Progetto Tecnologie Web 2025

Gruppo 09:

Mulloni Manuel
Pardini Fabrizio
Piersigilli Francesco
Tomassetti Mattia

➔ Contributo dei membri del gruppo

Come gruppo è stato deciso di utilizzare il software Github per tutta la durata del progetto per aumentare la produttività del team stesso, abbiamo comunque deciso di separarci i compiti sin dall'inizio in maniera equa, di seguito le percentuali approssimative di svolgimento generale del progetto.
All interno del codice stesso del nostro prodotto finale abbiamo una percentuale maggiore di scrittura da parte dei componenti Mulloni Manuel e Fabrizio Pardini, i quali sono riusciti a collaborare in maniera rapida e efficiente, per quanto riguarda invece la parte più generale, quindi di progettazione, di elaborazione e strutturazione del database e di relazione, il maggior contributo è stato dato da Tomassetti Mattia e Piersigilli Francesco.

Mulloni Manuel 30%
Pardini Fabrizio 30%
Piersigilli Francesco 20%
Tomassetti Mattia 20%

➔ Descrizione del sito

l'obiettivo del sito è la gestione delle prestazioni ambulatoriali specialistiche della struttura sanitaria Poliambulatorio Salus.
strutturata in 4 livelli:

a. Area pubblica

in quanto area pubblica le funzionalità di questo livello sono accessibili anche agli altri livelli:

1. presentazione del poliambulatorio e le sue funzionalità;
2. la descrizione dei dipartimenti specialistici e delle prestazioni offerte
3. la registrazione di un nuovo utente (solo per gli utenti di Livello 1) specificando username, immagine profilo, nome, cognome,data di nascita, telefono, indirizzo, e password (con conferma password).

b. Area utenti esterni registrati

area per gli utenti esterni registrati i quali possono:

1. modificare le informazioni del profilo
2. richiedere prestazioni
3. annullare prenotazioni non ancora usufruite
4. visualizzare prenotazioni (passate e future ad eccezione di quelle non usufruite)

c. Area riservata ai componenti dello staff

area riservata ai componenti dello staff che gestisce le prenotazioni, che consenta:

1. la visualizzazione dell'agenda giornaliera di ciascuna prestazione (solo per prestazioni ancora da erogare), con l'indicazione, specificando un giorno, l'ora e le azioni da fare (modifica/elimina);
2. la modifica delle agende delle prestazioni erogate dalla struttura (annullamento di prenotazioni)
3. l'inserimento di nuove prestazioni in agenda, a valle della richiesta degli utenti;

d. Area ad accesso esclusivo all'amministratore del sito

area ad accesso esclusivo dell'amministratore del sito che consenta:

1. gestione (modifica/eliminazione) degli utenti
2. gestione(creazione/modifica/eliminazione) delle prestazioni
3. gestione(creazione/modifica/eliminazione) dei dipartimenti
4. gestione delle ripartizioni per le prestazioni abilitate dello staff

5. la visualizzazione (attivazione) di analisi statistiche che consentano, relativamente ad un intervallo temporale specificato, di visualizzare:
    I. il numero di prestazioni erogate, suddivise per dipartimento
    II. il numero di prestazioni erogate ad un utente esterno
    III. il numero di prestazioni erogate, suddivise per prestazioni

➔ Diagrammi e schemi del sito

Diagramma dei link

Livello 1 **Livello 2** **Livello 3** **Livello 4**

- ➔ /
- ➔ admin
  - admin/dipartimenti
  - admin/dipartimento
    - admin/dipartimento/cancella/{id}
    - admin/dipartimento/edit/{id}
    - admin/dipartimento/new
  - admin/getStaff
  - admin/getUsers
  - admin/prestazioni
    - admin/prestazioni/byCliente
    - admin/prestazioni/byDipartimento
    - admin/prestazioni/byPrestazione
  - admin/staff/prestazioni
  - admin/statistiche
  - admin/user
    - admin/user/changePermission
    - admin/user/delete
- ➔ agenda
  - agenda/cancel/{id}
  - agenda/delete/{id}
  - agenda/new
    - agenda/new/{id}
  - agenda/{id}
- ➔ ajax
  - ajax/descrizione-dipartimento/{id}
- ➔ api
  - api/notifications
  - api/notifications_amount
- ➔ cardiologia
- ➔ cliente
  - cliente/prenotazione
    - cliente/prenotazione/delete/{id}
    - cliente/prenotazione/new

