**“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** | 23/12/2016 |
| **Deliverable** | 2 |
| **Team (Name)** | TV-LCD |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name & Surname** | **Matriculation Number** | **E-mail address** |
| Tiziano Santilli | *232412* | *tizianosantilli@gmail.com* |
| Vincenzo Apostolo | *236112* | *apos.vincenzo@gmail.com* |
| Luca Grillo | *236763* | [*lucag.8595@gmail.com*](mailto:lucag.8595@gmail.com) |
| Cristian Capozucco | *237945* | [*cristcapo@gmail.com*](mailto:cristcapo@gmail.com) |
| Davide Mariotti | *238185* | *davi95ma@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 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*

Table of Contents of this deliverable

Pagina 4: List of Challenging/Risky Requirements or Tasks

Pagina 5-6-7-8-9-10: Functional requirements, use case diagram e CockBurn’s template per la descrizione degli use case.

Pagina 11: Non functional requirements.

Pagina 12: Content e Assumptions.

Pagina 13-14: Prioritizzazione dei requisiti.

Pagina 15-16: Analysis model.

Pagina 17: Software Architecture: component diagram.

Pagina 18: Software Architecture: sequence diagram.

Pagina 20: Class diagram.

Pagina 21: Design decision.

Pagina 22: How FRs and NFRs are satisfied by design.

Pagina 23-24: Effort recording.

Pagina 25: Appendix code.

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** |
| Gestione concorrente di più richieste | 28/10/2016 |  |  |
| Mantenimento delle performance elevate | 28/10/2016 | 21/11/2016 | Abbiamo notato che la tecnologia NOSQL a grafi, mediante il DBMS Neo4j, la creazione del grafo, nonché il calcolo del cammino con la relativa somma dei valori degli attributi, è molto più veloce della tecnologia SQL (testata con MYSQL Workbench). Con quest’ultima tecnologia il grafo veniva creato in circa un’ora. Con la tecnologia NOSQL il grafo viene creato in pochi minuti |
| Tecnologie da usare | 28/10/2016 | 21/11/2016 | Effettuando vari test sulle performance tra le varie tecnologie |
| Organizzazione Logistica Gruppo | 28/10/2016 | 28/10/2016 | Abbiamo deciso di prendere appuntamenti fissi durante la settimana |
| Gestione Grafo | 28/10/2016 | 23/11/2016 | Il grafo verrà gestito interamente dal database |
| Connessioni Parte web/ logica/DB | 28/10/2016 | 12/12/2016 | Utilizziamo un Servlet (scritto in java) che opera all’interno del server Tomcat 9.0. Grazie a questo possiamo creare pagine web dinamiche a seconda dei parametri passati dall’utente al server. Questi parametri vengono quindi acquisiti e gestiti per creare l’albero, calcolare il percorso ed eliminare alberi. |

A. Requirements Collection

A.1 Functional Requirements

*A1.1 GUI Requirements*

* Il sistema deve prevedere una GUI.
* La GUI deve presentare una pagina web la quale permette di fare tutte le operazioni: Crea albero, Elimina albero e Calcolo Percorso.
* La sezione Crea Albero deve consentire l’immissione dei parametri per generare l’albero.
  + La GUI prevede un tasto “Crea Albero” il cui click deve generare l’albero.
* Nella sezione “Elimina Albero” deve essere inserito L’ID dell’albero che permette l’eliminazione dal DB dell’albero corrispondente.

* La Sezione Calcola Percorso deve permettere di inserire ID albero e i vertici di inizio e fine percorso. Questa sezione deve visualizzare il percorso corrispondente, la somma di ogni attributo e il tempo impiegato per calcolare la somma.

