0.1 Implementazione

0.1.1 Application Server

Web Layer

È il livello che espone le funzionalità offerte dal sistema verso il Client. In particolare, vi sono implementati:

- 2 Filtri derivati dalla classe `Filter`:
  - `AuthFilter` filtro per il controllo dell'autenticazione dell'utente all'interno del sistema;
  - `DocFilter` gestisce l'autirizzazione per le funzioni dedicate al docente.
- 16 Servlet per identificare le risorse. Ogni Servlet è una classe derivata da `HttpServlet`. Il flusso di esecuzione di una servlet è incentrato su due componenti fondamentali: la richiesta (request, inviata dal client verso il server) e la risposta (response, inviata dal server verso il client). In Java, questi due componenti sono identificati, rispettivamente, dalle seguenti interfacce: `javax.servlet.http.HttpServletRequest` e `javax.servlet.http.HttpServletResponse`.

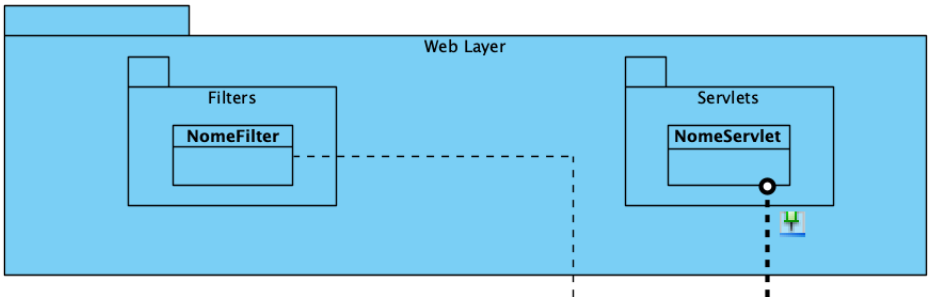


Figura 1: Package Web Layer

Service Layer

Nel Service Layer troviamo l'implementazione della logica di business. Tale package è composto da:

- Un package per ogni servizio offerto al cui interno troviamo:
  - Un'interfaccia con la definizione dei metodi relativi ai servizi;
  - Un package di implementazione dentro il quale vi è la classe di implementazione dei metodi definiti nell'interfaccia relativi a quel servizio.

