

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA INSEGNAMENTO DI

INGEGNERIA DEL SOFTWARE I ANNO ACCADEMICO 2021-2022

**Specifiche, progettazione, implementazione e
validazione del Sistema Informativo
“NaTour22”**

Autori:

GRUPPO: INGSW2122_S_14

Fabiola Salomone

MATRICOLA N86/2870 fab.salomone@studenti.unina.it

Docente:

Prof. Sergio Di Martino

Prof. Francesco Cutugno

Indice

Capitolo I – Descrizione del progetto	5
○ Introduzione	5
Capitolo II – Modello Funzionale	6
○ Documento dei Requisiti Software	6
- Modello Funzionale: Applicazione Mobile	6
• Requisiti Funzionali	6
• Requisiti non Funzionali	9
- Modellazione casi d’uso	10
- Tabella di Cockburn dei casi d’uso	11
• Registrazione alla piattaforma	11
• Inserimento itinerario	13
- Mock-up dell’applicazione mobile	15
• Registrazione alla piattaforma	15
• Inserimento itinerario	16
- Presentazione idea progettuale	18
- Individuazione target utenti	19
- Valutazione dell’usabilità	21
- Prototipazione funzionale via statechart	23
• Registrazione alla piattaforma	23
• Inserimento itinerario	24
- Glossario	25
Capitolo III – Modello di Dominio	27
○ Documento dei Requisiti Software	27
- Modello di Dominio: Applicazione Mobile	27
- Diagrammi delle classi di analisi	27
• Registrazione alla piattaforma	28
• Accedi alla piattaforma	29
• Uscita dalla piattaforma	30
• Visualizza lista itinerari	31
• Visualizza informazioni itinerario più posizione utente su mappa	32

• Visualizza profilo	33
• Inserimento recensioni	34
• Inserimento itinerario	35
• Inserimento tappa lungo un itinerario	36
• Visualizza lista recensioni	37
- Diagrammi di sequenza di analisi	38
• Registrazione alla piattaforma	38
• Accesso alla piattaforma	38
- Diagrammi di attività	39
• Registrazione alla piattaforma	39
• Accesso alla piattaforma	40
• Visualizza lista itinerario	41
• Visualizza informazioni che dettaglia l’itinerari	42
• Visualizza lista recensioni	43
• Inserimento recensione	44
• Indicare posizione dell’utente sulla mappa	45
• Inserimento punti d’interesse sulla mappa	46
• Inserimento nuovo itinerario	47
• Uscita dalla piattaforma	48
• Visualizza profilo	49
Capitolo IV – Design	50
○ Documento di Design del sistema	50
- Analisi dell’architettura	50
• Architettura esterna	50
• Architettura Server	51
• Architettura Client	54
- Modelli di Design: Applicazione Mobile	56
• Diagrammi delle classi di design	56
- Accedi alla piattaforma	56
- Registrazione alla piattaforma	57
- Visualizza lista itinerari	58
- Visualizza dettagli itinerari	59
- Visualizza lista recensioni	60

- Inserimento recensione	61
- Inserimento nuovo itinerario	62
- Inserimento tappa lungo un itinerario	63
- Visualizza profilo	64
• Diagrammi di sequenza di design	65
- Registrazione alla piattaforma	65
- Inserimento recensione	66
• Definizione della gerarchie funzionali	67
Capitolo V – Codice	68
○ Codice Sorgente sviluppato	68
Capitolo IV – Testing	69
○ Documento di Testing del sistema	69
- Codice xUnit per unit testing di 2 metodi	69
• Corrispondenza package progetto e stringa corrispondente	69
• Accesso all'applicazione	69
• Registrazione all'applicazione	71
Capitolo VII – Valutazione usabilità	75
○ Valutazione sull'usabilità sul prodotto finito	75

Capitolo I – Descrizione del progetto

O Introduzione

Si vuole realizzare un sistema informatico, denominato *NaTour22* il cui scopo è quello di offrire una piattaforma di supporto per le escursioni.

L'applicazione mobile dovrà consentire agli utenti:

- Di registrarsi e accedere alla piattaforma

L'utente può registrarsi alla piattaforma mediante email, password, nickname, data di nascita e genere ed accedere mediante email e password.

- Di inserire nuovi itinerari

L'utente può inserire itinerari indicando il titolo, livello di difficoltà (difficile, medio, semplice), il tempo impiegato (ore, minuti, secondi), una descrizione e un tracciato.

Il tracciato, potrà inserirlo manualmente interagendo con una mappa, o tramite file in formato GPX.

- Di visualizzare una schermata di dettaglio per ciascun itinerario

L'utente può visualizzare una schermata nella quale vengono mostrate le informazioni dettagliate sul itinerario, recensioni e fotografie. Inoltre il sistema offrirà una funzionalità aggiuntiva in cui l'utente può inserire una recensione indicando il titolo, la descrizione, la valutazione e un'immagine.

- Di inserire una tappa, lungo un itinerario

L'utente può inserire punti di interesse in un itinerario, indicando la tipologia e la posizione in cui deve essere inserita la tappa lungo l'itinerario.

- Di mostrare la propria posizione nella schermata di dettaglio di un itinerario.

L'utente può inserire la propria posizione nella schermata di dettaglio di un itinerario tramite una spunta.

Tutto ciò verrà sviluppato in *Android Studio* in un linguaggio di programmazione *Object Oriented*, in particolare *Java*, affiancato da servizi di public *Cloud Computing* come *Firebase*, al fine di massimizzare la scalabilità del sistema in vista di un possibile aumento del numero degli utenti.

Capitolo II – Modello Funzionale

O Documento dei Requisiti Software

- Modello Funzionale: Applicazione Mobile

• Requisiti Funzionali

Vengono ora elencati i requisiti funzionali ovvero tutti i servizi (o funzioni) che il sistema dovrà offrire.

ID	APPM_REQF01
Nome	Accesso alla piattaforma
Descrizione	<p>Il sistema deve consentire ad un utente registrato di poter effettuare l'accesso alla piattaforma indicandone</p> <ul style="list-style-type: none">• E-mail• Password.
ID	APPM_REQF02
Nome	Registrazione alla piattaforma
Descrizione	<p>Il sistema deve consentire ad un utente non registrato di potersi registrare alla piattaforma indicando:</p> <ul style="list-style-type: none">• E-mail• Nickname• Password• Data di nascita• Genere.
ID	APPM_REQF03
Nome	Visualizzare la lista degli itinerari presenti nella piattaforma
Descrizione	<p>Il sistema deve consentire ad un utente registrato di visualizzare tutti gli itinerari presenti sulla piattaforma.</p>

ID	APPM_REQF04
Nome	Visualizzare una schermata di dettaglio itinerario
Descrizione	Il sistema deve consentire ad un utente registrato di poter visualizzare una schermata di dettaglio dell’itinerario nel quale si troveranno tutte le informazioni dello stesso con recensioni e fotografie.
ID	APPM_REQF05
Nome	Visualizzare lista di recensioni relative ad un particolare itinerario
Descrizione	Il sistema deve consentire ad un utente registrato di visualizzare tutte le recensioni di un particolare itinerario.
ID	APPM_REQF06
Nome	Inserire recensioni ad un particolare itinerario
Descrizione	Quando un utente registrato visualizza un itinerario, il sistema deve consentire a quest’ultimo di poterlo recensire specificando: <ul style="list-style-type: none"> • Titolo • Descrizione • Immagine • Valutazione
ID	APPM_REQF07
Nome	Mostrare la posizione corrente dell’utente sulla mappa
Descrizione	Il sistema deve consentire ad un utente registrato di poter inserire la propria posizione corrente, tramite una spunta nella schermata che mostra i dettagli di un itinerario.
ID	APPM_REQF08
Nome	Inserire tappe lungo un itinerario
Descrizione	Il sistema deve consentire ad un utente registrato di poter inserire lungo un percorso un punto di interesse indicandone: <ul style="list-style-type: none"> • Tipologia • Posizione.

ID	APPM_REQF09
Nome	Inserire itinerari
Descrizione	<p>Il sistema deve consentire ad un utente registrato di poter inserire un itinerario indicandone:</p> <ul style="list-style-type: none"> • Titolo itinerario • Durata • Livello di difficoltà • Descrizione • Tracciato del percorso.

ID	APPM_REQF10
Nome	Uscire dalla piattaforma
Descrizione	Il sistema deve consentire ad un utente registrato di poter uscire dalla piattaforma.

ID	APPM_REQF11
Nome	Visualizzazione profilo utente
Descrizione	<p>Il sistema deve consentire ad un utente registrato di visualizzare la schermata del proprio profilo contenente le proprie informazioni:</p> <ul style="list-style-type: none"> • E-mail • NickName • Data di nascita • Genere

ID	APPM_REQF12
Nome	Eliminazione dell'utente
Descrizione	Il sistema deve consentire ad un utente registrato di poter eliminare il proprio account.

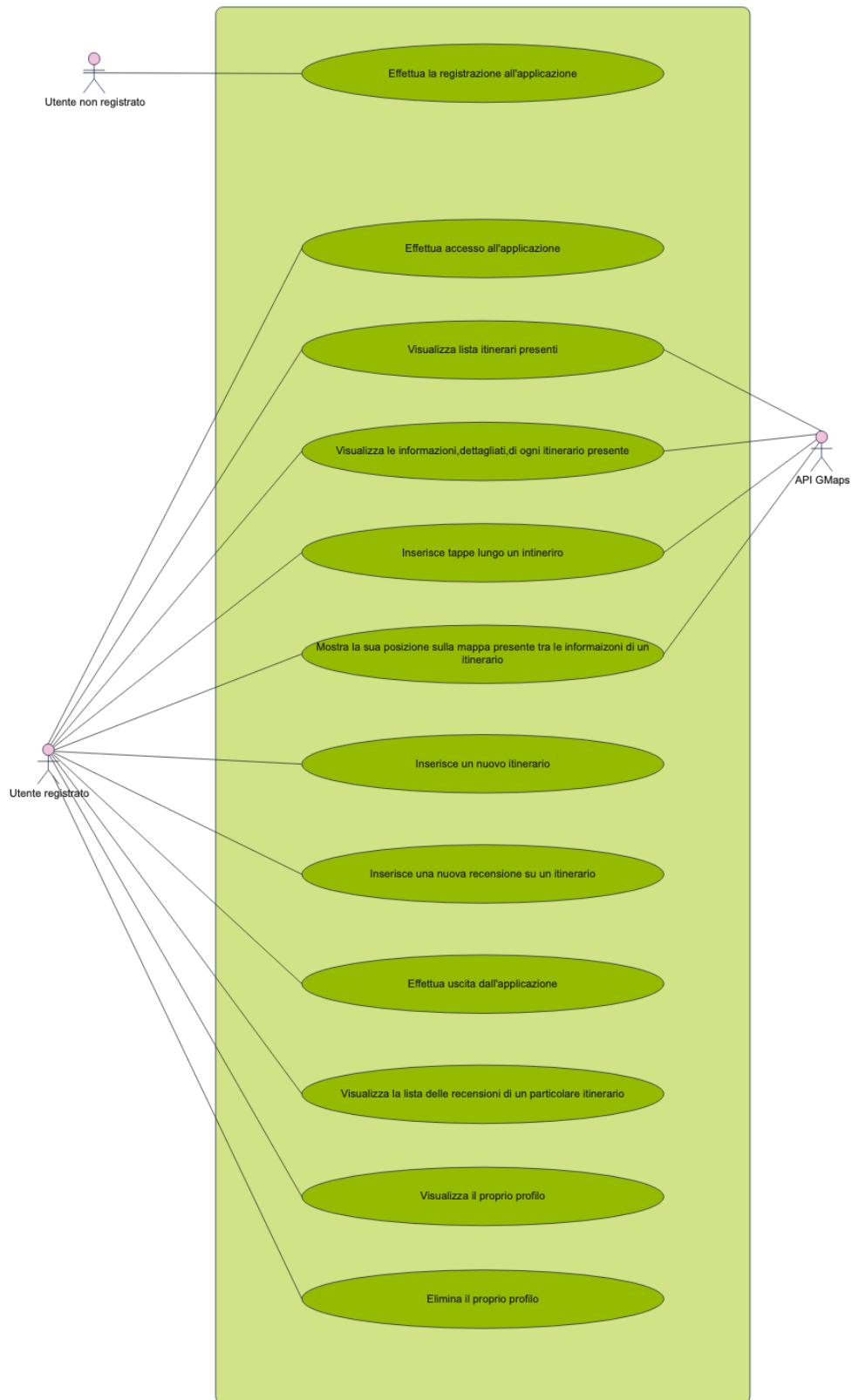
• Requisiti non Funzionali

Vengono ora elencati i requisiti non funzionali ovvero tutti i vincoli sui servizi offerti dal sistema.

ID	APPM_REQNF01
Nome	Usabilità dell'applicazione
Descrizione	Un utente deve riuscire ad utilizzare la piattaforma al massimo delle sue funzionalità, dopo una media di 2 ore di utilizzo.
ID	APPM_REQNF02
Nome	Policy password
Descrizione	Il sistema deve forzare l'utente ad inserire una password di almeno 8 caratteri contenente almeno una cifra numerica, almeno un carattere maiuscola ed almeno un carattere minuscolo.
ID	APPM_REQNF03
Nome	Policy email
Descrizione	Il sistema deve forzare l'utente ad inserire una email fatta nel seguente modo : <u>"nomeutente@dominio.it"</u> .

- Modellazione casi d'uso

Viene ora riportato lo Use Case Diagram dell'applicazione.



Use Case Diagram 1.1 - Use Case Diagram del sistema

- Tabella di Cockburn dei casi d'uso

Vengono ora elencate le tabelle di Cockburn relative a due casi d'uso, dello Use Case Diagram riportato nel paragrafo precedente.

- **Registrazione alla piattaforma**

USE CASE #1	Effettua registrazione		
Goal in Context	L'utente vuole effettuare la registrazione alla piattaforma		
Preconditions	Nessuna		
Success End Condition	L'utente riesce a registrarsi alla piattaforma		
Failed End Condition	L'utente non si registra alla piattaforma.		
Primary Actor	Utente non registrato		
Trigger	Pressione sul pulsante Sing Up nella schermata “APPM_MCK01-Partenza”		
DESCRIPTION	Step n°	Utente non registrato	Sistema
	1	Preme il pulsante “Sing Up” nella schermata “APPM_MCK01-Partenza”.	
	2		Mostra la schermata “APPM_MCK03-Registrazione”.
	3	Inserisce email, Nickname, password, data di nascita, genere e preme il pulsante “Sing Up”.	
	4		Mostra schermata “APPM_MCK04-Home”.
EXTENSION #1	Step n°	Utente non registrato	Sistema
L'utente non compila uno o più campi richiesti	3.1	Click sul pulsante “Registrati”	
	3.2		Mostra un messaggio di errore
	3.3		Torna allo step 3

Tabella di Cockburn : -Registrazione alla piattaforma - Inizio

EXTENSION #2	Step n°	Utente non registrato	Sistema
L'utente compila i campi con dati non validi.	3.1	Click sul pulsante “Registrati”	
	3.2		Mostra un messaggio di errore
	3.3		Torna allo step 3
EXTENSION #3	Step n°	Utente non registrato	Sistema
L'utente compila i campi con dati non validi: <ul style="list-style-type: none">• Formato email non valido;• password non valida;• Data di nascita non valida;• Genere non valido.	3.1	Click sul pulsante “Registrati”	
	3.2		Mostra un messaggio di errore
	3.3		Torna allo step 3

Tabella di Cockburn : -Registrazione alla piattaforma - Fine

- **Inserimento itinerario**

USE CASE #2	Inserimento un itinerario		
Goal in Context	L'utente vuole aggiungere un itinerario		
Preconditions	L'utente è autenticato		
Success End Condition	L'utente riesce ad aggiungere un itinerario		
Failed End Condition	L'utente non aggiunge la recensione. Chiude la schermata di inserimento itinerario		
Primary Actor	L'utente registrato		
Trigger	L'utente preme il pulsante “Pianifica” nella schermata “APPM_MCK04-Home”		
	Step n°	Utente registrato	Sistema
	1	Preme il pulsante “Pianifica” nella schermata “APPM_MCK04-Home”	
	2		Mostra la schermata “APPM_MCK06-Pianifica”
	3	Inserisce il titolo dell'itinerario, la descrizione, livello di difficoltà, durata e foto, il punto di partenza e di arrivo sulla mappa	
	4		Mostra la schermata “APPM_MCK05-Esplora”
EXTENSION #1 L'utente non compila uno o più campi richiesti	Step n°	Utente Registrato	Sistema
	3.1	Click sul pulsante “conferma”	
	3.2		Mostra un messaggio di errore
	3.3		Torno allo step 3
EXTENSION #2 L'utente compila i campi con dati non validi	Step n°	Utente Registrato	Sistema
	3.1	Click sul pulsante “conferma”	
	3.2		Mostra un messaggio di errore
	3.3		Torno allo step 3

EXTENSION #3	Step n°	Utente Registrato	Sistema
L'utente compila i campi con dati non validi: <ul style="list-style-type: none">• Titolo itinerario non compreso tra 30 e 100;• Descrizione itinerario non compresa tra 150 e 1000;• Recensione è più lunga di 150 caratteri;• Ora maggiore di 99;• Minuti maggiori di 59;• Secondi maggiori di 59;• Livello di difficoltà non è tra difficile, medio e semplice.	3.1 3.2 3.3	Click sul pulsante “conferma” Mostra un messaggio di errore Torno allo step 3	
EXTENSION #5	Step n°	Utente Registrato	Sistema
L'utente clicca sul pulsante annulla	3.1 3.2	Click sul pulsante “annulla” Mostra la schermata “APPM_MCK06-Pianifica” ripulita	
EXTENSION #6	Step n°	Utente Registrato	Sistema
L'utente clicca sul pulsante per tornare indietro	3.1 3.2	Click sul pulsante per tornare indietro Mostra la schermata “APPM_MCK04-Home”	

Tabella di Cockburn : -Inserimento itinerario alla piattaforma - Fine

- Mock-up dell'applicazione mobile

Vengono ora mostrati due mock-up dell'applicazione mobile.

Ogni mock-up esplicerà il tipo e la funzionalità di ogni componente.

- **Registrazione alla piattaforma**

APPM_MCK03-Registrazione	
	Componente [1]
	<p>Tipo Pulsante</p> <p>Funzionalità Torna alla schermata “APPM_MCK01-Partenza”</p>
<p>1</p> <p>E-mail</p> <p>Nickname</p> <p>*****</p> <p>Data compleanno</p> <p>Genere</p> <p>SING-UP</p>	Componente [2]
	<p>Tipo Campo testuale</p> <p>Funzionalità Campo testuale per l'inserimento dell'email</p>
<p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>6</p> <p>7</p>	Componente [3]
	<p>Tipo Campo testuale</p> <p>Funzionalità Campo testuale per l'inserimento del nickname</p>
<p>8</p> <p>9</p> <p>10</p> <p>11</p>	Componente [4]
	<p>Tipo Campo testuale</p> <p>Funzionalità Campo testuale per l'inserimento della password</p>
<p>12</p> <p>13</p> <p>14</p>	Componente [5]
	<p>Tipo Campo testuale</p> <p>Funzionalità Campo testuale per l'inserimento della data di nascita</p>
<p>15</p> <p>16</p>	Componente [6]
	<p>Tipo Campo testuale</p> <p>Funzionalità Campo testuale per l'inserimento del genere</p>
<p>17</p> <p>18</p>	Componente [7]
	<p>Tipo Pulsante</p> <p>Funzionalità Apre la schermata “APPM_MCK04-Home”</p>

- Inserimento itinerario

APPM_MCK06.1-Aggiungi itinerario	
	Componente [1]
Tipo	Pulsante
Funzionalità	Torna alla schermata "APPM_MCK04-Home"
	Componente [2]
Tipo	Campo testuale
Funzionalità	Campo testuale per l'inserimento del titolo
	Componente [3]
Tipo	Campo testuale
Funzionalità	Campo testuale per l'inserimento della descrizione dell'itinerario
	Componente [4]
Tipo	Campo testuale
Funzionalità	Campo testuale per l'inserimento del livello di difficoltà
	Componente [5]
Tipo	Campo testuale
Funzionalità	Campo testuale per l'inserimento dell'ora
	Componente [6]
Tipo	Campo testuale
Funzionalità	Campo testuale per l'inserimento dei minuti
	Componente [7]
Tipo	Campo testuale
Funzionalità	Campo testuale per l'inserimento dei secondi
	Componente [8]
Tipo	Pulsante
Funzionalità	Apre la galleria del dispositivo

APPM_MCK06.1 - Aggiungi itinerario	
Componente [9]	
Tipo	Mappa
Funzionalità	Consente all'utente di selezionare il punto di partenza e arrivo dell'itinerario
Componente [10]	
Tipo	Pulsante
Funzionalità	Apre la schermata "APPM_MCK06-Pianifica"
Componente [11]	
Tipo	Pulsante
Funzionalità	Apre la schermata "APPM_MCK05-Esplora"
Componente [12]	
Tipo	Pulsante
Funzionalità	Apre la cartella "documenti" del dispositivo

Mock-up : Schermata inserimento itinerario - Fine

- Presentazione idea progettuale

NaTour22 è un'applicazione mobile di navigazione Outdoor dedicata ad appassionati di escursionismo e attività all'aperto.

Il punto di forza dell'applicazione risiede nella sua facilità d'utilizzo. Già la prima versione consente di sfruttare numerose funzionalità anche se ancora disponibile solo su Android, si prevede, in futuro, lo sviluppo anche per altri sistemi.

Analizziamo le funzionalità offerte dall'applicazione :

- **Un utente può registrarsi e accedere** - L'utente può registrarsi fornendo un nickname, un'email, e una password, una data di nascita e un genere e accedere con email e password.
- **Un utente che ha effettuato può inserire nuovi itinerari** - L'utente può inserire un nuovo itinerario inserendo un titolo, una descrizione, la durata che si articola in ore, minuti e secondi, e il livello di difficoltà che si articola in semplice, medio e difficile. Inoltre si è scelto di permettere all'utente di inserire il tracciato geografico mediante mappa interattiva oppure importando un file GPX con le opportune coordinate.
- **Un utente può inserire punti d'interesse** - L'utente può inserire punti d'interesse sulla mappa di un particolare itinerario ciò viene fatto inserendo la tipologia del punto d'interesse e le coordinate geografiche in cui si vuole posizionare il punto d'interesse.
- **Un utente può visualizzare per ogni itinerario tutte le informazioni in modo dettagliato** - L'utente può visualizzare una schermata per ogni itinerario che mostra tutte le informazioni dell'itinerario come: il titolo, la descrizione, il livello di difficoltà, la durata, una mappa che indica il percorso dell'itinerario dal punto partenza al punto di arrivo , le recensioni e le relative foto.
- **Un utente può mostrare anche la propria posizione sulla mappa** - L'utente può mostrare la sua posizione corrente sulla mappa, nella schermata che contiene tutte le informazioni di un itinerario se il dispositivo mobile supporta localizzazione GPS.

Per rendere NaTour22 più competitiva in questo settore, si è previsto di aggiungere nuove funzionalità:

- **Un utente può visualizzare la lista degli itinerari presenti nella piattaforma** - L'utente può visualizzare tutti gli itinerari creati da se e da altri utenti.
- **Un utente può valutare un itinerario** - L'utente può recensire un itinerario indicando titolo, descrizione, valutazione e foto.
- **Un utente può visualizzare il proprio profilo** - L'utente può visualizzare le proprie informazioni accedendo al profilo personale.

- Individuazione target utenti

Il termine *target*, in questo contesto, rappresenta un gruppo di individui ai quali si vuole vendere i propri prodotti o servizi.