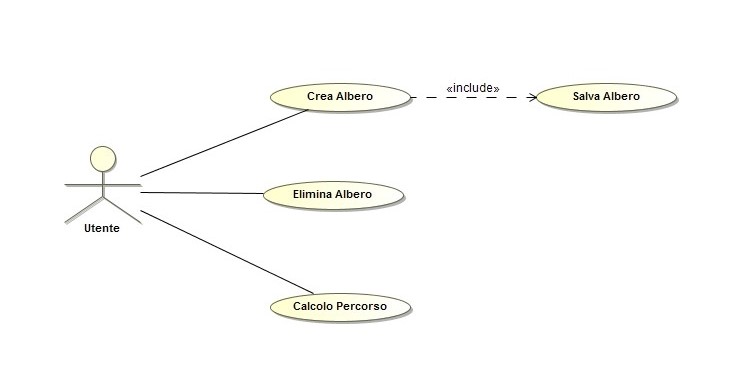
*A1.2 Business Logic e DB Requirements*

*(Nel nostro sistema Business logic e DB sono strettamente legati)*

* Il PPC (Planner path Calculator) deve generare l’albero corrispondete ai dati input e salvarlo sul DB.
* Il sistema deve garantire la creazione di un albero con al più 2 milioni di nodi aventi gli archi orientati.
* L‘albero generato deve avere i seguenti parametri:
* **Split Size**: Quanti vertici devo generare da un vertice dato
* **Depth**: Profondità dell’albero
* **AttributeList**: Lista di attributi da associare a vertici e archi
* *VertexAttributeList*: per esempio Ptime
* *EdgeAttributeList*: per esempio costo
* **AttributeValueGenerationRule:** servono per dire al sistema come generare i numeri. Possibile setup:
* **Random(K-N): to** Genera un valore random tra K e N
* Il PPC deve trovare il cammino ed eseguire l’operazione di somma di ciascun attributo su un percorso fornito in input dall’utente.

* Il PPC deve poter eliminare definitivamente un albero dal DB.

*USE CASE*



|  |  |  |
| --- | --- | --- |
| USE CASE 1 | Crea Albero | |
| Goal in Context | L’utente interagisce con il sistema e si aspetta la creazione di un albero | |
| Scope & Level | Primary Task | |
| Preconditions | L’utente si trova nella interfaccia grafica e conosce i dati da immettere | |
| Success End Condition | L’albero viene creato senza imprevisti | |
| Failed End Condition | L’albero non viene creato | |
| Primary | Utente | |
| Trigger | L’utente seleziona l’opzione “Crea albero” nella home page | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente seleziona l’opzione “Crea albero” nella home page |
|  | 2 | L’utente immette i dati |
|  | 3 | L’utente preme il pulsante “Crea albero” |
|  | 4 | L’albero viene creato |
|  | 5 | L’utente viene avvisato della corretta creazione dell’albero |
| EXTENSIONS | Step  Modifichiamo? | Branching Action |
|  | 4a | L’albero non viene creato.  Lancia messaggio di errore |

|  |  |  |
| --- | --- | --- |
| USE CASE 2 | Salva albero | |
| Goal in Context | L’albero viene salvato in una struttura dati | |
| Scope & Level | Primary Task | |
| Preconditions | L’utente ha creato con successo un albero | |
| Success End Condition | L’albero viene correttamente salvato | |
| Failed End Condition | L’albero non viene correttamente salvato | |
| Primary | Utente | |
| Trigger | L’albero è stato creato | |
| DESCRIPTION | Step | Action |
|  | 1 | L’albero è stato creato |
|  | 2 | L’albero viene salvato |
|  | 3 | Ritorna un messaggio di conferma |
| EXTENSIONS | Step | Branching Action |
|  | 1a | L’albero non è stato creato:  Lancia un messaggio d’errore |
|  | 2a | L’albero non viene salvato:  Lancia un messaggio d’errore |

