**Documentazione progetto G.E.S.A**

Alessandro Colombo 1066001, Gionatha Pirola 1066011



Corso di Progettazione, Algoritmi e Computabilità (9 CFU)

Gennaio 2023

[1 ITERAZIONE 0 3](#_Toc124790080)

[1.1 Introduzione 3](#_Toc124790081)

[1.2 Requisiti funzionali e analisi dei casi d’uso 3](#_Toc124790082)

[1.3 User story 4](#_Toc124790083)

[1.3.1 Gestore dell’edificio 4](#_Toc124790084)

[1.3.2 Utente (partecipante) 5](#_Toc124790085)

[1.3.3 Organizzatore 5](#_Toc124790086)

[1.3.4 Sistema di assegnazione stanze 5](#_Toc124790087)

[1.4 Requisiti non funzionali 6](#_Toc124790088)

[1.4.1 Manutenibilità 6](#_Toc124790089)

[1.4.2 Efficienza 6](#_Toc124790090)

[1.4.3 Usabilità 6](#_Toc124790091)

[1.5 Topologia del sistema 7](#_Toc124790092)

[1.6 Design pattern 8](#_Toc124790093)

[1.7 Toolchain e tecnologie utilizzate 9](#_Toc124790094)

[2 ITERAZIONE 1 10](#_Toc124790095)

[2.1 Introduzione 10](#_Toc124790096)

[2.2 UC1: Login/Registrazione 10](#_Toc124790097)

[2.3 UC2: Gestore Edificio 11](#_Toc124790098)

[2.3.1 Visualizza Stanze 11](#_Toc124790099)

[2.3.2 Inserimento/modifica stanze 11](#_Toc124790100)

[2.3.3 Eliminazione stanze 11](#_Toc124790101)

[2.3.4 Visualizzazione eventi 11](#_Toc124790102)

[2.4 UC3: Organizzatore di eventi 12](#_Toc124790103)

[2.4.1 Visualizza informazioni eventi 12](#_Toc124790104)

[2.4.2 Prenotare stanza per evento privato 12](#_Toc124790105)

[2.4.3 Organizzare un evento pubblico 12](#_Toc124790106)

[2.5 UC4: Utente partecipante 13](#_Toc124790107)

[2.5.1 Iscriversi ad un evento pubblico 13](#_Toc124790108)

[2.6 UC5: Profilo Utente 13](#_Toc124790109)

[2.7 UC6: Sistema Assegnazione Stanze 13](#_Toc124790110)

[2.8 UML Component Diagram 14](#_Toc124790111)

[2.9 UML Class Diagram per interfacce 15](#_Toc124790112)

[2.10 UML Class Diagram per tipo di dato 16](#_Toc124790113)

[2.11 Deployement Diagram 17](#_Toc124790114)

[2.12 Testing 18](#_Toc124790115)

[2.12.1 Analisi statica 18](#_Toc124790116)

[2.12.2 Analisi dinamica 18](#_Toc124790117)

[2.13 Documentazione API 21](#_Toc124790118)

[3 ITERAZIONE 2 24](#_Toc124790119)

[3.1 Introduzione 24](#_Toc124790120)

[3.2 UC1: Inserimento di nuove associazioni 24](#_Toc124790121)

[3.3 UC2: Selezione di un’associazione 24](#_Toc124790122)

[3.4 UC3: Iscrizione a nuove associazioni 25](#_Toc124790123)

[3.5 UC4: Visualizzazione associazioni 25](#_Toc124790124)

[3.6 UML Component Diagram 26](#_Toc124790125)

[3.7 UML Class Diagram per interfacce 27](#_Toc124790126)

[3.8 UML Class Diagram per tipo di dato 28](#_Toc124790127)

[3.9 Deployement Diagram 29](#_Toc124790128)

[3.4 Testing 30](#_Toc124790129)

[4 ITERAZIONE 3 31](#_Toc124790130)

[4.1 Introduzione 31](#_Toc124790131)

[4.2 UC1: Eliminazione evento 31](#_Toc124790132)

[4.3 UC2: Visualizzazione eventi a cui un utente si è iscritto 31](#_Toc124790133)

[4.4 UC3: Disiscrizione a eventi a cui un utente si è iscritto 32](#_Toc124790134)

[4.5 UML Component Diagram 33](#_Toc124790135)

[4.6 UML Class Diagram per interfacce 34](#_Toc124790136)

[4.7 UML Class Diagram per tipo di dato 35](#_Toc124790137)

[4.8 Deployement Diagram 36](#_Toc124790138)

[4.9 Testing 37](#_Toc124790139)

1. ITERAZIONE 0
   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, 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 della sua medesima associazione. Ogni evento pubblico ha un numero massimo di partecipanti e date limite entro il quale è possibile iscriversi. L’organizzatore potrà avere accesso a queste e 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 alla propria associazione.