L'analisi dei target è stata fatta considerando due fasi:

1. Segmentazione del mercato: si è diviso il mercato in vari segmenti in base a caratteristiche chiavi come:

- dati demografici in cui si intende: età, genere, reddito, stato civile, occupazione/settore e livello di istruzione;
- geolocalizzazione in cui si intende restringere l'insieme dei target in base a determinate aree geografiche
- Interessi e opinioni in cui si intendono informazioni sugli interessi, sul tempo libero, sport, hobby oppure informazioni su opinioni riguardo la sostenibilità e la tutela ambientale

2. Identificazione dei target group : si è diviso un insieme di persone con caratteristiche simili.

Per **identificare il target dei clienti** e dare il via alla ricerca di mercato, è necessario rispondere alle seguenti domande:

- Qual è l'immagine del cliente tipo?
- A quali problemi/bisogni del cliente si vuole rispondere con i propri prodotti/servizi?
- Quali clienti otterranno benefici dai prodotti che si vogliono realizzare?
- Ci sono nicchie di mercato a cui poter far riferimento?
- Chi sono i competitor?

Sulla base di ciò si è concluso che il target di persone su cui concentrale NaTour22 è molto ampio in quanto comprende tutte quelle persone che amano fare escursioni, attività all'aperto che vogliono immergersi per la prima volta nella natura.

Vengono ora indicati alcuni valutatori presi come campione, per valutare il prodotto una volta terminato :



Elvira Giretti

Biografia
Elvira è una donna pratica ed energetica, è in pensione e vive da sola nella sua casa di famiglia. Con l'avanzare dell'età, gode di uno stile di vita attivo facendo volontariato, prendendosi cura dei suoi riputi. Assume quotidianamente farmaci per controllare: diabete, colesterolo e osteoporosi. Visita il suo medico e mantiene una dieta alimentare sana ma ritiene che la sua vera medicina è il contatto con la natura.

Hobby e interessi
Ama fare ginnastica all'aria aperta e fare lunghe passeggiate insieme ad amici e parenti.

Informazioni Generali

Sesso	Maschio
Età	60
Locazione	Italia

"La natura è la mia medicina per sopravvivere."

Obiettivi

- Ricercare nuove destinazioni e itinerari/sentieri e farli conoscere ad amici e parenti.
- Tenersi in forma.



Antonio Laccio

Biografia
Antonio ex professore di lettere, dopo tanti anni di lavoro ora si gode la pensione riempiendo il suo tempo con passeggiate ed escursioni.

Hobby e interessi
Ama osservare la città alle prime ore del mattino e passeggiare tra la natura.

Informazioni Generali

Sesso	Maschio
Età	78
Locazione	Italia

Obiettivi

- Scoprire nuovi itinerari e sentieri.
- Mantenersi in salute.



Alice Bassi

Biografia

Alice è una studentessa universitaria di 23 anni ed un'escursionista per hobby. È cresciuta facendo escursioni nell'area intorno alla sua città, ad Alice piace fare escursioni ogni fine settimana, purché il tempo sia bello e spesso porta i suoi amici insieme a lei. Documenta le sue avventure sui social e spesso si avventura fuori pista per ottenere foto migliori.

Hobby e interessi

Ama fare escursioni e stare all'aria aperta quando ne ha l'occasione, passare il tempo nei suoi spazi preferiti con i suoi amici e la sua famiglia o scoprire nuovi paesaggi e le piace fotografarli.

Obiettivi

È in cerca di nuovi itinerari da mostrare ad amici e familiari e fotografarli per documentarli sui social.

Informazioni Generali

Sesso	Femmina
Età	25
Locazione	Italia

"L'escursionismo è il modo per schiarirmi le idee"



Marco Rossi

Biografia

Marco è uno sviluppatore software a tempo pieno che trascorre molto tempo al chiuso. Non ha molta esperienza nel fare qualsiasi cosa all'aperto tuttavia ha recentemente ha cercato di farlo, rimettersi in forma e fare più attività all'area aperta. È sempre in contatto con gli amici dell'università ed è ansioso di esplorare il mondo con loro.

Hobby e interessi

Non ama fare palestra o qualsiasi tipo di sport ma ama passare le giornate al computer, ma si sta adoperando per passare più tempo all'aperto.

Informazioni Generali

Sesso	Maschio
Età	37
Locazione	Italia

"Voglio iniziare ad esplorare la terra e intraprendere nuove avventure."

Obiettivi

- Trovare il tempo per condurre una vita più sana e provare nuove attività all'aperto come l'escursionismo.
- Trovare un modo per essere in grado di bilanciare stili di vita indoor e outdoor.
- Essere in grado di imparare ed esplorare di più la natura.

- Valutazione dell'usabilità

Per effettuare una valutazione completa dell'usabilità del prodotto software, si sono seguite due fasi complementari e non sostituibili l'una con l'altra:

2. Una prima fase , detta di *valutazione euristica* o di *valutazione dell'usabilità a-priori* ,in cui verranno eseguite valutazioni da parte di esperti di usabilità, appartenenti al team di sviluppo del prodotto software, senza alcun coinvolgimento dei clienti e degli stakeholders. Tale fase si effettua durante le fasi di progettazione dell'interfaccia utente nell'ambiente di sviluppo, quando ancora non si ha a disposizione il prodotto finito, ma solo dei prototipi.
3. Una seconda fase , detta *test dell'usabilità* o di *valutazione dell'usabilità sul campo* , in cui verranno eseguite valutazioni da parte di un campione di utenti, rappresentativo del target identificato al punto precedente, in un ambiente controllato da uno o più osservatori che sono esperti di usabilità.

In particolare, è possibile notare dalle seguenti tabelle quali sono le funzionalità implementate nei prototipi e a quale livello di dettaglio.

		Funzioni Implementate					
		Effettua registrazione	Effettua accesso	Visualizza lista itinerari	Visualizza informazioni dettagliate dell'itinerario	Inserimento punti d'interesse sulla mappa	Inserimento nuovo itinerario
Livello di dettaglio	MockUp	X	X	X	X	X	X
	Prototipo interattivo su Axure	X	X	X	X	X	X
	Statecharts	X					X
	Prima implementazione	X	X	X	X		X
	Beta testing	X	X	X	X		X
	Applicazione finale	X	X	X	X		X

		Funzioni Implementate				
		Inserimento posizione corrente dell'utente sulla mappa	Inserimento nuova recensione	Effettua logOut	Visualizza profilo	Visualizza lista delle recensioni
Livello di dettaglio	MockUp	X	X	X	X	X
	Prototipo interattivo su Axure	X	X	X	X	X
	Statecharts					
	Prima implementazione		X	X	X	X
	Beta testing		X	X	X	X
	Applicazione finale	X	X	X	X	X

Gli esperti dell'usabilità, valuteranno i prototipi forniti in fase di progettazione analizzando il comportamento complessivo del sistema e verificando la loro conformità ad alcune regole, come le [8 regole d'oro di Ben Shneiderman](#) o le [10 euristiche di Nielsen](#).

La valutazione dell'usabilità *a-priori* termina con la realizzazione di un diagramma che rappresenta una possibile risposta degli esperti di usabilità (sono a valutati solo i due casi d'uso significativi identificati via statecharts). In cui si indica:

- Sull'asse verticale i valutatori;
- Sull'asse orizzontale i problemi di usabilità riscontrati durante la valutazione
- L'intersezione tra un punto dell'asse orizzontale e dell'asse verticale, rappresenterà i diversi problemi riscontrato dai valutatori.

X					X	Esperto 4
	X	X	X	X		Esperto 3
	X				X	Esperto 2
			X	X		Esperto 1
L'utente non inserisce tutti i campi	L'utente inserisce un nickName con più di 15 caratteri	L'utente inserisce un email non valida	L'utente inserisce una Password con più di 8 caratteri	L'utente inserisce una password non valida		
PROBLEMI DI USABILITÀ						

Tabella 1 - Risultati della valutazione del prototipo di registrazione

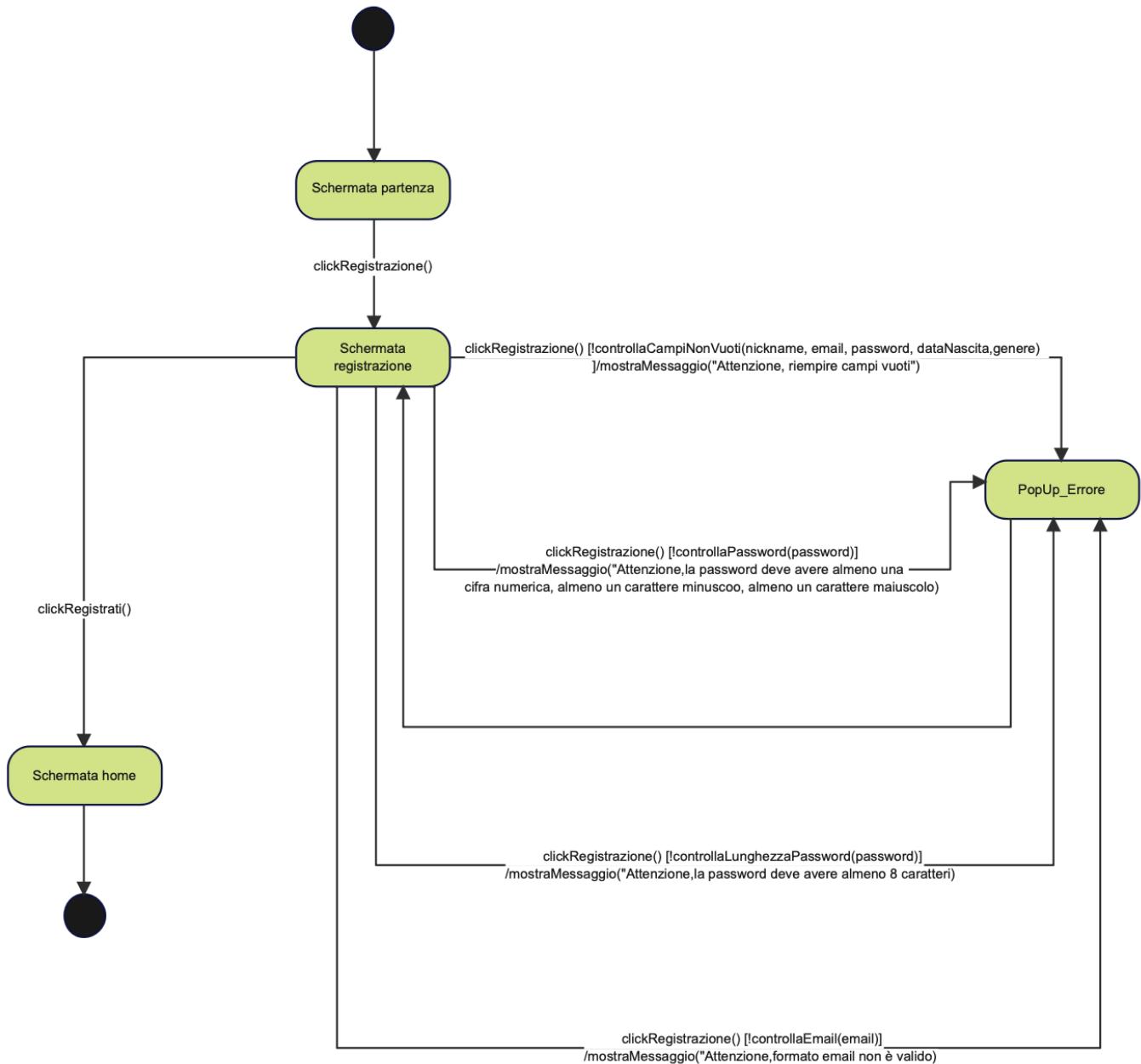
X		X	X	X			Esperto 4
	X						Esperto 3
X	X			X			Esperto 2
		X			X		Esperto 1
L'utente inserisce file non valido	L'utente non seleziona punto di arrivo e partenza sulle mappe	L'utente non inserisce tutti i campi	L'utente inserisce un titolo itinerario non valido	L'utente inserisce una descrizione troppo lunga	L'utente inserisce un'immagine non valida	L'utente inserisce una durata itinerario non valida	
PROBLEMI DI USABILITÀ							

Tabella 2 - Risultati della valutazione del prototipo di inserimento itinerario

- Prototipazione funzionale via statechart

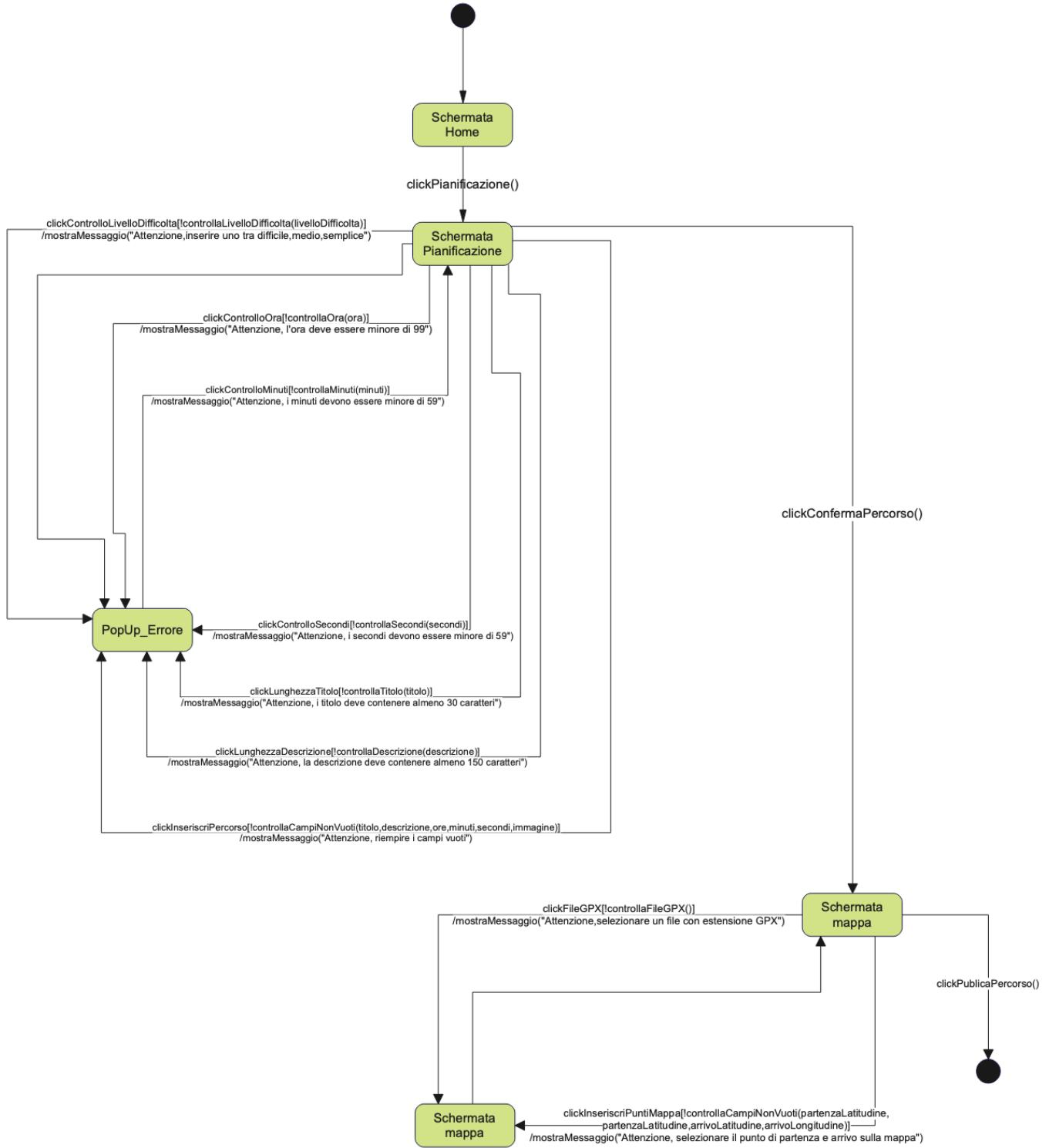
Vengono ora mostrati le prototipazione funzionali via statechar di due mock-up dell'applicazione mobile visti in precedenza.

- **Registrazione alla piattaforma**



statechart 01 - registrazione alla piattaforma

• Inserimento itinerario



statechart 02 - inserimento di un itinerario

- Glossario

Termino	Descrizione
Itinerario	Si intende un percorso diviso in più tappe.
Titolo dell'itinerario	Si intende il titolo di un itinerario
Durata	Si intende la durata, espressa in ore; minuti; secondi, dello svolgimento di un itinerario
Livello di difficoltà	Si intende la valutazione della difficoltà, cioè la scala di difficoltà tecnica indicata nelle escursioni, che prevede tre livelli di difficoltà: facile (per tutti), medio, difficile (per escursionisti esperti).
Punto di inizio	Si intende la località di partenza di un itinerario
Punto di fine	Si intende la località di arrivo di un itinerario
Tracciato	Indica l'intero percorso dal punto di partenza al punto d'arrivo
File formato GPX	È un formato che permette la memorizzazione e l'elaborazione di dati GPS
Tipologia	Si intende il tipo di punto d'interesse che l'utente desidera inserire sulla mappa di un itinerario
Posizione geografica	Si intende l'identificazione di un luogo specifico sul pianeta, indicato tramite due coordinate che sono x (ascissa) e y (ordinata).
Android Studio	È il tool ufficiale del sistema operativo Android per la realizzazione di nuove applicazioni.
Object Oriented	È l'abbreviazione di Object-Oriented Programming, programmazione orientata agli oggetti. L'Object-Oriented Programming è quindi un paradigma di programmazione basato sul concetto di oggetti, specifiche strutture di dati all'interno di una classe.
Java	È un linguaggio di programmazione sviluppato da Sun Microsystems nel 1995. Java è un linguaggio di programmazione utilizzato per lo sviluppo web, in particolare per le applicazioni web client-server.

Termino	Descrizione
Claud Computing	Indica un paradigma di erogazione di servizi offerti su richiesta da un fornitore a un cliente finale attraverso la rete internet, a partire da un insieme di risorse preesistenti, configurabili e disponibili in remoto sotto forma di architettura distribuita.
Scalabilità del sistema	Indica la capacità del sistema di aumentare o diminuire le funzionalità/prestazioni su una scala predefinita
Requisiti funzionali	Elenchi di funzionalità o servizi che il sistema deve fornire. Essi descrivono anche il comportamento del sistema a fronte di particolari input e come esso dovrebbe reagire in determinate situazioni.
Requisiti non funzionali	Rappresentano i vincoli e le proprietà/ caratteristiche relative ad sistema, come vincoli di natura temporale, vincoli sul processo di sviluppo e sugli standard da adottare.
Email	Si intende l'indirizzo di posta elettronica che l'utente inserisce in fase di registrazione.
NickName	Si intende un soprannome che l'utente non registrato associa al proprio profilo in fase di registrazione alla piattaforma NaTour22
Password	Si intende una sequenza di caratteri alfanumerici che l'utente non registrato associa al proprio profilo in fase di registrazione alla piattaforma NaTour22
Use Case Diagram	Diagramma dedicato alla descrizione delle funzioni o servizi offerti da un sistema
Mock-up	Permette di rappresentare l'idea del progetto finale ma senza l'interattività di un prototipo, rappresentando nel dettaglio i vari contenuti e le funzionalità base dell'applicazione web in maniera statica.
Prototipizzazione funzionale	Descrivere il comportamento di entità o di classi in termini di stato.
Usabilità	L'usabilità misura il grado di facilità e soddisfazione con cui gli utenti si relazionano con l'interfaccia di un sito o app

Capitolo III – Modello di Dominio

O Documento dei Requisiti Software

- Modello di Dominio: Applicazione Mobile

- Diagrammi delle classi di analisi

Vengono ora elencati i Class Diagram relativi alle funzionalità che l'applicazione mobile deve offrire.

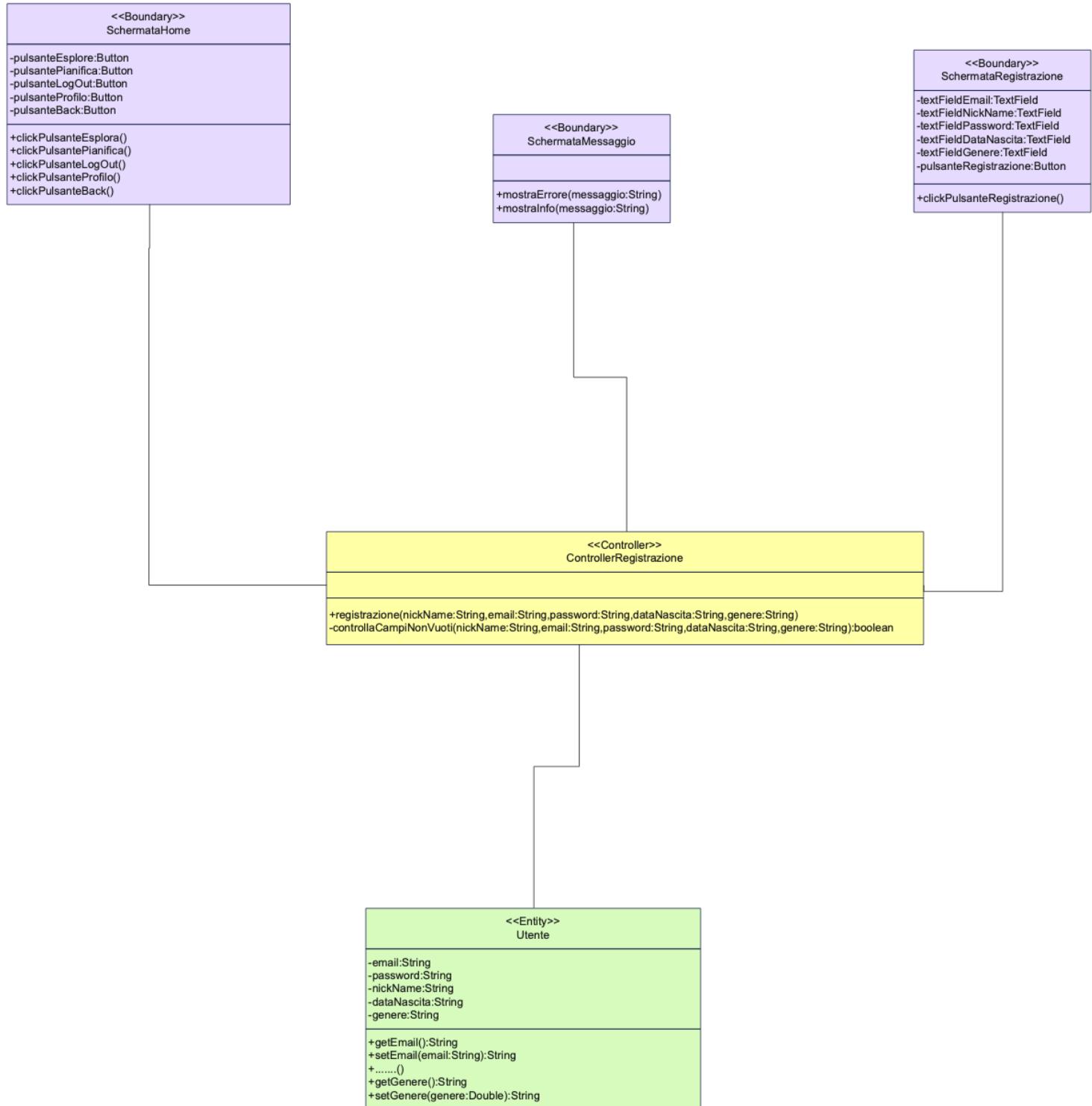
L'individuazione degli oggetti partecipanti è guidata dall'euristica Three - Object- Type, gli oggetti vengono dunque classificati in:

- Entity – Modellano l'informazione persistente.
- Boundary – Modellano le interazioni tra gli attori e il sistema.
- Control – Modellano la logica necessaria a svolgere lo use case.

