

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 “NaTour21”**

**Gruppo:** INGSW2122\_V\_13

**Docente:**

**Autori:**

Prof. Sergio Di Martino

Giuseppe Di Luca N86002488

Carmine Di Martino N86002553



# Indice

---

PRESENTAZIONE DEL PROGETTO .....	5
<b>1 DOCUMENTO DEI REQUISITI SOFTWARE .....</b>	<b>7</b>
<b>    1.1 DEFINIZIONE DEI REQUISITI .....</b>	<b>7</b>
<b>    1.1.1 REQUISITI FUNZIONALI .....</b>	<b>7</b>
<b>    1.1.2 REQUISITI NON FUNZIONALI .....</b>	<b>10</b>
<b>    1.1.3 REQUISITI DI DOMINIO .....</b>	<b>12</b>
<b>    1.2 MODELLO FUNZIONALE .....</b>	<b>13</b>
<b>    1.2.1 MODELLAZIONE DEI CASI D'USO .....</b>	<b>13</b>
<b>    1.2.2 TABELLE DI COCKBURN .....</b>	<b>16</b>
<b>    1.2.3 MOCK-UP .....</b>	<b>25</b>
<b>    1.2.4 GLOSSARIO .....</b>	<b>38</b>
<b>    1.3 MODELLI DI DOMINIO .....</b>	<b>40</b>
<b>    1.3.1 CLASS DIAGRAMS DI ANALISI .....</b>	<b>40</b>
<b>    1.3.2 SEQUENCE DIAGRAMS DI ANALISI .....</b>	<b>56</b>
<b>    1.3.3 STATE CHART DIAGRAMS DI ANALISI .....</b>	<b>59</b>
<b>    1.3.4 ACTIVITY DIAGRAMS DI ANALISI .....</b>	<b>74</b>
<b>2 DOCUMENTO DI DESIGN DEL SISTEMA .....</b>	<b>89</b>
<b>    2.1 ANALISI DEL'LARCHITETTURA.....</b>	<b>89</b>

<b>2.2 CLASS DIAGRAM DI DESIGN .....</b>	<b>93</b>
<b>2.3 STATECHART DIAGRAM DI DESIGN .....</b>	<b>107</b>
<b>2.4 SEQUENCE DIAGRAMS DI DESIGN .....</b>	<b>108</b>
<b>3 DOCUMENTO DI TESTING DEL SISTEMA .....</b>	<b>111</b>
<b>3.1 TEST PLAN PER SYSTEM TESTING .....</b>	<b>111</b>
<b>3.2 UNIT TESTING .....</b>	<b>126</b>

# Presentazione del progetto

---

*NaTour21* è un sistema complesso e distribuito finalizzato ad offrire un moderno *social network* multi-piattaforma per appassionati di escursioni.

Il sistema consiste in un back-end sicuro, performante e scalabile, e in un client mobile attraverso cui gli utenti possono fruire delle funzionalità del sistema in modo intuitivo, rapido e piacevole.

*NaTour21* permette agli utenti di connettersi ai propri amici appassionati di escursioni, di scoprire nuovi sentieri da visitare ed avere molte informazioni su quest'ultimi.

Le principali funzionalità offerte da *NaTour21* sono indicate di seguito:

- 1** Un utente può registrarsi o autenticarsi attraverso le credenziali richieste oppure utilizzando account su altre piattaforme come Google o Facebook.
- 2** Un utente autenticato può inserire nuovi itinerari (sentieri) in piattaforma. Un sentiero è caratterizzato da un nome, una durata, un livello di difficoltà, un punto di inizio, una descrizione (opzionale), una foto (opzionale) e un tracciato geografico (opzionale) che lo rappresenta su una mappa. Il tracciato geografico deve essere inseribile manualmente (interagendo con una mappa interattiva) oppure tramite file in formato standard GPX.
- 3** Un utente autenticato può inserire un tracciato geografico per un sentiero anche registrando una sequenza di posizioni GPS con il proprio dispositivo mobile, all'interno dell'app NaTour.
- 4** Effettuare ricerche di itinerari tra quelli presenti in piattaforma, con possibilità di filtrare i risultati per area geografica, per livello di difficoltà, per durata, e per accessibilità a disabili.
- 5** Visualizzare una schermata di dettaglio per ciascun sentiero. Questa schermata mostra tutte le informazioni note del sentiero, e visualizza su una mappa interattiva, il punto di inizio e il tracciato geografico, se disponibile.
- 6** Un utente autenticato può creare compilation di sentieri personalizzate, caratterizzate anche da un titolo e da una descrizione personalizzata.

