

FindLessons

Tecnologie Web

CHRISTOFER FANÒ

(MAT. 166121)

05/09/2024

Indice

Contents

Indice	1
1 Introduzione	2
1.1 Funzionalità per gli studenti	2
1.2 Funzionalità per gli insegnanti	2
1.3 Registrazione di nuovi utenti	3
1.4 Riepilogo requisiti	3
1.5 Autorizzazioni	4
2 Modelli e distinzioni tra utenti	4
2.1 Distinzione tra utenti	4
2.2 Descrizione delle classi utilizzate	4
2.3 Utilizzo di Signals	5
2.4 Diagramma delle classi	6
3 Tecnologie utilizzate	7
3.1 Front-end	7
3.2 Back-end	7
4 Organizzazione logica dell'applicazione	8
4.1 Main project directory	8
4.2 Applicazione user_creation	8
4.3 Applicazione reservation	8
4.4 Applicazione chat	9
5 Scelte progettuali	9
5.1 Gestione delle richieste di registrazione	9
5.1.1 Cifratura della password nelle richieste	10
5.1.2 Segnalazione esito della richiesta	10
5.1.3 Gestione dello username	10
5.2 Assenza di ricerca per nome	11
6 Unit testing	11
7 Interfaccia e risultato finale	12
7.1 Homepage e presentazione risultati	12
7.2 Pagina profilo dei diversi utenti	13
7.3 Calendario per la gestione degli eventi	15
7.4 Schermata della chat	16
8 Possibili miglioramenti futuri	17

1 Introduzione

La seguente applicazione web è stata realizzata per la ricerca e prenotazione di lezioni private, fornendo un servizio usufruibile sia da **studenti** che da **insegnanti**. È integrato un sistema di valutazione degli insegnanti e una chat per la comunicazione diretta con essi.

1.1 Funzionalità per gli studenti

Gli **studenti** possono visualizzare tutti i professori che insegnano una determinata materia nella città scelta, visualizzando gli insegnanti in ordine decrescente di valutazione o in ordine crescente di prezzo. Dalla schermata di ricerca è possibile filtrare i risultati inserendo una data di inizio e di fine, visualizzando solamente gli insegnanti che hanno registrato una propria disponibilità nel periodo selezionato.

Visualizzando il profilo di un insegnante, lo **studente** può visualizzare le disponibilità di quest'ultimo attraverso un comodo calendario e prenotare una lezione selezionando una delle disponibilità presenti, oppure può iniziare una conversazione con l'insegnante cliccando l'icona della chat.

Nel proprio profilo, lo **studente**, oltre a poter personalizzare le proprie informazioni personali e foto profilo, può visionare tutte le sue lezioni passate e future, può assegnare valutazioni agli insegnanti con cui ha già effettuato una lezione e disdire le lezioni future; può inoltre visualizzare e modificare tutte le valutazioni da lui assegnate agli insegnanti.

1.2 Funzionalità per gli insegnanti

Gli **insegnanti** possono sfruttare la webapp per promuoversi, personalizzando il profilo, inserendo i propri dati personali e informazioni relative alla loro attività. Possono gestire attraverso un comodo calendario le loro disponibilità e lezioni, settando o cambiando l'orario relativo alle loro disponibilità, eliminando o cambiando l'orario relativo alle loro lezioni. Quando una lezione viene prenotata da uno studente viene sostituita automaticamente la disponibilità dell'**insegnante** con la nuova lezione.

Visualizzando il profilo di uno studente, l'**insegnante** può visionare tutte le lezioni da sostenere e già sostenute con quest'ultimo.

1.3 Registrazione di nuovi utenti

La webapp prevede una registrazione differenziata per studenti e insegnanti:

- **studenti**: la registrazione avviene dopo aver scelto username e password;
- **insegnanti**: la registrazione avviene previa approvazione della richiesta da parte dell'admin. La richiesta prevede l'inserimento dei propri dati personali e di una fotografia del documento di identità;

Ad entrambi dopo il primo login comparirà un pop-up per ricordare di completare il proprio profilo con le informazioni mancanti.