Al fine di aumentare la leggibilità dei Class Diagram entity, boundary e control sono utilizzati colori differenti:

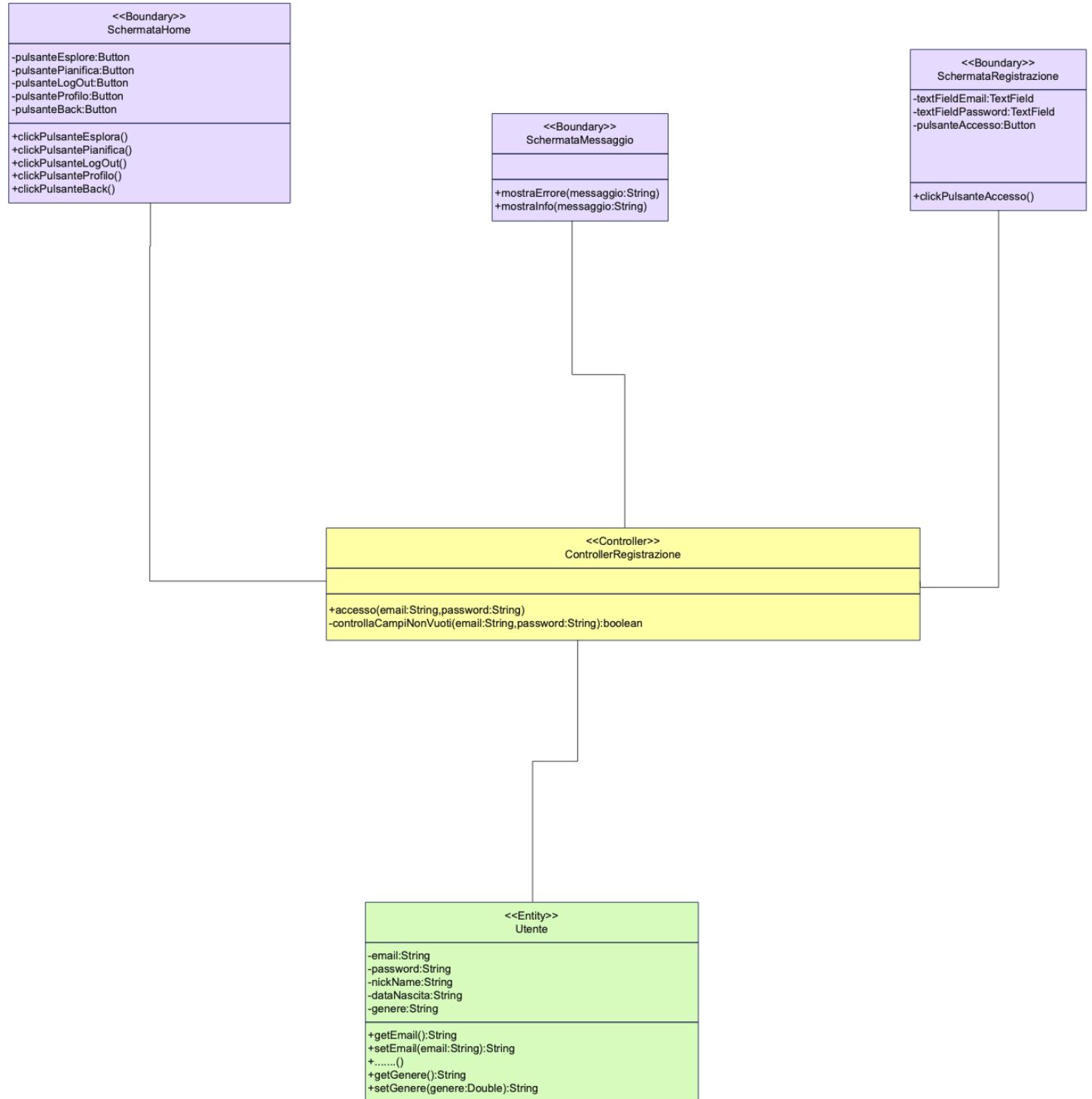
- Le classi entity sono colorate in verde
- Le classi boundary sono colorate in viola
- Le classi I sono colorate in giallo

• Registrazione alla piattaforma



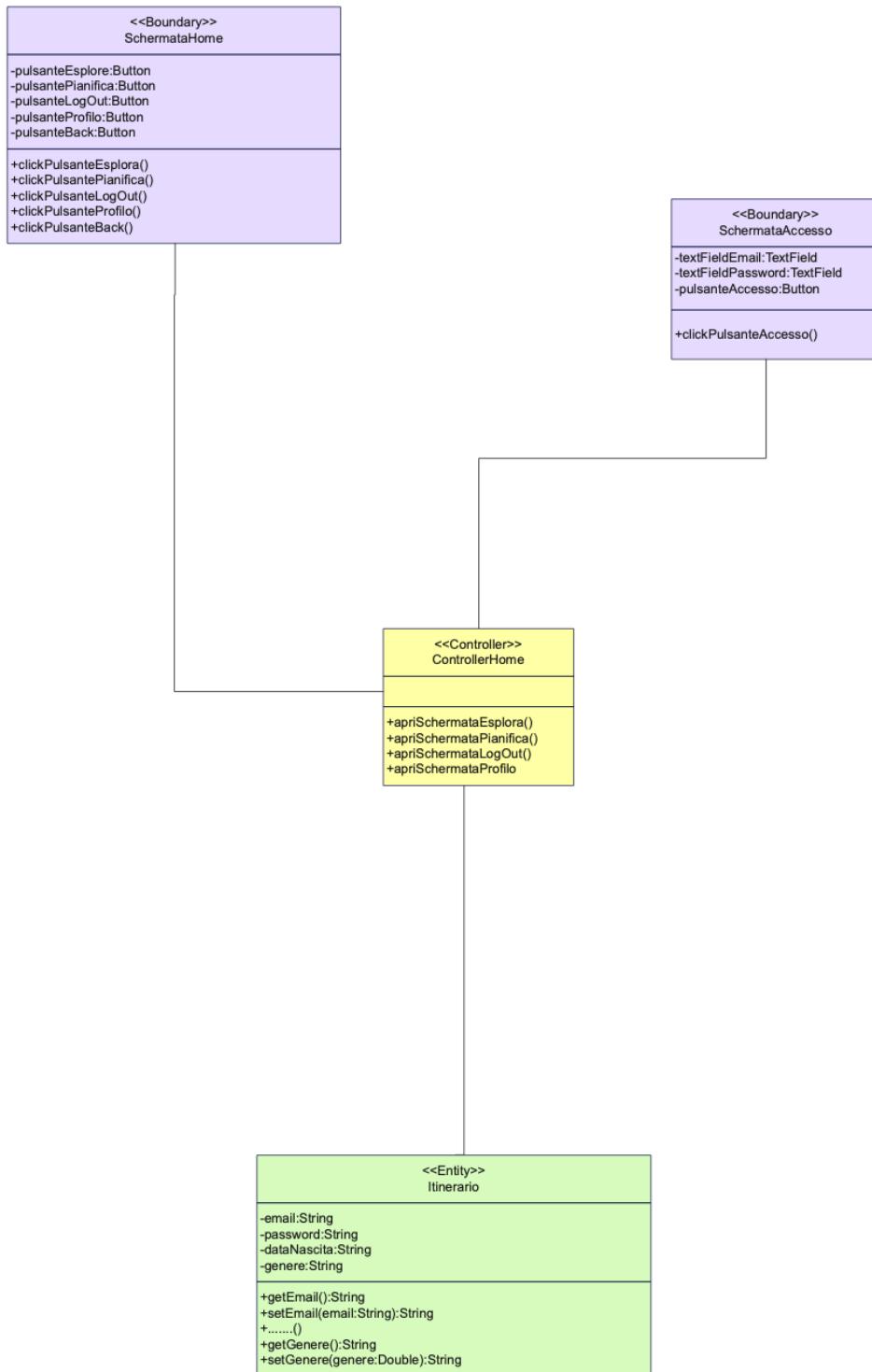
Class Diagram 01 - schermata registrazione alla piattaforma

- Accedi alla piattaforma



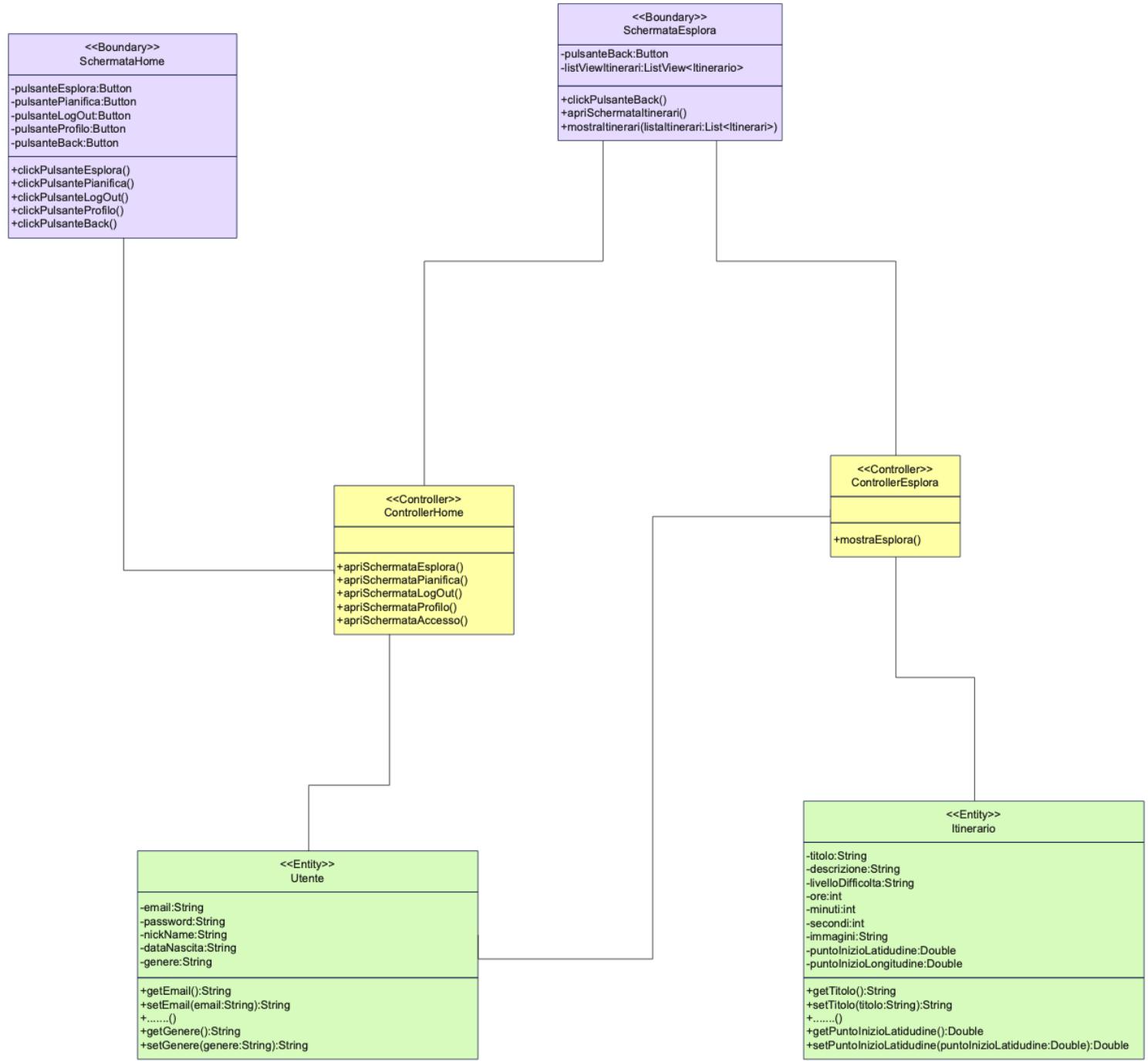
Class Diagram 02 - schermata accesso alla piattaforma

- **Uscita dalla piattaforma**



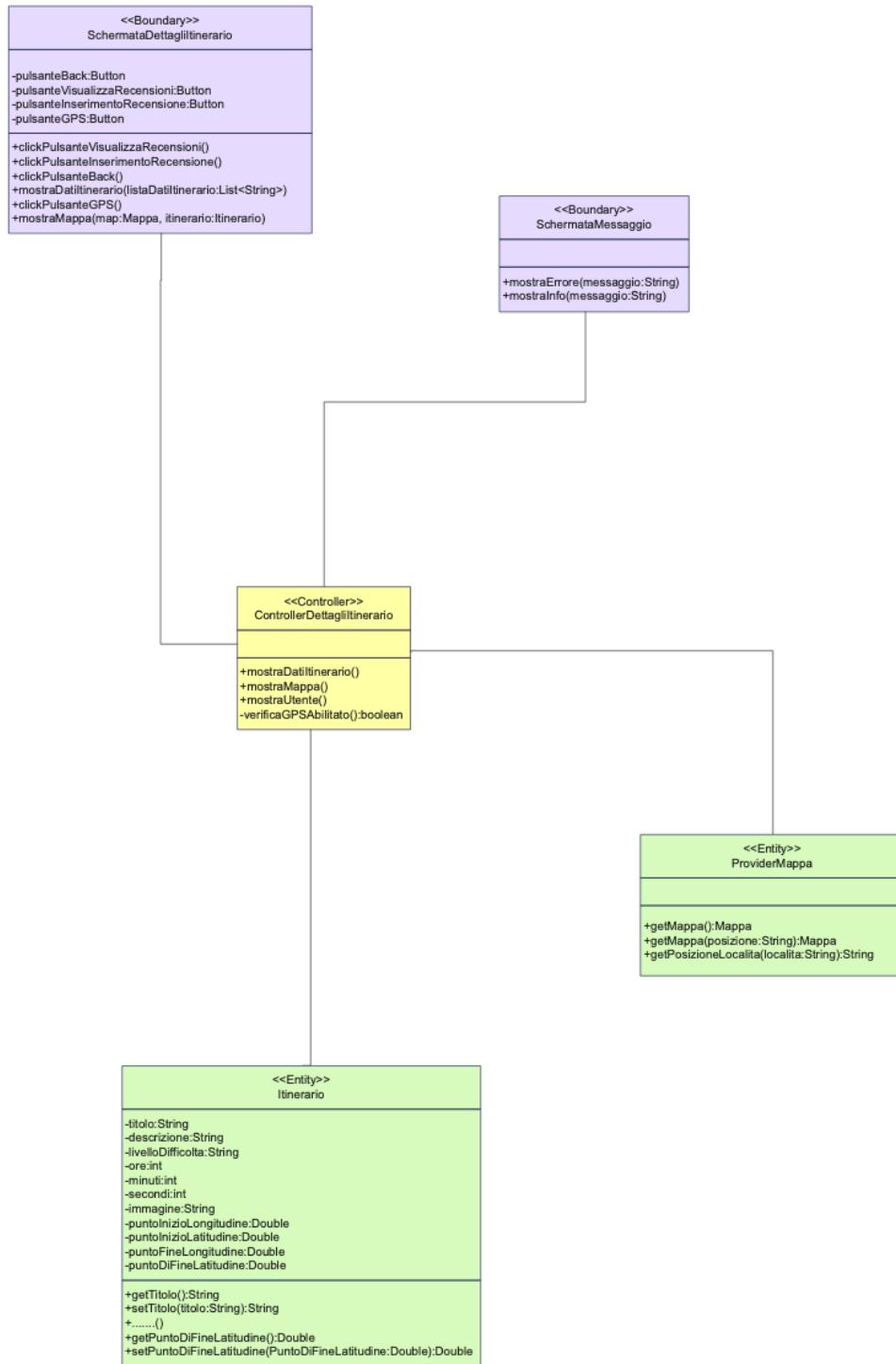
Class Diagram 03 - schermata uscita dalla piattaforma

• Visualizza lista itinerari



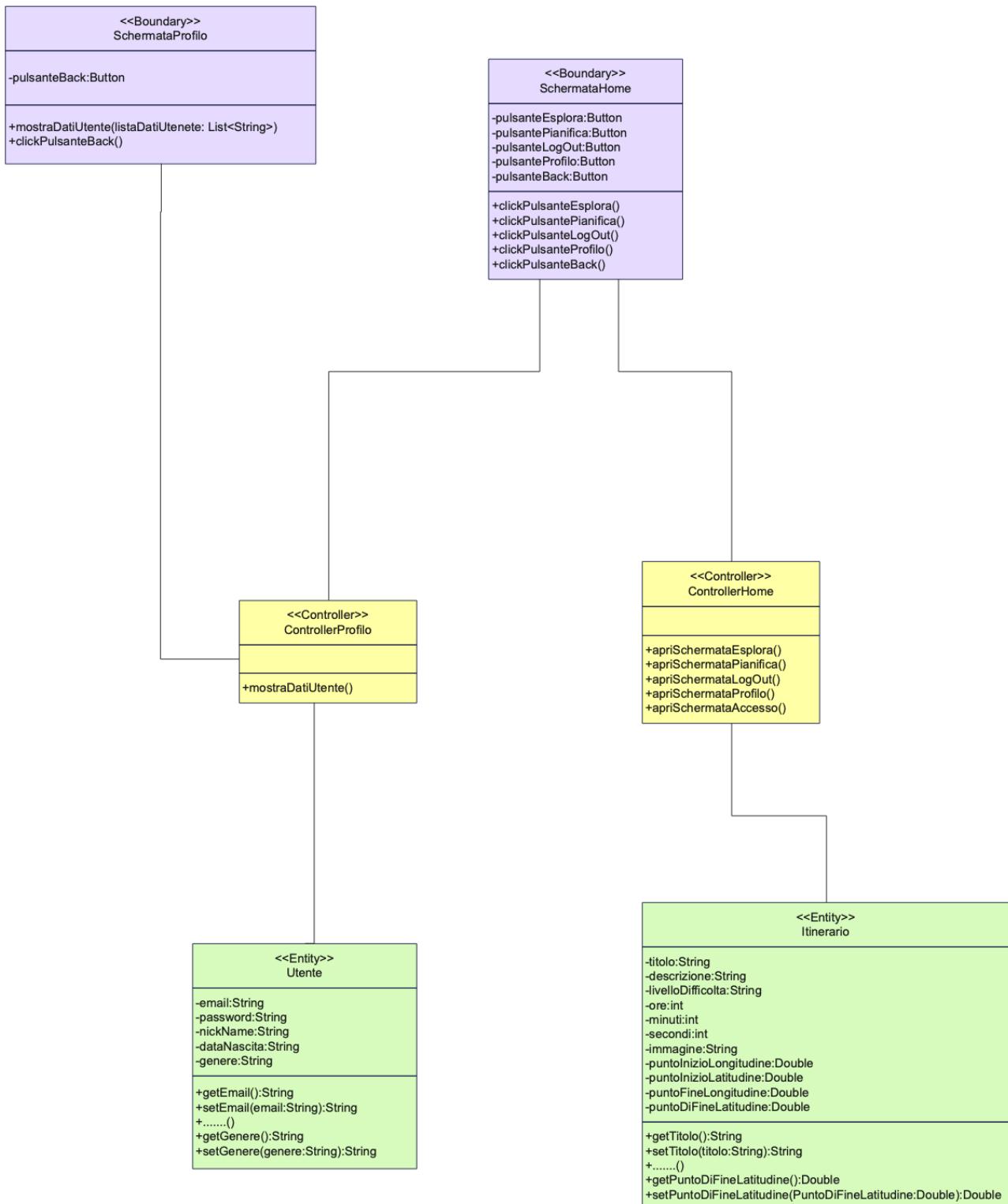
Class Diagram 04 : schermata visualizza lista itinerari

- Visualizza informazioni itinerario più posizione utente su mappa



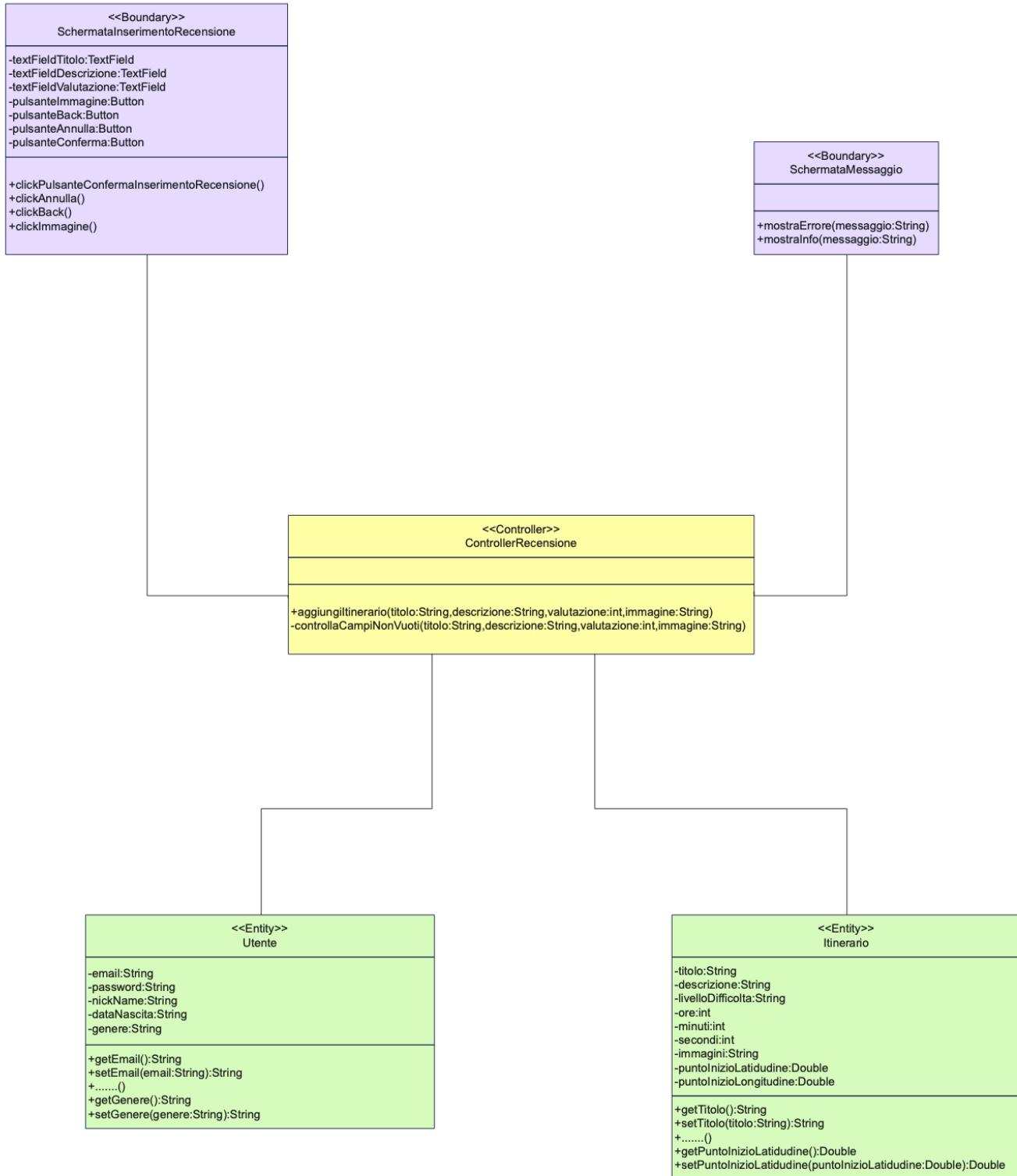
Class Diagram 05 - schermata visualizza informazioni itinerario più posizione corrente sulla mappa dell'utente

