

# Documentazione progetto G.E.S.A

Alessandro Colombo 1066001, Gionatha Pirola 1066011



Corso di Progettazione, Algoritmi e Computabilità (9 CFU)

Febbraio 2023

<b>1</b>	<b>ITERAZIONE 0.....</b>	<b>4</b>
<b>1.1</b>	<b>Introduzione .....</b>	<b>4</b>
<b>1.2</b>	<b>Requisiti funzionali e analisi dei casi d'uso .....</b>	<b>4</b>
<b>1.3</b>	<b>User story .....</b>	<b>5</b>
1.3.1	Gestore dell'edificio .....	5
1.3.2	Utente (partecipante) .....	6
1.3.3	Organizzatore.....	6
1.3.4	Sistema di assegnazione stanze.....	7
<b>1.4</b>	<b>Requisiti non funzionali .....</b>	<b>7</b>
1.4.1	Manutenibilità .....	7
1.4.2	Efficienza .....	7
1.4.3	Usabilità.....	7
<b>1.5</b>	<b>Topologia del sistema .....</b>	<b>9</b>
<b>1.6</b>	<b>Design pattern .....</b>	<b>10</b>
<b>1.7</b>	<b>Toolchain e tecnologie utilizzate .....</b>	<b>11</b>
<b>2</b>	<b>ITERAZIONE 1.....</b>	<b>12</b>
<b>2.1</b>	<b>Introduzione .....</b>	<b>12</b>
<b>2.2</b>	<b>UC1: Login/Registrazione .....</b>	<b>12</b>
<b>2.3</b>	<b>UC2: Gestore Edificio .....</b>	<b>13</b>
2.3.1	Visualizza Stanze .....	13
2.3.2	Inserimento/modifica stanze .....	13
2.3.3	Eliminazione stanze.....	13
2.3.4	Visualizzazione eventi.....	13
<b>2.4</b>	<b>UC3: Organizzatore di eventi .....</b>	<b>14</b>
2.4.1	Visualizza informazioni eventi .....	14
2.4.2	Prenotare stanza per evento privato .....	14
2.4.3	Organizzare un evento pubblico .....	14
<b>2.5</b>	<b>UC4: Utente partecipante .....</b>	<b>15</b>
2.5.1	Iscriversi ad un evento pubblico .....	15
<b>2.6</b>	<b>UC5: Profilo Utente.....</b>	<b>15</b>
<b>2.7</b>	<b>UC6: Sistema Assegnazione Stanze .....</b>	<b>15</b>

2.7.1	Stanza migliore per l'orario desiderato .....	15
2.7.2	Stanza migliore in assoluto ad un orario diverso da quello desiderato .....	18
<b>2.8</b>	<b>UML Component Diagram .....</b>	<b>21</b>
<b>2.9</b>	<b>UML Class Diagram per interfacce .....</b>	<b>22</b>
<b>2.10</b>	<b>UML Class Diagram per tipo di dato .....</b>	<b>23</b>
<b>2.11</b>	<b>Deployment Diagram.....</b>	<b>24</b>
<b>2.12</b>	<b>Testing.....</b>	<b>25</b>
2.12.1	Analisi statica.....	25
2.12.2	Analisi dinamica.....	25
<b>2.13</b>	<b>Documentazione API.....</b>	<b>28</b>
<b>3</b>	<b>ITERAZIONE 2.....</b>	<b>31</b>
<b>3.1</b>	<b>Introduzione .....</b>	<b>31</b>
<b>3.2</b>	<b>UC1: Inserimento di nuove associazioni.....</b>	<b>31</b>
<b>3.3</b>	<b>UC2: Selezione di un'associazione .....</b>	<b>31</b>
<b>3.4</b>	<b>UC3: Iscrizione a nuove associazioni .....</b>	<b>32</b>
<b>3.5</b>	<b>UC4: Visualizzazione associazioni .....</b>	<b>32</b>
<b>3.6</b>	<b>UML Component Diagram .....</b>	<b>33</b>
<b>3.7</b>	<b>UML Class Diagram per interfacce .....</b>	<b>34</b>
<b>3.8</b>	<b>UML Class Diagram per tipo di dato .....</b>	<b>35</b>
<b>3.9</b>	<b>Deployment Diagram.....</b>	<b>36</b>
<b>3.10</b>	<b>Testing.....</b>	<b>37</b>
3.10.1	Analisi statica.....	37
3.10.2	Analisi dinamica.....	37
<b>3.11</b>	<b>Documentazione API.....</b>	<b>39</b>
<b>4</b>	<b>ITERAZIONE 3.....</b>	<b>41</b>
<b>4.1</b>	<b>Introduzione .....</b>	<b>41</b>
<b>4.2</b>	<b>UC1: Eliminazione evento.....</b>	<b>41</b>
<b>4.3</b>	<b>UC2: Visualizzazione eventi a cui un utente si è iscritto .....</b>	<b>41</b>
<b>4.4</b>	<b>UC3: Disiscrizione a eventi a cui un utente si è iscritto .....</b>	<b>42</b>
<b>4.5</b>	<b>UC4: Disiscrizione ad associazioni.....</b>	<b>42</b>
<b>4.6</b>	<b>UC5: Selezione di un'associazione all'inserimento di un evento .....</b>	<b>43</b>

<b>4.7</b>	<b>UML Component Diagram .....</b>	<b>44</b>
<b>4.8</b>	<b>UML Class Diagram per interfacce .....</b>	<b>45</b>
<b>4.9</b>	<b>UML Class Diagram per tipo di dato .....</b>	<b>46</b>
<b>4.10</b>	<b>Deployment Diagram.....</b>	<b>47</b>
<b>4.11</b>	<b>Testing.....</b>	<b>48</b>
4.11.1	Analisi statica.....	48
4.11.2	Analisi dinamica.....	48
<b>4.12</b>	<b>Documentazione API.....</b>	<b>50</b>
<b>5</b>	<b>ITERAZIONE FINALE .....</b>	<b>52</b>
<b>5.1</b>	<b>Conclusioni .....</b>	<b>52</b>
<b>5.2</b>	<b>Repository GitHub .....</b>	<b>52</b>
<b>5.3</b>	<b>Sviluppi futuri .....</b>	<b>52</b>
<b>5.4</b>	<b>Manuale utente.....</b>	<b>53</b>

# **1 ITERAZIONE 0**

## **1.1 Introduzione**

Il sistema che verrà sviluppato in questo progetto tratterà della gestione di una struttura che mette a disposizione diversi spazi per attività di vario genere.

Sono offerte tre tipologie di servizio:

- Affitto di stanze per eventi privati.
- Affitto di stanze per eventi pubblici.
- Iscrizione ad eventi pubblici.

Ogni utente che si registra deve specificare se e di quali associazioni fa parte (tramite una password), in modo tale da avere accesso alle attività organizzate dalla propria associazione.

Per evento pubblico quindi si intende un evento organizzato da un singolo utente ma aperto a tutti i componenti facenti parte di una delle sue medesime associazioni. Ogni evento pubblico ha un numero massimo di partecipanti e l'associazione a cui si riferisce. L'organizzatore potrà avere accesso a queste ed altre informazioni.

Per evento privato invece si intendono quelle attività legate al singolo utente organizzatore, senza possibilità di iscrizione da parte di altri utenti e non collegate alle proprie associazioni.

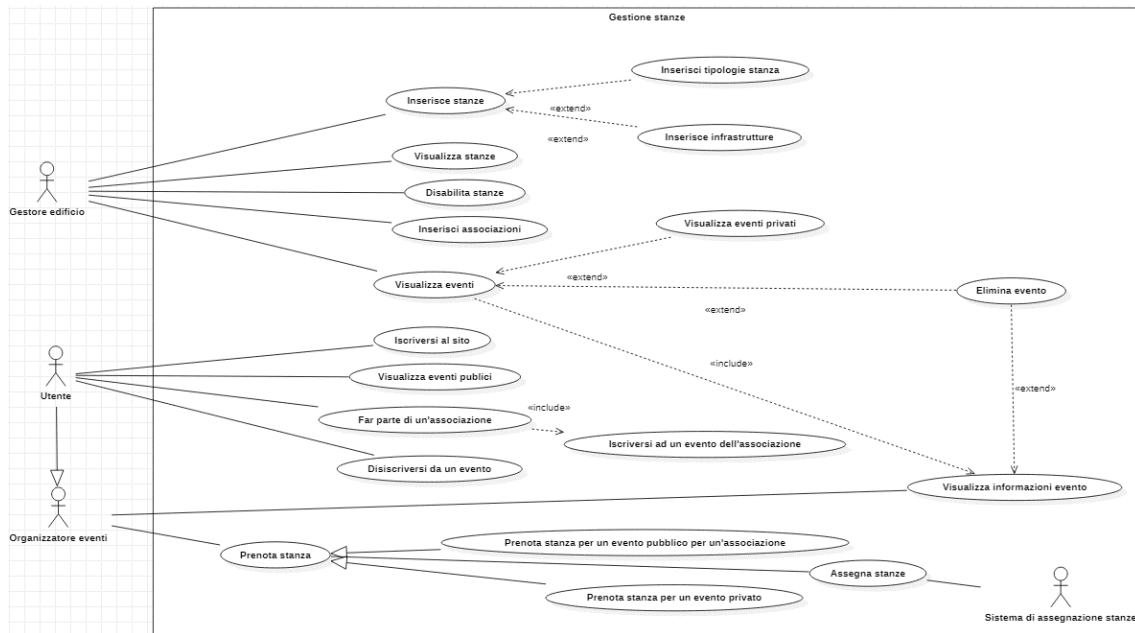
Un esempio di evento pubblico può essere un corso di un'attività sportiva, mentre un evento privato può essere una festa di compleanno con un definito numero di invitati.

Il sistema si occuperà quindi di assegnare le stanze disponibili ai diversi organizzatori in base alle varie richieste fatte, ottimizzando lo spazio e le risorse che ogni stanza offre.

## **1.2 Requisiti funzionali e analisi dei casi d'uso**

In questa sezione verranno introdotti i requisiti funzionali del sistema attraverso l'utilizzo dei casi d'uso.

Di seguito lo schema UML dei requisiti.



**Figura 1.1:** Use Case Diagram

Per un migliore sviluppo del software abbiamo diviso l'utente in tre categorie, in base alle funzioni che potranno utilizzare: si noti che l'organizzatore di un evento può essere visto come utente base per un altro evento al quale si iscrive.

Oltre alle due tipologie di utente, partecipante e organizzatore, vi è presente l'utente Amministratore, etichettato come Gestore Edificio nel diagramma: a lui è assegnata tutta la configurazione delle varie stanze a disposizione.

### 1.3 User story

Gli attori coinvolti nel sistema sono:

- Il gestore dell'edificio
- L'utente generico che si suddivide in partecipante a eventi e organizzatore.
- Il sistema di assegnazione delle stanze

#### 1.3.1 Gestore dell'edificio

Il gestore dell'edificio in quanto tale può:

- Scegliere quali stanze mettere a disposizione, inserendole a database. Le stanze hanno diverse caratteristiche che devono essere specificate durante la loro introduzione: la grandezza in metri quadri, il numero di persone che possono

accogliere, la tipologia di stanza (palestra, compleanni...), le infrastrutture disponibili (bagni, cucina, proiettore...), la posizione (aperto, chiuso), tempo di pulizia, costo orario...

- Disabilitare le stanze che per qualche motivo non sono più accessibili;
- Visualizzare un elenco di tutte le stanze con le loro caratteristiche;
- Visualizzare gli eventi previsti, sia pubblici che privati, con tutte le loro informazioni (data, luogo, numero di partecipanti...), con la possibilità di eliminazione;
- Inserire nuove associazioni a cui gli utenti potranno iscriversi (conoscendo la password).

### **1.3.2 Utente (partecipante)**

L'utente deve iscriversi al sito per poter utilizzare le varie funzionalità. Durante la sua registrazione dovrà specificare di che associazione fa parte tra quelle esistenti, se non fa parte di nessuna o registrarne una nuova. Dopo l'iscrizione avrà accesso a:

- Visualizzare eventi pubblici della sua associazione;
- Iscriversi ad eventi pubblici messi a disposizione dalle proprie associazioni;
- Iscriversi ad associazioni;
- Discriversi agli eventi.

Come utente facente parte di un'associazione, può anche essere un organizzatore di eventi pubblici.

### **1.3.3 Organizzatore**

- Organizzare eventi pubblici per la proprie associazioni, a cui altri utenti possono iscriversi se fanno parte della stessa organizzazione;
- Organizzare eventi privati;
- Visualizzare i dettagli dei propri eventi, ad esempio numero di persone iscritte, data, stanza assegnata...
- Eliminare i propri eventi.

Quando un utente organizza un nuovo evento, dovrà specificare le caratteristiche della stanza ed un orario, e un algoritmo si occuperà di proporgli la miglior stanza disponibile. L'utente potrà quindi accettare o rifiutare la stanza.

### **1.3.4 Sistema di assegnazione stanze**

Quando la richiesta di un nuovo evento viene ricevuta, un sistema confronta le caratteristiche della stanza richiesta con le stanze disponibili a tale orario, e propone una stanza all'organizzatore, che rispetti la domanda, minimizzando le aggiunte non richieste (un'eccessiva capienza, infrastrutture non obbligatorie...). All'assegnazione di una stanza viene anche calcolato il costo, sulla base della durata dell'evento, più il tempo di pulizia della stanza, per il costo orario.

Oltre a ciò, il sistema proporrà anche la stanza migliore presente nel database per le caratteristiche richieste, tralasciando l'orario.

A questo punto l'utente può scegliere se accettare una delle proposte oppure rifiutarle, magari volendo rivedere le caratteristiche richieste.

Se la stanza è accettata, non è più disponibile per la durata dell'evento più il tempo di pulizia.

Il funzionamento di questo sistema sarà spiegato dettagliatamente in seguito.

## **1.4 Requisiti non funzionali**

### **1.4.1 Manutenibilità**

L'inserimento, modifica ed eliminazione delle varie stanze saranno operazioni facilmente eseguibili dall'amministratore tramite una specifica vista del software, senza dover accedere al codice. Inoltre, si prevede l'utilizzo di un'architettura a servizi per rendere il sistema più facile da sistemare in caso di guasti, e per minimizzare i problemi che questi possono causare.

### **1.4.2 Efficienza**

Si vuole allocare le stanze in maniera efficiente ad ogni prenotazione, minimizzando lo spreco di risorse aggiuntive rispetto alla richiesta dell'utente.

