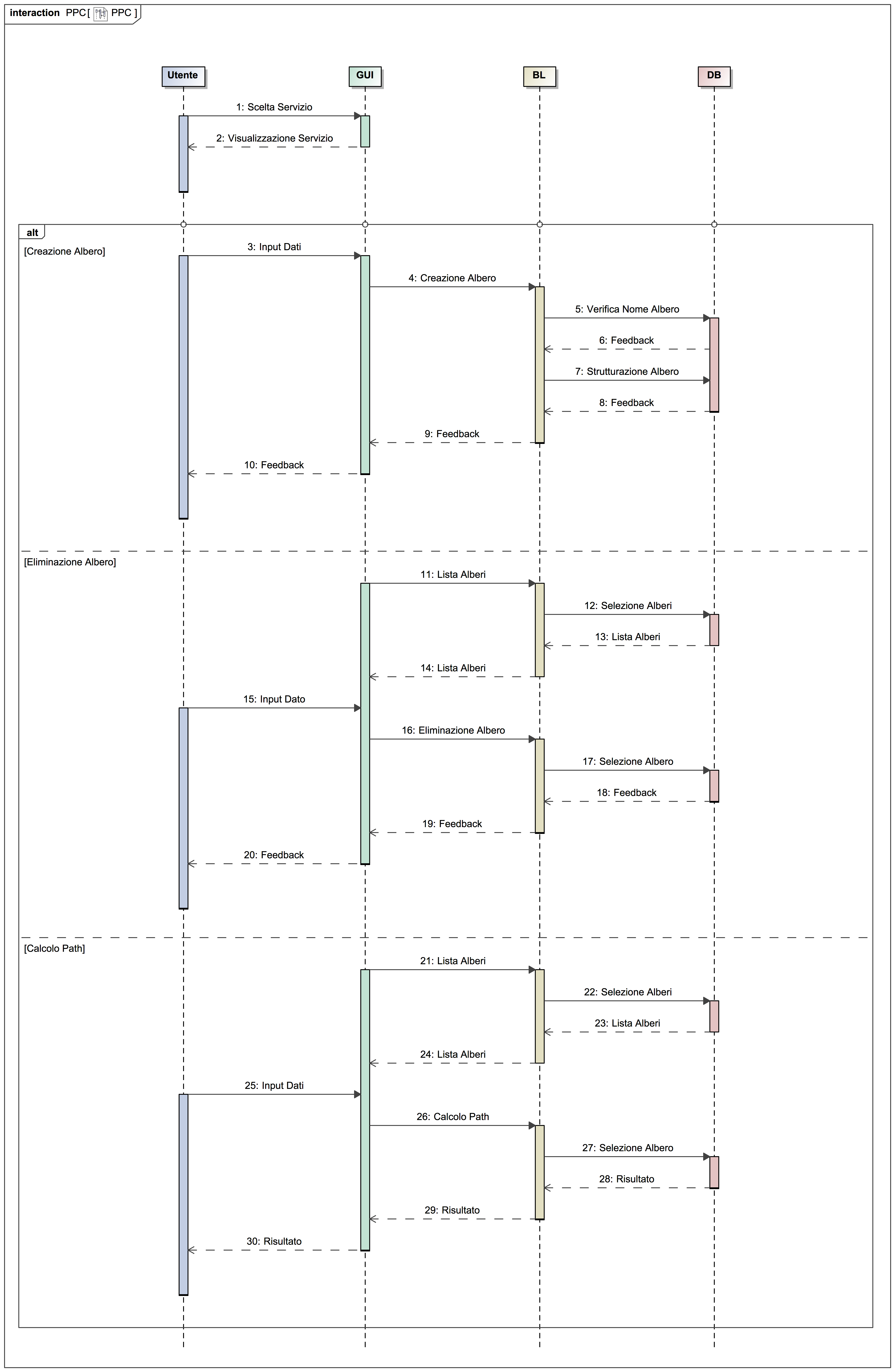
La GUI può richiedere tre diversi servizi: “Creazione Albero”, “Eliminazione Albero” e “Calcolo Path”, forniti dalla BL, la quale a sua volta dipende dal DB.

Il DB, a seconda del servizio richiesto, elimina gli alberi o struttura al suo interno i dati generati dalla BL.

L’artefatto albero dipende dal DB, in quanto è la rappresentazione fisica dei dati inseriti dagli utenti.