- **Visualizza profilo**



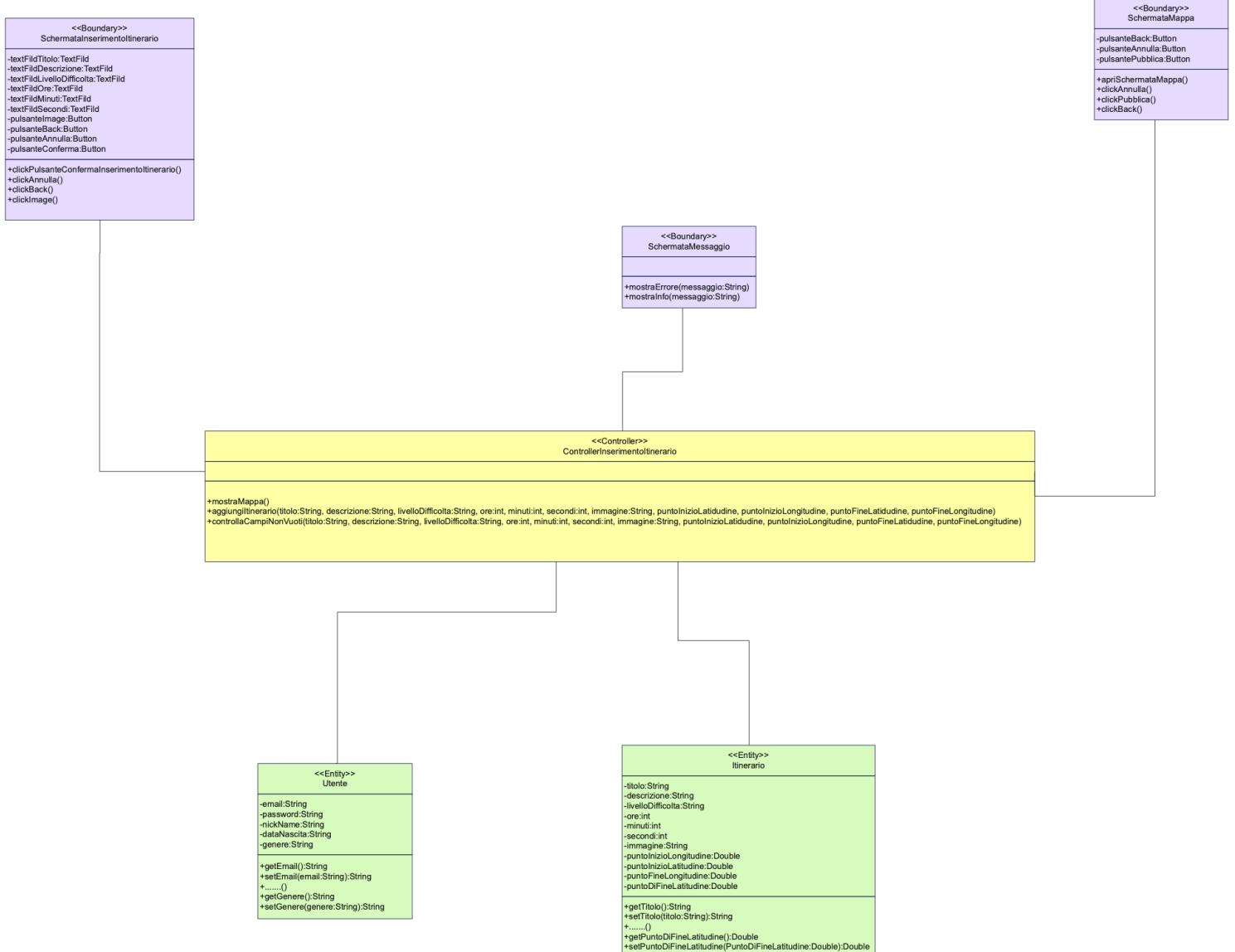
Class Diagram 06 - schermata visualizza profilo

• Inserimento recensioni



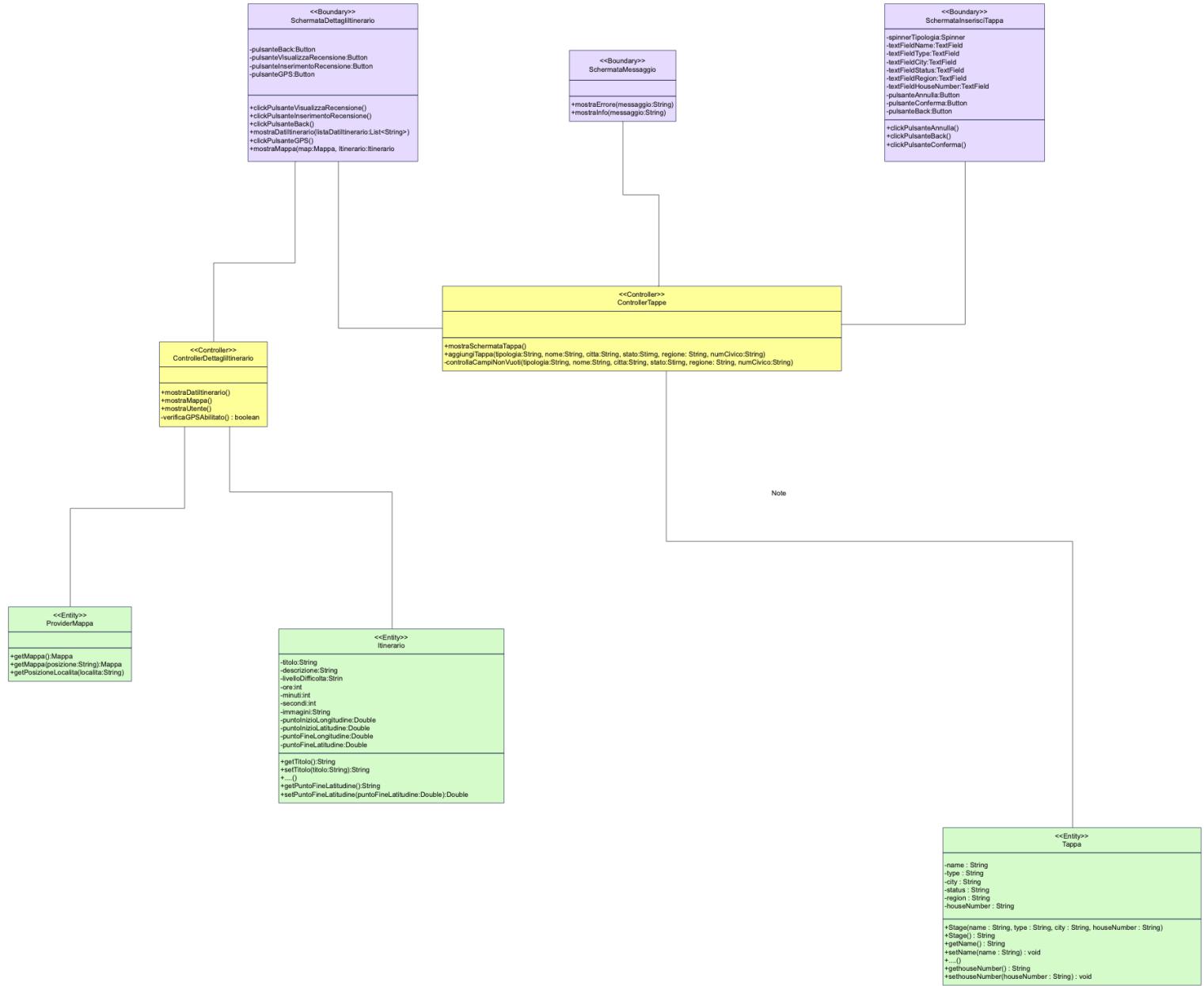
Class Diagram 07 - schermata inserimento recensioni

• Inserimento itinerario



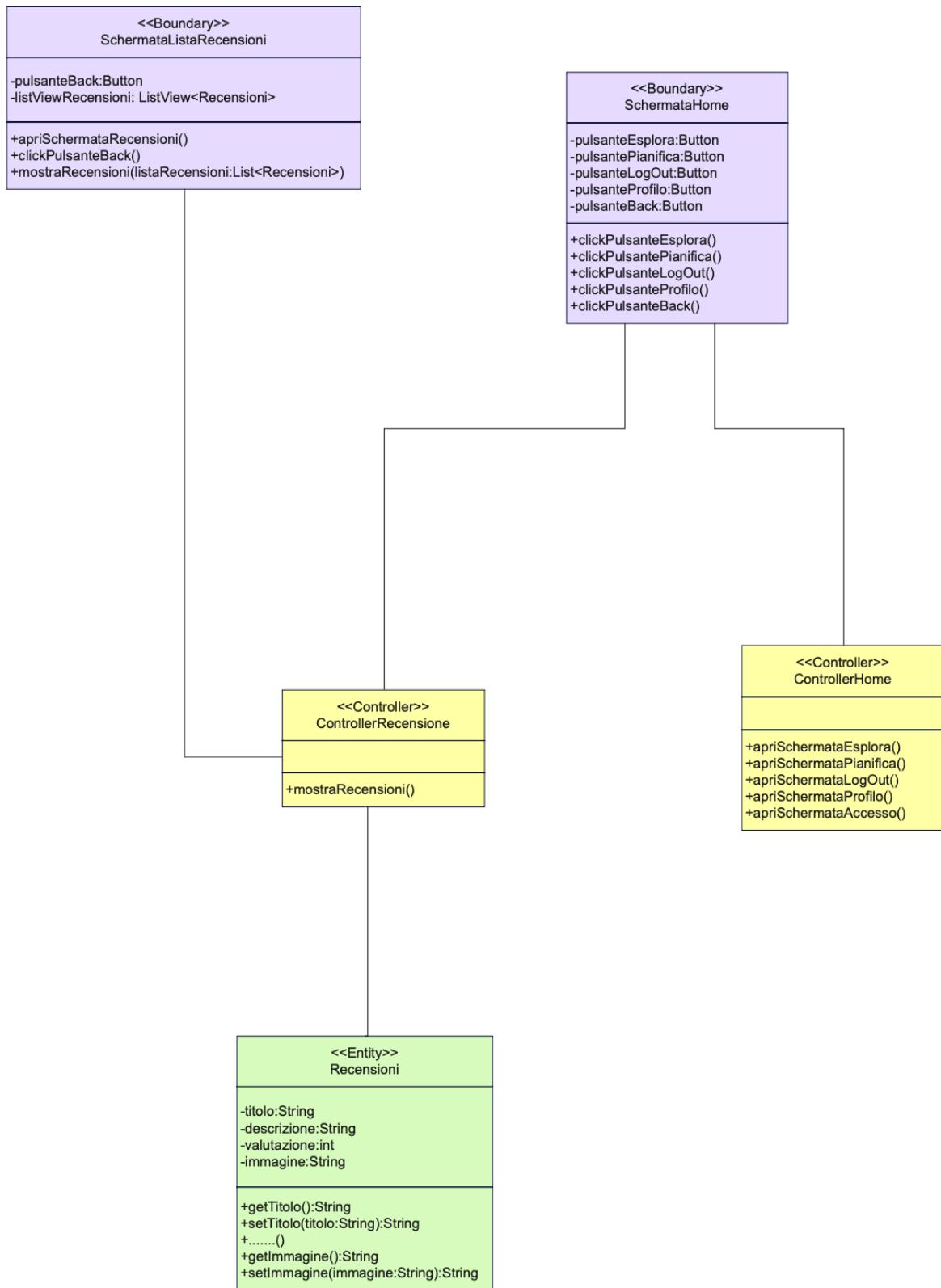
Class Diagram 08 - schermata inserimento itinerario

• Inserimento tappa lungo un itinerario



Class Diagram 09 - schermata inserimento punti d'interesse sulla mappa

- **Visualizza lista recensioni**

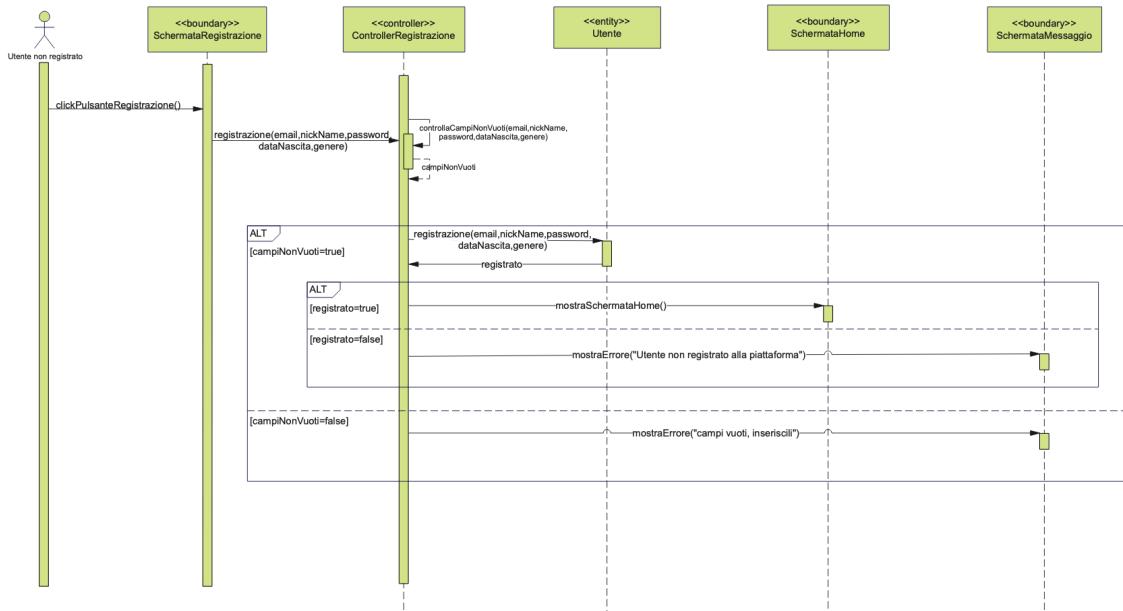


Class Diagram 10 - schermata lista recensioni

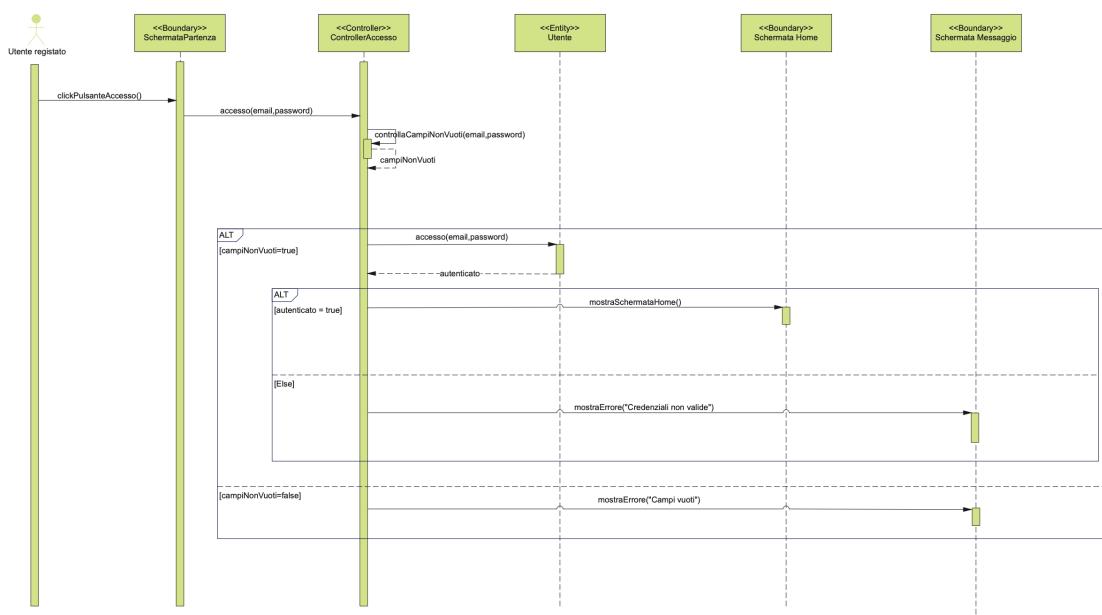
- Diagrammi di sequenza di analisi

Successivamente sono riportati i Sequence Diagram relativi a due funzionalità che l'applicazione mobile deve offrire.

• Registrazione alla piattaforma



• Accesso alla piattaforma



Sequence Diagram 02 - Schermata di accesso alla piattaforma

- Diagrammi di attività

Sono riportati i diagrammi di attività relativi alle funzionalità che l'applicazione Mobile deve offrire.