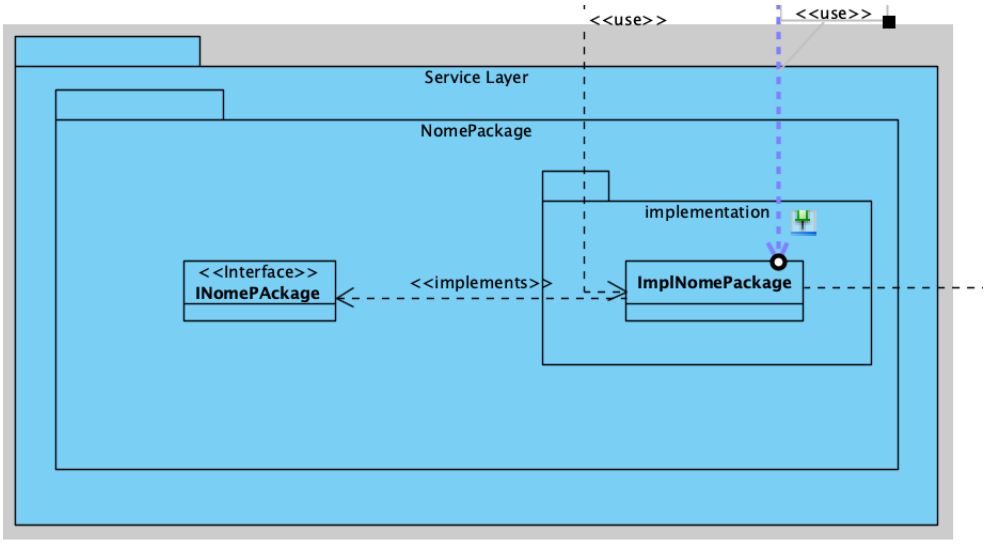


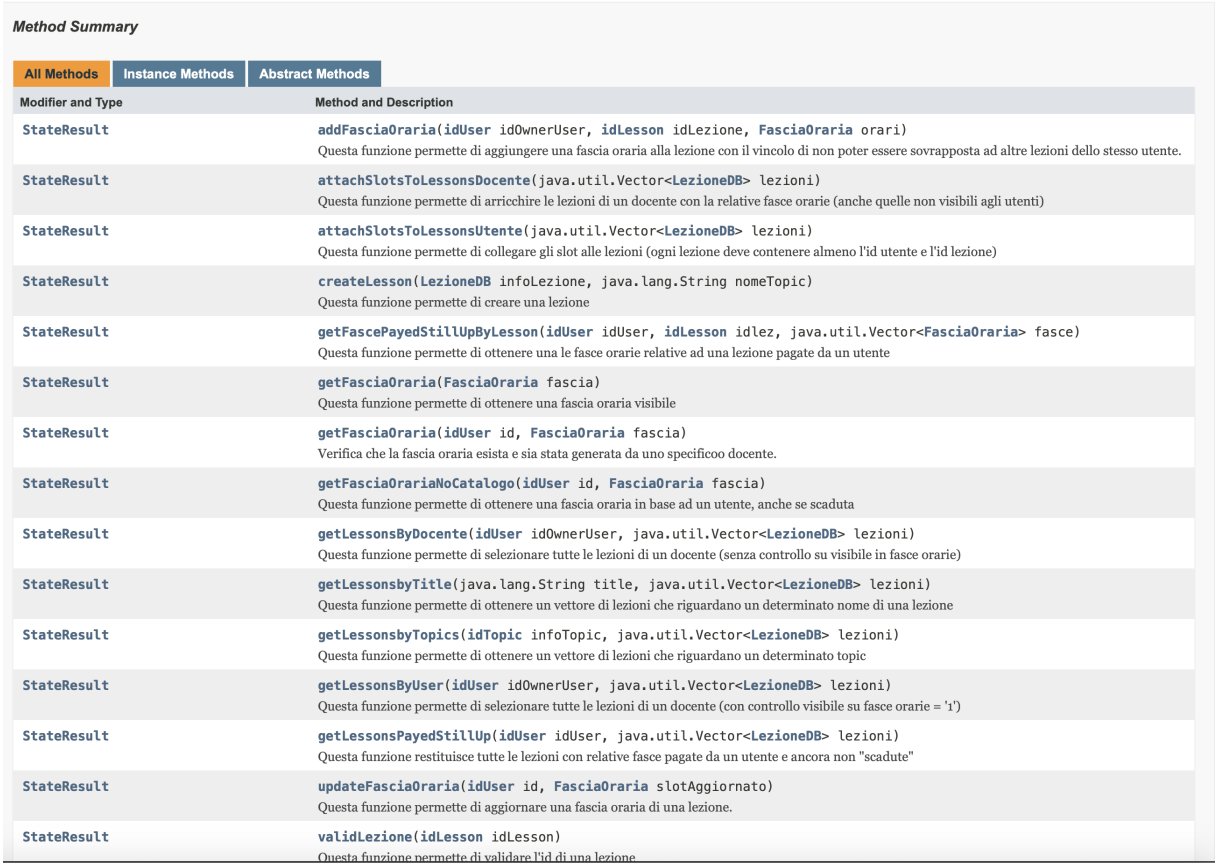
Figura 2: Package Service Layer

Data Layer

Il package Data Layer è composto da:

- Un package che prende il nome dell'entità da implementare al cui interno troviamo:
  - Il package "entities" nel quale vi è la classe che funge da contenitore d'informazioni dell'entità;
  - Il package "controller" nel quale vi è il controller, classe che implementa l'interfaccia che espone le funzioni pubbliche sull'entità;
  - L'interfaccia della classe con la definizione dei metodi relativi all'entità, i quali possono essere richiamati dal Service Layer. È possibile consultare la documentazione di tutte le interfacce, generata tramite Javadoc, all'interno della cartella "Codice\WebAppSmartLearning\doc". Ogni interfaccia segue il seguente standard:
    * ogni funzione ritorna sempre uno "StateResult" che rispecchia lo stato di completamento della funzione;
    * le funzioni utilizzano come contenitori di informazioni solo le entità presenti nel relativo package "entities";
    * le informazioni ottenute dall'elaborazioni delle funzioni sono restituiti tramite parametri di I/O.

    Nella seguente figura è mostrata come esempio la documentazione Javadoc relativa a "API_LezioneDB":



Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
StateResult	addFasciaOraria(idUser idOwnerUser, idLesson idLezione, FasciaOraria orari) Questa funzione permette di aggiungere una fascia oraria alla lezione con il vincolo di non poter essere sovrapposta ad altre lezioni dello stesso utente.
StateResult	attachSlotsToLessonsDocente(java.util.Vector<LezioneDB> lezioni) Questa funzione permette di arricchire le lezioni di un docente con la relative fasce orarie (anche quelle non visibili agli utenti)
StateResult	attachSlotsToLessonsUtente(java.util.Vector<LezioneDB> lezioni) Questa funzione permette di collegare gli slot alle lezioni (ogni lezione deve contenere almeno l'id utente e l'id lezione)
StateResult	createLesson(LezioneDB infoLezione, java.lang.String nomeTopic) Questa funzione permette di creare una lezione
StateResult	getFascePayedStillUpByLesson(idUser idUser, idLesson idlez, java.util.Vector<FasciaOraria> fasce) Questa funzione permette di ottenere una le fasce orarie relative ad una lezione pagate da un utente
StateResult	getFasciaOraria(FasciaOraria fascia) Questa funzione permette di ottenere una fascia oraria visibile
StateResult	getFasciaOraria(idUser id, FasciaOraria fascia) Verifica che la fascia oraria esista e sia stata generata da uno specificoo docente.
StateResult	getFasciaOrariaNoCatalogo(idUser id, FasciaOraria fascia) Questa funzione permette di ottenere una fascia oraria in base ad un utente, anche se scaduta
StateResult	getLessonsByDocente(idUser idOwnerUser, java.util.Vector<LezioneDB> lezioni) Questa funzione permette di selezionare tutte le lezioni di un docente (senza controllo su visibile in fasce orarie)
StateResult	getLessonsbyTitle(java.lang.String title, java.util.Vector<LezioneDB> lezioni) Questa funzione permette di ottenere un vettore di lezioni che riguardano un determinato nome di una lezione
StateResult	getLessonsbyTopics(idTopic infoTopic, java.util.Vector<LezioneDB> lezioni) Questa funzione permette di ottenere un vettore di lezioni che riguardano un determinato topic
StateResult	getLessonsByUser(idUser idOwnerUser, java.util.Vector<LezioneDB> lezioni) Questa funzione permette di selezionare tutte le lezioni di un docente (con controllo visibile su fasce orarie = '1')
StateResult	getLessonsPayedStillUp(idUser idUser, java.util.Vector<LezioneDB> lezioni) Questa funzione restituisce tutte le lezioni con relative fasce pagate da un utente e ancora non "scadute"
StateResult	updateFasciaOraria(idUser id, FasciaOraria slotAggiornato) Questa funzione permette di aggiornare una fascia oraria di una lezione.
StateResult	validLezione(idLesson idLesson) Questa funzione permette di validare l'id di una lezione

