**“Software Engineering”**

**Course**

**a.a. 2019-2020**

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

**<Progetto 4. Calendario, personalizzato, delle lezioni>**

|  |  |
| --- | --- |
| **Date** | <05/12/2019> |
| **Deliverable** | Documento Finale – D3 |
| **Team (Name)** | FDL Group |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name & Surname** | **Matricola** | **E-mail address** |
| **Fulvio Lapenna** | **252334** | **fulvio.lapenna@student.univaq.it** |
| **Federico Di Menna** | **253962** | **federico.dimenna@student.univaq.it** |

Table of Contents of this deliverable

[List of Challenging/Risky Requirements or Tasks](#_2xz3a4hmbptw)

[A. Stato dell’Arte](#_55b7dqt2si1)

[B. Raffinamento dei Requisiti](#_3o6zzq20essn)

[A.1 Servizi (con prioritizzazione)](#_66ty2yoh7h0o)

[A.2 Requisiti non Funzionali](#_66k5d169y8v2)

[A.3 Scenari d’uso dettagliati](#_6dyo4g528ln7)

[A.4 Excluded Requirements](#_rwhwl0ystre5)

[A.5 Assunzioni](#_28wqzm64p7fn)

[Use-Case Diagrams](#_9ffyl5dz0zox)

[C. Architettura Software](#_sdacaovarew4)

[C.1 The static view of the system: Component Diagram](#_33ubjk1tjnzx)

[C.2 The dynamic view of the software architecture: Sequence Diagram](#_csotg6gwsdw5)

[D. Dati e loro modellazione](#_p6ulllukmcvi)

[E. Design Decisions](#_ccbeldm2svkh)

[F. Design di Basso Livello](#_yxj22pe5gvji)

[G. Explain how the FRs and the NFRs are satisfied by design](#_tz0ke3qbb4j5)

[H. Effort Recording](#_mfiy1kwsnnna)

[Appendix. Prototype](#_jalmz5ceptxz)

## List of Challenging/Risky Requirements or Tasks

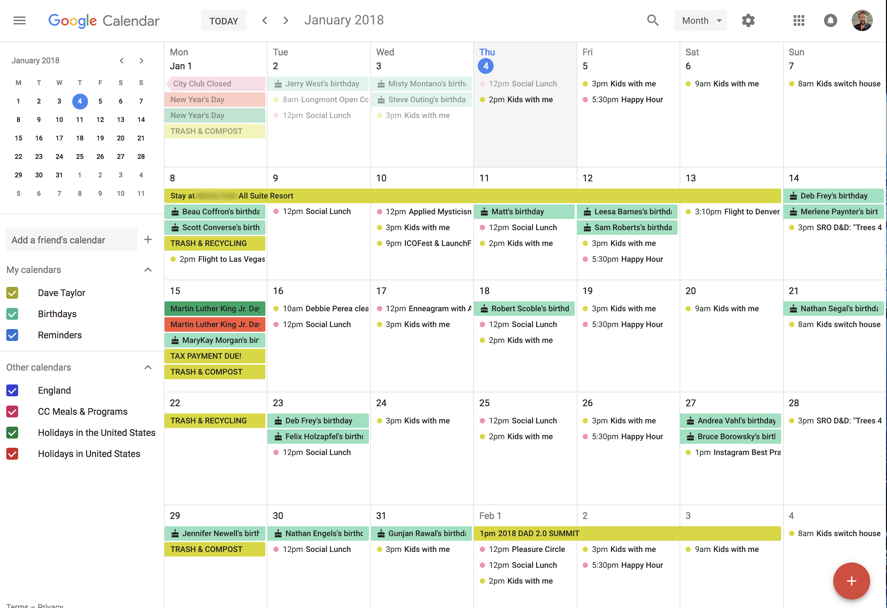
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Challenging Task** | **Date the task is identified** | **Date the challenge is resolved** | **Explanation on how the challenge has been managed** | | | |
| Visualizzazione di lezioni con orari sovrapposti | 24/11/19 | 04/12/19 | La sovrapposizione di due o più lezioni in un determinato arco di tempo verrà gestita a livello grafico tramite una label interattiva sullo slot orario che indicherà che due o più lezioni si sovrappongono e cliccando sullo stesso ne indicherà le lezioni interessate. | | | |
| Usability Interface | 24/11/19 | 02/12/19 | Il sistema avrà meno sistemi di navigazione possibile per fare in modo che l’utente possa accedere al calendario in modo più facile possibile. Infatti nella homepage si avrà una vista Real-Time delle lezioni in corso di svolgimento e uno switch-button che permetterà di passare dalla vista attuale alla vista di tipo calendario. | | | |
| Come far accedere tramite piattaforme esterne | 25/11/19 | 29/11/19 | Definizione di interfacce che definiscono quali dati ottenere per il corretto funzionamento del sistema che possono essere implementate in maniera differente a seconda della piattaforma universitaria utilizzata. | | | |
| Gestione dei corsi che possono essere seguiti oppure no dagli studenti | 27/11/19 | 29/11/19 | Gli studenti possono scegliere qualsiasi insegnamento dai corsi di laurea. | | | |

# A. Stato dell’Arte

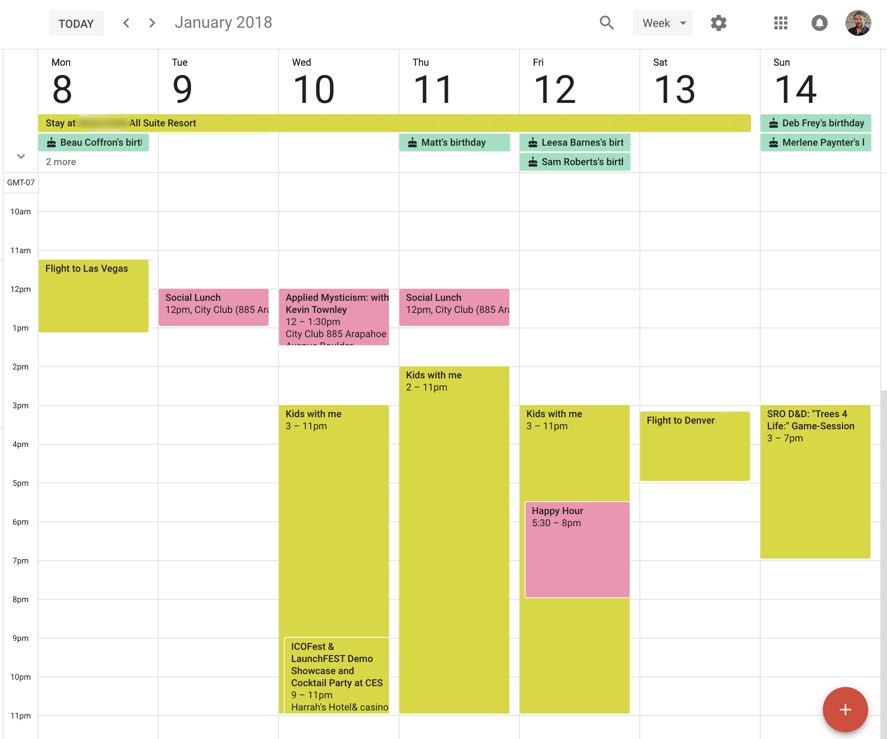
*Analizzate sistemi esistenti, prendete spunto da cio’ che esiste*

Attualmente sul mercato esistono diverse app per gestire il proprio calendario personale ma nessuna riguardante lo specifico ambito universitario.