- 7** Un utente può scaricare le informazioni riguardo un sentiero in formato GPX, e stampare le informazioni riepilogative in formato PDF.
- 8** Qualora lo ritenga necessario, un utente può anche indicare un punteggio di difficoltà e/o un tempo di percorrenza diverso da quello indicato dall'utente che ha inserito il sentiero. In questo caso, il punteggio di difficoltà e il tempo di percorrenza per il sentiero saranno ri-calcolati come la media delle difficoltà / dei tempi indicati.

# CAPITOLO I

## DOCUMENTO DEI REQUISITI SOFTWARE

---

### 1.1 Definizione Requisiti

#### 1.1.1 Requisiti Funzionali

Vengono riportati i requisiti funzionali ovvero tutti i servizi (o funzioni) che il sistema deve offrire.

Nome	Descrizione
Registrazione alla piattaforma	Il sistema deve consentire ad un utente non registrato di potersi registrare alla piattaforma indicando: e-mail, nome, cognome, nickname e password.
Accesso alla piattaforma	Il sistema deve consentire ad un utente registrato di poter effettuare l'accesso alla piattaforma indicando e-mail e password oppure utilizzando account su altre piattaforme come Google o Facebook.
Recupero dell'account	Il sistema deve consentire ad un utente registrato di poter recuperare il proprio account indicando l'e-mail utilizzata in fase di registrazione.

Salvataggio delle informazioni sul sentiero in formato GPX	Il sistema deve consentire ad un utente di scaricare le informazioni di un itinerario, in formato GPX, presente in piattaforma.
Inserimento di nuovi itinerari	Il sistema deve consentire ad un utente autenticato di inserire un nuovo itinerario in piattaforma indicando: un nome, una durata, un livello di difficoltà, un punto di inizio, una descrizione (opzionale), una foto (opzionale) e un tracciato geografico (opzionale) che lo rappresenta su una mappa.
Modifica difficoltà di un itinerario	Il sistema deve consentire ad un utente autenticato di modificare la difficoltà di un itinerario presente in piattaforma.
Creazione di una nuova compilation	Il sistema deve consentire ad un utente autenticato di creare una nuova compilation personalizzata indicando il titolo e la descrizione di quest'ultima.
Ricerche itinerari	Il sistema deve consentire ad un utente (anche non registrato) di effettuare una ricerca degli itinerari presenti nella piattaforma.
Filtraggio ricerche itinerari	Il sistema deve consentire ad un utente (anche non registrato) di filtrare i risultati delle ricerche degli itinerari. Il filtraggio può avvenire mediante: <ul style="list-style-type: none"><li>- area geografica.</li><li>- livello di difficoltà.</li><li>- durata.</li><li>- accessibilità a disabili.</li></ul>

Visualizzazione schermata di dettaglio	Il sistema deve consentire ad un utente (anche non registrato) di visualizzare una schermata di dettaglio per ciascun sentiero. Questa schermata mostra tutte le informazioni note del sentiero, e visualizza su una mappa, il punto di inizio e il tracciato geografico, se disponibile. Inoltre, la schermata di dettaglio mostra le eventuali recensioni degli utenti e fotografie caricate.
Modifica della password	Il sistema deve consentire ad un utente registrato di poter effettuare il cambio della password, indicando la vecchia password per impostare la nuova.
Modifica tempo di percorrenza di un itinerario	Il sistema deve consentire ad un utente autenticato di modificare il tempo di percorrenza di un itinerario presente in piattaforma.
Modifica del nickname	Il sistema deve consentire ad un utente registrato di poter effettuare il cambio del nickname.
Visualizzazione del profilo	Il sistema deve consentire ad un utente autenticato di poter visualizzare il proprio profilo e quello di chi desidera.
Stampa delle informazioni dell’itinerario in formato pdf	Il sistema deve consentire ad un utente di poter scaricare e stampare le informazioni riguardanti un itinerario presente in piattaforma in formato pdf.

## 1.1.2 Requisiti Non Funzionali

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

Nome	Descrizione
Usabilità dell'applicazione	Un utente deve riuscire ad utilizzare la piattaforma sfruttando tutte le funzionalità offerte da essa dopo una media di 3 ore di utilizzo.
Blocco profilo spam	Al termine della registrazione il sistema deve provvedere ad inviare una e-mail di verifica per l'attivazione dell'account.
Sicurezza password	Durante la registrazione il sistema deve imporre all'utente l'inserimento di una password di almeno 8 caratteri alfanumerici.
Scalabilità del back-end	Il sistema deve adattarsi ad eventuali variazioni dei carichi di lavoro.
Ricerca performante	Il sistema deve garantire che i risultati di una ricerca siano mostrati entro 3 secondi dall'avvio di quest'ultima, nel 90% dei casi.
Limitazione inserimento tracciato	Il sistema deve obbligare ad ogni utente di poter inserire il tracciato usando una sola delle tre opzioni che ha a disposizione.