1.4 Riepilogo requisiti

- ***ricerca di insegnanti***: basata su città, periodo e materie. Ordinati per valutazione o tariffa oraria
- ***personalizzazione del profilo utente***: cambio password, foto profilo, nome, cognome, email e numero telefonico
- ***personalizzazione profilo insegnante***: inserimento della città, materie insegnate e tariffa oraria
- ***gestione delle prenotazioni***: selezione delle disponibilità attraverso un calendario. Cancellazione o reinserimento delle disponibilità automatico, dopo una prenotazione effettuata o annullata
- ***possibilità di consultare e disdire le proprie lezioni da parte degli insegnanti e studenti***
- ***sistema di notifiche***: per segnalare nuove prenotazioni o la cancellazione di lezioni programmate
- ***sistema di valutazione degli insegnanti effettuabile dai soli studenti***
- ***registrazione degli insegnanti previa approvazione dell'admin, con email che comunica l'esito***
- ***chat diretta tra studenti e insegnanti***

1.5 Autorizzazioni

	Utente anonimo	Studente	Insegnante	Admin
Ricerca insegnante	*	*	*	*
Prenotazione/cancellazione lezione		*	*	*
Modifica orario lezione			*	*
Inserimento disponibilità			*	*
Assegnare valutazione insegnante		*		*
Accesso alla chat		*	*	*
Accettare richieste di registrazione				*

Table 1: Autorizzazioni di ogni tipologia di utente

2 Modelli e distinzioni tra utenti

2.1 Distinzione tra utenti

La distinzione tra utente **studente** e utente **insegnante** avviene attraverso l'assegnazione di gruppi differenti alle due tipologie di utente.

Ad ogni nuova registrazione l'utente deve indicare se si tratta di una registrazione di uno studente o di un insegnante, nel secondo caso viene redirezionato ad un form e per completare la domanda di registrazione deve inserire i propri dati personali e la foto del documento di identità.

Al momento della creazione di un nuovo utente quest'ultimo viene inserito nel gruppo **Students** o **Teachers** a seconda che la registrazione sia avvenuta in maniera diretta o previa richiesta.

Controllando il gruppo di appartenenza è quindi possibile distinguere all'interno delle view il tipo di utente che sta accedendo alla pagina ed eventualmente limitare l'accesso alla stessa solo ad una tipologia di utente specifica.

2.2 Descrizione delle classi utilizzate

Il DBMS utilizzato è **SQLite** e, oltre alla classe **User** di Django, sono stati utilizzati i seguenti modelli:

- **Profile**: ogni utente registrato ha una entry in questa tabella ed è utilizzata per memorizzare le sue informazioni personali
- **Teacher**: memorizza i dati relativi all'attività di un insegnante
- **Notification**: contiene le notifiche di avvenuta prenotazione o cancellazione delle lezioni. Il campo *read*, di default a *False*, permette di segnalare le notifiche non lette nella pagina del profilo utente
- **Availability**: memorizza le disponibilità degli insegnanti
- **Lesson**: contiene i dati relativi alle singole lezioni

- **Rating:** memorizza le valutazioni che gli studenti hanno assegnato ad ogni insegnante. Ogni studente può assegnare ad ogni insegnante al massimo una valutazione
- **Chat:** memorizza l'id della chat e i suoi partecipanti
- **Visibility:** permette la gestione della visibilità della conversazione all'interno della homepage della chat stessa attraverso il campo *visible*, che di default è a *True*. Quando uno dei due partecipanti cancella la conversazione il campo *visible* viene settato a *False*. Se tutti i partecipanti hanno campo *visible* a *False* la chat viene eliminata
- **Message:** contiene le informazioni relative al messaggio, come chat di appartenenza, mittente del messaggio e contenuto. Il campo *read*, che di default è a *False*, permette la segnalazione di nuovi messaggi.
- **Request:** dedicata a contenere le informazioni personali degli insegnanti che hanno effettuato la richiesta di registrazione. Il campo *identification* contiene la foto del documento di identità mentre il campo *encrypted_password* contiene la password scelta dall'insegnante criptata.

2.3 Utilizzo di Signals

All'interno del progetto, vista la relazione diretta tra gli elementi di alcune tabelle, è stato utilizzato **Django Signals** per la creazione e modifica automatica di alcune entry al verificarsi di determinati eventi.

