**“Software Engineering”**

**Course**

**a.a. 2019-2020**

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

**<5, Portami a Destinazione>**

|  |  |
| --- | --- |
| **Date** | <05/12/2019> |
| **Deliverable** | Documento Iniziale – D1 |
| **Team (Name)** | NewBie Team |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name & Surname** | **Matriculation Number** | **E-mail address** |
| Luca Di Laudo | *248444* | *luca.dilaudo@student.univaq.it* |
| Mattia Masci | *244146* | *mattia.masci@student.univaq.it* |
| Giovanni Gemini | *241972* | *giovanni.gemini@student.univaq.it* |

Table of Contents of this deliverable

Sommario:

List of Challenging/Risky Requirements or tasks………………………………3

A Stato dell’Arte…………….…………………………………….……………...….................4

B Raffinamento dei Requisiti………………………………….……………...................6

B.1 Requisiti Funzionali.................................................................................6

B.2 Requisiti non Funzionali........................................................................8

B.3 Scenari d’uso dettagliati.........................................................................9

B.4 Excluded Requirements.........................................................................11

B.5 Assunzioni..................................................................................................11

B.6 Use Case Diagram....................................................................................12

C Architettura Software..........................................................................................13

C.1 Component Diagram...............................................................................13

C.2 Sequence Diagram...................................................................................15

D Dati e la loro Modellazione................................................................................18

D.1 Modello E-R...............................................................................................18

D.2 Tabelle.........................................................................................................18

E Design Decisions...................................................................................................21

F Design di Basso Livello.......................................................................................22

F.1 Class Diagram...........................................................................................22

G FR’s & NFR’s...........................................................................................................23

H Effort Recording...................................................................................................25

H.1 Pert Diagram..................................................................................................25

H.2 Logging……………………………………………………………………………….….25

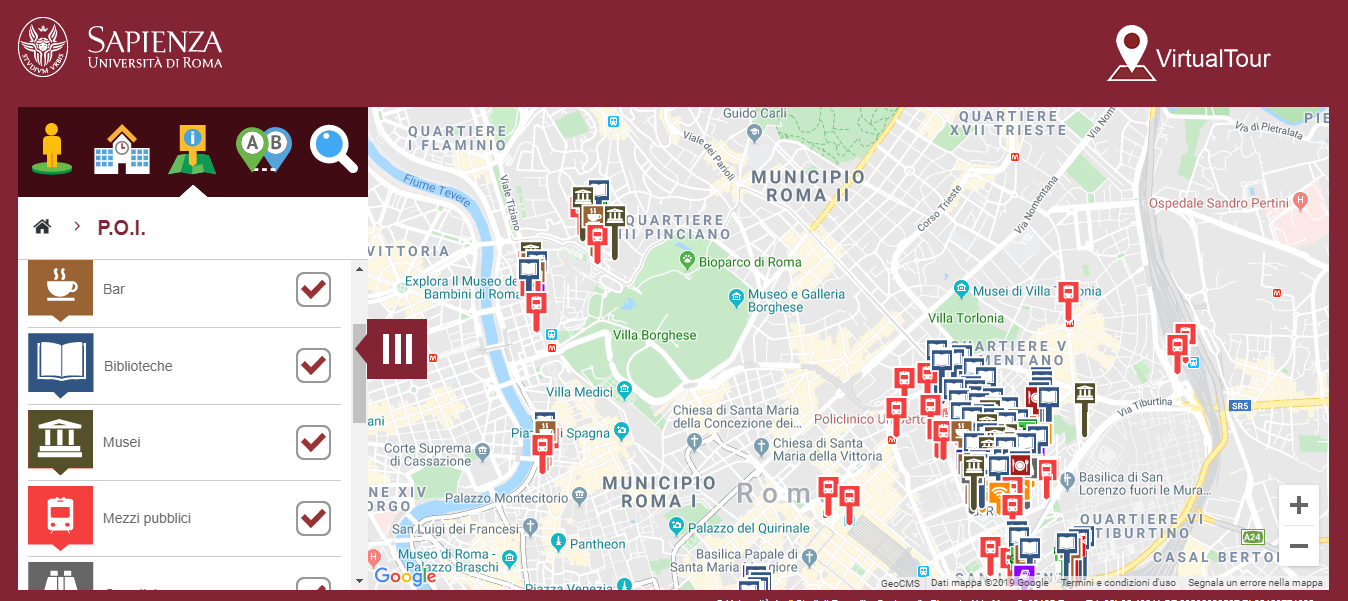
## List of Challenging/Risky Requirements or Tasks