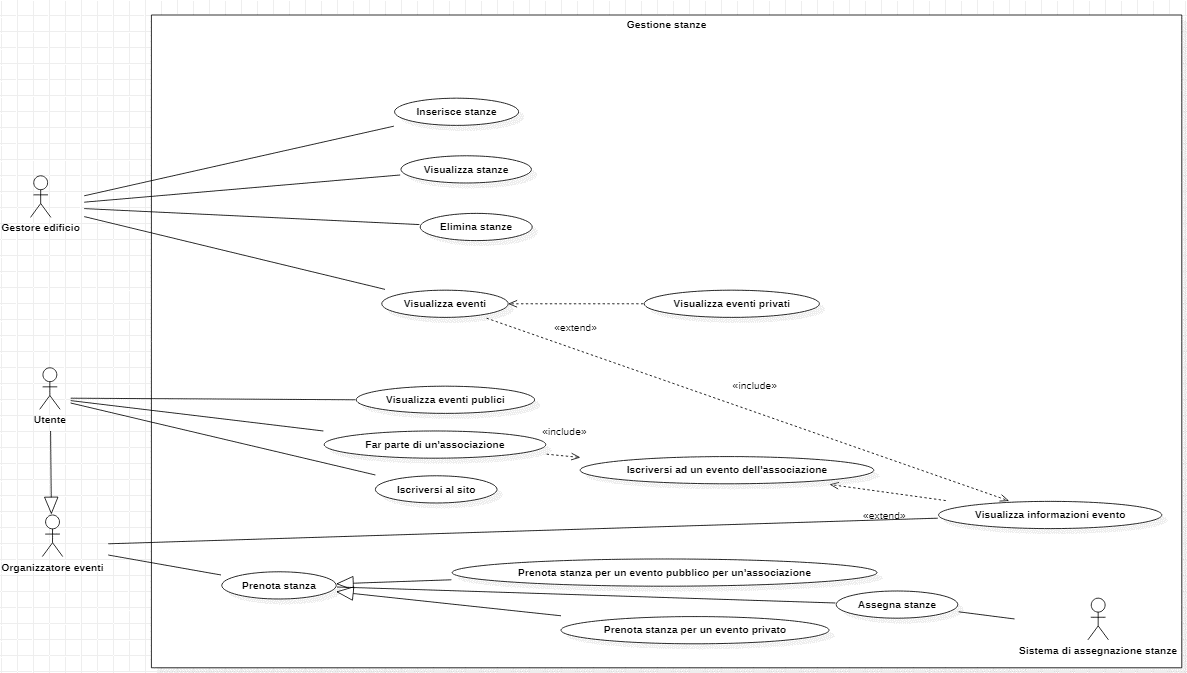
Un esempio di evento pubblico può essere un corso di un’attività sportiva, mentre un evento privato può essere una festa di compleanno con un definito numero di invitati.

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. 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. User story

Gli attori convolti 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

## 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…
* Eliminare 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…).

## 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 dalla propria associazione.

Come utente facente parte di un’organizzazione, può anche essere un organizzatore eventi.

## Organizzatore

* Organizzare eventi pubblici per la propria associazione, 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…

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.

## 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. A questo punto l’utente può scegliere se accettare la stanza o rifiutarla, 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.

* 1. Requisiti non funzionali

## 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.

## Efficienza

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

## 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. Topologia del sistema

La topologia del sistema rappresenta il requisito non funzionale dell’usabilità: sia gli utenti che il proprietario 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 tutti le informazioni necessarie.