In particolare è stato utilizzato *Signals* per le seguenti funzionalità:

- **creazione automatica del profilo** in seguito alla creazione di un nuovo utente
- **aggiornamento automatico della valutazione dell'insegnante** in seguito all'inserimento, modifica o cancellazione di una sua valutazione nella tabella *Rating* da parte di uno studente
- **creazione automatica di una entry nella tabella *Visibility*** relativa ad ogni profilo aggiunto al campo *participants* della tabella *Chat*

2.4 Diagramma delle classi

Di seguito è riportato un esplicativo diagramma delle classi descritte in precedenza, che evidenzia anche la relazione che unisce le varie tabelle del DB (one-to-one, many-to-one, many-to-many):

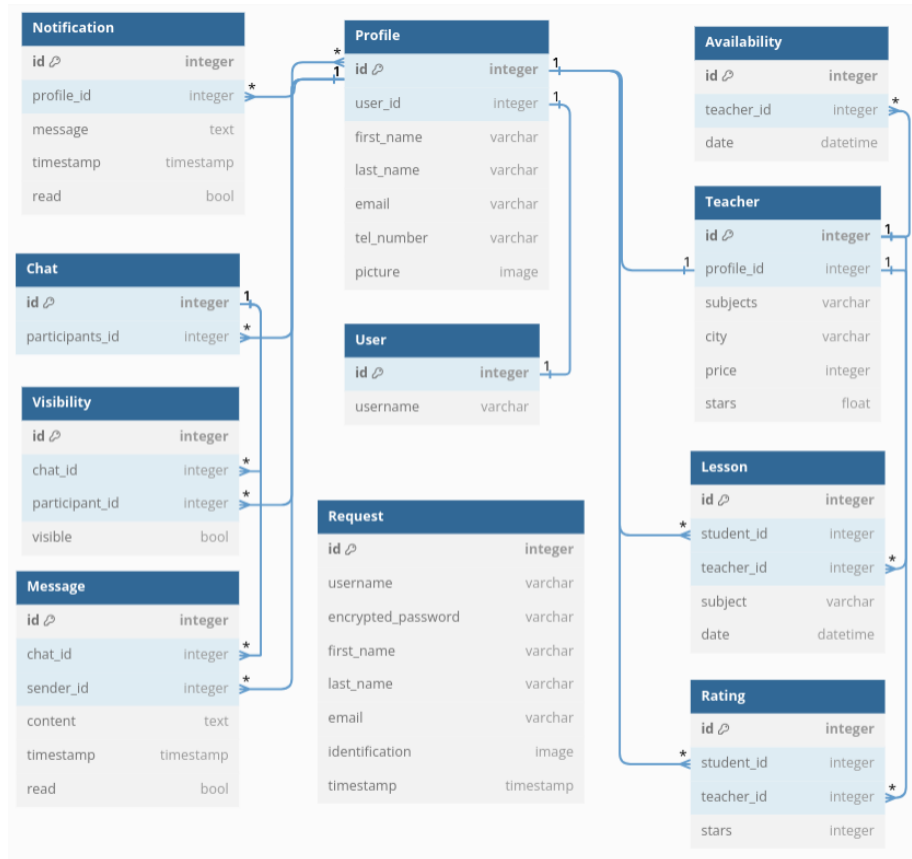


Figure 1: Diagramma delle classi

3 Tecnologie utilizzate

Di seguito sono elencate le tecnologie utilizzati per realizzare il front-end e il back-end del progetto.

3.1 Front-end

Per la realizzazione del front-end sono stati utilizzati principalmente **HTML**, arricchito dall'utilizzo di **CSS** e **Bootstrap4** a cui è stata affiancata anche la libreria **Tempus Dominus** per l'interfaccia grafica di selezione delle date, oltre che **JavaScript** e linguaggio **DTL** per l'interazione con il framework di Django.

In particolare **JavaScript** è stato utilizzato nel sorgente *star_rating.js*, per personalizzare la visualizzazione delle stelle relative alla valutazione di un insegnante espressa come *Float* (stella vuota, piena per un quarto, piena a metà, piena per tre quarti o totalmente piena).