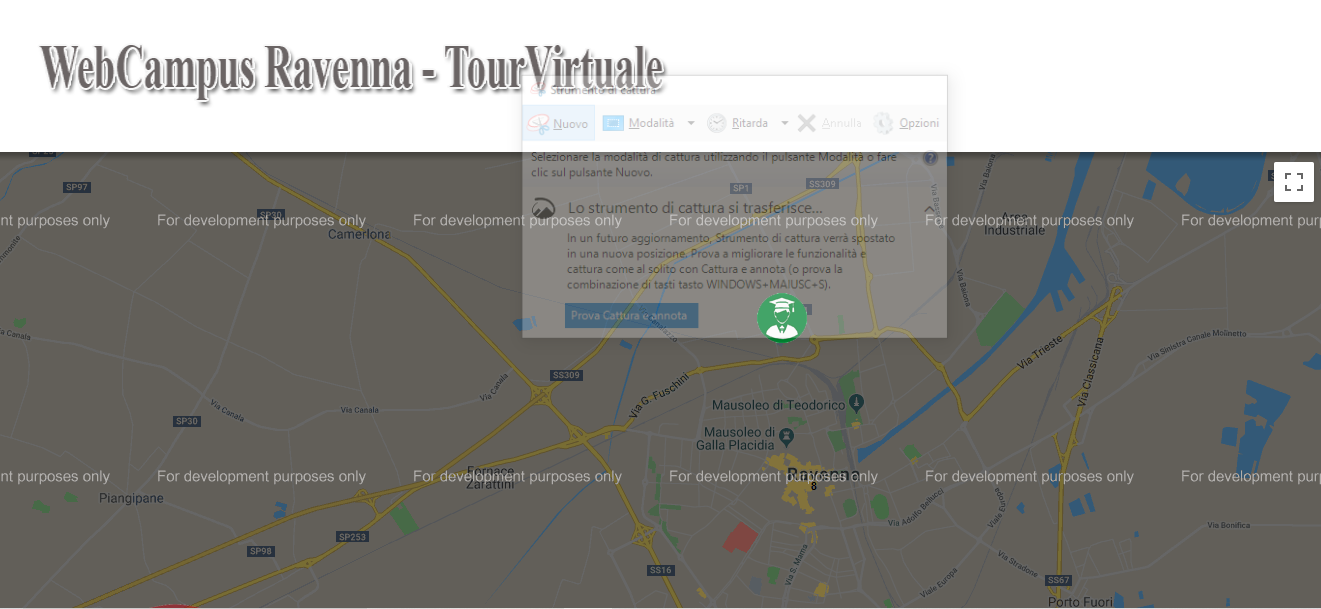
|  |  |  |  |
| --- | --- | --- | --- |
| **Challenging Task** | **Date the task is identified** | **Date the challenge is resolved** | **Explanation on how the challenge has been managed** |
| Come visualizzare una mappa all'interno del software | 20/11/19 | 20/11/19 | Dopo esserci documentati sulle tecniche più usate abbiamo deciso che importeremo una mappa dall'esterno. Controllando in giro abbiamo notato che la maggior parte delle aziende utilizza Google Maps per incorporare mappe nel proprio sito. Visto che da non molto sopra una certa soglia di visualizzazioni Google fà pagare il servizio abbiamo trovato un'alternativa: Google My Maps ci permetterà di incorporare e utilizzare una mappa in modo attinente ai nostri scopi e il servizio sarà gratuito inserendo un iframe che ci fornirà proprio Google. |
| Usare MySQL o NoSQL | 27/11/19 | 27/11/19 | Abbiamo deciso di utilizzare il database MySQL visto il precedente utilizzo in altri corsi. Inoltre dopo un'analisi veloce abbiamo notato che I dati da manipolare hanno una struttura definita per cui MySQL dovrebbe permetterci di lavorare con efficienza. In caso ci accorgeremo che non sia così ci impegneremo a valutare. |
| Come progettare il prototipo | 27/11/19 | 02/12/19 | Visto che il sistema è pensato per essere un servizio aggiunto al sito di un'università abbiamo deciso di implementare il progetto attraverso una pagina web. |

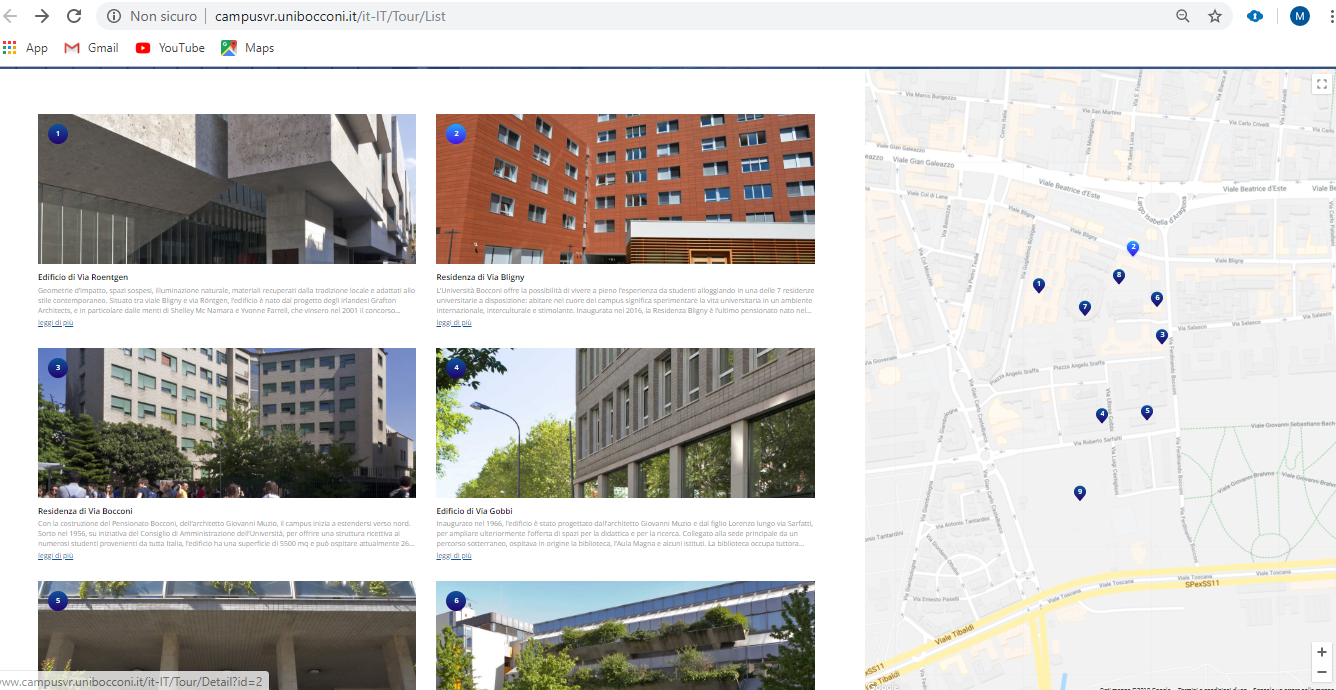
A. Stato dell’Arte

*Consultando vari siti universitari italiani abbiamo notato che molti di questi sono sprovvisti di una mappa virtuale, per quanto riguarda i pochi che ne sono dotati viene offerto solo un servizio base.*

*Sotto questo punto di vista il sito più fornito, tra quelli che abbiamo visionato, è quella della "Sapienza Università di Roma". La pagina è molto intuitiva e mette a disposizione una mappa con molte funzionalità, tra l'altro tante delle quali richieste nella specifica del nostro progetto, per questo nel realizzarlo ci siamo ispirati ad essa.*

**





B. Raffinamento dei Requisiti

Nel contesto dell’innovazione tecnologica che sta travolgendo tutto il mondo ormai da anni è stata riscontrata la necessità di realizzare un software, in ambito universitario, in grado di agevolare gli studenti, i docenti e tutto il personale di raggiungere i vari punti di interesse all'interno degli atenei. Si potrebbe voler sapere la posizione di edifici specifici per i propri studi, per svolgere attività amministrative, o ancora degli uffici dei professori, delle mense...

Il fine ultimo di questo software quindi è quello di condurre una persona in un determinato luogo con lo scopo di farlo nel modo più semplice e affidabile possibile.

Poiché l'applicativo dovrà essere utilizzato in un contesto prettamente universitario, e quindi da una vasta tipologia di persone che rivestono diversi ruoli, da uno studente immatricolato ad uno non immatricolato, a un nuovo professore o ancora persone invitate ad una discussione di laurea, l'interfaccia utente dovrà essere semplice ed intuitiva così da ottimizzare al meglio la User-Experience cercando di evitare il più possibile eventuali bug o errori nel sistema.

B.1 Servizi (con prioritizzazione)

Abbiamo prioritizzato i requisiti funzionali trovati in priorità bassa, media, alta, secondo lo schema a pag.35 delle slides riguardanti la “Requirement Engineering”.

Comandi Vocali (MEDIA)

L’utente avrà la possibilità di utilizzare i comandi vocali per interagire con il sistema, come ad esempio la ricerca di uno specifico POI o Edificio;

Registrazione (BASSA)

Un utente avrà la possibilità di registrarsi nel portale specificando Username, Password, e facoltativamente CdL, dipartimento, e-mail e università.

Login (BASSA)

L’utente avrà due possibilità di effettuare il Login:

* in una sezione dedicata utilizzando Username e Password (in base a questo il sistema poi specializzerà la visualizzazione della mappa in base al CdL frequentato);
* senza immettere Username e Password (con vista generale riguardo tutto il complesso Universitario).

Per quanto riguarda l'admin gli sarà possibile un accesso separato in cui immettere il proprio codice univoco.

Modifica dati personali (BASSA)

Sarà permesso all’Utente di poter cambiare le proprie informazioni personali in caso di particolari situazioni, come ad esempio il passaggio da una facoltà ad un’altra.

Visualizzazione mappa (ALTA)

Il sistema sarà in grado di visualizzare una mappa che segnalerà tutti i POI sul territorio in cui sono distribuiti. I POI saranno rappresentati con icone diverse a seconda della tipologia di questi.

Filtraggio POI (ALTA)

Il sistema sarà in grado di filtrare e rappresentare sulla mappa in base alle esigenze dell'utente, selezionando/deselezionando servizi o utilizzando una barra di ricerca, i vari POI desiderati. Inoltre, non appena un utente entrerà nella propria pagina visualizzerà una mappa all'interno di cui saranno segnalati automaticamente i POI collegati al proprio CdL, sarà poi premura dell'utilizzatore lasciare inalterate o cambiare le impostazioni divisualizzazione.

*Gestione back-end del sistema* (MEDIA)

L'admin, dopo aver effettuato il login tramite una UI specifica, potrà aggiungere/rimuovere POI, che siano edifici, uffici o fornitori di servizi. L’admin sarà in grado quindi di gestire in back-end tutto il sistema apportando le relative modifiche per aggiungere/rimuovere gli oggetti sopra descritti.

B.2 Requisiti non Funzionali

Usability

Il Sistema dovrà essere strutturato in modo basico e di facile comprensione per garantire una user-experience funzionale ma allo stesso tempo semplice. La GUI dovrà essere il più funzionale e user-friendly possibile.

Scalability

Il sistema dovrà essere in grado di gestire un gran numero di utenti in contemporanea, ancor di più nei periodi di iscrizione in cui l'affluenza nelle sedi si intensifica. Il DBMS, che sarà una delle parti del software messa più a dura prova in questi periodi, dovrà essere messo in condizione di eseguire un gran numero di transazioni all'occorrenza.

Capacity

Il software dovrà essere in grado di poter funzionare in una grande Università potenzialmente distribuita su più sedi, e con una molteplicità importante di POI*.*

Supportability

Gli utenti useranno il software attraverso i sistemi più disparati (dagli smartphone ai pc, con la conseguente varietà di piattaforme), per questo dovrà essere progettato tenendo conto di questa diversità.

Maintainability

Il software dovrà rispettare la Manutenibilità, ovvero la capacità di essere Modulare. Così sarà più facile individuare una parte di codice interessata ad eventuali modifiche o correzioni. Per fare ciò è necessario che il codice sia ben commentato e documentato;

B.3 Scenari d’uso dettagliati

*Raggiungimento di un POI (Utente con Account)*

1. L’utente apre la pagina web;
2. Una volta aperta la pagina, l’utente effettua il login con password;
3. Il software mostra una vista della mappa specializzando la vista in base al CdL inserito al momento della registrazione, ovvero mostrando tutti i POI vicini all’edificio di appartenenza del corso di laurea;
4. L'utente seleziona il POI desiderato;
5. Il software restituisce l’edificio o il luogo dov’è situato il POI scelto;

Raggiungimento di un POI (Utente senza account)

1. L’utente apre la pagina;
2. Una volta aperta la pagina, l’utente effettua il login come ospite;
3. Il software mostra una vista della mappa generalizzata, quindi non prioritizzando per mezzo del CdL come nel caso dell’utente registrato;
4. L’utente potrà scegliere quale POI raggiungere;
5. Appena l’utente sceglierà la destinazione da raggiungere il software elaborerà la destinazione e mostrerà dove si trova;

*Registrazione Utente*

1. L’utente apre la pagina;
2. Una volta aperta la pagina l’utente si reca nella sezione della registrazione;
3. Aperta la pagina di registrazione inserisce i dati richiesti dal software come E-mail, CdL, anno di immatricolazione, scelta della password ecc...;
4. Appena finito di immettere tutti i dati richiesti il sistema invia una mail di conferma all’indirizzo che l’utente ha inserito al momento della registrazione;
5. L’utente dovrà recarsi sulla propria casella di posta per cliccare sul link si avvenuta conferma e verrà rimandato alla pagina iniziale del software come utente registrato a tutti gli effetti;

*Modifica dati personali*

1. L’utente apre la pagina;
2. Una volta aperta la pagina, l’utente dovrà necessariamente effettuare il login;
3. Appena effettuato il login, l’utente si deve recare nella sezione del proprio profilo e cliccare sul tasto di modifica del profilo;
4. Una volta modificati i propri dati personali apparirà una finestra con la richiesta di conferma delle modifiche apportate;
5. Una volta confermato il tutto, il software registrerà nel database i dati modificati dall’utente, quindi le modifiche verranno apportate ufficialmente;

B.4 Excluded Requirements

-Non è previsto che l'utente possa mandare segnalazioni (come mandare messaggi, dare feedback o segnalare nuovi POI) al sistema per non aggiungere complessità al progetto.

B.5 Assunzioni

-I POI saranno divisi in edifici, fornitori di servizi e uffici.

-Ogni ufficio sarà collocato all'interno di un edificio.

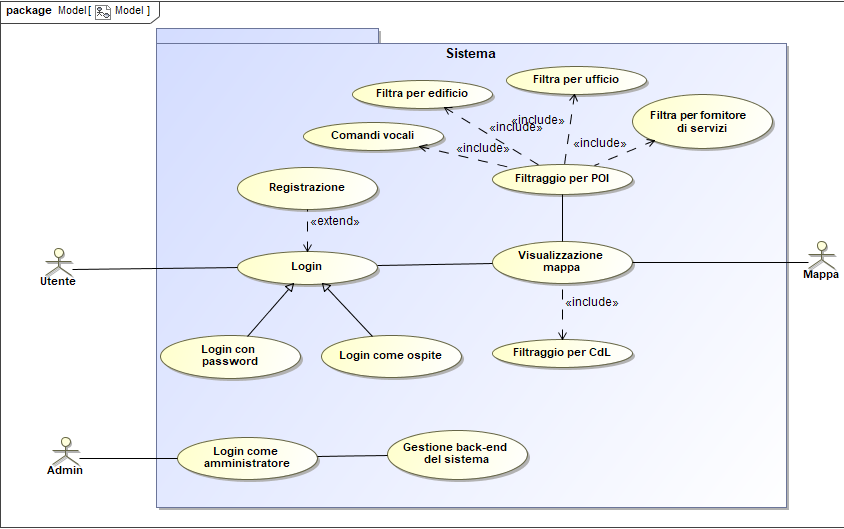
-Quando si parla di uffici ci si riferirà solo a uffici di professori.

-I fornitori di servizi saranno in qualche modo attinenti ai bisogni degli studenti, docenti e personale all'interno dell'università.

-Ogni utente potrà visualizzare tutti i POI in relazione con l'università.

-Il software utilizzerà un sistema di navigazione esterno che verrà importato.

B.6 Use Case Diagrams



C. Architettura Software

Per quanto riguarda la progettazione abbiamo utilizzato lo stile architetturale Model / View / Controller (MVC).

Le componenti sono classificate in:

* **modello**, per mantenere la conoscenza del dominio;
* **visualizzazione**, per presentare i dati all’utente;
* **controller**, per gestire l'elaborazione dei dati immessi dall'utente e rendere utilizzabili quelli di cui ha bisogno.

Questo paradigma trova la sua applicazione principale nei programmi con interfaccia grafica (GUI) e nei sistemi con interfaccia web.

Grazie a questo modello siamo riusciti a fornire un quadro chiaro del design del progetto da realizzare e a dettare le distinzioni di responsabilità tra le tre categorie e il flusso di informazioni tra di esse.

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

*Immagine che contiene mappa, testo

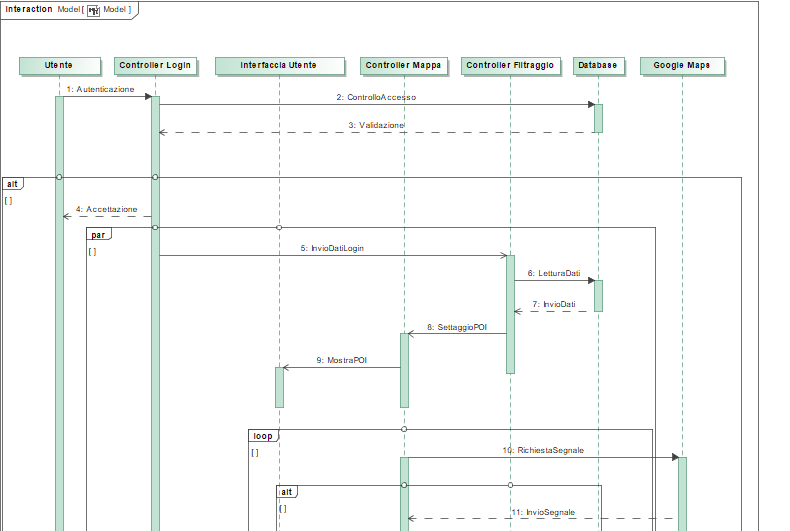
Descrizione generata automaticamente*

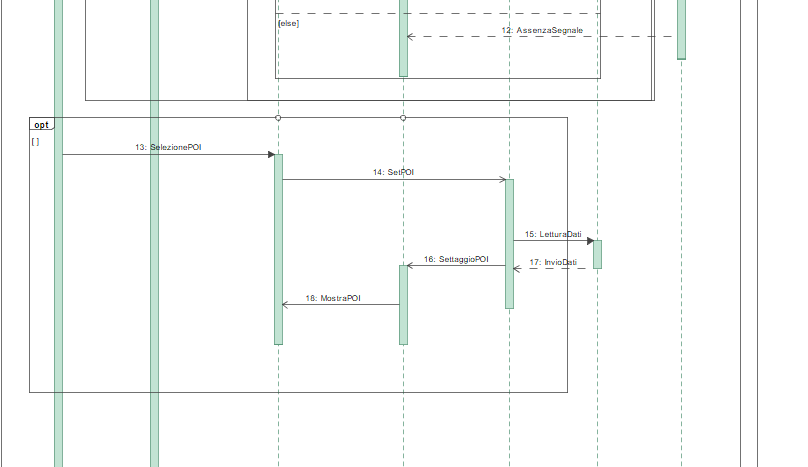
Il Component Diagram ha lo scopo di rappresentare la struttura interna del sistema software, modellando i componenti principali e le relazioni fra di essi.

Nel dettaglio, i componenti presenti nel nostro sistema sono:

* **LoginController**: i dati immessi dall'utente vengono processati da questa componente. Essa a sua volta richiede i dati alla componente “Database” e ne effettua una verifica sulla correttezza. Dopo aver effettuato con successo il login tale componente trasmette i dati alla componente “ControllerFiltraggio” e permette a “InterfacciaUtente” di iniziare a svolgere la propria esecuzione.
* **ControllerFiltraggio**: appena effettuato il login invia una richiesta di ricezione dati alla componente “LoginController”, dopo aver consultato il “Database” invierà al “ControllerMappa” solo i dati rappresentanti i POI attinenti all’utente. In generale tale componente si occuperà sempre di recuperare i dati di cui si avrà bisogno al fine di ottenere i soli punti di interesse richiesti da uno specifico utente.
* **InterfacciaUtente**: è la parte di sistema che interagisce con l’utente, comunica con la componente “ControllerMappa” in modo tale che possa visualizzare l'intera mappa dell'ateneo. Quando un utente richiede informazioni riguardanti un determinato servizio, oppure un determinato POI, tale componente invia una richiesta di ricezione dati alla componente “ControllerFiltraggio”. Grazie a quest'ultima avviene l'operazione di filtraggio dei POI sulla mappa, in modo tale che l'utente possa avere una vista focalizzata ai soli punti di interesse del servizio in questione.
* **ControllerMappa**: tale componente ha lo scopo di elaborare i dati da mostrare sulla mappa. Si occupa inoltre di comunicare all’esterno con “Google My Maps” per far sì che il servizio sia sempre disponibile.
* **Database**: si occupa di salvare sul server tutti i dati che il sistema gestisce. Questo ci permette di mantenere i dati acquisiti e di garantire sicurezza per l'accesso al sistema.
* **InterfacciaAdmin**: quest'ultima è la parte del sistema accessibile solo dagli admin, che potranno gestire in back-end tutto il sistema, aggiungendo e rimuovendo dati di uno specifico utente, servizio, POI, ecc...

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





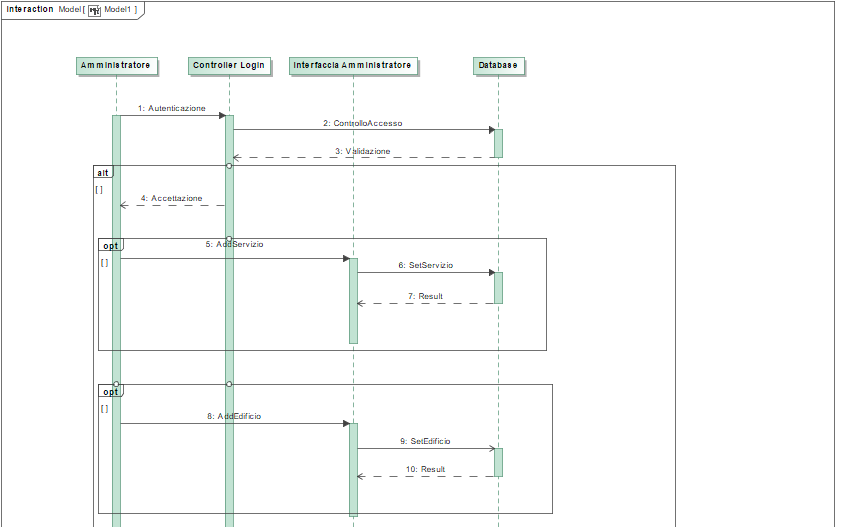


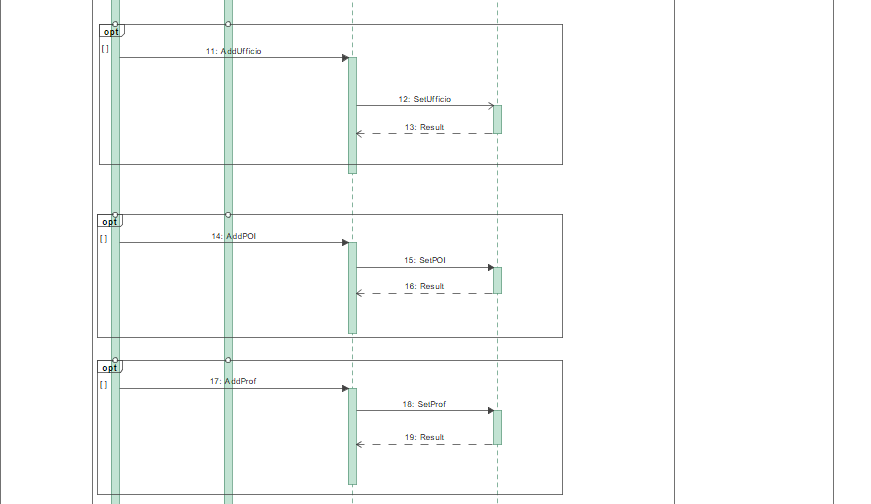
Il Sequence Diagram è un diagramma utilizzato per descrivere uno scenario, ovvero una determinata sequenza di azioni in cui tutte le scelte sono state già effettuate.

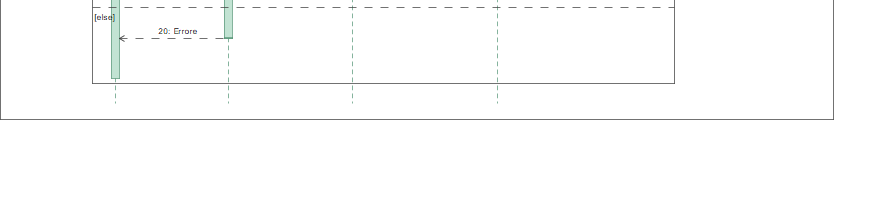
In questo esempio mostriamo come i dati acquisiti dall'utente vengono prelevati e gestiti dal **LoginController**, che li invierà al database per effettuare la validazione di tali dati, per poi permettere all’utente di entrare nella homepage del software.

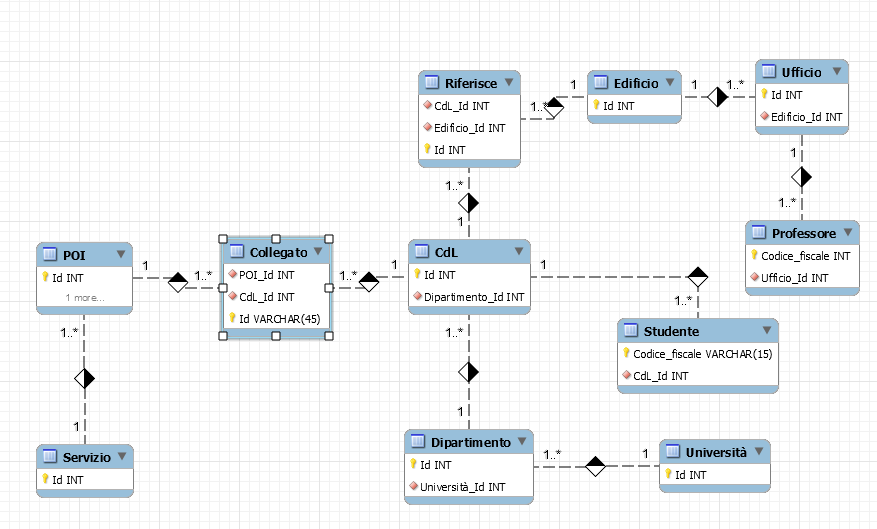
Il **ControllerMappa** si occuperà tra l’altro di comunicare continuamente con **Google My Maps** per tenere sempre a disposizione la visualizzazione della mappa.

L'**amministratore** accederà al sistema autenticandosi. Se risulterà presente nel database potrà visualizzare i dati e aggiungere e/o eliminare servizi, edifici, uffici, POI e professori, altrimenti gli verrà negato l'accesso al sistema.





D. Dati e loro modellazione

D.1 MODELLO E-R

D.2 TABELLE

-Nella tabella 'Studente' saranno memorizzati tutti gli utenti registrati immettendo Nome, Cognome, Codice fiscale e CdL.

-La tabella 'CdL' conterrà tutti i corsi di laurea che si svolgono nei 'Dipartimenti' di tutte le 'Università'.

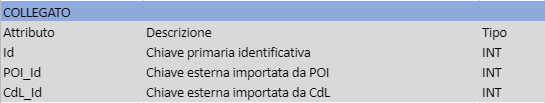
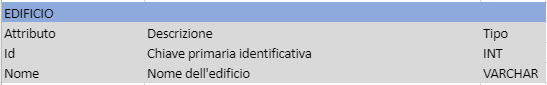
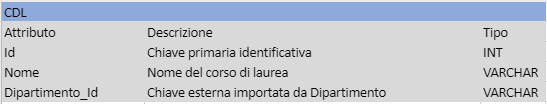
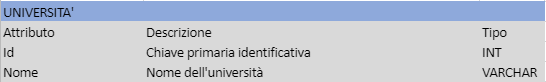
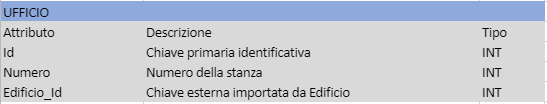
-La tabella 'Riferisce' ci serve a tenere traccia dei 'CdL' che posso essere frequentati all'interno di ogni 'Edificio'.

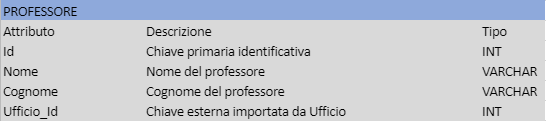
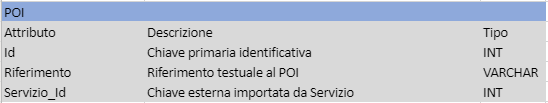
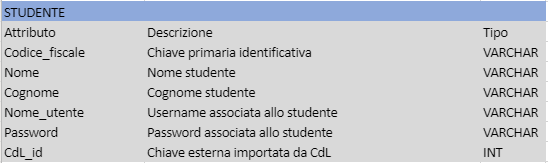
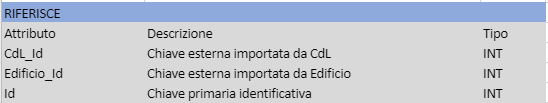
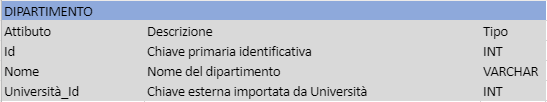
-La tabella 'POI' sarà quella con più istanze e starà a elencare tutti i possibili punti di interesse, la colleghiamo ai servizi così da poter dividere i POI per categorie.

-La tabella 'Collegato' ci aiuterà a rintracciare quali POI sono appunto

collegati, per vicinanza o attinenza, ai vari 'CdL'.

-Grazie alla relazione tra la tabella 'Ufficio' e 'Edificio' potremo risalire alla collocazione dei vari uffici all'interno dell'università.

-Attraverso la relazione che collega 'Ufficio' e 'Professore' potremo rintracciare l'aula assegnata a ogni professore dell'università.



E. Design Decisions

LOGIN

Abbiamo deciso che indifferentemente da quanti amministratori ci saranno accederanno tutti con lo stesso codice alfanumerico di otto cifre e tutti con gli stessi privilegi. Sarà possibile loggarsi nei vari modi scegliendolo nella pagina iniziale dedicata a questi.

*DATI E PROTEZIONE*

Non gestiremo in modo complesso malfunzionamenti da parte del server o del database per non aggiungere difficoltà allo sviluppo del prototipo.

Per quanto riguarda la protezione dei dati non adotteremo tecniche specifiche visto che non ci saranno dati sensibili all'interno del sistema*.*

LINGUAGGI UTILIZZATI

Per creare le pagine web utilizzeremo un pacchetto di linguaggi noti come HTML, CSS, JavaScript e PHP. Questi ci consentiranno di creare un sito dinamico e di gestire al meglio tutte le transazioni, oltre che a creare un’interfaccia grafica completamente personalizzata. Tra l'altro abbiamo scelto JavaScript perché sta guadagnando[sempre più impatto](http://redmonk.com/sogrady/2016/07/20/language-rankings-6-16/) ogni giorno e funziona quasi ovunque, minore sarà il codice scritto minori saranno gli errori, la distribuzione e la manutenzione.

Per lo storage dei dati abbiamo deciso di utilizzare MySQL che si integra alla perfezione con il PHP.

*MAPPA*

Abbiamo deciso che la scelta migliore è incorporare la mappa di Google nella nostra pagina visto il poco tempo a disposizione. La mappa sarà sempre visibile nella homepage e verrà inizializzata con icone diverse a seconda dell'utente che esegue il login.

Andremo a costruire il database in modo da poter fornire questa visualizzazione diversificata dei POI.

F. Design di Basso Livello

F.1 CLASS DIAGRAM

Immagine che contiene screenshot

Descrizione generata automaticamente

Abbiamo suddiviso il nostro class diagram secondo le funzionalità che offrono le varie classi.

Nella sezione **visualizzazione** possiamo trovare tutte le classi che interagiscono per andare a formare l’interfaccia del sistema.

La classe **controller** si occupa di controllare il flusso ed elaborare i dati.

Il **database** lavora allo scopo di memorizzare i dati e renderli disponibili nel momento del bisogno.

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

**NFRs**

Supportability e Maintanability

Per garantire la supportabilità del codice utilizzeremo I linguaggi già specificati che assicureranno la corretta eseguibilità su una grande varietà di sistemi.

Inoltre la specificità di ogni linguaggio ci faciliterà il compito di dividere il codice in moduli aventi ognuno una propria logica e di creare interfacce per determinati compiti che andremo a richiamare al momento opportuno*.*

Usabilità

Garantiremo la navigabilità del sito organizzando una homepage minimalista e il più intuitiva possibile, ci impegneremo allo scopo di rendere I comandi e l'interfaccia grafica semplici per quanto ci sarà possibile. A questo scopo per la GUI ci ispireremo ad altri siti web affermati in modo da renderla familiare.

Capacity e Scalability

Gestiremo il database in modo da rendere il più efficienti possibili le transizioni e le queries, inoltre per i periodi di grande attività terremo conto di poter aggiungere un altro database di supporto in modo da incrementare le performance.

**FRs**

Nella parte logica del sistema dovrà esserci un componente che si occuperà di eseguire queries sul database per poter filtrare correttamente I POI e visualizzarli nella mappa, realizzeremo questa funzione attraverso codice PHP.

Il sistema dovrà fornire un metodo per controllare l’accesso da parte degli utenti effettuando un controllo e successivamente riportandoli nella homepage del sito. Realizzeremo del codice lato client per verificare che verranno immesse informazioni importanti e con il PHP gestiremo e controlleremo i dati inseriti nel form della Login. Richiameremo una queries che controllerà se I dati inseriti esistono all'interno del database, visualizzando poi la giusta vista all'interno della homepage.

Per quanto riguarda la visualizzazione di una mappa forniremo il servizio importandola attraverso Google My Maps che ci permetterà di incorporarla e utilizzarla in modo attinente ai nostri scopi. Per rispettare Il copyright inseriremo un iframe che ci fornirà Google nel nostro codice. Nel complesso attraverso l'interazione tra codice PHP e database andremo a organizzare le icone da visualizzare nella mappa secondo I vari elementi da filtrare.

H. Effort Recording

H.1 PERT

*Immagine che contiene screenshot

Descrizione generata automaticamente*

H.2 LOGGING

*Immagine che contiene screenshot

Descrizione generata automaticamente*

***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***

Appendix. Prototype   
*<Provide a brief report on your prototype, and especially: information on what you have implemented, how the implementation covers the FR and NFR, how the prototypes demonstrates your project correctness with respect to the FR and NFR. You may add some screenshots to describe what required above. Be ready to show your prototype during the oral examination>*