**Figura 1.**Errore. Nel documento non esiste testo dello stile specificato.**.1:** Topologia del sistema

* 1. Design pattern

Useremo un’architettura Model-View-Presenter, 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. Toolchain e tecnologie utilizzate

|  |  |
| --- | --- |
| **TOOL** | **FUNCTION** |

|  |  |
| --- | --- |
| Visual Studio Code | Sviluppo codice |
| Notepad++ | Sviluppo codice |
| Git e Github | Piattaforma e Software per la condivisione e controllo delle modifiche del progetto |
| Google Meet | Piattaforma di comunicazione per *meeting* 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. |

1. ITERAZIONE 1
   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
* Elimina 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
  1. UC1: Login/Registrazione

1. L’utente accede alla pagina web dell’edificio.
2. Se non è registrato, l’utente può registrarti cliccando sull’apposito link, che lo reindirizzerà alla pagina di registrazione.
3. Qui inserisce le informazioni richieste: nome utente, password, mail, associazioni di cui fa parte (selezionabili da un elenco).
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.
   1. UC2: Gestore Edificio

## 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 il loro stato (disponibile o non disponibile).

## 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 Eliminazione stanze

1. Dal menu di visualizzazione delle stanze, il gestore è in grado di selezionare una stanza ed eliminarla. In questo modo, la stanza non è solo disabilitata ma viene rimossa completamente dal database.

## 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.
   1. UC3: Organizzatore di eventi

## Visualizza informazioni eventi

Dalla homepage l’utente può accedere ad una pagina dedicata alla gestione dei propri eventi.

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 e la data ultima di iscrizione. È possibile anche inserire una breve descrizione.
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 si che non ci siano sovrapposizioni con eventi della stessa organizzazione.
4. A questo punto, l’utente può scegliere se confermare o rifiutare la stanza. In caso affermativo, l’evento viene registrato a database.
   1. UC4: Utente partecipante

## 2.5.1 Iscriversi 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 “partecipa”.
4. Da un'altra apposita vista, l’utente può vedere tutti gli eventi a cui si è iscritto e eventualmente annullare un’iscrizione.
   1. UC5: Profilo Utente
5. Dal menu di navigazione l’utente può accedere alla pagina del proprio profilo.
6. Qui, può vedere tutte le informazioni inserite nel momento della registrazione.
   1. UC6: Sistema Assegnazione Stanze

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

Ad ogni caratteristica della stanza e dell’evento è assegnata un fattore di importanza. Il primo controllo viene fatto sulla tipologia di stanza. In seguito avviene un controllo sulla capienza e sulla posizione (aperto/chiuso), privilegiando le stanze che rispettano il vincolo della posizione e che si avvicinano di più alla capienza. Per ultimo si controllano le infrastrutture.

Quando un utente inserirà un evento, il sistema gli proporrà in automatico una stanza in base alla stanza con le caratteristiche più adatte all’evento inserito.

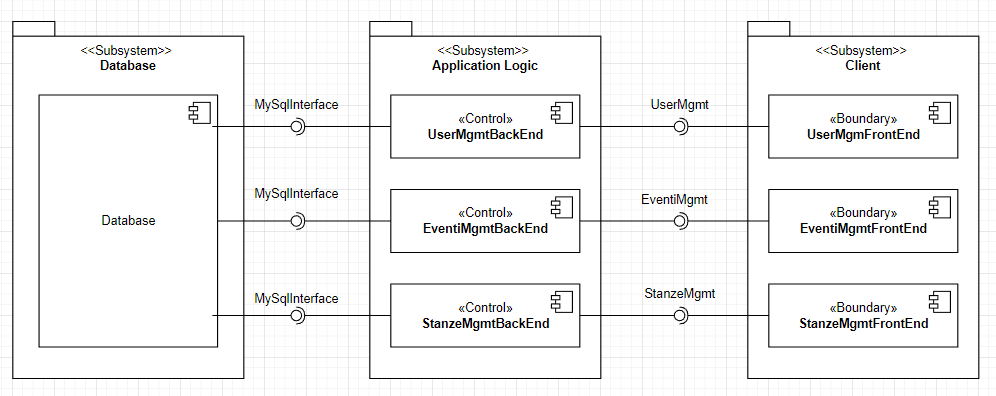
Oltre la stanza più adatta per l’orario inserito, verrà mostrata anche la stanza migliore presente nel database, con l’orario di disponibilità più vicino a quello dell’evento.

* 1. 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:

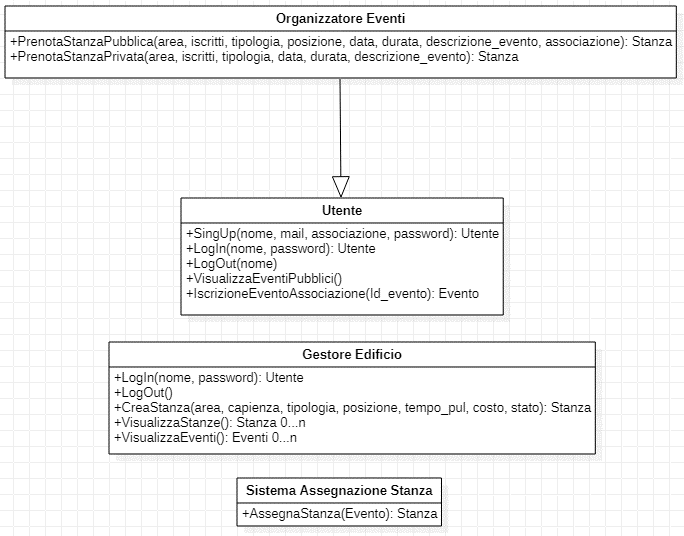
* <<Boundary>>: componenti lato front-end, ovvero ciò con cui l’utente si interfaccia;
* <<Control>>: componenti lato back-end, ovvero ciò che gestisce la logica applicativa, esponendo delle API al front-end e richiedendone a loro volta al database.



**Figura 2.1:** Component Diagram

* 1. UML Class Diagram per interfacce

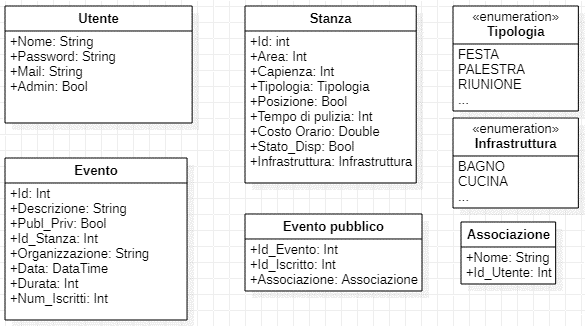
Tramite il seguente diagramma è possibile vedere le principali funzioni sviluppate durante questa iterazione, con i relativi input e output generati.



**Figura 2.2:** Class Diagram per interfacce

* 1. UML Class Diagram per tipo di dato

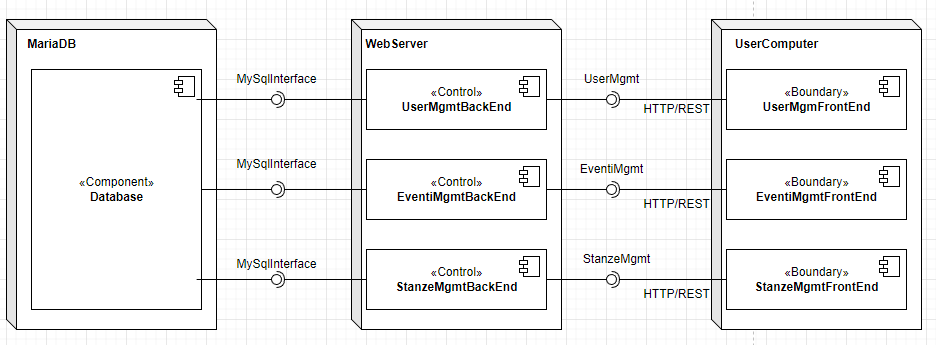
In questo diagramma vengono rappresentati i tipi di dato presenti in ogni classe del software, sui quali si basa tutto il database.



**Figura 2.3:** Class Diagram per tipo di dato

* 1. Deployement Diagram

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



**Figura 2.4:** Deployment Diagram

* 1. Testing

## 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.

## Analisi dinamica

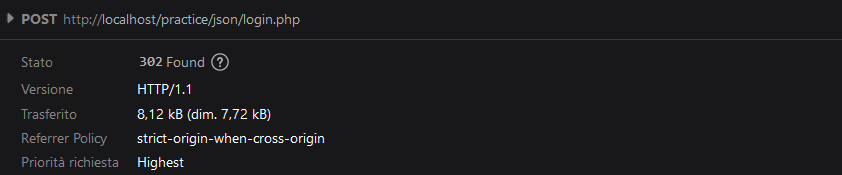
Durante questa iterazione è stato effettuato il test delle seguenti funzioni:

* 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;

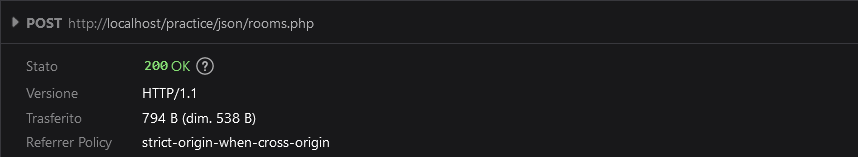
I risultati sono riportati di seguito.