|  |  |  |
| --- | --- | --- |
| USE CASE 3 | Elimina albero | |
| Goal in Context | Il sistema elimina l’albero | |
| Scope & Level | Subfunction | |
| Preconditions | L’utente si trova nell’interfaccia grafica e conosce l’id dell’albero da eliminare | |
| Success End Condition | L’albero è stato eliminato con successo | |
| Failed End Condition | L’albero non è stato eliminato | |
| Primary | Utente | |
| Trigger | L’utente preme sul pulsante “Elimina albero” | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente preme sul pulsante “Elimina albero” |
|  | 2 | L’albero viene eliminato |
| EXTENSIONS | Step | Branching Action |
|  | 2.a | L’albero non viene eliminato:  lancia messaggio di errore |

|  |  |  |
| --- | --- | --- |
| USE CASE 4 | Calcolo Percorso | |
| Goal in Context | Il Sistema calcola il percorso scelto dall’utente | |
| Scope & Level | Summary | |
| Preconditions | Gli alberi sono stati salvati e l’utente si trova nell’interfaccia grafica e conosce i dati da immettere | |
| Success End Condition | Restituzione del risultato | |
| Failed End Condition | Il Calcolo Perscorso non viene eseguito | |
| Primary | Utente | |
| Trigger | L’utente immette i parametri per il calcolo del percorso | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente immette i parametri |
|  | 2 | L’utente fa partire il calcolo del percorso |
|  | 3 | Il sistema esegue le operazioni |
|  | 4 | Il sistema restituisce il risultato |
| EXTENSIONS | Step | Branching Action |
|  | 1.a | In mancanza di paramentri: Genera Errore |

A.2 Non Functional Requirements

* La GUI deve essere implementata con interfaccia web che utilizza la tecnologia HTML5.
* Il DBMS scelto deve essere open-source.
* La sicurezza non è necessaria: il sistema verrà inserito in un ambiente web sicuro
* Se l’albero da eliminare è in uso, l’eliminazione deve essere messa in attesa fino al completamento di tutte le operazioni su quell’albero.
* Il sistema deve garantire un uso concorrenziale di 10-100 utenti.

***Usability:*** La GUI deve essere il più possibile User-Friendly. L’interfaccia sarà divisa in tre differenti sezioni, ognuna di queste attribuita alle singole operazioni. Esagerata attenzione all’estetica non è richiesta poiché il sistema sarà usato solamente all’interno dell’azienda.

***Performance:*** La creazione e il salvataggio del grafo sono operazioni poco frequenti, per questo sono accettabili tempi di circa un’ora.  
L’operazione più eseguita è il calcolo del percorso che richiede tempi pari a 30 secondi per alberi da 1 milione di nodi e 60 secondi per alberi da 2 milioni di nodi.

A.3 Content

Per semplicità dividiamo il nostro progetto in tre parti:

* Parte web: useremo HTML5, Tomcat (una distribuzione Apache gratuita contenente Catalina per il servlet) e come Ide per la gestione della parte web useremo Eclipse.
* Parte logica: useremo Eclipse come Ide per il linguaggio java.
* Parte DB: useremo (alto fattore di rischio) il database NOSQL neo4j basato su grafi.

A.4 Assumptions

* L’hardware del server è abbastanza potente da sostenere l’esecuzione concorrente di 100 utenti.
* La creazione del grafo può impiegare tempi che vanno da pochi minuti ad al massimo un paio d’ore.
* Decidiamo di creare un’interfaccia adatta all’azienda dal punto di vista grafico e della facilità d’uso.
* Utilizziamo il linguaggio java per la sua portabilità, per le strutture dati incluse nelle librerie, per la facilità di connesione con il DBMS e per la conoscenza acquisita durante gli studi.
* Utilizzo del DBMS NoSQL “Neo4J” a grafi. Le prestazioni, all’aumentare del volume dei dati, sono migliori rispetto a un DBMS SQL.

