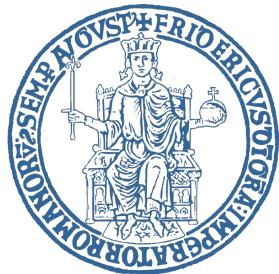


Università degli Studi di Napoli Federico II



Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Scuola Politecnica e delle Scienze di Base

Elaborato finale per l'esame di Software Architecture Design

Scendo: Un'app per organizzare uscite di
gruppo

Autori:

Daniele Marfella M63001365
Raffaele del Gaudio M63001389
Simone D'Orta M63001283

Anno Accademico
2021/2022

Indice

Elenco delle figure	v
1 Introduzione al Documento	1
1.1 Ruolo del Committente	1
1.2 Metodologia di lavoro	1
1.2.1 Strumenti a supporto dello sviluppo	2
2 Specifica dei Requisiti	5
2.1 Cos'è Scendo	5
2.1.1 Requisiti Informali	5
2.2 Requisiti Funzionali	7
2.3 Requisiti non funzionali	9
3 Analisi del Requisiti	10
3.1 Modello di Dominio	10
3.2 Diagramma di Contesto	11
3.3 Architettura preliminare	12
3.4 Diagramma dei Casi d'Uso	13
3.4.1 Activity Diagrams	14
3.4.1.1 Accetta invito	14
3.4.1.2 Crea nuova uscita	15
3.4.1.3 Invita utente	16
3.4.1.4 Promuovi partecipante	17
3.4.1.5 Registrazione	18
3.4.1.6 Consulta calendario uscite	19
3.4.1.7 Invia email registrazione	20
3.4.2 Scenari	21
3.4.2.1 Scenario di Registrazione	21
3.4.2.2 Scenario di Crea Uscita	21
3.4.2.3 Scenario di Promuovi Partecipante	22
3.4.2.4 Scenario di Consulta calendario uscite	22

3.4.2.5	Scenario di Accetta invito	22
3.4.2.6	Scenario di Invita Utente	23
3.4.2.7	Scenario di Invia email di registrazione	23
3.5	Stima dei costi	24
3.5.1	Unadjusted Use Case Weight (UUCW)	24
3.5.2	Unadjusted Actor Weight(UAW)	24
3.5.3	Fattori di complessità tecnica	25
3.5.4	Fattori di complessità dell'ambiente	26
3.5.5	Calcolo finale degli Use Case Points e stima	26
4	Documenti di Sviluppo	28
4.1	Architettura di dettaglio	28
4.2	Spring back-end	29
4.2.1	Spring Boot	29
4.2.2	Package Diagram	30
4.2.3	Class Diagram	30
4.2.4	Controller layer	32
4.2.5	Service layer	32
4.2.6	Repository layer	33
4.2.7	Ulteriori package	34
4.3	Interfacce REST	35
4.3.1	Accetta invito	36
4.3.2	Calendario uscite	36
4.3.3	Crea invito	37
4.3.4	Crea uscita	38
4.3.5	Info uscita con partecipanti	39
4.3.6	Info uscita senza partecipanti	40
4.3.7	Leggi Inviti	41
4.3.8	Login	42
4.3.9	Promuovi un partecipante	43
4.3.10	Registra utente	44
4.3.11	Rifiuta invito	45
4.3.12	Verifica registrazione	46
4.3.13	Bad Request	46
4.3.14	Login	47
4.4	Database	48
4.5	React Front-end	49
4.5.1	React component diagram	50
4.6	Sequence di dettaglio	51
4.6.1	Accetta invito	51
4.6.2	Consulta calendario uscite	52

4.6.3	Crea nuova uscita	53
4.6.4	Invia email registrazione	53
4.6.5	Invita utente	54
4.6.6	Promuovi partecipante	55
4.6.7	Registrazione	56
4.7	Deployment Diagram	57
4.7.1	Avviamento del sistema	57
5	Testing	59
5.1	Test d'Unità	59
5.2	Test End To End	60
5.3	Test di User Acceptance	60
6	Sviluppi futuri	62

Elenco delle figure

1.1	Funzionamento Github	3
1.2	Interfaccia Insomnia	4
1.3	Interfaccia Overleaf	4
3.1	Modello di dominio	10
3.2	Diagramma di contesto	11
3.3	Architettura preliminare	12
3.4	Diagramma dei casi d'uso	13
3.5	Activity Diagram Accetta invito	14
3.6	Activity Diagram Crea nuova uscita	15
3.7	Activity Diagram Invita utente	16
3.8	Activity Diagram Promuovi partecipante	17
3.9	Activity Diagram Registrazione	18
3.10	Activity Diagram Consulta calendario uscite	19
3.11	Activity Diagram Invia email registrazione	20
4.1	Logo di Spring	29
4.2	Package Diagram	30
4.3	Class Diagram	31
4.4	Controller layer	32
4.5	Service layer	32
4.6	Repository layer	33
4.7	Sequence diagram JWT	34
4.8	API accetta invito	36
4.9	API calendario uscite	36
4.10	API crea invito	37
4.11	API crea uscita	38
4.12	API info uscita con partecipanti	39
4.13	API info uscita senza partecipanti	40
4.14	API leggi inviti	41
4.15	API login	42

4.16 API promuovi un partecipante	43
4.17 API registra utente	44
4.18 API rifiuta invito	45
4.19 API verifica registrazione	46
4.20 Formato risposta bad request	46
4.21 Interfaccia login	47
4.22 Data Model	48
4.23 Logo di react	49
4.24 Front end component diagram	50
4.25 Sequence diagram Accetta invito	51
4.26 Sequence diagram Consulta calendario uscite	52
4.27 Sequence diagram Crea nuova uscita	53
4.28 Sequence diagram Invia email registrazione	53
4.29 Sequence diagram Invita utente	54
4.30 Sequence diagram Promuovi partecipante	55
4.31 Sequence diagram Registrazione	56
4.32 Deployment diagram	58
5.1 User Acceptance Test book	61

Capitolo 1

Introduzione al Documento

Nell'ambito dell'esame di Software Architecture Design è stata richiesto di costruire un sistema software e di presentarlo in sede d'esame con una adeguata documentazione.

Questo documento, insieme agli allegati che verranno segnalati durante la lettura dello stesso, rappresentano il documento Tecnico/Funzionale del progetto.

1.1 Ruolo del Committente

Per simulare quanto più realisticamente uno scenario reale di sviluppo, un membro scelto casualmente nel team ha ricoperto il ruolo del committente del progetto.

Questo ha permesso di simulare, durante le settimane dello svolgimento del progetto, dei meeting "col cliente" dove il membro prescelto vestiva i panni del committente che rilasciava feedback come prescritto usualmente dalle metodologie agili.

1.2 Metodologia di lavoro

Per la finalità del software prodotto, il tipo di architettura utilizzata e le esigenze di progetto, è stata scelta una metodologia di lavoro agile basata su UP.

Si identifica come data di partenza del progetto il 6 Giugno e data di conclusione dei lavori il 13 Luglio. Vista la natura dell'applicazione, ovvero una webapp composta da un backend e da un frontend, e tenuto conto della stima dei costi, la programmazione del lavoro è descritta in tab. 1.1.

Fase	Settimane	Obiettivi	Documenti prodotti/rifiniti
Ideazione	Dal 6/06 al 12/06	Ideazione e studio di fattibilità del progetto	<ul style="list-style-type: none"> - System Domain Model - Use Case Diagram - Data Model - Context Diagram - Bozza dell'architettura - Activity diagram di analisi
Iterazione 1	Dal 13/06 al 26/06	Realizzazione, testing e deploy del backend	<ul style="list-style-type: none"> - Sequence di dettaglio - Architettura di dettaglio - Deployment diagram
Iterazione 2	Dal 27/06 al 10/07	Realizzazione, testing e deploy del frontend	<ul style="list-style-type: none"> - Architettura di dettaglio - Deployment diagram

Tabella 1.1: Pianificazione del lavoro

In fase di ideazione sono stati svolti due workshop dei requisiti con il committente ad inizio e fine settimana dove l'obiettivo è stato definire lo scopo dell'applicazione e le funzionalità oltre che i requisiti di qualità.

Nella prima iterazione ci si è concentrati sulla costruzione del backend. Si è contestualmente sviluppato e prodotta documentazione di dettaglio. Il mercoledì della seconda settimana sono stati eseguiti gli Unit Test, il giovedì i test End-To-End in presenza del cliente e il Venerdì è stato prodotto l'eseguibile del backend.

Nella seconda iterazione si è costruito il frontend contestualmente al raffinamento della documentazione architettonica e di deployment. Il Venerdì della seconda settimana sono stati eseguiti i test di accettazione in presenza del cliente e il giorno seguente il frontend è stato deployato.

Si sottolinea che non sono stati specificati e realizzati tutti i casi d'uso scoperti in fase di ideazione ma si è seguito l'ordine dettato dalle metodologie incrementali, ovvero sviluppare per primi i casi d'uso che danno uno scheletro completo a tutta l'architettura del sistema.

1.2.1 Strumenti a supporto dello sviluppo

Lavorare contemporaneamente ad un progetto è fondamentale per qualsiasi team di sviluppo ed è proprio per questo motivo che abbiamo scelto

di utilizzare un tool che supporta a pieno questa funzionalità. Github è un servizio di hosting di repository git, il quale permette a più utenti di accedere a quest'ultima, oltre che tenere traccia di tutte le versioni e le relative modifiche. Le operazioni fondamentali più utilizzate sono state il commit, il pull ed il push. Queste sono fondamentali per poter gestire il lavoro concorrente tra più membri del team.

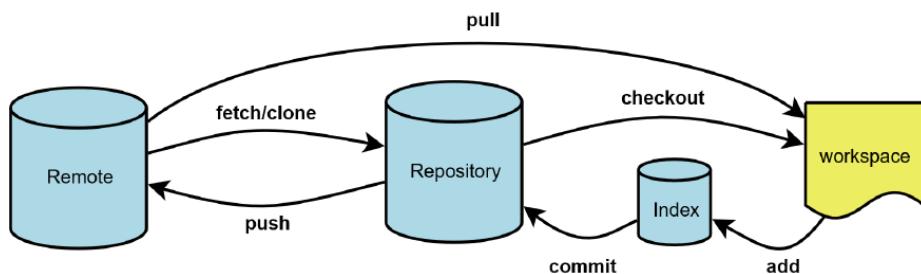


Figura 1.1: Funzionamento Github

Il commit registra i cambiamenti effettuati e li salva all'interno della repository locale, il pull recupera il file dalla repository remota e li unisce a quelli della repository locale, mentre il push invia i file modificati alla repository remota. È possibile che si verifichino conflitti nel caso in cui due membri del team modifichino contemporaneamente il progetto, ma nel caso in cui non siano stati modificati due file uguali questi sono solitamente risolti tramite un merge automatico, altrimenti il conflitto dovrà essere risolto manualmente.

Un altro software di vitale importanza è sicuramente Insomnia, attraverso il quale è possibile organizzare, salvare ed eseguire richieste basate sul protocollo HTTP ed analizzarne la risposta. È possibile utilizzare questo strumento per il testing e la documentazione dell'interfaccia delle nostre API.

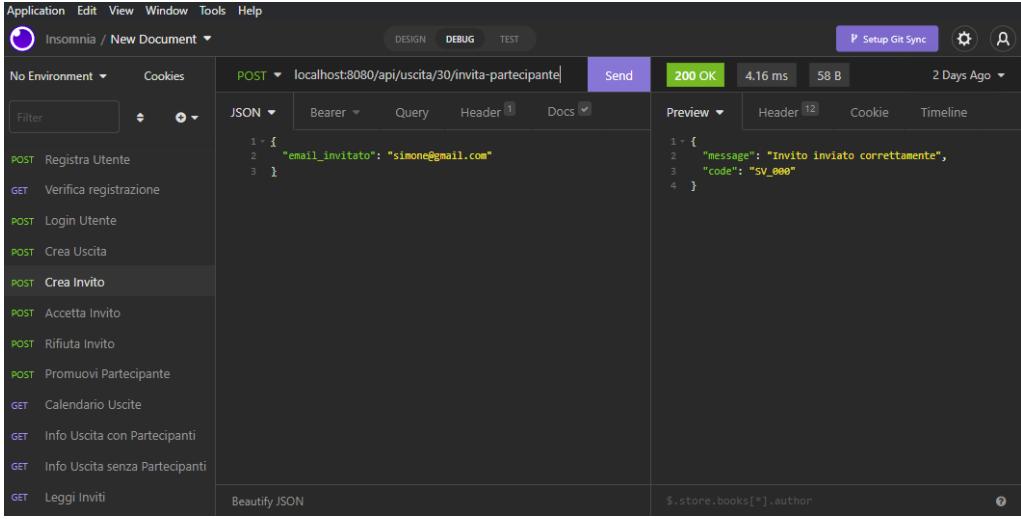


Figura 1.2: Interfaccia Insomnia

L'ultimo tool che è risultato molto utile nella scrittura del documento è Overleaf. Si tratta di un editor Latex cloud-based che permette la collaborazione in tempo reale oltre che offrire un sistema di versioning.

Figura 1.3: Interfaccia Overleaf

Capitolo 2

Specifiche dei Requisiti

2.1 Cos'è Scendo

Le applicazioni social e di chatting attualmente in voga non sono specializzate per organizzare uscite tra amici e dunque spesso risulta difficile reperire le informazioni dell'uscita nell'enorme quantità di dati presenti all'interno dell'applicazione. Scendo è un social network ideato per agevolare l'organizzazione di uscite tra amici, supportando una gestione chiara delle informazioni.

2.1.1 Requisiti Informali

Si vuole realizzare un'applicazione web per l'organizzazione e la gestione di uscite di gruppo tra utenti.

Gli utenti possono creare una o più uscite diventandone i creatori, oppure, partecipare ad uscite già create in qualità di partecipanti. Un'uscita viene creata da un utente il quale ne diventa il creatore ed è caratterizzata da una categoria (discoteca, pranzo, cena, cinema, etc...), una descrizione, un orario, un luogo d'incontro, un eventuale luogo di partenza, il numero massimo dei partecipanti e dalla lista di partecipanti e degli organizzatori. Si distinguono uscite pubbliche e private. Le uscite pubbliche sono visibili da tutti gli utenti della piattaforma e costoro ne possono richiedere la partecipazione mentre quelle private sono visibili solo se invitati a parteciparvi dal creatore o dagli organizzatori. Il creatore è unico per ogni uscita ed è l'unico a poterla cancellare. Egli definisce inoltre tutte le informazioni iniziali e necessarie alla sua creazione.

Il creatore può eleggere uno o più partecipanti al ruolo di organizzatore. Gli organizzatori ed il creatore sono partecipanti che possono modificare le caratteristiche dell'uscita, e con il diritto di invitare utenti inserendo l'email

ed accettare richieste di partecipazione. Organizzatori e creatori possono creare un link di invito all'uscita oltre che invitare utenti nella propria lista amici. L'utente partecipante non può modificare le caratteristiche dell'uscita ma può proporre eventuali modifiche al creatore e agli organizzatori che decideranno di rispondere e accettandole o rifiutarle.

Un utente può inviare una richiesta d'amicizia ad un altro utente, il quale può accettarla o rifiutarla.

Ogni utente può consultare tutte le uscite a cui partecipa, consultare una lista amici composta da altri utenti e cercare le uscite pubbliche in programma.

Gli utenti possono usufruire della propria lista amici per facilitare l'invito alle uscite e per informarsi sugli eventi degli amici. Infine prima di poter accedere ai servizi ogni utente deve registrarsi specificando nome, cognome, indirizzo email, password, età, sesso, data di nascita e le informazioni di residenza e loggarsi.