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:*

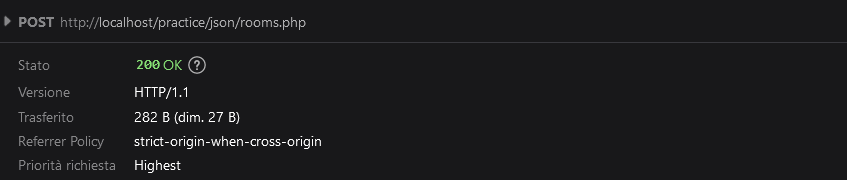


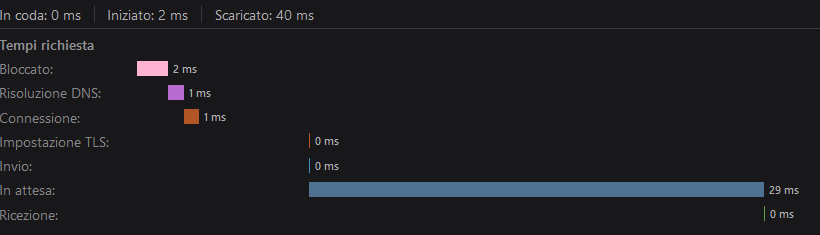
*Visualizzazione stanze:*



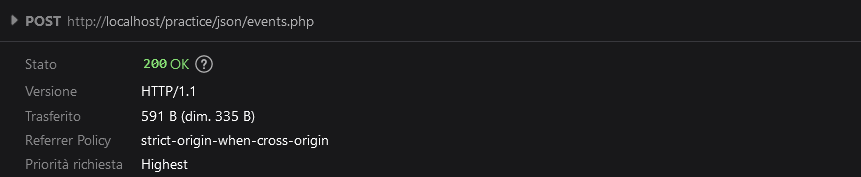


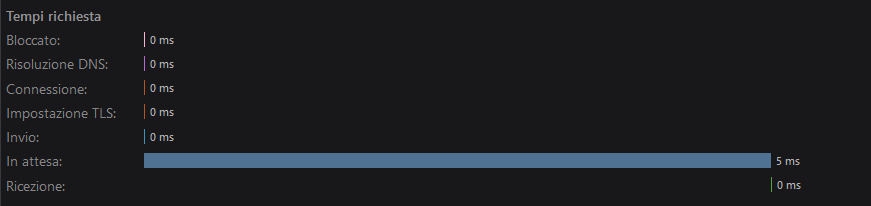
*Creazione stanze:*



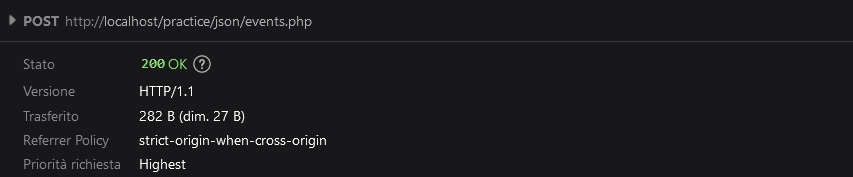


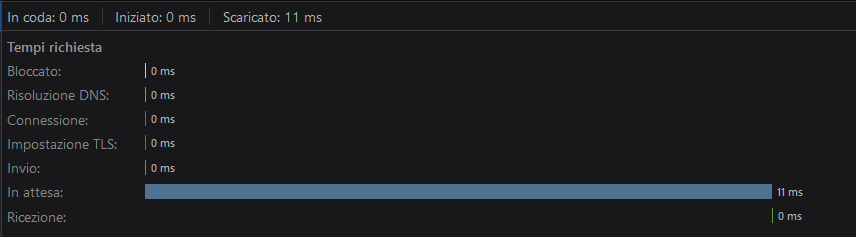
*Creazione evento:*





*Iscrizione ad un evento:*





* 1. Documentazione API

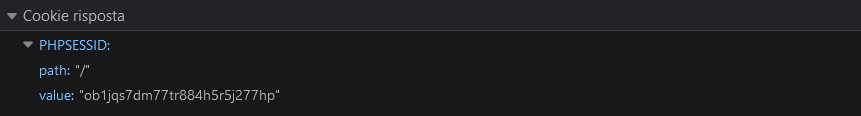
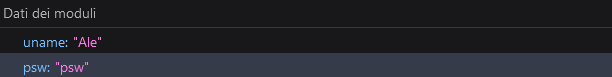
In questo sotto capitolo sono mostrate alcune delle API sviluppate durante Iterazione 1.