***A.5 Prioritizatione****Requisiti listati secondo ordine di priorità.*

1)**PARAMETRI GRAFO:**

**Split Size**: Quanti vertici devo generare da un vertice dato

**Depth**: Profondità dell’albero

**AttributeList**: : lista di attributi da associare a vertici e archi

* + **VertexAttributeList**: per esempio Ptime
  + **EdgeAttributeList:** per esempio costo

**AttributeValueGenerationRule:** servono per dire al sistema come generare i numeri. Possibile setup:

* + - **Random(K-N): to** Genera un valore random tra K e N

La GUI può essere usata per generare un albero alla pressione del tasto “Crea Albero”.

Il risultato è salvato nel database.

2) **SCELTA DB:**

Il team dovrà selezionare la tecnologia di database da utilizzare.

Nel caso in cui il team scelga un database diverso da MYSQL dovrà adattare lo schema del DB fornito alla tecnologia scelta. Il team dovrà motivare la scelta della tecnologia del database come parte delle loro decisioni di progettazione.

Possono essere scelti DBMS non MYSQL a condizione che siano open-source.

3)**OPERAZIONI SU GRAFO**:

L’interfaccia deve essere usata per reperire un grafo salvato sul databese: gli utenti possono selezionare 2 vertici A e B, e il Sistema deve ritornare la lista dei vertici da “A” a “B” insieme alla somma di ogni attributo.

4)**DIMENSIONE MASSIMA GRAFO:**

Il Sistema deve essere in grado di gestire 2 milioni di vertici

5)**NUMERO UTENTI CHE USANO IL SISTEMA**

Ci aspettiamo 10-100 utenti che usino il sistema concorrentemente. Il sistema deve essere in grado di funzionare senza nessun impatto sulle prestazioni

6)**PERFORMANCE**

La GUI deve anche mostrare il tempo di esecuzione del calcolo.

Dato che questo sistema sarà interrogato da un sistema di produzione,

dovrebbe essere "veloce" per restituire il risultato.

7)**INTERFACCIA**

La GUI deve essere implementata come un’interfaccia web basata su tecnologia HTML5

8)**SICUREZZA**

Sicurezza: non abbiamo bisogno di requisiti sulla sicurezza in quanto il Sistema sarà integrato in un Sistema web sicuro.

B. Analysis Model

**Boundary Object:**

**Interfaccia Home:**

Dall’interfaccia home l’utente può, tramite le sotto sezioni della pagina, fare direttamente tutte le operazioni che il PPC è in grado di svolgere.

**Form Crea Albero:**

Tramite una form apposita l’utente può inserire i parametri per la creazione di un albero.

**Form Elimina Albero:**

Tramite una form apposita l’utente può inserire l’ID dell’albero da eliminare.

**Form Calcola Percorso:**  
Tramite una form apposita l’utente può inserire i parametri dell’albero sul quale calcolare il percorso.

**Controller Object:**

**Crea Albero:**

Attività del sistema che procede alla creazione di un albero. Una volta creato correttamente, ci sarà l’azione di salvataggio.

**Elimina Albero:**

Attività del sistema che esegue l’eliminazione di un albero creato in precedenza.