JavaScript è stato inoltre utilizzato per la gestione lato template del WebSocket relativo alla chat, per instaurare la connessione, gestire la ricezione dei messaggi distinguendo messaggi ricevuti da messaggi inviati, la chiusura della connessione e lo scroll automatico del pannello dei messaggi ogni volta che avviene la ricezione di uno nuovo.

3.2 Back-end

Per la realizzazione del back-end sono stati utilizzati i seguenti software:

- **Pillow**: per la gestione delle immagini nel modello *Profile*
- **Python-decouple**: per estrarre dal file *.env* le configurazioni di ambiente come la chiave crittografica e le credenziali di accesso alla mail associata alla webapp, il cui utilizzo verrà approfondito meglio in seguito.
- **Cryptography**: per cifrare e decifrare la password nella tabella *Request*
- **Framework di Django per l'invio di email**: per comunicare l'esito della richiesta di registrazione di un nuovo insegnante
- **WebSocket**: utilizzato per istanziare una comunicazione real-time full-duplex, consente lo scambio di messaggi, in tempo reale tra il client e il server, all'interno della chat
- **Django Channels**: utilizzato per la gestione delle operazioni asincrone, amplia il supporto di Django per protocolli come *WebSocket* e offre una struttura per la gestione della comunicazione real-time
- **Redis**: viene utilizzato come layer di comunicazione tra i diversi processi che gestiscono la messaggistica real-time in *Django Channels*, fungendo da broker per l'invio e la ricezione di messaggi

- **Daphne:** è un server **ASGI** che gestisce le connessioni *HTTP* e *Web-Socket*, e funziona come server front-end per le applicazioni *Django con Channels*

4 Organizzazione logica dell'applicazione

Il progetto *FindLessons* oltre alla *Main project directory* utilizza le tre applicazioni descritte di seguito per suddividere le diverse funzionalità logiche.

4.1 Main project directory

La **Main project directory** contiene le view e i template dedicati alla creazione di nuovi utenti, al cambio di password e alla visualizzazione della schermata home contenente il form per la ricerca dell'insegnante dati materia e città.

4.2 Applicazione user creation

L'app **user.creation** contiene i modelli *Profile*, *Teacher*, *Notification* e *Request* e gestisce la logica relativa alla visualizzazione dei vari profili utente e alla creazione di nuove richieste di registrazione.

Il template *profile.html* è utilizzato come base per i tre template dedicati ai diversi tipi di utente:

- *student_profile.html*
- *teacher_profile.html*
- *admin_profile.html*

Questi template mettono in pratica il principio **DRY** di Django, infatti estendono *profile.html* mantenendo la metà sinistra della pagina, che è dedicata alla visualizzazione dei dati contenuti nel modello *Profile*, pressoché identica, andando a personalizzare a seconda dell'utente la parte destra.

I seguenti template sono inoltre utilizzati anche per la visualizzazione dei profili di utenti diversi da quello loggato, personalizzandone l'aspetto attraverso il flag *is_me* passato attraverso il contesto.

4.3 Applicazione reservation

L'app **reservation** contiene i modelli *Availability*, *Lesson* e *Rating*. Le views di questa applicazione sono dedicate alla visualizzazione dei risultati della ricerca del professore, del filtraggio per data e ordinamento per prezzo o valutazione; si occupa inoltre della creazione, eliminazione e modifica di nuove disponibilità, lezioni e valutazioni.

Le lezioni e le disponibilità possono essere gestite dagli insegnanti attraverso un comodo **calendario** realizzato passando al template una struttura dati contenente le tre settimane successive a partire dal lunedì della settimana corrente. Ogni giorno della settimana contiene le disponibilità e le lezioni dell'insegnante, anche in questo caso, rispettando il principio **DRY**, attraverso un flag *is_me* vengono personalizzate le azioni che possono essere compiute dall'utente che visualizza il calendario.

La view che gestisce l'eliminazione di una lezione permette di modificarne il comportamento attraverso il parametro *action* passato nell'url, questo parametro può avere valore *"reset"* o *"noreset"* per, rispettivamente, resettare o meno le disponibilità dell'insegnante.