### **1.4.3 Usabilità**

L'usabilità è garantita dalla decisione di sviluppare il programma tramite un insieme di pagine web raggiungibili tramite internet e quindi accessibili da qualsiasi dispositivo,

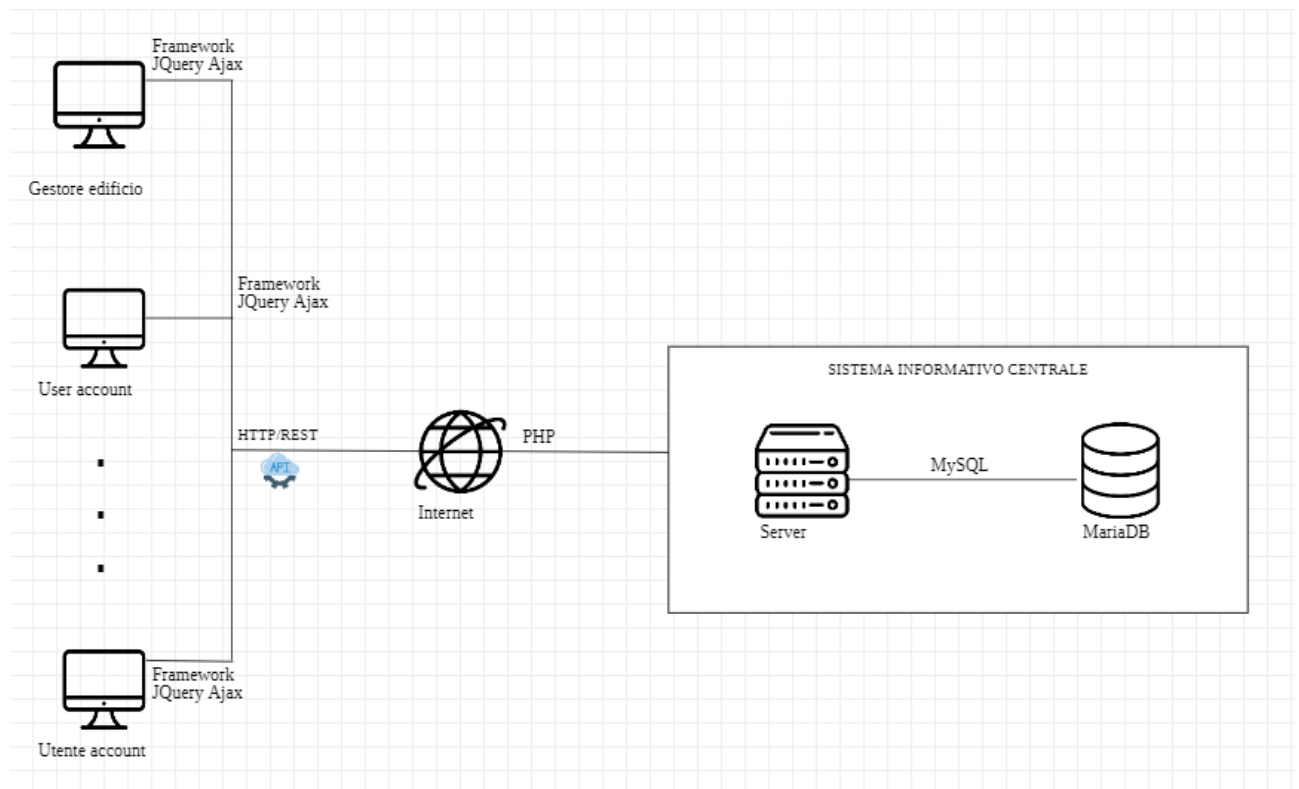


rendendo così immediata l'organizzazione, ma anche comoda e veloce l'iscrizione per gli utenti che possono verificare quali siano i nuovi eventi in qualsiasi momento.

## 1.5 Topologia del sistema

La topologia del sistema rappresenta il requisito non funzionale dell'usabilità: sia gli utenti che il proprietario del servizio potranno accedere al servizio tramite una pagina web.

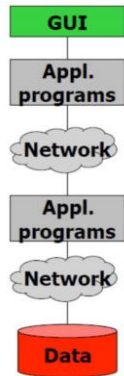
Le pagine raggiunte lato client invieranno al server richieste tramite un set di API e il protocollo HTTP/Rest. Inoltre, il server sarà collegato ad un database contenente tutte le informazioni necessarie.



**Figura 1.1:** Topologia del sistema

## 1.6 Design pattern

Useremo un'architettura 3-tier, nello specifico un'architettura di questo tipo:



Grazie a questa architettura, l'utente può fare richieste al server tramite un interfaccia web, e il server richiede i dati al database, li elabora ed invia una risposta al client, la quale verrà rappresentata graficamente per l'utente.

## 1.7 Toolchain e tecnologie utilizzate

TOOL	FUNCTION
Visual Studio Code	Sviluppo codice
Visual Studio Code <i>PHP Tools for VS Code</i>	Tool per analisi statica del codice
Git e Github	Piattaforma e Software per la condivisione e controllo delle modifiche del progetto
Google Meet, Discord	Piattaforma di comunicazione per <i>meeting</i> in remoto
Xampp	Ambiente di sviluppo PHP e web hosting locale, contenente MySQL, Apache.
Framework jQuery/Ajax	Framework per effettuare richieste HTTP/Rest.
Microsoft Word	Software per la stesura della documentazione.
Live Share	Estensione di Visual Studio Code per effettuare pair programming.
Star UML/Draw.io	Software per la realizzazione grafici e diagrammi UML.

## **2 ITERAZIONE 1**

### **2.1 Introduzione**

Nella prima iterazione mostreremo i seguenti casi d'uso, divisi per tipologia di utente.

- UC1: Login/registrazione
- UC2: Gestore Edificio
  - Visualizza stanze
  - Inserimento nuova stanza/modifica stanza esistente
  - Disabilita stanza
  - Visualizza eventi (Privati e Pubblici)
- UC3: Organizzatore di eventi
  - Visualizza informazioni eventi
  - Prenotare stanza per evento privato
  - Organizzare evento pubblico
- UC4: Partecipante
  - Partecipare ad un evento (pubblico)
- UC5: Profilo Utente
- UC6: Sistema Assegnazione Stanze

### **2.2 UC1: Login/Registrazione**

1. L'utente accede alla pagina web dell'edificio.
2. Se non è registrato, l'utente può registrarsi cliccando sull'apposito link, che lo reindirizzerà alla pagina di registrazione.
3. Qui inserisce le informazioni richieste: nome utente, password, mail, e opzionalmente l'associazione di cui fa parte.
4. Cliccando su "invia" i dati vengono inseriti nel database e l'utente viene registrato, può tornare alla pagina precedente.
5. L'utente completa un form con nome utente e password, in seguito, cliccando su "accedi" viene reindirizzato alla home page del sito.

## **2.3 UC2: Gestore Edificio**

### **2.3.1 Visualizza Stanze**

1. Dopo avere eseguito il login, l'utente "gestore" può accedere dall'homepage alla pagina di gestione delle stanze tramite un menu di navigazione.
2. Selezionando la voce corretta, viene reindirizzato ad una pagina in cui avrà un elenco completo di tutte le stanze, con tutte le loro informazioni.

### **2.3.2 Inserimento/modifica stanze**

1. Dal menu di visualizzazione delle stanze, il gestore può selezionare una stanza per passare alla visualizzazione di modifica, oppure crearne una nuova tramite l'apposito pulsante.
2. Una volta selezionata l'operazione si apre una schermata con le caratteristiche della stanza (vuota nel caso fosse una nuova stanza). Qui, il gestore può compilare i diversi campi a seconda di ciò che la stanza offre (costo, capienza...). Sempre in questa pagina il gestore può decidere se rendere una stanza non disponibile.
3. Terminata la compilazione, cliccando sul tasto "invio" il gestore può completare la modifica o inserimento della nuova stanza.

### **2.3.3 Disabilitazione stanze**

1. Dal menu di visualizzazione delle stanze, il gestore è in grado di selezionare una stanza e disabilitarla.

### **2.3.4 Visualizzazione eventi**

1. Dalla homepage, il gestore può anche accedere al calendario generale dell'edificio. In questo modo riesce ad ottenere una lista di tutti gli eventi attualmente calendarizzati.
2. Cliccando su un evento, avrà accesso alle informazioni relative a tale evento: la stanza in cui si svolgerà, orario di inizio, orario di fine, numero di partecipanti. Si noti bene che il gestore può visualizzare tutti gli eventi che avverranno nell'edificio, indipendentemente che siano pubblici o privati.

## **2.4 UC3: Organizzatore di eventi**

### **2.4.1 Visualizza informazioni eventi**

1. Dalla homepage l'utente può accedere ad una pagina dedicata alla gestione dei propri eventi.
2. Qui, ogni utente vede gli eventi che lui stesso ha organizzato con le relative informazioni, come per esempio vedere l'elenco dei nomi dei partecipanti, le caratteristiche precedentemente impostate...

### **2.4.2 Prenotare stanza per evento privato**

1. Dalla homepage l'utente può accedere ad una pagina dedicata alla creazione di nuovi eventi.
2. Selezionando su "evento privato" all'utente viene mostrato un form da compilare con i dettagli dell'evento e le caratteristiche che la stanza richiesta deve avere.
3. Una volta compilato e inviato il form, il sistema proporrà all'utente una stanza ed un prezzo, se l'utente accetta tramite l'apposito pulsante, l'evento viene registrato ed inserito nel sistema.

### **2.4.3 Organizzare un evento pubblico**

1. Dalla homepage l'utente può accedere ad una pagina dedicata alla creazione di nuovi eventi.
2. Selezionando su "evento pubblico" all'utente viene mostrato un form da compilare con i dettagli dell'evento e le caratteristiche che la stanza richiesta deve avere, inoltre deve specificare per quale organizzazione l'evento è aperto, il numero massimo di partecipanti.
3. Una volta compilato e inviato il form, il sistema propone una duplice scelta: si può selezionare una data e il sistema cerca la stanza migliore disponibile per tale data, come per gli eventi privati, oppure lasciare che sia il sistema a decidere una data in modo tale da offrire la stanza migliore e far sì che non ci siano sovrapposizioni con altri eventi.
4. A questo punto, l'utente può scegliere se confermare o rifiutare la stanza. In caso affermativo, l'evento viene registrato a database.

## **2.5 UC4: Utente partecipante**

### **2.5.1 Iscrivere ad un evento pubblico**

1. Dalla homepage l'utente può accedere ad una pagina dedicata alla visualizzazione degli eventi pubblici delle proprie associazioni.
2. Da qui l'utente può leggere le caratteristiche dei vari eventi.
3. Trovato un evento di proprio interesse, può iscriversi cliccando su "iscriviti".

## **2.6 UC5: Profilo Utente**

1. Dal menu di navigazione l'utente può accedere alla pagina del proprio profilo.
2. Qui, può vedere tutte le informazioni inserite nel momento della registrazione.
3. L'utente può da qui visualizzare tutti gli eventi a cui si è iscritto ed eventualmente annullare un'iscrizione.

## **3.7 UC6: Sistema Assegnazione Stanze**

Qui è presente la parte fondamentale del software, ovvero l'algoritmo per l'assegnazione delle stanze.

Inserendo un evento all'utente verranno mostrate due proposte, ovvero la stanza più adatta alle caratteristiche dell'evento inserito all'orario corretto, e la stanza migliore in assoluto disponibile però ad un orario diverso da quello voluto.

Ovviamente se la stanza migliore in assoluto è disponibile all'orario voluto, le due proposte coincideranno.

L'utente dovrà quindi decidere ciò che preferisce tra queste due scelte ed effettuare una prenotazione.

Di seguito vengono riportati i due algoritmi sviluppati.

### **2.7.1 Stanza migliore per l'orario desiderato**

L'algoritmo in questo caso prende in ingresso i seguenti argomenti:

- Elenco stanze (da database);
- Data e ora di inizio dell'evento inserito;
- Durata dell'evento inserito;



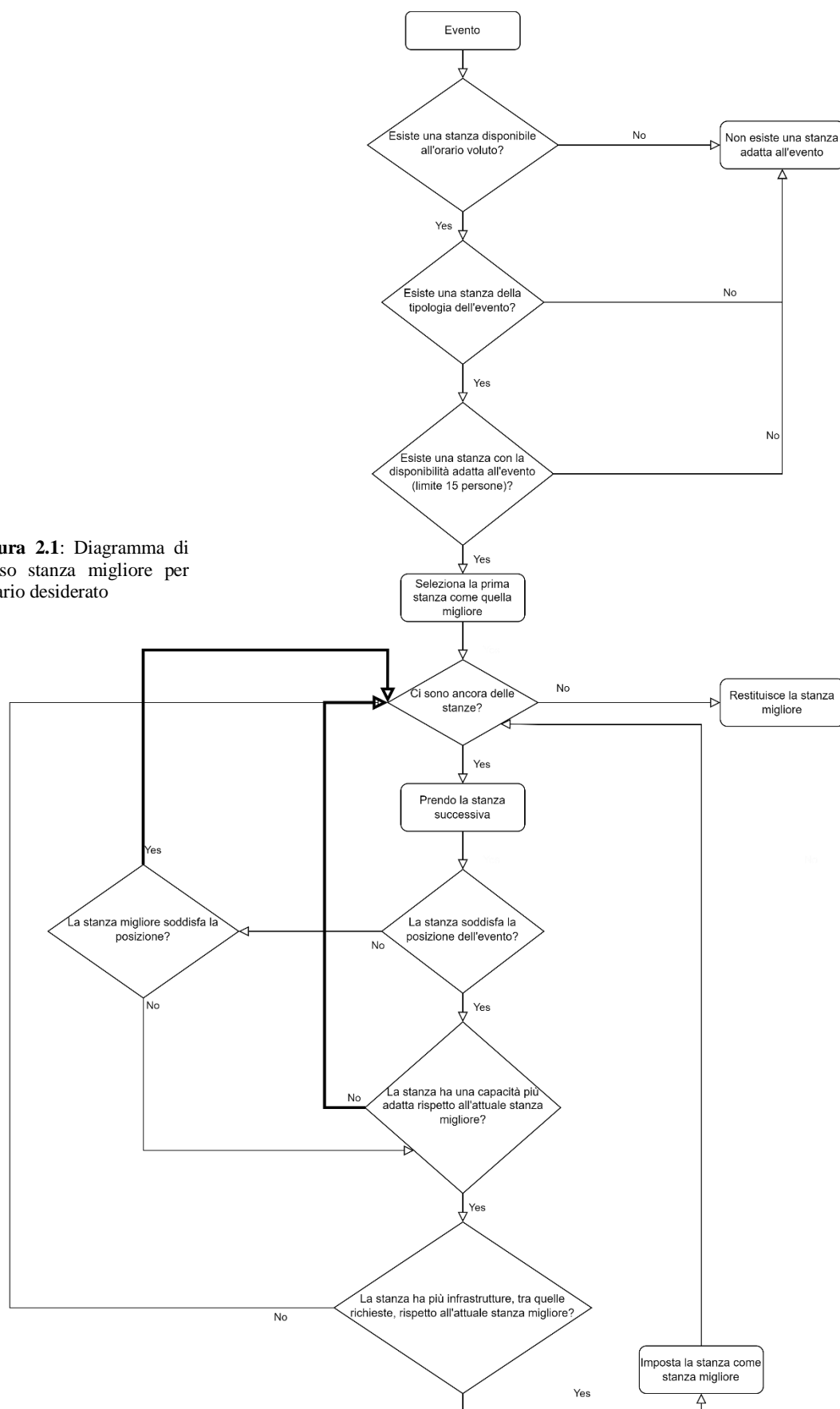
- Elenco infrastrutture dell'evento inserito;
- Numero di posti dell'evento inserito;
- Tipologia della stanza voluta per l'evento inserito;
- Posizione della stanza voluta per l'evento inserito (aperta/chiusa).

Avendo questi dati il funzionamento dell'algoritmo è il seguente:

1. Dall'elenco delle stanze vengono selezionate quelle che sono disponibili dall'orario di inizio fino alla fine dell'evento, compresi i tempi necessari per le pulizie (inserite al momento di creazione della stanza).
2. Da queste stanze avviene un ulteriore filtraggio in base alle seguenti caratteristiche, considerate fondamentali:
  - **Tipologia:** solo le stanze che rispettano la tipologia richiesta possono essere etichettate come utilizzabili per l'utente;
  - **Capienza:** vengono selezionate le stanze con capienza compresa tra capienza voluta e 15+capienza voluta.
3. Effettuata questa prima cernita, ogni stanza rimasta viene controllata per ottenere in output quella che più si avvicina alle caratteristiche desiderate dall'utente. Le caratteristiche confrontate sono le seguenti:
  - **Posizione**, ovvero stanza aperta/chiusa;
  - **Capienza**, ovvero la stanza che più si avvicina alla capienza desiderata (quella con capienza minore);
  - **Infrastrutture**, viene cercata la stanza con il numero (e il tipo) di infrastrutture che più si avvicina a quelle desiderate nell'evento.