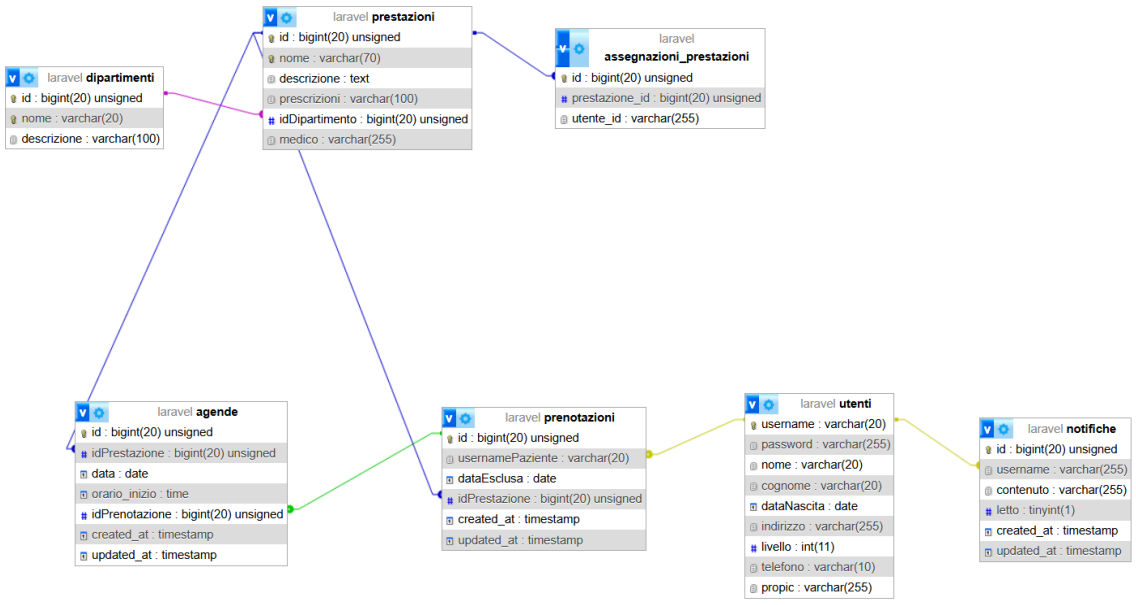
- cliente/prenotazione/prestazione-autocomplete
- cliente/prenotazione/show
  - cliente/prenotazioni/store
  - cliente/prestazioni/{dipartimento}

➔ dermatologia
➔ dipartimenti
  - dipartimenti/{id}

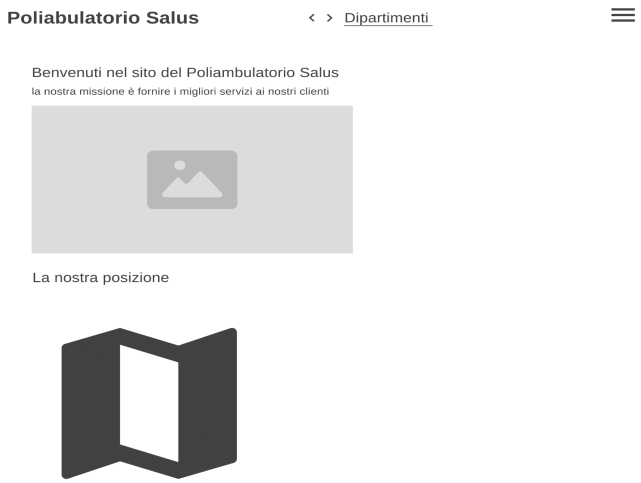
➔ login
➔ logout
➔ oculistica
➔ ortopedia
➔ pediatria
➔ prestazione
  - prestazione/delete/{id}
  - prestazione/edit/{id}
  - prestazione/{id}

➔ prestazioni/new
➔ prestazioni/show
➔ radiologia
➔ register
➔ staff
➔ user/edit

Database



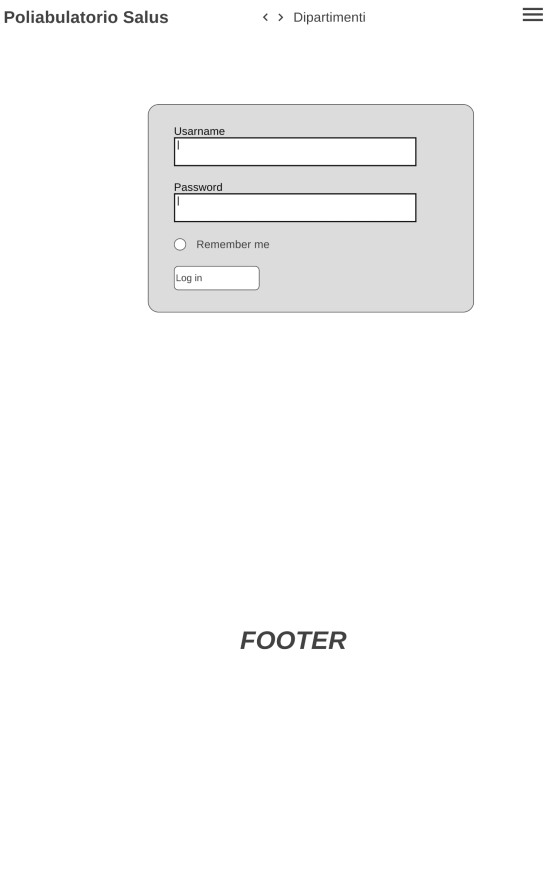
Mockup

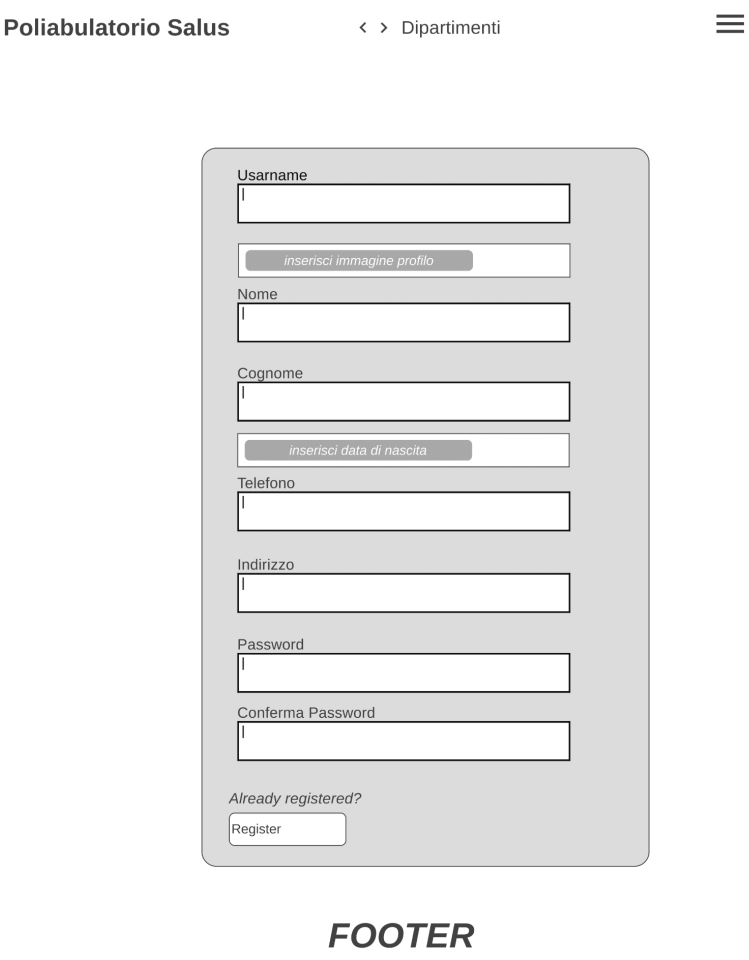


FOOTER

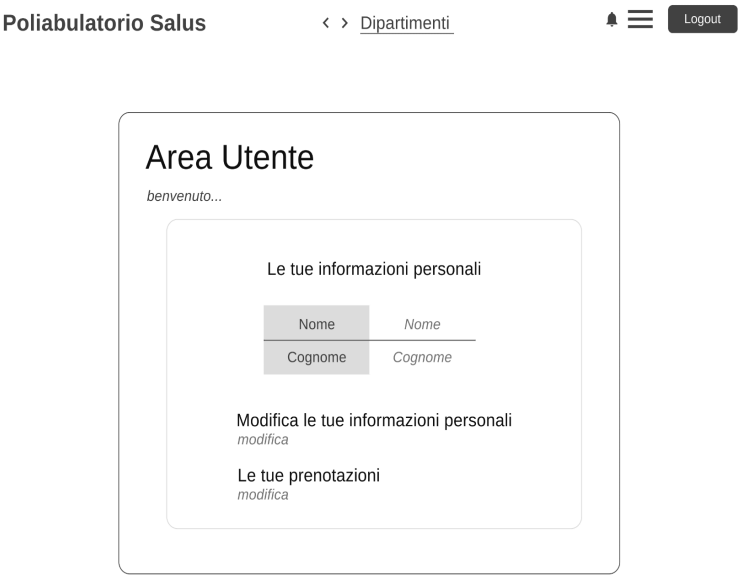
FOOTER

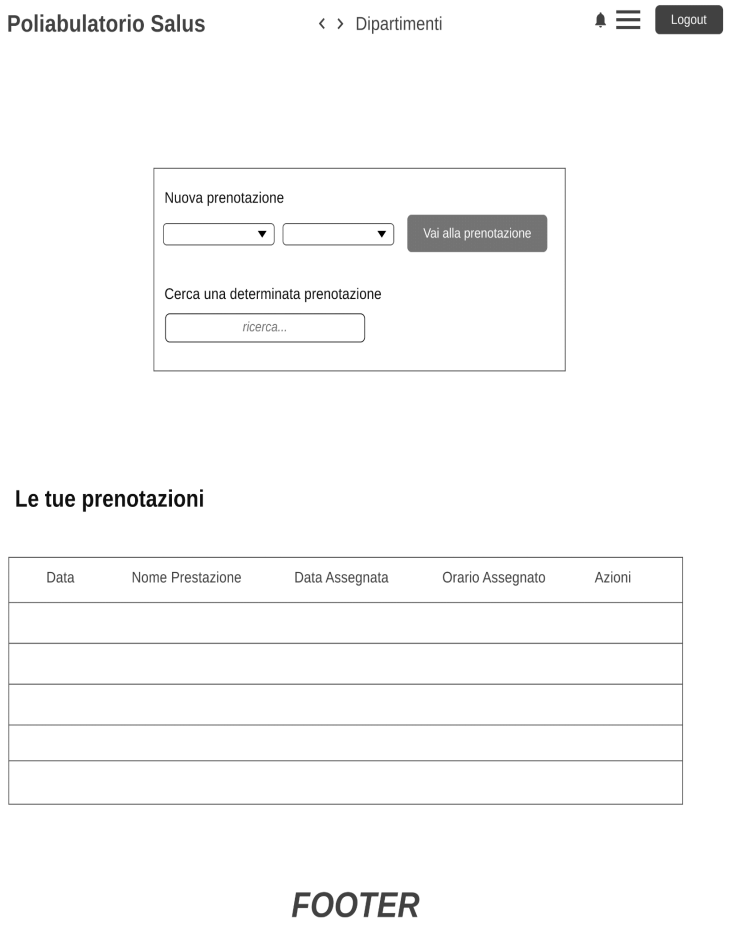
Qui sopra abbiamo i mockup relativi all' Area Pubblica.
Quindi la prima pagina accedendo al nostro sito, poi la pagina entrando all'interno di dipartimenti con i vari dipartimenti disponibili e infine la descrizione per ogni dipartimento.



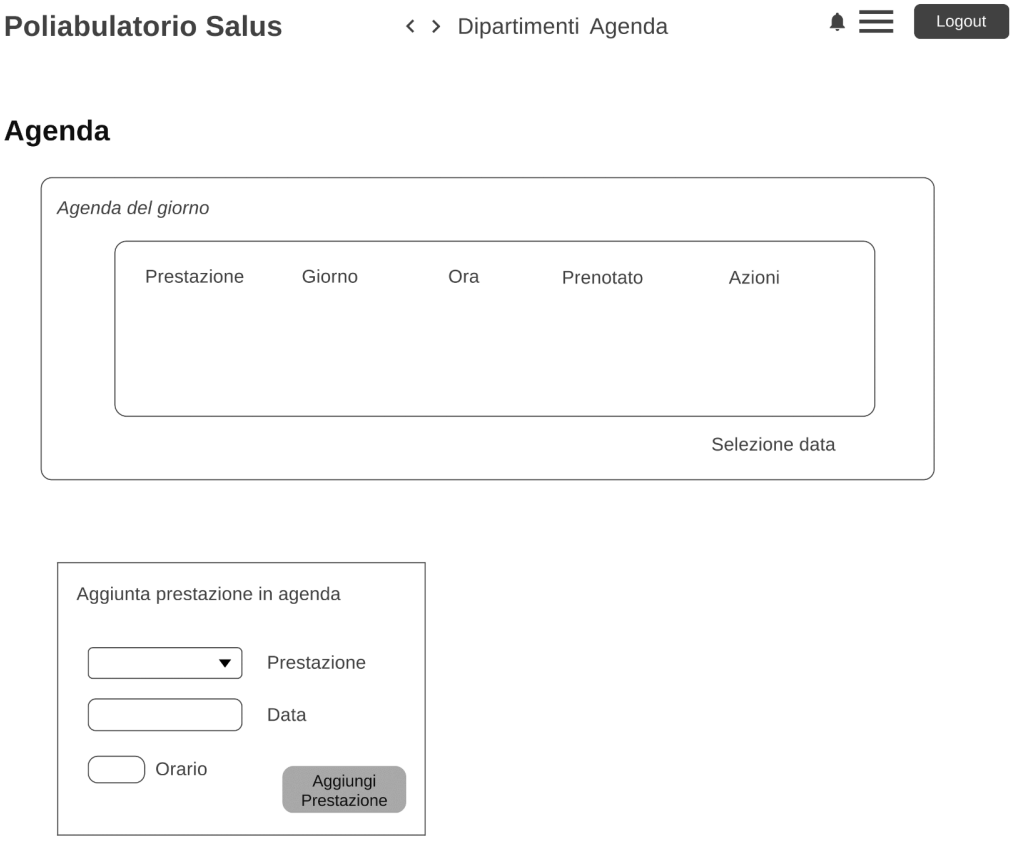


Qui sopra abbiamo le due pagine di Login e Registrazione.



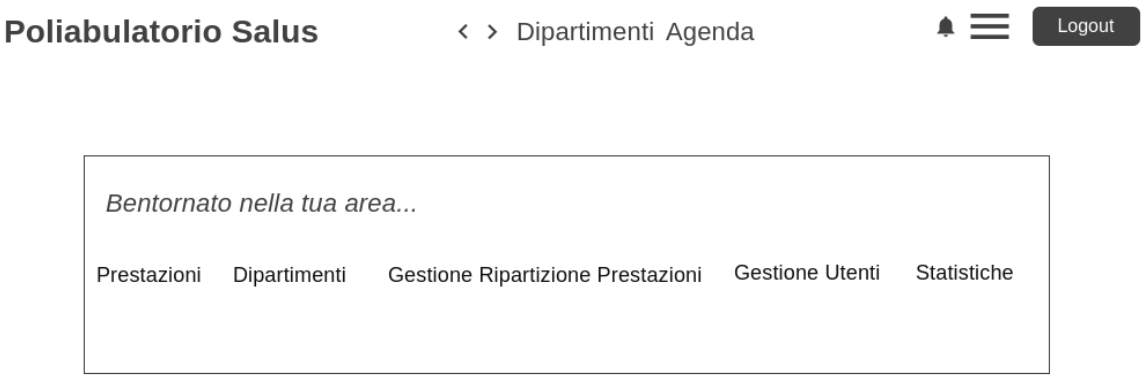


Qui sopra abbiamo le pagine relative all'Area Utente, con le informazioni personali e con la possibilità di andare a modificare le informazioni personali e modificare le sue prenotazioni, a destra proprio la modifica delle prenotazioni.



FOOTER

Qua sopra abbiamo l'Area Staff, ad accesso ed uso esclusivo dei membri dello staff, dove abbiamo un'agenda con le varie prestazioni, dove si possono andare a visionare, modificare e eliminare le prestazioni di un determinato giorno.
Inoltre è possibile aggiungere una determinata prestazione.



FOOTER

qui sopra abbiamo la pagina per l’admin dove è possibile gestire tutto il sito, dall’aggiunta,modifica e eliminazione delle prestazioni alla visualizzazione delle statistiche.

➔ Funzionalità Interessanti

Richiesta Prestazione

Per poter richiedere una prestazione l'utente deve accedere alla propria area riservata, per poi accedere ad un'ulteriore pagina nella quale avrà la possibilità di compilare un form per scegliere la prestazione che intende prenotare e una data da escludere.

```
{{ html()->form('POST', route('prenotazione.create'))->class('form-style')->open() }}

<h3 class="form-description">Nuova Prenotazione</h3>
{{ html()->select('idDipartimento')->class('select_style')->id('idDipartimento')->required()->options($dipartimenti->pluck('nome', 'id')) }}

{{ html()->select('idPrestazione')->class('select_style')->id('idPrestazione')->required()->options([]) }}

<div class="form-group">
    {{ html()->label('Data Esclusa')->for('dataEsclusa') }}
    {{ html()->date('dataEsclusa')->class('form-control')->required() }}
</div>

<div class="form-group">
    {{ html()->hidden('usernamePaziente', Auth::user()->username) }}
</div>

{{ html()->submit('Aggiungi Prenotazione')->class('submit-button button-style') }}

{{ html()->form()->close() }}

<script src="{{ asset('assets/js/prenotazioniCreate.js') }}"></script>
```