2.2 Requisiti Funzionali

Il comportamento di particolari operazioni effettuate dal sistema viene riconosciuto attraverso i requisiti funzionali, da cui, è possibile definire il documento dei casi d'uso. Gli attori rappresentano entità che svolgono dei specifici ruoli all'interno dell'applicativo, mentre, ogni caso d'uso è un' interazione fra un attore ed il sistema, al fine di svolgere un'unità di lavoro utile. Gli attori identificati sono:

- Utente non registrato: utente che non ha accesso alle funzionalità del sistema fin quando non esegue la registrazione.
- Utente: un utente che si è loggato al sistema.
- Amministratore: responsabile dell'avvio e dell'arresto fisico del sistema.
- Partecipante: un utente che nel contesto di una determinata uscita ne è un semplice partecipante.
- Organizzatore: un utente che ha i diritti necessari per apportare modifiche all'uscita, inviare inviti e accettare richieste di partecipazione.
- Creatore: è l'utente che ha creato quella particolare uscita, la gestisce completamente ed è l'unico a poterla cancellare.
- Scendo application: particolari parti interne al sistema che in modo automatico delegano l'invio della email di registrazione al servizio esterno di posta elettronica.
- Servizio di posta elettronica: attore secondario che rappresenta il servizio esterno di posta elettronica per l'invio delle email.

Nel processo di sviluppo adottato, sono stati scelti i seguenti requisiti funzionali da implementare:

Id	Descrizione
RF/01	Registrazione
RF/02	Creazione di una uscita
RF/03	Fornire i diritti da organizzatore ad un partecipante della uscita
RF/04	Consultare il calendario delle uscite in programma
RF/05	Accettare invito di partecipazione ad un'uscita
RF/06	Invitare un utente ad una uscita
RF/07	Invio automatico della mail di conferma registrazione

Tutti i requisiti funzionali identificati dalle specifiche informali invece, sono stati modellati da uno o più casi d'uso riportati nella seguente tabella.

Caso d'uso	Breve descrizione	Attori primari	Attori secondari	Casi d'uso inclusi	Punti d'estensione
Avvia il sistema	L'applicazione viene avviata	Amministratore			
Arresta il sistema	L'applicazione viene arrestata e smette di funzionare	Amministratore			
Riavvia il sistema	L'applicazione viene riavviata	Amministratore			
Registrazione	L'utente che non ha ancora effettuato la registrazione al sistema, interagisce con la pagina apposita inserendo i dati necessari alla registrazione e clicca il pulsante apposito per effettuare l'operazione	Utente non registrato			
Invia email di registrazione	Il sistema a seguito dell'operazione di registrazione provvede all'invio automatico di una mail per confermare l'avvenuta operazione	Scendo application	Servizio di posta elettronica	Registrazione	
Verifica registrazione	L'utente dopo aver cliccato sul link fornito dal servizio esterno di posta elettronica, viene indirizzato alla pagina ed il sistema conferma la registrazione	Utente non registrato			
Richiede di partecipare ad un'uscita pubblica	L'utente dopo aver selezionato un'uscita pubblica ne invia la richiesta di partecipazione	Utente			
Consulta calendario uscite	Il sistema mostra tutte le uscite di cui l'utente ne è coinvolto	Utente			
Cerca le uscite pubbliche	Cerca tutte le uscite pubbliche in una determinata zona specificata dall'utente	Utente			
Invia una richiesta d'amicizia	L'utente invia una richiesta d'amicizia al destinatario scrivendone la mail nello spazio apposito	Utente			
Consulta lista amici	Viene visualizzata la lista di tutti gli amici nell'apposita pagina richiesta dall'utente	Utente			
Risponde a richiesta di amicizia	Accetta o rifiuta una richiesta di amicizia	Utente			
Crea nuova uscita	Dopo aver inserito i dati necessari, l'utente conferma la creazione dell'uscita cliccando il pulsante apposito	Utente			
Accetta invito	L'utente decide di accettare l'invito cliccando il pulsante della risposta scelta	Utente			
Rifiuta invito	L'utente decide di rifiutare l'invito cliccando il pulsante della risposta scelta	Utente			
Proponi modifiche ad un'uscita	Il partecipante dell'uscita suggerisce agli organizzatori e al creatore eventuali modifiche attraverso un relativo form	Partecipante			
Abbandona un'uscita	L'utente sceglie di abbandonare l'uscita cliccando l'apposito pulsante	Partecipante / Organizzatore		Cancella un'uscita	
Accetta richieste di partecipazione	Gli organizzatori o il creatore accettano la richiesta di partecipazione presentata da un utente	Organizzatore / Creatore			
Modifica caratteristiche uscita	Modifica alcuni parametri di un'uscita	Organizzatore / Creatore			
Invita utente	Il sistema invia un invito dell'uscita all'utente destinatario dopo che l'utente invitante ha scritto la mail e cliccato "invita"	Organizzatore / Creatore			
Cancella un'uscita	Il creatore decide di cancellare una sua uscita eliminandola dal calendario	Creatore			
Promovi partecipante	Promozione di un partecipante al ruolo di organizzatore	Organizzatore / Creatore			

Tabella 2.1: Modellazione dei casi d'uso

2.3 Requisiti non funzionali

Seguendo le specifiche, sarà realizzato un sistema di gestione delle uscite che dovrà fornire una serie di funzionalità che l'utente può accedere attraverso diverse interfacce. Il sistema deve tener traccia di ogni uscita e dei relativi partecipanti, che dovranno essere mantenuti all'interno di una periferica di archiviazione. Fra le informazioni che il sistema deve gestire vi sono inoltre tutte quelle anagrafiche riguardo gli utenti che si registrano al sistema. Ogni funzionalità del sistema deve garantire all'utente finale una facilità d'utilizzo dell'applicativo rimanendo coerente con le specifiche proposte. A valle delle considerazioni ottenute possiamo definire alcuni dei requisiti di qualità richiesti all'architettura:

- Sicurezza: solo un utente registrato e loggato può utilizzare le funzionalità del sistema.
- Performance: ogni richiesta nei confronti del server deve essere evasa in meno di mezzo secondo.
- Portabilità: l'architettura deve permettere futuri sviluppi di differenti tipologie di client oltre quello browser.
- Modificabilità: deve essere garantita una ottima separazione delle responsabilità per permettere la massima agevolazione nei processi di application maintenance e change request. Aggiunte e rimozioni di attributi al modello dei dati (che non modificano la logica di controllo) devono essere integrabili nel software in 3-7 MD. z
- Robustezza: il software deve comportarsi in maniera accettabile anche in situazioni non specificate dai requisiti, ad esempio, se occorrono problemi nelle interfacce.
- Usabilità: l'applicativo dev'essere facile da utilizzare per tutti gli utenti finali.

Capitolo 3

Analisi del Requisiti

3.1 Modello di Dominio

Il team di sviluppo attraverso la metodologia di lavoro adottata ha individuato tre particolari entità su cui basare le prime iterazioni di elaborazione, in particolar modo: uscita, invito ed utente. Per definire anticipatamente i concetti principali di un ambito applicativo, nonché le sue associazioni, viene adoperato un particolare modello concettuale detto modello di dominio. Secondo lo schema logico accennato dal committente del progetto, ogni istanza dell'entità uscita interagisce con uno o più utenti rappresentando una classica associazione "molti a molti". Per quanto concerne invece l'entità invito, le sue associazioni con le altre istanze hanno una cardinalità "molti ad uno". In fig. 3.1 è riportato un modello di dominio che riassume quanto detto.

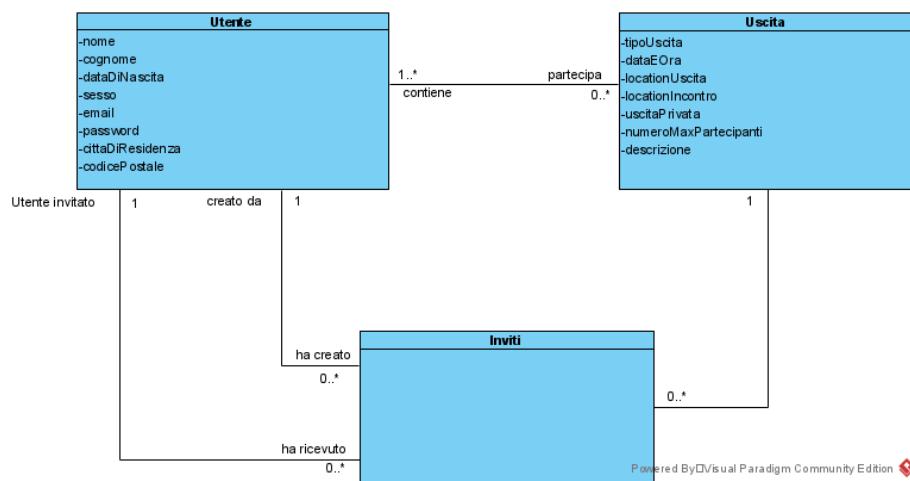


Figura 3.1: Modello di dominio

3.2 Diagramma di Contesto

Al fine di avere un quadro chiaro degli sviluppi implementativi che seguiranno è doveroso rappresentare sinteticamente le relazioni che ha il sistema con l'ambiente esterno, mediante l'ausilio di un diagramma di contesto.

E' importante specificare che ogni utente esterno al sistema comunica con esso attraverso delle interfacce grafiche, il cui sviluppo verrà spiegato nei successivi capitoli. L'unica funzionalità che non prevede un'interfaccia grafica è quella riguardante l'invio mail, in quanto, il sistema comunica con il servizio esterno in modo del tutto implicito.

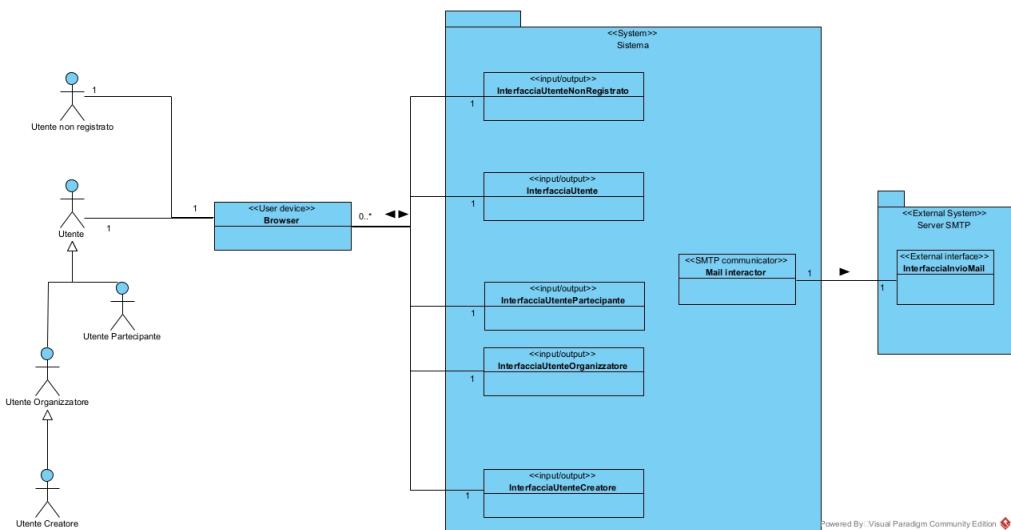


Figura 3.2: Diagramma di contesto

3.3 Architettura preliminare

Le applicazioni web prevedevano la generazione di pagine HTML lato server in base alla richiesta effettuata (eventualmente usando template per renderle dinamiche), le quali venivano inviate al browser client per la rende- rizzazione. Oggi, nelle moderne applicazioni web la logica di presentazione è completamente demandata ai client, i quali effettuano una chiamata iniziale per ottenere file HTML, CSS e JavaScript ed attraverso l'esecuzione di tale codice JavaScript comunicano con il server back end per ricevere dati, i quali sono manipolati per generare dinamicamente l'interfaccia a schermo da mostrare all'utente. Nella figura in seguito, è mostrata l'architettura preliminare del nostro sistema.

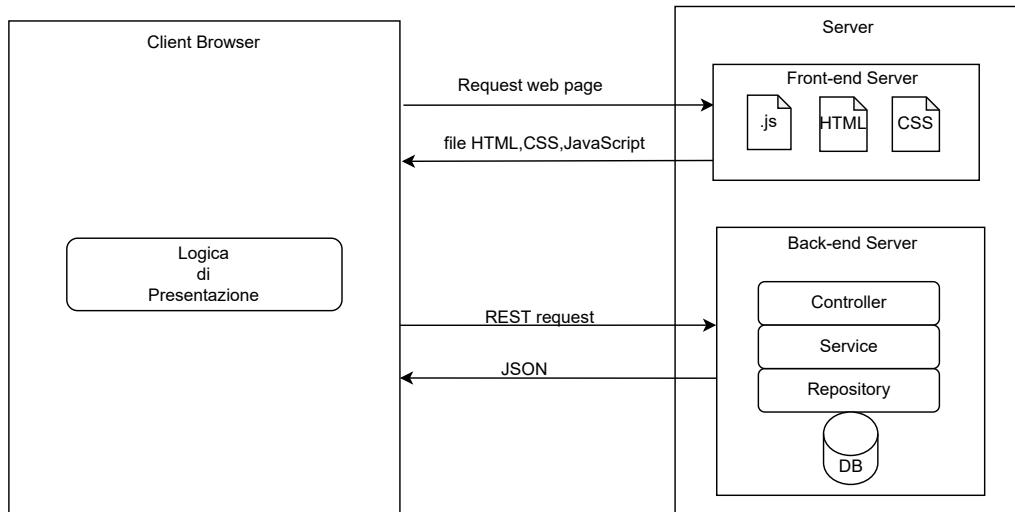


Figura 3.3: Architettura preliminare

3.4 Diagramma dei Casi d'Uso

Per concretizzare la modellazione dei casi d'uso è necessario distinguere i ruoli degli attori all'interno del sistema a seconda delle funzionalità interessate. E' buona norma quindi definire un diagramma dei casi d'uso che si occupa proprio di quanto detto, di seguito mostrato in fig. 3.3, dove in arancione sono evidenziati i casi d'uso implementati nell'ambito di questo progetto.

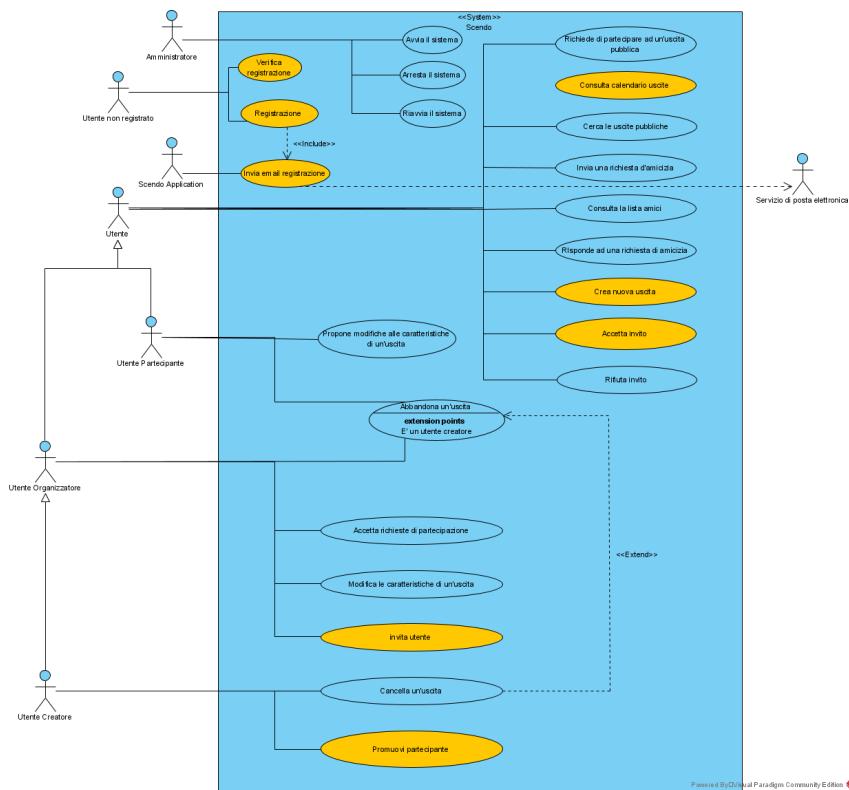


Figura 3.4: Diagramma dei casi d'uso

E' importante notare che le scelte implementative hanno portato all'inclusione del caso d'uso "Registrazione" in "Invia email di registrazione" in quanto quest'ultimo non può svolgersi senza che l'utente non registrato abbia compilato il form di registrazione.