Figura 3: Documentazione Javadoc API_LezioneDB

- Il package "ConnectionManager" in cui troviamo la classe che implementa le funzioni per la connessione con il database e tutte le funzioni che effettuano le query al database;

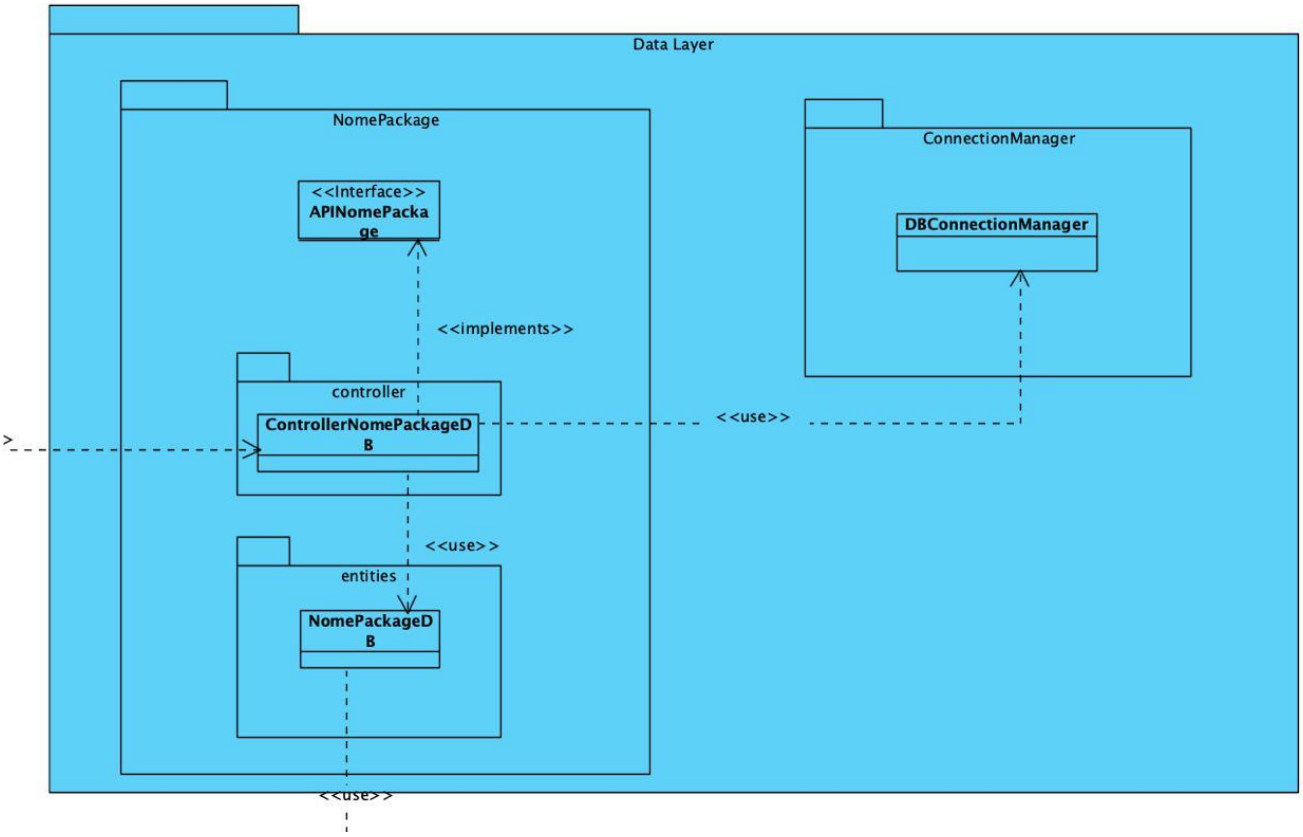


Figura 4: Package Data Layer

Utilities

Nel package "utilities" si trovano le classi relative ai vari id, che si riferiscono alle varie tabelle del database, e una classe `enum` "StateResult" che rappresenta lo stato di fine elaborazione di una funzione.