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

**FIGURA 4: SEQUENCE DIAGRAM**



Il Sequence Diagram è un diagramma utile per modellare la comunicazione tra oggetti in relazione al trascorrere del tempo.

Il Sequence Diagram riportato corrisponde ad uno specifico caso di successo, in cui cioè l’esecuzione risulta corretta e priva di errori o eccezioni.

Le LIFELINE presenti all’interno del nostro sistema sono:

* UTENTE
* GUI
* BL
* DB

L’UTENTE, come risulta visibile dal diagramma realizzato, si interfaccia esclusivamente con la GUI scegliendo uno dei tre servizi proposti: “Creazione Albero”, “Eliminazione Albero” e “Calcolo Path”.

A questo punto il flusso si divide in tre sotto flussi, ognuno adibito a soddisfare un servizio differente:

**“Creazione Albero”:** l’utente inserisce i dati in input tramite la GUI che, dopo aver verificato la correttezza dei dati inseriti (GUI-check), li invia alla BL. Quest’ultima, tramite query, verifica l’esistenza dell’albero nel DB (DB-check), il quale restituisce un feedback positivo con il quale la BL elabora i dati per effettuare l’operazione di creazione dell’albero. Il DB struttura tali dati al suo interno. Il FEEDBACK positivo viene inviato dal DB fino alla GUI e viene visualizzato all’utente.

**“Eliminazione Albero”:** La GUI richiede alla BL la lista degli alberi; quest’ultima, tramite query, seleziona gli alberi idonei presenti all’interno del DB, il quale restituisce tale lista. La BL la restituisce a sua volta alla GUI. L’utente inserisce il dato in input tramite la GUI che, dopo aver verificato la correttezza del dato inserito (GUI-check), lo invia alla BL. Quest’ultima, tramite query, verifica l’esistenza dell’albero nel DB (DB-check), il quale provvede ad eliminarlo. Il FEEDBACK positivo viene inviato dal DB fino alla GUI e viene visualizzato all’utente.

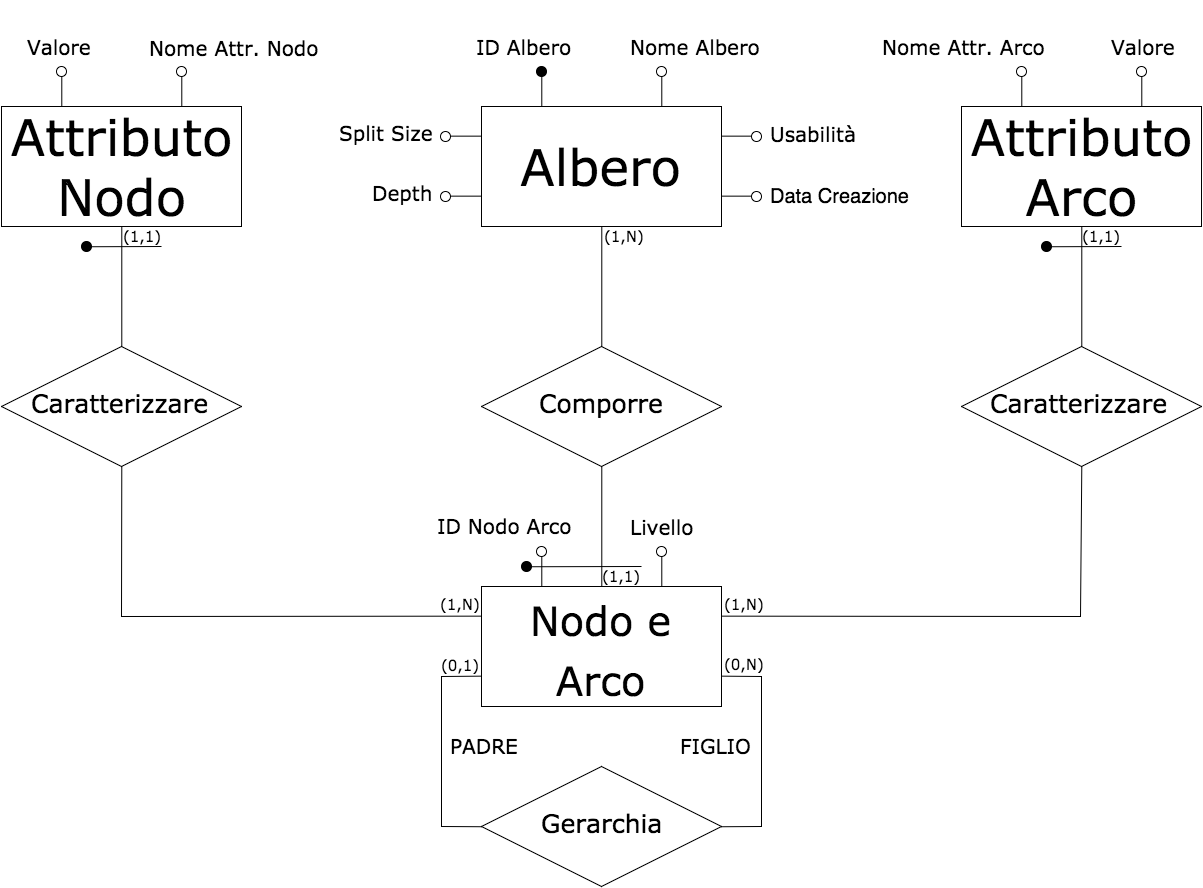
**“Calcolo Path”:** La GUI richiede alla BL la lista degli alberi; quest’ultima, tramite query, seleziona gli alberi idonei presenti all’interno del DB, il quale restituisce tale lista. La BL la restituisce a sua volta alla GUI. L’utente inserisce i dati in input tramite la GUI che, dopo aver verificato la correttezza dei dati inseriti (GUI-check), li invia alla BL. Quest’ultima, tramite query, verifica l’esistenza dell’albero nel DB (DB-check). La BL, in seguito, provvede ad effettuare il calcolo del path. Restituisce tale valore (insieme al tempo di esecuzione) alla GUI e viene visualizzato all’utente.