3.4.1 Activity Diagrams

È stato implementato un activity diagram per ogni caso d'uso. Questi risultano utili in fase di analisi per descrivere graficamente l'ordine delle attività all'interno del caso d'uso.

3.4.1.1 Accetta invito

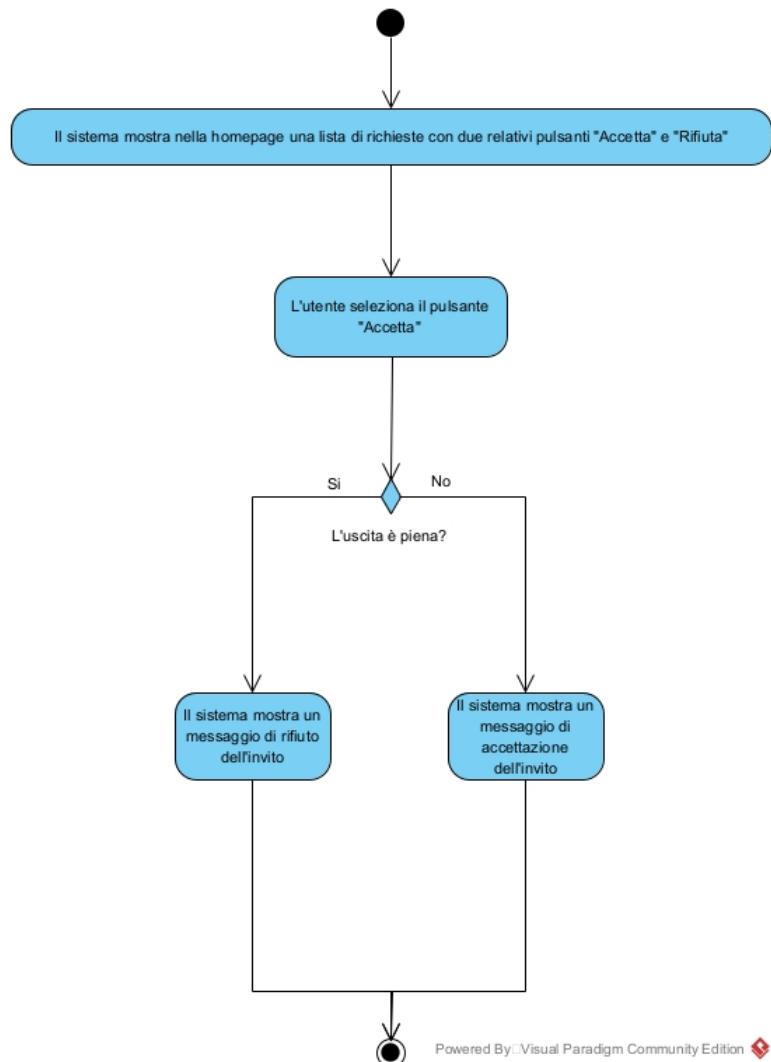


Figura 3.5: Activity Diagram Accetta invito

3.4.1.2 Crea nuova uscita

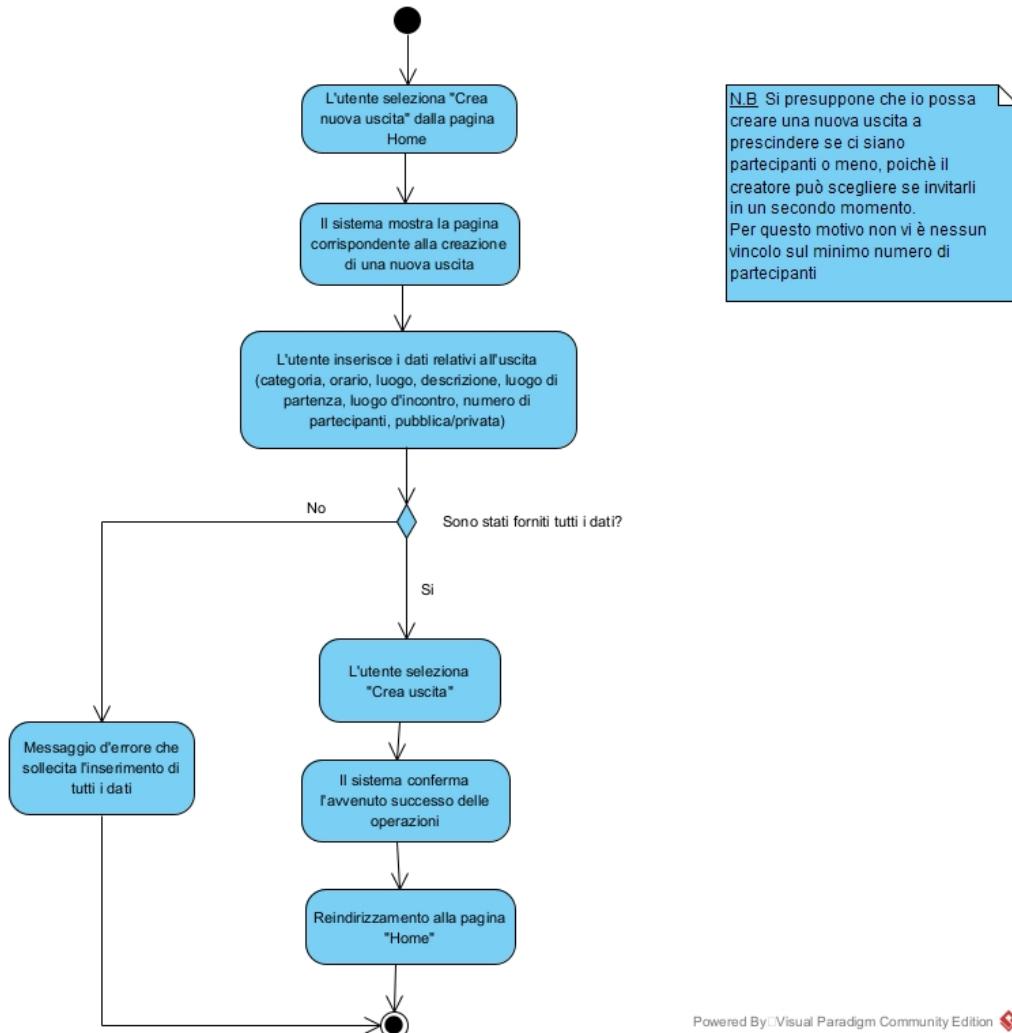


Figura 3.6: Activity Diagram Crea nuova uscita

3.4.1.3 Invita utente

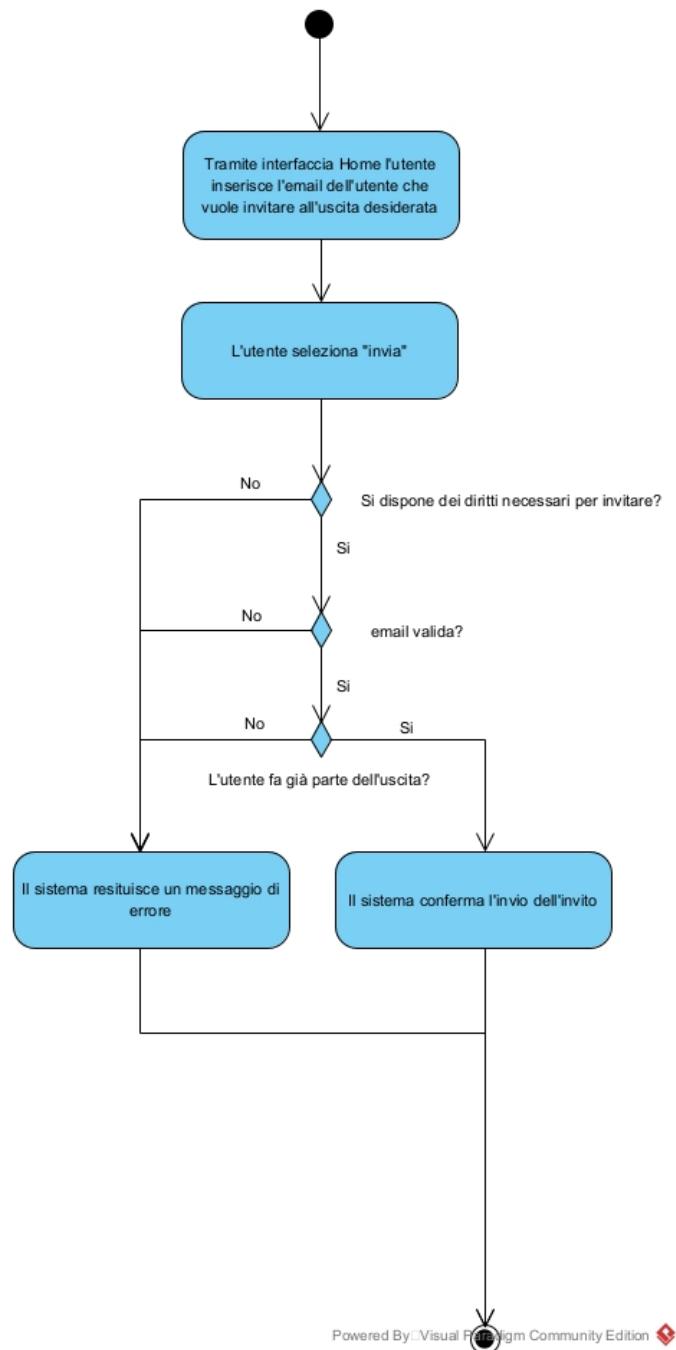


Figura 3.7: Activity Diagram Invita utente

3.4.1.4 Promuovi partecipante

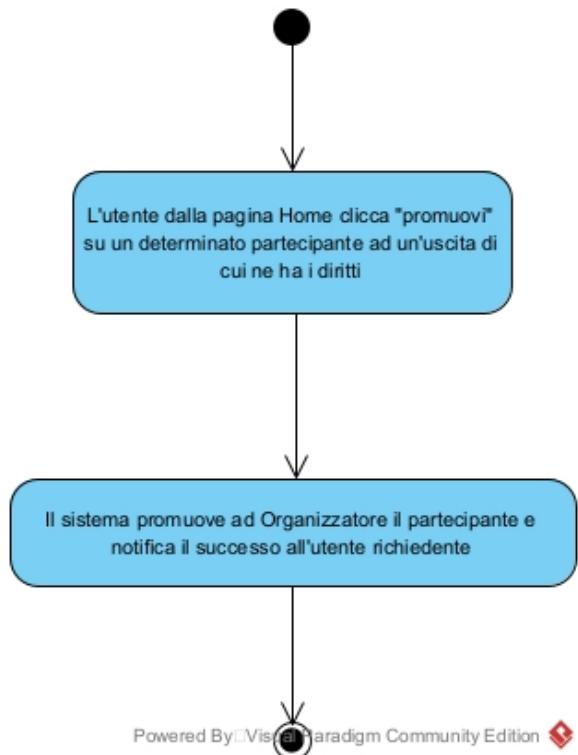


Figura 3.8: Activity Diagram Promuovi partecipante

3.4.1.5 Registrazione

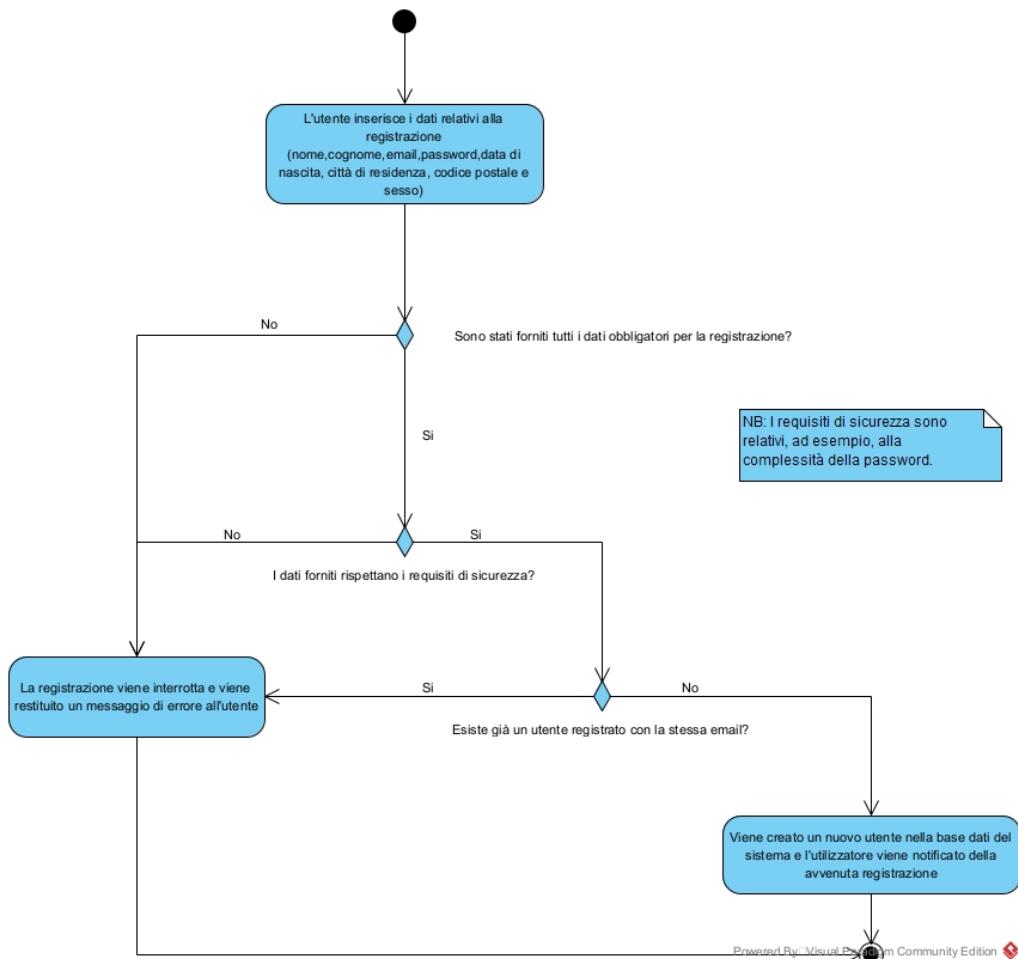


Figura 3.9: Activity Diagram Registrazione

3.4.1.6 Consulta calendario uscite

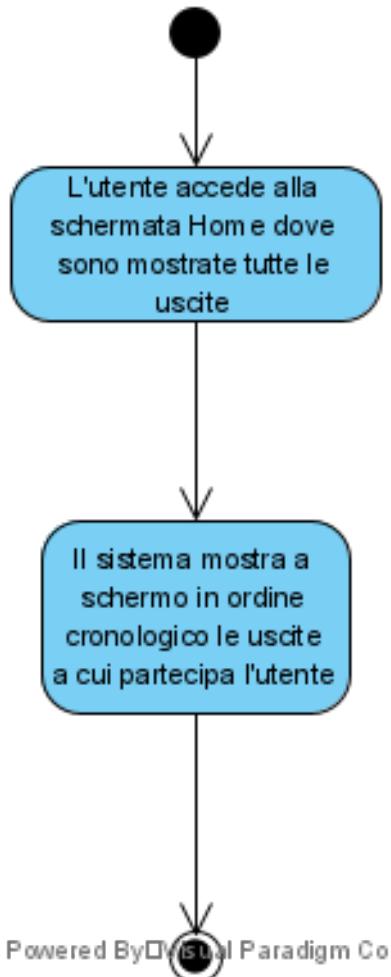


Figura 3.10: Activity Diagram Consulta calendario uscite

3.4.1.7 Invia email registrazione



Figura 3.11: Activity Diagram Invia email registrazione

3.4.2 Scenari

Per descrivere le sequenze di eventi che occorrono all'innesto di un caso d'uso vengono definiti gli scenari, di seguito quelli riportati per i casi d'uso scelti.