Non appena l'utente cliccherà il bottone della prenotazione verrà inviata una richiesta di tipo POST che invierà tutti i dati del FORM ad un apposito metodo nel Controller della classe Prenotazione.

```
Route::post('/cliente/prenotazione/new', [PrenotazioneController::class, 'createPrenotazione'])->name('prenotazione.create');
```

Il metodo createPrenotazione si occuperà di verificare se l'utente è di livello 2 (e quindi autorizzato a richiedere una prenotazione) e in tal caso di inserire la prenotazione all'interno di un'apposita tabella all'interno del database.
Per quanto riguarda i dati da inserire all'interno della tabella, lo username dell'utente viene preso dall'utente che è attualmente loggato all'interno del sito, mentre i campi restanti (id della prestazione prenotata e la data esclusa) vengono presi dal FORM compilato in precedenza, dopo aver passato un controllo di conformità.

```
public function createPrenotazione(NewPrenotazioneRequest $request)
{
    $user = Auth::user();
    if ($user->livello == 2) { // Livello 4 per admin, 3 per dipendente
        $prenotazione = new Prenotazione();
        $prenotazione->fill($request->validated());
        $prenotazione->usernamePaziente = $user->username;
        $prenotazione->save();
        return redirect()->back()->with('success', 'Prenotazione creata con successo.');
    } else {
        return redirect()->back()->with('error', 'Non hai i permessi per creare una prenotazione.');
    }
}
```

La classe NewPrenotazioneRequest si occupa di verificare se l'utente è autorizzato ad effettuare l'operazione e se i dati che si stanno inviando sono conformi a quanto presente

nel database.

```php
class NewPrenotazioneRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
     */
    public function rules(): array
    {
        return [
            'dataEsclusa' => 'nullable|date|after_or_equal:today',
            'idPrestazione' => 'required|integer|exists:prestazioni,id',
        ];
    }

}
```

Assegnazione Prestazione

L'assegnazione di uno slot ad una determinata richiesta di una prestazione da parte di un utente può essere effettuata solamente dallo staff (livello 3).
Per prima cosa viene mostrata una pagina nella quale, selezionando il giorno, è possibile vedere tutti gli slot disponibili per quella giornata. Viene poi offerta la possibilità, tramite 2 bottoni, di eliminare o modificare quello slot.

```blade
<p id="agenda-date"> Agenda del giorno: {{ $showDate }}</p>
<table>
    <thead>
        <tr>
            <th>Prestazione</th>
            <th>Giorno</th>
            <th>Ora</th>
            <th>Prenotato</th>
            <th>Azioni</th>
        </tr>
    </thead>
    <tbody id="agendaTableBody">
        @foreach ($agendaElements as $element)
            <tr class="agenda-day">
                <td>{{ $element->prestazione->nome }}</td>
                <td>{{ $element->data }}</td>
                <td>{{ $element->orario_inizio }}</td>
                <td>{{ $element->prenotazione ? 'Sì' : 'No' }}</td>
                <td class="flex-center">
                    <form action="{{ route('agenda.show', $element->id) }}" method="GET">
                        <button type="submit" class="button-style edit-button">Modifica</button>
                    </form>
                    <form action="{{ route('agenda.delete', $element->id) }}" method="POST" onsubmit="return confirm('Sei sicuro di voler e
                        @csrf
                        @method('DELETE')
                        <button type="submit" class="button-style delete-button">Elimina</button>
                    </form>
                </td>
```

A questo punto se si clicca sul bottone modifica, si viene reindirizzati ad un metodo del controller che si occupa degli slot

```php
Route::get('/agenda/{id}', [AgendaController::class, 'show_agenda_element'])
    ->middleware('auth', 'isStaff')
    ->name('agenda.show');
```

Si andrà a ricercare all'interno del database lo slot che si vuole andare a modificare e verrà ritornata una view che ne permetterà la visualizzazione.

```php
public function show_agenda_element($id)
{
    // Trova l'elemento dell'agenda specifico
    $agendaElement = Agenda::find($id);

    if ($agendaElement) {
        return view('areaPrestazioni.agenda_element')->with('agendaElement', $agendaElement);
    } else {
        return redirect()->back()->with('error', 'Elemento dell\'agenda non trovato.');
    }
}
```