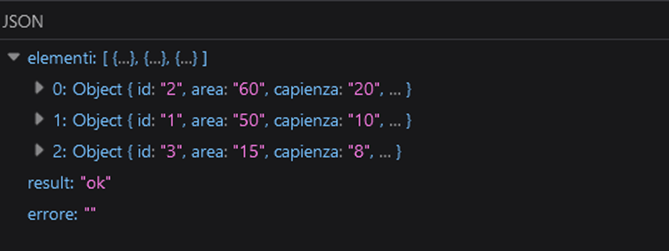
In particolare, vengono riportate:

* API per il Login dell’utente (e gestore);
* API per la visualizzazione dele 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 un la riposta 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.

*Login:*

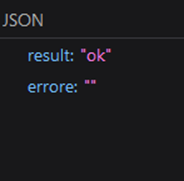


Immagine che contiene testo

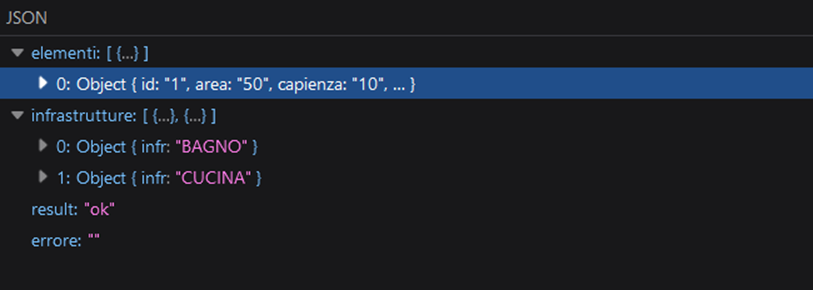
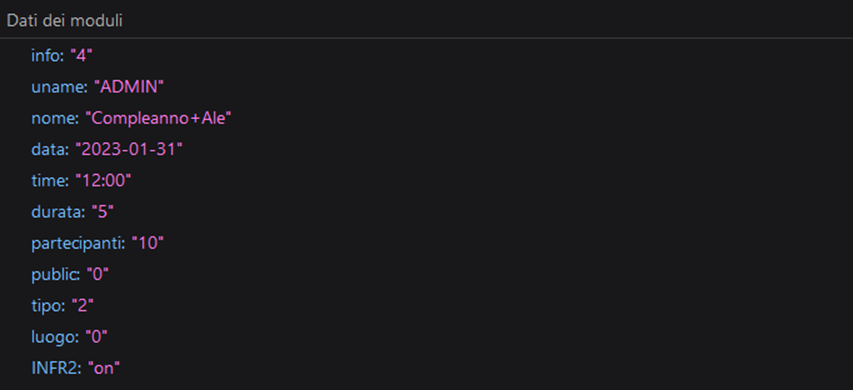
Descrizione generata automaticamente*Visualizzazione stanze:*

*Creazione stanze:*

**

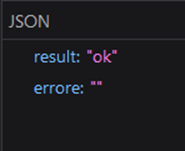
**

*Creazione evento:*



*Iscrizione ad un evento:*

**



1. ITERAZIONE 2
   1. Introduzione

Durante questa iterazione si è voluta implementare la possibilità di 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
  1. 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.

* 1. 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.

* 1. 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.

* 1. 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.

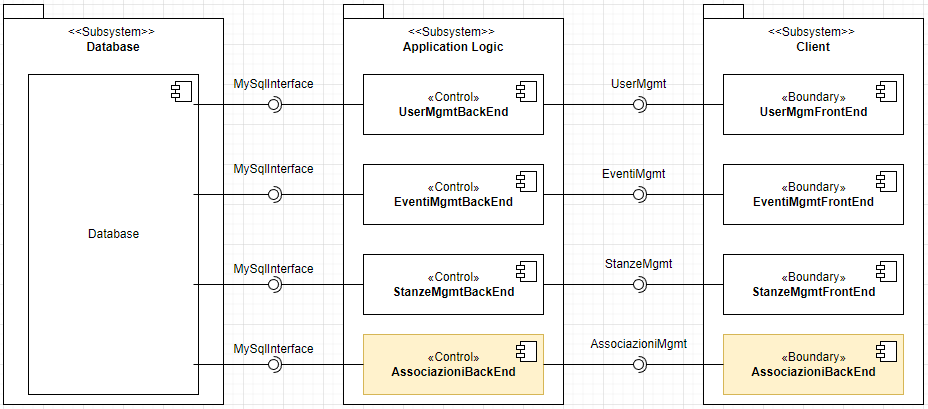
Dalla pagina “profilo”, l’utente potrà quindi avere una lista di tutte le associazioni a cui è iscritto.