3.4.2.1 Scenario di Registrazione

Caso d'uso	Registrazione
ID	RF/01
Breve descrizione	Creazione di un nuovo utente nel sistema
Attori primari	Utente non registrato
Attori secondari	Nessuno
Pre-condizione	Nessuna
Sequenza eventi principali	<ol style="list-style-type: none"> 1. Il caso d'uso inizia quando l'utente dopo aver inserito i dati necessari clicca "Registrati" 2. Il sistema controlla se sono stati forniti tutti i dati necessari e se essi rispettano tutti i requisiti di sicurezza 3. Il sistema controlla se la mail è già stata usata per la registrazione da un altro utente 4. Il sistema restituisce un messaggio di utente creato e viene istanziata la nuova informazione all'interno del database
Post-condizione	Il database contiene le informazioni sul nuovo utente e altre componenti del sistema vengono informate dell'invio mail per confermare la registrazione all'utente.
Sequenza degli eventi alternativa	<ol style="list-style-type: none"> 2b. Se non sono stati forniti i dati necessari alla registrazione, o, se non vengono rispettati i requisiti di sicurezza, il sistema restituisce un messaggio di errore all'utente. 4b. Se la mail è già in uso da un altro utente all'interno del Database, allora viene restituito un messaggio di errore all'utente.

Tabella 3.1: Caso d'uso registrazione

3.4.2.2 Scenario di Crea Uscita

Caso d'uso	Crea nuova uscita
ID	RF/02
Breve descrizione	Creazione di una nuova uscita nel sistema
Attori primari	Utente
Attori secondari	Nessuno
Pre-condizione	L'utente è loggato al sistema e si trova nella pagina "Crea uscita"
Sequenza eventi principali	<ol style="list-style-type: none"> 1. L'utente inserisce i dati dell'uscita nell'apposito form e clicca "Crea" 2. Il sistema controlla se sono stati inseriti tutti i dati 3. Il sistema restituisce un messaggio di uscita creata con successo 4. Il sistema reindirizza l'utente alla pagina "Home"
Post-condizione	Il database contiene le informazioni sulla nuova uscita
Sequenza degli eventi alternativa	<ol style="list-style-type: none"> 3b. Se non sono stati forniti tutti i dati necessari viene visualizzato un messaggio d'errore

Tabella 3.2: Caso d'uso crea uscita

3.4.2.3 Scenario di Promuovi Partecipante

Caso d'uso	Promuovi partecipante
ID	RF/03
Breve descrizione	Fornire i diritti da organizzatore a un partecipante
Attori primari	Creatore
Attori secondari	Nessuno
Pre-condizione	L'utente target partecipa all'uscita
Sequenza eventi principali	<ol style="list-style-type: none"> 1. Il caso d'uso inizia quando dalla pagina "Home" il creatore dell'uscita clicca "Promuovi" su un partecipante 2. Il sistema aggiorna il database con il nuovo organizzatore dell'uscita 3. Viene notificato l'utente dell'avvenuto successo dell'operazione
Post-condizione	Il database contiene la nuova informazione inerente all'utente organizzatore e dalla pagina "Home" non è più possibile promuoverlo
Sequenza degli eventi alternativa	

Tabella 3.3: Caso d'uso promuovi partecipante

3.4.2.4 Scenario di Consulta calendario uscite

Caso d'uso	Consulta calendario uscita
ID	RF/04
Breve descrizione	Cercare le uscite in programma
Attori primari	Utente
Attori secondari	Nessuno
Pre-condizione	L'utente si è loggato al sistema
Sequenza eventi principali	<ol style="list-style-type: none"> 1. Il caso d'uso inizia quando viene l'utente accede alla schermata Home 2. Il sistema controlla se ci sono uscite inerenti all'utente 3. Il sistema mostra nella pagina la lista di uscite in programma
Post-condizione	Viene mostrato il risultato nella pagina "Home"
Sequenza degli eventi alternativa	

Tabella 3.4: Caso d'uso consulta calendario uscite

3.4.2.5 Scenario di Accetta invito

Caso d'uso	Accetta invito
ID	RF/05
Breve descrizione	Accettare l'invito di partecipazione ad un'uscita
Attori primari	Organizzatore/Creatore
Attori secondari	Nessuno
Pre-condizione	L'utente non partecipa all'uscita ed ha ricevuto l'invito di partecipazione visualizzabile dalla pagina "Home"
Sequenza eventi principali	<ol style="list-style-type: none"> 1. Il caso d'uso inizia quando dalla pagina "Home" il creatore dell'uscita clicca "Accetta" su un invito 2. Il sistema controlla se l'uscita ha raggiunto il numero massimo di partecipanti 3. Se l'uscita non ha raggiunto il numero massimo di partecipanti aggiunge l'utente ad essa e aggiorna il database 4. Il sistema restituisce un messaggio di avvenuto successo
Post-condizione	Il database contiene le informazioni sul nuovo utente e l'invito non è più visualizzabile
Sequenza degli eventi alternativa	<ol style="list-style-type: none"> 3b. Se l'uscita contiene già il massimo numero di partecipanti viene restituito un messaggio che avverte che l'uscita è piena

Tabella 3.5: Caso d'uso accetta invito

3.4.2.6 Scenario di Invita Utente

Caso d'uso	Invita utente
ID	RF/06
Breve descrizione	Invitare un utente ad un'uscita
Attori primari	Organizzatore/Creatore
Attori secondari	Nessuno
Pre-condizione	Si ha almeno un'uscita in programma di cui si dispone dei diritti necessari per l'invito.
Sequenza eventi principali	1. Il caso d'uso inizia quando dalla pagina "Home" il creatore dell'uscita scrive la mail dell'utente da invitare e clicca "Invita" 2. Il sistema controlla se l'utente esiste all'interno del sistema e può essere invitato 3. Il sistema aggiorna le informazioni al database col nuovo invito e notifica l'utente destinatario
Post-condizione	Il database contiene le informazioni sul nuovo invito e l'utente destinatario può ora visualizzare l'invito dalla sua "Home"
Sequenza degli eventi alternativa	3b. Se l'utente non esiste o se fa già parte dell'uscita, viene restituito un messaggio di errore invio

Tabella 3.6: Caso d'uso invita utente

3.4.2.7 Scenario di Invia email di registrazione

Caso d'uso	Invia email registrazione
ID	RF/07
Breve descrizione	Viene inviata un'email automatica all'utente registrato
Attori primari	Scendo application
Attori secondari	Nessuno
Pre-condizione	L'utente ha appena effettuato la registrazione al sistema
Sequenza eventi principali	1. Il sistema invia il messaggio da spedire all'utente tramite mail al servizio esterno di posta elettronica 2. Il servizio esterno di posta elettronica manda la mail all'indirizzo dell'utente.
Post-condizione	Il database contiene la nuova informazione inerente al token di registrazione che l'utente dovrà usare per confermare la registrazione.

Tabella 3.7: Caso d'uso invia email di registrazione

3.5 Stima dei costi

Per l'analisi dei costi si è scelto di utilizzare il metodo degli Use Case Points che permettono di stimare la dimensione di un'applicazione e la durata del progetto.

Come suggerisce il nome, i punti che andremo ad utilizzare dipendono dalla complessità dei casi d'uso analizzati, in particolar modo, si è scelto di fare la stima solo sui casi d'uso da implementare in modo da poter confrontare la durata effettiva con la durata ottenuta.

3.5.1 Unadjusted Use Case Weight (UUCW)

Calcolando il numero di transazioni di ognuno dei sette casi d'uso analizzati, è possibile stimarne la complessità ed il peso.

Caso d'uso	Complessità	Numero di transazioni	Peso
RF/01	Media	6	10
RF/02	Media	5	10
RF/03	Semplice	3	5
RF/04	Semplice	3	5
RF/05	Media	5	10
RF/06	Media	4	10
RF/07	Semplice	2	5

Tabella 3.8: Complessità dei casi d'uso

Moltiplicando i casi d'uso per il loro peso e sommandoli otteniamo:

$$UUCW = 4 * 10 + 5 * 3 = 55$$

3.5.2 Unadjusted Actor Weight(UAW)

Analogamente al passaggio precedente, vengono calcolati i pesi di ognuno degli attori facenti parte dei casi d'uso scelti.

Attore	Complessità	Peso
Utente non registrato	Complesso	3
Utente	Complesso	3
Organizzatore	Complesso	3
Creatore	Complesso	3
Scendo application	Semplice	1
Servizio di posta elettronica	Medio	2

Tabella 3.9: Complessità degli attori

Sommando il peso di ciascun attore otteniamo:

$$UAW = 4 * 3 + 2 + 1 = 15$$

Dunque gli Unadjusted Use Case Points sono:

$$UUCP = UUCW + UAW = 70$$

3.5.3 Fattori di complessità tecnica

I requisiti non funzionali complicano ulteriormente lo sviluppo di un sistema software. La stima dei costi, per affrontare anche questi vincoli, definisce tredici fattori di complessità tecnica a cui associare un peso ed una valutazione, riassunti nella successiva tabella.

Fattore	Peso	Valutazione	Impatto
Sistema distribuito	2	3	6
Prestazioni	2	3	6
Efficienza end user	3	2	6
Elaborazioni complesse	1	2	2
Riusabilità	2	3	6
Facilità d'installazione	2	2	4
Facilità d'uso	2	3	6
Portabilità	2	2	4
Facilità di cambiamento	1	2	2
Utilizzo concorrente	3	1	3
Sicurezza	3	3	9
Accesso da terze parti	2	1	2
Necessità di aggiornamenti	1	1	1

Tabella 3.10: Valutazione fattori di complessità tecnica

Sommando il valore d'impatto di ogni fattore si ottiene il Tfactor che nel nostro caso risulta essere:

$$Tfactor = 57$$

Calcolo ora il Technical Complexity Factor:

$$TCF = 0.6 + (0.01 * Tfactor) = 1.17$$

3.5.4 Fattori di complessità dell'ambiente

Similmente al paragrafo precedente, viene eseguito anche il calcolo del peso dei fattori di complessità ambientali.

Fattore	Peso	Valutazione	Impatto
Familiarità coi processi di sviluppo	1.5	2	3
Esperienza nelle applicazioni	1	2	2
Esperienza programmazione ad oggetti	2	2	4
Capacità da analista	2	2	4
Motivazione	1	4	4
Requisiti di stabilità	1	2	2
Part-time staff	0	0	0
Linguaggi di programmazione difficili	1	1	1
Facilità di cambiamento	1	2	2
Utilizzo concorrente	3	1	3
Sicurezza	3	3	9
Accesso da terze parti	2	1	2
Necessità di aggiornamenti	1	1	1

Tabella 3.11: Valutazione fattori di complessità ambientale

Da cui è possibile ricavare l'E-factor:

$$EF = 1.4 + (-0.03 \times 20) = 0.8$$

3.5.5 Calcolo finale degli Use Case Points e stima

Moltiplicando tutti i risultati ottenuti è possibile calcolare gli UCP:

$$UCP = UUCP * TCF * EF = 70 * 1.17 * 0.8 = 65,52$$

Essendo quasi del tutto novizi nello sviluppo software abbiamo considerato un fattore di 26 ore per use case point.

Ore di lavoro totali :

$$UCP * 26 = 1703,52$$

Poichè nel team lavorano tre persone, le ore settimanali totali risultano essere 78.

Per un'iterazione di due settimane occorrono quindi 156 ore di lavoro.

Dividendo le ore di lavoro totali per quelle necessarie ad un'iterazione otteniamo:

$$1703,52 / 156 = 10,92$$

Quindi si stima che occorrono approssimando per eccesso 11 iterazioni allo sviluppo del progetto, nonchè 22 settimane.

Capitolo 4

Documenti di Sviluppo

4.1 Architettura di dettaglio

Lo stile architettonico adottato è quello client-server. Come si è visto nell’architettura preliminare, viene adottato un front-end server per fornire le interfacce grafiche ed un back-end server per fornire degli endpoint per accedere alla logica di business.

Il browser, quando richiede una pagina web, viene risposto dal front-end ottenendo l’applicazione sviluppata in React, ovvero un insieme di file HTML, CSS e JavaScript. Questi file vengono elaborati grazie al motore interno al browser.

Si tratta quindi di uno stile architettonico di tipo code on demand utile per alleggerire il carico totale tra le due entità server. L’interazione tra l’utente e l’interfaccia web si traduce dunque in esecuzione di codice JavaScript, attraverso il quale verranno effettuate chiamate REST al server per ottenere, modificare o cancellare dati, che saranno poi mostrati all’utente secondo la logica di presentazione sviluppata nell’applicazione React.

4.2 Spring back-end



Figura 4.1: Logo di Spring

Spring è un framework Java utilizzato per lo sviluppo di applicazioni enterprise che presenta una struttura modulare il cui utilizzo è possibile sia nella sua interezza, che solo in parte. Le funzionalità fondamentali del framework si possono racchiudere in un modulo Core che implementa la gestione delle transazioni, la struttura di applicazioni Web, l'accesso ai dati, la messaggistica, i test e altro.

I due principi su cui è basato il framework sono l’Inversion of Control e la Dependency Injection.

Grazie all’Inversion of Control(IoC) la logica del flusso di controllo che comprende le operazioni di creazione, inizializzazione degli oggetti ed invocazione dei metodi è delegata al framework in modo che non sia più lo sviluppatore a doversi preoccupare di questi aspetti. La Dependency Injection(DI) è una specifica implementazione dell’IoC. La DI prevede che tutti gli oggetti all’interno della nostra applicazione accettino le dipendenze, ovvero gli (altri) oggetti di cui hanno bisogno, tramite costruttore o metodi setter. Non sono quindi gli stessi oggetti a creare le proprie dipendenze, ma esse vengono iniettate dall’esterno. Gli oggetti istanziati vengono chiamati bean e sono dichiarati all’interno del progetto, l’IoC container si occupa poi di reperirli e iniettare tutte le dipendenze associati ad essi.

4.2.1 Spring Boot

Spring Boot è un’estensione dello Spring framework che utilizza l’auto-configurazione per velocizzare lo sviluppo. Mentre nella versione "classica" di Spring i bean vengono configurati manualmente, Spring Boot contiene una serie di classi che creano automaticamente i bean necessari. Tra i bean più importanti figura il DispatcherServlet, ovvero un controller responsabile dell’indirizzamento delle HttpServletRequest in ingresso a tutti gli altri controller. Un altro vantaggio di questo framework è che non c’è bisogno di gestire il web server in quanto questa estensione lo fa automaticamente implementando Tomcat.

Per tutta questa serie di vantaggi si è deciso di utilizzare tale framework per lo sviluppo dell'applicativo.

4.2.2 Package Diagram

Il package diagram è fondamentale per analizzare i moduli principali della nostra applicazione e come essi interagiscono tra loro. È stato utilizzato per sottolineare la struttura a layer del sistema, oltre che per evidenziare la presenza degli altri package di supporto. La struttura è di tipo closed poichè ogni layer può utilizzare solamente l'interfaccia del layer immediatamente sottostante. Ogni layer ha responsabilità ben definite e presenta un'interfaccia che astrae gli altri layer dalla particolare implementazione, favorendo così la modificabilità ed il riuso.

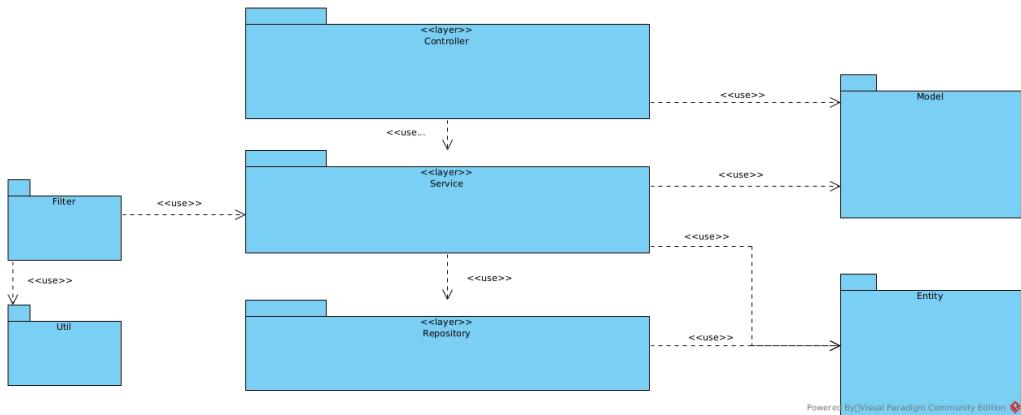


Figura 4.2: Package Diagram

4.2.3 Class Diagram

Per una descrizione ancora più dettagliata del sistema, si è ricorsi ad un class diagram di dettaglio, il quale descrive staticamente la struttura del sistema implementato in termini di classi e package, sottolineando la dipendenze di uso tra di loro. Essendo il sistema di grandi dimensioni, il diagramma risulta poco leggibile e quindi se ne consiglia la visione sul vpp presente nella documentazione.