Verrà ora visualizzata un'altra pagina dove sarà possibile vedere in modo esteso tutte le informazioni relative a quello slot, dove verrà indicato anche se quello slot è già stato assegnato ad una prenotazione di un utente oppure no. Sarà inoltre possibile eliminare la prenotazione se già associata o aggiungerne una nuova se lo slot è libero.

```blade
<div class="two-by-two-grid">
    {{-- Grid Item 1: Slot Data --}}
    <div class="grid-item">
        <small>Slot dello</small> <br>
        <strong>{{ $agendaElement->data }}</strong>
    </div>

    {{-- Grid Item 2: Prestazione --}}
    <div class="grid-item">
        <small>Prestazione corrispondente</small> <br>
        <strong>{{ $agendaElement->prestazione->nome }}</strong>
    </div>

    {{-- Grid Item 3: Orario --}}
    <div class="grid-item">
        <small>Orario</small> <br>
        <strong>{{ $agendaElement->orario_inizio }}</strong>
    </div>

    {{-- Grid Item 4: Prenotato da --}}
    <div class="grid-item">
        @if (isset($agendaElement->prenotazione->cliente))
            <small>Prenotato da</small> <br>
            <strong>{{ $agendaElement->prenotazione->cliente->nome }}
                {{ $agendaElement->prenotazione->cliente->cognome }}</strong>
        @else
            <p>Non prenotato</p>
        @endif
```

```
@if (isset($agendaElement->prenotazione->cliente))
    <div class="">
        <form action="{{ route('agenda.appointment.cancel', $agendaElement->id) }}" method="POST">
            @csrf
            @method('DELETE')
            <button type="submit" class="button-style delete-button">Annulla Prenotazione</button>
        </form>
    </div>
@else
    <div class="">
        <form action="{{ route('agenda.appointment.new', $agendaElement->id) }}" method="GET">
            <button type="submit" class="submit-button button-style">Aggiungi Prenotazione</button>
        </form>
    </div>
@endif
```

Se si clicca sul bottone ‘Aggiungi Prenotazione’, si verrà reindirizzati ad un apposito metodo del controller degli slot.

```
Route::get('/agenda/new/{id}', [AgendaController::class, 'add_appointment'])
    ->middleware('auth', 'isStaff')
    ->name('agenda.appointment.new');
```

Quel determinato slot verrà ricercato all’interno del database e, se trovato, verranno prese dal database tutte le prenotazioni degli utenti per quella determinata prestazione e che non hanno indicato come data esclusa quella dello slot che si sta visualizzando. Verrà poi ritornata una view per far vedere queste informazioni.

```
public function add_appointment($id)
{
    // Trova l'elemento dell'agenda da modificare
    $agenda = Agenda::find($id);

    if ($agenda) {

        $prestazione = $agenda->idPrestazione;

        $dataAgenda = $agenda->data;

        $prenotazioni = Prenotazione::where('idPrestazione', $prestazione)
            ->where(function ($query) use ($dataAgenda) {
                $query->where('dataEsclusa', '!=', $dataAgenda)
                    ->orWhereNull('dataEsclusa');
            })
            ->get();


        return view('areaPrestazioni.agenda_add_appointment')
            ->with('agendaElement', $agenda)
            ->with('prestazione', $prestazione)
            ->with('prenotazioni', $prenotazioni);
    } else {
        return redirect()->route('agenda')->with('error', 'Elemento dell\'agenda non trovato.');
    }
}
```

Comparirà una pagina nella quale compariranno tutte le prenotazioni compatibili con quello slot e dalla quale sarà possibile selezionare la prenotazione da associare allo slot.

```blade
<table>
    <thead>
        <tr>
            <th>Nome</th>
            <th>Cognome</th>
            <th>Azione</th>
        </tr>
    </thead>
    <tbody id="agendaTableBody">
        @foreach ($prenotazioni as $prenotazione)
            <tr class="agenda-day">
                <td>{{ $prenotazione->cliente->nome }}</td>
                <td>{{ $prenotazione->cliente->cognome }}</td>
                <td>
                    <form method="POST" action="{{ route('agenda.appointment.new', ['id' => $agendaElement->id]) }}">
                        @csrf
                        <button type="submit" class="button-style submit-button">
                            Aggiungi Prenotazione
                        </button>
                        <input type="hidden" name="idPrenotazione" value="{{ $prenotazione->id }}">
                    </form>
                </td>
        @endforeach
    </tbody>
</table>
```