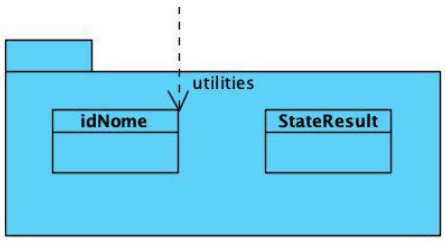


Figura 5: Package Utilities

0.1.2 Database

Il Database è di tipo **relazionale** ed è rappresentabile attraverso il seguente **modello E-R** (entità-relazione):

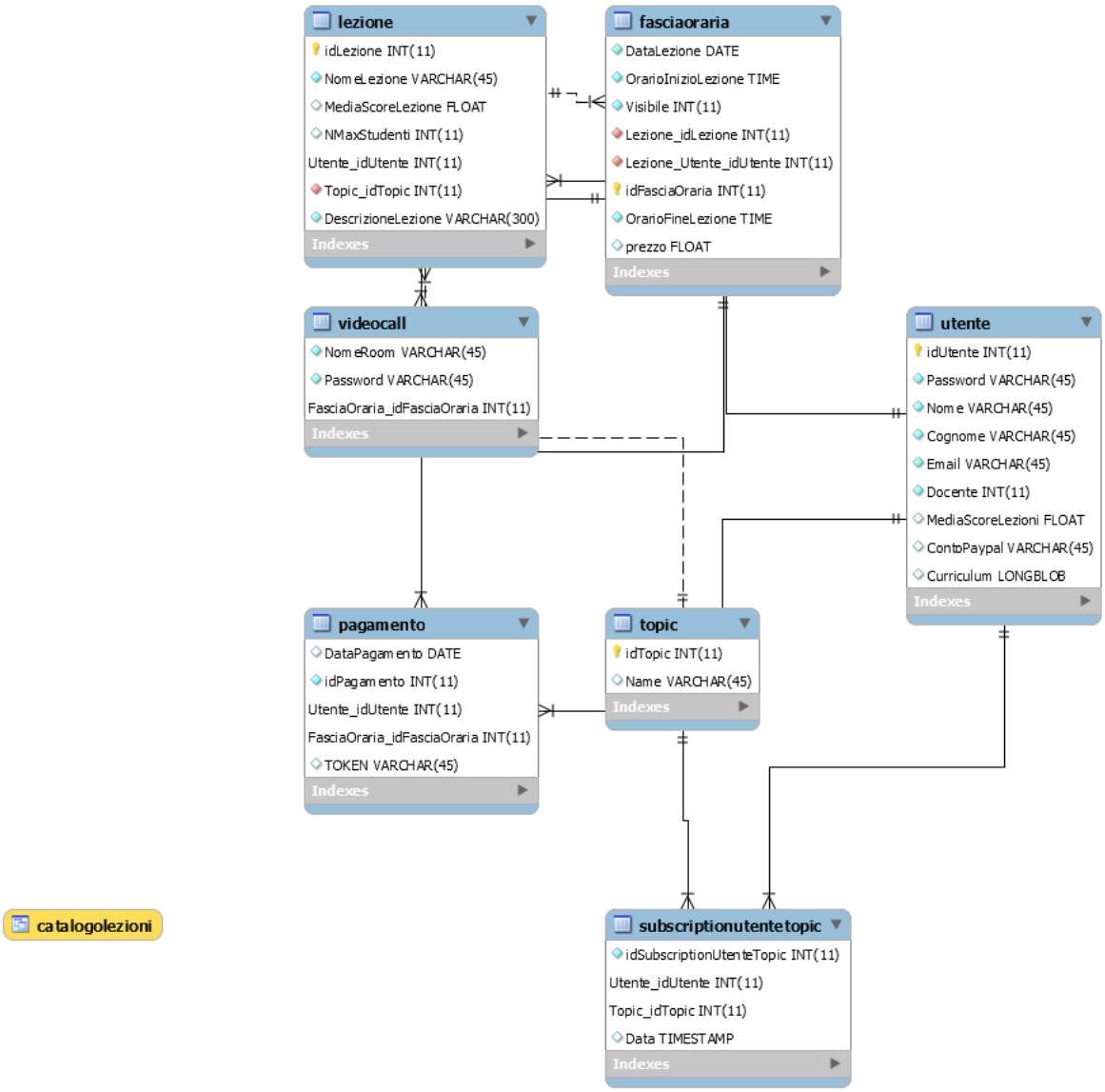
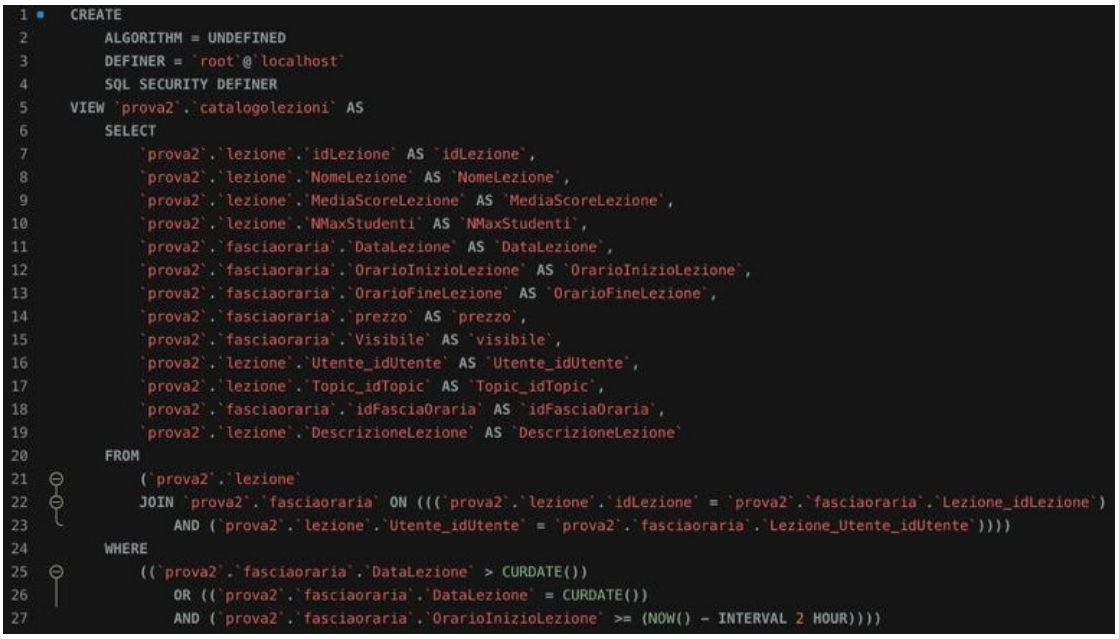