Quando è uno **studente** a disdire la lezione viene passata come azione *"reset"*, mentre quando è l'insegnante a disdire la lezione gli viene data la possibilità di scegliere se resettarla o meno.

4.4 Applicazione chat

Questa app contiene le definizioni dei modelli *Chat*, *Visibility* e *Message* e gestisce la logica relativa alla gestione della chat.

In *routing.py* viene definito l'url da utilizzare per connettersi ad una chat attraverso l'utilizzo di WebSocket, in *consumers.py* vengono invece gestite le funzionalità real-time della chat sincronizzando la ricezione dei messaggi e l'eliminazione di una chat con le relative tabelle del database.

Il template *chat.html* è l'unico utilizzato dall'applicazione e permette di stabilire la connessione WebSocket, di gestire l'inoltro e la ricezione real-time di messaggi e la disconnessione dal server.

5 Scelte progettuali

Di seguito sono spiegate e motivate le due principali scelte progettuali .

5.1 Gestione delle richieste di registrazione

Per fornire un servizio migliore, sia lato studente che lato insegnante, si è scelto di permettere la registrazione di nuovi insegnanti previa richiesta, al fine di riconoscerne l'identità.

Questa funzionalità è stata realizzata attraverso la personalizzazione del form di login, nel quale, oltre all'inserimento dello *username* e *password* è necessario indicare se si vuole registrare uno studente o un insegnante. Nel secondo caso si viene redirezionati verso un nuovo form in cui è richiesta la compilazione dei campi relativi al proprio *nome*, *cognome*, *email* e l'inserimento di una fotografia

del *documento di identità*.

5.1.1 Cifratura della password nelle richieste

Per memorizzare le informazioni sopracitate nella tabella **Request**, è stato necessario gestire la presenza del campo *password* e la sua memorizzazione cifrata. Django gestisce le password nella tabella **User** memorizzando il loro **hash** invece che la password in chiaro, per ovvi motivi di sicurezza, e fornisce dei metodi per generare l'hash a partire da stringhe.

Questa soluzione non era tuttavia percorribile per la tabella **Request** in quanto è necessario utilizzare una funzione reversibile per ottenere nuovamente la password in chiaro, da passare all'oggetto della tabella **User** al momento della creazione di un nuovo utente.

Per questo motivo si è deciso di utilizzare il pacchetto Python **cryptography** che fornisce l'algoritmo di cifratura simmetrica **Fernet**. La chiave di cifratura è memorizzata e tenuta al sicuro in un file **.env**.

5.1.2 Segnalazione esito della richiesta

Le richieste possono essere visionate, accettate o rifiutate attraverso la schermata del profilo dell'**admin**.

Per la segnalazione dell'esito della richiesta si è deciso di utilizzare il framework messo a disposizione da Django per l'invio delle email.

Si è creata l'email **findlessons2024@gmail.com** dedicata interamente a questo scopo, attivando l'autenticazione a due fattori di Google e generando una password apposita per la webapp *FindLessons*. Le configurazioni relative all'email sono state memorizzate all'interno del file **.env**.

5.1.3 Gestione dello username

Per garantire che lo **username** scelto dall'insegnante al momento della richiesta di registrazione sia ancora disponibile nel caso quest'ultima venisse accettata, si è scelto di creare per ogni richiesta un utente con lo **username** scelto ma con **password randomica**.

In caso di accettazione della richiesta la password viene settata a quella scelta dall'utente, in caso di rifiuto invece l'utente viene eliminato.

5.2 Assenza di ricerca per nome

Si è deciso di non fornire la possibilità di ricercare insegnanti e studenti per nome in quanto, sia nel profilo insegnante che in quello studente, è possibile visualizzare le lezioni passate e future e attraverso esse visitare, rispettivamente, il profilo dello studente e quello dell'insegnante.

Per iniziare una chat con un nuovo utente è sufficiente quindi visitare un server ASGI (Asynchronous Server Gateway Interface) che gestisce le connessioni HTTP e WebSocket, e funziona come server front-end per le applicazioni Django con Channels. il suo profilo e cliccare sull'icona della chat.

6 Unit testing

Lo **unit testing** è effettuato all'interno dell'applicazione *chat*, in essa sono testati i tutti i modelli e le loro funzioni, tutte le view e il template *chat.html*.