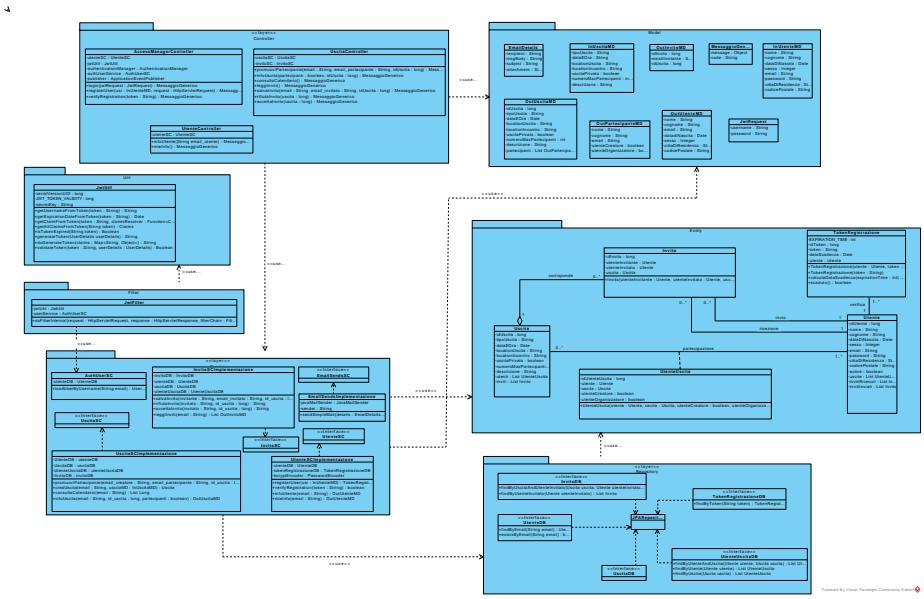


Figura 4.3: Class Diagram

4.2.4 Controller layer

Il controller layer espone le interfacce REST definite al suo interno e riceve richieste dall'esterno. Delega al service layer la vera e propria esecuzione della richiesta.

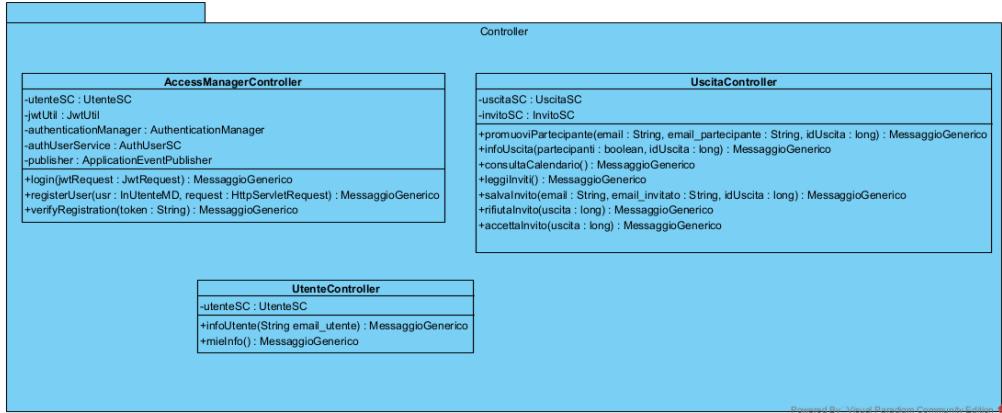


Figura 4.4: Controller layer

4.2.5 Service layer

Il service layer implementa al suo interno la business logic dell'applicazione e delega al repository layer la responsabilità di rendere persistenti i dati modificati.

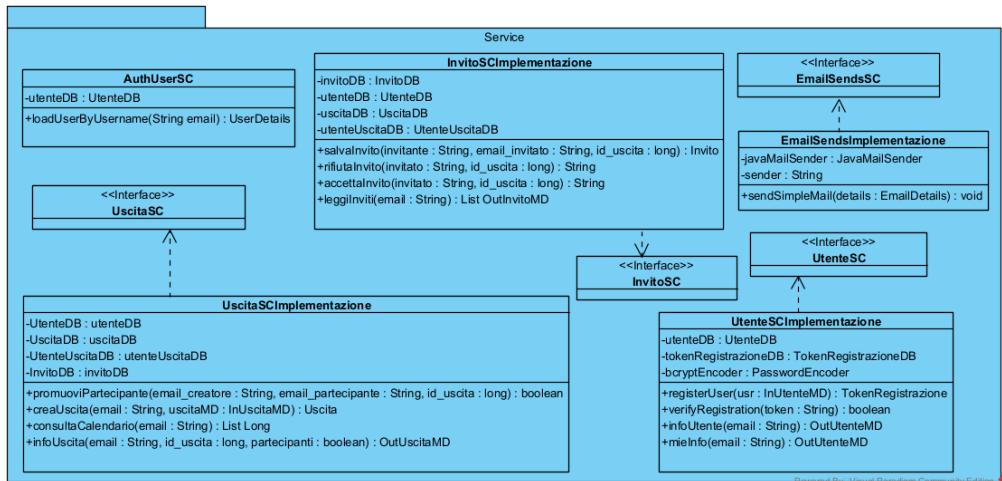


Figura 4.5: Service layer

4.2.6 Repository layer

Il repository layer si interfaccia effettivamente con il database utilizzando statement SQL per eseguire operazioni CRUD. Questo favorisce il disaccoppiamento tra il service ed il database effettivamente utilizzato.

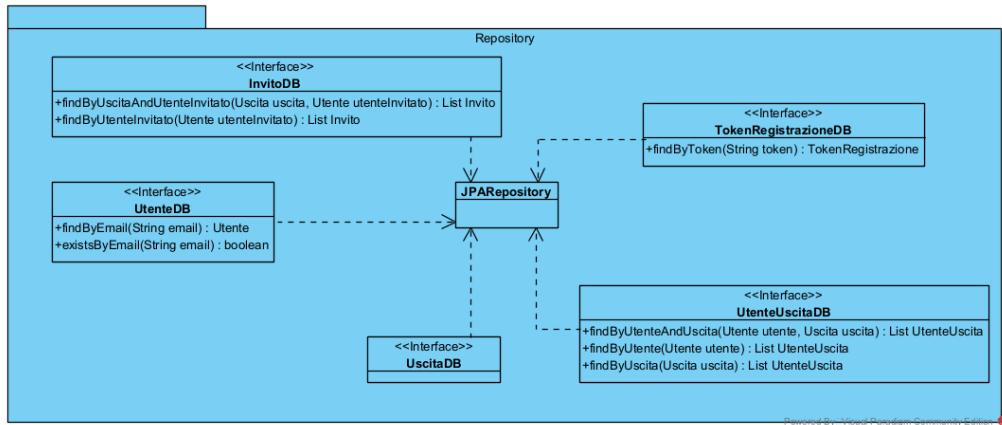


Figura 4.6: Repository layer

4.2.7 Ulteriori package

Il package Entity è fondamentale per il corretto funzionamento dell'architettura. Al suo interno sono definite tutte le classi rappresentati le entità di interesse della la nostra applicazione. Ogni entità è mappata uno ad uno con una tabella all'interno del database.

Il package Model è necessario al Controller per ricevere ed inviare dati a seguito di richieste REST. Risulta fondamentale perchè quando viene richiesto lo stato di un'entità potremmo voler omettere delle proprietà della nostra entità e quindi inviare non l'intero stato, ma una rappresentazione di esso. I package jwtFilter e jwtUtil contengono classi necessarie all'implementazione del login tramite JSON web Token. Si tratta di un open standard (RFC 7519) che definisce un modo sicuro di trasmettere informazioni tra due nodi tramite un JSON. Il client fornisce le credenziali ed una volta verificatane la correttezza il server ritorna un JSON Web Token che il client userà per le future richieste. Il JWT è firmato digitalmente dal server con una chiave privata, in questo modo ci si assicura che nessuno possa manipolarne il contenuto.

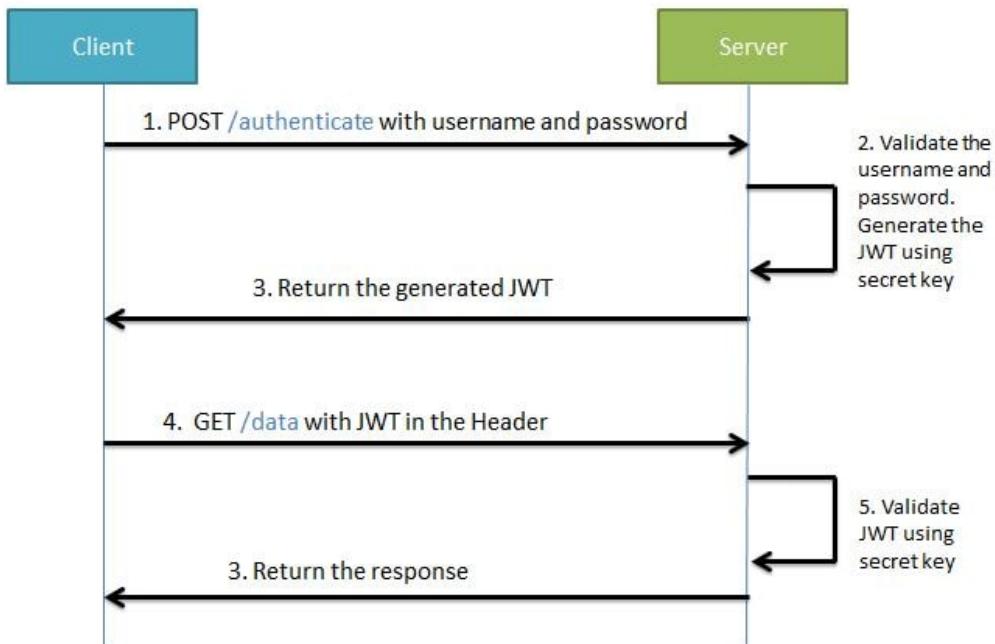


Figura 4.7: Sequence diagram JWT

4.3 Interfacce REST

REST (REpresentational State Transfer) definisce un insieme di principi architetturali per la progettazione di servizi Web basati sul protocollo HTTP. Le API d'interesse prendono quindi il nome di RESTful API e sono basate sull'architettura Client-Server. La comunicazione è stateless, ovvero ogni richiesta è distinta e non connessa. Ogni endpoint è accessibile tramite URL ed è strutturato come segue:

- root-endpoint: rappresenta il punto di partenza dell'API da richiedere
- path: determina la risorsa da richiedere
- eventuali parametri: iniziano sempre con un ? e sono separati da & e vengono utilizzati dal sistema per eseguire delle query.

Quando una richiesta viene inviata tramite un'API RESTful, questa trasferisce al richiedente o all'endpoint uno stato rappresentativo della risorsa. L'informazione, o rappresentazione, viene consegnata in uno dei diversi formati tramite HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP o testo semplice. Il formato JSON è uno dei formati più diffusi, perché, a dispetto del nome, è indipendente dal linguaggio e facilmente leggibile da persone e macchine.

Per i sistemi RESTful risulta di fondamentale importanza una adeguata documentazione delle API messe a disposizione. Per tale ragione si è scelto di documentare le API REST del sistema back-end.

La documentazione è disponibile eseguendo i seguenti passi:

- Installare node.js se necessario.
- Aprire il terminale e spostarsi all'interno della cartella `InterfacceREST` presente all'interno della cartella `doc`.
- Eseguire il comando: `npx serve`
Questo comando creerà un server in esecuzione in `http://localhost:5000` o in un'altra porta specificata dal terminale a seguito dell'esecuzione del comando.
- Aprire il browser ed inserire l'URL.

Nel caso in cui non fosse possibile esaminare la documentazione in questo modo, si riportano delle immagini di seguito.

4.3.1 Accetta invito

POST Accetta Invito

```
localhost:8080/api/accetta-invito?uscita=11
```

L'utente autenticato accetta l'invito relativo all'uscita con id=11

Risposta

```
{
  "message": "Invito accettato",
  "code": "AV_000"
}
```

Headers

Authorization

```
Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJzaW1vbmxvAZ21haWwY29tIiwiZXhwIjoxNjU3ODI0NjE3LCJpYXQiOjE2NT
c4MDY2MTd9.V9wnZ27qUPgtEshx_EaR9xYZr6y-8_NFN5twPS_f6By7BCtqaoIN-
Uk6660fxpQ603DEKny8mt4J03k5HV0XPA
```

Figura 4.8: API accetta invito

4.3.2 Calendario uscite

GET Calendario Uscite

```
localhost:8080/api/calendario-uscite
```

L'utente autenticato ottiene la lista degli id delle uscite a cui partecipa

Risposta

```
{
  "message": [
    9,
    11
  ],
  "code": "CC_000"
}
```

Headers

Authorization

```
Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJzaW1vbmxvAZ21haWwY29tIiwiZXhwIjoxNjU3ODI0NjE3LCJpYXQiOjE2NT
c4MDY2MTd9.V9wnZ27qUPgtEshx_EaR9xYZr6y-8_NFN5twPS_f6By7BCtqaoIN-
Uk6660fxpQ603DEKny8mt4J03k5HV0XPA
```

Figura 4.9: API calendario uscite

4.3.3 Crea invito

POST Crea Invito

localhost:8080/api/uscita/11/invita-partecipante

L'utente autenticato invita l'utente specificato nel body a partecipare all'uscita avente id=11

Risposta

```
{ "message": "Invito inviato correttamente", "code": "SV_000" }
```

response

Headers

Content-Type	application/json
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJyYWZlQGVtM2RKYWlsLmNvbSIsImV4cCI6MTY1NzgyNTM1OSwiwF0IjoxNjU3ODA3MzU5fQ.Lm-ikoGsS1-6B8NC0DZfttY-T24DpdBta6QssNgK5YRh81SVRM3_pxMG1azIRJRND0mh-_hs9oyb7UN931TCdw

Body json

```
{ "email_invitato": "simone@gmail.com" }
```

Figura 4.10: API crea invito

4.3.4 Crea uscita

POST Crea Uscita

localhost:8080/api/crea-uscita

L'utente autenticato crea un'uscita definita nel body

Risposta

```
{  
    "message": "Uscita creata con successo",  
    "code": "CU_000"  
}
```

Headers

Content-Type

application/json

Authorization

```
Bearer  
eyJhbGciOiJIUzIzUmJ9.eyJzdWIiOiJyYWZlQGVtM2RKYW1sLmNvbSIsImV4cCI6MTY1NzgyNTM1OSwiwF0IjoxNj  
U3ODAAMzU5fQ.Lm-1koGsS1-6BBNC0DZFttY-T24DpdBta6QssNgK5YRh81SVRM3_pxMGlaZIRJRND0mh-  
_hs9oyb7UN931TCdw
```

Body **json**

```
{  
    "tipoUscita": "Passeggiata",  
    "dataOra": "2022-07-22T18:30",  
    "locationUscita": "Dante",  
    "locationIncontro": "Via Roma",  
    "uscitaPrivata": false,  
    "numeroMaxPartecipanti": 15,  
    "descrizione": "Shopping a via Roma"  
}
```

Figura 4.11: API crea uscita

4.3.5 Info uscita con partecipanti

GET Info Uscita con Partecipanti

localhost:8080/api/uscita/9

Ottieni le informazioni relative all'uscita avente id=9, le informazioni contengono una lista di partecipanti. L'utente autenticato deve far parte dell'uscita.