**Individua Albero: (ritrova)**Individua l’albero presente nel database sul quale fare operazioni.

**Calcola Percorso:**Individua il percorso, calcola la somma degli attributi e il tempo impiegato per fare la somma.

**Entity Object:**

**Albero:**

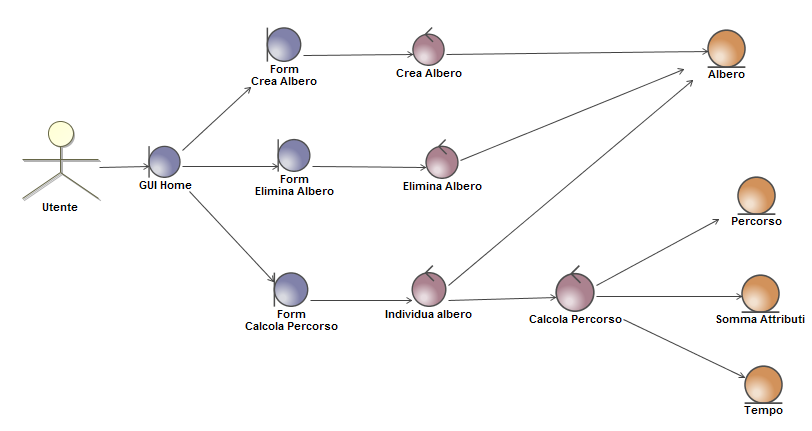
Rappresenta l’insieme di tutti gli grafi salvati sul DB. Gli alberi sui quali andremo a lavorare sono l’oggetto principale del nostro sistema.

**Percorso:**

Lista dei nodi che fanno parte del percorso richiesto dell’utente.

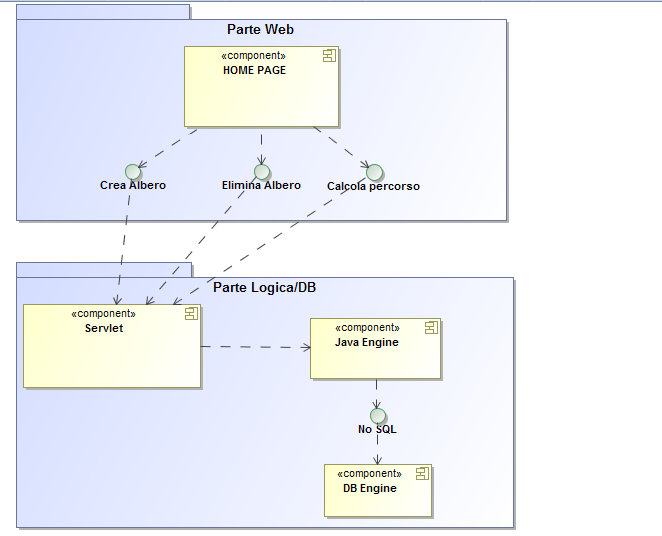
**Somma Attributi:**  
Lista delle somme di ogni attributo presente sui nodi e sugli archi.