Figura 6: Modello E-R

Esso, inoltre, è caratterizzato dall'utilizzo di una vista denominata "catalogolezione", e delle stored procedures. In particolare:

- "catalogolezione" raggruppa i dati della tabella "lezione" e di "fasciaoraria" in modo da poter effettuare i controlli sulla data e sugli orari per il retrieve di una fascia oraria di una lezione non scaduta. La data di creazione deve essere la data corrente oppure maggiore della data corrente, l'orario di inizio di una lezione deve essere maggiore o uguale dell"orario corrente;



```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `prova2`.`catalogolezioni` AS
    SELECT
        `prova2`.`lezione`.`idLezione` AS `idLezione`,
        `prova2`.`lezione`.`NomeLezione` AS `NomeLezione`,
        `prova2`.`lezione`.`MediaScoreLezione` AS `MediaScoreLezione`,
        `prova2`.`lezione`.`NMaxStudenti` AS `NMaxStudenti`,
        `prova2`.`fasciaoraria`.`DataLezione` AS `DataLezione`,
        `prova2`.`fasciaoraria`.`OrarioInizioLezione` AS `OrarioInizioLezione`,
        `prova2`.`fasciaoraria`.`OrarioFineLezione` AS `OrarioFineLezione`,
        `prova2`.`fasciaoraria`.`prezzo` AS `prezzo`,
        `prova2`.`fasciaoraria`.`Visibile` AS `visibile`,
        `prova2`.`lezione`.`Utente_idUtente` AS `Utente_idUtente`,
        `prova2`.`lezione`.`Topic_idTopic` AS `Topic_idTopic`,
        `prova2`.`fasciaoraria`.`idFasciaOraria` AS `idFasciaOraria`,
        `prova2`.`lezione`.`DescrizioneLezione` AS `DescrizioneLezione`
    FROM
        (`prova2`.`lezione`
        JOIN `prova2`.`fasciaoraria` ON (((`prova2`.`lezione`.`idLezione` = `prova2`.`fasciaoraria`.`Lezione_idLezione`)
            AND (`prova2`.`lezione`.`Utente_idUtente` = `prova2`.`fasciaoraria`.`Lezione_Utente_idUtente`))))
    WHERE
        ((`prova2`.`fasciaoraria`.`DataLezione` > CURDATE())
            OR ((`prova2`.`fasciaoraria`.`DataLezione` = CURDATE())
            AND (`prova2`.`fasciaoraria`.`OrarioInizioLezione` >= (NOW() - INTERVAL 2 HOUR))))
```

Figura 7: Vista “catalogolezione” (MySQL)

- la procedura "updateCond" permette di aggiornare una fascia oraria di una lezione e la data di una lezione effettuando un controllo sulla nuova fascia oraria rispetto alla precedente evitando che si incrocino;
- la procedura "inserisciVideoCall" inserisce una nuova room con un id di fascia oraria e un token docente generato in modo randomico;
- la procedura "inserisciCond" permette di inserire una nuova fascia oraria ad una lezione creata evitando che il docente inserisca una fascia oraria che si incrocia con una già creata;
- la procedura "genTokens" crea un token in modo randomico per ogni prenotazione per ogni studente pagante per una fascia oraria.

0.2 Implementazione Client ed Esecuzione

Per l’esecuzione dell’applicazione web è necessario collegarsi all’indirizzo: `https://webappsmartlearning.azurewebsites`

Per l’interfacciamento con il client si sono sviluppate diverse pagine html, in particolare:

- indexSmartLearning.html: pagina di login;
- RegistrationPage.html: pagina di registrazione;
- Homepage.html: homepage (ricerca lezioni);
- ProfiloPage.html: pagina del profilo, contenente informazioni sull’utente, le lezioni alle quali è prenotato, le proprie lezioni in caso di Docente. Su questa pagina è inoltre possibile effettuare le videocall.

Inoltre si è deciso di utilizzare diversi file css per la visualizzazione degli elementi. In particolare, per tutte le pagine elencate si utilizzeranno i css della libreria “bootstrap” così come il file “demo.css”, appositamente sviluppato. L’ultimo css utilizzato è “index.css”, utile ad una corretta visualizzazione della interfaccia di videochiamata.

Oltre ai file html e css, lato client risiederanno anche degli script Javascript, embedded nelle pagine oppure in file separati. Si è deciso di lasciare gli script embedded nelle pagine laddove le funzioni sono specifiche e non riusabili per altre pagine. Tuttavia è possibile trovare 3 file Javascript:

- janus.js: libreria fornita da janus per l’interfacciamento clientside verso il server Janus;

- mainpage.js: file contenente script che utilizzano funzioni contenute in janus.js ed eseguono l'effettiva creazione degli handler e l'esecuzione delle operazioni per l'interazione con il server Janus e la trasmissione/ricezione dei flussi multimediali;
- jsSmartLearning.js: contiene tutte e sole le funzioni che vengono utilizzate da più pagine.