* 1. 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, visualizzazione delle associazioni.

Nel diagramma sono messe in evidenza le nuove componenti.

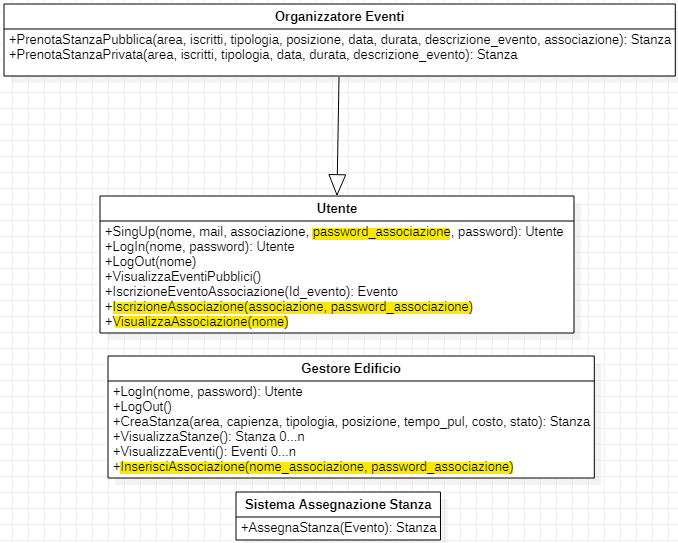


**Figura 3.1:** Component Diagram

* 1. UML Class Diagram per interfacce

È stato necessario aggiungere nuove operazioni al class diagram.

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



**Figura 3.2:** Class Diagram per interfacce

* 1. UML Class Diagram per tipo di dato

Le modifiche in questo diagramma, rispetto all’iterazione uno, sono minime.

Oltre la password, non sono stati inseriti nuovi dati, in quanto in questa iterazione vengono solo modificati i dati già presenti.

Di seguito viene rappresentato quindi il class diagram per tipo di dato, mettendo in evidenza l’unica aggiunta.

Immagine che contiene tavolo

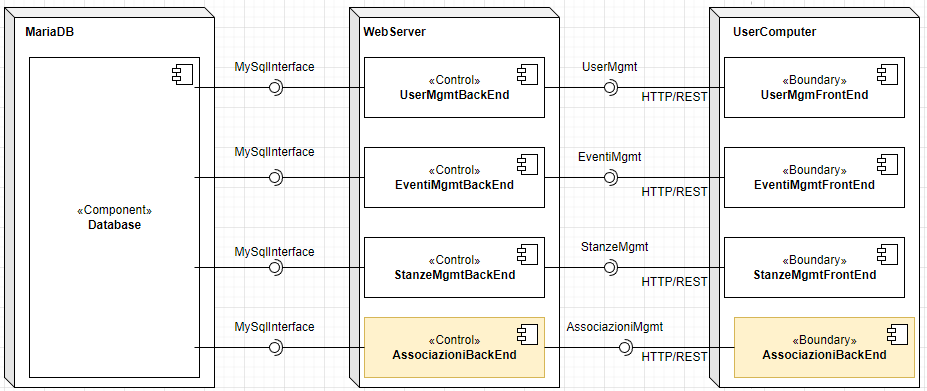
Descrizione generata automaticamente

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

* 1. Deployement 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

* 1. Testing

blablabla

1. ITERAZIONE 3
   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.

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;
  1. 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.

* 1. 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 implementate nella pagina “profilo”, ovvero la stessa pagina dove può visualizzare le informazioni utente (nome utente…).

* 1. 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.

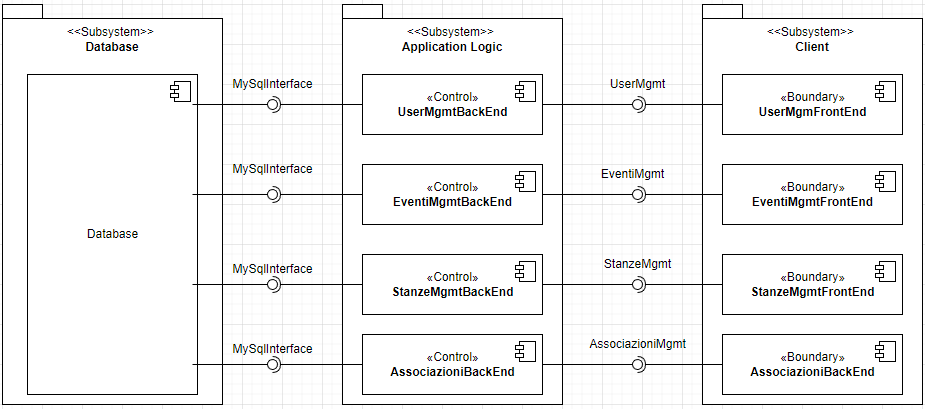
L’utente potrà effettuare la disiscrizione ad un evento a cui è iscritto dalla stessa pagina di visualizzazione di questi eventi.

* 1. 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, tutte queste nuove funzionalità riguardano gli eventi, rappresentati tramite il componente *EventMgm*.

Il diagramma è quindi il seguente.



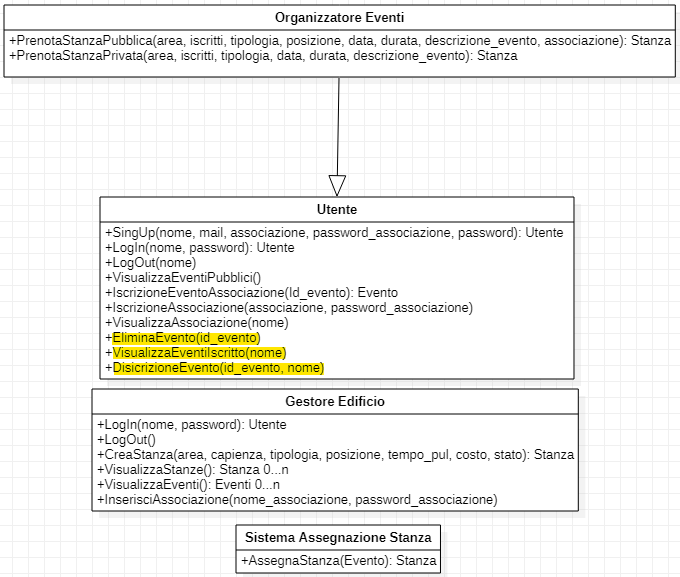
**Figura 4.1:** Component Diagram

* 1. 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.

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

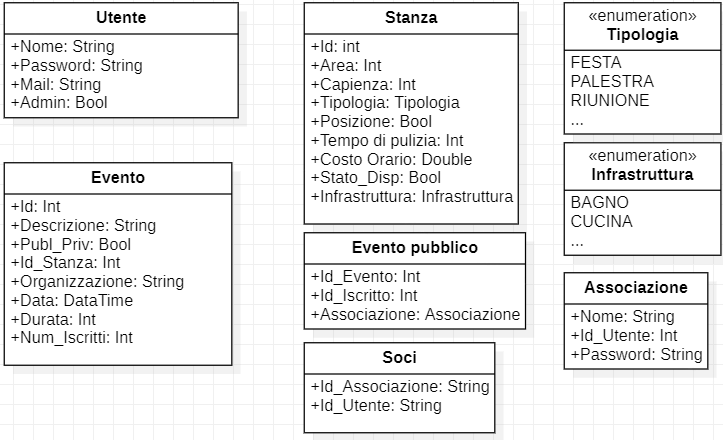


**Figura 4.2:** Class Diagram per interfacce

* 1. 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.



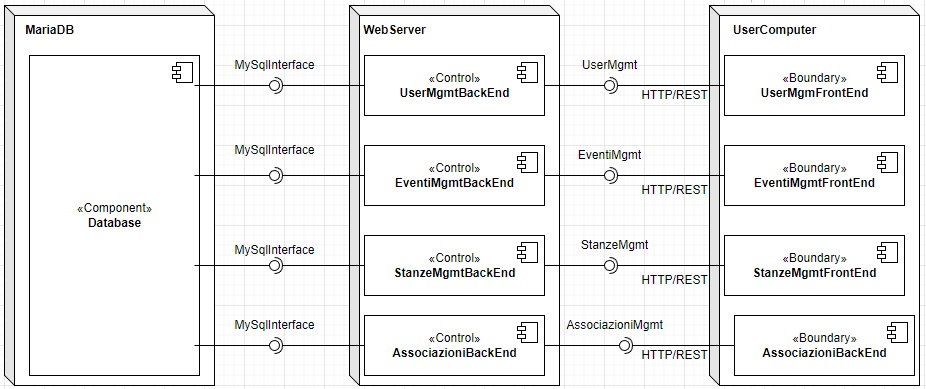
**Figura 4.3:** Class Diagram per tipo di dato

* 1. Deployement 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

* 1. Testing

blablabla