Al termine di questa procedura, all'utente verrà mostrato in output la stanza che più si avvicina alle sue richieste nell'orario desiderato.

Per comprenderne meglio il funzionamento, di seguito il diagramma di flusso dell'algoritmo.



**Figura 2.1:** Diagramma di flusso stanza migliore per l'orario desiderato

### 2.7.2 Stanza migliore in assoluto ad un orario diverso da quello desiderato

Oltre all'algoritmo appena mostrato, il sistema propone all'utente un'ulteriore scelta, andando a selezionare, in tutto il database, la stanza migliore in assoluto per l'evento inserito dall'utente, anche se disponibile ad un orario diverso.

In questo caso l'algoritmo proposto è simile a quello precedente, con la differenza che il controllo iniziale sull'orario non viene effettuato.

Inoltre, avendo la stanza migliore è stato necessario l'implementazione di un nuovo algoritmo che permettesse di selezionare l'orario migliore per l'utente (essendo che in questo caso questo potrebbe essere diverso da quello inserito dall'utente).

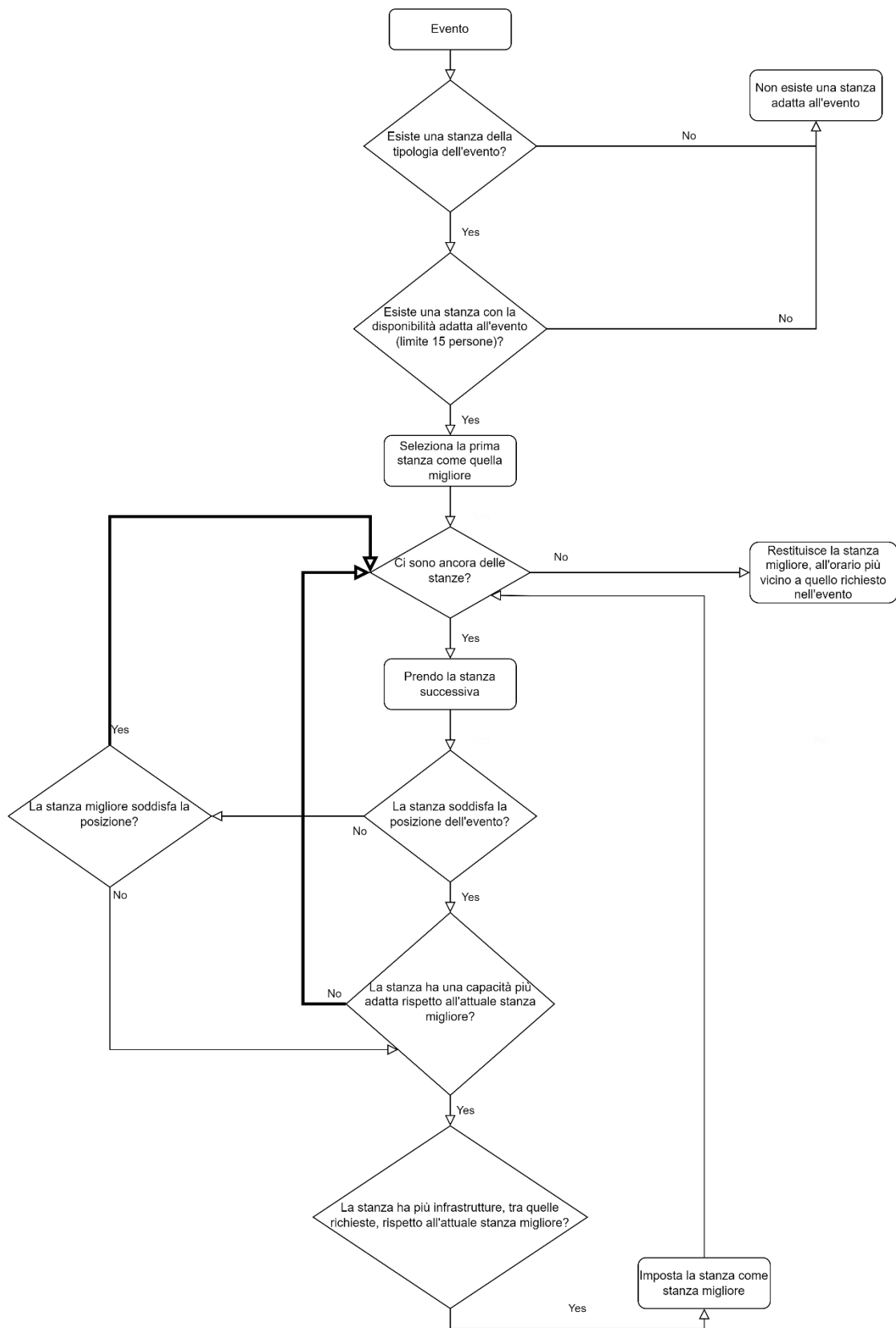
Il funzionamento di questa parte di algoritmo è il seguente:

1. Vengono presi tutti gli eventi presenti in quella determinata stanza che si svolgono dopo l'orario di inizio dell'evento inserito dall'utente;
2. Viene preso il primo evento tra quelli selezionati e viene controllato se l'evento inserito dall'utente finisce prima dell'inizio di questo:
  - Se la risposta è positiva vuol dire che la stanza migliore è disponibile all'orario voluto dall'utente. L'output generato sarà identico a quello dell'algoritmo precedente (questo vale solo alla prima iterazione);
  - Se la risposta è negativa, viene preso come orario di inizio temporaneo l'orario di fine dell'evento in questione e viene generato un nuovo orario di fine temporaneo.
3. Il passo precedente viene effettuato per tutti gli eventi, utilizzando diversi orari inizio e fine, finché non viene trovato un periodo di tempo in cui è possibile inserire l'evento.

Come ulteriore controllo, l'orario di fine (temporaneo) dell'evento non deve superare le ore 20, ovvero l'orario di chiusura della struttura.

Al termine dell'esecuzione si avrà la stanza migliore per l'evento inserito dall'utente, con l'orario più adatto (il meno lontano possibile rispetto a quello voluto).

Di seguito vengono riportati i diagrammi di flusso di questi due algoritmi implementati.



**Figura 2.2:** Diagramma algoritmo stanza migliore in assoluto, ad un orario diverso

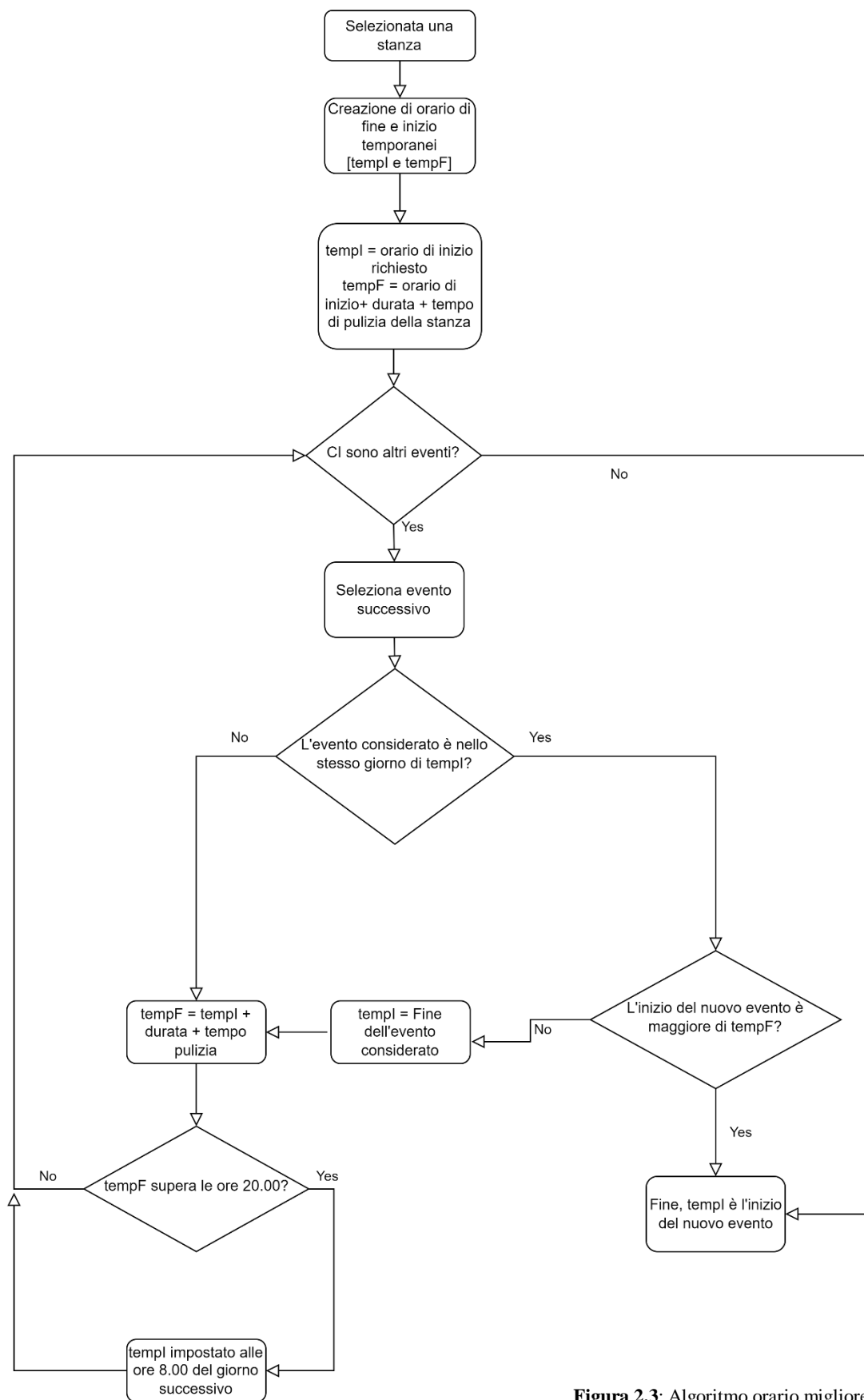


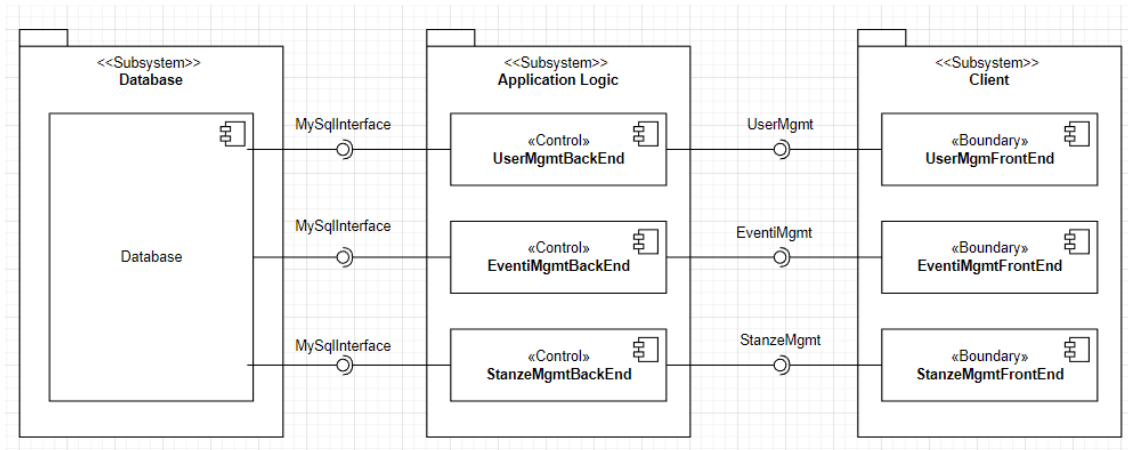
Figura 2.3: Algoritmo orario migliore

### 3.8 UML Component Diagram

Considerando i casi d'uso presi in considerazione durante questa iterazione è possibile, tramite il class diagram, rappresentare i principali elementi del sistema e studiare come questi interagiscono tra loro.

Sono stati creati due tipi di componenti:

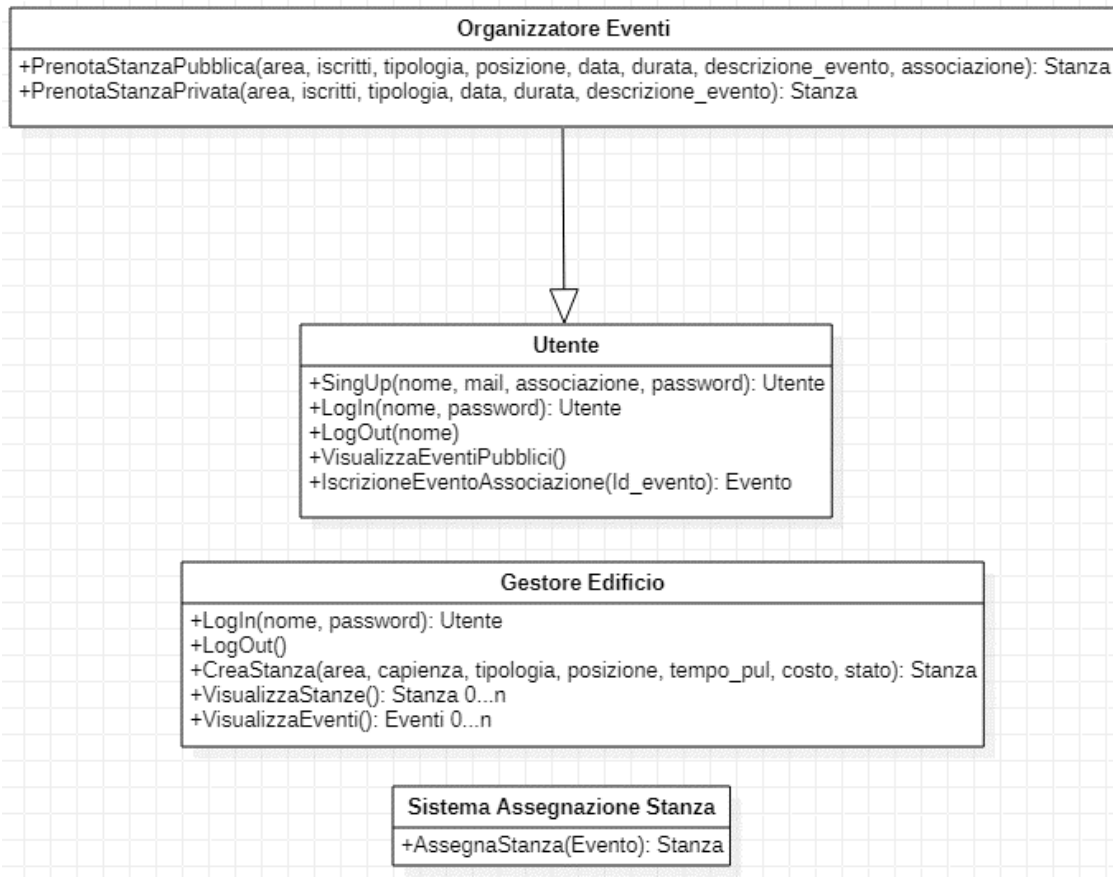
- **<<Boundary>>**: componenti lato front-end, ovvero ciò con cui l'utente si interfaccia;
- **<<Control>>**: componenti lato back-end, ovvero ciò che gestisce la logica applicativa, esponendo delle API al front-end e richiedendone a loro volta al database.



**Figura 2.4:** Component Diagram

### 3.9 UML Class Diagram per interfacce

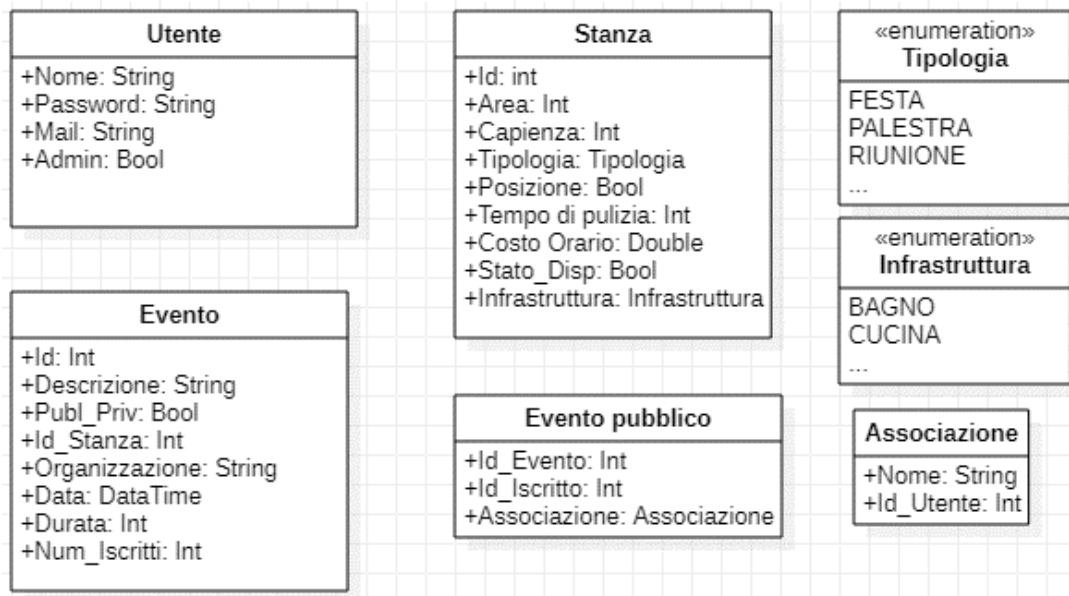
Tramite il seguente diagramma è possibile vedere le principali funzioni sviluppate durante questa iterazione, con i relativi input e output generati.



**Figura 2.5:** Class Diagram per interfacce

### 3.10 UML Class Diagram per tipo di dato

In questo diagramma vengono rappresentati i tipi di dato presenti in ogni classe del software, sui quali si basa tutto il database.

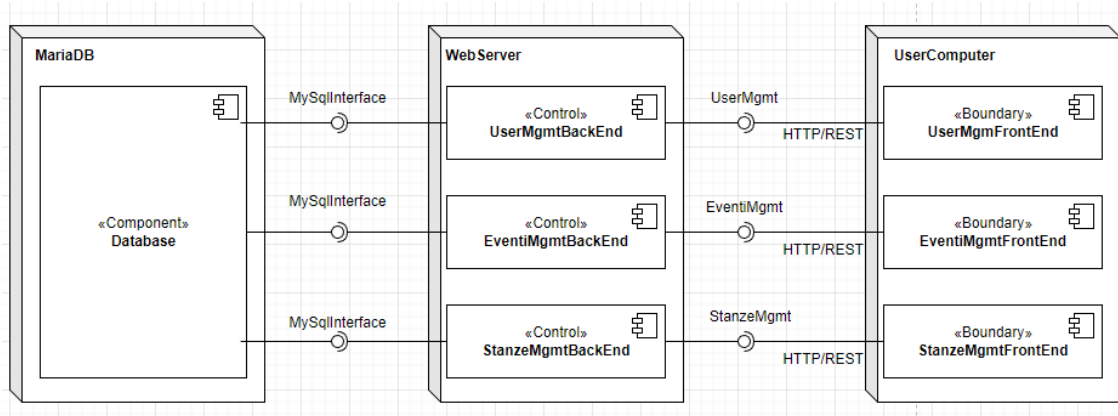


**Figura 2.6:** Class Diagram per tipo di dato



### 3.11 Deployment Diagram

Tramite l'utilizzo di un Deployment Diagram è possibile mostrare la rappresentazione hardware e software del sistema.



**Figura 2.7:** Deployment Diagram

## 3.12 Testing

### 3.12.1 Analisi statica

Per effettuare l'analisi statica del codice è stata utilizzata l'estensione fornita dall'IDE Visual Studio Code *PHP Tools for VS Code*.

La scelta dell'utilizzo di questa estensione è dovuta al fatto che tutti i file presenti all'interno del progetto sono scritti con PHP.

Ciò permette di creare una sessione ogni volta che un utente esegue l'accesso al server senza dover richiedere i dati ad ogni cambio di pagina.

### 3.12.2 Analisi dinamica

Durante questa iterazione è stato effettuato il test delle seguenti funzioni:

- Login;
- Visualizzazione stanze da parte del gestore;
- Creazione di una stanza da parte del gestore;
- Creazione di un evento da parte di un utente (con assegnazione stanza);
- Iscrizione ad eventi della propria associazione.

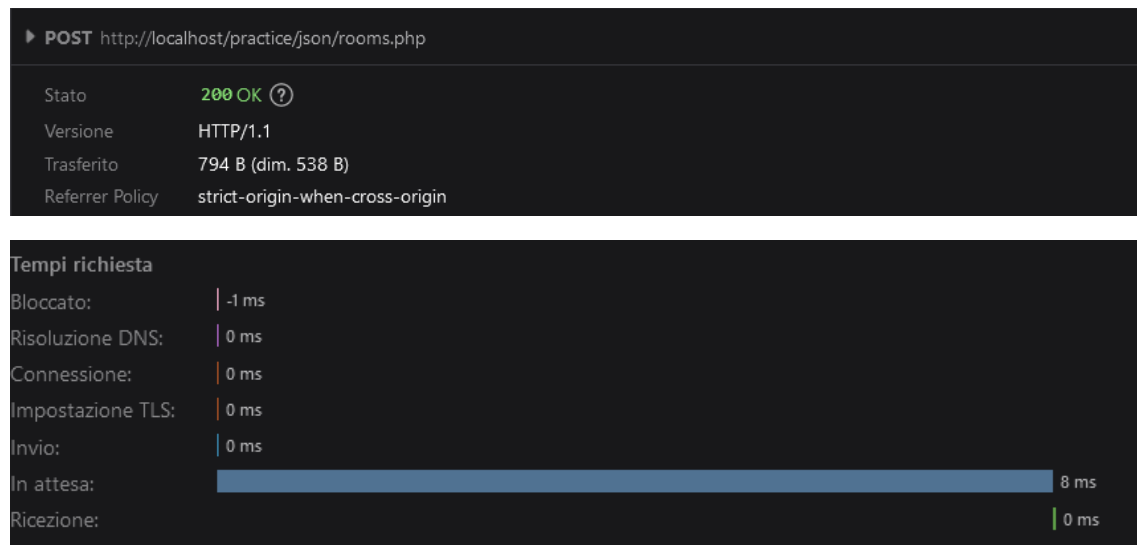
I risultati sono riportati di seguito.

A seguito del login viene ritornato lo stato 302, poiché l'utente viene reindirizzato ad una nuova pagina: l'homepage principale del programma. Negli altri casi viene invece ritornato lo stato 200 per indicare che la richiesta è andata a buon fine, e sono riportati i tempi per le diverse azioni compiute durante la richiesta.

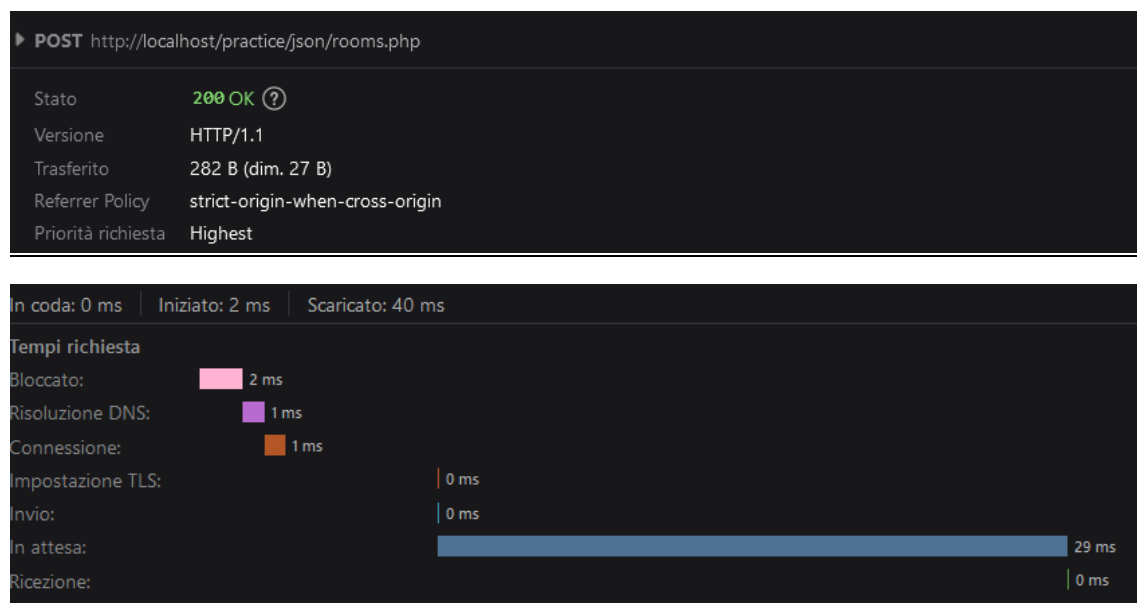
*Login:*

▶ <b>POST</b> http://localhost/practice/json/login.php	
Stato	302 Found ?
Versione	HTTP/1.1
Trasferito	8,12 kB (dim. 7,72 kB)
Referrer Policy	strict-origin-when-cross-origin
Priorità richiesta	Highest

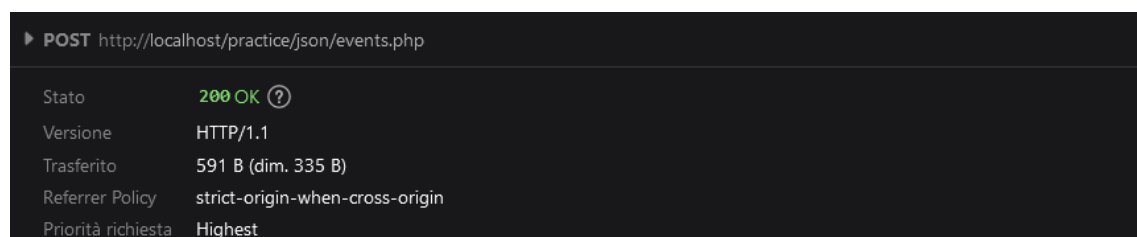
### Visualizzazione stanze da parte del gestore

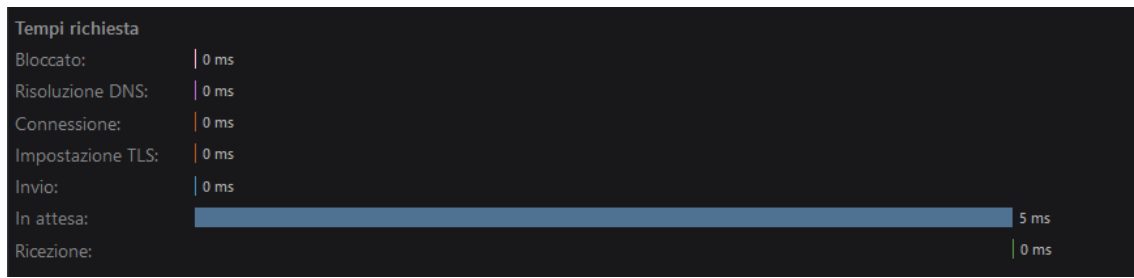


### Creazione di una stanza da parte del gestore

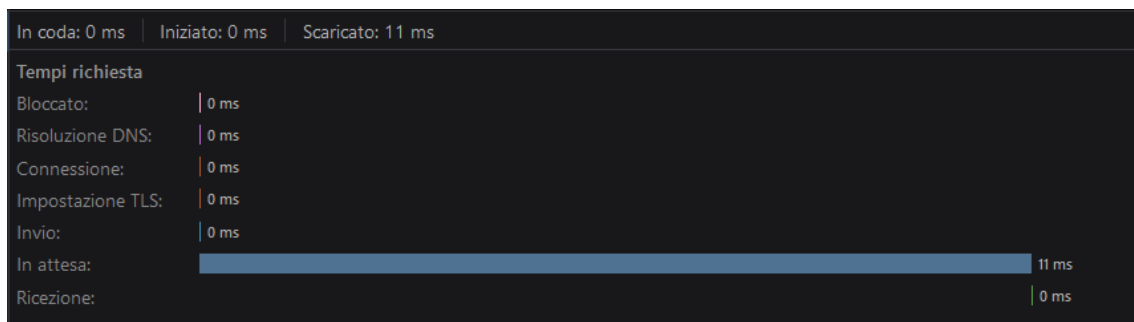
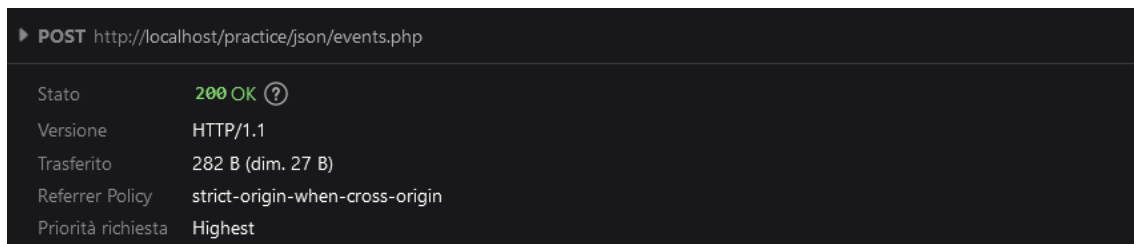


### Creazione di un evento da parte di un utente (con assegnazione stanza)





### *Iscrizione ad eventi della propria associazione*



### 3.13 Documentazione API

In questo sotto capitolo sono mostrate alcune delle API sviluppate durante Iterazione 1.

In particolare, vengono riportate:

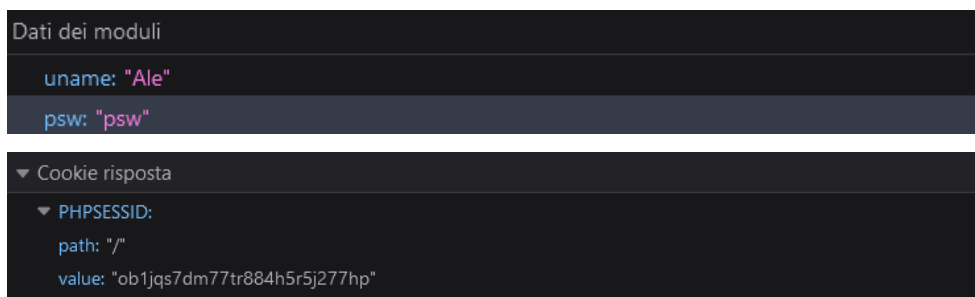
- API per il Login dell'utente (e gestore);
- API per la visualizzazione delle stanze;
- API per la creazione di una stanza;
- API per la creazione di un evento;
- API per l'iscrizione ad un evento pubblico.

Per ogni API vengono mostrati i campi passati in POST nell'immagine "dati dei moduli" e una risposta fornita dal server.

A seguito del login effettuato con successo viene creata una sessione PHP personale per l'utente, così che possa essere riconosciuto durante le successive richieste, ad esempio durante la creazione di eventi.

Le altre API invece ricevono diversi elementi JSON che sono poi utilizzati dal client per modificare la view dell'utente o per visualizzare dei messaggi, a seconda della buona riuscita o meno dell'operazione svolta.

*API per il Login dell'utente (e gestore)*



### *API per la visualizzazione delle stanze*

```
Dati dei moduli
info: "3"

JSON
▼ elementi: [ {...}, {...}, {...} ]
  ► 0: Object { id: "2", area: "60", capienza: "20", ... }
  ► 1: Object { id: "1", area: "50", capienza: "10", ... }
  ► 2: Object { id: "3", area: "15", capienza: "8", ... }
result: "ok"
errore: ""
```

### *API per la creazione di una stanza*

```
Dati dei moduli
info: "2"
tipo: "2"
area: "100"
capienza: "40"
pulizia: "10"
costo: "15"
posizione: "1"

JSON
result: "ok"
errore: ""
```

### API per la creazione di un evento

Dati dei moduli

```

info: "4"
uname: "ADMIN"
nome: "Compleanno+Ale"
data: "2023-01-31"
time: "12:00"
durata: "5"
partecipanti: "10"
public: "0"
tipo: "2"
luogo: "0"
INFR2: "on"

```

JSON

```

▼ elementi: [ {...} ]
  ► 0: Object { id: "1", area: "50", capienza: "10", ... }
▼ infrastrutture: [ {...}, {...} ]
  ► 0: Object { infr: "BAGNO" }
  ► 1: Object { infr: "CUCINA" }
result: "ok"
errore: ""

```

### API per l'iscrizione ad un evento pubblico

Dati dei moduli

```

info: "1"
uname: "GIONNY"
idEvento: "1"
nome: "COMPLEANNO+ALE"
stanza: "1"
organizz: "ALE"
data: "202301311100"
durata: "3"
partecipanti: "2"
associazione: "UNIBG"

```

JSON

```

result: "ok"
errore: ""

```

## **3 ITERAZIONE 2**

### **3.1 Introduzione**

Durante questa iterazione si è voluta implementare la gestione delle associazioni.

I casi d'uso presi in considerazione sono quindi:

- UC1: Inserimento di nuove associazioni
- UC2: Selezione di un'associazione
- UC3: Iscrizione a nuove associazioni
- UC4: Visualizzazione associazioni

### **3.2 UC1: Inserimento di nuove associazioni**

Prima di questa iterazione, un utente quando si registrava inseriva il nome di un'associazione, la quale nel caso non fosse già presente nel database, veniva inserita come nuova associazione.

Questa modalità non è ideale in quanto gli utenti possono “inventarsi” le associazioni, inoltre potrebbero inserire un nome errato della propria associazione, non avendo più modo di modificarla.

Per evitare tutti questi problemi è stata implementata questa funzione.

Il gestore dell'edificio (l'admin), tramite l'apposito tasto presente nella pagina “profilo” può inserire nuove associazioni, inserendo semplicemente il nome ed una password.

L'aggiunta della password è stata ritenuta utile al fine di evitare l'iscrizione di un utente in associazioni a cui nella realtà non è veramente iscritto.

Questa funzionalità è stata sviluppata aggiornando la tabella associazioni ad ogni trigger di questa funzione.

### **3.3 UC2: Selezione di un'associazione**

Avendo implementato la funzione per inserire nuove associazioni, ora gli utenti al momento dell'iscrizione dovranno poter scegliere un'associazione tra quelle presenti.



La vecchia modalità di inserimento manuale dell'associazione da parte dell'utente è stata quindi modificata, obbligando l'utente a selezionare un'associazione di quelle presenti.

Selezionando un'associazione verrà chiesta anche la password corrispondente (inserita dal gestore dell'edificio al momento dell'inserimento dell'associazione).

Quando un utente si iscrive al sito, non è obbligato a selezionare un'associazione, in quanto un utente potrà semplicemente iscriversi per prenotare una stanza per un evento privato.

### **3.4 UC3: Iscrizione a nuove associazioni**

Finora un utente poteva iscriversi ad una sola associazione (scelta al momento della registrazione).

È stato deciso quindi di implementare la possibilità di iscriversi a più associazioni, in quanto un utente non deve essere vincolato ad una sola associazione.

Un utente dalla pagina “profilo” potrà selezionare dalla lista di associazioni presenti nel database, una nuova associazione.

Ovviamente anche qui dopo aver selezionato l'associazione voluta, l'utente dovrà inserire la password corretta.

### **3.5 UC4: Visualizzazione associazioni**

Avendo inserito la possibilità di iscrizione a più associazioni da parte di un utente, è stato ritenuto importante inserire la possibilità di visualizzazione di tutte le associazioni a cui un utente è iscritto.

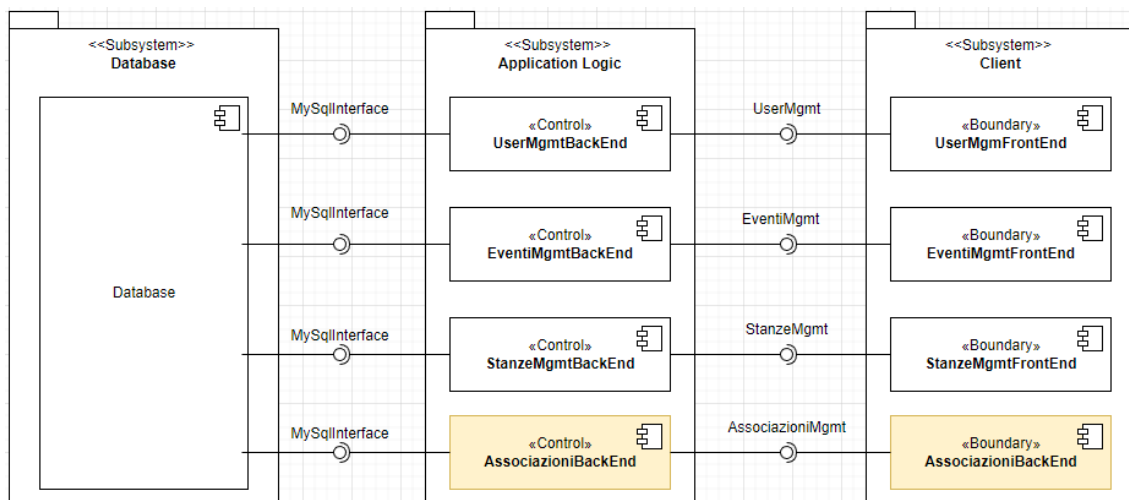
Dalla pagina “profilo”, l'utente potrà quindi avere una lista di tutte le associazioni a cui è iscritto.

### 3.6 UML Component Diagram

Al component diagram dell'iterazione 1 sono state inserite due nuovi componenti, ovvero quelle riguardanti la gestione delle associazioni.

Comunicando tra loro e il database, scambiano informazioni per gestire l'inserimento e la visualizzazione delle associazioni.

Nel diagramma sono messe in evidenza le nuove componenti.



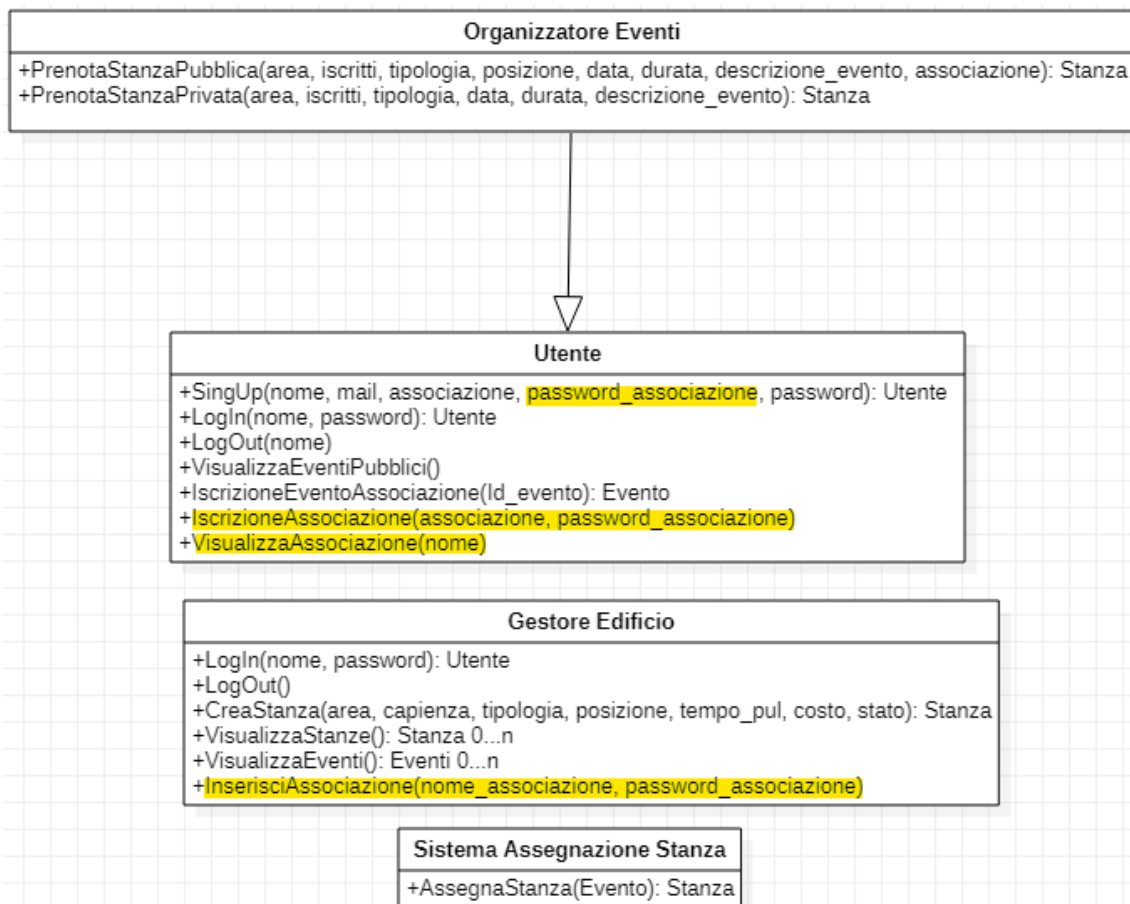
**Figura 3.1:** Component Diagram

### 3.7 UML Class Diagram per interfacce

È stato necessario aggiungere nuove operazioni al class diagram.

Le nuove operazioni inserite sono le 3 funzioni create in questa iterazione, più il campo *passwordAssociazione* nel momento di registrazione di un utente (questo campo può essere vuoto nel caso un utente non voglia iscriversi a nessuna associazione).

Le modifiche sono messe in evidenza nel diagramma delle classi di seguito.



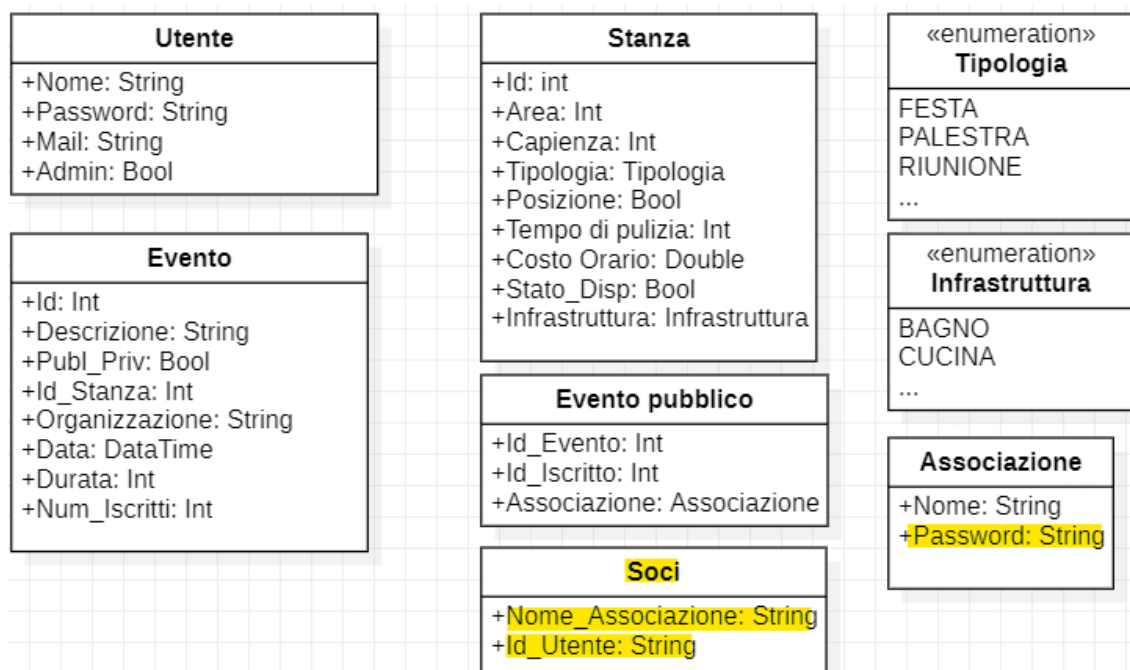
**Figura 3.2:** Class Diagram per interfacce

### 3.8 UML Class Diagram per tipo di dato

Rispetto all'iterazione precedente, qui sono state aggiunti due elementi principali, ovvero la password nelle associazioni ed una nuova tabella *Soci*.

La tabella *Soci* serve per far da tramite tra gli utenti e le associazioni, è infatti presente la lista di tutti i nomi utenti e il nome delle associazioni a cui sono iscritti.

Di seguito viene rappresentato quindi il class diagram per tipo di dato, mettendo in evidenza le aggiunte effettuate.

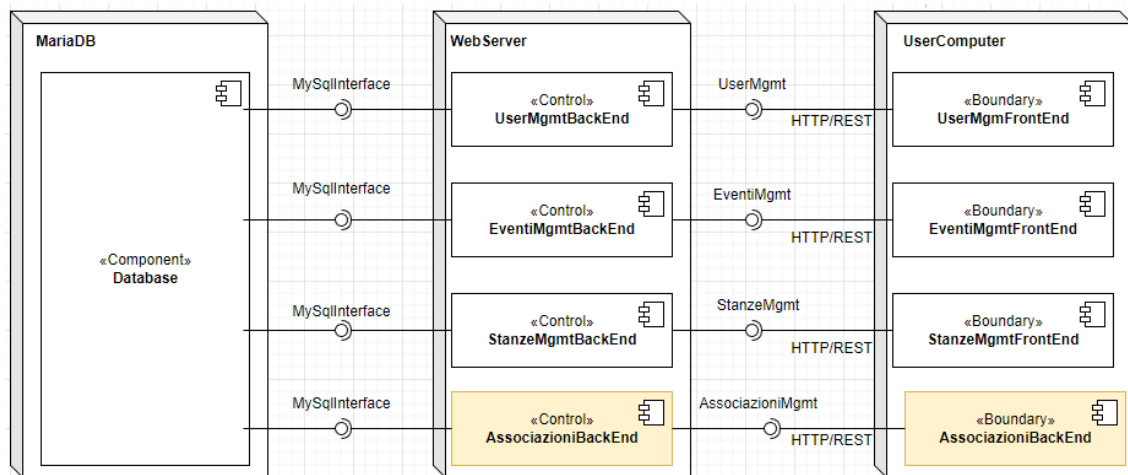


**Figura 3.3:** Class Diagram per tipo di dato

### 3.9 Deployment Diagram

Ovviamente la rappresentazione hardware e software del sistema sono simili a quelle dell'iterazione precedente, con però l'aggiunta delle associazioni.

Di seguito viene riportato il diagramma, mettendo in evidenza le aggiunte.



**Figura 3.4:** Deployment Diagram

## 3.10 Testing

### 3.10.1 Analisi statica

Per effettuare l'analisi statica del codice è stata utilizzata, come nell'iterazione precedente l'estensione fornita dall'IDE Visual Studio Code *PHP Tools for VS Code*.

Valgono quindi le considerazioni fatte in precedenza.

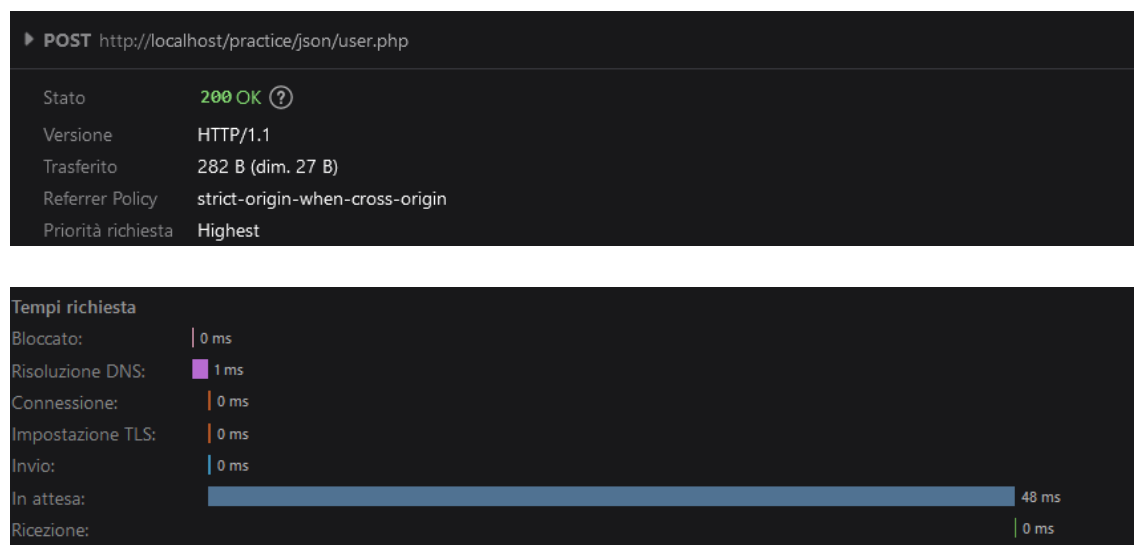
### 3.10.2 Analisi dinamica

Si è deciso di effettuare testing dinamico delle seguenti funzioni implementate in questa iterazione:

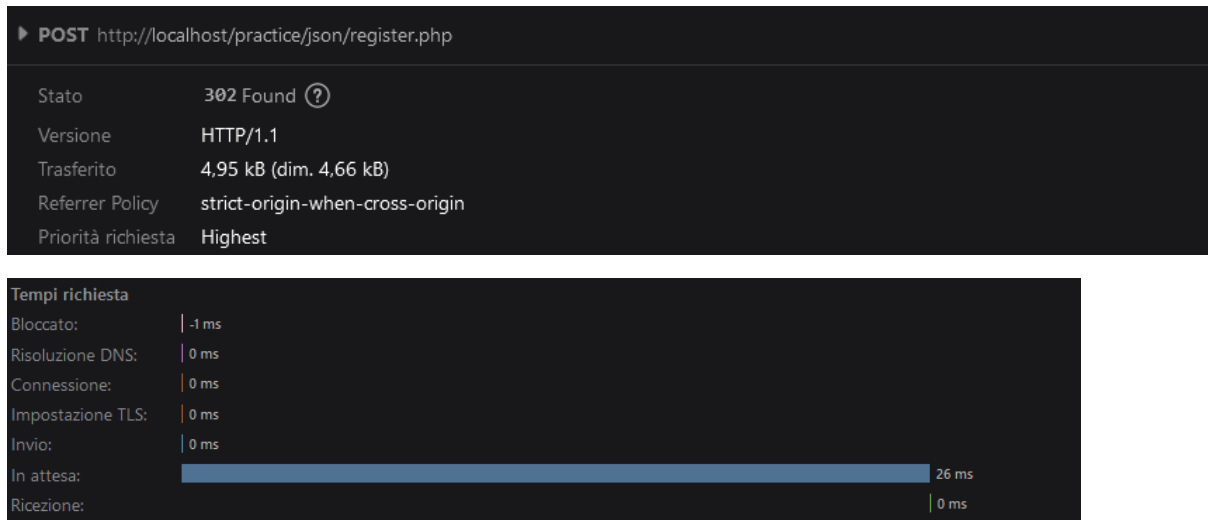
- Inserimento di nuove associazioni da parte del gestore (admin);
- Selezione di un'associazione al momento della registrazione di un utente;
- Iscrizione ad una nuova associazione da parte dell'utente inserendo psw corretta;

I risultati ottenuti sono riportati di seguito.

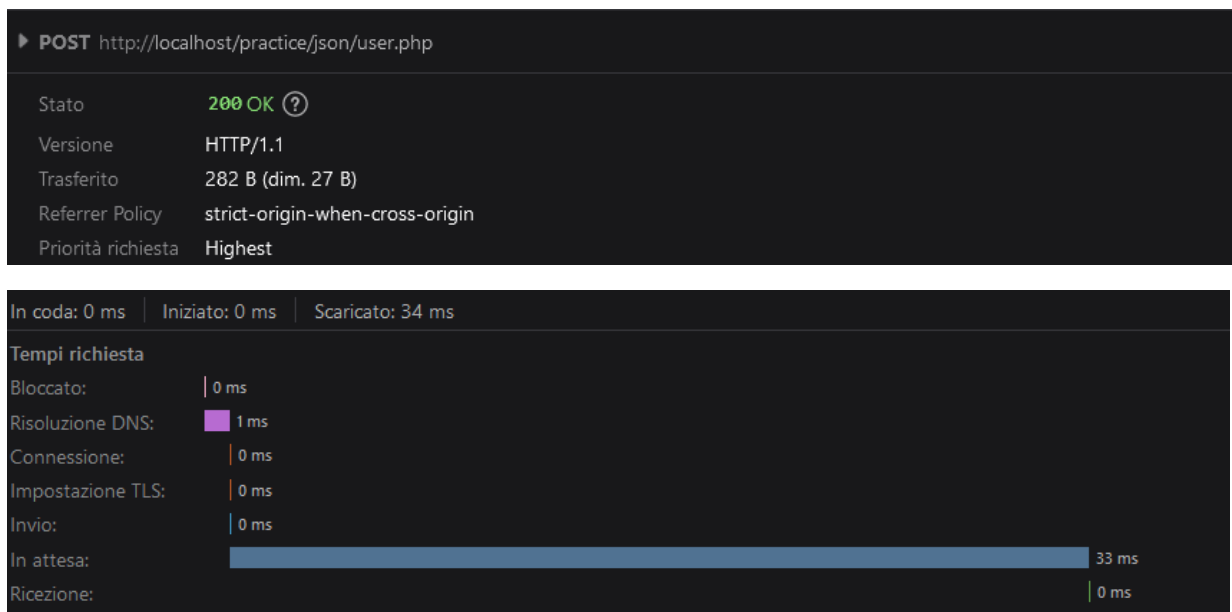
*Inserimento di nuove associazioni da parte del gestore*



*Selezione di un'associazione al momento della registrazione di un utente*



*Iscrizione ad una nuova associazione da parte dell'utente inserendo psw corretta*



### 3.11 Documentazione API

Vengono qui riportate alcune delle API implementate durante questa iterazione.

In particolare:

- API per l'inserimento di nuove associazioni da parte dell'admin;
- API per l'iscrizione ad una nuova associazione da parte dell'utente;
- API per la visualizzazione delle associazioni da parte dell'utente.

Per ogni API vengono mostrati i campi passati in POST nell'immagine "dati dei moduli" e una risposta fornita dal server.

Di seguito alcune delle API implementate.

*API per l'inserimento di nuove associazioni da parte dell'admin*

Dati dei moduli

```
info: "2"  
asso: "NUOTO"  
psw: "nuoto"
```

JSON

```
result: "ok"  
errore: ""
```

*API per l'iscrizione ad una nuova associazione da parte dell'utente*

Dati dei moduli

```
info: "3"  
asso: "NUOTO"  
uname: "ALE "  
psw: "nuoto"
```

JSON

```
result: "ok"  
errore: ""
```



*API per la visualizzazione delle associazioni da parte dell'utente*

Dati dei moduli

```
info: "1"  
uname: "ALE"
```

JSON

```
▶ elementi: [ {...}, {...} ]  
▼ associazioni: [ {...}, {...} ]  
  ▶ 0: Object { associazione: "UNIBG" }  
  ▶ 1: Object { associazione: "NUOTO" }  
result: "ok"  
errore: ""
```

## **4 ITERAZIONE 3**

### **4.1 Introduzione**

In seguito alle prime due iterazioni, dove è stata implementata la parte principale del progetto, si è proseguito con un'altra iterazione dove sono implementate nuove funzioni per aumentare l'usabilità del sistema.

Principalmente saranno implementate funzioni di gestione eventi da parte dell'utente e dal gestore stanze, aggiungendo inoltre la possibilità di disiscriversi dalle associazioni.

Nello specifico, i casi d'uso presi in considerazione per questa iterazione sono i seguenti:

- UC1: Eliminazione evento;
- UC2: Visualizzazione eventi a cui un utente si è iscritto;
- UC3: Disiscrizione a eventi a cui un utente si è iscritto;
- UC4: Disiscrizione ad associazioni;
- UC5: Selezione di un'associazione all'inserimento di un evento.

### **4.2 UC1: Eliminazione evento**

Un utente (di tipologia "organizzatore evento"), oltre a poter inserire un evento deve essere pure in grado di eliminarlo nel database.

Nella prima iterazione è stata implementata solo la funzione di inserimento di un evento. È stato deciso di permettere la cancellazione in quanto un utente può decidere di disdire la prenotazione, liberando la stanza occupato dal proprio evento.

### **4.3 UC2: Visualizzazione eventi a cui un utente si è iscritto**

Finora, all'utente era permesso solamente l'iscrizione e la visualizzazione di tutti gli eventi riguardanti le associazioni a cui è affiliato.

Si vuole quindi permettere all'utente, una volta iscritto ad uno o più eventi, di poter visualizzare tutti gli eventi a cui si è iscritto (oltre quelli a cui non si è iscritto, sempre delle proprie associazioni).

Le informazioni visualizzate saranno:

- Descrizione evento;
- Stanza assegnata all'evento;
- Nome dell'organizzatore dell'evento;
- Ora di inizio, durata e data dell'evento;
- Iscritti all'evento (nel caso in cui l'evento di riferimento è stato generato dall'utente "creatore" viene visualizzato il nome degli iscritti, altrimenti viene visualizzato solo il numero degli iscritti).

Questa funzione verrà implementata nella pagina "profilo", ovvero la stessa pagina dove può visualizzare le informazioni utente (nome utente...).

#### **4.4 UC3: Disiscrizione a eventi a cui un utente si è iscritto**

Collegato al caso d'uso precedente, dopo che l'utente si iscrive ad eventi delle proprie associazioni ottenendo una vista di tutto questo sottoinsieme, è stato deciso di implementare la funzione che permetta all'utente di disiscriversi ad eventi.

Questa funzione è stata ritenuta importante per aumentare l'usabilità del sistema.

L'utente potrà effettuare la disiscrizione ad un evento a cui è iscritto dalla stessa pagina di visualizzazione di questi eventi.

#### **4.5 UC4: Disiscrizione ad associazioni**

Durante l'iterazione due, è stata inserita la possibilità da parte dell'utente di iscriversi a più associazioni.

Si è pensato però che l'utente dovrebbe avere anche la possibilità di disiscriversi da un'associazione.

Questa funzione è disponibile nella schermata "profilo", dove è presente anche la visualizzazione di tutte le associazioni.

Passando con il mouse sopra una specifica associazione, questa cambierà aspetto per una più facile comprensione della funzione.

Cliccando verrà attivata questa funzione.

#### **4.6 UC5: Selezione di un'associazione all'inserimento di un evento**

Durante questa iterazione è stata aggiunta la possibilità di selezionare un'associazione al momento di creazione di un evento pubblico.

Questa funzione è necessaria in quanto, un utente, una volta inserito un evento pubblico deve selezionare a quale evento fa riferimento, altrimenti il software lo inserirà nel calendario di tutti gli utenti iscritti alle associazioni dell'utente "creatore".

Questa selezione si svolge al momento di inserimento di un evento grazie ad un menù a tendina contenete tutte le associazioni a cui l'utente è iscritto.

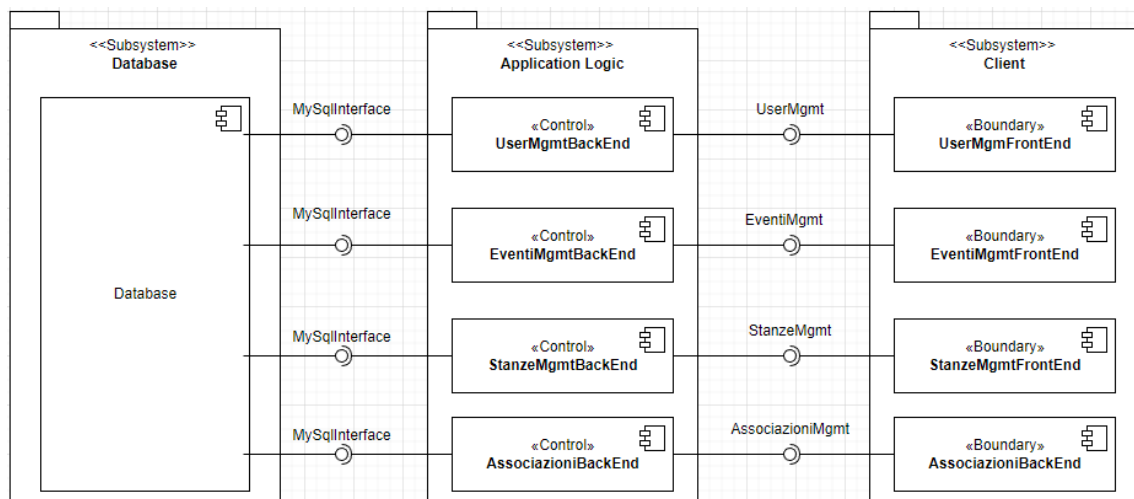
## 4.7 UML Component Diagram

Dato che i casi d'uso implementati in questa iterazione sono presenti all'interno dei componenti considerati nelle scorse iterazioni, il diagramma dei componenti è identico a quello dell'iterazione precedente.

Infatti, le nuove funzionalità riguardano gli eventi, rappresentati tramite il componente *EventMgm*.

Le funzionalità riguardanti le associazioni sono invece presenti nel componente *AssociazioniMgm*.

Il diagramma è quindi il seguente.