Una delle più utilizzate e popolari è [**Google Calendar**](https://www.google.com/calendar) alla quale ci ispireremo poiché è estremamente semplice da utilizzare nonostante le numerose funzionalità che possiede (in particolare alla sua vista *settimanale*).



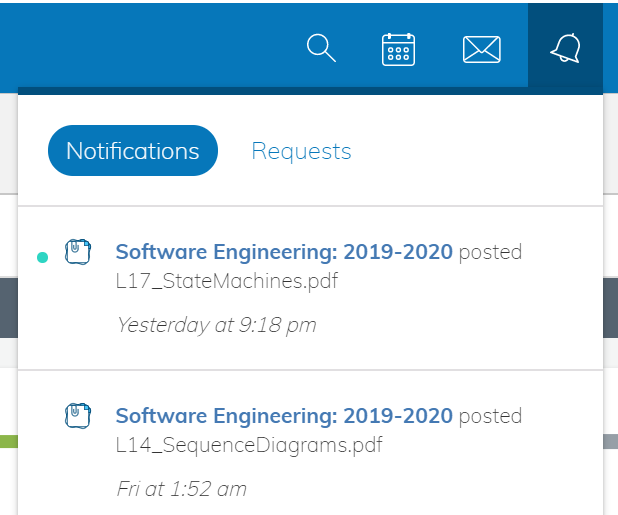
*Google Calendar Month View*



*Google Calendar Week View*

L’interfaccia principale di *Google Calendar* si pone come un calendario interattivo dove è possibile consultare, inserire e rimuovere eventi che ha diversi modi di visualizzazione (agenda, settimana, mese, anno ecc). Nella nostra applicazione il calendario avrà il solo scopo illustrativo sulla base degli insegnamenti scelti.

Per quanto riguarda gli aspetti legati alla segnalazione di cambiamenti nell’orario oltre alle notifiche che il sistema può inviare tramite il browser/app/email (a seconda dell’implementazione) abbiamo analizzato il sistema di notificazione della piattaforma di learning [**Schoology**](https://app.schoology.com/) che tiene conto (ovviamente per gli utenti registrati) se le notifiche sono state lette o meno.



*Pannello di notifiche di Schoology*

Il sistema universitario che abbiamo principalmente analizzato è quello dell’**Università degli Studi dell’Aquila**. Per quanto riguarda gli insegnamenti che uno studente può inserire nel proprio calendario, visto che quelli a scelta possono essere presi da qualsiasi corso di laurea abbiamo deciso di non imporre nessun vincolo di esclusione sugli stessi indipendentemente dal corso di laurea scelto. Ovviamente l’interfaccia dalla quale sarà possibile scegliere i propri insegnamenti prioritizzerà nella visualizzazione quelli appartenenti al proprio corso di laurea.

Abbiamo analizzato anche l’app dell’**Università degli studi “G. D’Annunzio” di Chieti** e abbiamo visto che non esiste una sezione dedicata al calendario personalizzato delle lezioni. Eventuali spostamenti vengono comunicati tramite il sito ufficiale dell’università sotto la rispettiva sezione (es. “*Avvisi di psicologia*”).

Per minimizzare il *data-entry* è possibile utilizzare piattaforme per la gestione del profilo universitario come ***ESSE3***. Quindi il sistema può far registrare un utente utilizzando le sue credenziali della piattaforma.

# B. Raffinamento dei Requisiti

*A partire dai servizi minimali richiesti, raffinate la descrizione dei servizi offerti dal vostro applicativo. Descrivete anche I requisiti non funzionali.*

## A.1 Servizi (con prioritizzazione)

*Descrivete in* ***dettaglio*** *i servizi offerti dal vostro Sistema, insieme a quelli che ritenete siano le soluzioni concettuali necessarie. In questa fase, non fate riferimento ad alcuna tecnologia specifica. Se volete, intervistate stakeholder e collezionate dati dal web o da altre sorgenti. Dovete acquisire una conoscenza avanzata dei problemi associate ai vostri servizi. Assegnate un ID a ciascun servizio. Prioritizzate inoltre I servizi in base a due scale: importanza alta, media, bassa. Complessità alta, media, bassa.*

|  |  |  |  |
| --- | --- | --- | --- |
| ***ID*** | **Descrizione** | **Importanza** | **Complessità** |
| ***1*** | Scegliere un corso di laurea e poter visualizzare le materie di quel corso di laurea che si svolgono quel giorno. | Alta | Media |
| ***2*** | Registrazione: qualsiasi utente può registrarsi al sistema. | Alta | Bassa |
| ***2.1*** | Registrazione tramite piattaforme esterne | Alta | Alta |
| ***2.2*** | Registrazione con data-entry | Alta | Bassa |
| ***3*** | Login. | Alta | Bassa |
| ***4*** | Layout di scelta tra tutti gli insegnamenti di ateneo da inserire nel calendario ( prioritizzando quelli di maggior interesse in base al corso ) | Alta | Media |
| ***5*** | Visualizzazione delle lezioni settimanali mediante un calendario con le relative informazioni (orario, aula, docente). | Alta | Media |
| ***6*** | Lista docenti con le relative informazioni (orari di ricevimento ecc…). | Media | Bassa |
| ***7*** | Dashboard | Alta | Alta |
| ***8*** | Inserire/cancellare lezioni e modificare orario e aula assegnata. | Alta | Media |
| ***9*** | Inserire lezioni annullate. | Media | Media |
| ***10*** | CRUD di eventi speciali specifici ad un gruppo di corsi di laurea. | Bassa | Media |
| ***11*** | Invio di notifiche agli utenti legate alle funzionalità precedenti. | Media | Media |
| ***12*** | conservare informazioni riguardanti alle aule: dove si trovano e come raggiungerle. | Media | Alta |
| ***13*** | CRUD aule | Media | Media |
| ***14*** | CRUD Insegnamenti | Alta | Media |
| ***15*** | Visualizzazione dettagli insegnamenti | Alta | Media |
| ***16*** | CRUD corsi di laurea | Alta | Media |
| ***17*** | Widget in home page che mostra le lezioni in corso di svolgimento (real-time) | Alta | Media |
| ***18*** | CRUD professori | Alta | Media |

**Dettagli e soluzioni concettuali adottate**

**1** L’utente (*anche senza autenticazione*) deve poter scegliere, o modificare, il proprio corso di laurea e il proprio anno di corso così da poter includere nel proprio calendario l’orario settimanale delle sue lezioni.

Bisognerà quindi creare un’interfaccia in cui, tramite dei menù a scelta, potrà essere selezionato il corso di laurea e l’anno.

**2-3** Procedure di autenticazione e registrazione al sistema. Effettuando la registrazione sarà possibile ricevere notifiche dall’applicazione in seguito a spostamenti delle lezioni e/o eventi straordinari e fornire automaticamente al sistema le informazioni riguardanti il proprio corso di laurea.

***Soluzioni scartate***: riservare tutte le funzionalità agli utenti registrati, e quindi negare la visualizzazione del calendario settimanale agli utenti senza registrazione.

***Soluzione adottata***: anche l’utente anonimo potrà ugualmente visionare il calendario fornendo il suo corso di laurea e anno, ovviamente con restrizioni legate alla sua assenza nel database del sistema (es. notificazione).

***2.1*** *Lo studente deve poter usufruire delle piattaforme che l’Università mette a disposizione per la gestione del proprio profilo universitario per accedere al nostro sistema limitando il data-entry dell’applicazione.*

***2.2*** *Il sistema deve essere indipendente da tutti gli applicativi che l’Università ha a disposizione per cui deve fornire un meccanismo di registrazione autonomo di tipo data-entry.*

**4** L’utente studente registrato, tramite un menù simile a quello del punto 1 deve poter personalizzare il proprio orario settimanale aggiungendo insegnamenti al suo corso di laurea ( a scelta o opzionali ).

***Soluzioni scartate***: vincoli sulla possibilità di seguire degli insegnamenti o meno dato un corso di laurea.

***Soluzione adottata***: Il sistema permette di seguire un insegnamento di qualsiasi corso di laurea indipendentemente da quello fornito. Nonostante ciò l’applicativo possiede un meccanismo di prioritizzazione dei corsi in base al proprio corso di laurea.

**5** Una volta selezionati i propri corsi da seguire un utente dal calendario dovrà poter visualizzare le informazioni di base riguardanti le lezioni dei corsi che ha scelto : professore, aula e orario.

**6** Il sistema deve fornire una sezione nella quale l’utente deve poter visualizzare la lista completa dei docenti, con una sezione in evidenza che elenca quelli che tengono i corsi inseriti nel calendario. Ogni elemento sarà un hyperlink dal quale poter accedere alle informazioni di ognuno, inserendo anche un meccanismo di ricerca. Quindi ogni docente avrà associata una pagina con tutte le informazioni utili.

La pagina delle informazioni di un professore deve essere raggiungibile anche dallo stesso calendario cliccando sul link del relativo docente all’interno della lezione.

**7** Il sistema deve possedere una dashboard per l’amministratore del sistema, qui si potranno eseguire tutte le attività di amministrazione del sistema descritte in seguito. La dashboard deve essere intuitiva e semplice.

***Soluzioni scartate***: account admin fornito ad ogni docente per modificare autonomamente le lezioni. Abbiamo scartato questa soluzione poiché a livello di trasparenza e sicurezza sulla disponibilità delle aule è opportuno che le richieste passino attraverso un organo specifico.

***Soluzioni adottate***: la dashboard sarà accessibile da un account di tipo admin che verrà fornito agli addetti della segreteria didattica o a chi si occupa di gestire quei servizi.

**8** Ogni lezione necessita di procedure di tipo CRUD (Create Read Update Delete) per permette agli amministratori di poter modificare le stesse. Se una lezione viene modificata viene inviata una notifica agli utenti interessati.

**9** Poter marcare una lezione della settimana come annullata in caso un professore decide di annullare una lezione. Il sistema deve fornire una procedura che permetta di visualizzare una lezione annullata nel calendario in modo diverso dalle altre e un meccanismo di notifica agli utenti interessati.

**10** L’amministratore deve poter creare eventi speciali unici ( es. Career Day ) da inserire nei calendari degli studenti dei corsi inerenti l’evento.

**11** Il sistema fornisce un meccanismo di notificazione per gli utenti registrati.

***Soluzione scartata***: la notifica viene gestita globalmente e ogni utente registrato legato ad un corso di laurea ha una sezione con gli ultimi aggiornamenti.

***Soluzione adottata***: il sistema fornisce un meccanismo di conferma di lettura della notifica, quindi concettualmente la notifica deve essere direttamente collegata con il singolo utente per gestire tale meccanismo.

**12** Il sistema dovrebbe contenere un lista delle aule, in maniera analoga a quella per i professori, da cui un utente può accedere alla pagina di un’aula per poter visualizzare tutte le informazioni utili, per esempio come raggiungerla.

**13** L’amministratore deve poter eseguire le operazioni di base CRUD sulle aule.

**14** L’amministratore deve poter modificare tutte le informazioni relative ad un insegnamento: docenti, lezioni, periodo, nome, ecc…

Inoltre può creare nuovi insegnamenti, eliminare insegnamenti esistenti e visualizzare la lista intera dei corsi.

**15** L’utente, una volta selezionato un insegnamento deve poter visualizzare tutti i suoi dettagli.

**16** L’amministratore deve poter modificare i dettagli dei corsi di laurea comprese le sue relazioni con gli insegnamenti.

**17** In Homepage il sistema fornisce un widget che mostra in real-time quali lezioni (tra quelle incluse nel calendario) si stanno svolgendo attualmente.