• Registrazione alla piattaforma

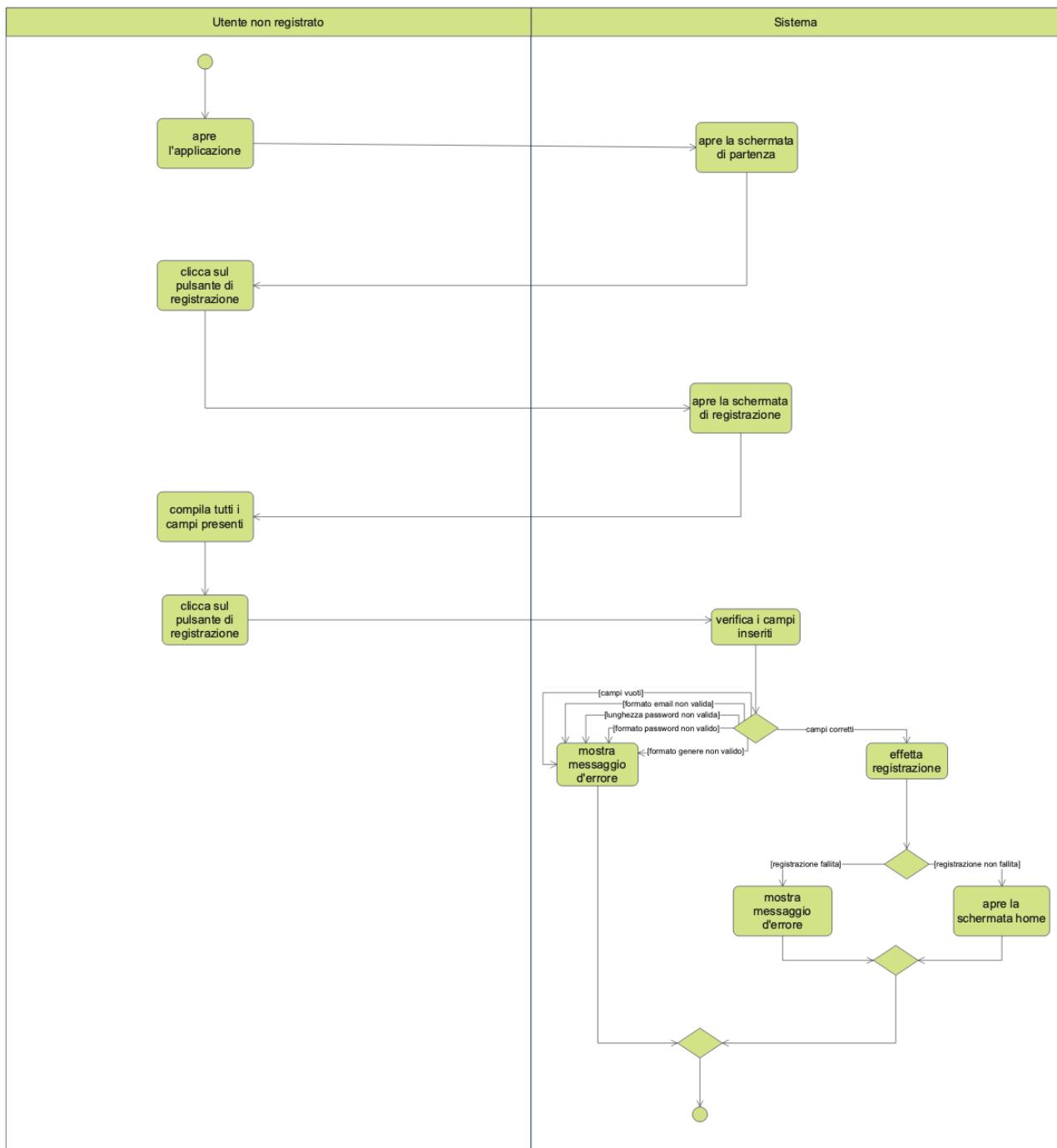


Diagramma d'attività 01 - Schermata registrazione alla piattaforma

• Accesso alla piattaforma

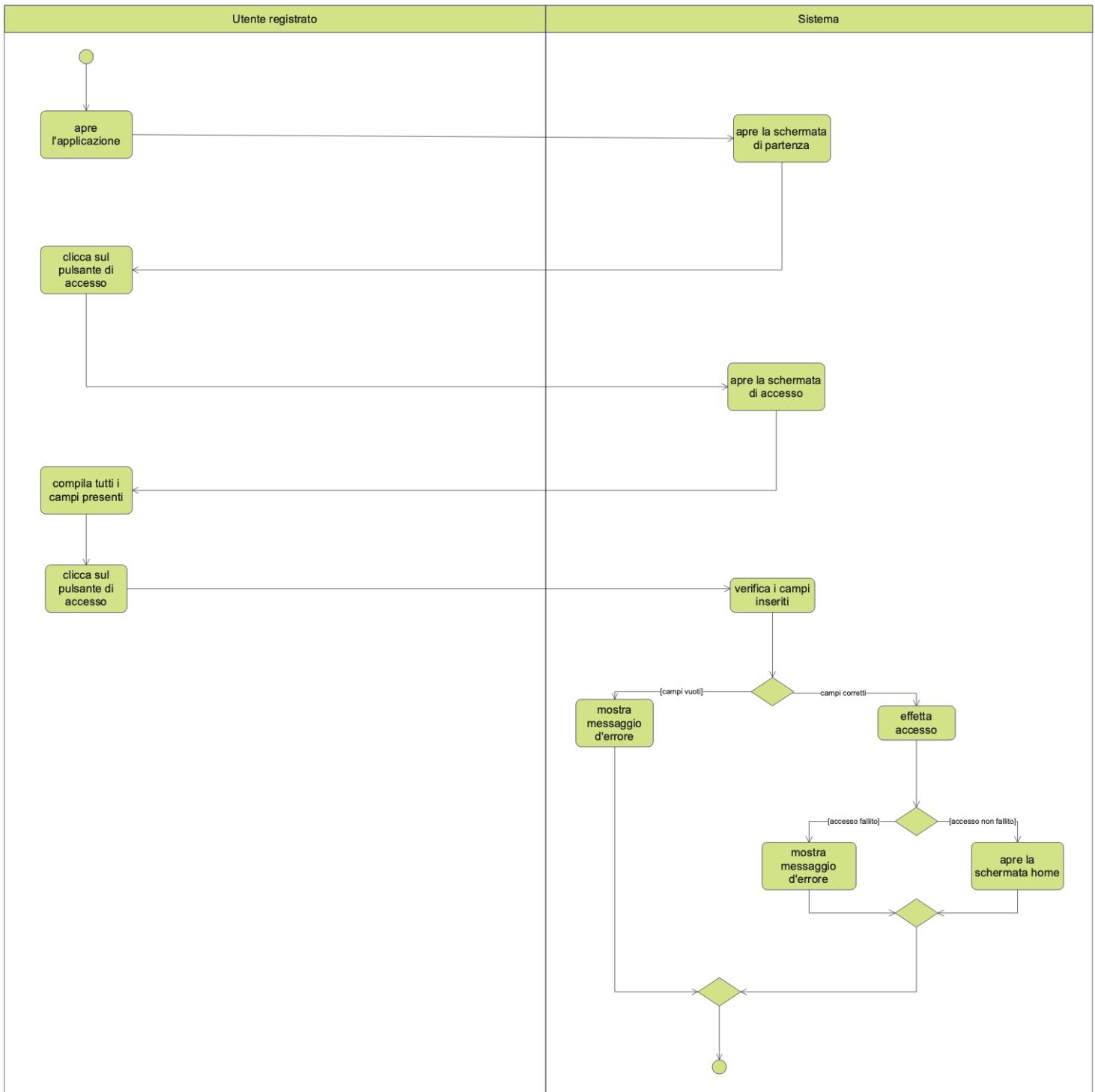


Diagramma d'attività 02 - Schermata accesso alla piattaforma

- **Visualizza lista itinerario**

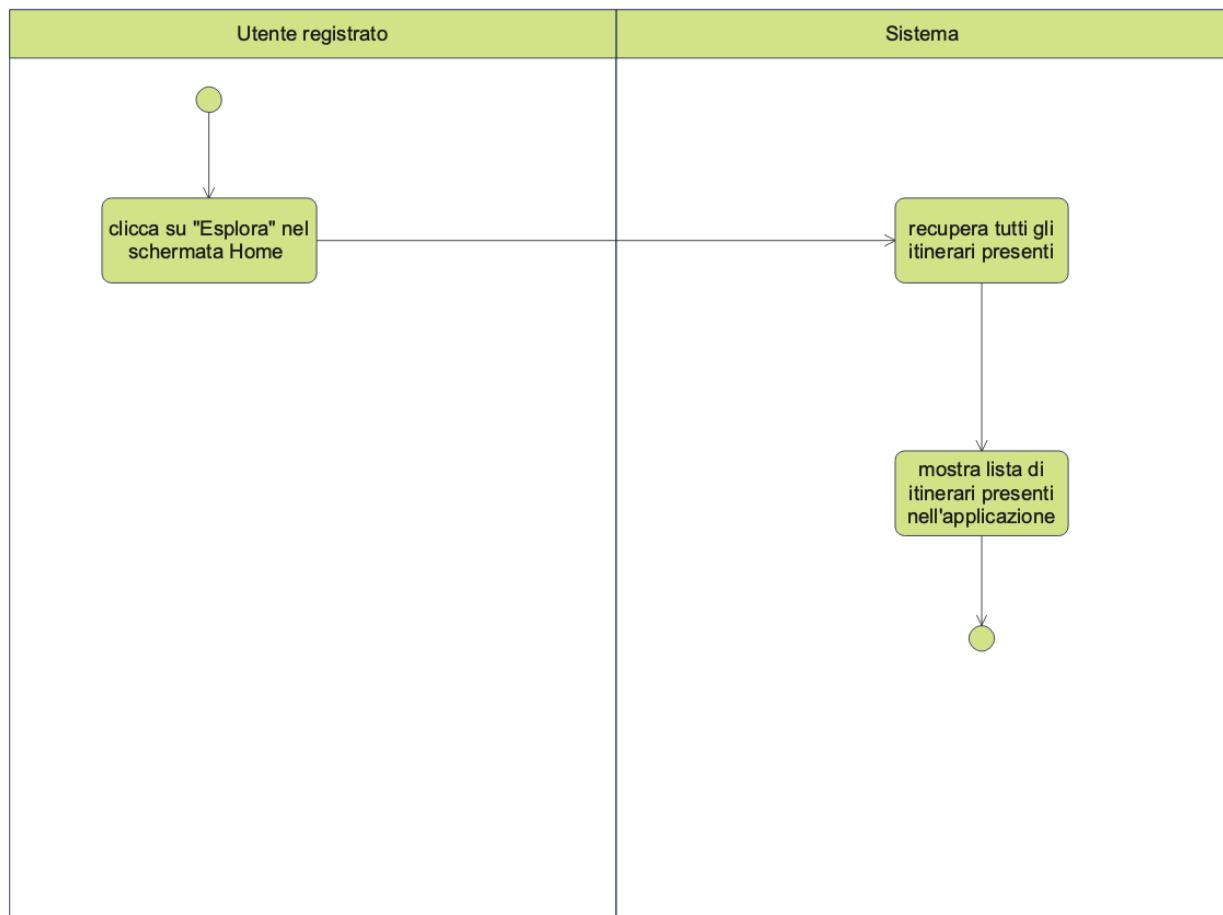
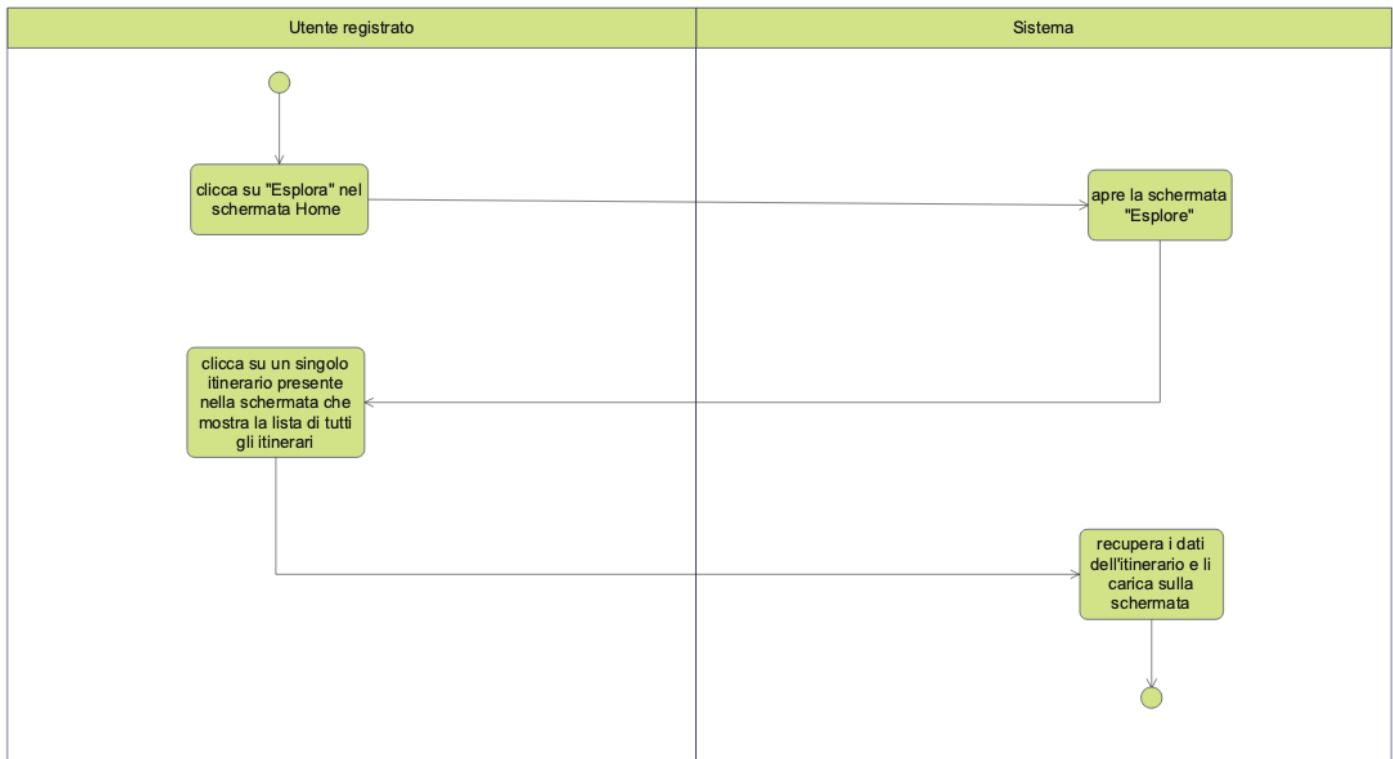


Diagramma d'attività 03 - Schermata visualizza lista itinerari

- **Visualizza informazioni che dettaglia l'itinerari**



- **Visualizza lista recensioni**

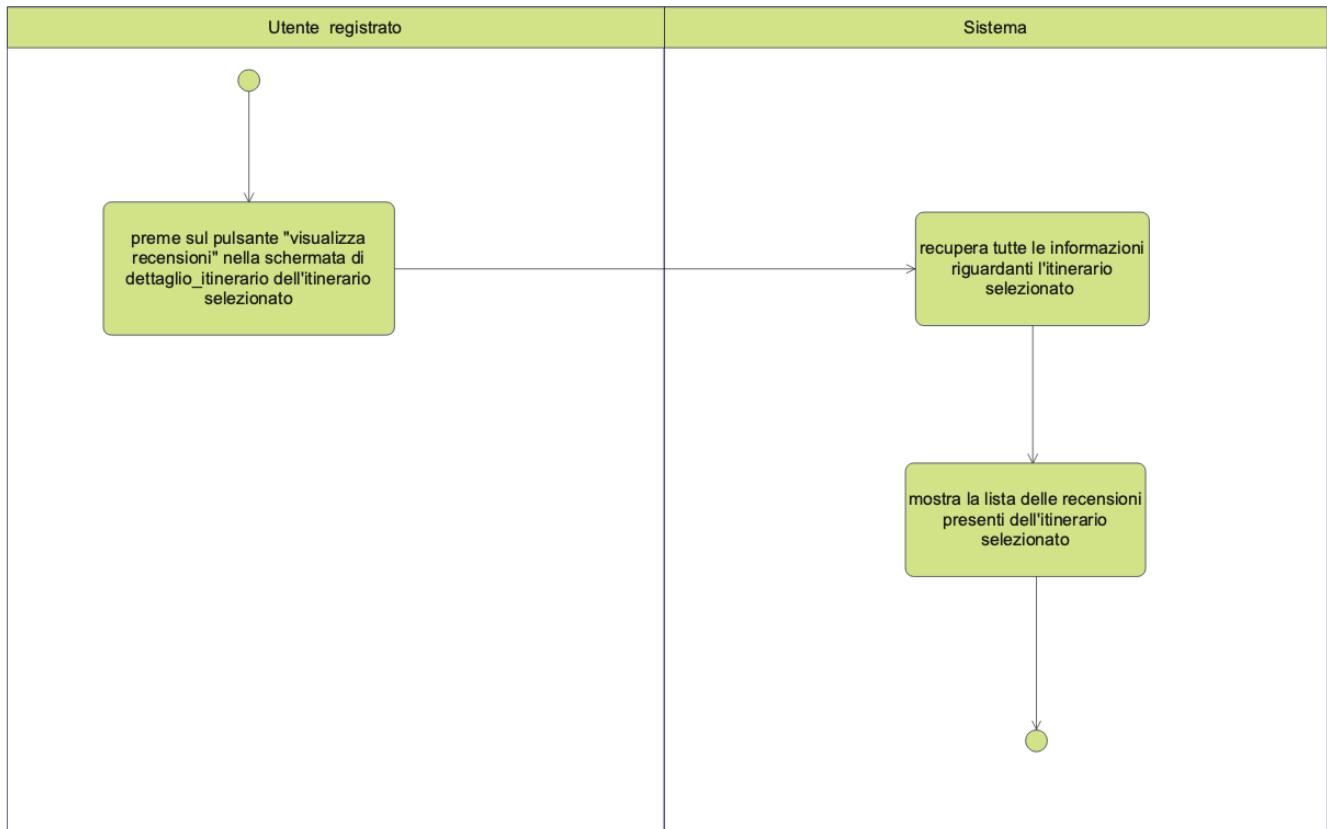


Diagramma d'attività 05 - Schermata lista recensioni di un singolo itinerario

• Inserimento recensione

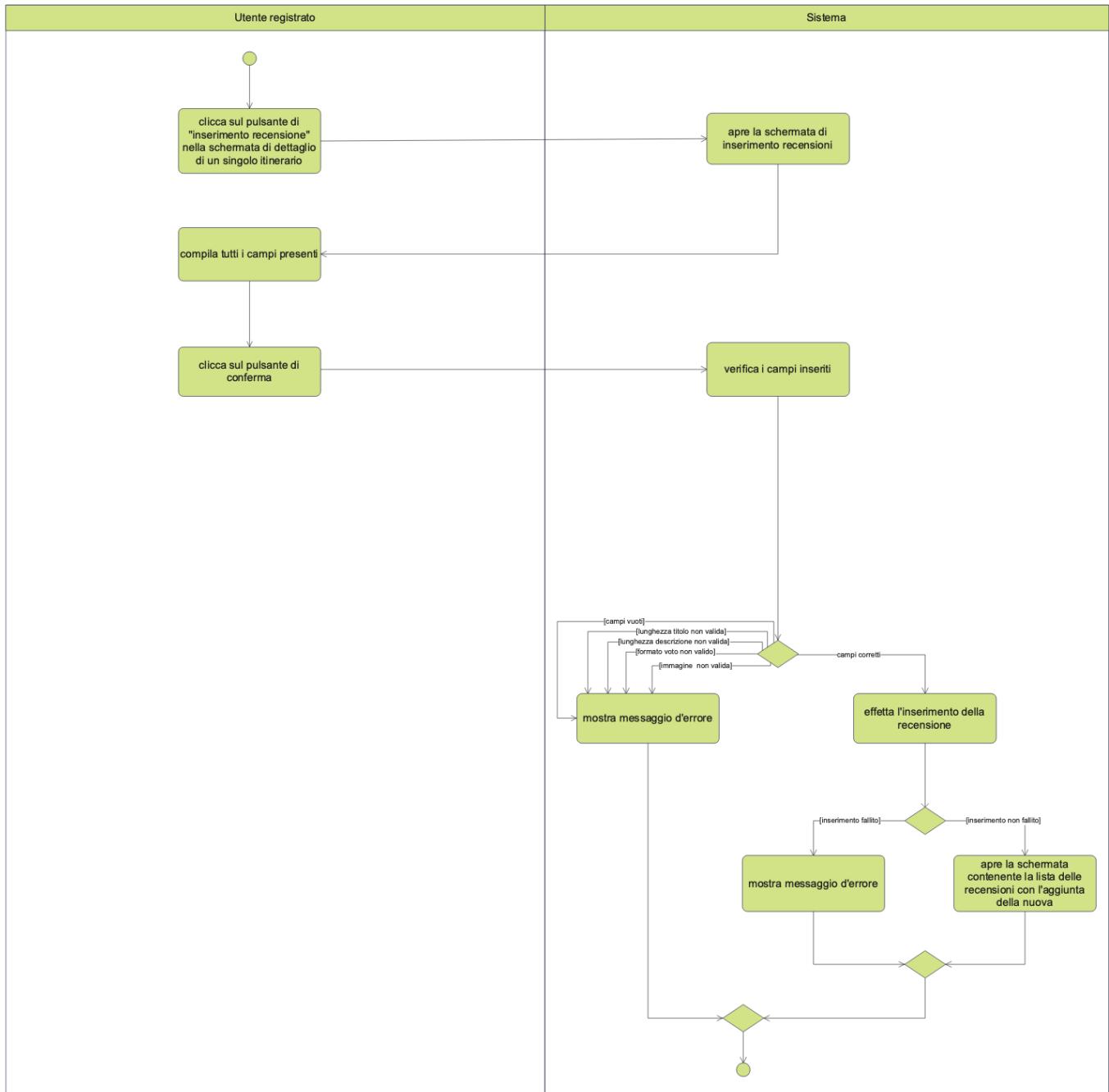


Diagramma d'attività 06 - Schermata di inserimento recensione

- Indicare posizione dell'utente sulla mappa

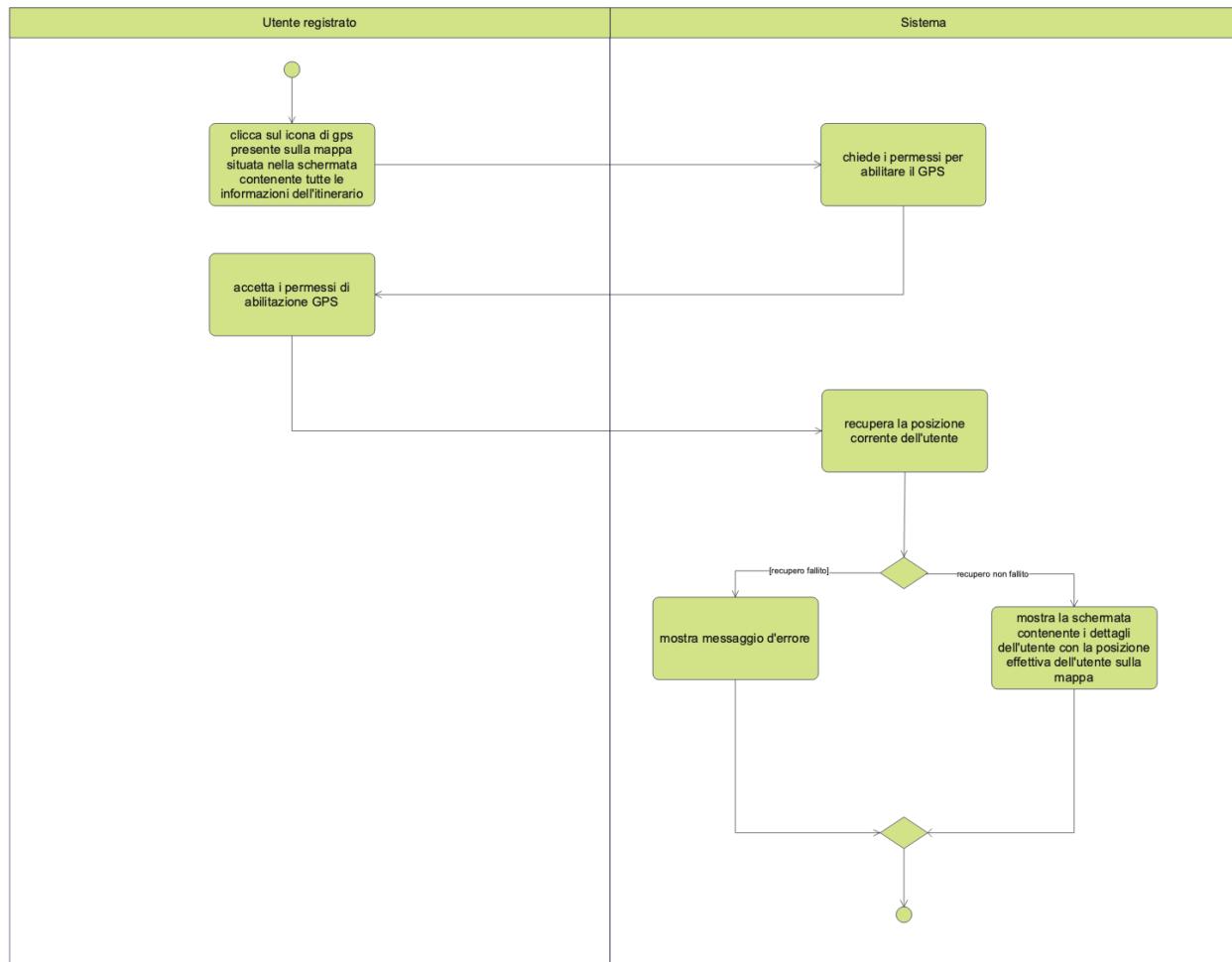


Diagramma d'attività 07 - Schermata che permette di indicare la posizione attuale dell'utente sulla mappa presente nella schermata di dettaglio itinerario

• Inserimento punti d'interesse sulla mappa

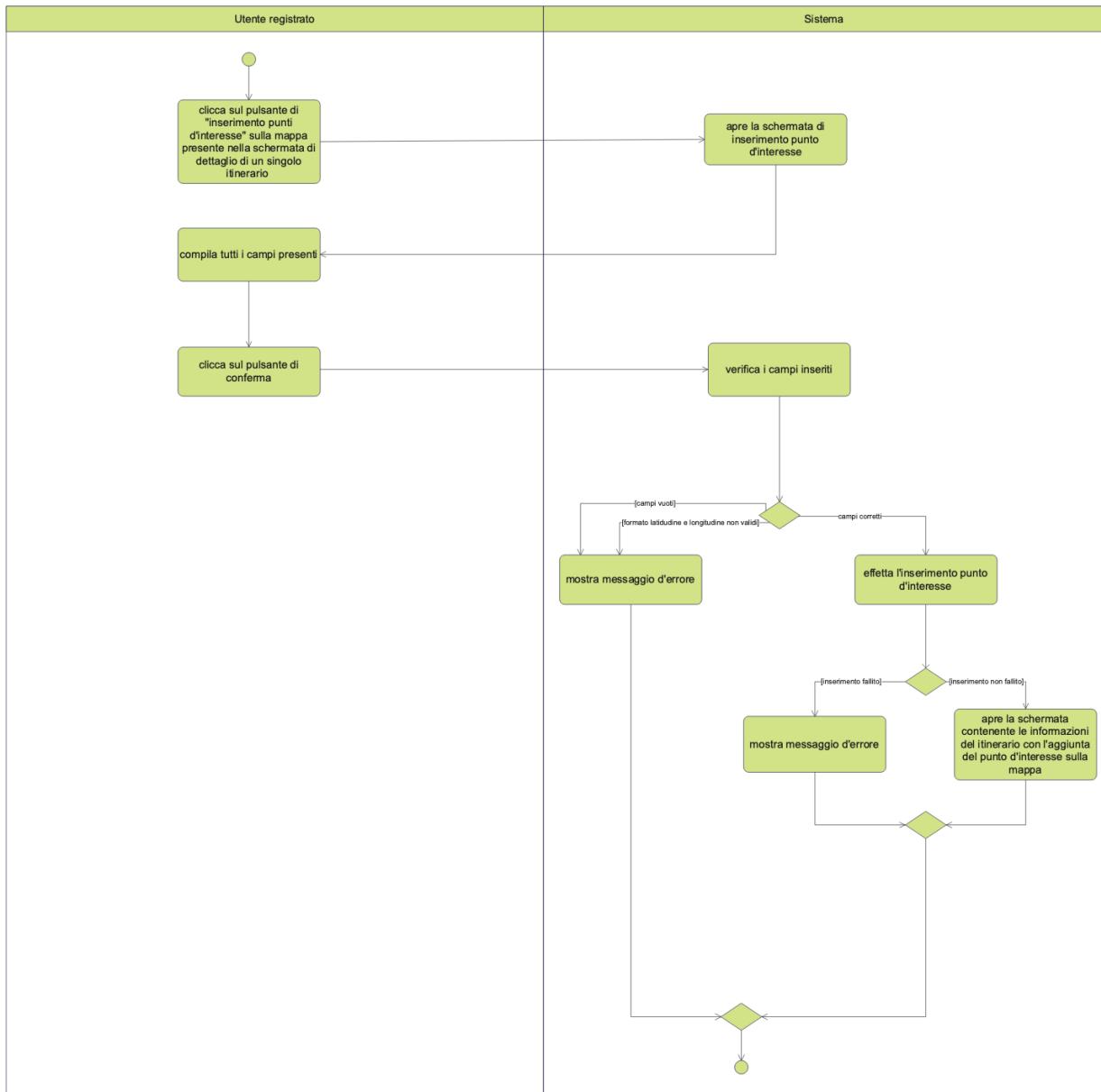


Diagramma d'attività 08 - Schermata che permette di inserire sulla mappa nella schermata di dettaglio itinerario uno o più punti d'interesse