**Tempo:**Rappresenta il tempo che il sistema ha impiegato per fare il calcolo della somma degli attributi.



C. Software Architecture

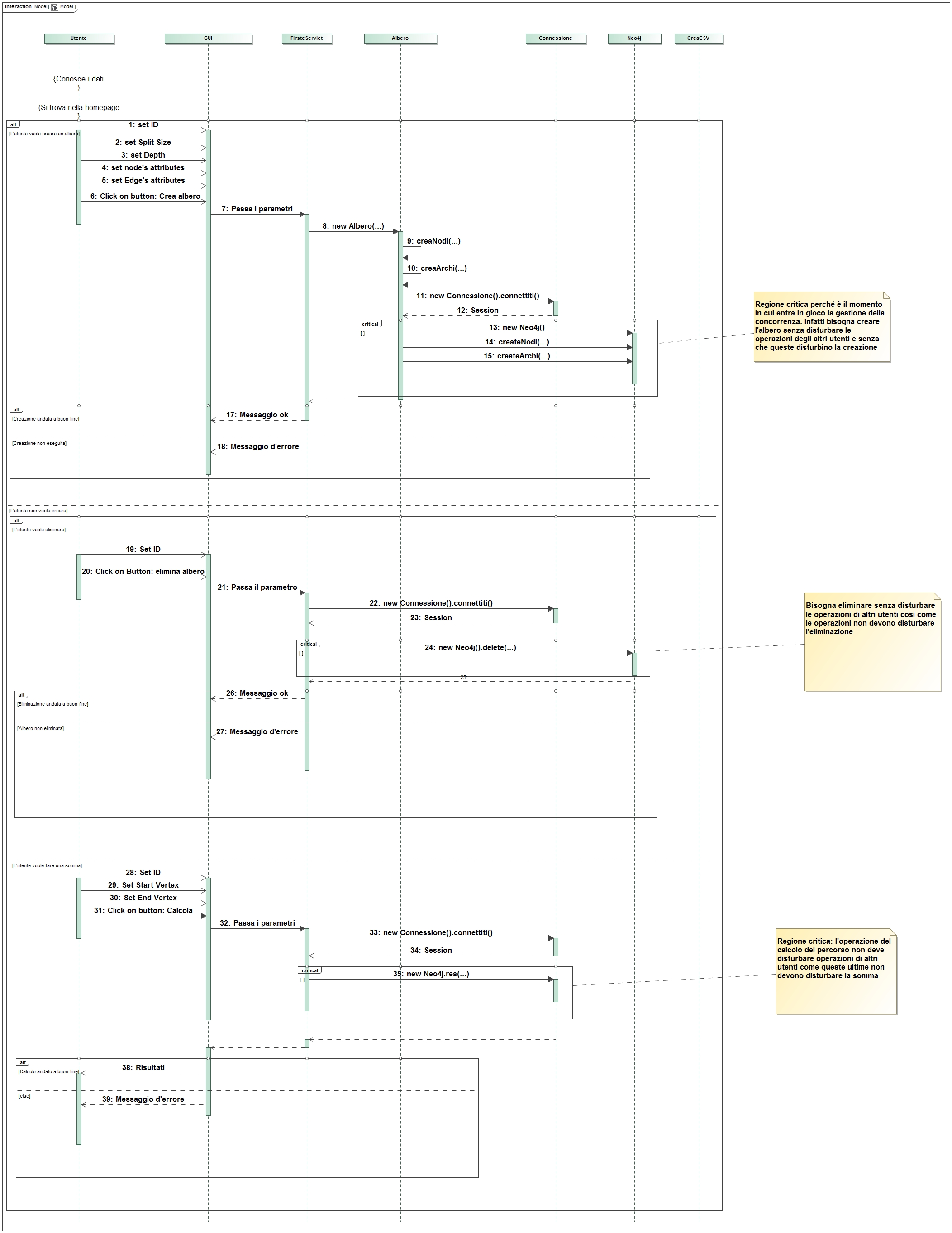
C.1The static view of the system: Component Diagram



Abbiamo deciso di unire la parte Logica alla parte DB in un unico package poiché nel nostro sistema sono strettamente legate.

