Progetto

Corso di Programmazione per il Web

A.A. 2025/2026

Docenti: Marco Mesiti, Emanuele Cavalleri

Titolo esercitazione:

Play Room Planner

Gruppo composto da:

31480A, Lorenzo Piralla, lorenzo.piralla@studenti.unimi.it

29264A, Carlo Ancri, carlo.ancri@studenti.unimi.it

41436A, Leandro Suanno, andrealeandro.suanno@studenti.unimi.it

Data 1° consegna: *25/10/2025*
Data 2° consegna: *16/12/2025*
Data 3° consegna: 21/01/2026

Il progetto è già stato consegnato in precedenza? NO

1. Progettazione concettuale

Lo schema E-R proposto è stato progettato per rappresentare in modo completo e non ambiguo tutte le entità, le relazioni e le informazioni di interesse per il dominio applicativo. L'obiettivo è stato creare un modello dati robusto, normalizzato e in grado di supportare tutte le funzionalità richieste.

1.1 Descrizione dello Schema Concettuale

Lo schema si articola attorno ai concetti principali del sistema: gli **Iscritti**, i **Settori**, le **Sale prove** e le **Prenotazioni**.

- **Iscritto e Responsabile**: L'entità Iscritto rappresenta l'utente generico dell'associazione, identificato univocamente dalla sua Email. Per rispondere alla necessità di avere figure con privilegi diversi, è stata modellata una gerarchia tramite la relazione “è un”, dove il Responsabile è una specializzazione dell'Iscritto. Questo garantisce che ogni responsabile sia anche un iscritto, ereditandone gli attributi (nome, cognome, etc.), ed evitando così ridondanze di dati.
- **Settore e Sala prove**: L'associazione è organizzata in Settori, identificati dal nome del corso specifico, ognuno gestito (Gestisce) da un unico Responsabile. Ogni Sala prove appartiene (Fa riferimento) a un solo settore e può essere dotata di Strumentazione specifica. La sala è identificata da un NumAula, scelto come chiave primaria per garantire l'unicità a livello di intera associazione.
- **Prenotazione e Invito**: Il cuore funzionale del sistema è rappresentato dall'entità Prenotazione. Ogni prenotazione è creata (Crea) da un Responsabile, avviene in (Avviene in) una specifica Sala prove e ha una data e una fascia oraria. Per scelta, si è voluto mantenere il dato sul termine della prenotazione e non la sua durata. Per gestire la partecipazione degli iscritti, è stata introdotta l'entità Invito. Non è necessario fornire motivazione nel caso di invito accettato.atomico ogni singolo invito inviato (Invia) da un responsabile a un iscritto (Riceve) per una determinata prenotazione (Per). L'entità Invito è identificata dall’identificativo della prenotazione e dalla e-mail del ricevente.