In seguito si spiegheranno, con l'ausilio di alcuni diagrammi, i principali flussi di esecuzione sviluppati.

Login e Registrazione

La prima pagina che verrà visualizzata nel caso in cui non si sia effettuato il login è "indexSmartLearning.html". Come si può notare dalla figura sottostante, tale pagina consente di effettuare l'accesso all'applicazione se si possiedano già le credenziali. In caso di esito positivo, verrà caricata "Homepage.html", ovvero la pagina principale dell'applicazione, dove sarà possibile effettuare la ricerca delle lezioni. Se invece non si possiedono delle credenziali, tramite il testo "registrati ora" sarà possibile caricare la pagina "RegistrationPage.html" che mostra un form attraverso il quale è possibile effettuare la creazione di un account, inoltre sono stati implementati gli opportuni controlli sui dati che si immettono. Dopo aver effettuato la procedura, ogni utente sarà registrato in qualità di studente e solo in un secondo momento, avrà la possibilità di effettuare l'upgrade ad un account docente. In caso di registrazione effettuata correttamente l'utente verrà reindirizzato alla pagina di login.

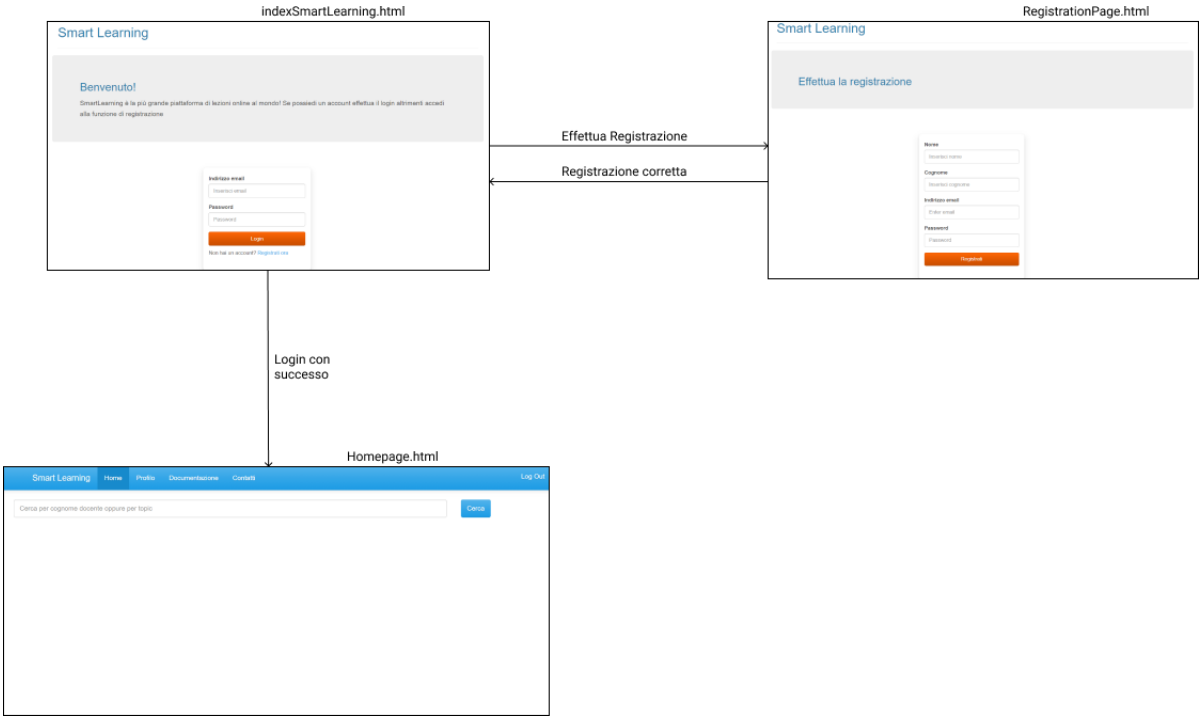


Figura 8: Mockup Login - Registrazione

Interfacce Studente

Effettuato correttamente il login, un utente si ritroverà nella homepage. Con riferimento alla figura sottostante, si fa notare che dalla homepage è possibile effettuare la ricerca di una lezione secondo due criteri:

- Il cognome del docente
- Il topic della lezione

Digitando un qualsiasi valore all'interno della barra di ricerca, questo verrà confrontato dapprima con i cognomi dei docenti presenti nel DB e, in caso di riscontro positivo, verranno visualizzate tutte le lezioni create dal docente, con relative programmazioni. Viceversa, se non vi è un riscontro positivo con un docente, il valore verrà confrontato con

i topic, restituendo tutte le lezioni relative al topic digitato. In caso di esito negativo della ricerca, verrà mostrato un alert.

Se la ricerca ha prodotto almeno un risultato, cliccando su "prenota ora" è possibile vedere tutte le programmazioni di una lezione e prenotarsi.

Cliccando, invece, su "Profilo" nella barra di navigazione principale è possibile raggiungere la pagina "ProfiloPage.html", la quale racchiude la maggior parte delle funzionalità fin ora sviluppate. In questa pagina si avrà a disposizione un'ulteriore barra di navigazione verticale, tramite la quale è possibile accedere a diverse informazioni:

- Al caricamento della pagina verrà mostrato il tab "utente", dove è possibile reperire tutte le informazioni relative all'utente loggato.
- Il tab "Lezioni Sottoscritte", il quale permette di vedere tutte le informazioni relative alle lezioni prenotate. Successivamente, cliccando su "vedi programmazioni" è possibile vedere nello specifico tutte le prenotazioni per una lezione e, se la lezione è stata avviata, prendere parte alla videochiamata cliccando su "partecipa".
- Mentre il tab "Lezioni mie" riguarda funzionalità del docente, approfondite nel paragrafo successivo.

Tornando alla barra principale, se è stato effettuato il login con un account Studente, sarà presente un tasto "Upgrade Docente", attraverso il quale è possibile raggiungere la pagina "upgrade.html". Qui sarà presentato un form all'interno del quale è possibile inserire l'email relativa ad un account PayPal ed il proprio Curriculum Vitae; infatti, queste informazioni sono necessarie per creare un account di tipo Docente.

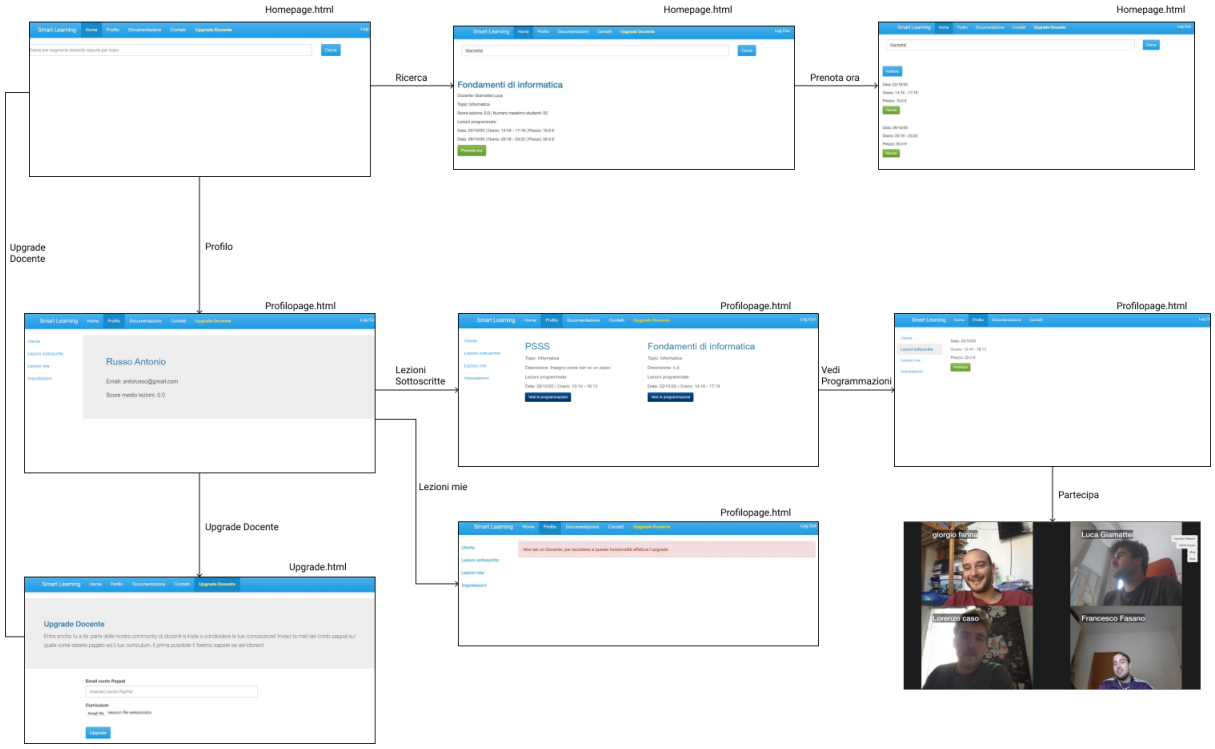


Figura 9: Mockup Studente

Interfacce Docente

L'account docente è un'estensione dell'account Studente e, per questo motivo, include tutte le funzionalità precedentemente illustrate.

Tuttavia il docente avrà accesso alle funzionalità del tab "Lezioni mie" nella pagina "ProfiloPage.html". Dove è possibile vedere le proprie lezioni create con relative programmazioni, oltre a poter creare nuove lezioni e programmazioni. Nella figura sottostante è possibile vedere il form da compilare per la creazione di una nuova lezione, così come quello per la creazione di una nuova programmazione. Aperta la visualizzazione delle programmazioni, nel

caso in cui ci si trovi nell'orario relativo ad una di queste, verrà visualizzato il bottone "avvia videochiamata", che consente la creazione e l'accesso alla VideoRoom relativa alla lezione.