Se si clicca il bottone si invierà una richiesta di tipo POST che porterà ad un altro metodo del Controller.

```php
Route::post('/agenda/new/{id}', [AgendaController::class, 'add_appointment_to_agenda'])
    ->middleware('auth', 'isStaff')
    ->name('agenda.appointment.new');
```

Lo slot sarà ricercato nuovamente all'interno del database e verrà aggiornato il campo relativo alla prenotazione associata, dove verrà inserito l'id della prenotazione scelta. Sarà poi ricercato l'utente che aveva effettuato la prenotazione e verrà salvata all'interno del database un messaggio di notifica.

```php
public function add_appointment_to_agenda(Request $request, $id)
{
    // Trova l'elemento dell'agenda da modificare
    $agenda = Agenda::find($id);

    if ($agenda) {
        // Aggiorna l'ID della prenotazione nell'elemento dell'agenda
        $agenda->idPrenotazione = $request->input('idPrenotazione');
        $agenda->save();
        Log::debug("Updated agenda element with ID: $id");


        $idPrenotazione = $agenda->idPrenotazione;
        $utente = Prenotazione::find($idPrenotazione)->cliente;

        $notifica = new Notifica();
        $notifica->username = $utente->username;
        $notifica->contenuto = "L'appuntamento del " . $agenda->data . " alle " . $agenda->orario_inizio . " è stato aggiunto all'agenda.";
        $notifica->save();


        return redirect()->route('agenda')->with('success', 'Appuntamento aggiunto all\'agenda con successo.');
    } else {
        return redirect()->route('agenda')->with('error', 'Elemento dell\'agenda non trovato.');
    }
}
```

Gestione notifiche

Se un utente di livello 2 entra nell'area riservata viene visualizzato un simbolo di una campanellina che, se cliccato, mostrerà le notifiche.

```blade
@if (Auth::user()->isCliente())
    <!-- From https://docs.fontawesome.com/web/style/layer -->
    <span class="fa-layers fa-fw notification-close" id="notificationIcon"> <i class="fa-solid fa-bell"></i> </span>

    <div class="client-nofication" id="userNotificationContainer" hidden>

        <div class="flex-space-between">
            <h4>Notifiche</h4>
        </div>

        <div id="notificationList"></div>

        <div id="notificationPagination" class="notification-pagination"></div>
    </div>

    <script src="{{ asset('assets/js/userNotification.js') }}"></script>
@endif
```

Appena cliccato il bottone, se il contenitore in cui verranno visualizzate le notifiche è già aperto verrà chiuso, altrimenti verrà apero e si verrà reinderizzati ad un altro metodo che si occuperà di mostrare le notifiche.

```javascript
$('#notificationIcon').on('click', function () {
    // Check if the container is already visible
    if ($('#userNotificationContainer').is(':visible')) {
        $('#userNotificationContainer').fadeOut();
        return;
    } else {
        loadNotificationsPage(route('notifications.get'));
        $('#userNotificationContainer').fadeIn();
    }
});
```

Il metodo come prima cosa farà una richiesta ad un metodo del controller delle notifiche, il quale si occuperà di ricercare all'interno del database tutte le notifiche relative all'utente corrente e segnerà come lette quelle che già non lo erano.

```javascript
function loadNotificationsPage(url) {
    $('#notificationList').html('<p>Caricamento notifiche...</p>');
    $('#notificationPagination').html('');

    $.ajax({
        url: url,
        type: "GET",
        dataType: "json",
        success: function (response) {
```

```php
Route::get('api/notifications', [NotificationController::class, 'get_notifications'])
    ->name('notifications.get')
    ->middleware(['auth']);
```

```php
public function get_notifications(Request $request)
{
    $notifications = Notifica::where('username', $request->user()->username
        ->orderBy('created_at', 'desc')
        ->paginate(3);

    foreach ($notifications as $notification) {
        // Mark the notification as read if it is not already
        if (!$notification->letto) {
            $notification->letto = true;
            $notification->save();
        }
    }

    return $notifications;
}
```

Verranno infine aggiunte le notifiche all'interno del container HTML dedicato.

```javascript
success: function (response) {
    $('#notificationList').html('');

    // Inizio la visualizzazione delle notifiche se ce ne sono
    if (response.data && response.data.length > 0) {
        response.data.forEach(function (notification) {
            $('#notificationList').append(

                `<div class="notification-item ${notification.letto === 1 ? "" : "notification-item-new"} " id="notif-page-${notification.id}">` +
                `<p>${notification.contenuto}</p>` +
                `<small class="notification-date">${notification.created_at}</small>` +
                `</div>`
            );
        });
```