1.1.2 Assunzioni e Scelte Progettuali Chiave

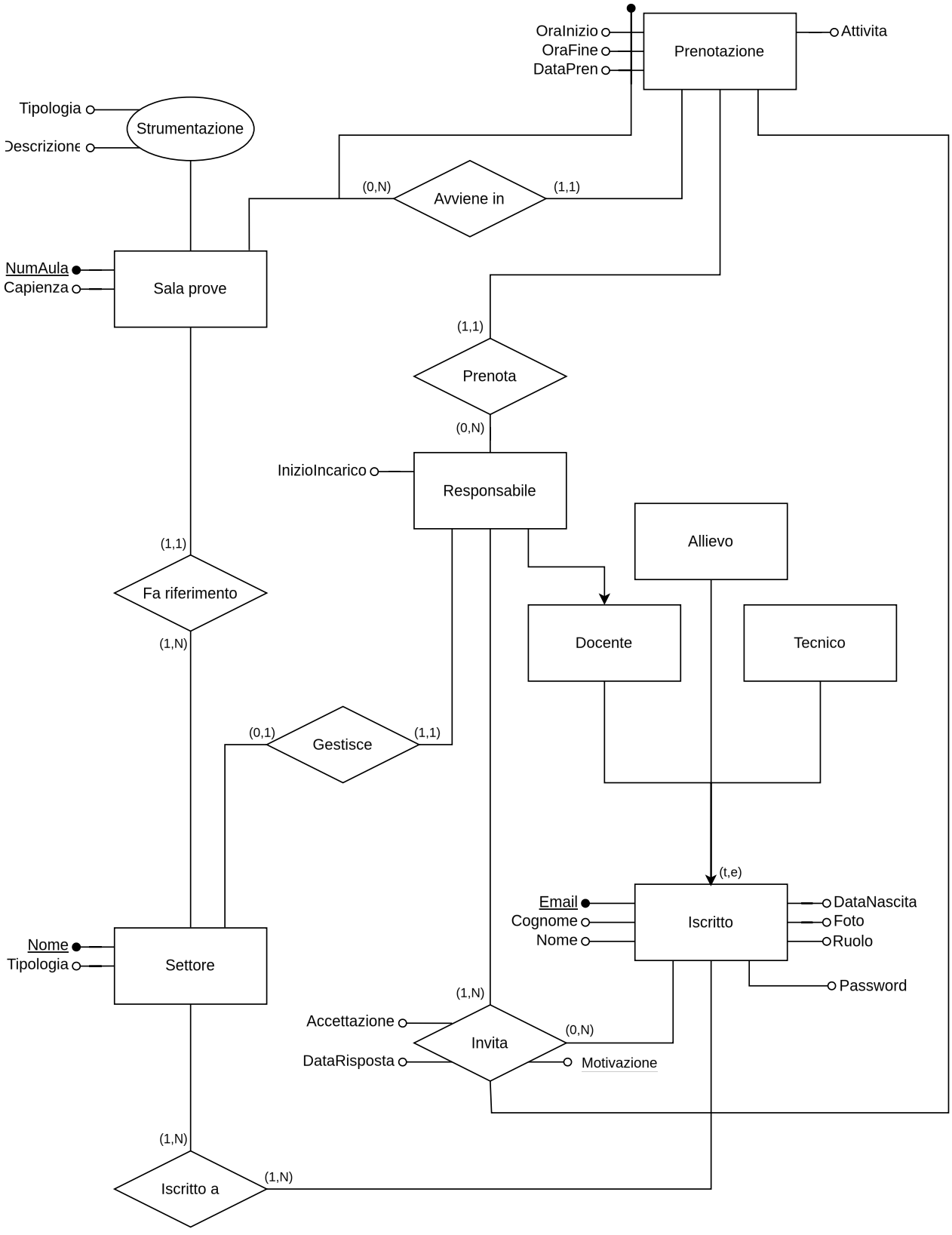
Durante la progettazione sono state fatte le seguenti assunzioni per risolvere ambiguità e definire una struttura solida:

1. **Modellazione del Responsabile come Iscritto Specializzato**: Si è assunto che un Responsabile sia prima di tutto un Iscritto dell'associazione, e che in particolare sia un Docente. La modellazione tramite una relazione di specializzazione (è un) è la scelta più coerente, in quanto permette di fattorizzare gli attributi comuni nell'entità Iscritto e di aggiungere attributi specifici (come Inizio incarico) solo per il Responsabile.
2. **Gestione dei Dati Calcolabili**: Le specifiche richiedono di tracciare il numero di anni di servizio di un responsabile, il numero di iscritti per settore e la durata della prenotazione. Si è

scelto di non memorizzare questi dati come attributi statici nel database, ma di calcolarli quando richiesti: per i primi due per evitare problemi legati ad aggiornamenti dei dati errati o mancanti, mentre per la durata è per come verrà implementato il modulo per la gestione delle prenotazioni, dove abbiamo previsto sarebbe stato necessario calcolare ogni volta l'orario di fine prenotazione a partire dalla durata per alcune operazioni, decidendo quindi per la strada opposta, ovvero calcolare la durata quando necessario.

3. **L'Invito come Entità Associativa**: Invece di modellare l'invito come una relazione complessa tra Responsabile, Iscritto e Prenotazione, si è scelto di promuoverlo a entità (Invito). Questa scelta è fondamentale perché l'invito possiede attributi propri (stato di accettazione, motivazione, data di risposta) che descrivono l'interazione tra l'utente e la prenotazione. Questo modello semplifica notevolmente le operazioni di interrogazione e aggiornamento dello stato di un invito.
4. **Identificatore Univoco per la Sala Prove**: Le specifiche indicano che una sala è identificata da un "nome nell'ambito del settore". Per evitare possibili omonimie e per avere un identificatore più robusto a livello globale, si è scelto di utilizzare l'attributo NumAula come chiave primaria dell'entità Sala prove, assumendo che questo sia un codice univoco per ogni sala all'interno dell'associazione, associandovi il settore di riferimento.
5. **Formato degli Orari di Prenotazione**: Le specifiche menzionano prenotazioni per "ore intere". Si è quindi assunto che gli attributi OraInizio e OraFine dell'entità Prenotazione conterranno valori di tipo intero (es. 9, 14, 22), semplificando i controlli di validità e di non sovrapposizione.

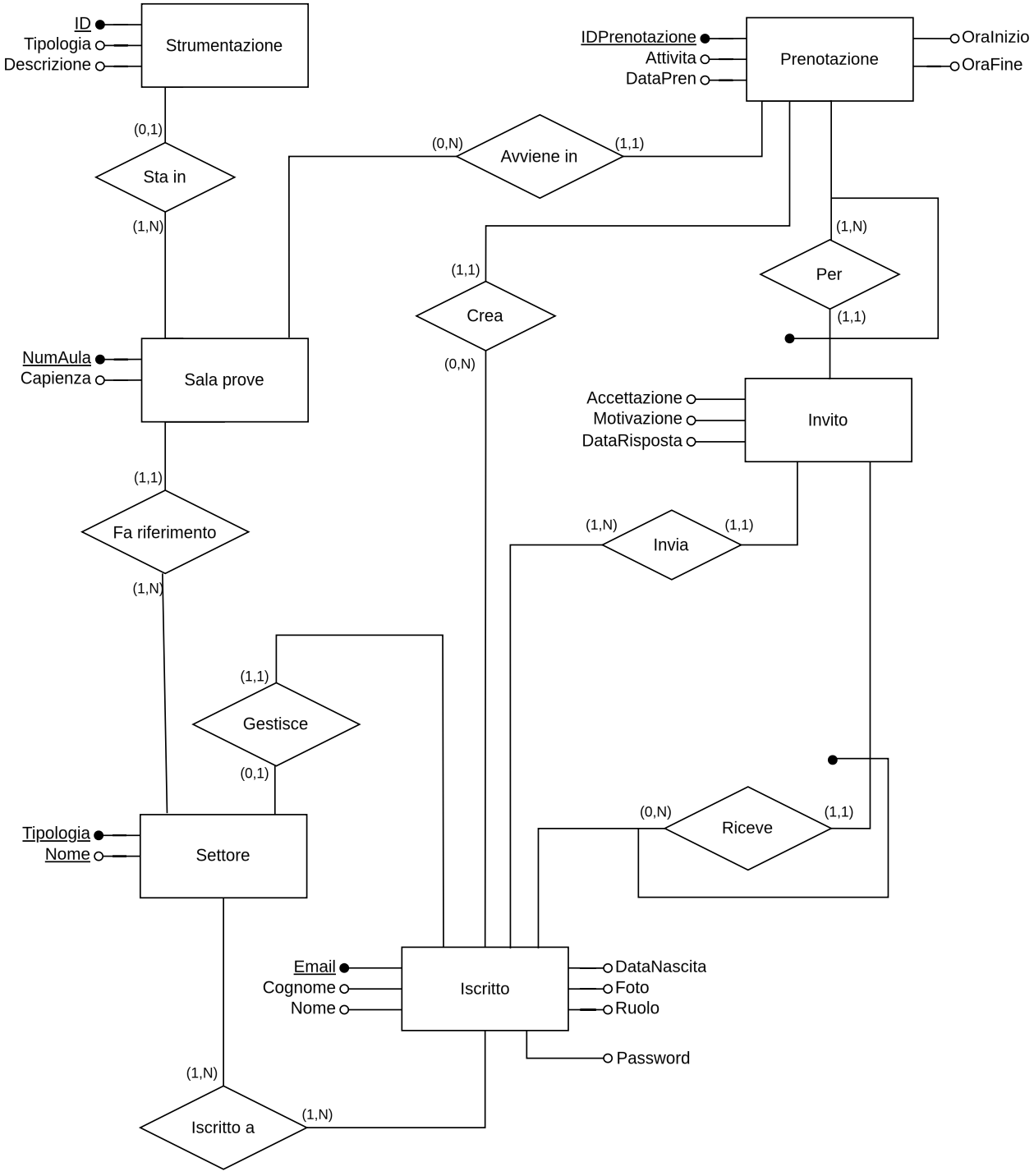
1.2 Schema ER


OraInizio
OraFine
DataPren
Prenotazione
Attivita
Tipologia
Strumentazione
Descrizione
(0,N)
Avviene in
(1,1)
NumAula
Capienza
Sala prove
(1,1)
Prenota
(0,N)
InizioIncarico
Responsabile
Allievo
(1,1)
Fa riferimento
(1,N)
Docente
Tecnico
(0,1)
Gestisce
(1,1)
(t,e)
Email
Cognome
Nome
Iscritto
DataNascita
Foto
Ruolo
Password
Nome
Tipologia
Settore
(1,N)
Accettazione
DataRisposta
Invita
(0,N)
Motivazione
(1,N)
Iscritto a
(1,N)

1.2.1 Vincoli di dominio

1. **Vincolo sull'orario delle prenotazioni**: Gli attributi OraInizio e OraFine di una Prenotazione devono essere numeri interi compresi tra 9 e 23 (inclusi). Inoltre, per ogni prenotazione, il valore di OraFine deve essere strettamente maggiore del valore di OraInizio.
2. **Vincolo di non sovrapposizione delle prenotazioni per sala**: Non possono esistere due prenotazioni per la stessa Sala prove i cui intervalli di tempo (DataPren, OraInizio, OraFine) si sovrappongono.
3. **Vincolo di non sovrapposizione degli impegni per un iscritto**: Un Iscritto non può avere due partecipazioni confermate a prenotazioni diverse i cui intervalli di tempo si sovrappongono. Il controllo va effettuato nel momento in cui un iscritto accetta un invito.
4. **Vincolo sulla capacità della sala**: Per ogni Prenotazione, il numero totale di iscritti che hanno confermato la loro partecipazione (ovvero, che hanno accettato l'invito) non può superare l'attributo Capienza della Sala prove in cui si svolge la prenotazione. È possibile da parte del Responsabile creare una prenotazione con più invitati della capienza della sala. Al raggiungimento della soglia di capienza, gli inviti in sospeso verranno automaticamente rifiutati con motivazione “Capienza massima raggiunta”.
5. **Vincolo di partecipazione subordinata ad invito accettato**: Un'istanza nella relazione Partecipa a (che lega un Iscritto a una Prenotazione) può esistere solo se esiste una corrispondente istanza nella relazione Invita tra lo stesso iscritto e la stessa prenotazione, e il cui attributo Accettazione è impostato su "TRUE". Questo avviene automaticamente quando un responsabile crea una prenotazione, poiché egli diventa il primo invitato con accettazione automatica della prenotazione al momento della creazione della stessa.
6. **Vincolo sulla coerenza della risposta**: L'attributo DataRisposta nella relazione Invita può avere un valore nullo solo se anche l'attributo Accettazione ha valore nullo (ossia quando non è stata data risposta all’invito).
7. **Vincolo sulla motivazione del rifiuto**: L’attributo Motivazione (nella relazione Invita) può contenere un valore nullo solo se l'attributo Accettazione è impostato su "TRUE" o “NULL”. Non è quindi necessario fornire motivazione nel caso di invito accettato.
8. **Vincolo di autorità sulla prenotazione**: Un Responsabile può creare una Prenotazione (Prenota) solo per una Sala prove che appartiene (Fa riferimento) alla tipologia di Settore (musica, danza, teatro) in cui quel responsabile ha la gestione (Gestisce).

1.3 Schema ER ristrutturato



2. Progettazione logica e comandi SQL

2.1 Entità modellate

Iscritto(Email, Cognome, Nome, DataNascita, Foto, Ruolo, Password)

Settore(Nome, Tipologia, ResponsabileEmail$_{Iscritto}$)

Iscrizione(IscrittoEmail$_{Iscritto}$, SettoreNome$_{Settore}$)

Prenotazione(IDPrenotazione, DataPren, OraInizio, OraFine, Attivita, NumAula$_{SalaProve}$, ResponsabileEmail$_{Iscritto}$)

Invito(IDPrenotazione$_{Prenotazione}$, IscrittoEmail$_{Iscritto}$, Accettazione, Motivazione, DataRisposta)

SalaProve(NumAula, Capienza, SettoreNome$_{Settore}$)

Strumentazione(ID, NumAula$_{SalaProve}$, Tipologia, Descrizione)

2.1.1 Chiavi sintetiche

- **Prenotazione** (**IDPrenotazione**): per semplificare le interrogazioni tra Invito, Prenotazione e in generale le interazioni nell'ambito delle prenotazioni, si è scelto un identificativo numerico univoco, che risulta più facilmente gestibile e trasportabile rispetto a una chiave composta da tre diversi attributi.
- **SalaProve** (**NumAula**): abbiamo scelto un codice numerico come chiave primaria per l'entità SalaProve, e non SettoreNome (già chiave primaria in Settore), per prevedere la possibilità che un settore abbia più aule.
- **Strumentazione** (**ID**): l'identificativo per la strumentazione si rivela necessario per distinguere strumenti diversi situati nella stessa aula, senza vincolarsi al nome proprio dello specifico oggetto (comunque descritto da "Descrizione" e categorizzato da "Tipologia").

2.1.2 Vincoli ulteriori

- Solo i docenti possono essere responsabili
- Un responsabile (docente) deve essere iscritto al settore che gestisce.
- Un settore può non avere un responsabile, come nel caso di eliminazione del responsabile corrente e in attesa dell'assegnazione di uno nuovo.
- Il responsabile puo' prenotare aule diverse dal corso che supervisiona, ma non di tipologie (musica, teatro, danza) diverse.
- L'attributo Ruolo di Iscritto non deve accettare testo libero, ma un insieme predefinito di valori (studente, docente, tecnico).
- L'attributo Tipologia di Settore non deve accettare testo libero, ma un insieme predefinto di valori (danza, musica, teatro)

- L'attributo Tipologia di Strumentazione non deve accettare testo libero, ma un insieme predefinito di valori (strumenti musicali, impianti audio, specchi, palcoscenico). La descrizione indica a testo libero il tipo di oggetto indicato.
- L’eliminazione di un iscritto responsabile elimina anche la relativa entry in Responsabile, le sue prenotazioni e rende “NULL” ResponsabileEmail in Settore.
- L’eliminazione di un settore elimina anche le relative entries di Iscrizione, e rende “NULL” i relativi attributi SettoreNome in SalaProve.
- L’eliminazione di un Iscritto elimina anche le relative entries in Invito e Iscrizione.
- L’eliminazione di un’aula elimina le relative prenotazioni, e rende “NULL” il valore “NumAula” in Strumentazione

2.2 Script di CREATE e QUERIES

Nella cartella “sql” sono presenti i files .sql contenenti gli scripts di creazione (*create.sql*) e popolamento (*populate.sql*) delle tabelle, nonché il file con le queries richieste dalla consegna (*queries.sql*). Il popolamento presente nel file copre tutte le possibili casistiche di inserimenti di dati corretti.

Il nome del database a cui la funzione *connection.php* fa riferimento è ‘playroomplanner’.

3. Caratteristiche del CSS

Abbiamo scelto dei **fogli css** presi da un template (*https://templatemo.com/tm-574-mexant*) online, nel quale è incluso anche un framework di Bootstrap. Il template in questione presenta elementi versatili, adatti alla struttura delle nostre pagine; le modifiche apportate, presenti in altri file chiamati *custom_style.css*, sono utili a rendere coeso lo stile di tutti gli elementi da noi implementati all'interno dell'applicativo web.

Abbiamo posto attenzione a tutti gli elementi interagibili dell'applicativo, in particolare all'associazione tra colore e funzionalità: i pulsanti che corrispondono ad azioni critiche (come eliminazione di dati e reset di form) presentano colore rosso, gli elementi che permettono la navigazione, azioni di conferma e minori (accesso ad Area Personale, Login, submit di form, etc..) hanno colore arancione o verde, gli elementi restanti mantengono un colore conforme a quello presente nel template originale. Per quanto riguarda lo stile dei messaggi di errore, essi vengono visualizzati attraverso la funzione *Alert()* di JavaScript, in modo da segnalare il problema in modo immediato all'utente; essi hanno il compito di guidare l'utente per risolvere il problema sollevato in precedenza.

4. Funzionalità dell'applicazione

Nel seguente paragrafo sono esposte le funzionalità dell'applicazione, organizzate gerarchicamente, con i dettagli su input/output e la loro implementazione nel codice.