• Inserimento nuovo itinerario

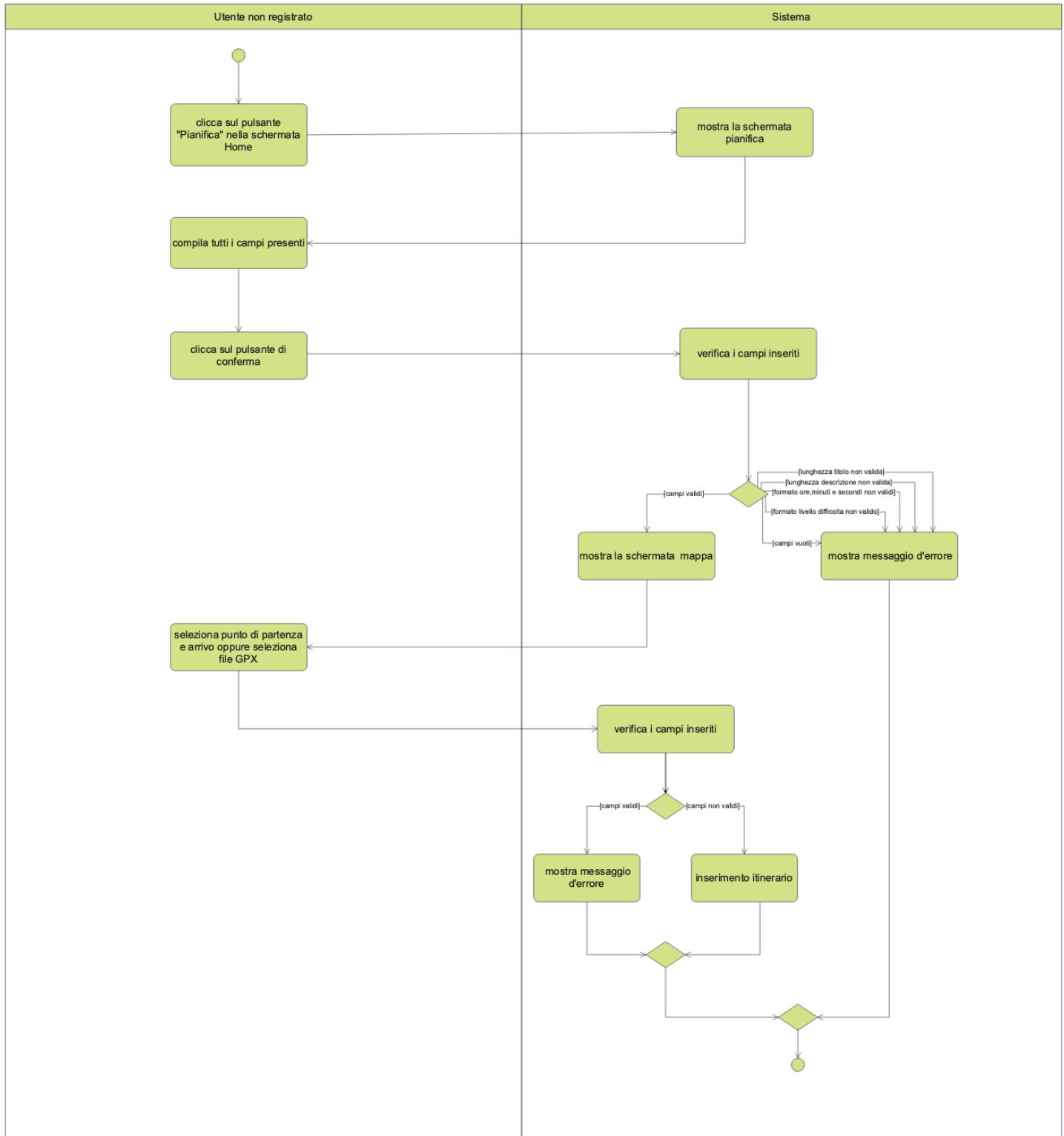


Diagramma d'attività 09- Schermata che permette di inserire un nuovo itinerario

• Uscita dalla piattaforma

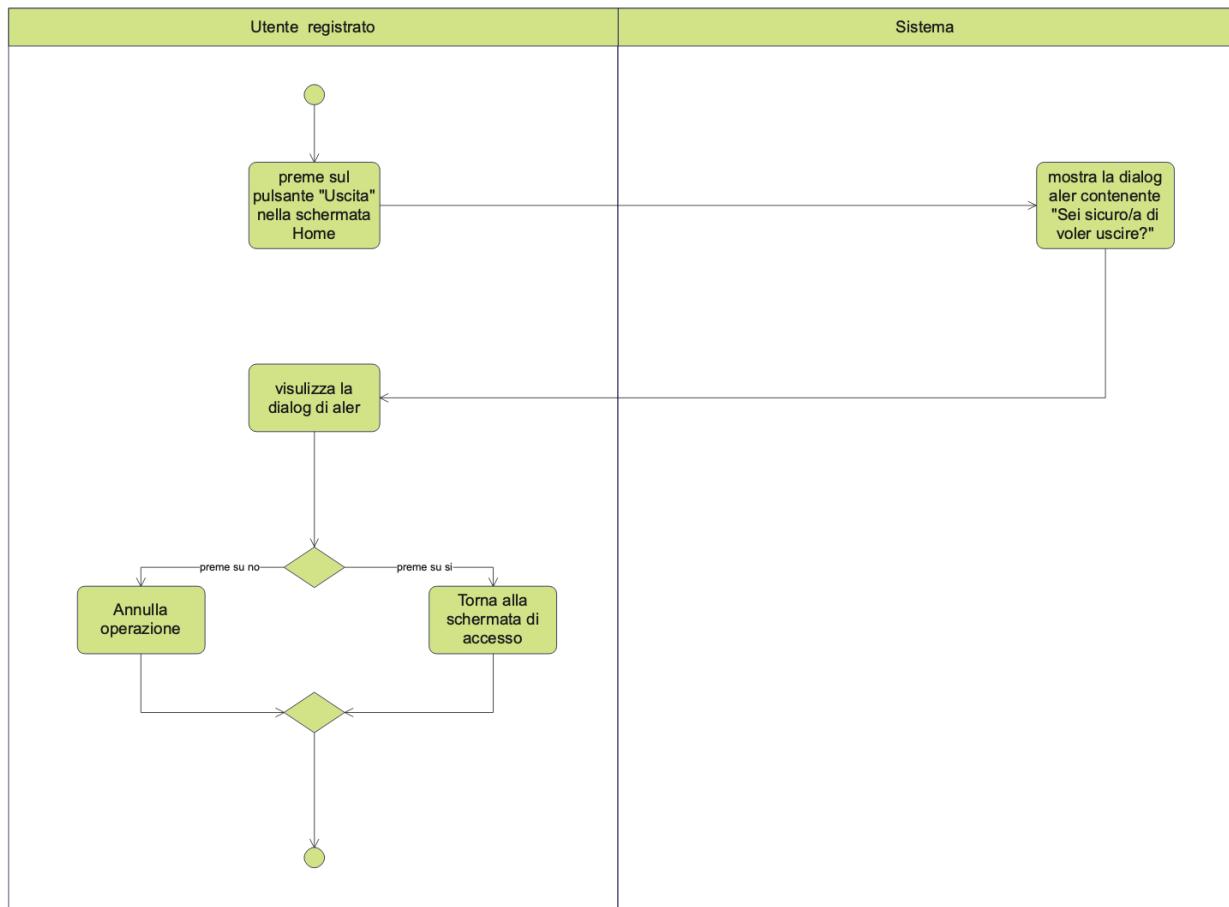


Diagramma d'attività 10 - Schermata che permette di uscire dall'applicazione

- **Visualizza profilo**

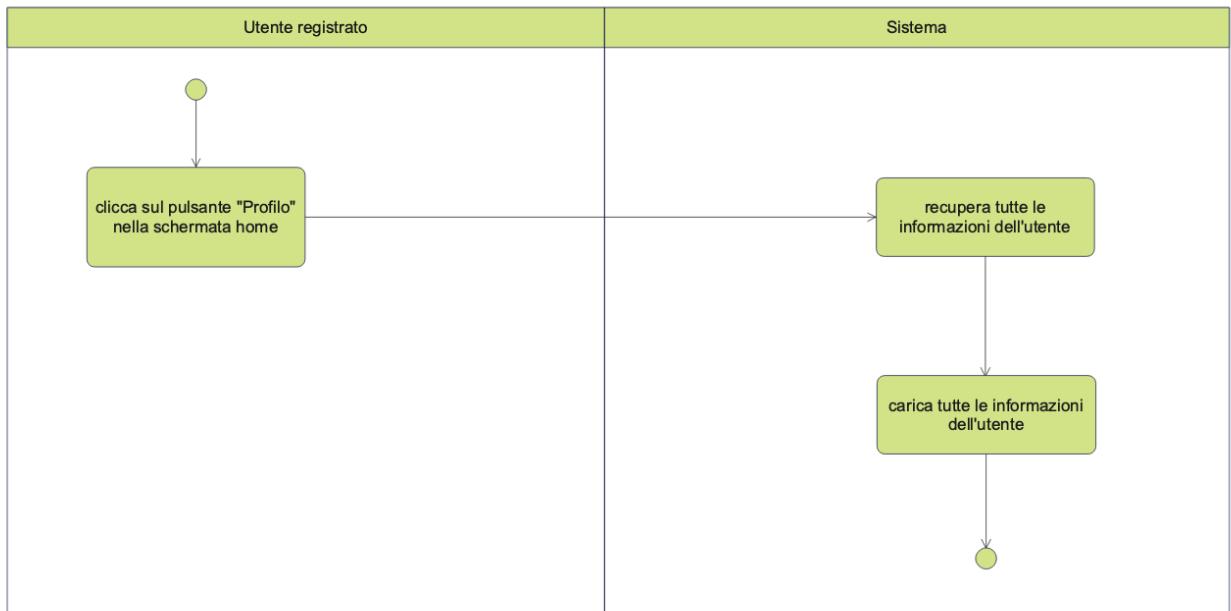


Diagramma d'attività 10 - Schermata che permette di visualizzare il profilo dell'utente

Capitolo IV – Design

O Documento di Design del sistema

- Analisi dell'architettura

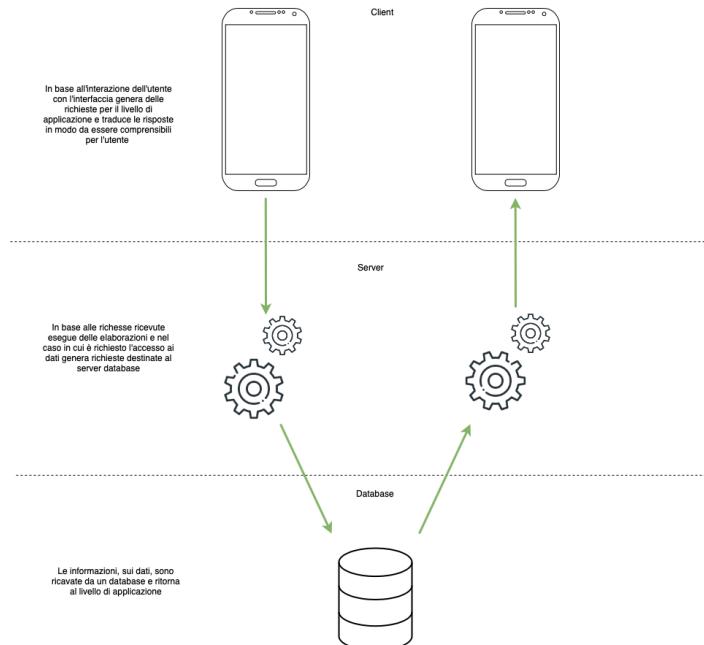
• Architettura esterna

L'architettura esterna dell'applicazione NaTour22 è stata realizzata tramite l'architettura "a tre livelli", che si articola in tre differenti moduli:

- l'interfaccia utente;
- la logica dell'applicazione;
- l'archiviazione dei dati;

Inoltre l'accesso agli stessi è sviluppato e mantenuto sotto forma di moduli indipendenti. Questo permette che nel caso in cui si presentano delle variazioni nei requisiti o nella tecnologia, è possibile aggiornare o sostituire qualsiasi dei tre livelli senza dover modificare gli altri.

Nel dettaglio l'architettura si presenta come nell'immagine sottostante:



I client inviano richieste al back-end, questo interpreta ciò che è contenuto nelle richieste, esegue eventuali elaborazioni e a sua volta genera altre richieste che vengono destinate al motore del database. Appena il livello intermedio riceve i dati dal database, genera il risultato in un output che viene reso comprensibile all'utente nella forma più appropriata.

• Architettura Server

Per l'architettura del Server, poiché si vuole realizzare un prodotto che risulti scalabile e affidabile, si è preferito utilizzare servizi di tipo MBaaS (Mobile back-end as a service) che permettono allo sviluppatore di dedicarsi alla creazione del prodotto al fine di restituire in output un prodotto che risulti scalabile e affidabile anziché concentrarsi sulla gestione e il funzionamento di server e di runtime.

Tuttavia l'applicazione sarà comunque eseguita, lo stesso, sul server ma la gestione sarà a carico di Firebase.

Lo sviluppo di Firebase iniziò dall'omonima azienda che nel 2011 sviluppò la piattaforma con l'idea di fornire un servizio in grado di sincronizzare dati in tempo reale, successivamente ricevette grande interessa da parte di Google che nel 2014 acquistò Firebase e altre startup simili, integrandole con i suoi servizi Google Cloud Platform.

Firebase offre diversi servizi, mettendo a disposizione anche SDK (Software Development Kit) e API (Application Programming Interface) multi-piattaforma (Android, iOS, JavaScript, C++) per interagire con essi. I servizi offerti da Firebase sono circa 20 e tra i vari servizi forniti, quelli utilizzati per la produzione di NaTour22 sono:

FIREBASE AUTENTIFICATION

Permette di implementare funzioni di autenticazione degli utenti nell'applicazione attraverso l'uso di passwords, numeri di telefono, o identity providers come Google, Facebook, Twitter e altri.

Utilizzando la SDK di Firebase Authentication è possibile integrare manualmente uno o più metodi di sign-in nella propria applicazione.

Nel caso di studio si è scelto di basare l'autenticazione sull'email e sulla password dell'utente. Per prima cosa quindi verranno inserite le credenziali (in questo caso email e password), che verranno passate alla SDK di Firebase Authentication. I servizi back-end di Firebase verificheranno tali credenziali. Dopodiché sarà possibile accedere alle informazioni di base dello user e controllare il suo accesso ai dati immagazzinati in altri strumenti Firebase, ossia Firebase Database.

REALTIME DATABASE

È un Database NoSQL cloud-hosted. In quanto NoSQL ha diverse ottimizzazioni e funzionalità rispetto ad un database relazionale, che verranno trattate più nel dettaglio in seguito. L'API di Realtime Database è disegnata per permettere solamente operazioni che possano essere eseguite rapidamente. Questo fa sì che pur avendo milioni di utenti attivi sull'applicazione, non ci siano problemi di responsività. Realtime Database utilizza la sincronizzazione dei dati invece delle solite richieste HTTP. In questo modo ogni volta che i dati vengono modificati, il client connesso riceve tali cambiamenti istantaneamente.

I dati vengono sincronizzati attraverso i clients in realtime, e rimangono disponibili anche se l'applicazione va offline; nel momento in cui la connessione riprende, la funzione di sincronizzazione del Realtime Database provvede tutti i file mancanti al client, cosa molto importante per la stabilità del lavoro svolto dall'applicazione.

Analizziamo di seguito le differenze tra database relazionale e i database NoSQL :

- La strutturazione rigida dei contenuti, tipica dei database relazionali, è un elemento assente nei database NoSQL, e proprio tale assenza è uno degli aspetti che maggiormente ne hanno permesso il successo.
- Le informazioni non sono più memorizzate in tabelle, ma in oggetti completamente diversi e non necessariamente strutturati, come ad esempio documenti archiviati in collezioni.
- Un oggetto JSON ad esempio rappresenta un documento da inserire nel database.
- La chiave primaria del documento è un campo denominato '_id' che, se non viene fornito in fase di inserimento, verrà aggiunto automaticamente dal DBMS.
- Nei DBMS NoSQL è significativa l'assenza delle relazioni. I meccanismi con cui vengono collegate le informazioni sono infatti due:
 - Embedding: significa annidare un oggetto JSON all'interno di un altro. Questa tecnica sostituisce molto spesso le relazioni 1-a-1 e 1-a-molti. È tuttavia sconsigliabile utilizzarla quando i documenti (quello annidato e quello che lo contiene) crescono di dimensione in maniera sproporzionata tra loro, oppure se la frequenza di accesso ad uno dei due è molto minore di quella dell'altro;
 - Referencing: somiglia molto alle relazioni dei RDBMS, e consiste nel fare in modo che un documento contenga, tra i suoi dati, l'id di un altro documento. Molto utile per realizzare strutture complesse, relazioni molti-a-molti.

Di conseguenza si predilige un database NoSQL in quanto non sempre i DBMS tradizionali si dimostrano in grado di gestire quantità di dati molto grandi, se non al prezzo di performance molto più limitate e costi piuttosto elevati. Qui di seguito sono stati raccolti alcuni punti di debolezza ravvisati nell'uso dei DBMS relazionali:

- I Join: nonostante la loro efficacia, queste operazioni coinvolgono spesso più righe del necessario (soprattutto se le query non sono ottimizzate), limitando le performance delle interrogazioni eseguite;
- struttura rigida delle tabelle, che si rivela utile finché si ha necessità di introdurre informazioni caratterizzate sempre dalle medesime proprietà, ma molto meno efficiente per informazioni di natura eterogenea;
- conflitto di impedenza, che consiste nella differenza strutturale tra i record delle tabelle e gli oggetti comunemente utilizzati per gestire i dati nei software che interagiscono con le basi di dati. Questo problema ha prodotto – tra l'altro - la nascita di strumenti come gli ORM, librerie in grado di convertire oggetti in record (e viceversa);
- proprietà ACID: ogni transazione deve essere atomica, consistente, isolata e duratura. Implementare queste proprietà, però, comporta una serie di controlli e precauzioni che penalizzano le prestazioni del sistema. I database non relazionali sono meno stringenti, offrendo una maggiore elasticità.

FIREBASE CLOUD STORAGE

È un Servizio di storage offerto da Firebase. Le SDK di Firebase per Cloud Storage fanno sì che le operazioni di upload e download dei file vengano eseguite secondo i criteri di sicurezza di Google.

Grazie ad esse è possibile immagazzinare immagini, audio, video e altri contenuti user-generated direttamente da client.

Le principali caratteristiche di Cloud Storage sono:

- Operazioni robuste: gli SDK Firebase per Cloud Storage eseguono caricamenti e download indipendentemente dalla qualità della rete. I caricamenti e i download sono robusti, il che significa che si riavviano da dove sono fermati, risparmiando tempo e larghezza di banda agli utenti
- Forte sicurezza: gli SDK Firebase per Cloud Storage si integrano con l'autenticazione Firebase per fornire l'autenticazione semplice e intuitiva per gli sviluppatori.
- Alta scalabilità: Cloud Storage è progettato per una scalabilità di exabyte quando l'applicazione diventa virale.

FIREBASE GOOGLE ANALYTICS

Google Analytics è una soluzione per misurare l'utilizzo delle applicazioni da parte degli utenti. In particolare fornisce informazioni dettagliate sull'utilizzo delle applicazioni e sul coinvolgimento degli utenti.

Le principali caratteristiche di google analytics sono:

- Reportistica illimitata : analytics fornisce rapporti illimitati su un massimo di 500 eventi distinti
- Segmentazione del pubblico : I segmenti di pubblico personalizzati possono essere definiti nella console Firebase in base ai dati del dispositivo, agli eventi personalizzati o alle proprietà dell'utente.

• Architettura Client

Il client di NaTour22 è stato sviluppato su piattaforma Android Studio. Si è scelto di utilizzare Android Studio per la vastità di materiale e documentazione disponibile in internet. Inoltre delle ricerche effettuate, per la valutazione delle tecnologie da utilizzare, Android Studio è risultato quello più adatto per la semplicità e la chiarezza di utilizzo dell'interfaccia, nonché lo strumento più utilizzato a livello mondiale per lo sviluppo di applicazioni Android.

Android Studio è un ambiente di sviluppo integrato (IDE) basato sul software di JetBrains IntelliJ IDEA per lo sviluppo di applicazioni Android. Le versioni di Android Studio sono compatibili con Linux, Windows e Macintosh. Android Studio rende lo sviluppo dell'applicazione più facile e veloce. Inoltre essendo stato sviluppato da Google appositamente per Android, è possibile realizzare applicazioni per dispositivi mobili con sistema operativo Android che dialoghino facilmente con gli svariati servizi di cloud forniti da Google.

Altre caratteristiche messe a disposizione da Android Studio sono:

- editor per layout visuale utilizzabile in modalità drag and drop;
- accesso a SDK Manager e AVD Manager;
- inline debugging;
- monitoraggio delle risorse di memoria e della CPU utilizzate dall'app.

Per l'implementazione si è scelto di utilizzare un linguaggio *Object Oriented*, in particolare *Java*, con l'ausilio del design pattern “Model View Presenter” (MVP).

Il MVP (Model View Presenter) è un pattern architettonico attraverso il quale è possibile trarre vantaggio in termini di prestazioni, leggibilità e modularità del codice e si è preferito in quanto si adatta meglio all'ambiente Android.

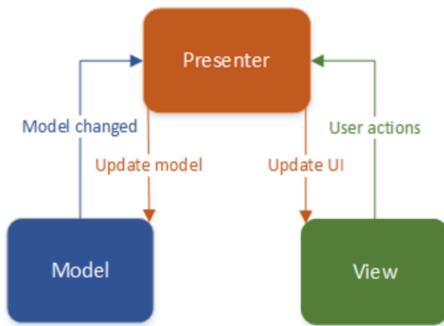
La sua caratteristica principale è quella di separare il livello di presentazione dalla logica, in modo che tutto ciò che riguarda l'interazione dell'utente sia separato da come vengono rappresentati i dati.

Il pattern MVP deriva dal pattern MVC (Model View Controller) che ha 3 concetti base, che lo definiscono:

- Model: Il modello dei dati da visualizzare.
- View: L'interfaccia utente che visualizza i dati.
- Controller: Controlla l'interazione tra Model e View.

La principale differenza tra i due pattern risiede nel Presenter che gestisce la logica tra la View e il Model. Come il pattern MVC, anche il pattern MVP permette di rendere le View indipendenti dalla gestione dei dati, dividendo la logica dell'applicazione in tre livelli distinti, livelli che possono essere testati separatamente.

Nel dettaglio il design pattern MVP si presenta come nell'immagine sottostante:



- Model - rappresenta il modello dei dati di interesse per l'applicazione. Tale livello si occupa di incapsulare lo stato dell'applicazione, gestisce l'accesso alla sorgente dei dati, fornisce funzionalità per l'aggiornamento dello stato e l'accesso ai dati.
- View - appresenta l'interfaccia utente, permettendo l'interazione con esso e fornendo una rappresentazione grafica ed interattiva del model. La responsabilità di questo livello è la presentazione dei dati e dello stato dell'applicazione. Inoltre, esso riceve notifiche dal Presenter e aggiorna la visualizzazione.
- Presenter - tale livello definisce la logica di controllo e le funzionalità applicative. Quindi, è compito di questo livello gestire gli eventi ed i comandi generati dall'utente: in base a questi ultimi, esso opera sul model. Infine, tale livello si occupa di selezionare/aggiornare il livello View in base ai dati recuperati.

- Modelli di Design: Applicazione Mobile

- Diagrammi delle classi di design

Vengono ora elencati i Class Diagram di design relativi alle funzionalità che l'applicazione mobile deve offrire.

