**“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> <21/12/2019> |
| **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](#_5wi2t5occ1i8)

[B. Raffinamento dei Requisiti](#_3o6zzq20essn)

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

[Dettagli e soluzioni concettuali adottate](#_916m71fy8oe2)

[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)

[Visualizzazione calendario generico](#_8yq67697rclo)

[Modifica il tuo calendario personale](#_lj7etp4oove9)

[Visualizza calendario personale](#_z8dq3mow547d)

[Dashboard](#_xql07vc26auu)

[Visualizza dettagli lezioni e insegnamenti](#_33rona563zdq)

[D. Dati e loro modellazione](#_7x6ni99jb8sr)

[Descrizione delle entità del sistema](#_dd24bt32hqbp)

[Database Design Decisions Description](#_l3dse3cna9ko)

[Notifiche](#_viyjpgbzckla)

[Nuove notifiche non lette](#_nqk90j3nrwe9)

[Gestione dei calendari personalizzati](#_w82sgyrjt80b)

[Utilizzo di una vista materializzata per le lezioni settimanali](#_uge5ryole6u9)

[Raggruppamento dei corsi di laurea](#_aofnf6gv9c10)

[Dizionario dei termini](#_v2gl98wmjkgw)

[Tabelle aggiuntive](#_yck42epckwg2)

[Diagramma ER Ristrutturato](#_ti1vcyek7b3g)

[Vista Materializzata](#_2ox2awjqq4k2)

[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

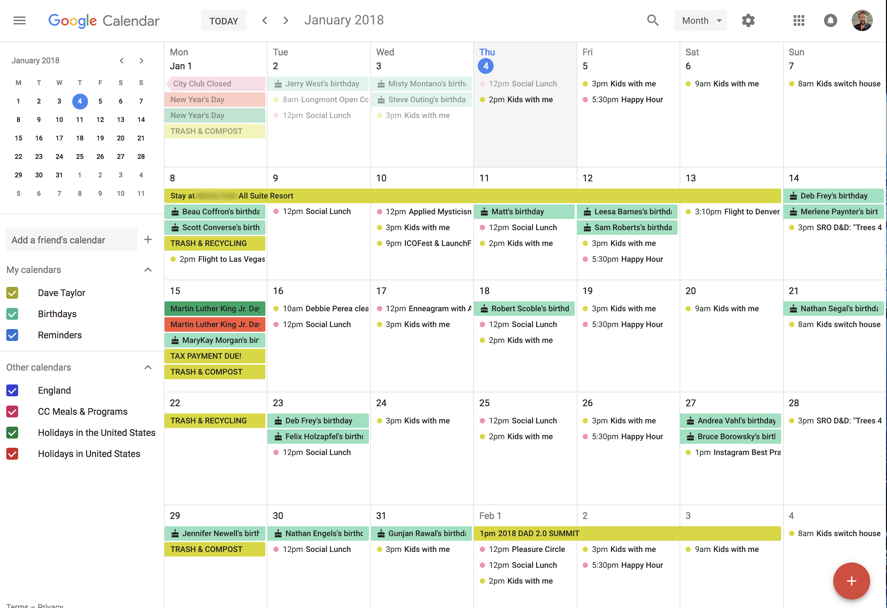
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Challenging Task** | **Date the task is identified** | **Date the challenge is resolved** | **Explanation on how the challenge has been managed** | | | |
| Visualizzazione di lezioni con orari sovrapposti | 24/11/19 | 04/12/19 | La 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. | | | |
| Informare gli studenti sul cambio delle lezioni | 14/12/19 | 14/12/19 | Ogni insegnamento contiene un insieme di avvisi in cui vengono salvate le modifiche fatte all’orario delle lezioni. Il sistema non controlla se l’utente ha letto o meno la notifica. | | | |

# 

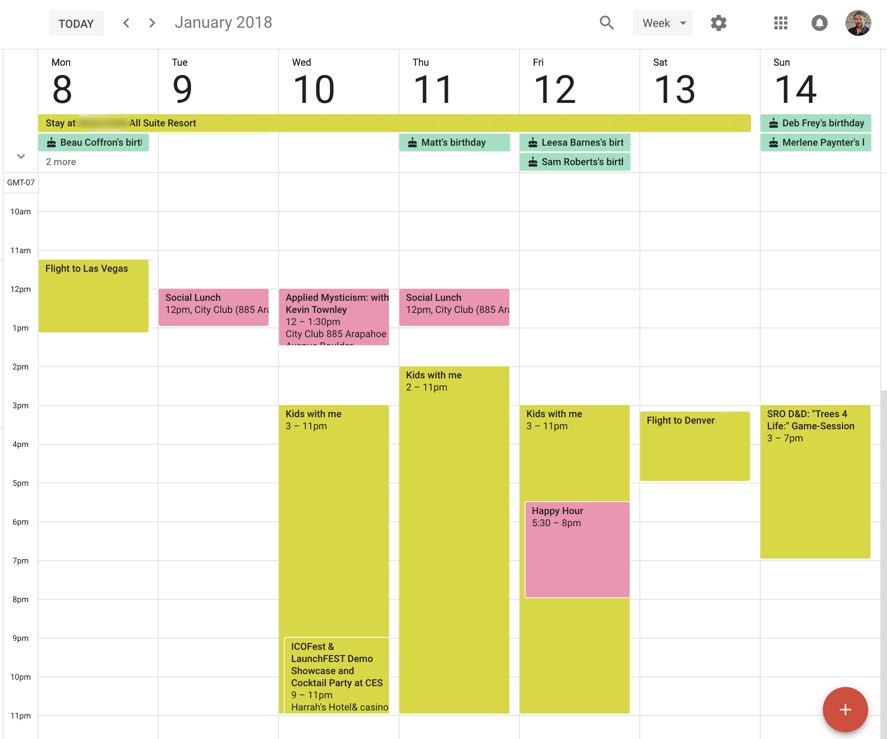
# A. Stato dell’Arte

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*).



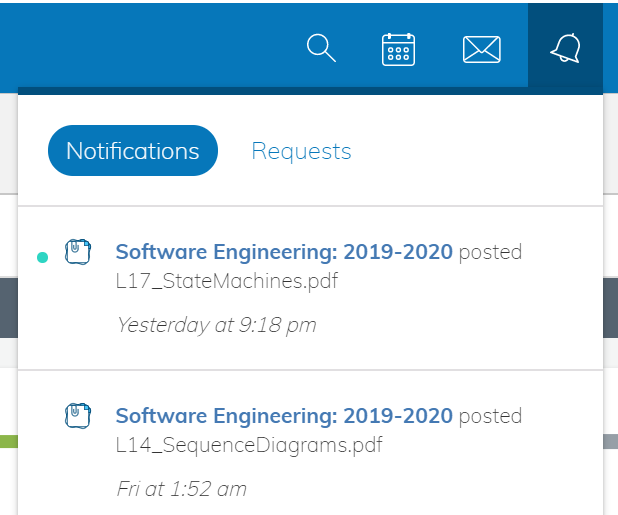
*Google Calendar Month View*



*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.