**Assunzione:** si è deciso di utilizzare esclusivamente chiamate SINCRONE. Quando dunque un oggetto invia un messaggio a un secondo, il mandante rimane IN ATTESA che l’oggetto ricevente ritorni senza poter proseguire in parallelo nella propria elaborazione. Ad esempio la GUI, una volta inseriti i dati dall’utente, blocca quest’ultimo impedendogli di usare altri servizi fino al totale completamento del processo. Ovviamente tale assunzione riguarda solamente un singolo utente.

D. ER Design

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

**FIGURA 5: SCHEMA E-R**



**Premessa:** si è deciso di non seguire lo schema presente all’interno del documento del progetto a favore di una migliore ottimizzazione del DB. Le entità “Attributo Nodo” e “Attributo Arco”, pur essendo uguali, non sono state unite per facilitare l’importazione da parte del DB di due file .CSV distinti.

La prima entità che risulta fondamentale da definire è “Albero”. Tramite essa è possibile salvare all’interno del DB gli attributi che ne descrivono l’anagrafica e la struttura: “ID Albero” (chiave primaria), “Nome Albero”, “Usabilità” (booleano che specifica la possibilità o meno di utilizzare tale albero da parte di un utente per eseguire servizi), “Split Size”, “Depth” e “Data Creazione”.

Tale entità risulta essere legata tramite la relazione “Comporre” all’entità “Nodo e Arco”, a sua volta caratterizzata da una relazione ricorsiva (“Gerarchia”) che ci permette di conoscere la relazione padre-figlio che caratterizza tutti i nodi di un generico albero.

L’entità “Nodo e Arco” ha i seguenti attributi che la caratterizzano: “ID Nodo Arco” e “Livello” (altezza di un nodo in un albero, assumendo che l’altezza della root sia zero). La chiave esterna è rappresentata da “ID Nodo Arco” e da “ID Albero”, ovvero l’albero di appartenenza. Risulta chiaro che ad un albero possono corrispondere da 1 a N nodi (1 è il numero minimo per poterlo definire tale), mentre un generico nodo ha sempre uno e un solo padre (fatta eccezione per la root) cosi come può avere da zero (nel caso di una foglia) a N figli.