7 Interfaccia e risultato finale

In ogni pagina della webapp è presente una navbar attraverso la quale è possibile ritornare alla pagina home cliccando su **FindLessons** oppure alla pagina del proprio profilo cliccando sulla propria **immagine profilo**.

7.1 Homepage e presentazione risultati

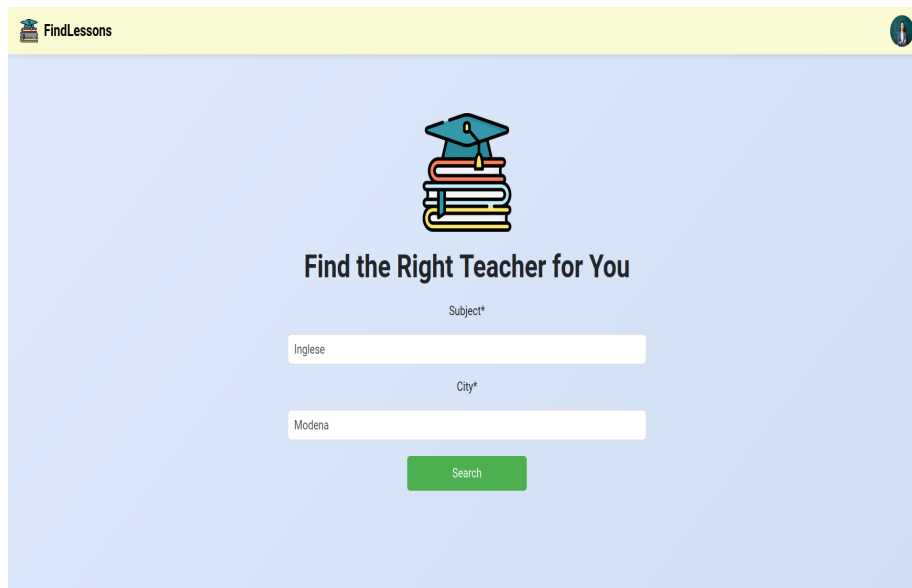


Figure 2: Homepage

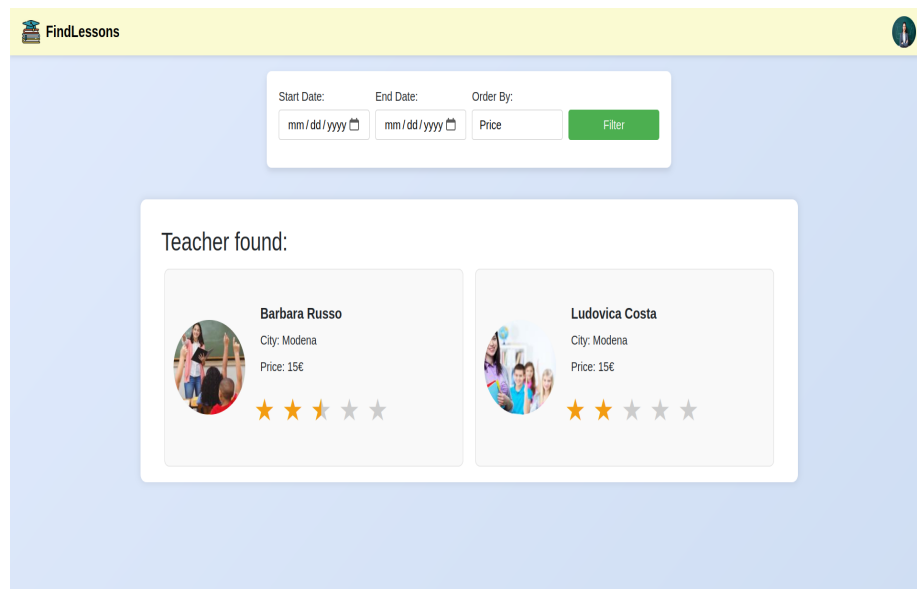


Figure 3: Visualizzazione risultati

7.2 Pagina profilo dei diversi utenti

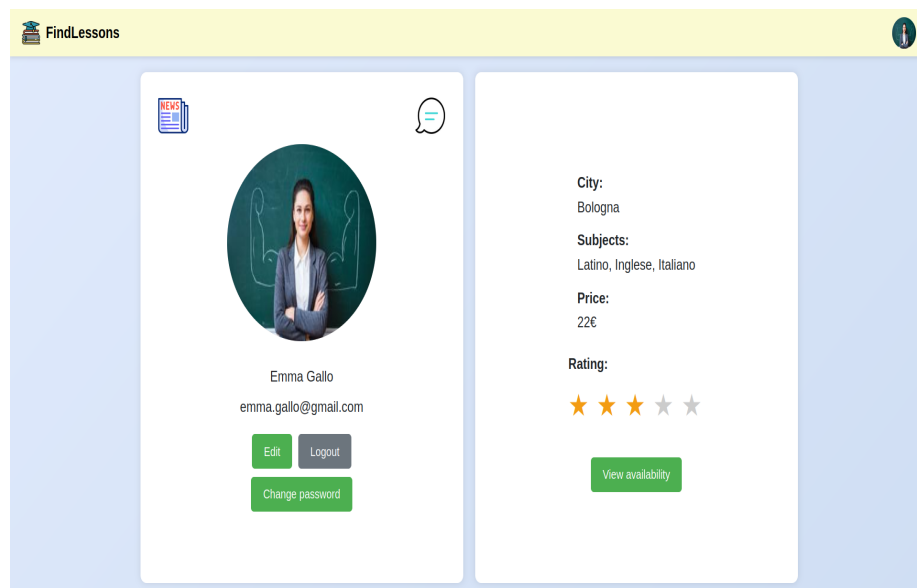


Figure 4: Profilo insegnante

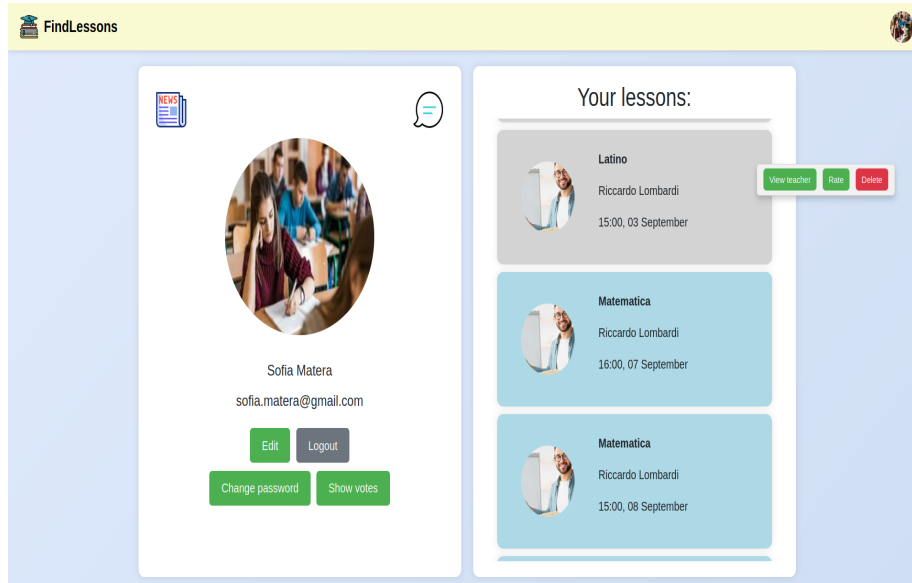


Figure 5: Profilo studente

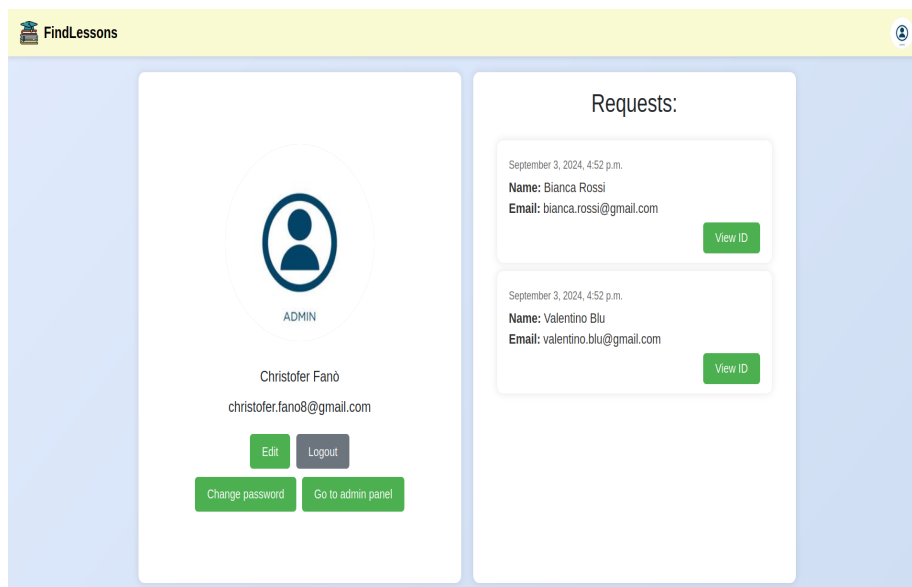


Figure 6: Profilo admin

7.3 Calendario per la gestione degli eventi