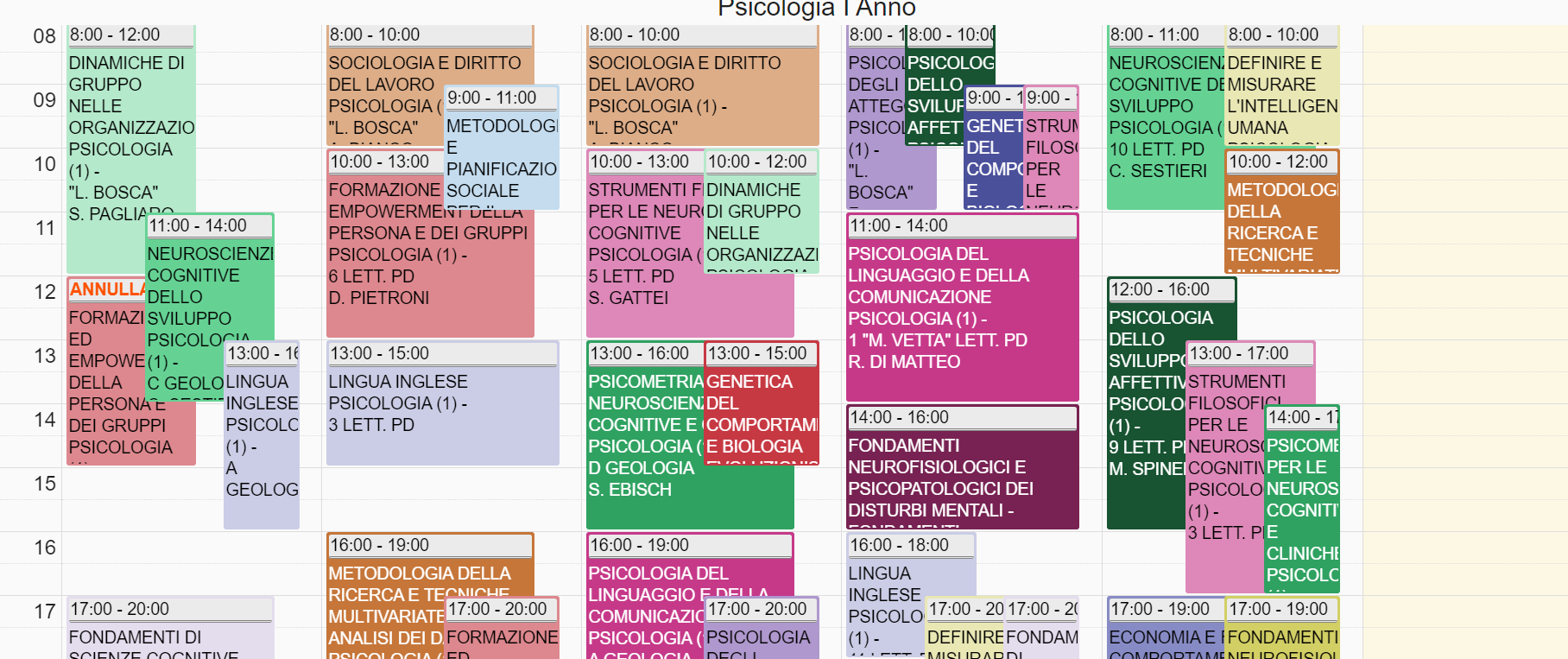
Per quanto riguarda gli aspetti legati alla segnalazione di cambiamenti nell’orario oltre alle notifiche che il sistema può inviare tramite il browser/app/email (a seconda dell’implementazione) abbiamo analizzato il sistema di notificazione della piattaforma di learning [**Schoology**](https://app.schoology.com/).



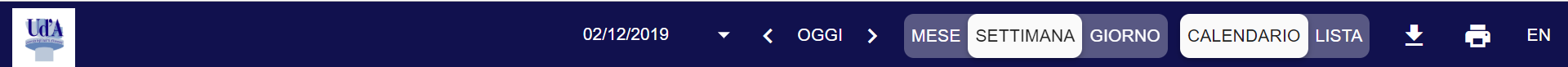
*Pannello di notifiche di Schoology*

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.

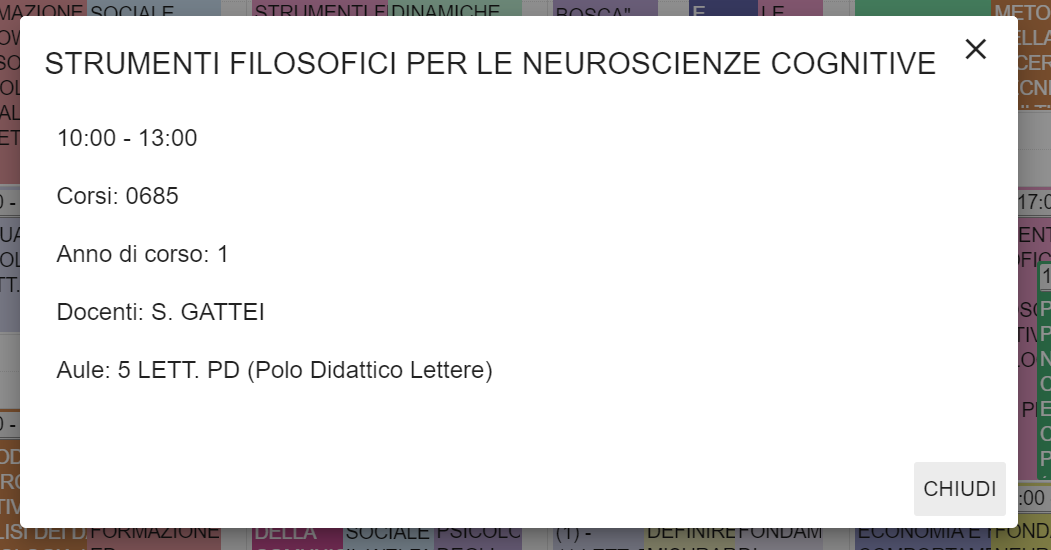
Abbiamo analizzato anche l’app dell’**Università degli studi “G. D’Annunzio” di Chieti** e abbiamo visto che non esiste una sezione dedicata al calendario personalizzato delle lezioni. Eventuali spostamenti vengono comunicati tramite il sito ufficiale dell’università sotto la rispettiva sezione (es. “*Avvisi di psicologia*”). Invece per quanto riguarda il [**sito web**](https://www.unich.it/) della stessa offre un calendario diviso per corsi di laurea e anno (e di conseguenza semestre) :



Con una sezione a inizio pagina di setting:



Possiamo scegliere il giorno che ci interessa, switchare il modo di visualizzazione (mese, settimana, giorno, e calendario o lista), creare un iCal link (che offrirà anche il nostro sistema), la stampa e selezionare la lingua. Questo è un valido spunto, infatti gestiremo il layout allo stesso modo. Le lezioni annullata vengono identificate con un’etichetta in cima allo slot. Cliccando sulla lezione avremo i relativi dettagli:



I dettagli che inseriremo noi saranno: *corso di laurea, curriculum, insegnamento, intervallo orario, anno di corso, aula e docente.* Nel nostro caso il docente sarà un link che redireziona l’utente verso la pagina di informazioni del docente. Stessa cosa per quanto riguarda l’aula. La visualizzazione mensile del calendario non tiene conto (graficamente) delle sovrapposizioni, infatti elenca le lezioni per giorno in ordine cronologico di inizio. Il calendario mensile si presenta come la vista settimanale ma ripetuta per 4 volte di seguito (ovvero tutto il mese). Nel nostro caso la vista mensile funzionerà in modo simile, ma verrà visualizzato soltanto nome e orario di inizio e fine lezione e cliccando su di esso si avranno tutti i dati.

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*** | Dettagli 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 creare un proprio calendario personale delle lezioni da seguire.

***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.

***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.

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

***Soluzione scartata***: 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.*** Sono state prese in considerazioni opzioni più performanti (come slot di 1 ora / 30 min) ma abbiamo scelto di lasciare maggiore flessibilità per eventi speciali in aggiunta alle lezioni.

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.