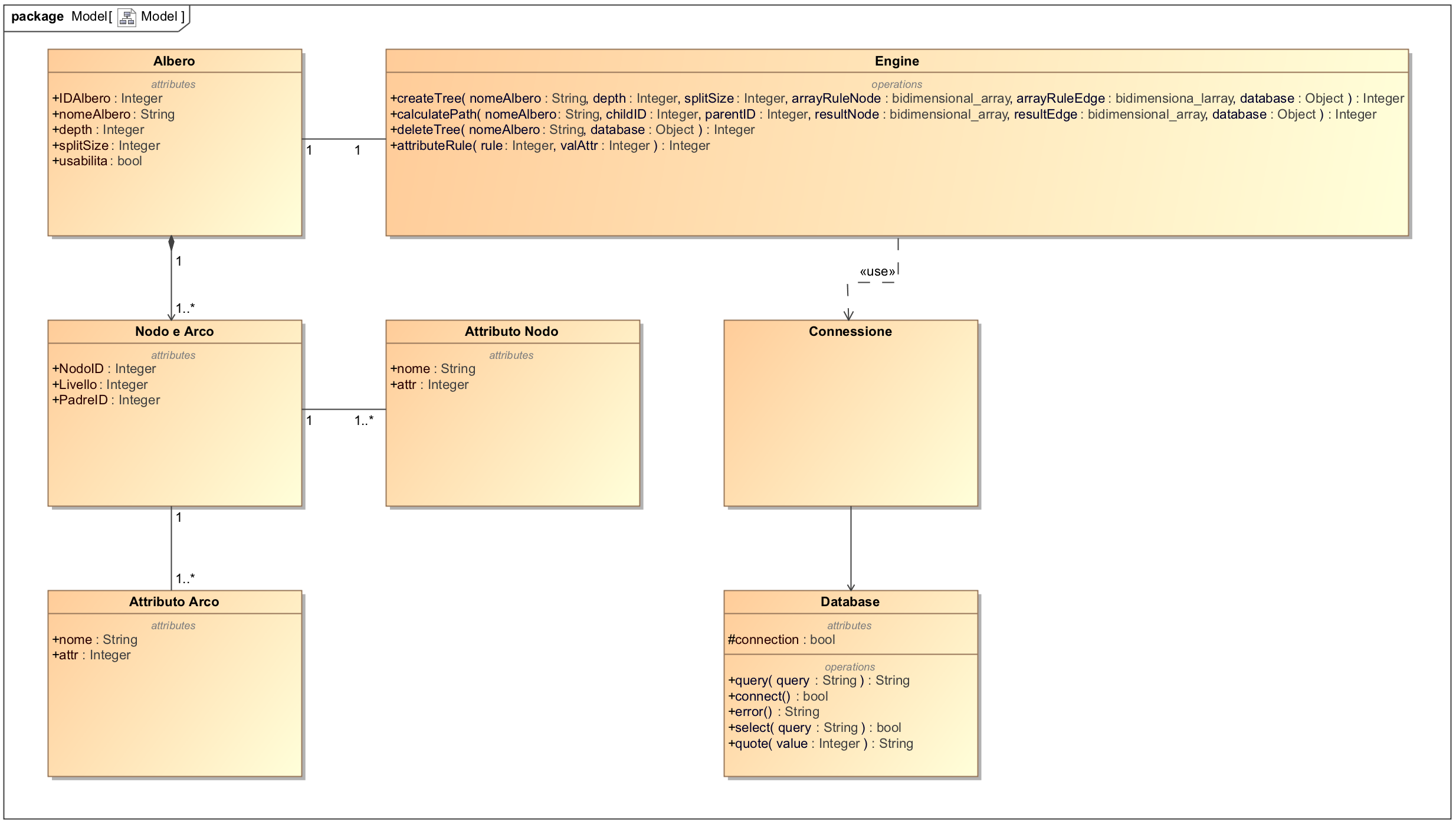
L’entità “Nodo e Arco”, oltre alla relazione con l’entità “Albero”, è caratterizzata anche da altre due relazioni, entrambe chiamate “Caratterizzare”, l’una con “Attributo Nodo” e l’altra con “Attributo Arco”.

Entrambe le entità “Attributo Nodo” e “Attributo Arco” sono costituite dai medesimi attributi: “Valore” (che specifica qual è il valore numerico di un determinato attributo di uno specifico nodo o arco), e rispettivamente “Nome Attr. Nodo” per l’entità “Attributo Nodo” e “Nome Attr. Arco” per l’entità “Attributo Arco”. Entrambe le entità sono caratterizzate dalle chiavi esterne con l’ID di “Nodo e Arco”.

È evidente che un generico nodo o arco può essere caratterizzato da 1 a N attributi cosi come un attributo può corrispondere ad uno ed un solo arco o nodo.

E. Class Diagram of the implemented System

**FIGURA 6: CLASS DIAGRAM**



Descriviamo, in seguito, i componenti del Class Diagram.