Figure 7: Visualizzazione calendario

7.4 Schermata della chat

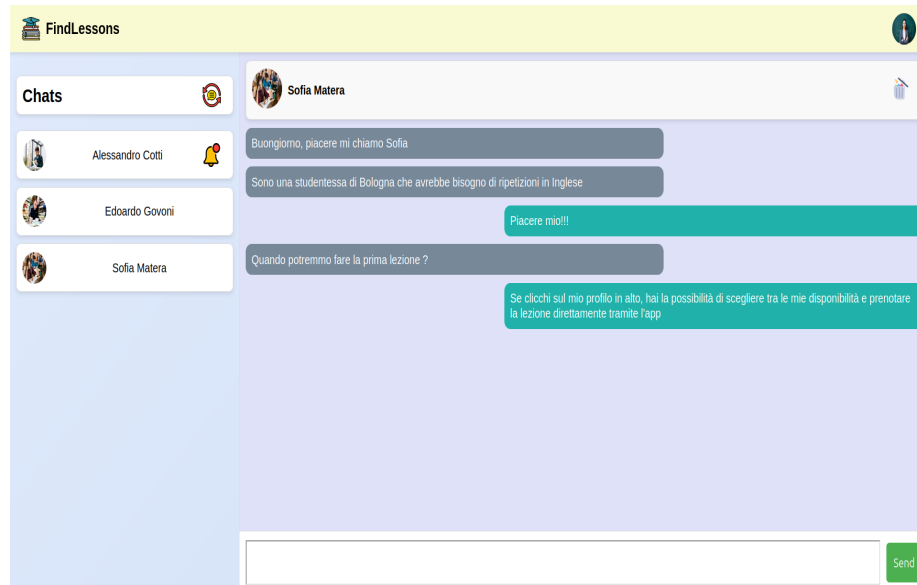


Figure 8: Schermata della chat

8 Possibili miglioramenti futuri

Di seguito sono riportate alcune modifiche che potrebbero essere apportate per migliorare ulteriormente l'esperienza nella webapp:

- **aggiornamento automatico delle chat** per segnalare la presenza di nuove chat o nuovi messaggi.

Al momento nella schermata delle chat vengono segnalati nuovi messaggi attraverso un'icona, tuttavia se si riceve un nuovo messaggio su una delle chat dell'utente, diversa da quella attualmente aperta con la connessione WebSocket, esso non viene segnalato. Visto che la funzionalità dei messaggi è una feature di supporto e non la funzionalità principale del progetto, si assume che gli insegnanti ricevano pochi messaggi dagli studenti e che l'aggiornamento possa essere effettuato periodicamente dall'insegnante attraverso l'apposita icona, dedicata al ricaricamento della pagina e al conseguente aggiornamento delle notifiche dei messaggi.

- **notifiche gestite attraverso websocket** per garantire la segnalazione real-time di nuove notifiche.

Al momento le notifiche sono consultabili cliccando sull'apposita icona nella schermata del profilo utente, attualmente si assume che l'utente, periodicamente o in prossimità dei propri impegni, apra la pagina del profilo per controllare la presenza di notifiche o nuovi messaggi.