Risposta

```
{
  "message": {
    "idUscita": 9,
    "tipoUscita": "Cena",
    "dataOra": "2022-07-22T18:30:00.000+00:00",
    "locationUscita": "Vomero",
    "locationIncontro": "Dante",
    "uscitaPrivata": false,
    "numeroMaxPartecipanti": 15,
    "descrizione": "Cena con amici al vomero",
    "partecipanti": [
      {
        "nome": "Simone",
        "cognome": "D'Orta",
        "email": "simone@gmail.com",
        "utenteCreatore": true,
        "utenteOrganizzatore": false
      },
      {
        "nome": "Raffaele",
        "cognome": "Del Gaudio",
        "email": "rafe@emddail.com",
        "utenteCreatore": false,
        "utenteOrganizzatore": true
      }
    ],
    "code": "IU_000"
  }
}
```

Parameters

partecipanti	true
--------------	------

Headers

Authorization	Bearer eyJhbGciOiJIUzIwMiJ9.eyJzdWIiOiJzaW1vbmlAZZ21haWwvY29tLjwiZXhwIjoxNjU3ODI2MjUwLCJpYXQiOjE2NTc4MDgyNTB9.PjNKGZW7VK65ehCSNbndVpta_qeQ08fHENQBMVhcDowp1uaMa0b9M7g8dRwyeyCiBuqQ0dsA1VviVpOUZwaBg
---------------	---

Figura 4.12: API info uscita con partecipanti

4.3.6 Info uscita senza partecipanti

GET Info Uscita senza Partecipanti

localhost:8080/api/uscita/9

Ottieni le informazioni relative all'uscita avente id=9, le informazioni omettono la lista di partecipanti. L'utente autenticato deve far parte dell'uscita.

Risposta

```
{  
    "message": {  
        "idUscita": 9,  
        "tipoUscita": "Cena",  
        "dataOra": "2022-07-22T18:30:00.000+00:00",  
        "locationUscita": "Vomero",  
        "locationIncontro": "Dante",  
        "uscitaPrivata": false,  
        "numeroMaxPartecipanti": 15,  
        "descrizione": "Cena con amici al vomero",  
        "partecipanti": null  
    },  
    "code": "IU_000"  
}
```

Parameters

partecipanti false

Headers

Authorization

```
Bearer  
eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJzaW1vbmvAZ21halwuY29tIiwiZXhwIjoxNjU3ODI2MjUwLCJ  
pYXQiOjE2NTc4MDgyNTB9.PjNkGZw7VK65ehCSNbndVptq_qe008fHENQBMVHcDowp1UaMa0b9M7g8dR  
wyeyCiBuqQ0dsAlVviVpOUZWaBg
```

Figura 4.13: API info uscita senza partecipanti

4.3.7 Leggi Inviti

GET Leggi Inviti

localhost:8080/api/leggi-inviti

L'utente autenticato ottiene la lista di inviti per i quali lui è utente invitato

Risposta

```
{  
    "message": [  
        {  
            "idInvito": 13,  
            "emailInvitante": "rafe@em3ddail.com",  
            "idUscita": 11  
        },  
        {  
            "idInvito": 16,  
            "emailInvitante": "rafe@em3ddail.com",  
            "idUscita": 14  
        }  
    ],  
    "code": "LV_000"  
}
```

Headers

Authorization

```
Bearer  
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJzaW1vbmcVAZ21halwuY29tIiwizXhwIjoxNjU3ODI0NjE3LCJ  
pYXQiOjE2NTc4MDY2MTd9.V9wnZ27qUPgtEsHx_EaR9xYZr6y-8_NFN5twPS_f6By7BCtqaoIN-  
Uk6660fxpQ603DEKny8mt4J03k5HV0XPA
```

Figura 4.14: API leggi inviti

4.3.8 Login

POST Login Utente

localhost:8080/api/login

L'utente si autentica tramite le credenziali inserite nel body. Nel campo message del body di ritorno è presente il JWT necessario all'utente per eseguire richieste autenticate.

Risposta

```
{  
    "message":  
        "eyJhbGciOiJIUzIwMiJ9.eyJzdWIiOiJzak1vbmlVAZ21halwuY29tIiwiZXhwIjoxNjU3ODI0NjE3LCJpYXQiOjE2NTc4MDY2MTd9.V9wnZ27qUPgtEsHx_E  
        "code": "LG_000"  
}
```

Headers

Content-Type application/json

Body **json**

```
{  
    "username": "simone@gmail.com",  
    "password": "password"  
}
```

Figura 4.15: API login

4.3.9 Promuovi un partecipante

POST Promuovi Partecipante

localhost:8080/api/uscita/9/promuovi-partecipante

Se l'utente autenticato è creatore dell'uscita avente id=9, allora l'utente specificato nel body viene promosso ad utente organizzatore

Risposta

```
{  
    "message": "Utente promosso ad Organizzatore",  
    "code": "PP_000"  
}
```

Headers

Content-Type application/json

Authorization

```
Bearer  
eyJhbGciOiJIUzIwMiJ9.eyJzdWIiOiJzaW1vbmlVAZ21haWwuY29tIiwiZXhwIjoxNjU3ODI0NjE3LCJpYXQiOjE2NT  
c4MDY2MTd9.V9wnZ27qUPgtEsHx_EaR9xYZr6y-8_NFN5twPS_f6By78CtqaoIN-  
Uk6660fxpQ603DEKny8mt4J03k5HV0XPA
```

Body **json**

```
{  
    "email_partecipante": "rafe@em3ddail.com"  
}
```

Figura 4.16: API promuovi un partecipante

4.3.10 Registra utente

POST Registra Utente

localhost:8080/api/registrazione

Effettua la registrazione di un utente specificato nel body

Risposta

```
{  
    "message": "Registrazione completata con successo. Clicca sul link che riceverai per email per attivare il tuo account.",  
    "code": "RG_000"  
}
```

Headers

Content-Type	application/json
--------------	------------------

Body **json**

```
{  
    "nome": "Simone",  
    "cognome": "D'Orta",  
    "dataDiNascita": "1997-10-10",  
    "sesto": 0,  
    "email": "simone@gmail.com",  
    "password": "password",  
    "cittaDiResidenza": "Napoli",  
    "codicePostale": "80141"  
}
```

Figura 4.17: API regista utente

4.3.11 Rifiuta invito

POST Rifiuta Invito

localhost:8080/api/rifiuta-invito?uscita=14

L'utente autenticato rifiuta l'invito relativa all'uscita con id=14

Risposta

```
{  "message": "Invito rifiutato",  "code": "RV_000"}
```

Headers

Authorization

```
Bearer eyJhbGciOiJIUzIwMiJ9.eyJzdWIiOiJzaW1vbmcVAZ21haWwuY29tIiwiZXhwIjoxNjU3ODI0NjE3LCJpYXQiOjE2NTc4NDY2MTd9.V9wnZ27qUPgtEshx_Eak9xyZr6y-8_NFN5twPS_f6By78CtqaoIN-Uk666OfxpQ6o3DEKny8mt4J03k5HV0XPA
```

Figura 4.18: API rifiuta invito

4.3.12 Verifica registrazione

GET Verifica registrazione

```
localhost:8080/api/verifica-registrazione
```

Attiva l'utente avente token=5173425c-0c64-42bc-bcd6-785eedd5eb9b

Risposta

```
{  
    "message": "Account verificato",  
    "code": "VR_000"  
}
```

Parameters

token	5173425c-0c64-42bc-bcd6-785eedd5eb9b
-------	--------------------------------------

Figura 4.19: API verifica registrazione

4.3.13 Bad Request

In seguito è riportato il formato standard della risposta in seguito ad una bad request.

POST BAD REQUEST

```
localhost:8080/api/crea-uscita
```

Nel caso in cui venga effettuata una bad request, è possibile comprenderne il motivo nel campo message del response body.

Risposta

```
{  
    "message": "La data e l'ora dell'uscita devono essere nel futuro",  
    "code": "NF_001"  
}
```

Figura 4.20: Formato risposta bad request

4.3.14 Login

POST Login Utente

localhost:8080/api/login

L'utente si autentica tramite le credenziali inserite nel body. Nel campo message del body di ritorno è presente il JWT necessario all'utente per eseguire richieste autenticate.

Risposta

```
{  
    "message":  
        "eyJhbGciOiJIUzI1nXMiJ9.eyJzdWIiOiJzak1vbmlVAZ21halwUY29tIiwiZXhwIjoxNjU3ODI0NjE3LCJpYXQiOjE2NTc4MDY2MTd9.V9wnZ27qUPgtEsHx_E  
        "code": "LG_000"  
}
```

Headers

Content-Type	application/json
--------------	------------------

Body **json**

```
{  
    "username": "simone@gmail.com",  
    "password": "password"  
}
```

Figura 4.21: Interfaccia login

4.4 Database

La scelta del DBMS è una delle scelte più importanti nell'ambito di una applicazione web in quanto essa influisce sulle performance dell'intero sistema.

Nell'ambito di questo progetto si è scelto di utilizzare H2, un database in-memory, per la sua facilità di configurazione e per la sua portabilità.

Springboot fornisce un ampio supporto, tramite la libreria JPA, a diversi DBMS, quindi sarà molto facile eventualmente in futuro cambiare modello di database. Questa facilità di cambiamento è stata da noi supportata sviluppando una fornita test suite anche per il livello di repository così da poter testare in automatico il corretto funzionamento di qualsiasi DBMS si scelga di utilizzare.

In figura seguente viene mostrato il Data model del nostro sistema.

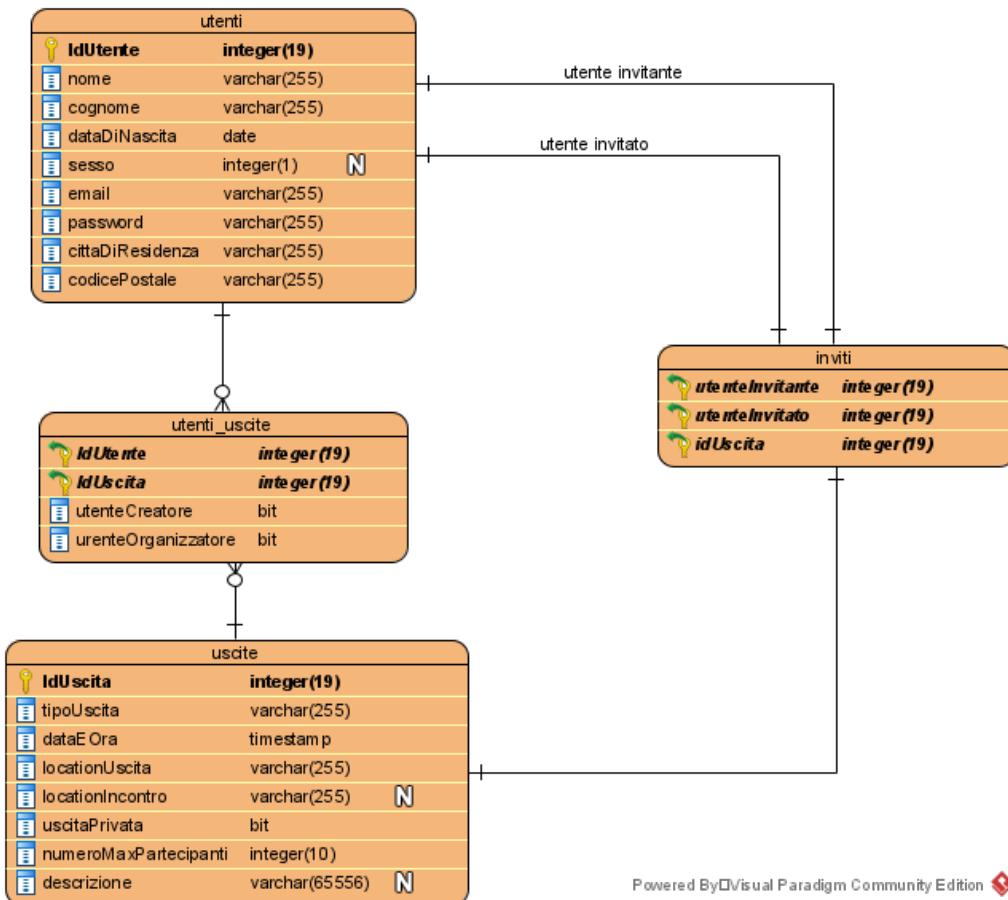


Figura 4.22: Data Model

4.5 React Front-end

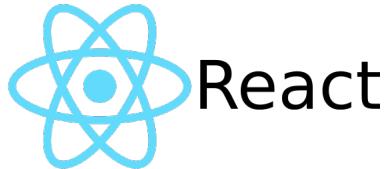


Figura 4.23: Logo di react

L'applicazione che viene eseguita sul browser è effettivamente il punto di interfacciamento tra l'intero sistema e l'utente finale. Essa deve essere dunque reattiva, veloce e con un design accattivante. Per raggiungere questi obiettivi si è deciso di utilizzare una delle migliori librerie in circolazione, ovvero React.

React è una libreria JavaScript ideata per costruire interfacce utente dinamiche e complesse. Il linguaggio utilizzato da react è chiamato JSX, questo adotta un approccio dichiarativo molto simile all'HTML, ma a differenza di quest'ultimo non è in grado di essere interpretato dal browser. Per questo motivo è necessario un pre-compilatore, Babel, in grado di tradurre il codice in JavaScript.

Per generare dinamicamente le interfacce c'è bisogno di mantenere uno stato ed essere in grado di modificarlo. Questa funzione è offerta dagli hooks, ovvero delle funzioni messe a disposizione dalla libreria React. L'hook useState è in grado di definire una variabile di stato(ed un metodo per modificarla) condivisa e preservata tra le ri-renderizzazioni all'interno del componente React, si può poi definire una logica basata su queste variabili per creare un'interfaccia dinamica. Inoltre, una modifica ad una variabile porta ad una nuova renderizzazione automatica dei componenti dipendenti da essa.

Un altro hook molto utile è l'useEffect, attraverso il quale è possibile definire una funzione che viene chiamata ogni volta che la pagina è renderizzata oppure quando viene modificata una delle variabili indicate nel vettore delle dipendenze. Questo hook si rivela molto utile ad esempio, quando bisogna popolare inizialmente la pagina attraverso dei dati che arrivano in risposta a richieste REST.

Inviare codice JSX al browser, implica la necessità di dover inviare anche il pre-compilatore Babel e tutto ciò risulterebbe in un overhead dato dalle dimensioni del compilatore (nell'ordine dei MB) e dal tempo impiegato dalla pre-compilazione. In produzione dunque si sceglie di effettuare la pre-compilazione in locale, in modo tale da ottenere script pronti ad essere ese-

guiti dal browser. È necessario dunque un ambiente runtime per l'esecuzione di codice JavaScript come Node.js. I file HTML, CSS e JavaScript prodotti in questo modo possono poi essere ospitati sul server che comunicherà con il browser.

4.5.1 React component diagram

Questo diagramma mostra la gerarchia e le dipendenze tra i componenti sviluppati in React. In ogni componente sono specificate le variabili di stato («state») e le proprietà («prop»). Nel componente <app/> sono definite le corrispondenze tra URL e componenti.