**Figura 4.1:** Component Diagram

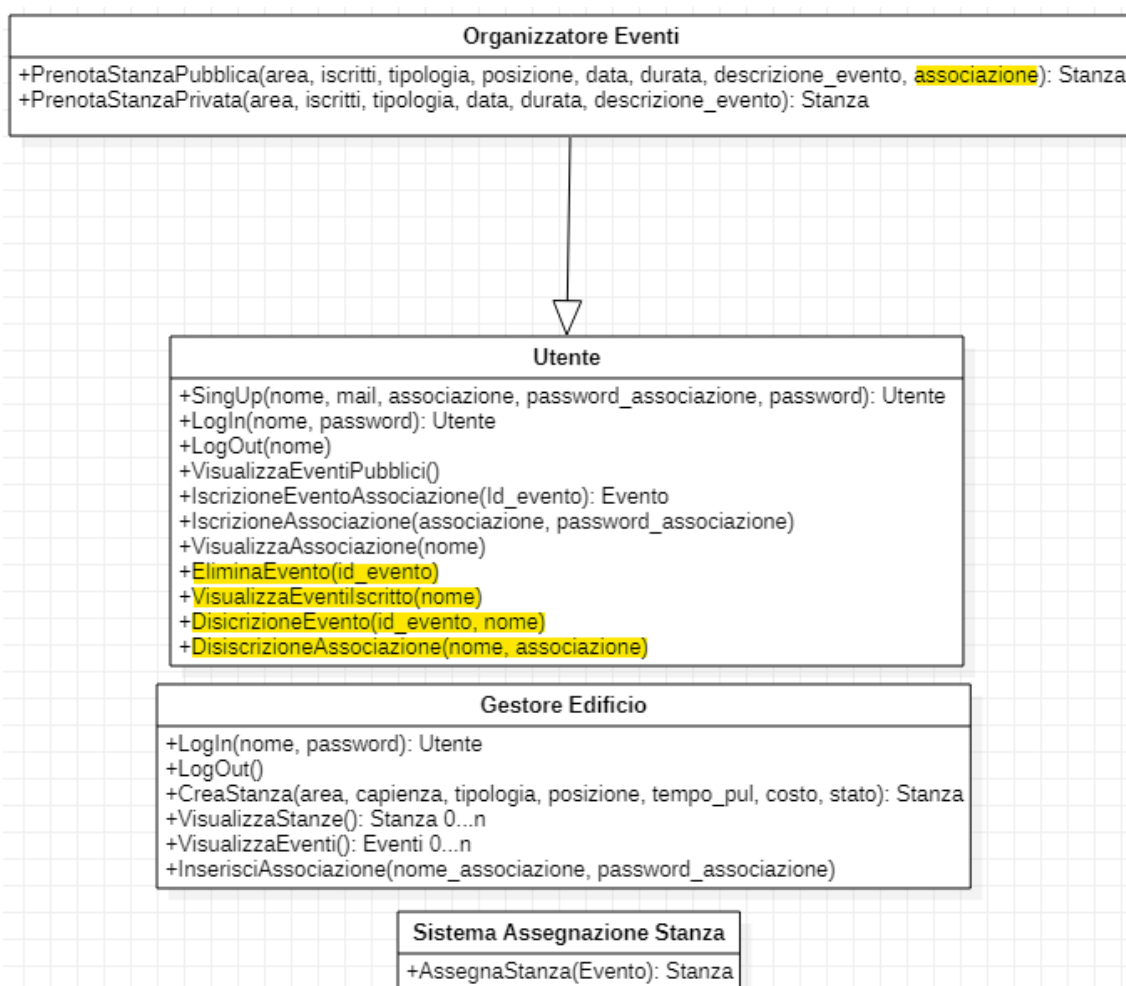
## 4.8 UML Class Diagram per interfacce

In questo caso il diagramma presenta delle aggiunte rispetto al Class Diagram dell'iterazione precedente.

Sono state inserite le 3 nuovi funzioni che rappresentano i casi di studio sviluppati durante questa iterazione.

È stata inoltre evidenziato il campo associazione in quanto, in seguito a questa iterazione all'utente è permesso di selezionare l'associazione per la quale è organizzato l'evento.

Il diagramma delle classi è il seguente, dove le nuove funzioni sono messe in evidenza.

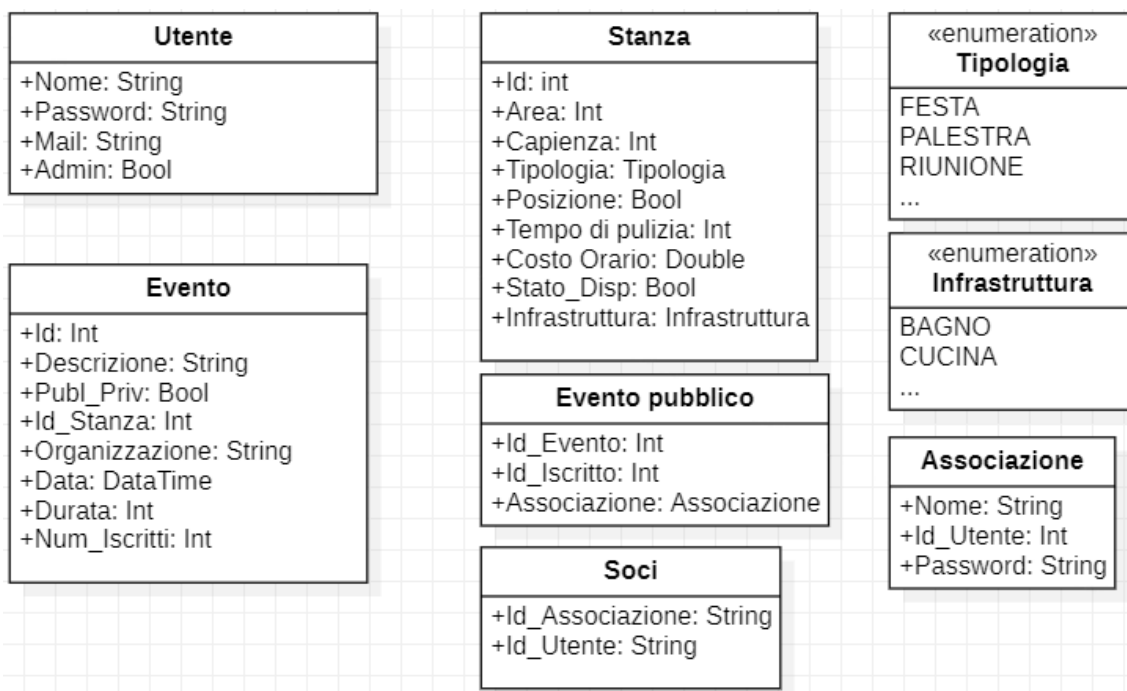


**Figura 4.2:** Class Diagram per interfacce

## 4.9 UML Class Diagram per tipo di dato

Anche in questo diagramma non sono state apportate modifiche rispetto all'iterazione uno, in quanto le funzioni implementati non modificano o aggiungono nessun dato nel database.

Il diagramma viene riportato di seguito.



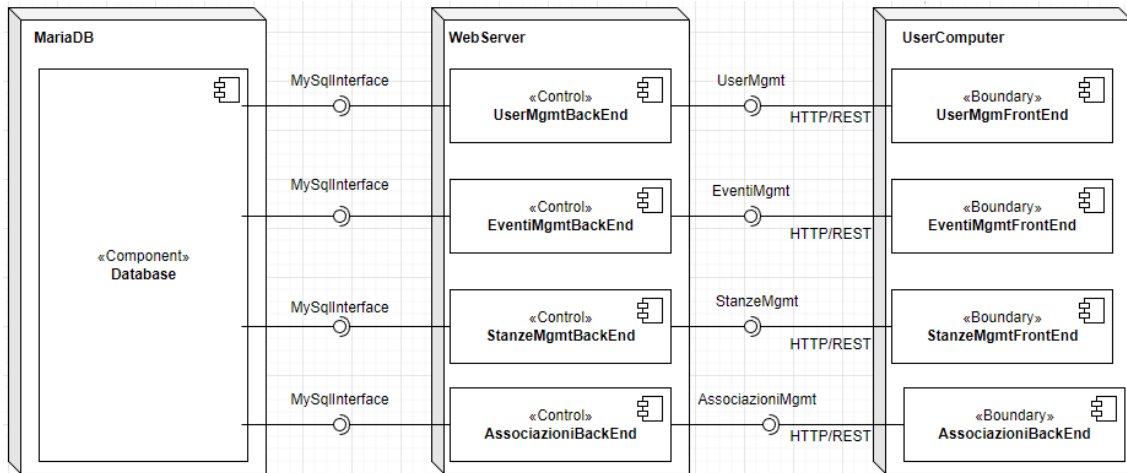
**Figura 4.3:** Class Diagram per tipo di dato

## 4.10 Deployment Diagram

Come per il class diagram, anche qui non ci sono stati cambiamenti.

Questo è dovuto al fatto che i casi d'uso considerati in questa iterazione sono già inclusi nei componenti esistenti.

Per completezza, il diagramma viene comunque riportato.



**Figura 4.4:** Deployment Diagram



## 4.11 Testing

### 4.11.1 Analisi statica

Sempre per le stesse motivazioni delle iterazioni precedente, per l'analisi statica si è deciso di utilizzare l'estensione fornita dall'IDE Visual Studio Code *PHP Tools for VS Code*.

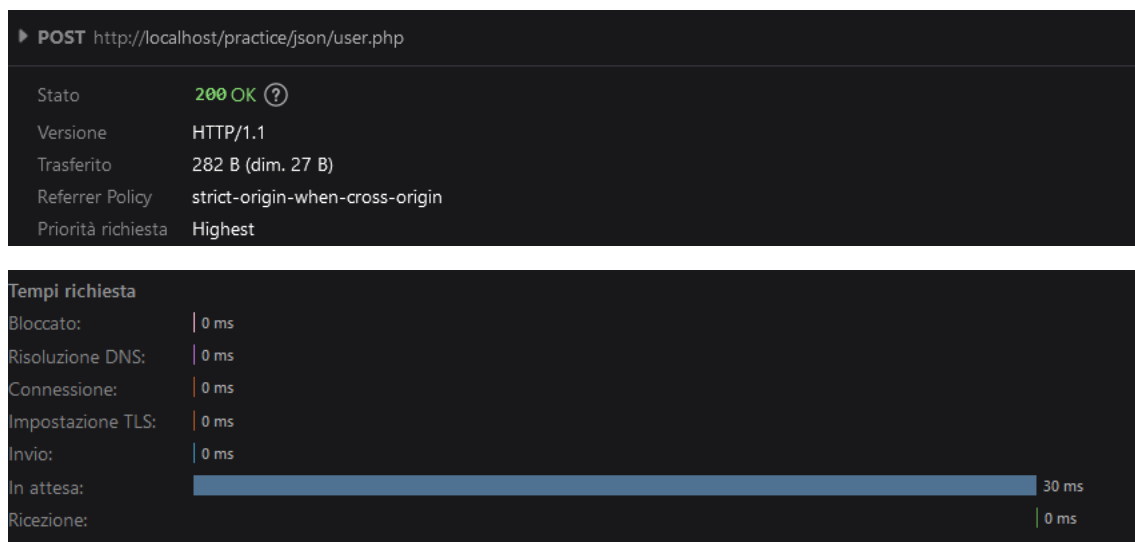
### 4.11.2 Analisi dinamica

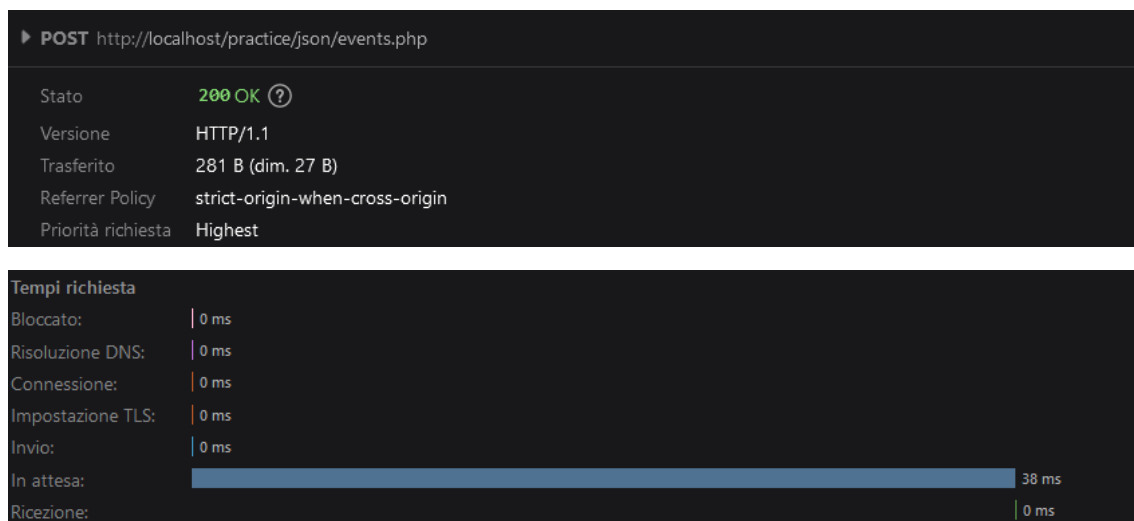
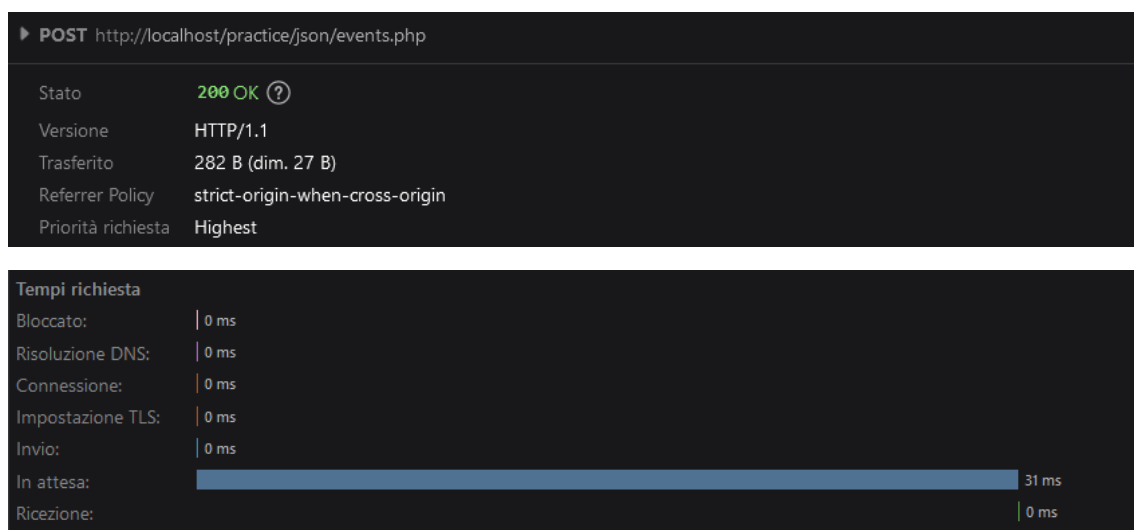
Essendo un'iterazione incentrata sull'implementazione di nuove funzioni, si è deciso di testare le seguenti componenti:

- Eliminazione di un evento da parte dell'admin;
- Disiscrizione da un evento da parte dell'utente;
- Disiscrizione ad un'associazione da parte dell'utente.

I risultati vengono quindi riportati di seguito.

*Eliminazione di un evento da parte dell'admin*



*Disiscrizione da un evento da parte dell'utente**Disiscrizione ad un'associazione da parte dell'utente*

## 4.12 Documentazione API

Come nelle iterazioni precedenti, per ultime vengono mostrate le API sviluppate in questa iterazione.

Nello specifico:

- API per l'eliminazione di un evento da parte dell'admin;
- API per la disiscrizione ad un evento da parte dell'utente;
- API per la disiscrizione ad associazioni da parte dell'utente.

Di seguito vengono riportati i risultati ottenuti.