- Accedi alla piattaforma



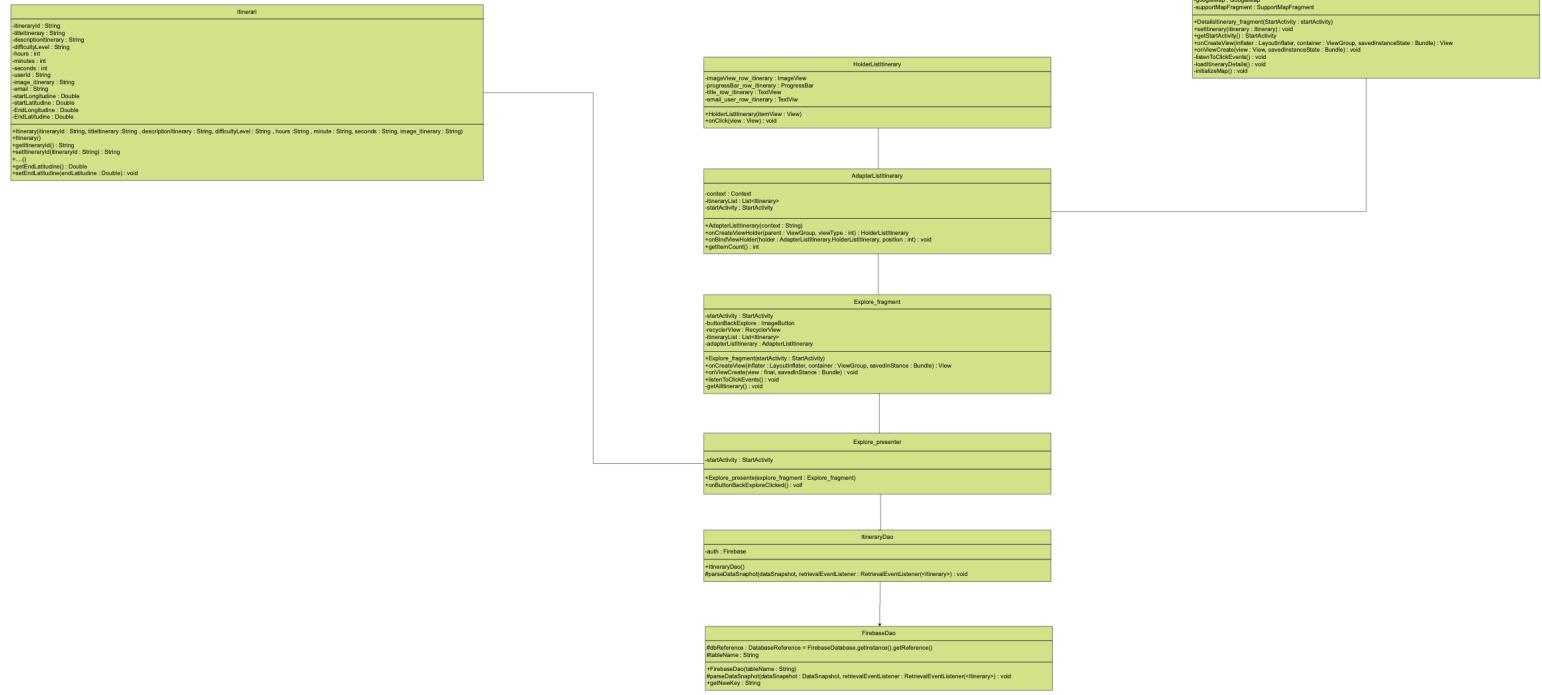
Class Diagram 01 - accesso alla piattaforma

- Registrazione alla piattaforma



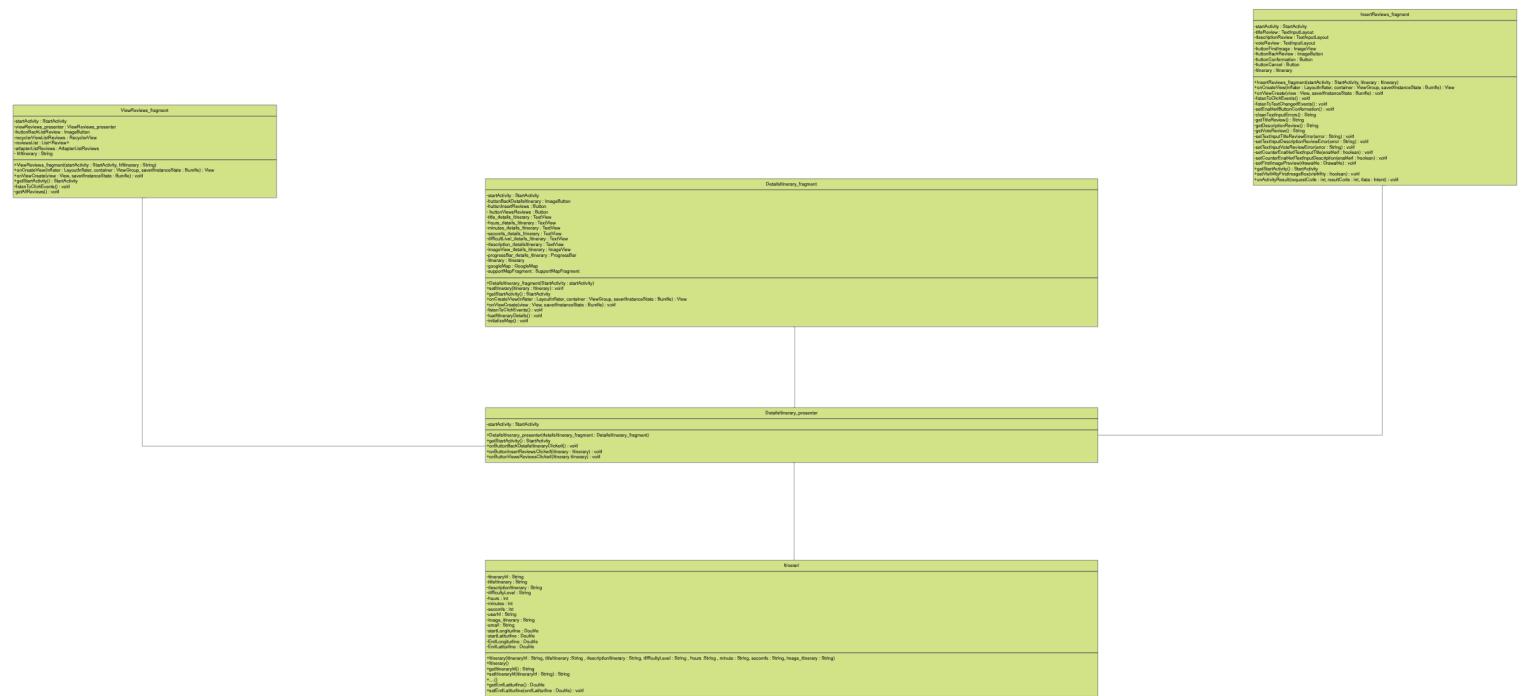
Class Diagram 02 - registrazione alla piattaforma

- Visualizza lista itinerari



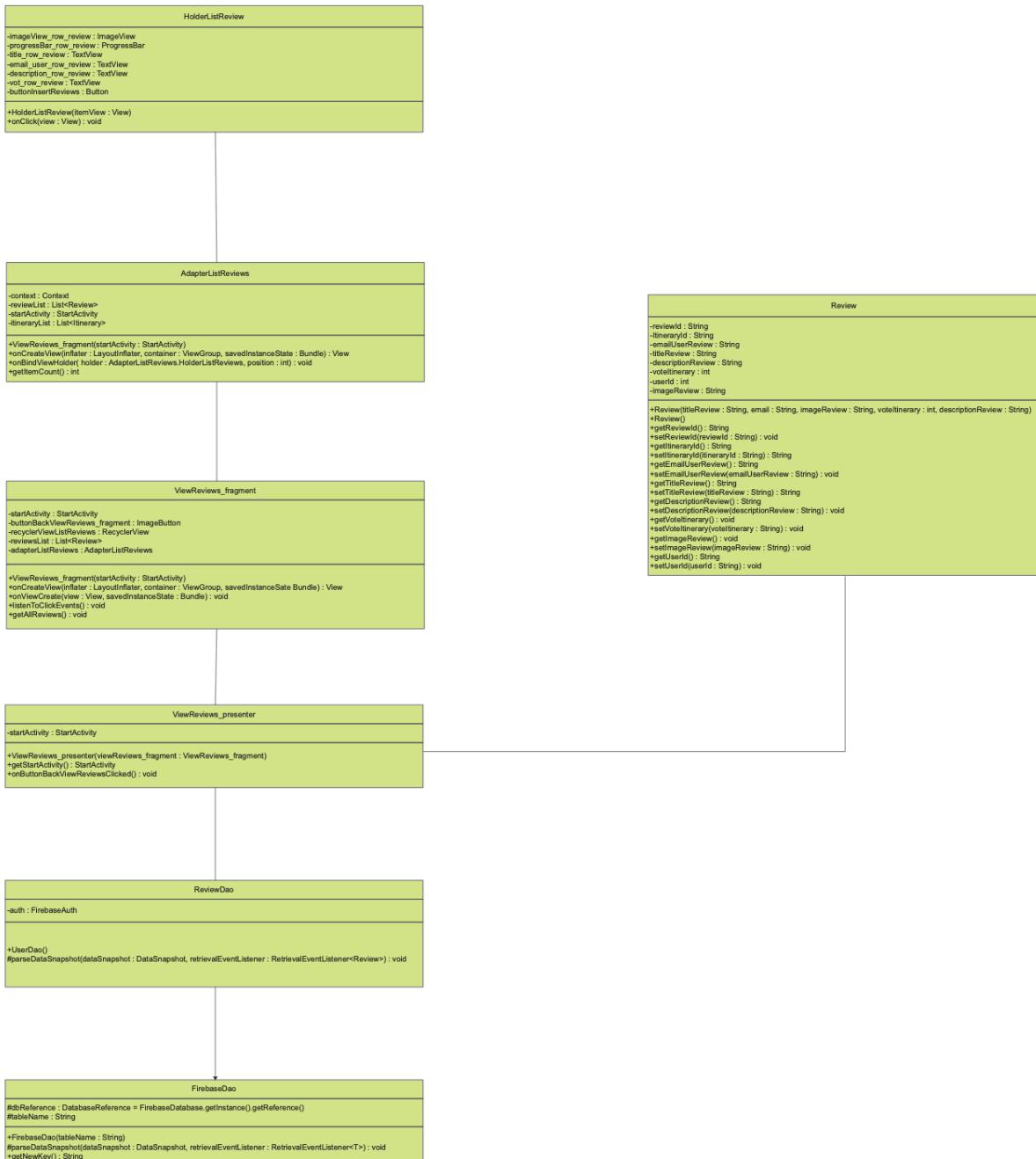
Class Diagram 03 -visualizza lista itinerari

- Visualizza dettagli itinerari



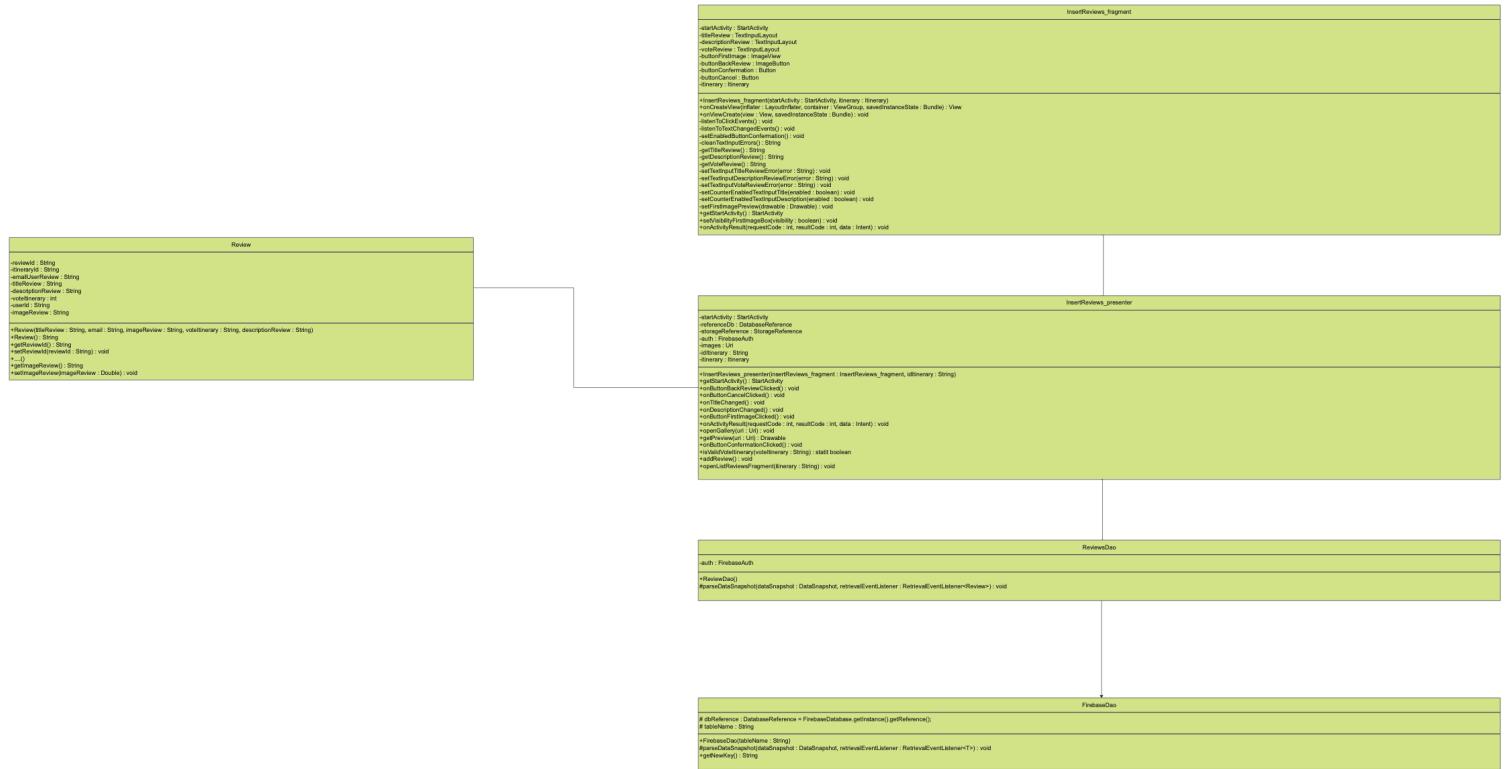
Class Diagram 04- visualizza dettagli itinerari

- Visualizza lista recensioni



Class Diagram 05 -visualizza lista recensioni

- Inserimento recensione



Class Diagram 06 -Inserimento di una recensione per l'itinerario

- Inserimento nuovo itinerario



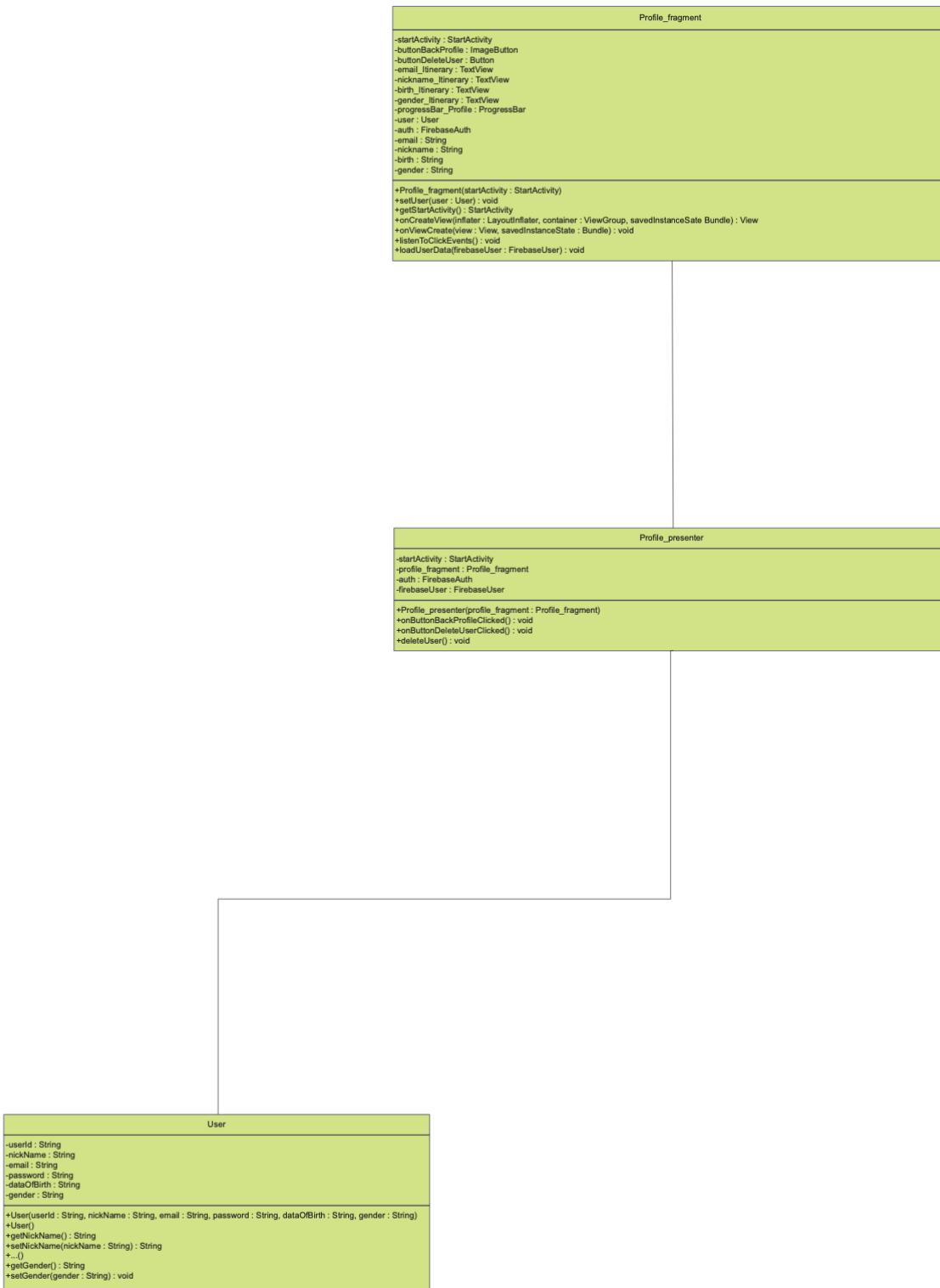
Class Diagram 07 -Inserimento di un nuovo l'itinerario

- Inserimento tappa lungo un itinerario



Class Diagram 08 -Inserimento di una tappa lungo un itinerario

- Visualizza profilo

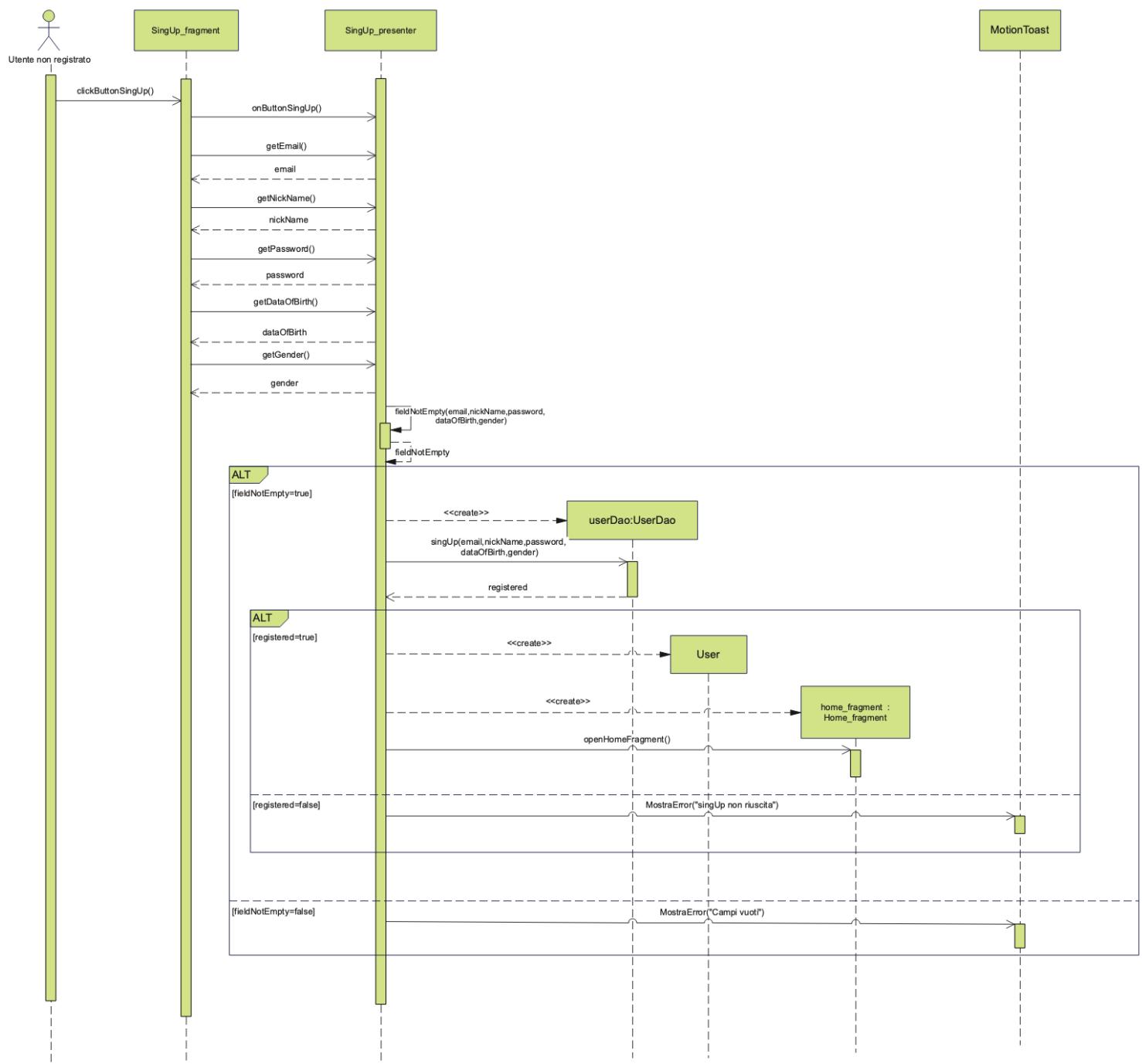


Class Diagram 09 -visualizza profilo utente

• Diagrammi di sequenza di design

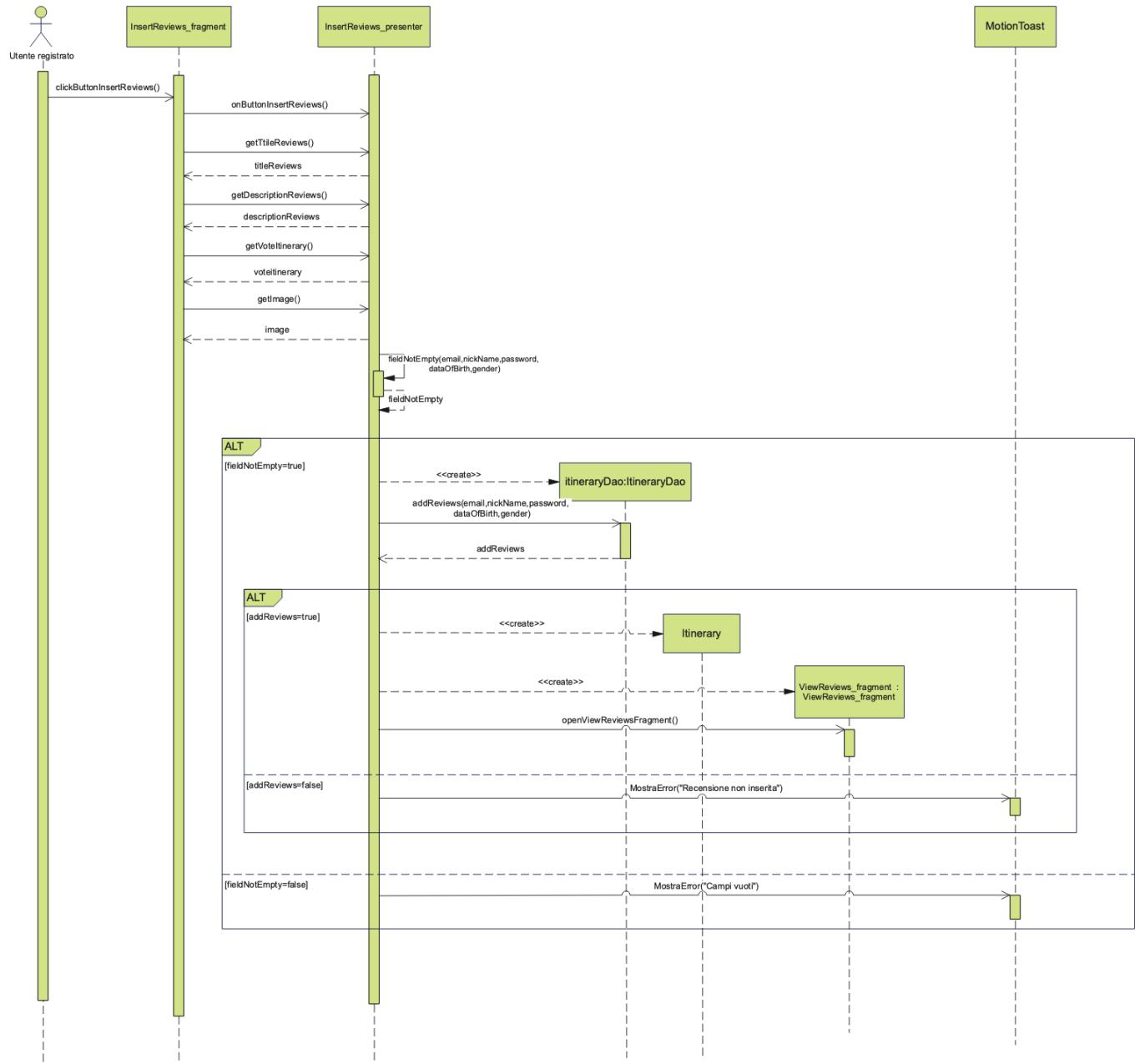
Sono elencati i Sequence diagram di design relativi alle funzionalità che l'applicazione mobile deve offrire.