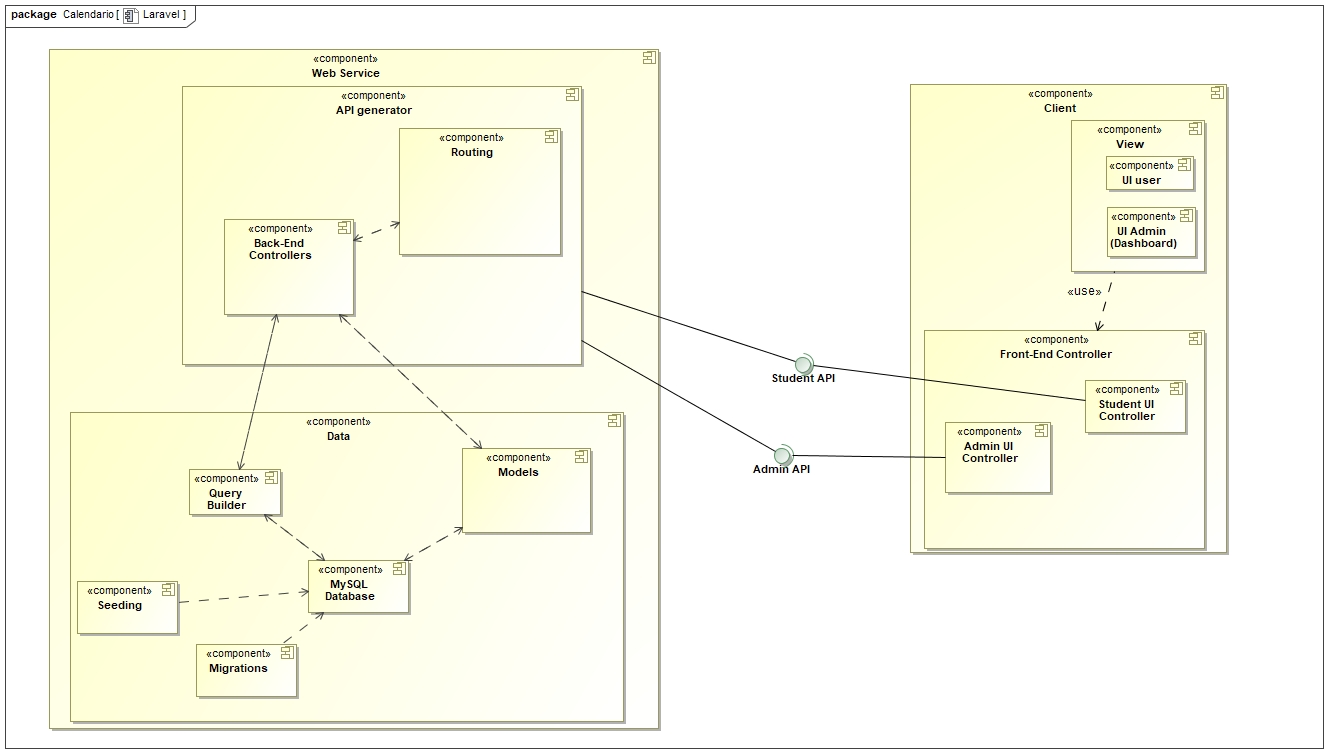
# Use-Case Diagrams



# 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



Le APIs sono fornite ai Front-End Controller da parte di :

**Routes**: gestiscono le richieste in arrivo dal Client in base all’URL. In base all’operazione da svolgere chiamano i controller di Back-End specifici.

**Back-End Controllers**: insieme di classi in cui è contenuta la logica del web service.

Comunicano con il database direttamente tramite un “**query builder**” e indirettamente tramite i **Models**.

**Models**: modellano le varie entità del sistema, prendono il dato dal database e lo restituiscono modellato al Back-End Controller.

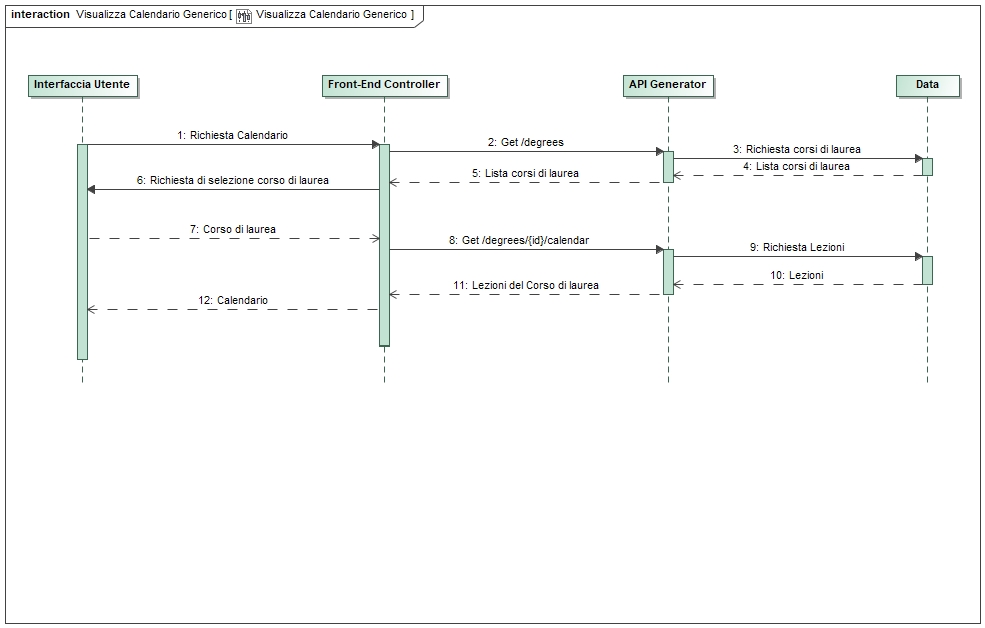
**Migrations** : aiutano a creare lo schema del database tramite lo “schema builder”.

**Seeding** : aiutano a popolare il database per il testing.

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

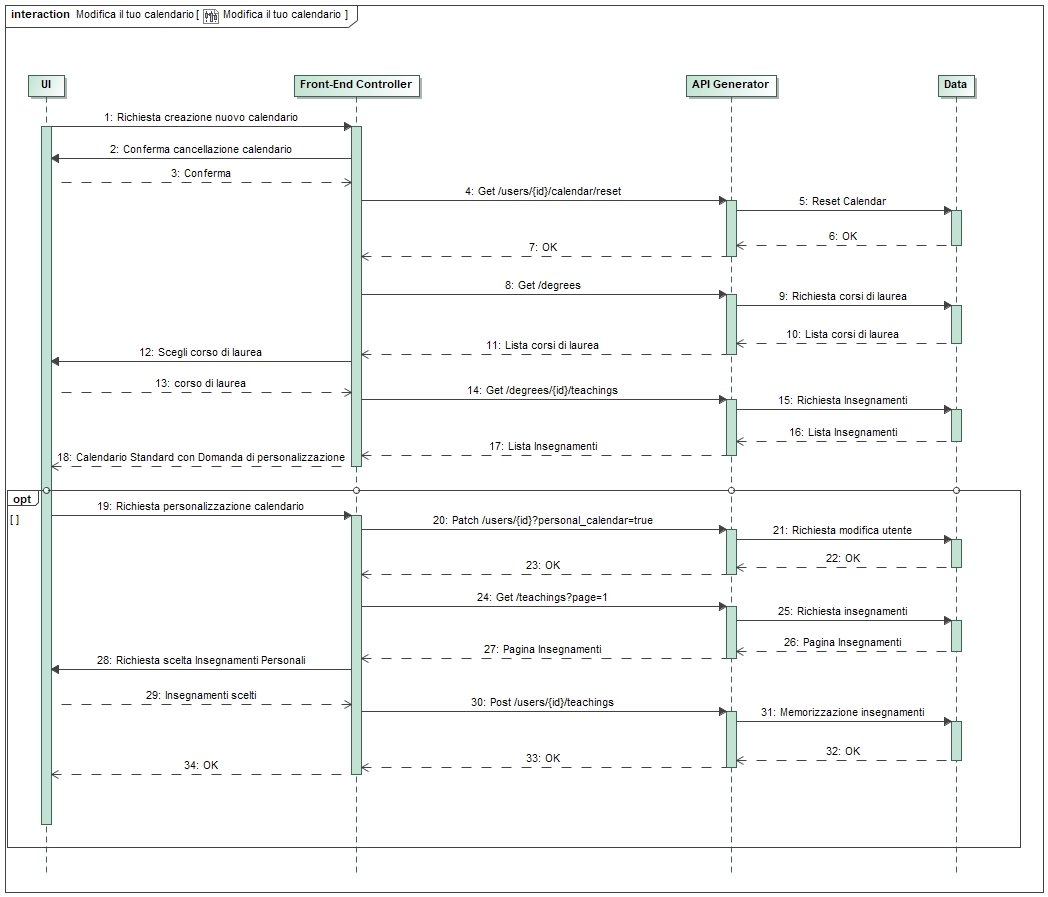
### Visualizzazione calendario generico

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.



### Modifica il tuo calendario personale

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 ).

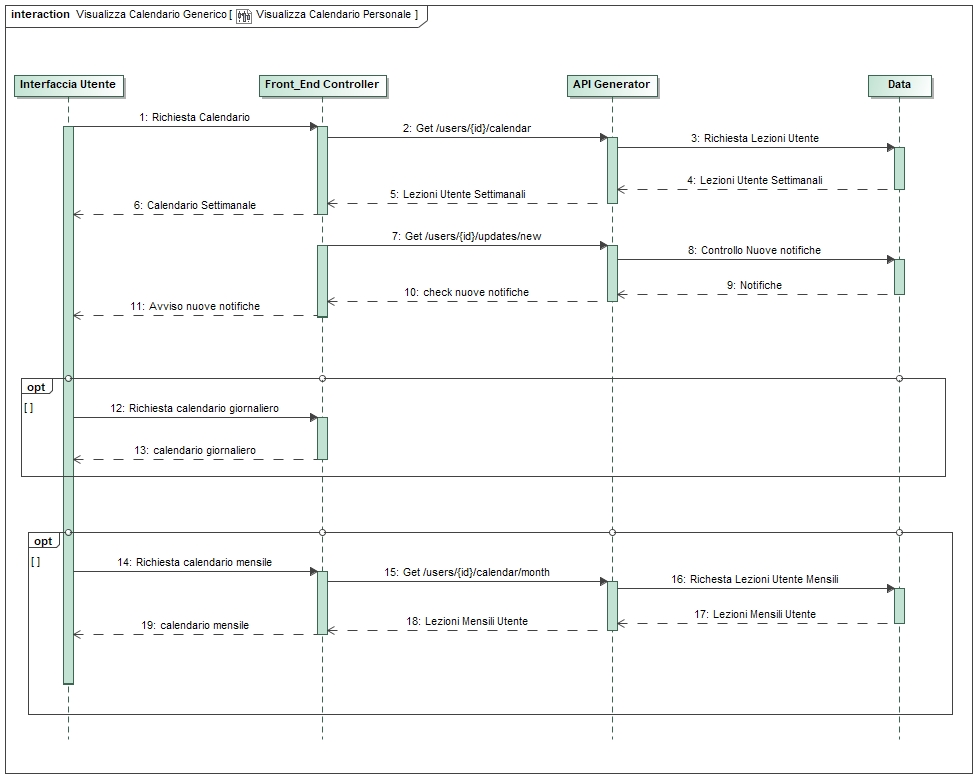


### Visualizza calendario personale

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, Qui potrà visualizzare le sue lezioni in due modi diversi, o visualizzare le lezioni della giornata corrente, oppure visualizzare le lezioni della settimana corrente.

Inoltre dalla home l’utente verrà avvisato, mediante un icona evidenziata, se ci sono nuove notifiche inerenti i propri insegnamenti.

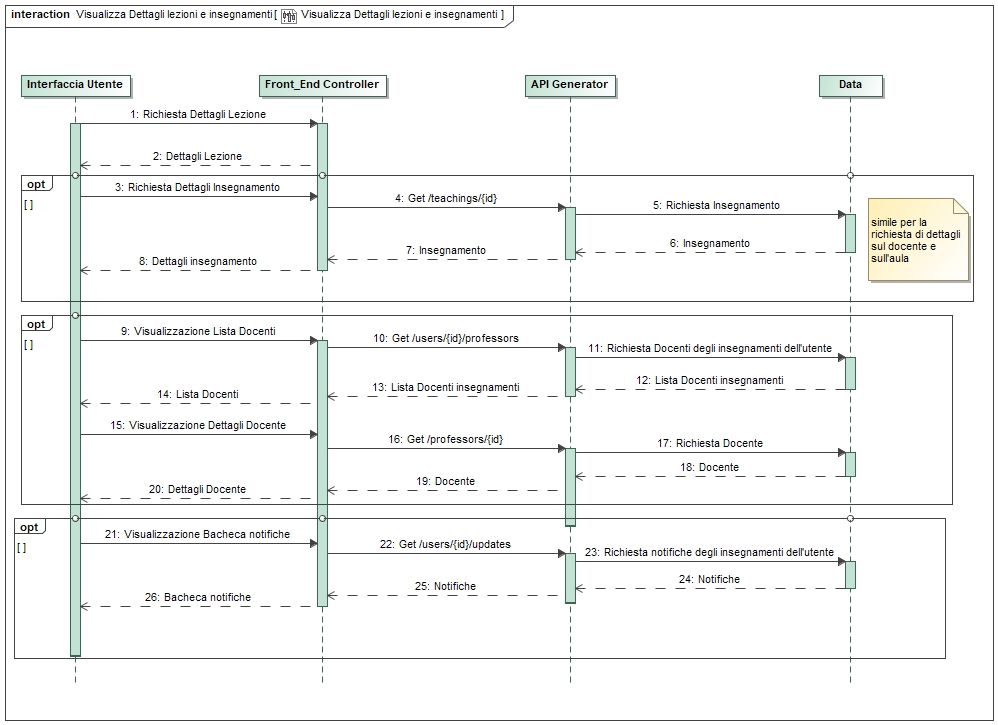


### Visualizza dettagli lezioni e insegnamenti

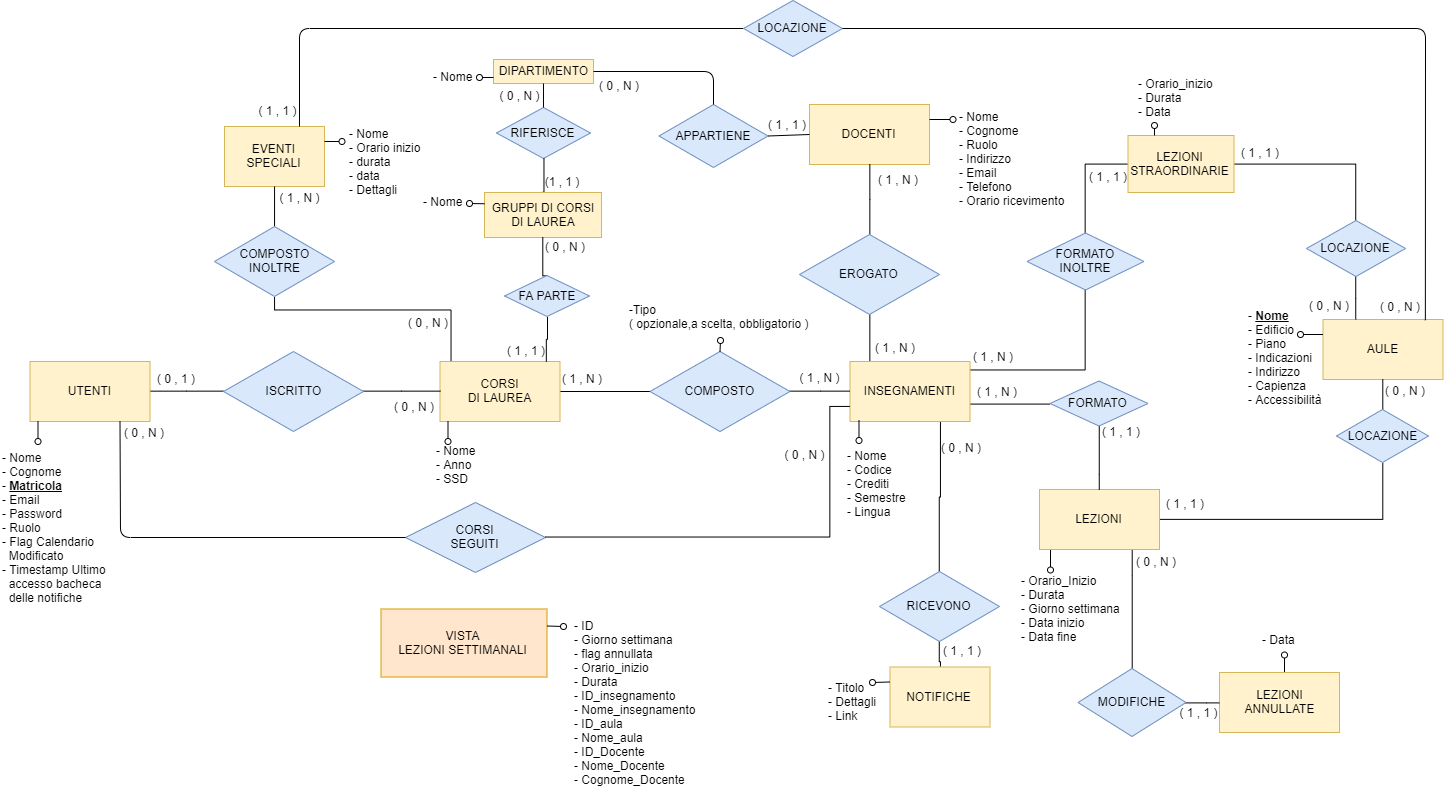
4 - L’utente, una volta acceduto al calendario delle lezioni, potrà cliccare su una delle lezioni per accedere a tutti i dettagli di quella lezione, da qui potrà accedere anche ai dettagli dell’insegnamento, del professore e dell’aula.

Dal proprio calendario l’utente potrà accedere anche alla lista dei docenti dei suoi insegnamenti.

L’utente potrà anche accedere alla propria bacheca delle notifiche in cui potrà vedere le notifiche associate agli insegnamenti seguito.



# D. Dati e loro modellazione



### Descrizione delle entità del sistema

**Utenti** : gli utenti del sistema, ogni utente avrà associato il proprio corso di laurea 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.

Flag Calendario Modificato: questa flag avrà valore 0 se l’utente non ha modificato il proprio calendario. 1 se invece l’ha modificato ( verrà spiegato meglio più avanti cosa si intende con calendario modificato )

Timestamp ultimo accesso bacheca delle notifiche: contiene il timestamp dell’ultimo accesso dell’utente all’area contenente tutte le notifiche del proprio calendario.

**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.

semestre è un attributo che può assumere 3 valori, 0 se l’insegnamento è svolto durante il primo semestre, 1 se è svolto durante il secondo semestre, 2 se viene svolto durante entrambi i semestri.

**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.

ogni lezione è inoltre descritta da due date: ‘data inizio’ e ‘data fine’, queste due date servono a descrivere il periodo in cui quella determinata lezione è svolta, questo permette, per esempio, ad un corso di avere una determinata lezione il giovedì per metà semestre, e di spostarla al martedì per il resto del semestre.

**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.

**Notifiche** : questa tabella conserva le informazioni sulle notifiche riguardo un particolare insegnamento, verranno salvati informazioni come lezioni annullate, aggiunte. ogni notifica sarà associata all’insegnamento a cui fa riferimento.

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.

NOTE SU ATTRIBUTI:

**Giorno settimana** : rappresenta il giorno della settimana dal lunedì al venerdì, abbiamo utilizzato un intero che può andare da 0 a 5.

**Durata**: rappresenta il numero di slot che la lezione occupa ( ricordando che gli slot sono di 15 minuti ), se quindi una lezione inizia alle 9:30 e ‘durata’ ha valore 8 la lezione finirà alle 11:30

### Database Design Decisions Description

#### Notifiche

Abbiamo deciso di associare le notifiche all’insegnamento relativo, invece che associarlo all’utente.

La prima soluzione aveva 2 lati negativi rilevanti:

* inviare una notifica a 2.000 studenti richiede la creazione di 2.001 nuove entry nel DB (una per la notifica e 2.000 per la relativa associazione agli utenti). Con la nuova soluzione abbiamo ridotto notevolmente la complessità computazionale riducendo alla creazione di 1 sola entry.
* se uno studente decidesse di modificare il proprio calendario in un momento qualsiasi durante il semestre dell’insegnamento, ( ad esempio decidendo di cambiare corso a scelta o riniziare a seguire un corso precedentemente abbandonato ) non avrebbe avuto modo di ricevere le notifiche precedenti riguardanti i nuovi corsi che voleva seguire. Associando invece le notifiche all’insegnamento corrispondente l’utente subito dopo aver modificato il proprio calendario potrà accedere a tutti gli avvisi dei nuovi corsi che ha scelto, in questo modo il sistema viene anche semplificato a livello implementativo.

Uno svantaggio riguardante questo cambiamento è che non si ha più modo di salvare le notifiche riguardo gli eventi speciali, il sistema di notifiche si concentrerà solo su modifiche al proprio calendario personale.

Questo è comunque un problema secondario, visto che esistono già altri sistemi che avvisano gli studenti riguardo eventi secondari alle lezioni ( ad esempio il Bot UNIVAQ di Telegram ).

Il problema principale è che non c’è modo di salvare nel database la ***lettura/visualizzazione*** di una notifica da parte di un utente, quindi un utente non potrà marcare una sua notifica come letta e non c’è modo di avvisare l’utente di notifiche non marcate come lette, nel prossimo paragrafo descriverò una soluzione a questo problema.

#### Nuove notifiche non lette

Per permettere agli utenti di essere avvisati in caso di nuove notifiche aggiungiamo un **timestamp** nel database associato ad ogni utente. Il timestamp è associato all’ultimo accesso da parte dello studente alla bacheca delle notifiche, in questo modo per avvisare lo studente sulle notifiche successive basterà controllare la presenza di notifiche con un Timestamp di inserimento più recente rispetto al Timestamp di ultimo accesso alla bacheca delle notifiche.

#### Gestione dei calendari personalizzati

Per gestire questa problematica, avevamo presentato la seguente soluzione:

Creazione di due relazioni (*corsi aggiunti* e *corsi rimossi*) per la 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”.

Il problema di questa soluzione è la complessità implementativa e il costo computazionale, infatti per generare il calendario degli insegnamenti seguiti da uno studente dovevamo accedere agli insegnamenti obbligatori legati al corso di laurea scelto dallo studente, successivamente rimuovere da questi ultimi i corsi all’interno della tabella *corsi rimossi* e poi aggiungere quelli all’interno della tabella *corsi aggiunti*, effettuando così l’accesso a 3 tabelle.

Abbiamo per questo deciso di sostituire queste relazioni con un’unica relazione:

*corsi seguiti*.

La tabella Corsi Seguiti associa uno studente a tutti i corsi che segue in quel semestre, questo riduce la complessità del sistema, per ottenere i corsi seguiti da uno studente basterà accedere a questa tabella.

Ovviamente se per ognistudente avessimo tutti i dettagli sui corsi seguiti in questa tabella ci sarebbe una notevole ridondanza di informazioni, infatti esistono molti corsi di laurea in cui non sono presenti insegnamenti a scelta o obbligatori, inoltre esistono anche molti corsi di laurea in cui frequentare le lezioni è obbligatorio.

In tutti questi casi salvare le informazioni del calendario personale per tutti questi studenti sarebbe un inutile spreco di memoria.

Abbiamo per questo deciso di associare all’utente la flag ‘*Calendario Modificato*’ .

Durante la creazione del proprio calendario gli studenti, dopo aver selezionato il proprio corso di laurea, sceglieranno fra:

* Accettare il proprio calendario formato da tutti i corsi obbligatori, in questo caso verrà associato valore FALSE alla flag;
* Modificare il proprio corso di laurea aggiungendo o eliminando insegnamenti, in questo caso verrà assegnata alla flag valore TRUE;

A livello applicativo con questa modifica quando un Utente accede al sistema per generare il suo calendario si dovrà prima controllare il valore di questa flag, se ha valore negativo per ottenere i corsi dello studente basterà controllare la tabella Corsi di Laurea per ottenere i corsi seguiti dallo studente, se invece la flag avesse valore positivo per ottenere i corsi seguiti dallo studente dovremo andare ad accedere alla tabella ‘Corsi Seguiti’.

Il problema di questa implementazione e che se uno studente decide di modificare i propri insegnamenti e, per esempio, dei 5 insegnamenti obbligatori in un semestre decide di seguirne solo 4 verranno create 4 entry nella tabella ‘Corsi Seguiti’, abbiamo considerato comunque questi come casi sporadici, quindi non ci aspettiamo di vedere la tabella ‘Corsi Seguiti’ raggiungere dimensioni ingestibili.

#### Utilizzo di una vista materializzata per le lezioni settimanali

Abbiamo deciso di creare una vista materializzata contenente le lezioni che si svolgono all’interno della settimana attuale, all’interno di questa vista ci saranno i dati riguardanti le lezioni della settimana corrente, questa schema verrà riempito a partire dagli Schemi : LEZIONI STRAORDINARIE , LEZIONI, LEZIONI ANNULLATE, e EVENTI SPECIALI.

questi schemi dovranno essere processati per ottenere le informazioni riguardo il calendario della settimana corrente.

Abbiamo deciso di creare questa vista per velocizzare il processo di generazione del calendario e per diminuire il numero di tabelle a cui si deve accedere.

Il problema di questo soluzione e che la vista si dovrà aggiornare ad ogni modifica delle lezioni, per esempio se una lezione viene annullata ( è però da tenere presente che non sono operazioni frequenti ) si dovrà controllare se la lezione fa parte della settimana corrente e se la risposta è positiva si dovrà modificare la entry associata nella vista materializzata, discorso simile si deve fare per le lezioni di recupero e gli eventi speciali.

La vista verrà ricalcolata la domenica a mezzanotte ( un periodo in cui i server non sono sovraccarichi di lavoro ) per la settimana che deve venire.

si potrebbe persino pensare di inserire questa tabella in una cache ( anche se potrebbe rivelarsi operazione impossibile vista le grandi dimensioni di questa tabella ).

**Giorno settimana** contiene un’ numero da 0 a 4 ( 0 per lunedì, 1 per martedì ecc. fino a venerdì )

**flag cancellato** tiene traccia se la lezione è stata annullata ( in quel caso assumerà valore 1 )

#### Raggruppamento dei corsi di laurea

Un’altra questione che abbiamo preso in considerazione è quella dell’anno all’interno del corso di laurea, quindi all’interno della tabella ‘corsi di laurea’ abbiamo dovuto decidere se, per esempio per la triennale in informatica, inserire 3 entry diverse ( primo, secondo e terzo anno ) oppure creare un’unica entry per il corso di laurea per poi modellare gli anni del corso in una seconda tabella associata alla tabella ‘corsi di laurea’.

Abbiamo alla fine deciso che all’interno di un calendario aveva più senso considerarli come 3 entità diverse all’interno della tabella ‘corsi di laurea’ in quanto hanno lezioni e insegnamenti completamente differenti, inoltre avremmo aumentato la complessità delle relazioni con le tabelle ‘utenti’ e ‘insegnamenti’.

abbiamo deciso invece di dividere i corsi di laurea in gruppi, questo per facilitare allo studente la scelta del proprio corso di laurea durante la modifica del proprio calendario, quindi per esempio il gruppo ‘informatica’ conterrà i corsi di laurea triennali e magistrali della laurea in informatica, mentre il gruppo ‘ingegneria dell’informazione’ conterrà i corsi di ingegneria informatica,elettronica automatica ecc…

quindi quando lo studente dovrà scegliere corso di laurea dovrà prima scegliere un dipartimento, dopo una categoria ed infine il corso specifico

#### Dizionario dei termini

avendo deciso di utilizzare Laravel nello sviluppo del sistema abbiamo deciso di seguire il più possibile le sue convenzioni, per questo abbiamo deciso di implementare il database con nomi inglesi sia per il nome della tabella che per gli attributi, i nomi sono comunque intuibili ma riportiamo qui la traduzione per completezza.

***RELATIONSHIPS***

|  |  |
| --- | --- |
| AULE | classrooms |
| CORSI DI LAUREA | degrees |
| DIPARTIMENTO | departments |
| DOCENTI | professors |
| EVENTI SPECIALI | special\_events |
| GRUPPI DI CORSI DI LAUREA | degree\_groups |
| INSEGNAMENTI | teachings |
| LEZIONI | lessons |
| LEZIONI ANNULLATE | canceled\_lessons |
| LEZIONI STRAORDINARIE | extra\_lessons |
| NOTIFICHE | updates |
| UTENTI | users |
| VISTA LEZIONI SETTIMANALI | view\_weekly\_lessons |

per le tabelle pivot verrà seguita la convenzione

( ‘nome prima tabella’\_’nome seconda tabella’ ) dove il nome delle due tabelle devono essere in ordine alfabetico e al singolare ( user\_teaching non va bene, è teaching\_user ).

***ATTRIBUTES***

|  |  |
| --- | --- |
| ACCESSIBILITÀ | accessibility |
| CAPIENZA | capacity |
| SSD | SSD |
| CODICE INSEGNAMENTO | code |
| COGNOME | surname |
| CREDITI | CFU |
| DATA | data\_lesson ( extra\_lessons ), data\_event ( special\_events ) |
| DATA INIZIO | start\_date |
| DATA FINE | end\_date |
| DETTAGLI | info |
| DURATA | duration |
| EDIFICIO | building |
| EMAIL | email |
| FLAG CALENDARIO MODIFICATO | personal\_calendar |
| FLAG ANNULLATA | canceled |
| GIORNO SETTIMANA | week\_day |
| INDICAZIONI | directions |
| INDIRIZZO | office\_address ( professor ) address ( building ) |
| LINGUA | language |
| LINK | link |
| MATRICOLA | matric\_no |
| NOME | name |
| ORARI DI RICEVIMENTO | office\_hours |
| ORARIO INIZIO | start\_time |
| PASSWORD | password |
| PIANO | floor |
| RUOLO | role |
| SEMESTRE | semester |
| TELEFONO | telephone\_no |
| TIMEST. ULTIMO ACC. BACHECA | LAU |
| TIPO | type |
| TITOLO | title |

### 

#### Tabelle aggiuntive

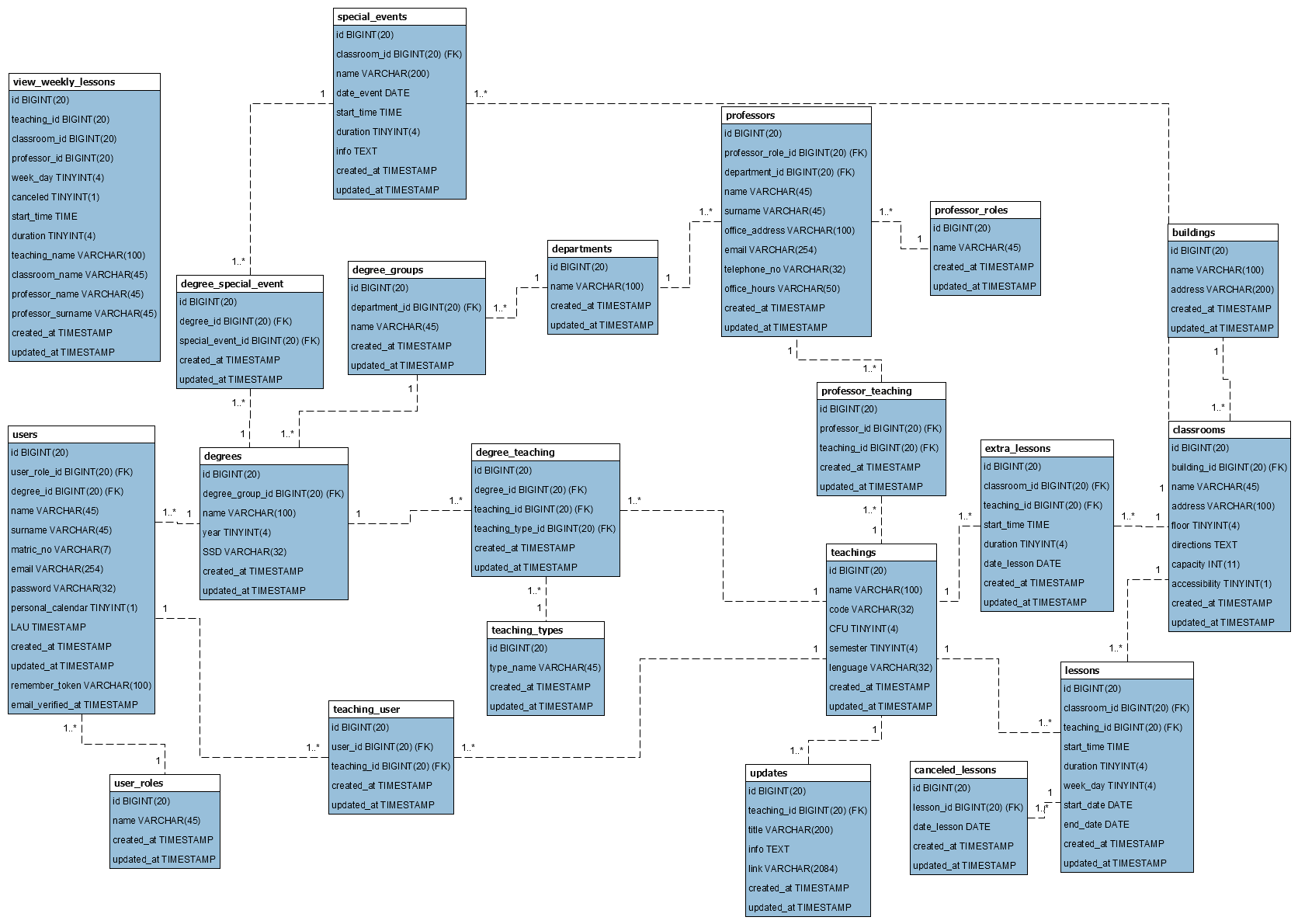
Abbiamo creato 4 nuove tabelle per modellare i dati del sistema:

* “***buildings***” : sarà associata alla tabella classrooms e sarà munito anche dell’indirizzo
* “***professor\_roles***” associata alla tabella professors,
* “***user\_roles***” associata alla tabella user,
* “***teaching\_types***” associata alla tabella pivot degree\_teaching.

Queste 4 tabelle sono state create a partire da l’attributo loro corrispondente nella tabella loro associata per alleggerire il peso delle entry, ora infatti dovremmo memorizzare soltanto l’id dell’entità corrispondente invece che la stringa intera.

*es. la ‘classroom’ C 1.6 non conterrà più al suo interno la stringa “Coppito 2” ma l’id corrispondente alla entry “Coppito 2” nella tabella ‘buildings’*

# Diagramma ER Ristrutturato



### Vista Materializzata

La vista materializzata verrà generata in questo modo:

Dalla tabella ‘lessons’ verranno prese solo quelle tuple in cui l’intersezione tra l’intervallo di tempo formato dai giorni della settimana attuale e l’intervallo formato dalle due date ‘start\_date’ ‘end\_date’ non è vuoto, quindi detto semplicemente verranno prese tutte le lezioni che vengono svolte in questa settimana.

Verrà poi fatta la Join tra la tabella ottenuta e la tabella ‘canceled\_lessons’ per ottenere le lezioni che questa settimana sono state annullate.

ora abbiamo ottenuto le lezioni di questa settimana regolari, dovremmo poi unire questa tabella a quelle di ‘extra\_lessons’ e ‘special\_events’ ( queste tabelle dovranno però prima essere manipolate per ottenere solo gli eventi e le lezioni di questa settimana ).

questa operazione avverrà ogni settimana, la vista verrà poi modificata durante la settimana tramite trigger ( per esempio quando viene annullata una lezione della settimana corrente )

NOTA:

Le variabili **start\_time** ( presenti in più tabelle ) rappresentano l’orario di inizio della lezione/evento, per ottimizzare al livello del database ( visto che abbiamo previsto slot di un quarto d’ora ) invece di salvare queste variabili come timestamp verranno salvati come TINYINT, dove il valore della variabile rappresenta il numero di quarti d’ora a partire dalle 00:00, quindi per esempio una lezione che inizia alle 8:30 avrà start\_time inizializzato a 34, una che inizia alle 12:30 avrà valore 50 ecc...

E. Design Decisions

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.

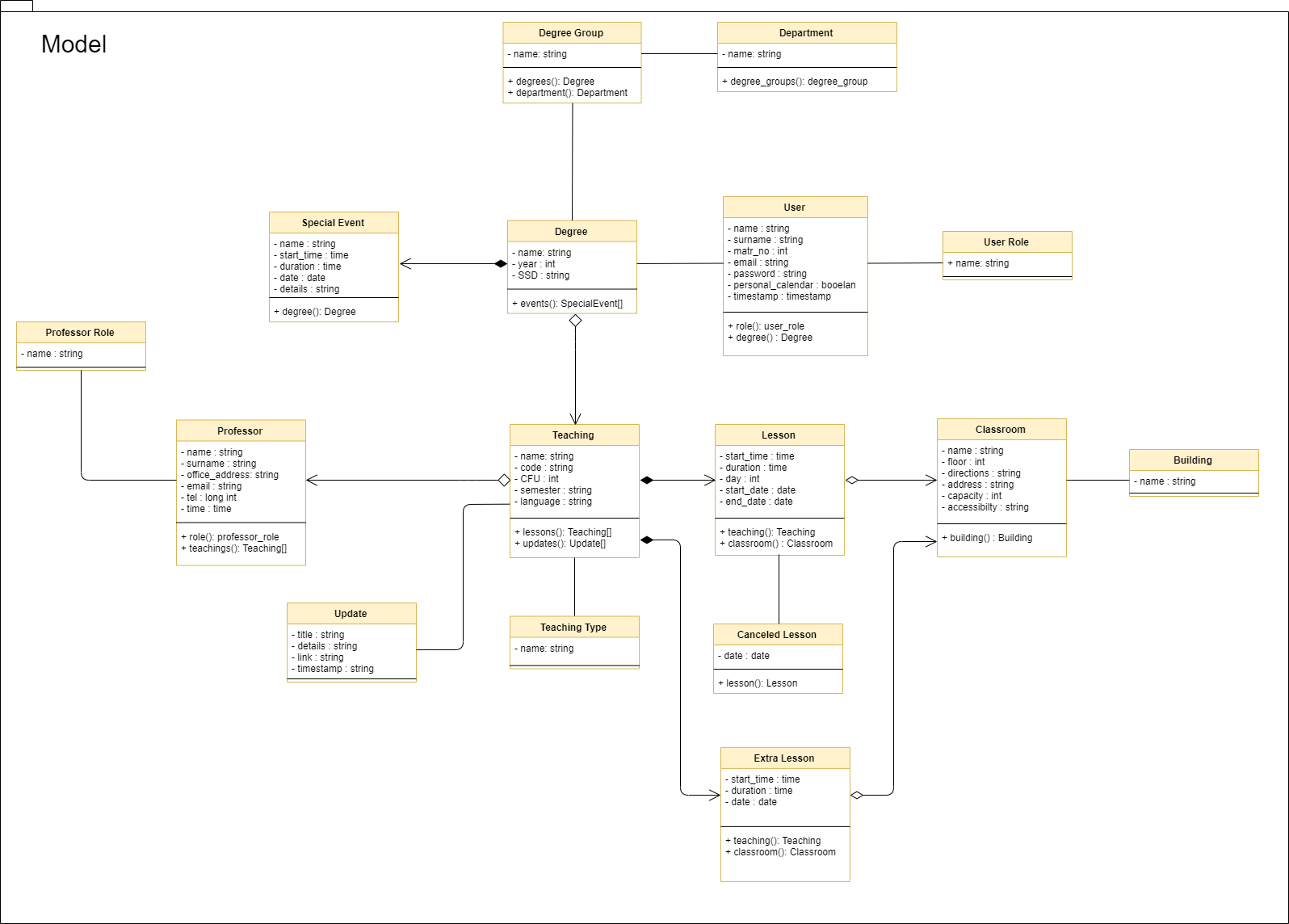
1. **Un insegnamento è singolare per ogni corso di laurea?**

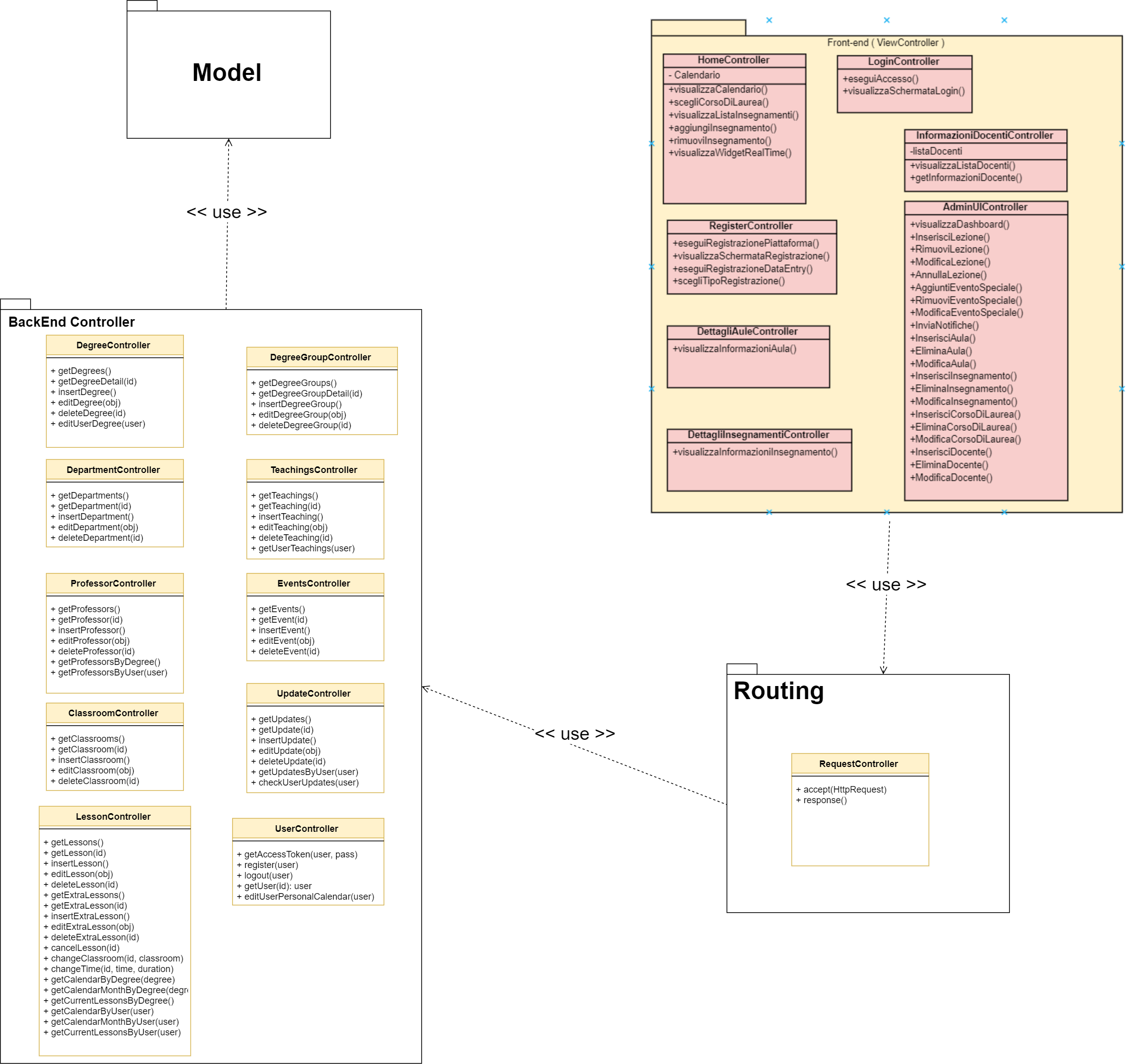
Ci sono degli insegnamenti che sono comuni a più corsi di laurea.

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

Abbiamo deciso di realizzare un Class Diagram a parte per il Model vista la sua complessità.





Nel Class Diagram sono stati omessi i metodi del Model in quanto non sappiamo ancora precisamente quali ci occorrono. Inoltre sono stati omessi i dettagli di backend poiché verranno definiti successivamente in quanto dipendono anche dall’implementazione.

La componente di Routing che utilizzeremo sarà quella predefinita nel framework Laravel. Per darne una simbolica rappresentazione abbiamo soltanto inserito una classe con metodo di ricezione richiesta e invio risposta all’interno del package **Routing**.

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

**Requisiti funzionali**

Link documentazione API Postman : <https://documenter.getpostman.com/view/9807452/SWLYAqeX>

Per quanto riguarda i requisiti funzionali di visualizzazione del calendario e del widget real time saranno esposte le seguenti URL:

* *utente anonimo*:
  + GET {{baseUrl}}/degrees/{{id\_degree}}/calendar
  + GET {{baseUrl}}/degrees/{{id\_degree}}/calendar/month
  + GET {{baseUrl}}/degrees/{{id\_degree}}/current
* *utente loggato*:
  + GET {{baseUrl}}/users/{{id\_user}}/calendar
  + GET {{baseUrl}}/users/{{id\_user}}/calendar/month
  + GET {{baseUrl}}/users/{{id\_user}}/current

Riguardo autenticazioni sono state progettate le seguenti rotte:

* POST {{baseUrl}}/users
* POST {{baseUrl}}/users/{id\_user}/accessToken
* POST {{baseUrl}}/users/{{id\_user}}/logout

Per la creazione / modifica del calendario:

* *utente di ogni tipo*:
  + GET {{baseUrl}}/degrees
  + GET {{baseUrl}}/teachings?page
  + GET {{baseUrl}}/degrees/{{id\_degree}}/teachings
* *utente loggato*:
  + PATCH {{baseUrl}}/users/{{id\_user}}?id\_degree
  + PATCH {{baseUrl}}/users/{{id\_user}}?personal\_calendar=true
  + GET {{baseUrl}}/users/{{id\_user}}/reset\_calendar
  + GET {{baseUrl}}/users/{{id\_user}}/current

Riguardo tutti i requisiti funzionali per quanto riguarda l’amministrazione:

* Tutte le classiche rotte per effettuare le operazioni CRUD sulle entità
  + es. GET/POST/PUT/DELETE {{baseUrl}}/degrees
* POST {{baseUrl}}/lessons/{{id\_lesson}}/cancel
* PATCH {{baseUrl}}/lessons/{{id\_lesson}}/classroom
* PATCH {{baseUrl}}/lessons/{{id\_lesson}}?start\_time&duration

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.

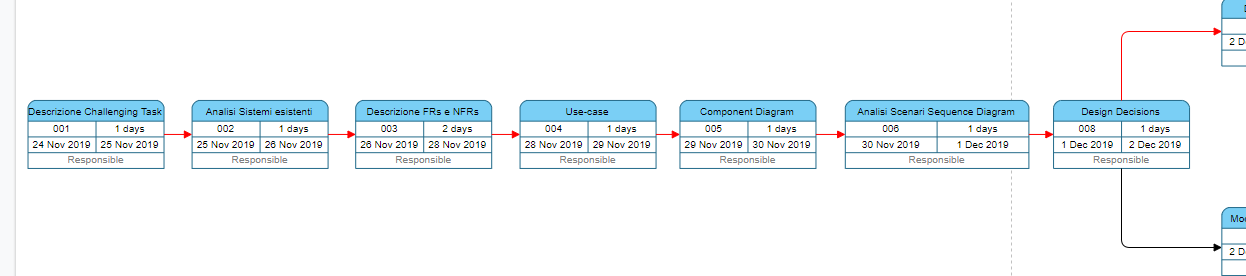
Riguardo tutti i requisiti funzionali per quanto riguarda l’amministrazione è stato modellato tutto tramite ***AdminUIController*** che si occupa di gestire tutte queste funzionalità.

**Requisiti Non Funzionali**

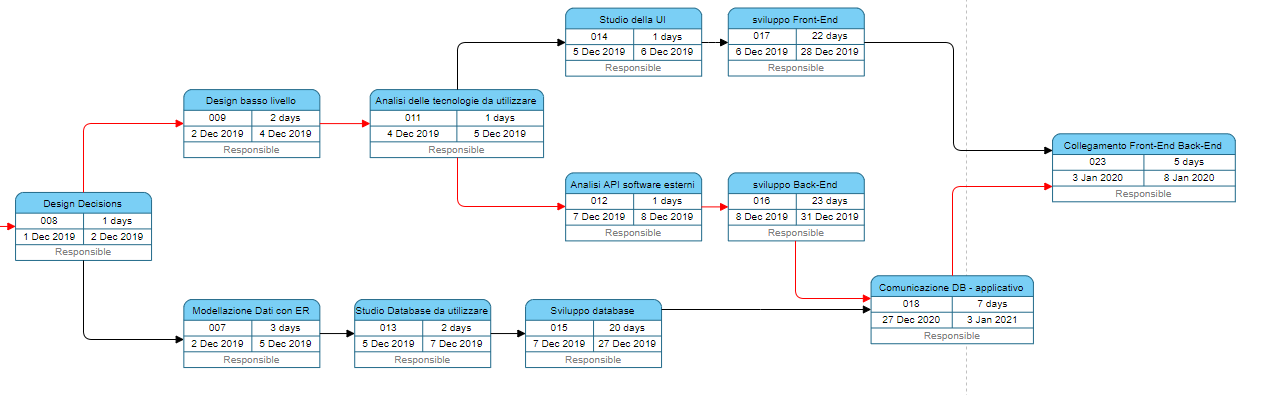
|  |  |
| --- | --- |
| ***Requisito*** | **Descrizione** |
| ***Simplicity / Usability*** | Il layout è progettato in maniera tale che il calendario sia raggiungibile facilmente (essendo situato nella homepage) e tutte le informazioni che l’utente deve inserire nel proprio calendario vengono richieste 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.

**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>*