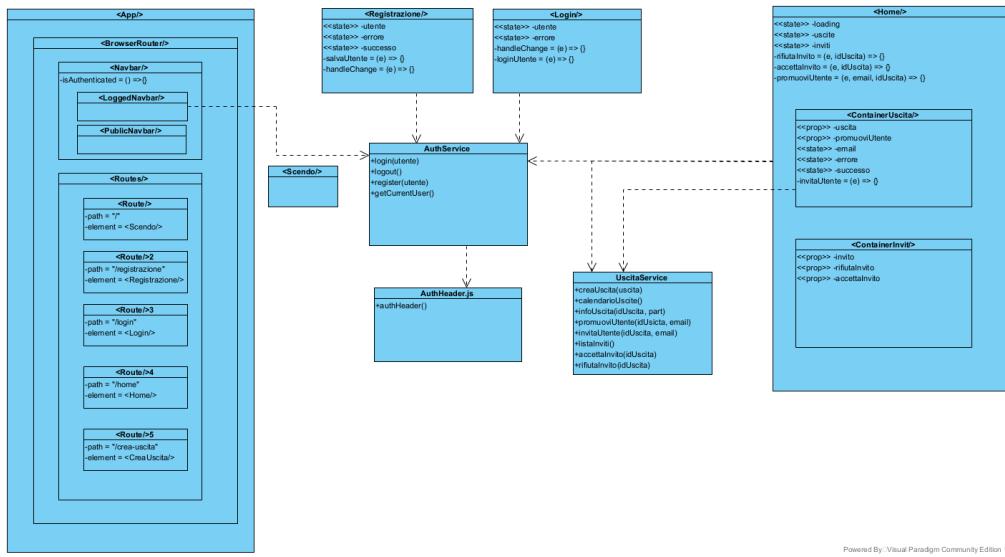


Figura 4.24: Front end component diagram

4.6 Sequence di dettaglio

È stato sviluppato un sequence diagram di dettaglio per ogni caso d'uso, utile per analizzare le chiamate REST effettuate dal client ed il flusso di esecuzione all'interno dell'applicazione sviluppata in Spring.