- Registrazione alla piattaforma



Sequence Diagram 01 - Schermata di registrazione alla piattaforma

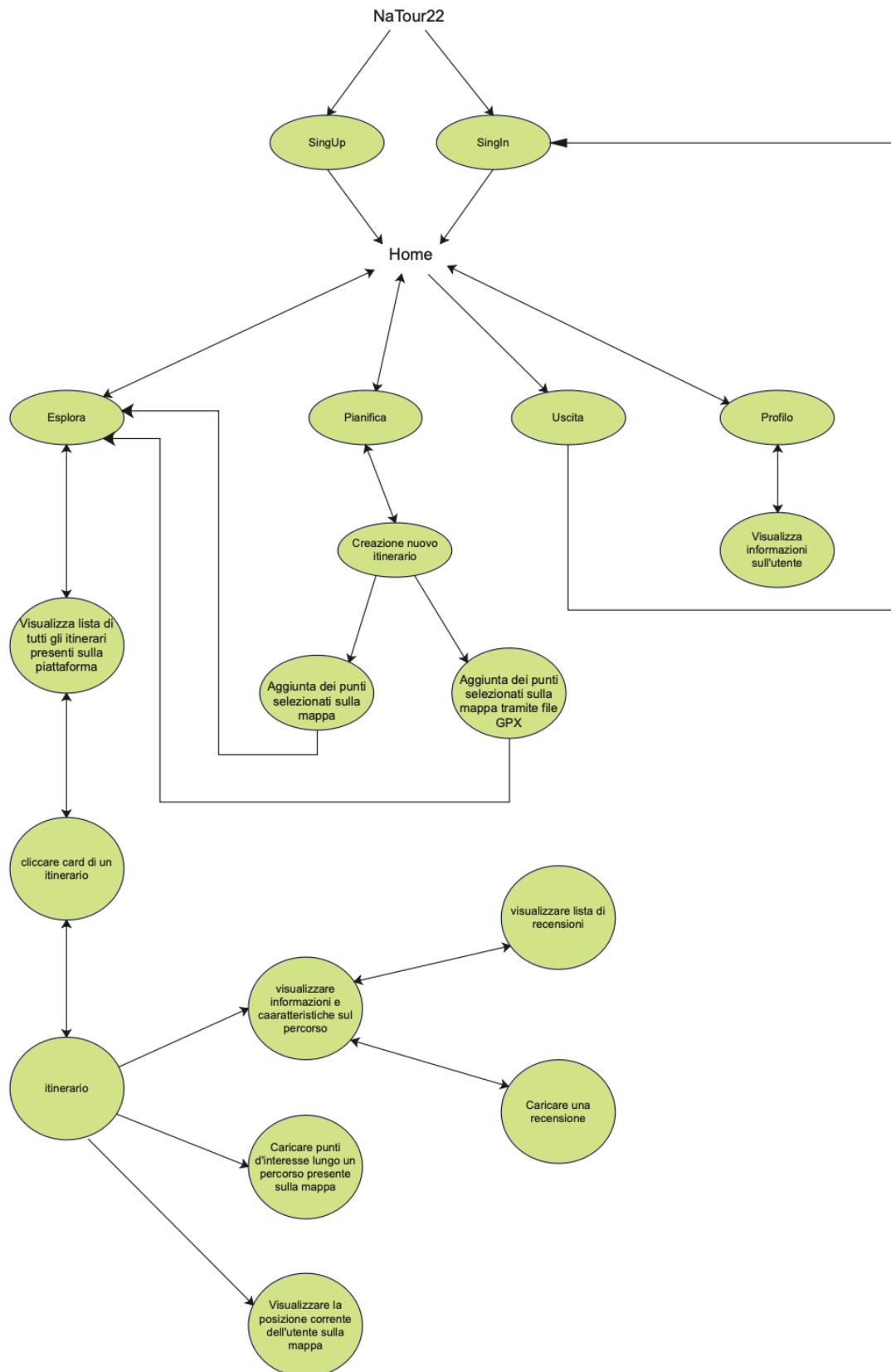
- Inserimento recensione



Sequence Diagram 02 - Schermata di inserimento recensione

• Definizione della gerarchie funzionali

Viene ora mostrata la definizione della gerarchia funzionale dell'applicazione:



Gerarchia funzionale 01 - gerarchia funzionale dell'applicazione

Capitolo V – Codice

○ **Codice Sorgente sviluppato**

Il codice sorgente di “*NaTour22*” è disponibile sulla piattaforma GitHub visitando il seguente link :

<https://github.com/fabiola29/NaTour22>

Capitolo IV – Testing

○ Documento di Testing del sistema

- Codice xUnit per unit testing di 2 metodi

L'obiettivo principale di cui ci si occupa in questa fase è quello di isolare il codice scritto da testare e determinare se funziona come previsto.

- Corrispondenza package progetto e stringa corrispondente

```
1. package com.example.natour2022;
2.
3. import android.content.Context;
4.
5. import androidx.test.platform.app.InstrumentationRegistry;
6. import androidx.test.ext.junit.runners.AndroidJUnit4;
7.
8. import org.junit.Test;
9. import org.junit.runner.RunWith;
10.
11. import static org.junit.Assert.*;
12.
13. @RunWith(AndroidJUnit4.class)
14. public class ExampleInstrumentedTest {
15.
16.     //Controllo che il nome del package corrisponda alla stringa "com.NaTour2022"
17.     @Test
18.     public void useApplicationContext() {
19.
20.
21.         Context applicationContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
22.         assertEquals("com.example.NaTour2022", applicationContext.getPackageName());
23.     }
24. }
```

- Accesso all'applicazione

```
1. package com.example.natour2022;
2.
3. import static org.junit.Assert.fail;
4.
5. import androidx.annotation.NonNull;
6. import androidx.test.ext.junit.runners.AndroidJUnit4;
7.
8. import com.google.android.gms.tasks.OnCompleteListener;
9. import com.google.android.gms.tasks.Task;
10. import com.google.firebase.auth.AuthResult;
11. import com.google.firebase.auth.FirebaseAuth;
12. import com.google.firebase.auth.FirebaseAuthInvalidCredentialsException;
13.
14. import org.junit.Assert;
15. import org.junit.Test;
16. import org.junit.runner.RunWith;
17.
18. @RunWith(AndroidJUnit4.class)
19. public class SingIn_Test {
20.     FirebaseAuth auth;
21.     boolean test;
22.
23.     @Test
24.     public void ValidEmailPasswordTest(){
25.
26.         test = false;
27.         auth = FirebaseAuth.getInstance();
28.
29.         auth.signInWithEmailAndPassword("prova1@live.it", "Prova3456").addOnCompleteListener(new
30.             OnCompleteListener<AuthResult>() {
31.                 @Override
```

```

32.     public void onComplete(@NonNull Task<AuthResult> task) {
33.         if (task.isSuccessful()) {
34.             Assert.assertTrue(true);
35.
36.         } else {
37.             Assert.assertFalse(false);
38.         }
39.         test = true;
40.     }
41. );
42. while(!test){
43.
44. }
45.
46.
47. @Test(expected = IllegalArgumentException.class)
48. public void InvalidEmptyPasswordTest(){
49.
50.     test = false;
51.     auth = FirebaseAuth.getInstance();
52.
53.     auth.signInWithEmailAndPassword("", "").addOnCompleteListener(new
54.     OnCompleteListener<AuthResult>() {
55.         @Override
56.         public void onComplete(@NonNull Task<AuthResult> task) {
57.             if (task.isSuccessful()) {
58.                 Assert.assertTrue(true);
59.
60.             } else {
61.                 Assert.assertFalse(false);
62.             }
63.             test = true;
64.         }
65.     });
66.     while(!test){
67.
68. }
69.
70.
71. @Test(expected = IllegalArgumentException.class)
72. public void InvalidEmptyEmailTest(){
73.
74.     test = false;
75.     auth = FirebaseAuth.getInstance();
76.
77.     auth.signInWithEmailAndPassword("", "Prova3456").addOnCompleteListener(new
78.     OnCompleteListener<AuthResult>() {
79.         @Override
80.         public void onComplete(@NonNull Task<AuthResult> task) {
81.             if (task.isSuccessful()) {
82.                 fail();
83.
84.             } else {
85.                 Assert.assertTrue(true);
86.             }
87.             test = true;
88.         }
89.     });
90.     while(!test){
91.
92. }
93.
94. }
95.
96.
97. @Test(expected = IllegalArgumentException.class)
98. public void InvalidEmptyPasswordTest(){
99.
100.    test = false;
101.    auth = FirebaseAuth.getInstance();
102.
103.    auth.signInWithEmailAndPassword("provalive.it", "").addOnCompleteListener(new
104.    OnCompleteListener<AuthResult>() {
105.        @Override
106.        public void onComplete(@NonNull Task<AuthResult> task) {
107.            if (task.isSuccessful()) {
108.                Assert.assertTrue(true);
109.                fail();
110.
111.
112.            } else {
113.                //Assert.assertFalse(false);
114.                Assert.assertTrue(true);
115.
116.            }
117.            test = true;
118.        }
119.    });
120.    while(!test){
121.
122. }

```

```

123. }
124.
125. @Test
126. public void InvalidFormatEmailPasswordTest(){
127.
128.     test = false;
129.     auth = FirebaseAuth.getInstance();
130.
131.     auth.signInWithEmailAndPassword("prova1@live", "Prova").addOnCompleteListener(new
132.     OnCompleteListener<AuthResult>() {
133.         @Override
134.         public void onComplete(@NonNull Task<AuthResult> task) {
135.             if (task.isSuccessful()) {
136.                 fail();
137.
138.             } else {
139.                 Assert.assertTrue(true);
140.             }
141.             test = true;
142.         }
143.     });
144.     while(!test){
145.
146.     }
147. }
148.
149. @Test
150. public void InvalidFormatPasswordTest(){
151.
152.     test = false;
153.     auth = FirebaseAuth.getInstance();
154.
155.     auth.signInWithEmailAndPassword("prova1@live.it", "Prova").addOnCompleteListener(new
156.     OnCompleteListener<AuthResult>() {
157.         @Override
158.         public void onComplete(@NonNull Task<AuthResult> task) {
159.             if (task.isSuccessful()) {
160.                 Assert.assertTrue(true);
161.
162.             } else {
163.                 Assert.assertFalse(false);
164.             }
165.             test = true;
166.         }
167.     });
168.     while(!test){
169.
170.     }
171. }
172.
173.
174. @Test
175. public void InvalidFormatEmailTest(){
176.
177.     test = false;
178.     auth = FirebaseAuth.getInstance();
179.
180.     auth.signInWithEmailAndPassword("prova1@live", "Prova3456").addOnCompleteListener(new
181.     OnCompleteListener<AuthResult>() {
182.         @Override
183.         public void onComplete(@NonNull Task<AuthResult> task) {
184.             if (task.isSuccessful()) {
185.                 fail();
186.
187.             } else {
188.
189.                 Assert.assertTrue(true);
190.
191.             }
192.             test = true;
193.         }
194.     });
195.     while(!test){
196.
197.     }
198. }
199. }
200. }
```

• Registrazione all'applicazione

```

1. package com.example.natour2022;
2.
3. import androidx.annotation.NonNull;
4. import androidx.test.ext.junit.runners.AndroidJUnit4;
5.
6. import com.example.natour2022.model.User;
```

```

7. import com.google.android.gms.tasks.OnCompleteListener;
8. import com.google.android.gms.tasks.Task;
9. import com.google.firebase.auth.FirebaseAuth;
10. import com.google.firebaseio.database.DatabaseReference;
11. import com.google.firebaseio.database.FirebaseDatabase;
12.
13. import org.junit.Assert;
14. import org.junit.Test;
15. import org.junit.runner.RunWith;
16.
17. @RunWith(AndroidJUnit4.class)
18. public class SingUp_Test {
19.
20.     FirebaseAuth auth;
21.     DatabaseReference referenceDb;
22.     boolean test;
23.
24.     @Test
25.     public void ValidSingUpTest() {
26.         test = false;
27.         auth = FirebaseAuth.getInstance();
28.         referenceDb = FirebaseDatabase.getInstance().getReference();
29.         String uid =
30.             FirebaseAuth.getInstance().getCurrentUser().getUid();
31.         User currUser = new User("PRpqGWkrosXoxubMfbLF3G0gYtF3", "Prova", "provalive.it", "Prova3456",
32.         "14/04/1996", "female");
33.         referenceDb.child("Users").child(uid).setValue(currUser).addOnCompleteListener(new
34.         OnCompleteListener<Void>() {
35.             @Override
36.             public void onComplete(@NonNull Task<Void> task) {
37.                 if (task.isSuccessful()) {
38.                     Assert.assertTrue(true);
39.                 } else {
40.                     Assert.assertFalse(false);
41.                 }
42.                 test = true;
43.             }
44.         });
45.         while (!test) {
46.     }
47.     }
48.
49.     @Test
50.     public void EmptySingInTest1() {
51.         test = false;
52.         auth = FirebaseAuth.getInstance();
53.         referenceDb = FirebaseDatabase.getInstance().getReference();
54.         String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
55.         User currUser = new User("PRpqGWkrosXoxubMfbLF3G0gYtF3", "", "", "", "", "");
56.         referenceDb.child("Users").child(uid).setValue(currUser).addOnCompleteListener(new
57.         OnCompleteListener<Void>() {
58.             @Override
59.             public void onComplete(@NonNull Task<Void> task) {
60.                 if (task.isSuccessful()) {
61.                     Assert.assertTrue(true);
62.                 } else {
63.                     Assert.assertFalse(false);
64.                 }
65.                 test = true;
66.             }
67.         });
68.         while (!test) {
69.     }
70.     }
71. }
72.
73.     @Test
74.     public void EmptyEmailTest() {
75.         test = false;
76.         auth = FirebaseAuth.getInstance();
77.         referenceDb = FirebaseDatabase.getInstance().getReference();
78.         String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
79.         User currUser = new User("PRpqGWkrosXoxubMfbLF3G0gYtF3", "Prova", "", "Prova3456", "14/04/1996",
80.         "female");
81.         referenceDb.child("Users").child(uid).setValue(currUser).addOnCompleteListener(new
82.         OnCompleteListener<Void>() {
83.             @Override
84.             public void onComplete(@NonNull Task<Void> task) {
85.                 if (task.isSuccessful()) {
86.                     Assert.assertTrue(true);
87.                 } else {
88.                     Assert.assertFalse(false);
89.                 }
90.                 test = true;
91.             }
92.         });
93.         while (!test) {
94.     }
95.     }
96. }
97.

```

```

98.     @Test
99.     public void EmptyPasswordTest() {
100.         test = false;
101.         auth = FirebaseAuth.getInstance();
102.         referenceDb = FirebaseDatabase.getInstance().getReference();
103.         String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
104.         User currUser = new User("PRpqGkrosXoxubMfbLF3G0gYtF3", "Prova", "prova1@live.it", "", "14/04/1996", "female");
105.         referenceDb.child("Users").child(uid).setValue(currUser).addOnCompleteListener(new
106.             OnCompleteListener<Void>() {
107.                 @Override
108.                 public void onComplete(@NonNull Task<Void> task) {
109.                     if (task.isSuccessful()) {
110.                         Assert.assertTrue(true);
111.                     } else {
112.                         Assert.assertFalse(false);
113.                     }
114.                     test = true;
115.                 }
116.             });
117.         while (!test) {
118.             }
119.         }
120.     }
121. }
122.
123. @Test
124. public void EmptyNicknameTest() {
125.     test = false;
126.     auth = FirebaseAuth.getInstance();
127.     referenceDb = FirebaseDatabase.getInstance().getReference();
128.     String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
129.     User currUser = new User("PRpqGkrosXoxubMfbLF3G0gYtF3", "", "prova1@live.it", "Prova3456", "14/04/1996", "female");
130.     referenceDb.child("Users").child(uid).setValue(currUser).addOnCompleteListener(new
131.         OnCompleteListener<Void>() {
132.             @Override
133.             public void onComplete(@NonNull Task<Void> task) {
134.                 if (task.isSuccessful()) {
135.                     Assert.assertTrue(true);
136.                 } else {
137.                     Assert.assertFalse(false);
138.                 }
139.                 test = true;
140.             }
141.         });
142.     while (!test) {
143.         }
144.     }
145. }
146.
147.
148. @Test
149. public void EmptyGenderTest() {
150.     test = false;
151.     auth = FirebaseAuth.getInstance();
152.     referenceDb = FirebaseDatabase.getInstance().getReference();
153.     String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
154.     User currUser = new User("PRpqGkrosXoxubMfbLF3G0gYtF3", "Prova", "prova1@live.it", "Prova3456", "14/04/1996", "");
155.     referenceDb.child("Users").child(uid).setValue(currUser).addOnCompleteListener(new
156.         OnCompleteListener<Void>() {
157.             @Override
158.             public void onComplete(@NonNull Task<Void> task) {
159.                 if (task.isSuccessful()) {
160.                     Assert.assertTrue(true);
161.                 } else {
162.                     Assert.assertFalse(false);
163.                 }
164.                 test = true;
165.             }
166.         });
167.     while (!test) {
168.         }
169.     }
170. }
171.
172.
173. @Test
174. public void EmptyDataOfBirthTest() {
175.     test = false;
176.     auth = FirebaseAuth.getInstance();
177.     referenceDb = FirebaseDatabase.getInstance().getReference();
178.     String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
179.     User currUser = new User("PRpqGkrosXoxubMfbLF3G0gYtF3", "Prova", "prova1@live.it", "Prova3456", "", "female");
180.     referenceDb.child("Users").child(uid).setValue(currUser).addOnCompleteListener(new
181.         OnCompleteListener<Void>() {
182.             @Override
183.             public void onComplete(@NonNull Task<Void> task) {
184.                 if (task.isSuccessful()) {
185.                     Assert.assertTrue(true);
186.                 } else {
187.                     }
188.             }

```

```
189.             Assert.assertFalse(false);
190.         }
191.     }
192. }
193. });
194. while (!test) {
195. }
196. }
197.
198. @Test
199. public void InvalidGenderTest() {
200.     test = false;
201.     auth = FirebaseAuth.getInstance();
202.     referenceDb = FirebaseDatabase.getInstance().getReference();
203.     String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
204.     User currUser = new User("PRpqGWkrosXoxubMfbLF3G0gYtF3", "Prova", "prova1@live.it", "Prova3456",
205.     "14/04/1996", "hgjhghj");
206.     referenceDb.child("Users").child(uid).setValue(currUser).addOnCompleteListener(new
207.     OnCompleteListener<Void>() {
208.         @Override
209.         public void onComplete(@NonNull Task<Void> task) {
210.             if (task.isSuccessful()) {
211.                 Assert.assertTrue(true);
212.             } else {
213.                 Assert.assertFalse(false);
214.             }
215.             test = true;
216.         }
217.     });
218. }
219. while (!test) {
220. }
221. }
222. }
```

Capitolo VII – Valutazione usabilità

○ Valutazione sull'usabilità sul prodotto finito

Per valutare l'usabilità sul prodotto sono stati presi in considerazione diversi valutatori, indicati precedentemente nel paragrafo “*Individuazione target utenti*” e si sono raccolte sintetiche valutazioni dopo l'utilizzo dell'applicazione:

- Elvira Giretti

Una valida alternativa per Android dedicate al trekking. NaTour22 è apprezzatissima perché permette di creare in modo intuitivo e originale nuovi percorsi che possono essere personalizzati con l'inserimento di tappe, a proprio piacimento, lungo un sentiero. In fine l'ultima cosa che preferisco è il design: semplice e intuitivo. Peccato che sia ancora in fase di sviluppo ma di sicuro in futuro la scaricherò.

- Antonio Laccio

Applicazione utilissima in fase di pianificazione dal design semplice e intuitivo. Comprendo che sia un'applicazione realizzata da poco ma la migliorerei non solo completando e migliorando le funzioni già create ma anche introducendo ad esempio nella creazione dell'itinerario la pendenza del percorso e aggiungendo una nuova sezione contenente tutti gli itinerari creati dall'utente.

- Alice Bassi

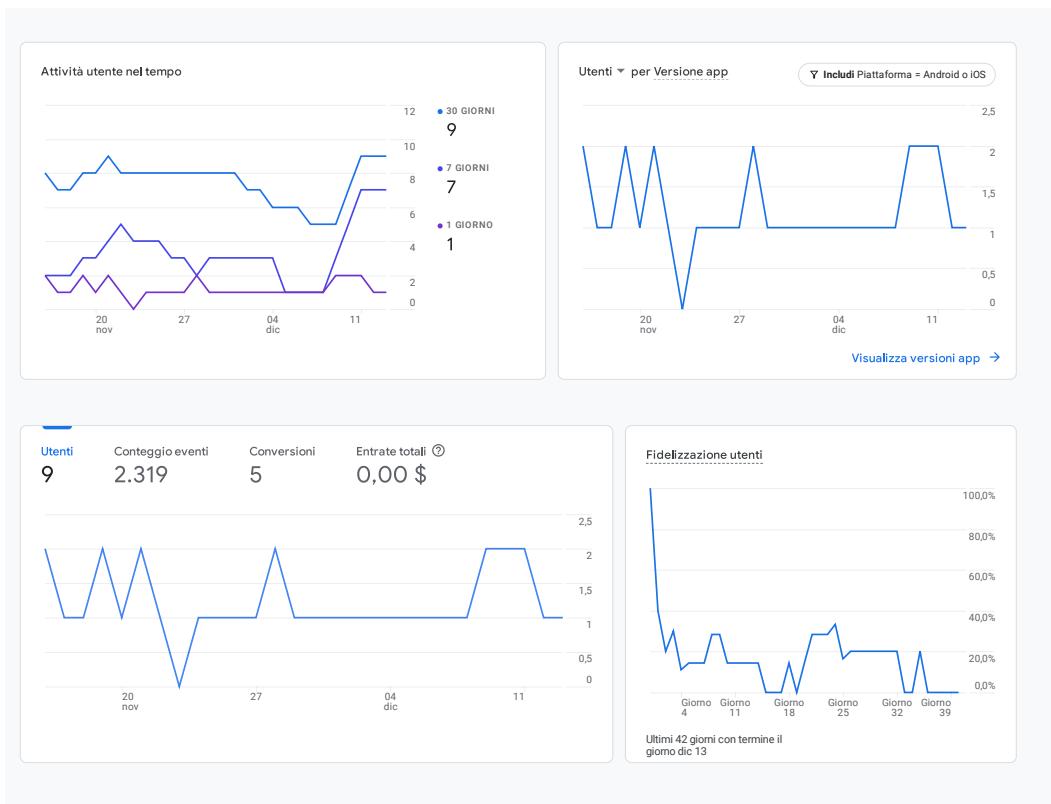
Applicazione dal design semplice per chiunque e molto colorata. Buona startup per chi voglia tracciare i propri itinerari e condividerli con la community, inutile dire che va migliorata, cosa ovvia in quanto è all'inizio.

- Marco Rossi

NaTour22, è un'applicazione che permette di creare itinerari in modo semplice e intuitivo. È disponibile un ampio database di percorsi, spesso caricati dagli stessi utenti, che si possono consultare. L'interfaccia di navigazione è molto intuitiva e permette con grande facilità di trovare l'itinerario giusto. Un'aspetto che aggiungerei è quello di realizzare una maschera di ricerca in modo tale che inserendo la tipologia di attività, la località, la difficoltà del percorso e altre informazioni utili permetta di individuare in fretta il percorso più giusto. Nonostante ciò, si tratta di un'app quindi fiducioso in un possibile miglioramento delle funzionalità offerte al momento.

Oltre a ciò si è cercato di valutare l'usabilità attraverso l'analisi mediante file di logging, utilizzando il framework Google Analytics for Mobile Apps che si occupa di fornire informazioni del prodotto software da parte degli utenti.

Vengono ora riportati alcuni estratti presi dalle statistiche di utilizzo ottenuti dalla console di Firebase:



Vengono ora riportati altri report ottenuti sempre dalla console di FireBase:

