**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, 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.

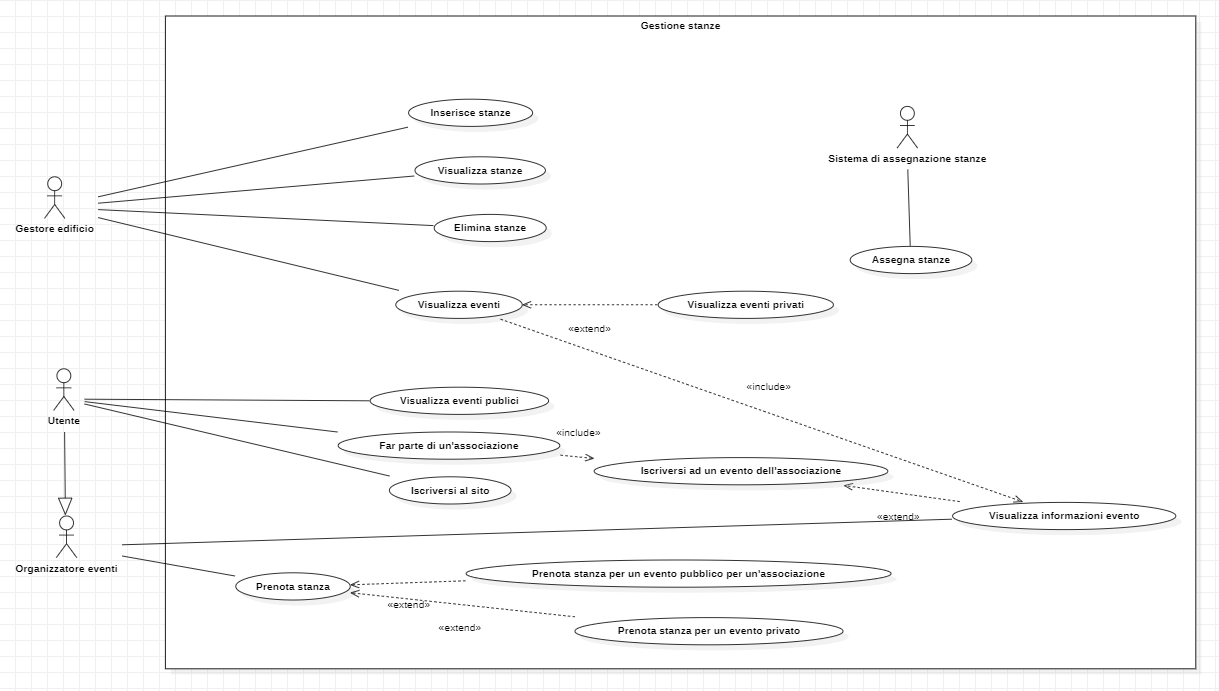
Un esempio di evento pubblico può essere un corso di un’attività sportiva, mentre un evento privato può essere una festa di compleanno con un definito numero di invitati.

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.

**1.2 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.



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 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

**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…
* 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…).

**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 dalla propria associazione.

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

**1.3.3 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.

**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. 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.

**1.4 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 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.



**1.6 Design pattern**

Useremo un’architettura Multi-Layer, 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 |
| 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. |
| Star UML | Software per la realizzazione grafici e diagrammi UML. |

**2 ITERAZIONE 1**