4.6.1 Accetta invito

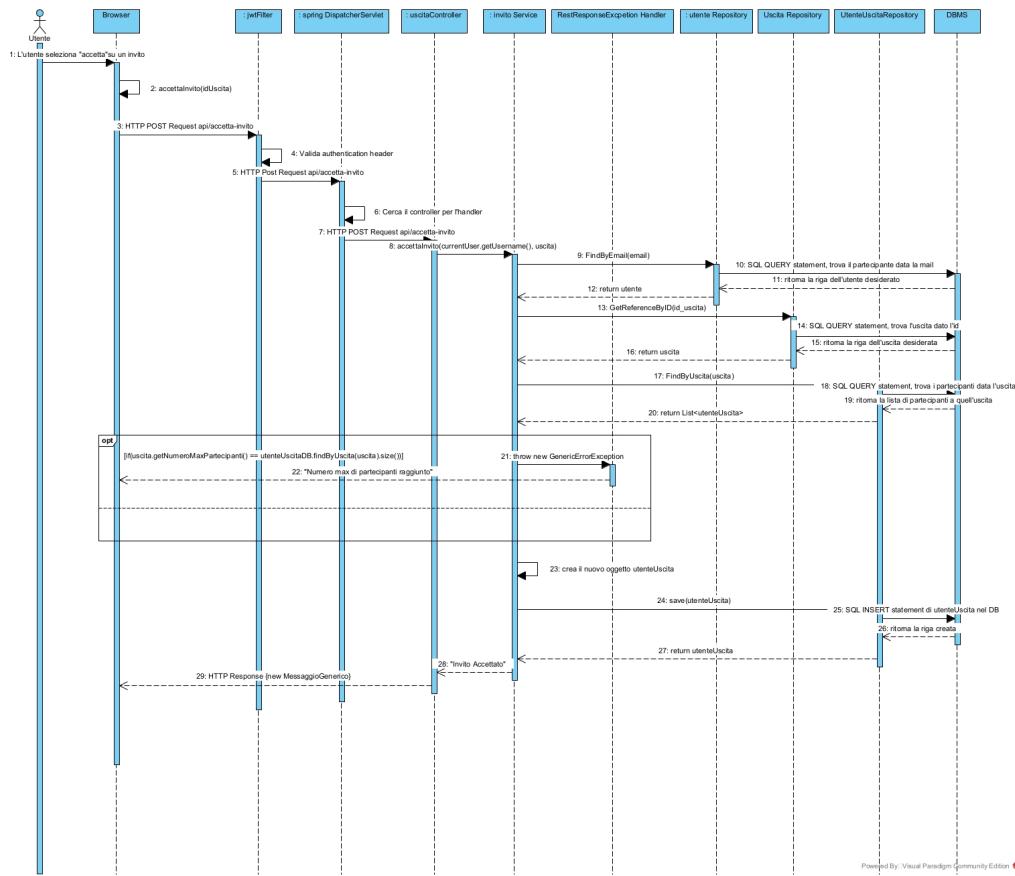


Figura 4.25: Sequence diagram Accetta invito

4.6.2 Consulta calendario uscite

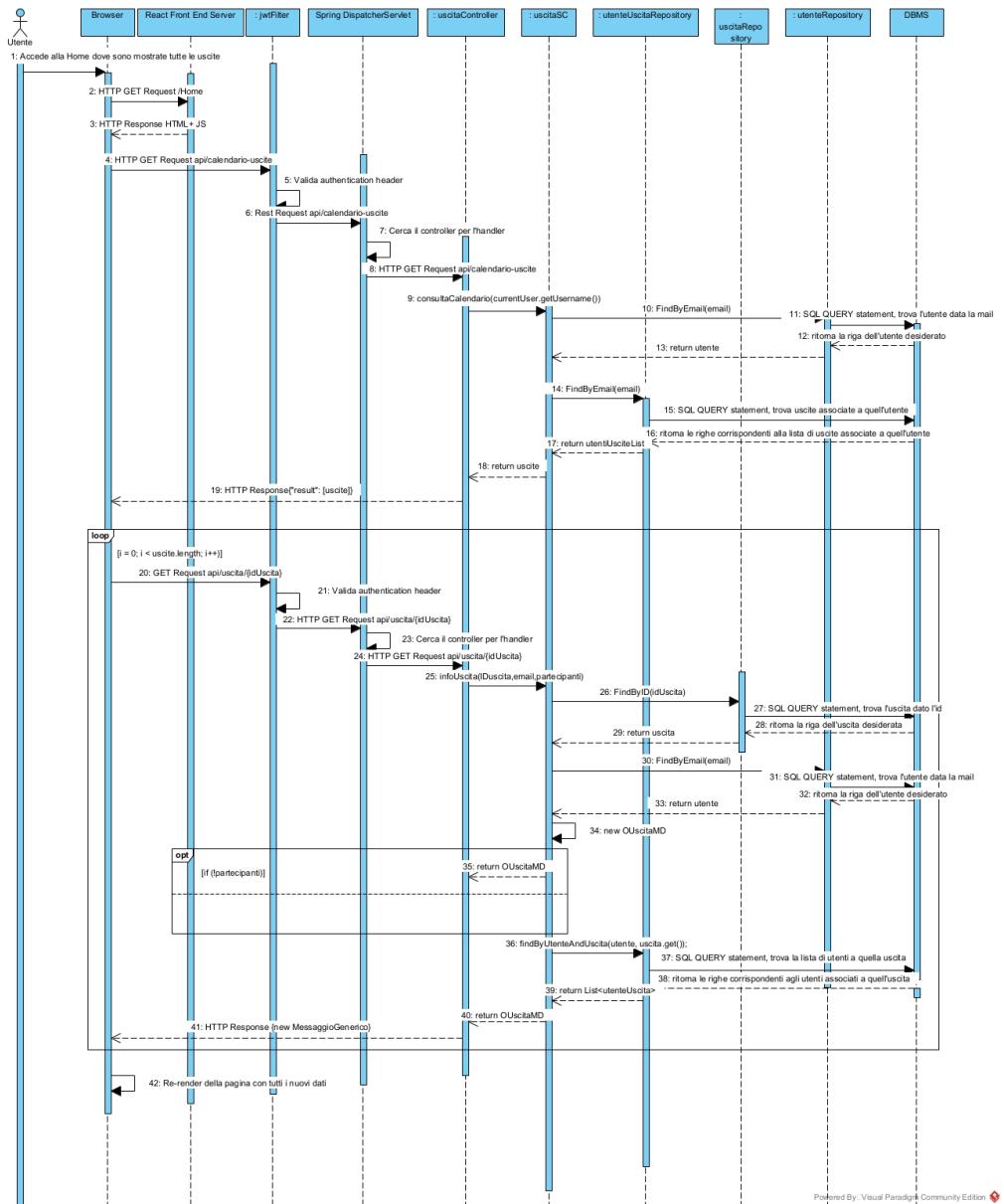


Figura 4.26: Sequence diagram Consulta calendario uscite

4.6.3 Crea nuova uscita

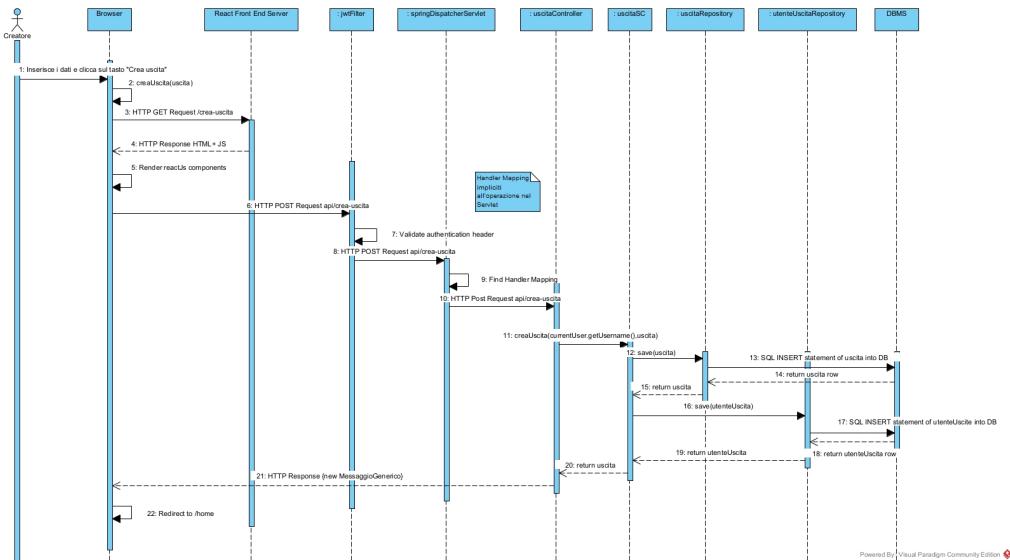


Figura 4.27: Sequence diagram Crea nuova uscita

4.6.4 Invia email registrazione

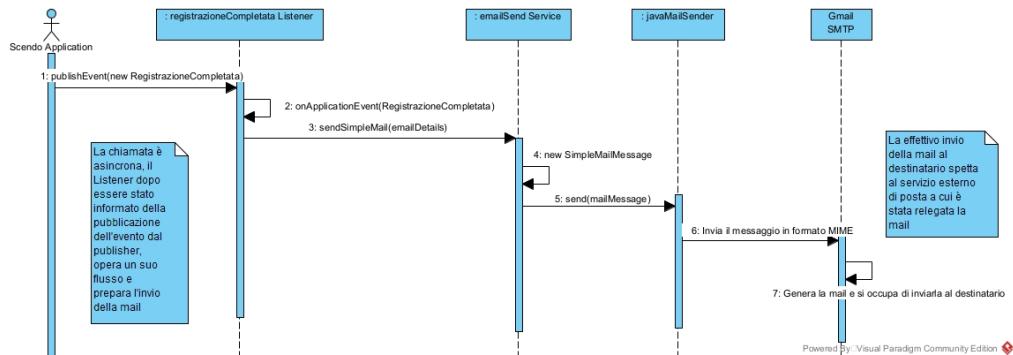


Figura 4.28: Sequence diagram Invia email registrazione

4.6.5 Invita utente

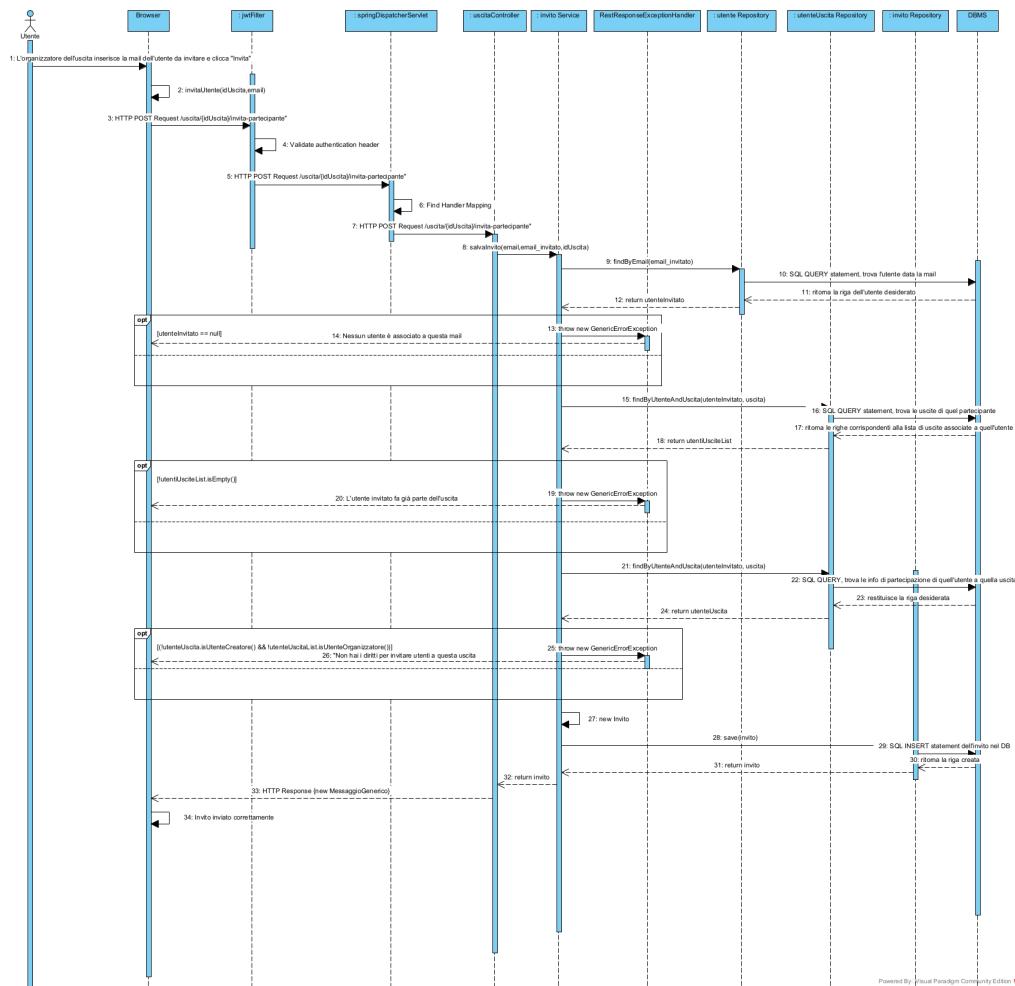


Figura 4.29: Sequence diagram Invita utente

4.6.6 Promuovi partecipante

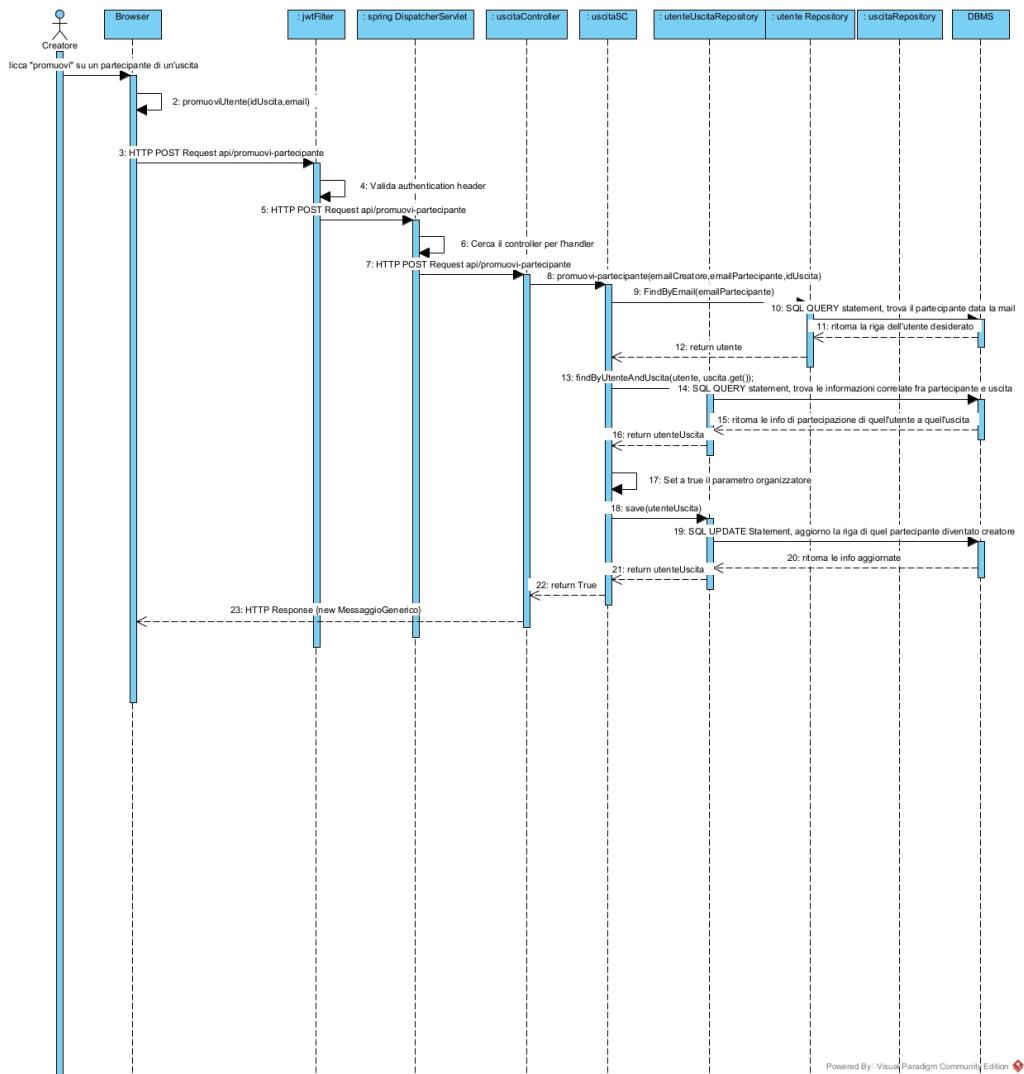


Figura 4.30: Sequence diagram Promuovi partecipante

4.6.7 Registrazione

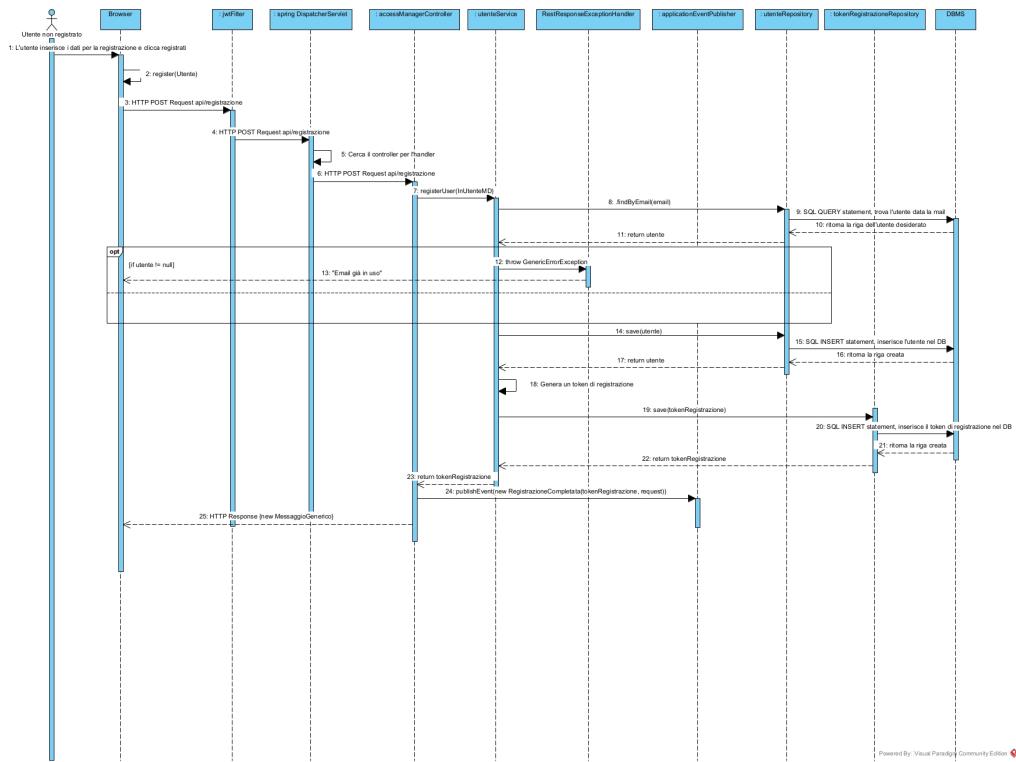


Figura 4.31: Sequence diagram Registrazione

4.7 Deployment Diagram

Il diagramma in fig. 4.32 mostra la configurazione di deploy del sistema e contestualmente fornisce una vista di installazione attraverso lo stereotipo «manifest».

Il back-end server viene buildato in un solo file jar che include al suo interno un webserver Tomcat 9.0.63, quindi non è necessario fare altro che avviare tale jar per far partire una istanza di Tomcat in cui viene effettuato in automatico il deploy dei servizi del backend.

Il front-end invece viene buildato in un'insieme di file Javascript, html e CSS. Per questo progetto si è scelto di utilizzare l'applicativo SERVE per deployarlo.

4.7.1 Avviamento del sistema

L'avviamento del sistema è composto di due step:

- Avviamento del Backend server tramite il comando `./run-backend` eseguito nella cartella `run` del progetto.
- Avviamento del Frontend server tramite il comando `./run-frontend` eseguito nella cartella `run` del progetto.

Per lo spegnimento del sistema è sufficiente arrestare i due processi creati in precedenza.

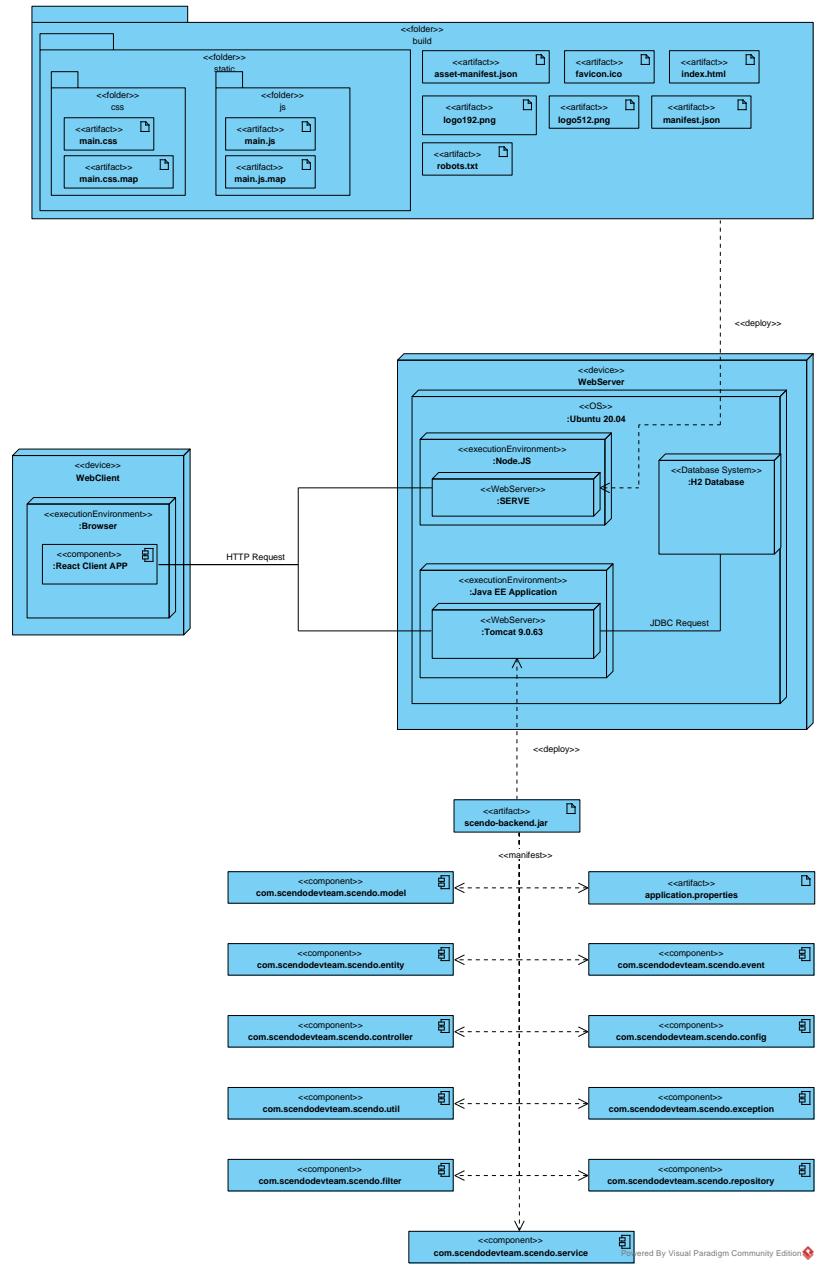


Figura 4.32: Deployment diagram

Capitolo 5

Testing

Il testing è una delle attività fondamentali a corredo dello sviluppo software perché permette di verificarne la correttezza. Ad oggi esistono metodologie di sviluppo basate sulla costruzione dei test ancor prima del codice del software.

In questo progetto si è utilizzato un approccio tradizionale al testing, dove il sistema software è stato testato alla fine di ogni iterazione.

5.1 Test d'Unità

Il Mercoledì della seconda settimana della prima interazione è stato svolto dal team di sviluppo un insieme di test di unità per verificare il comportamento delle singole entità.

Per eseguire questi test si è utilizzato JUnit in combinata con le annotazioni di Spring e Mockito per generare gli Stub delle classi richiamate da quelle sotto test. Sono state testate tutte le funzionalità di tutte le classi del Controller, Service e Repository cercando di coprire diversi scenari di esecuzione costruendo in totale 61 test case. I test case sono quelli inseriti nel package `units` presente in `scendo/src/test/java/com/scendodevteam/scendo`

I test sono automatici ed è possibile rieseguirli con il comando:
`./mvnw -Dtest=\!EndToEndTest test`
eseguito nella cartella `scendo` oppure, se si utilizza VSCode, con il supporto dell'interfaccia grafica.

5.2 Test End To End

Il giorno dopo dei test di unità sono stati eseguiti quelli End-To-End per verificare il corretto funzionamento delle Rest API del sistema backend in maniera complessiva. Tali test, avvenuti in presenza del Committente, sono stati realizzati sempre tramite l'ausilio di JUnit e Spring.

Questi test sono stati inseriti nel package `e2e` presente in `scendo/src/test/java/com/scendodevteam/scendo` e sono rieseguibili con il comando:

```
./mvnw -Dtest=EndToEndTest test
```

eseguito nella cartella `scendo` oppure, se si utilizza VSCode, con il supporto dell'interfaccia grafica.

5.3 Test di User Acceptance

Il Venerdì della seconda settimana della seconda iterazione sono stati eseguiti i test di user acceptance. Questi test sono stati condotti in presenza del cliente per mostrare il funzionamento completo del sistema compreso di interfaccia grafica.

Questi test sono stati eseguiti manualmente e per questo è stato prodotto un testbook per definire tutti i testcase, i passaggi da eseguire e gli esiti attesi per confermare superato il test. Il documento è consultabile in `doc/test/Scendo_UAT_Testbook.xlsx` oppure in fig. 5.1, mentre la registrazione video dell'esecuzione dei test a <https://youtu.be/YswbybwsVHg>.

User Acceptance Tests						
ID	Scenario Description	Status	Owner	Expected Outcome	Results	Notes
1 Registro utente						
1.1	- L'utente naviga su www.scendo.com/registrazione - Inserisce tutti i dati richiesti dal form in maniera corretta - Clicca sul pulsante "Crea Account"	Passed	Scendo team	- Nessun messaggio di errore viene visualizzato a schermo - L'utente viene redirezionato alla pagina di login		
1.2	- L'utente naviga su www.scendo.com/registrazione - Inserisce tutti i dati richiesti dal form ma inserisce un'email già presente nel database di utenti - Clicca sul pulsante "Crea Account"	Passed	Scendo team	- L'utente riceve un messaggio di errore che gli indica la presenza di un altro account nel sistema con la stessa email		
1 Verifica Registrazione						
1.1	- L'utente clicca sul link presente nell'email	Passed	Scendo team	- L'utente visualizza un messaggio che conferma l'avvenuta attivazione del suo account		
2 Login utente						
2.1	- L'utente naviga su www.scendo.com/login - Inserisce l'email e la password correttamente - Clicca sul pulsante "Login"	Passed	Scendo team	- L'utente viene redirezionato correttamente alla Home del sito dove gli vengono mostrate la lista delle sue uscite e degli inviti ricevuti		
3 Creazione Uscita						
3.1	- L'utente naviga su www.scendo.com/crea-uscita - Compila tutti i campi del form correttamente - Clicca sul pulsante "Crea Uscita"	Passed	Scendo team	- Nessun errore viene mostrato nell'interfaccia - L'utente viene redirezionato correttamente alla Home del sito dove gli vengono mostrate la lista delle sue uscite e degli inviti ricevuti		
3.2	- L'utente naviga su www.scendo.com/crea-uscita - Compila il form lasciando vuoto un campo obbligatorio o immettendo una data di uscita non valida - Clicca sul pulsante "Crea Uscita"	Passed	Scendo team	- L'utente riceve un messaggio che lo informa di tutti gli errori commessi nel form		
4 Inviti						
4.1	- L'utente creatore od organizzatore dell'uscita inserisce l'email della persona che desidera invitare nell'apposito box - Clicca il tasto "Invita"	Passed	Scendo team	- L'utente invitato visualizza nella sua Home, nel pannello degli inviti, un nuovo invito con le informazioni dell'utente invitante e dell'uscita		
4.2	- L'utente invitato accetta l'invito cliccando l'apposito tasto nel box dell'invito	Passed	Scendo team	- Il box dell'invito scompare dalla pagina dell'utente invitato - L'utente invitato visualizza la nuova uscita nella Home insieme alle altre - L'utente invitante vede l'utente invitato tra i partecipanti dell'uscita a seguito di un refresh della pagina		
4.3	- L'utente invitato rifiuta l'invito cliccando l'apposito tasto nel box dell'invito	Passed	Scendo team	- Il box dell'invito scompare dalla pagina dell'utente invitato		
5 Promozione utente						
5.1	- L'utente creatore dell'uscita clicca il tasto "Promuovi" in corrispondenza dell'utente appena promosso si disabilita per prevenire molteplici promozioni - L'utente promosso acquisisce la facoltà di invitare persone all'uscita vedendo comparire l'apposito box	Passed	Scendo team	- Il tasto "Promuovi" in corrispondenza dell'utente appena promosso si disabilita per prevenire molteplici promozioni - L'utente promosso acquisisce la facoltà di invitare persone all'uscita vedendo comparire l'apposito box		

Figura 5.1: User Acceptance Test book

Capitolo 6

Sviluppi futuri

Possibili evoluzioni e sviluppi futuri di questa applicazione potrebbero estendere il comparto di funzionalità messe a disposizione del sistema per gli utenti.

Lo sviluppo di un'applicazione per smartphone potrebbe essere una ottima evoluzione e sarebbe fattibile grazie alle scelte architettoniche fatte in principio. In aggiunta alla creazione di un nuovo client, sicuramente potrebbe essere interessante migliorare quello già sviluppato.

Poiché nelle due iterazioni svolte non sono stati implementati tutti i requisiti funzionali, in futuri incrementi vanno realizzati quelli rimanenti e potrebbero essere svolti workshop per identificarne di nuovi.