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

*Nella cartella è presente l’immagine per una lettura migliore.*

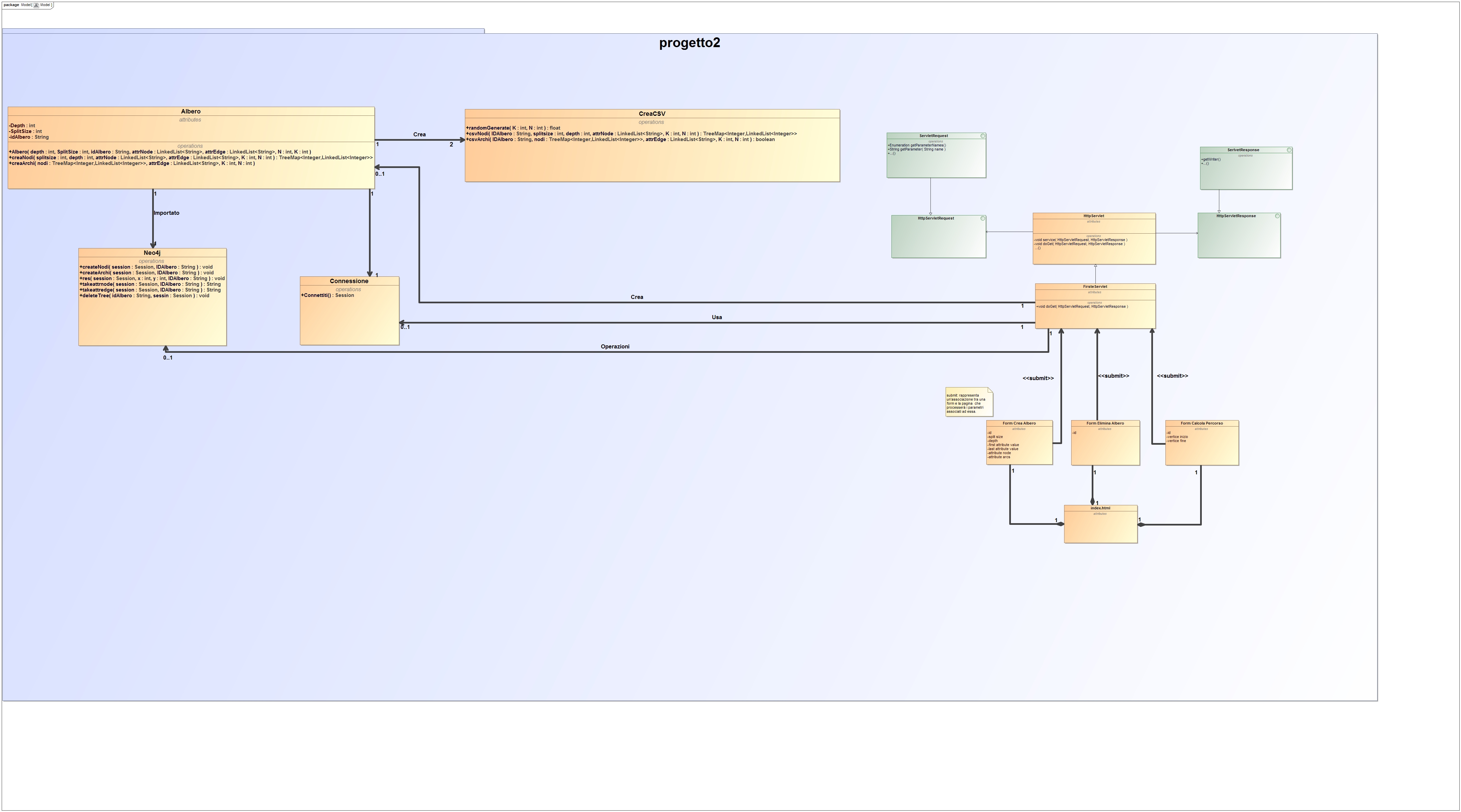


D. ER Design

Scegliendo un DB NOSQL non è necessario fare un Digramma ER.

E. Class Diagram of the implemented System

*Nella cartella è presente l’immagine per una lettura migliore.*



F. Design Decisions

**Scelta DB**: Abbiamo notato che mediante la tecnologia NOSQL a grafi (DBMS Neo4j), il calcolo del cammino con la relativa somma dei valori degli attributi, nonché la creazione del grafo, è molto più veloce della tecnologia SQL (testata con MySQL workbench). Con tecnologia NOSQL il calcolo del percorso viene effettuato rispettando i requisiti di tempo stabiliti.

**Scelta Linguaggio**: Abbiamo deciso di utilizzare il linguaggio java per la sua portabilità, per le strutture dati incluse nelle librerie, per la facilità di connessione con il DBMS e per la conoscenza acquisita durante gli studi.

**Scelta “Dove eseguire il calcolo del cammino”**: abbiamo deciso che il calcolo verrà effettuato sul database. Questo perché non si ha possibilità di errore di effettuare scritture concorrenti ma solo letture. Inoltre si presume che l’hardware del server sia abbastanza potente da sostenere l’esecuzione concorrenti di più utenti.