*API per l'eliminazione di un evento da parte dell'admin*

Dati dei moduli

```
info: "4"  
uname: "ALE "  
asso: "NUOTO"
```

JSON

```
result: "ok"  
errore: ""
```

*API per la disiscrizione ad un evento da parte dell'utente*

Dati dei moduli

```
info: "8"  
idEvento: "2"  
uname: "ALE"
```

JSON

```
result: "ok"  
errore: ""
```

*API per la disiscrizione ad associazioni da parte dell'utente*

Dati dei moduli

info: "7"

idEvento: "4"

JSON

result: "ok"

errore: ""

## 5 ITERAZIONE FINALE

### 5.1 Conclusioni

Il risultato delle iterazioni precedenti ha portato ad avere un prodotto abbastanza completo, dove è possibile gestire stanze, eventi, associazioni e utenti di un possibile edificio.

Tutto lo sviluppo ha seguito il *processo di sviluppo agile*, suddividendo le fasi di implementazioni e ottenendo un programma funzionante alla fine di ogni iterazione.

Ogni iterazione si è infatti divisa in una prima fase di brainstorming in cui si andava a decidere le componenti da sviluppare in quella iterazione, procedendo in seguito nell'implementazione raffinando le idee avute a inizio iterazione.

Al termine dell'implementazione di ogni iterazione sono stati eseguiti dei test sulle funzioni implementate per accertarsi che le funzioni implementate e le API esposte fossero corrette.

### 5.2 Repository GitHub

Tutto il codice relativo al software è stato fin da subito inserito in un repository *GitHub*.

Ciò ha permesso di tener traccia dei cambiamenti effettuati, di agevolare lo scambio di file e le operazioni da svolgere al passare del tempo.

Nel repository si trovano tutti i diagrammi rappresentati nella documentazione in ogni iterazione, il codice ed il file di documentazione completo.

### 5.3 Sviluppi futuri

Eventuali sviluppi futuri potrebbero essere:

- Verifica della mail, da utilizzare come metodo di conferma;
- Possibilità di pagamento online;
- Possibilità di inserire immagini della stanza da parte dell'admin.
- Possibilità di personalizzare le stanze prenotate (aggiungendo infrastrutture...)

Queste sono solo alcune idee, le quali in futuro potrebbero essere sviluppate aggiungendole al software già sviluppato in questo progetto.

## 5.4 Manuale utente

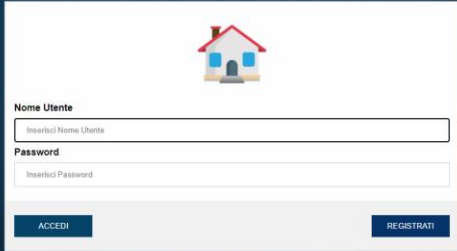
Tramite il file presente nel repository GitHub è possibile creare automaticamente sul proprio server un nuovo database con tutte le tabelle, impostate correttamente.

Una volta create le tabelle, l'admin dovrà popolare il database inserendo nuove stanze, associazioni...

Eseguite queste operazioni, è possibile mettere online il sito, permettendo agli utenti di registrarsi ed utilizzare le funzionalità implementate durante il progetto.

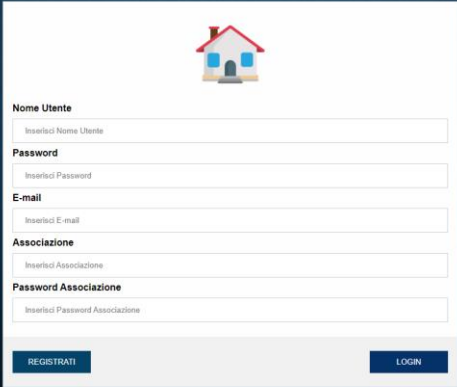
Di seguito vengono riportate le interfacce sviluppate, con una breve descrizione sulle funzionalità presente in ognuna.

### *Login*

The image shows a login form centered on a dark blue background. The form is a white rectangle with a small house icon at the top center. Below the icon, there are two input fields: the first is labeled 'Nome Utente' and the second is labeled 'Password'. Both fields have placeholder text 'Insert User Name' and 'Insert Password' respectively. At the bottom of the form, there are two buttons: 'ACCEDI' on the left and 'REGISTRATI' on the right.

Da questa pagina è possibile effettuare il login di un utente (o admin) esistente.

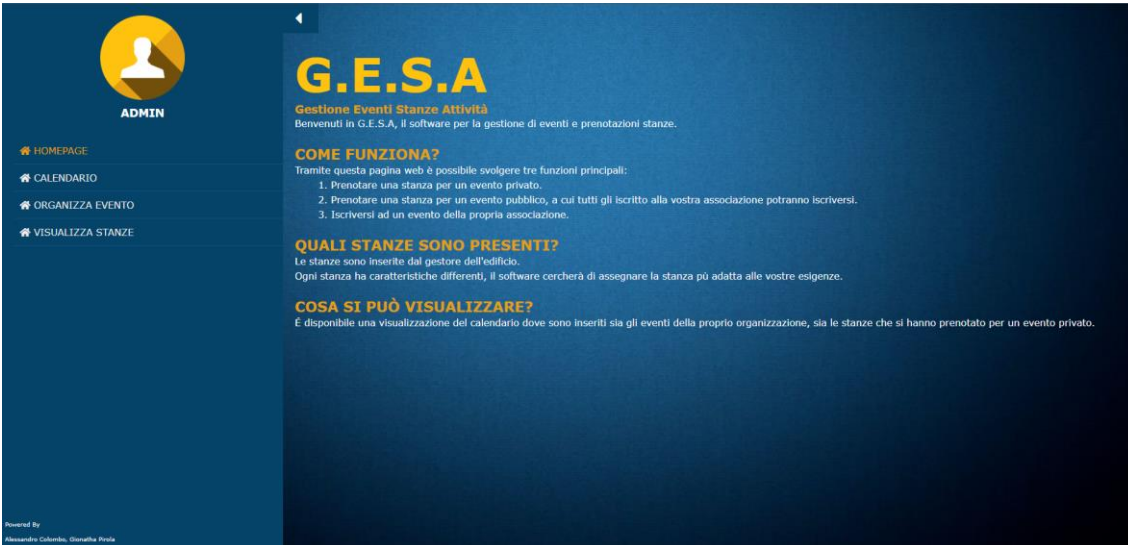
## Registrazione



The registration form is centered on a dark blue background. It features a small house icon at the top. Below the icon, there are six input fields with labels: 'Nome Utente', 'Password', 'E-mail', 'Associazione', 'Password Associazione', and a 'REGISTRATI' button. The 'REGISTRATI' button is highlighted in a lighter blue color.

Da questa pagina è possibile registrarsi come nuovo utente.

## HomePage



The Home Page of the G.E.S.A. system is displayed. It features a dark blue background with a sidebar on the left containing a user profile icon and the word 'ADMIN'. The main content area includes the G.E.S.A. logo, a welcome message, and three sections: 'COME FUNZIONA?' (How it works), 'QUALI STANZE SONO PRESENTI?' (Which rooms are present), and 'COSA SI PUÒ VISUALIZZARE?' (What can be visualized). The sidebar also contains a menu with links to 'HOME PAGE', 'CALENDARIO', 'ORGANIZZA EVENTO', and 'VISUALIZZA STANZE'.

**ADMIN**

- HOME PAGE
- CALENDARIO
- ORGANIZZA EVENTO
- VISUALIZZA STANZE

**G.E.S.A.**  
**Gestione Eventi Stanze Attività**  
 Benvenuti in G.E.S.A., il software per la gestione di eventi e prenotazioni stanze.

**COME FUNZIONA?**  
 Tramite questa pagina web è possibile svolgere tre funzioni principali:

1. Prenotare una stanza per un evento privato.
2. Prenotare una stanza per un evento pubblico, a cui tutti gli iscritti alla vostra associazione potranno iscriversi.
3. Iscriversi ad un evento della propria associazione.

**QUALI STANZE SONO PRESENTI?**  
 Le stanze sono inserite dal gestore dell'edificio. Ogni stanza ha caratteristiche differenti, il software cercherà di assegnare la stanza più adatta alle vostre esigenze.

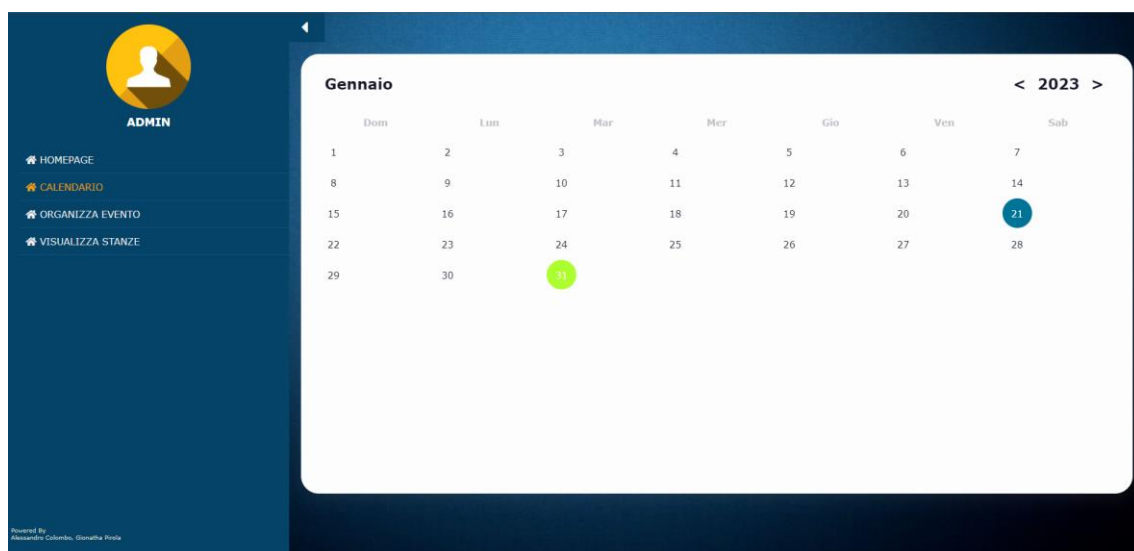
**COSA SI PUÒ VISUALIZZARE?**  
 È disponibile una visualizzazione del calendario dove sono inseriti sia gli eventi della propria organizzazione, sia le stanze che si hanno prenotato per un evento privato.

Powered By  
 Alessandro Colombo, Gianluca Rosta

## OrganizzaEvento

Da qui è possibile inserire un evento, il software proporrà due stanze seguendo l'algoritmo progettato durante la prima iterazione.

## VisualizzaEventi



Da questa pagina è possibile visualizzare tutti gli eventi organizzati oppure quelli a cui è possibile iscriversi (organizzati per la propria associazione).



L'admin da questa pagina può invece visualizzare tutti gli eventi presenti nel database.

### VisualizzaStanze

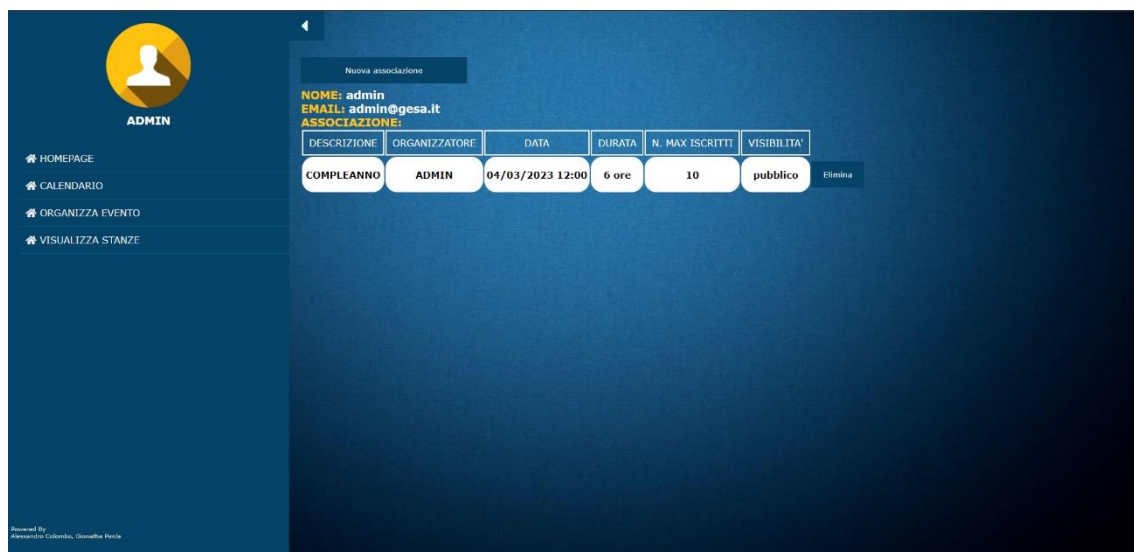
ELENCO STANZE								
Nuova stanza			Nuova infrastruttura		Nuova tipologia stanza			
MODIFICA	ID: 1	AREA: 50 m²	CAPIENZA: 10 persone	TIPO: festa	POSIZIONE: chiuso	PULIZIA: 5 ore	COSTO: 20 \$	STATUS: DISPONIBILE
MODIFICA	ID: 2	AREA: 60 m²	CAPIENZA: 20 persone	TIPO: palestra	POSIZIONE: chiuso	PULIZIA: 7 ore	COSTO: 25 \$	STATUS: DISPONIBILE
MODIFICA	ID: 3	AREA: 15 m²	CAPIENZA: 8 persone	TIPO: riunione	POSIZIONE: chiuso	PULIZIA: 1 ore	COSTO: 5 \$	STATUS: DISPONIBILE
MODIFICA	ID: 4	AREA: 100 m²	CAPIENZA: 40 persone	TIPO: festa	POSIZIONE: aperto	PULIZIA: 10 ore	COSTO: 15 \$	STATUS: NON DISPONIBILE

Questa vista è disponibile solo all'admin.

Da qui l'admin può:

- Visualizzare le stanze presenti nel database;
- Modificare le stanze presenti nel database;
- Inserire nuove stanze;
- Inserire nuove infrastrutture;
- Inserire nuove tipologie di stanze.

## Profilo



The screenshot shows a web application interface for a user profile. On the left is a dark blue sidebar with a user icon and the name 'ADMIN'. Below the icon are menu items: 'HOME PAGE', 'CALENDARIO', 'ORGANIZZA EVENTO', and 'VISUALIZZA STANZE'. The main content area has a dark blue background. At the top, it says 'Nuova associazione'. Below that, the user's details are listed: 'NOME: admin', 'EMAIL: admin@gesa.it', and 'ASSOCIAZIONE:'. A table below displays a list of associations with columns: DESCRIZIONE, ORGANIZZATORE, DATA, DURATA, N. MAX ISCRITTI, and VISIBILITA'. The first row shows 'COMPLEANNO' as the description, 'ADMIN' as the organizer, '04/03/2023 12:00' as the date, '6 ore' as the duration, and '10' as the maximum number of registrants. The visibility is set to 'pubblico'. An 'Elimina' button is located to the right of the row. At the bottom left of the sidebar, it says 'Powered by Alessandro Colombo, Gianluca Pirella'.

DESCRIZIONE	ORGANIZZATORE	DATA	DURATA	N. MAX ISCRITTI	VISIBILITA'
COMPLEANNO	ADMIN	04/03/2023 12:00	6 ore	10	pubblico

Da qui è possibile visualizzare tutte le informazioni riguardanti il profilo.

Nello specifico oltre le informazioni inserite al momento della registrazione, è possibile visualizzare:

- Tutti gli eventi a cui si è iscritti (con la possibilità di disiscriversi);
- Tutte le associazioni a cui si è iscritti (con la possibilità di disiscriversi);
- Tutti gli eventi organizzati dall'utente stesso (con la possibilità di eliminarli).