**Classe “Albero”**

Attributi**:**

* **IDAlbero**: codice intero positivo identificativo dell’albero, assegnato automaticamente seguendo un ordine crescente al momento della sua creazione.
* **nomeAlbero**: nome, di tipo stringa, assegnato all’albero da parte dell’utente in fase di creazione dello stesso.
* **depth:** numero intero positivo rappresentante l’altezza dell’albero.
* **splitSize:** numero intero positivo che rappresenta il numero di figli di ciascun nodo ad eccezione delle foglie.
* **usabilita:** booleano che indica la disponibilità dell’albero (ad esempio un utente non può effettuare operazioni di cancellazione o di calcolo del path di un albero in creazione poiché la sua usabilità è settata a “false”).

Relazioni con altre classi:

La classe “Albero” è caratterizzata da una relazione (1,N) con la classe “Nodo e Arco” poiché, banalmente, un albero può essere composto da un nodo (minimo) fino a N nodi. Tale classe presenta inoltre una relazione (1,1) con “Engine” dato che ogni servizio offerto dal PPC può essere effettuato su uno ed un solo albero.

**Classe “Nodo e Arco”**

Attributi:

* **IDNodoArco:** codice intero positivo identificativo del nodo/arco che viene assegnato automaticamente secondo un ordine crescente al momento della creazione dell’albero**.**
* **livello:** numero intero positivo che indica il livello (altezza) in cui si trova il nodo.

Relazioni con altre classi:

La classe “Nodo e Arco” ha una relazione (1,N) con “Attributo nodo” e “Attributo arco”, poiché per ogni nodo/arco esiste da un minimo di uno ad un massimo di N attributi per nodo/arco.

La classe “Nodo e Arco” ha inoltre una relazione (1,1) con “Albero”, poiché ciascun nodo e ciascun arco fanno parte di uno e un solo albero.

**Classe “Attributo Nodo”**

Attributi:

* **nomeAttributoNodo:** nome, di tipo stringa, assegnato dall’utente all’attributo al momento di creazione dell’albero.
* **valore:** numero intero positivo generato a seguito della scelta da parte dell’utente di una delle funzioni per la generazione di valore messe a disposizione dal sistema.

Relazioni con altre classi:

La classe “Attributo Nodo” ha una relazione (1,1) con “Nodo e Arco”, poiché ogni attributo fa parte di uno e un solo nodo.

**Classe “Attributo Arco”**

Attributi:

* **nomeAttributoArco:** nome, di tipo stringa, assegnato dall’utente all’attributo al momento di creazione dell’albero.
* **valore:** numero intero positivo generato a seguito della scelta da parte dell’utente di una delle funzioni per la generazione di valore messe a disposizione dal sistema.

Relazioni con altre classi:

La classe “Attributo Arco” ha una relazione (1-1) con “Nodo e Arco”, poiché ogni attributo fa parte di uno e un solo arco.

**Classe “Engine”**

Metodi:

* **createTree (nomeAlbero: String, depth: Integer, splitSize: Integer, arrayRuleNode: array[][], arrayRuleEdge: [][], database: Object):** funzione di creazione dell’albero che prende in input il suo nome, la sua altezza, il numero di figli e le coppie attributo-valore dei nodi e degli archi. La funzione ritorna uno dei seguenti valori:
* **3:** errore di connessione
* **2:** albero già esistente
* **1:** errore in fase di creazione dell’albero
* **0:** albero creato con successo
* **calculatePath (nomeAlbero: String, childID: Integer, parentID: Integer, resultNode: array[][], resultEdge: array[][], database: Object ):** funzione di calcolo del path che prende in input il nome dell’albero, il nodo di destinazione, il nodo di partenza e due variabili passategli per riferimento dove verranno restituite la somma dei nodi e degli archi come array bidimensionali. La funzione ritorna uno dei seguenti valori:
* **3:** Errore di connessione
* **2:** Albero Inesistente
* **1:** Nodo Inesistente
* **0:** Path Inesistente

La funzione inoltre modifica le variabili “resultNode” e “resultEdge” passate per riferimento alla funzione.

* **deleteTree (nomeAlbero: String, database: Object):** funzione di cancellazione dell’albero che prende in input il nome dell’albero. La funzione ritorna uno dei seguenti valori:
* **3:** Errore di connessione
* **2:** Albero Inesistente
* **1:** Eliminazione avvenuta con successo
* **0:** Errore in fase di eliminazione
* **attributeRule (rule: Integer, valAttr: Integer):** funzione che permette all’utente di scegliere tra le tre regole fornite di default per la generazione del valore degli attributi. La funzione ritorna un intero, ovvero il valore dell’attributo del nodo o arco.

Relazioni con altre classi:

La classe “Engine” utilizza la “Connessione” per accedere al DB.

**Classe “Database”**

Attributi:

* **connection:** variabile booleana per il controllo di connessione con il DB.

Metodi:

* **query (query: String):** funzione che, passando una query al database, ritorna “false” se la connessione al DB non ha avuto successo. In caso contrario, invece, ritorna il risultato della query.
* **select (query: String):** funzione di selezione che ritorna, sotto forma di array, il risultato della query in caso di successo mentre in caso di errore restituisce “false”.
* **error():** funzione che ritorna il messaggio di errore del database.
* **quote (value: Integer):** funzione che skippa i caratteri speciali per la query.

Relazioni con altre classi**:**

La “Connessione” è dipendente da “Database” poiché, senza una connessione avvenuta con successo con quest’ultimo, non è possibile accedere a nessun servizio offerto dal PPC.

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.>

1. La GUI non è composta da pagine differenti, ma sono tutte su una stessa pagina dinamica. L’utente si troverà di fronte ad un’unica pagina iniziale dove verranno visualizzate tre scelte: “Creazione albero”, “Eliminazione albero” e “Calcolo path”. A seconda della scelta, verrà visualizzata l’interfaccia corrispondente. Utilizzando una pagina dinamica, si riesce ad alleggerire il carico sul server in quanto in essa sono presenti tutte le features necessarie, riducendo così le richieste di pagina.
2. Il calcolo del path viene eseguito tramite una query sul database. Abbiamo ritenuto troppo complesso utilizzare un linguaggio di programmazione (es. Java) per accedere al DB e selezionare il path corrispondente per copiarlo in locale ed effettuarne la somma. Tale scelta è giustificata dalla semplicità con la quale si possono effettuare query riguardanti la somma del path direttamente in SQL.
3. Il DBMS che utilizzeremo è MySQL. Tale scelta è giustificata in modo approfondito nel punto “A.3 Content”.
4. La GUI effettuerà un check iniziale sui dati inseriti dall’utente. Tale check potrà però riguardare esclusivamente il valore dei relativi dati (es. caselle lasciate vuote, altezza o numero di figli negativo o decimale, ecc…). Tutto ciò su cui la GUI non può effettuare alcun controllo, verrà riservato al DB che si occuperà di verificare la presenza di eventuali errori (es. verificare che l’albero da voler eliminare non sia stato già eliminato, verificare l’esistenza dei vertici in un determinato albero durante il calcolo del path, verificare l’esistenza di un cammino tra i due vertici, ecc…).
5. La creazione di un albero viene effettuata dalla BL con l’utilizzo di file .CSV. Tale scelta è giustificata in modo approfondito nel punto “A.3 Content”.

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>

|  |  |
| --- | --- |
| **Requisiti** | **Design Decision** |
| A1.1 GUI Requirements | F.1, F.4 |
| A1.1.3 Calcolo Path | F.2 |
| A1.3 DB Requirements | F.3, F.4 |
| A1.2.1 Creazione Albero | F.5 |
| Tempo calcolo path (NF) | F.2 |
| Scelta del DBMS (NF) | F.3 |

Elenchiamo in seguito i restanti requisiti non funzionali e le relative design decision non elencate nella sezione F del deliverable:

1. “Il range (K-N) deve essere differente per ogni categoria di attributo. I valori possono essere sia interi che decimali.”. Tale requisito non funzionale è soddisfatto dalla GUI, la quale permette all’utente di inserire diverse categorie di attributo ognuna con il proprio valore.
2. “Gli alberi possono essere visualizzati e cancellati da qualsiasi utente, non solo da chi li ha inizialmente generati.”. Tale requisito non funzionale è soddisfatto innanzitutto dalla GUI che permette all’utente di selezionare qualsiasi albero all’interno del DB, e in secondo luogo dal DB stesso il quale non memorizza, all’interno di un generico albero, l’utente che lo ha creato.
3. “Gli alberi, una volta salvati nel DB, non possono essere modificati.”. Tale requisito non funzionale è soddisfatto dalla GUI, la quale non permette all’utente di modificare un albero precedentemente creato.

Non abbiamo assunto nessun’altra Design Decision per i restanti requisiti non funzionali non elencati.

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.*

***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***

Totale ore spese per il Deliverable v2: 85 ore di cui 10 spese per il learning e 75 per il doing.

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