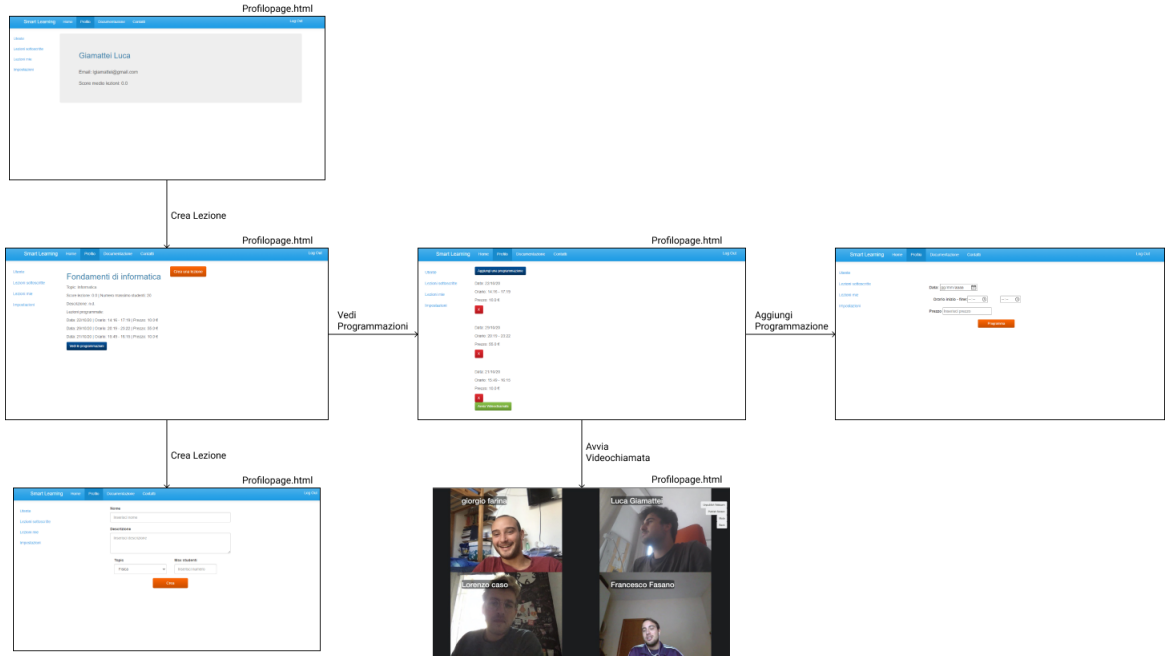


Figura 10: Mockup Docente

0.3 Configurazione

Per eseguire l'applicazione è necessario un web browser, nei successivi esempi è stato adoperato Google Chrome (consigliato).

Per una corretta esecuzione, in particolare delle funzionalità relative a Janus, è necessario effettuare la seguente procedura:

1. Digitare la seguente stringa all'interno della barra degli indirizzi:

   `chrome://flags/#unsafely-treat-insecure-origin-as-secure`

2. Inserire il seguente URL all'interno dell'opzione denominata "**Insecure origins treated as secure**":

   `http://20.49.195.171:8088`

3. Selezionare "Enabled" dal menù corrispondente;

4. Riavviare il browser mediante il comando "Relaunch Now".

Infatti il server Janus è considerato come un'origine insicura, questa procedura permetterà di trattarlo come sicuro ed eseguire la funzionalità di videochiamata.

Il procedimento è mostrato nella seguente figura:

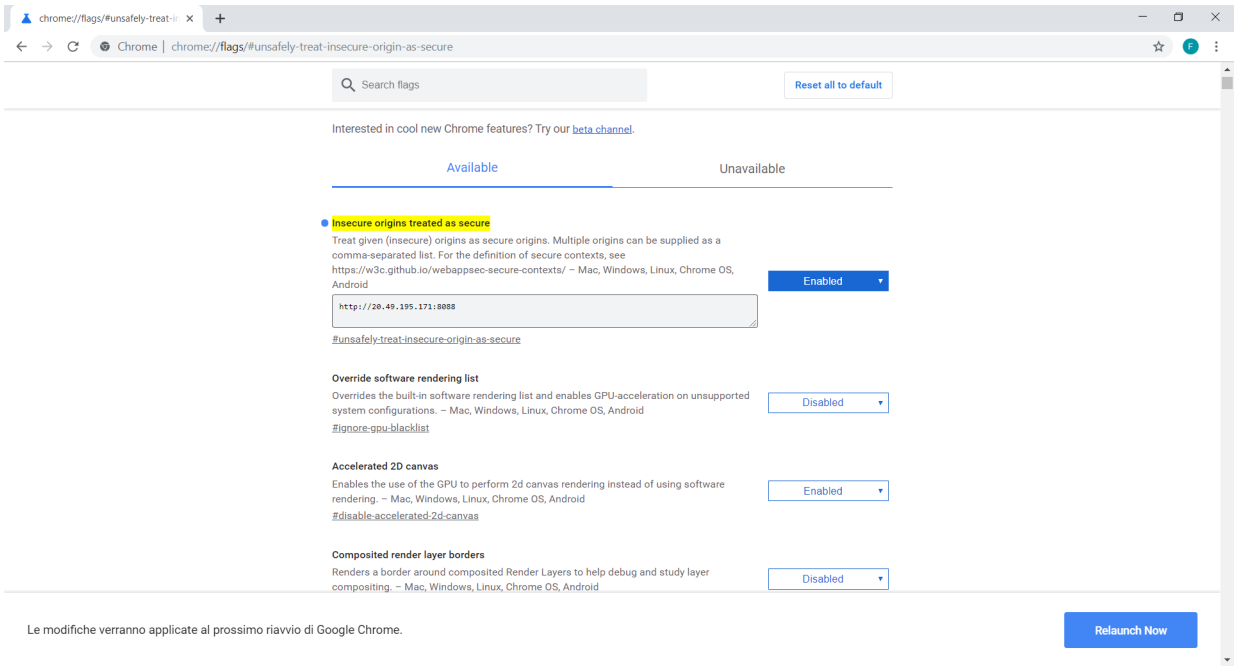


Figura 11: Configurazione Google Chrome