**18** L’amministratore deve poter eseguire le operazioni di base CRUD sui professori.

abbiamo deciso di non inserire il professore come utente del sistema, in quanto questo sistema ha come target solo gli studenti dell’università, quindi le richieste di annullamento di una lezione e aggiunta di lezioni straordinarie deve avvenire all’esterno del sistema

**19** Abbiamo deciso di dare la possibilità all’amministratore di aggiungere delle lezioni straordinarie ( come quelle di recupero ) ad un insegnamento, in modo da farle comparire nel database

## A.2 Requisiti non Funzionali

*Listate i requisiti non funzionali più importanti per il vostro Sistema*

|  |  |
| --- | --- |
| **Requisito** | **Descrizione** |
| **Simplicity / Usability** | Il sistema deve essere progettato in modo tale che sia semplice da utilizzare |
| **Interoperability** | il sistema deve essere in grado di interoperare con altri applicativi quali:  - applicativo di gestione utenza universitaria (es. ESSE3) per minimizzare il data entry |
| **Performance** | *Firm real-time system*  Il sistema deve fare un check in real-time per mostrare quali lezioni si stanno svolgendo, la compromissione non è una total system failure ma compromette la qualità del servizio |
| **Scalability** | Il sistema deve garantire la scalabilità dei dati in quanto potrebbe essere applicato a qualsiasi università |
| **Security** | Il sistema deve garantire riservatezza dei dati dei vari utenti, e deve permettere soltanto agli admin di poter modificare le lezioni e gli eventi |
| **Data Integrity** | Il sistema deve garantire che le lezioni in esso contenute sono consistenti con quelle realmente effettuate, e aggiornate agli ultimi spostamenti |

## A.3 Scenari d’uso dettagliati

*Descrivete gli scenari piu’ comuni, piu’ interessanti, o piu’ complicati d’uso dei vostri servizi.*

**Utente (anonimo) che visualizza il suo calendario:**

Un utente anonimo accede al sistema e anche senza autenticazione può chiedere di visualizzare il calendario del suo corso di laurea, il sistema chiederà quindi all’utente di specificare il suo corso di laurea, una volta scelto un corso tra quelli proposti l’utente potrà vedere il calendario per quel corso di laurea composto solo dalle lezioni obbligatorie.

**Utente che crea/modifica il suo calendario:**

Procedimento per un utente appena iscritto al sistema oppure un utente che decide di cambiare corso di laurea o modificare il proprio calendario : all’utente verrà prima chiesto il suo corso di laurea di appartenenza, successivamente il sistema chiederà all’utente di modificare il suo calendario delle lezioni, aggiungendo o eliminando degli insegnamenti ( per esempio quindi aggiungere dei corsi a scelta o opzionali al calendario, oppure eliminare un insegnamento di cui non si vuole seguire le lezioni ).

**Utente (registrato) che accede al sistema e visualizza il suo calendario:**

Un utente (studente) effettua il login con le sue credenzialied entra nel sistema. All’interno della HomePage troverà di default una vista in real-time delle lezioni che sono in corso di svolgimento. Per visualizzare il suo calendario ci sarà uno switch-button che permetterà di passare dalla vista “real-time” alla vista “calendario”. Il calendario di default sarà impostato sulla vista settimanale, ma tramite un menù a scelta possiamo ottenere una vista settimanale, giornaliera o mensile del calendario.

**Utente che visualizza i dati di una lezione:**

Un utente che si trova davanti la vista del calendario può cliccare su una lezione specifica per ottenere maggiori dettagli sull’insegnamento, docente e aula.

**Admin che gestisce il sistema:**

Un utente di tipo amministratore accede tramite le sue credenziali e sarà riconosciuto dal sistema come tale. Una volta eseguito l’accesso, si troverà davanti una dashboard dalla quale poter gestire e quindi eseguire operazioni di tipo CRUD (Create Read Update Delete) su corsi di laurea, insegnamenti, lezioni, aule, docenti. Inoltre potrà inserire eventi speciali e lezioni di recupero. Automaticamente il sistema notificherà gli utenti interessati.

## A.4 Excluded Requirements

*Descrivete i servizi da voi esclusi, e spiegatene il perchè*

**Servizio calendario anche per i professori:**

Abbiamo deciso di inserire la figura del docente solo a scopo informativo per gli studenti e non come utente del sistema. Questo perché ci è sembrato ridondante e quindi overengineering rispetto alla specifica che richiede il calendario personalizzato delle lezioni per gli studenti. ***Design Decision 2***.

## A.5 Assunzioni

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

1 - Abbiamo definito gli slot di tempo del calendario con la precisione del quarto d’ora, quindi assumiamo che le lezioni non inizino in orari che non siano ***xx:00 / xx:15 / xx:30 / xx:45.***

2 - Assumiamo che l’Università metta a disposizione la propria piattaforma di gestione del profilo universitario per registrarsi al sistema.

3 - Assumiamo che l’orario di ricevimento del docente non sia singolare per ogni corso che tiene ma unico per docente.

4 - Assumiamo che nel momento in cui si ritenga necessario verificare la disponibilità dell’aula quando l’amministratore effettua un’aggiunta o cambio aula di una lezione, l’informazione sulla possibilità di utilizzo dell’aula in una determinata fascia oraria ci venga fornita dall’esterno, poiché il nostro sistema può verificarne la disponibilità solo in base alle lezioni in essa svolte.

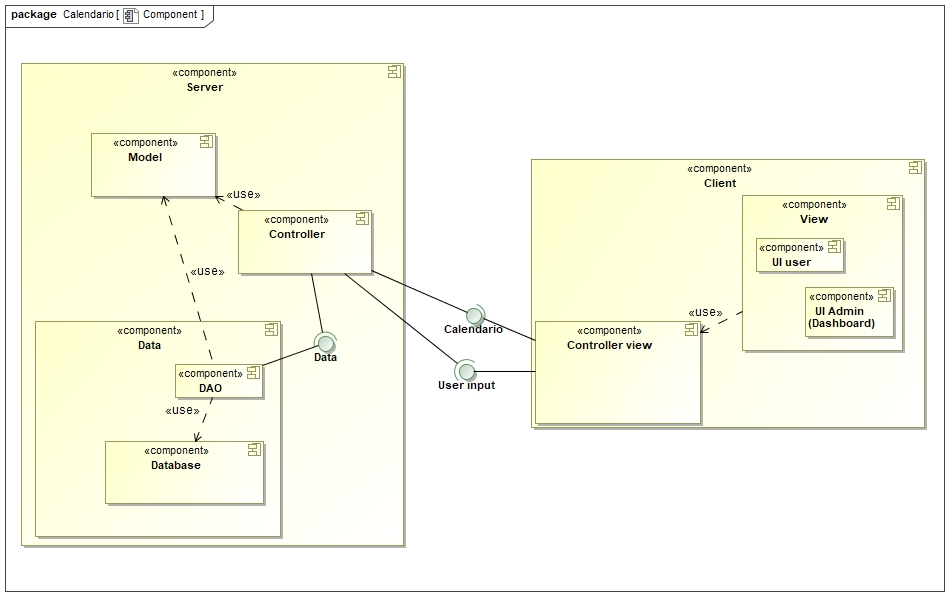
# Use-Case Diagrams



# C. Architettura Software

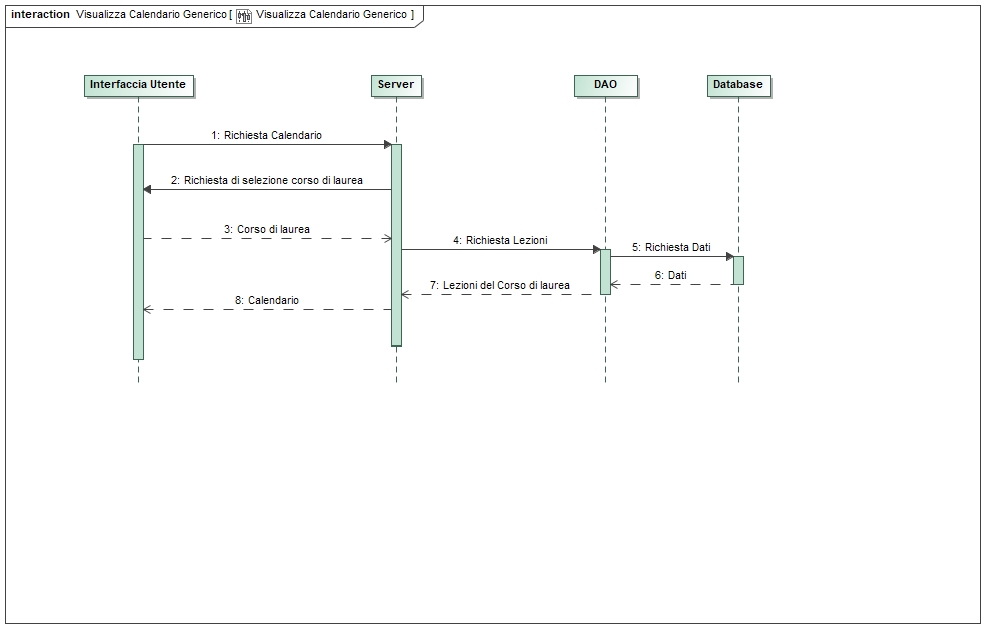
*<IF RELEVANT, Report here both the static and the dynamic view of your system design, in terms of a Component Diagram, and their related Sequence Diagrams >*

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

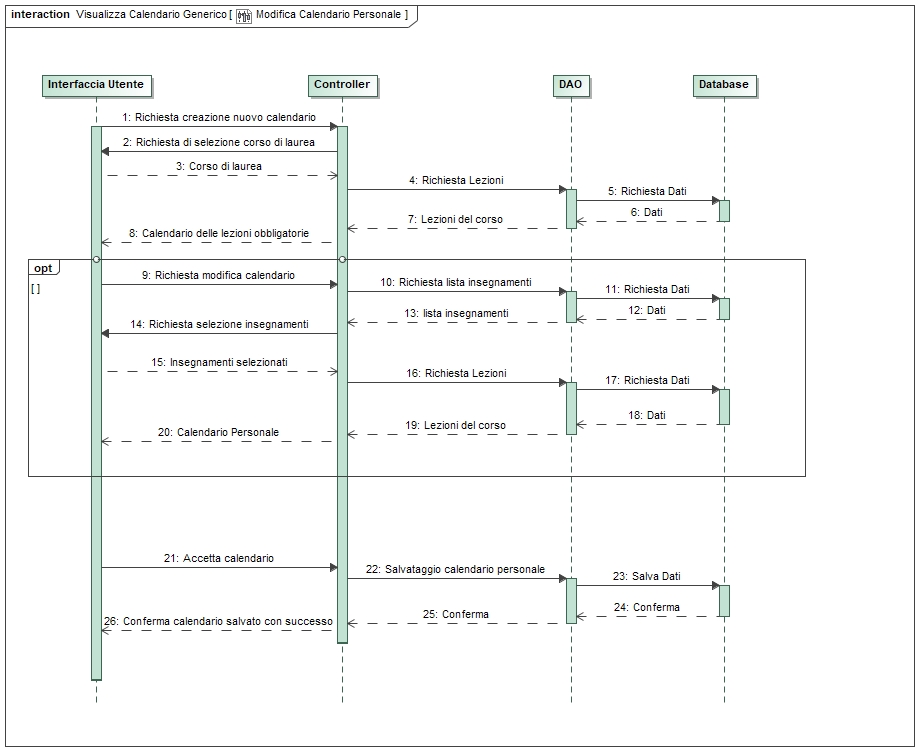


## C.2 The dynamic view of the software architecture: Sequence Diagram

1 - Un utente anonimo accede al sistema e senza fare nessun processo di autenticazione può chiedere di visualizzare il calendario del suo corso di laurea, il sistema chiederà quindi all’utente di specificare il suo corso di laurea, una volta scelto un corso tra quelli proposti l’utente potrà vedere il calendario per quel corso di laurea composto solo dalle lezioni obbligatorie.

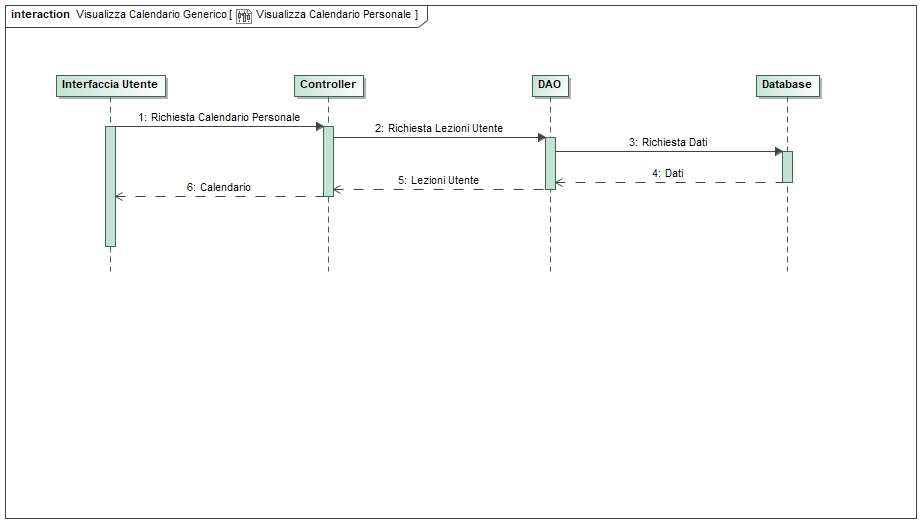


2 - Un utente si autentica per la prima volta sul sistema ( oppure decide di cambiare il suo corso di laurea all’interno del sistema, oppure decide di modificare il suo calendario ) : all’utente verrà prima chiesto il suo corso di laurea di appartenenza, successivamente l’utente potrà decidere se vuole modificare il suo calendario delle lezioni, aggiungendo o eliminando degli insegnamenti ( per esempio quindi aggiungere dei corsi a scelta o opzionali al calendario, oppure eliminare un insegnamento di cui non si vuole seguire le lezioni ).



3 - un utente accede al sistema per consultare il calendario delle sue lezioni:

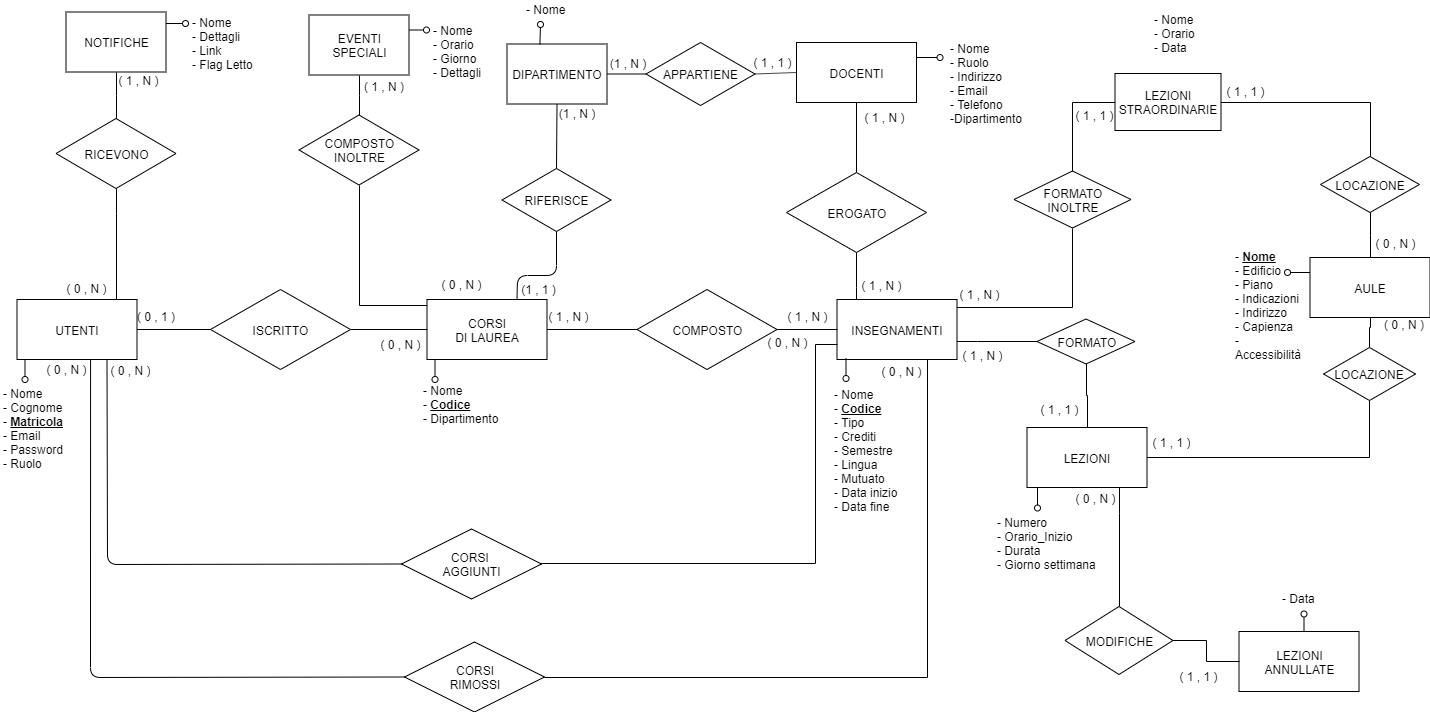
l’utente dopo essersi autenticato nella home del sistema potrà visualizzare il suo calendario, da qui potrà selezionare una lezione visualizzando tutti i suoi dettagli,potrà inoltre accedere ai dettagli di quell’insegnamento, ai dettagli dell’aula, alla pagina del docente.



4- L’amministratore, dopo essersi autenticato come tale potrà accedere alla dashboard del sistema da qui potrà modificare tutti i dati del sistema: aule, corsi di laurea, insegnamenti, lezioni, docenti, potrà inoltre creare eventi speciali, inviare notifiche agli utenti, annullare lezioni e aggiungere lezioni di recupero.

# D. Dati e loro modellazione

Definite le sorgenti di dati a voi necessarie per realizzare I servizi di cui sopra. Modellate tali dati tramite un ER o similari. Specificate se e quali di tali dati sono gia’ forniti da applicativi esistenti.



Schemi:

**Utenti** : gli utenti del sistema, ogni utente avrà associato il proprio corso di laurea, le notifiche inviategli e le informazioni aggiuntive sul suo corso. Anche gli Admin sono considerati utenti ma avranno i campi riguardanti gli studenti a NULL. Questa scelta ci è sembrata opportuna visto che gli amministratori sono in numeri nettamente inferiori agli studenti.

**Corsi di laurea** : contiene tutti i corsi di laurea presenti all’interno dell’università. ogni corso avrà associato gli eventi speciali legati a quel corso di laurea, e gli insegnamenti obbligatori e opzionali per quel corso di laurea.

**Insegnamenti** : contiene tutti gli insegnamenti erogati dall’università, ogni insegnamento avrà associato i propri docenti e le proprie lezioni.

**Lezioni** : Le lezioni associate ai vari insegnamenti verranno gestite nel seguente modo per limitare le ripetizioni:

per ciascun insegnamento abbiamo deciso di inserire solamente l’orario settimanale delle lezioni ( quindi per un insegnamento classico formato da due lezioni a settimana lo schema conterrà soltanto due entry, una per ogni lezione settimanale, *ad esempio per un corso che fa lezione il giovedì non ci sarà una entry per ogni lezione del giovedì per tutto il semestre, ma soltanto una sola entry che ricorda che quell’insegnamento ha una lezione il giovedì* ).

Ogni lezione ha associata un aula

***\* Le due relazioni “Corsi Aggiunti” e “Corsi Rimossi” servono alla personalizzazione del calendario: di base un utente possiede i corsi obbligatori in base al suo corso di laurea, anno di corso e semestre. Per eliminare parte di questi ultimi utilizziamo la relazione “Corsi Rimossi”. Invece per aggiungere corsi non presenti in quelli di base viene utilizzata la relazione “Corsi Aggiunti”. La scelta è stata di questo tipo poiché la maggior parte degli studenti intende seguire esclusivamente i corsi obbligatori.***

**Docenti** : contiene tutti i dettagli riguardanti i docenti, sono associati agli insegnamenti da loro erogati.

**Aule** : contiene tutti i dettagli riguardanti le aule dell’università: nome, edificio, piano.

Le prossime due tabelle servono a modificare le singole lezioni all’interno del calendario, cosa non possibile nello schema ‘Lezioni’

**Lezioni annullate** : questo schema contiene una entry per ogni lezione annullata, ogni lezione è associata alla lezione della settimana a cui si riferisce, in più contiene la data che specifica quale delle lezioni è stata annullata.

**Lezioni straordinarie** : questo schema contiene una entry per ogni lezione straordinaria ( per esempio una lezione di recupero o una lezione con membri esterni in aula magna ecc..), ogni lezione è associata all’insegnamento a cui si riferisce.

**Eventi Speciali** : qui vengono conservate le informazioni riguardanti eventi speciali ( career day ecc.. ) al di fuori delle lezioni associati ad uno o più corsi di laurea.

11- **Notifiche** : questa tabella conserva le informazioni sulle notifiche inviate agli utenti, una entry per ogni notifica .

**NOTA:** *con la gestione attuale non viene permesso ad una lezione di essere associata a più insegnamenti (questo avrebbe aumentato molto la complessità del sistema per dei casi molto rari), nel caso quindi di una lezione seguita da più insegnamenti ci sarà bisogno di una entry per ogni insegnamento riferente alla stessa lezione.*

# 

# E. 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. **DataBase Relazione o non relazionale?**

Abbiamo valutato entrambe le opzioni e la scelta è stata quella di utilizzare un DBMS Relazionale poiché i dati da modellare sono eterogenei e non sono una quantità estremamente grande. In più il tempo a disposizione per la realizzazione del sistema non è sufficiente per apprendere in modo esaustivo NoSQL.

Nonostante ciò abbiamo preso in considerazione l’opzione di utilizzare un meccanismo NoSQL sulla relazione tra Studente e Insegnamento che è quella più corposa. Fino ad ora il database è stato completamente modellato in maniera relazione, ma una ristrutturazione del tipo precedente non costa molto tempo in quanto basta eliminare le due relazioni esistenti e aggiungere un campo di tipo JSON all’entità studente che ne descrive i vari insegnamenti.

1. **Professore come utente oppure come Entità?**

Una delle scelte più influenti su tutto il design del sistema è stata quella della modellazione della figura del professore. La scelta iniziale è stata quella di inserire il professore come tipologia di utente visto che poteva essere utile per un professore avere un calendario delle sue lezioni e dei meccanismi di richiesta automatici di cambio/annullamento lezioni con gli amministratori (didattica).

Una scelta di questo tipo avrebbe aumentato di molto la complessità del sistema poiché avremmo avuto dovuto inserire dei casi d’uso specifici per il professore e un nuovo attore. Inoltre, analizzato l’attuale sistema universitario, abbiamo notato che ogni professore contatta la didattica esternamente come gli è più comodo, quindi sarebbe stato necessario studiare un modo estremamente semplice per fare in modo che il professore contatti in modo automatico la segreteria.

Visto che la specifica ci richiede un ***calendario personalizzato delle lezioni per gli studenti*** abbiamo deciso di non aggiungere questi requisiti funzionali.

1. **Il sistema interagisce con altri applicativi per mantenere la Data Integrity?**

Visto che il sistema deve essere in grado di mantenere la Data Integrity, e quindi coerente con quello che realmente accade in aula, abbiamo pensato che il sistema deve avere delle interfacce che comunicano con delle API fornite da sistemi esterni almeno per la verifica se un’aula è libera in un intervallo di tempo.

1. **La lezione straordinaria viene considerata un evento oppure un’entità a parte?**

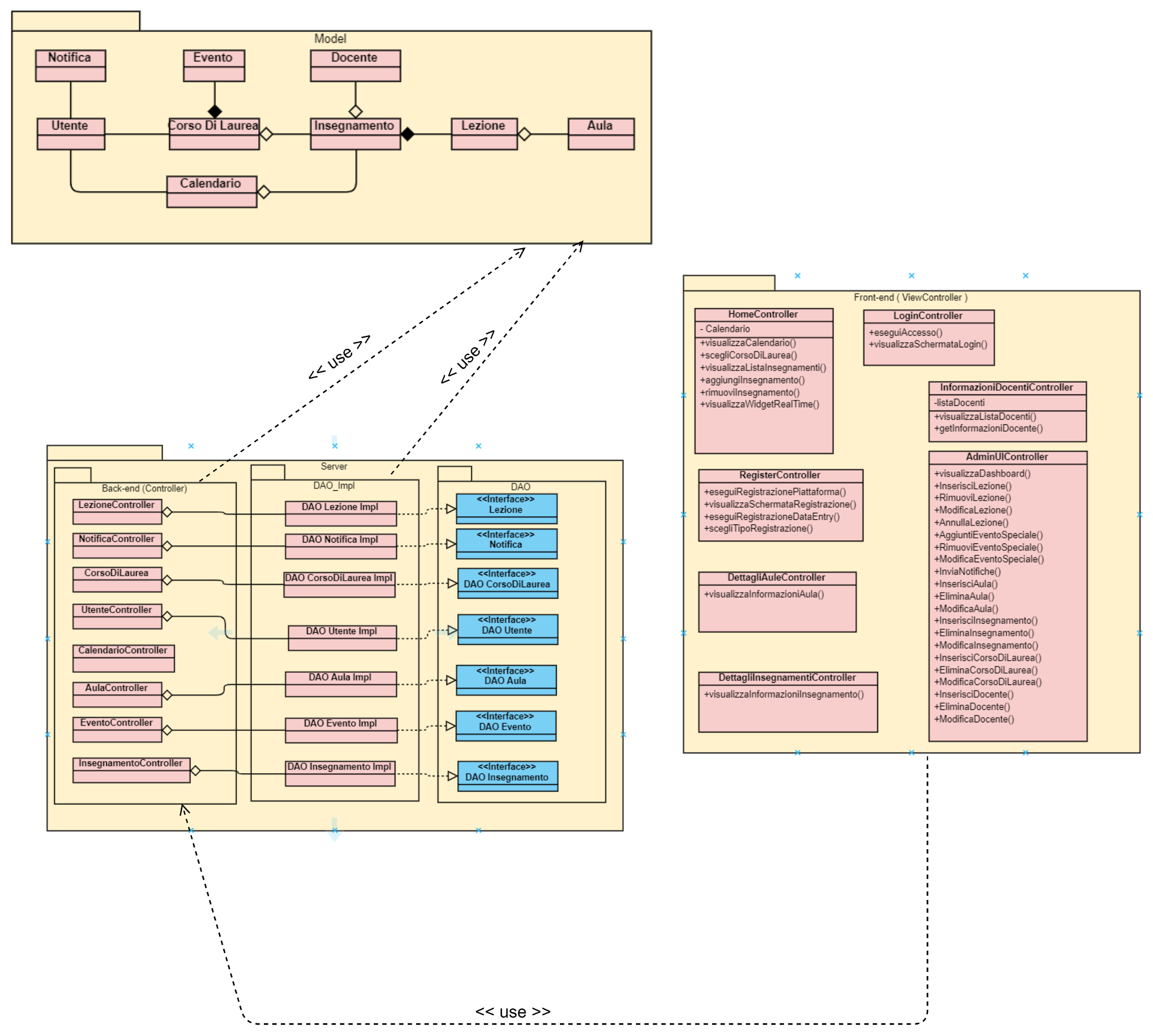
La lezione straordinaria viene considerata un’entità a parte perché un evento riferisce ad un gruppo di corsi di laurea e una lezione straordinaria riferisce a un insegnamento, quindi è opportuno modellare i due concetti diversamente.

1. **Un insegnamento è singolare per ogni corso di laurea?**

Ci sono degli insegnamenti che sono comuni a più corsi di laurea.

Abbiamo scelto di gestire questi insegnamenti come se fossero diversi visto che nella maggior parte dei casi hanno anche denominazione diversa. Quindi esisteranno diverse istanze dell’insegnamento per ogni corso di laurea al quale appartiene.

# F. Design di Basso Livello



Nel Class Diagram sono stati omessi attributi e metodi del Model in quanto gli attributi sono quelli del modello ER e i metodi sono i getter e setter.

Inoltre sono stati omessi i dettagli di backend poiché verranno definiti successivamente in quanto dipendono anche dall’implementazione.

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

*<Report in this section how the architectural and low level design you produced satisfies the FRs and the NFRs>*

**Requisiti funzionali**

Per quanto riguarda i requisiti funzionali di visualizzazione del calendario e del widget real-time in home page il sistema provvederà tramite i controller di front-end a restituire le dovute viste all’utente.

Riguardo autenticazione e registrazione sono stati progettati i relativi controller front-end, con in aggiunta dei metodi per fare in modo che l’utente possa registrarsi tramite piattaforme esterne.

Per la creazione / modifica del calendario sono stati progettati i metodi all’interno di ***HomeController*** che permettono di soddisfare tali operazioni.

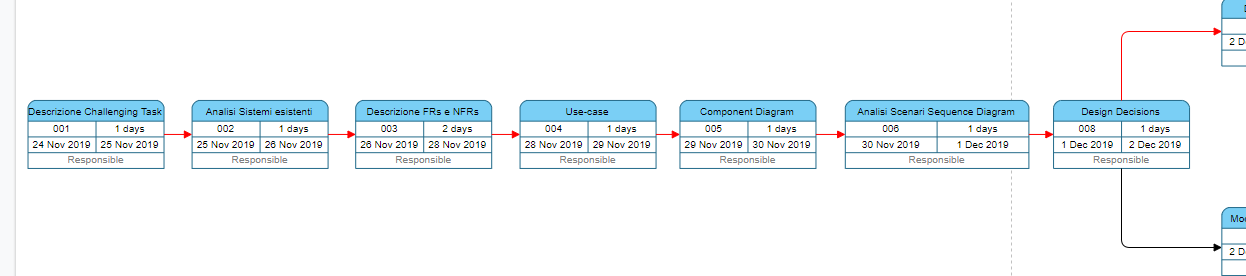
Riguardo tutti i requisiti funzionali per quanto riguarda l’amministrazione è stato modellato tutto tramite ***AdminUIController*** che si occupa di gestire tutte queste funzionalità.

**Requisiti Non Funzionali**

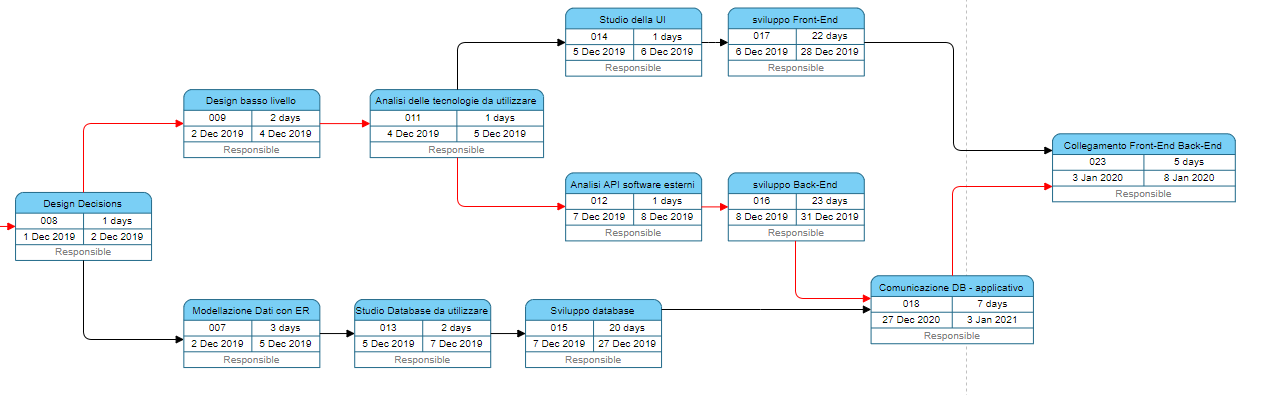
|  |  |
| --- | --- |
| ***Requisito*** | **Descrizione** |
| ***Simplicity / Usability*** | Il layout è progettato in maniera tale che il calendario sia raggiungibile facilmente (essendo situato nella homepage) e tutte le informazioni che l’utente deve inserire nel proprio calendario vengono richieste automaticamente dal sistema tramite un unico Wizard. |
| ***Interoperability*** | Il sistema possiede le interfacce necessarie all’interoperabilità con il software che gestisce le aule, e quello che gestisce il profilo universitario degli studenti. |
| ***Performance*** | Il sistema genera automaticamente ogni settimana una vista materializzata nella quale saranno contenute le informazioni da visualizzare nei calendari limitando gli accessi a più tabelle sul database. |
| ***Scalability*** | Abbiamo fatto in modo di rappresentare in maniera più astratta possibile tutte le informazioni inerenti le Università in modo tale favorire il più possibile la condivisione di informazioni tra le varie componenti senza doverle duplicare. |
| ***Security*** | Procedure necessarie di controllo nei meccanismi di autenticazione e autorizzazione. |
| ***Data Integrity*** | Il sistema fornisce un meccanismo di verifica sulle aule libere in una determinata fascia oraria. |

# 

# H. Effort Recording

**PERT (part1)  
**

**PERT (part2)**

****

Il PERT è stato Splittato per ragioni di leggibilità.

**Logging**   
*Vedi allegato* ***Logs.xlsx***

**Categorization**  
  
**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>*