4.1 Gestione Utenti

Funzionalità relative all'autenticazione e alla gestione dei profili utente.

4.1.1 Login (Autenticazione)

Descrizione: Permette agli utenti registrati di accedere all'area riservata.

Input: *email*, *password*.

Risultato (Successo): Reindirizzamento all'area personale o all'URL a cui si vuole accedere, impostazione delle variabili di sessione (*$_SESSION['logged_in']*, *$_SESSION['user']*, ecc.). Se l'utente che accede è *admin*, esso verrà reindirizzato alla pagina di *rootAccount.php*.
Risultato (Errore): Reindirizzamento alla pagina di login con messaggio di errore ("Password non corretta", "Email non trovata").

Implementazione: *login_controller.php*

4.1.2 Logout (Disconnessione)

Descrizione: Termina la sessione dell'utente correntemente loggato.

Risultato: Distruzione della sessione e reindirizzamento alla homepage.

Implementazione: *logout.php*

4.1.3 Gestione Profilo

Descrizione: Permette di **inserire**, **modificare** o **eliminare** i dati di un utente (Iscritto). Queste operazioni possono essere svolte da Iscritti senza privilegi sui loro stessi dati, mentre il Root Account può eseguirle su qualsiasi account presente nel database.

Input: *name*, *surname*, *email*, *pwd*, *DOB* (data di nascita), *photo*, *action* ("inserisci", "modifica", "elimina").

Risultato (Successo): Messaggio di conferma esecuzione query.
Risultato (Errore): Messaggio di errore SQL o parametri mancanti.

Implementazione: *user_data_API.php*

4.2 Gestione Prenotazioni e Inviti

Funzionalità principali per la gestione delle prenotazioni delle aule.

4.2.1 Visualizzazione, creazione, modifica ed eliminazione Prenotazioni

Descrizione: mostra tutte le prenotazioni effettuate dal responsabile loggato (o Account Root) permettendone la **modifica**, l'**eliminazione** e la possibilità di **crearne nuove invitando utenti** ad esse.

Input:

- [per creazione e modifica] → *DataPren*, *OraInizio*, *OraFine*, *NumAula*, *Attivita*.
- [per invito utenti] → *IscrittoEmail*.

Risultato (Successo):

- *Alert* di conferma per operazioni definitive come submit di form.
- Visualizzazione aggiornata della vista prenotazioni dopo inserimento, modifica ed eliminazione.
- Aggiornamento lista invitati dopo modifica.

Risultato (Errore):

- Alert per errori inaspettati (con log in console), *Alert* per inserimenti non validi.

Implementazione: *api-gestionePrenotazioni.php* *gestionePrenotazioni.js*

4.2.2 Visualizzazione, accettazione e rifiuto Inviti

Descrizione: mostra tutti gli inviti inviati ad un utente permettendone la **visualizzazione**, il **rifiuto** e l'**accettazione**.

Input: funzione *get_bookings()*, passando come paramentro la stringa "*invites*". In frontend viene chiamata la funzione *showBookings()* per la visualizzazione a calendario e *scroll_from_invites()* per la visualizzazione verticale degli inviti.

Implementazione: *bookings_API.php* per il recupero delle informazioni nel backend, mentre *calendarManager.js* per la visualizzazione su client. Tutte queste funzioni vengono implementate in *area_personale.php*.

4.3 Gestione Aule

Funzionalità per la visualizzazione e il recupero delle informazioni sulle sale prova. Le informazioni possono essere rese accessibili a qualsiasi utente, purché sia un iscritto e sia loggato nel sistema.

4.3.1 Visualizzazione Sale per Categoria

Descrizione: Genera l'HTML per mostrare le card delle sale prova filtrate per tipologia (Danza, Musica, Teatro).

Input: tipologia (stringa).

Risultato: Codice HTML con le informazioni e le immagini delle sale.

Implementazione: Funzione *mostraSale()* in *functions.php* (utilizzata in *sale_prova.php*).

4.3.2 Recupero Lista Aule

Descrizione: Fornisce l'elenco delle aule disponibili (es. per popolare le select nei form).

Input: *azione="getAule"* (inviato a *api-gestionePrenotazioni.php*).

Risultato: JSON con la lista delle aule.

Implementazione: *api-gestionePrenotazioni.php*

4.3.3 Visualizza impegni per Sala

Descrizione: Fornisce una vista a griglia in cui sono segnate le prenotazioni effettuate da responsabili / admin per la Sala prove selezionata.

Input: *numAula*

Risultato: Reindirizzamento alla pagina *prenotazioni_aula.php* in cui viene mostrato il contenuto sopra citato.

Implementazione: Nella pagina *sale_prova.php,* cliccando il bottone “Mostra Prenotazioni”.

4.4 Account Root

L'account root possiede privilegi estesi che gli consentono di gestire l'intera piattaforma. Tutte le funzionalità descritte vengono implementate in *rootAccount.php* e *rootAccount.js*.

4.4.1 Gestione responsabili corsi

Descrizione: L’admin può visualizzare e modificare chi è a capo dei vari settori, in particolare può nominare responsabile un qualsiasi utente a patto che esso abbia come ruolo *docente*.

Implementazione: *mostraSezione(manager-responsabili)*.

4.4.2 Gestione dati utenti e ruoli

Descrizione: L'admin possiede controllo completo sui dati di ogni singolo iscritto, con possibilità di modificarli, eliminarli e visualizzare i loro dati.

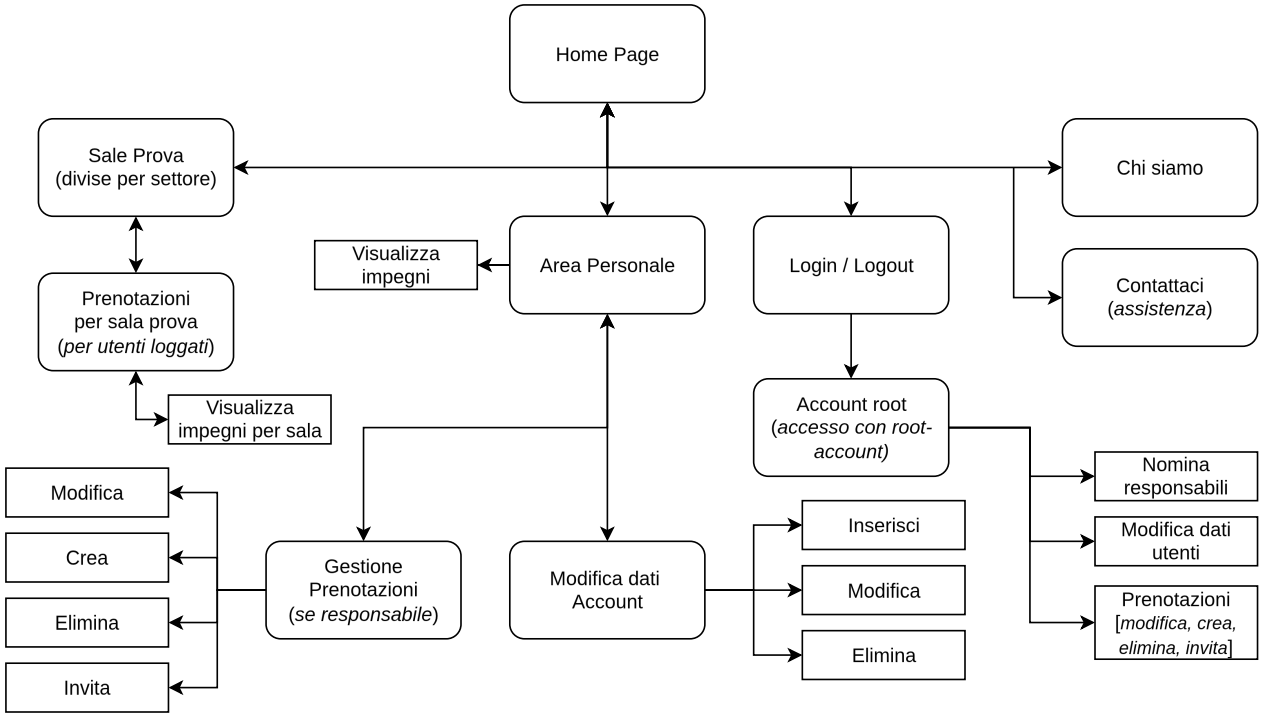
Implementazione: *mostraSezione(edit-user-data)*.

4.4.3 Gestione prenotazioni

Descrizione: *[si faccia riferimento al punto 4.2.1]*.

5. Struttura dell'applicativo Web

Il nostro applicativo web è strutturato nel seguente modo:



Nel diagramma, i rettangoli con angoli smussati rappresentano delle vere e proprie pagine raggiungibili tramite il browser, mentre i rettangoli con angoli appuntiti rappresentano le funzioni logiche di alto livello offerte.

Ogni pagina all'interno dell'applicativo ha una struttura ben definita e coerente: presenta una *navbar*, una sezione di *header*, la sezione *body* nella quale c'è l'effettivo contenuto dinamico ed infine il *footer*. È importante notare la gestione degli accessi a ciascuna pagina, onde evitare un accesso diretto ad una pagina accessibile solo ad utenti loggati o ad amministratori; all'interno del file *auth_check.php* è stata implementata una serie di istruzioni che indirizza l'utente nella pagina desiderata (se loggato) o al login (se non è loggato).

È stato implementato un **root-account** con privilegi assoluti, esso può: modificare il ruolo degli utenti (a differenza degli stessi) e nominare i responsabili per i rispettivi settori.

Abbiamo ritenuto di fondamentale importanza il **criptaggio delle password** per ogni Iscritto, sia nel passaggio da client a server, sia nel database. Tutto questo per garantire la riservatezza, l'integrità dei dati e la protezione contro sniffing e altre pratiche malevole per estorsione dei dati.

Nella directory */sql* si trova il file *password_non_criptate.txt* che contiene tutti e le mail degli utenti associate alle rispettive password in chiaro al fine di testare l'applicativo.

All'interno dell'applicativo abbiamo utilizzato **riferimenti** sia **assoluti** (*/PlayRoomPlanner/…*) alla directory PlayRoomPlanner, sia **relativi** (*../common/*) al file in cui vengono chiamati. La scelta migliore sarebbe utilizzare dei riferimenti *relativi* per rendere gli accessi sicuri, ma nel caso del file *navbar.php,* utilizzato in ogni pagina, è fondamentale utilizzare dei riferimenti assoluti.

5.1 Tecnologie utilizzate nell'applicativo

5.1.1 Javascript

Utilizzato principalmente nei file presenti sotto la cartella */js* (come *calendarManager.js, gestionePrenotazioni.js, userData.js*) e nel file *functions.js* all'interno della cartella */common,* gestisce la logica dell'applicativo **lato client**: rende l'interfaccia dinamica senza dover ricaricare l'intero contenuto ad ogni cambiamento. Un compito fondamentale affidato al codice scritto in *Javascript* è la validazione dell'input immesso dall'utente all'interno di form, oltre alla gestione degli eventi del *DOM* (click su bottoni, aperture menu a tendina, etc…).

5.1.2 Sessioni

Gestite **lato server** tramite *PHP*, sono fondamentali per mantenere lo stato di autenticazione dell'utente attraverso le pagine dell'applicativo. Le sessioni memorizzano i dati in variabili (di sessione appunto) chiamate *$_SESSION[]* come 'foto', 'nome', 'cognome', 'ruolo', 'email', 'logged_in' per personalizzare l'esperienza d'uso e per limitare l'accesso ad alcune pagine ad utenti non loggati o non amministratori. Le variabili sopra citate vengono inizializzate in *login_controller.php*, per poi essere controllate da *auth_check.php* all'inizio di qualsiasi pagina che richiede il login per l'accesso. Esse le ritroviamo in qualsiasi funzione che debba generare contenuti personalizzati per l'utente.

5.1.3 Cookies

Servono a memorizzare l'**ID di sessione** sul browser del client per legarlo alla sessione attiva sul server. In fase di *logout*, vengono utilizzati per invalidare la sessione lato client: lo script aggiorna il cookie di sessione con una data scaduta nel passato per cancellarlo dal browser, garantendo una disconnessione sicura. Nell'applicativo sviluppato, si è scelto di affidare il meccanismo di sessione a *PHP* tramite l'utilizzo delle variabili di sessione.

5.1.4 AJAX (Fetch API)

Questa tecnologia permette di effettuare **chiamate asincrone al server** e scambiare dei dati con esso, senza dover ricaricare la pagina; si è scelto di utilizzare il metodo più moderno *fetch()* per via della

minor complessità rispetto ad *AJAX* classico. Queste chiamate si possono trovare in *calendarManager.js*, *gestionePrenotazioni.js*, *userData.js*, *functions.js*.