**Scelta collegamento parte Web-Logica-DB:** abbiamo deciso che il collegamento tra le varie componenti del progetto sarà fatto da varie strutture in codice java. Per il collegamento dalla parte web alla parte Logica abbiamo deciso di utilizzare un servlet, ovvero una struttura java che può connettere facilmente delle parti in html a delle parti in java. Per il collegamento tra la parte di logica e il database non è stato necessario altro che configurare i driver java di neo4j.

**Scelta interfaccia user-friendly:** abbiamo deciso che per rendere l’interfaccia user-friendly la scelta migliore è stata quella di mettere in una pagina web tutte e tre le opzioni (creazione, eliminazione e calcolo cammino) cosi che l’utente finale abbia direttamente a portata di mano tutte le scelte disponibili.

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

**GUI**:

La GUI che andremo a realizzare è formata da un’unica pagina web dove l’utente potrà selezionare le tre operazioni (crea albero, elimina albero, calcolo percorso), attraverso tre pulsanti.

Si aprirà una form sotto ogni pulsante, quando quest’ultimo sarà premuta. L’utente potrà cosi immettere tutti i parametri necessari per le diverse operazioni, e proseguire premendo il pulsante corrispondente.

Per l’operazione “calcola”, i vari risultati verranno mostrati a video in una nuova pagina web.

La GUI sarà inoltre formata da pop-up per ogni tipo di notifica (in caso di successo o errore di creazione e eliminazione).

Sarà “user-friendly e garantirà dunque la USABILITY, definita nei non functional requirements.

La GUI verrà implementata con l’HTML5(come da requisito).

Qualsiasi tipo di operazione che l’utente scelga, la GUI comunicherà solo ed esclusivamente con la parte Business-Logic (Java) ed il database sarà completamente trasparente ad essa.

Non ci sarà una form di Login iniziale, poiché la sicurezza non è necessaria dato che l’applicazione girerà in un ambiente web sicuro all’interno dell’azienda.

**Business logic e DB:**

Il database che andremo a utilizzare è neo4j. Tale database crea e salva l’albero corrispondente ai dati dell’input, elimina alberi contrassegnati da id e svolge su di essi le operazioni definite dalle specifiche.

Garantisce inoltre la creazione con alberi di due milioni di nodi e archi orientati.

Il database scelto è completamente open-source e prende i suoi parametri di creazione, eliminazione e calcolo percorso da un applicativo java che si interfaccia alla GUI mediante un servlet.

Mediante questa unione di business logic e db i requisiti di performance vengono ampiamente soddisfatti.

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.***

***Deliverable version 2:***

* *Total hours spent by the group: 50h.*
* *Time spent for Learning: 9h.*
* *Time spent for Doing: 41h.*

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

**Test sul codice:**

La prima versione del prototipo ci ha permesso di capire se l’implementazione del nostro progetto soddisfa i requisiti più importanti.   
Il calcolo del percorso avviene nei tempi richiesti ed è molto efficiente. La visualizzazione dei risultati in questa versione del prototipo avviene solo su console.   
La creazione dell’albero avviene in tempi ragionevoli ovvero (5/10 minuti), rispetto ad un tempo pari a circa un’ora richiesto dal cliente. Essendo un prototipo versione 1 non è ancora possibile inserire range dei valori randomici diversi per ogni attributo, ma un unico range per tutti gli attributi. La versione completa sarà in grado di soddisfare questo requisito.

La funzionalità di eliminazione dell’albero è stata implementata ma non è ancora in grado di attendere che tutti gli utenti abbiano finito di usare l’albero da eliminare.

I test sull’uso concorrenziale da parte di 10-100 utenti sono risultati i più critici. Inizialmente il prototipo non era in grado di garantire la concorrenza per problemi legati alla comunicazione tra la parte Business-Logic e il Database.. Dopo vari raffinamenti la concorrenza è possibile. Purtroppo la mancanza di un hardware adeguato non ci permette di testare il nostro sistema su più di 6-7 utenti.