Limitazione inserimento itinerario	Il sistema deve impedire ad ogni utente di poter inserire più di una volta lo stesso itinerario.
Limitazione dell'accesso ai dati	Il sistema non deve avere accesso diretto alla base di dati.
Adattabilità al back-end	Il sistema deve essere adattabile ai possibili cambiamenti del back-end e quindi essere completamente disaccoppiato rispetto a quest'ultimo.

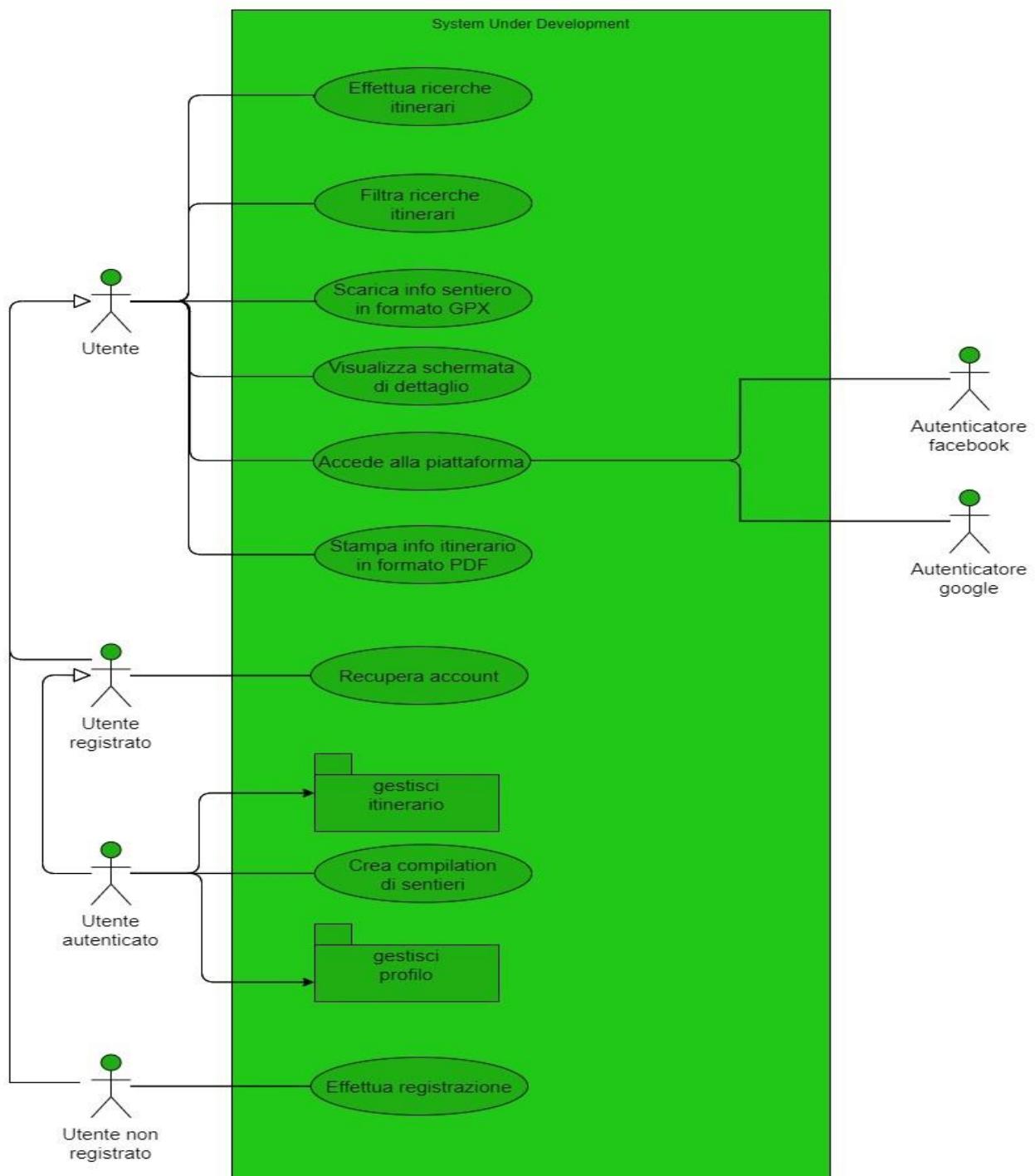
### **1.1.3 Requisiti Di Dominio**

Vengono riportati i requisiti di dominio ovvero tutti i vincoli che vengono dal dominio (contesto) nel quale opera il sistema e a cui l'applicativo deve attenersi.

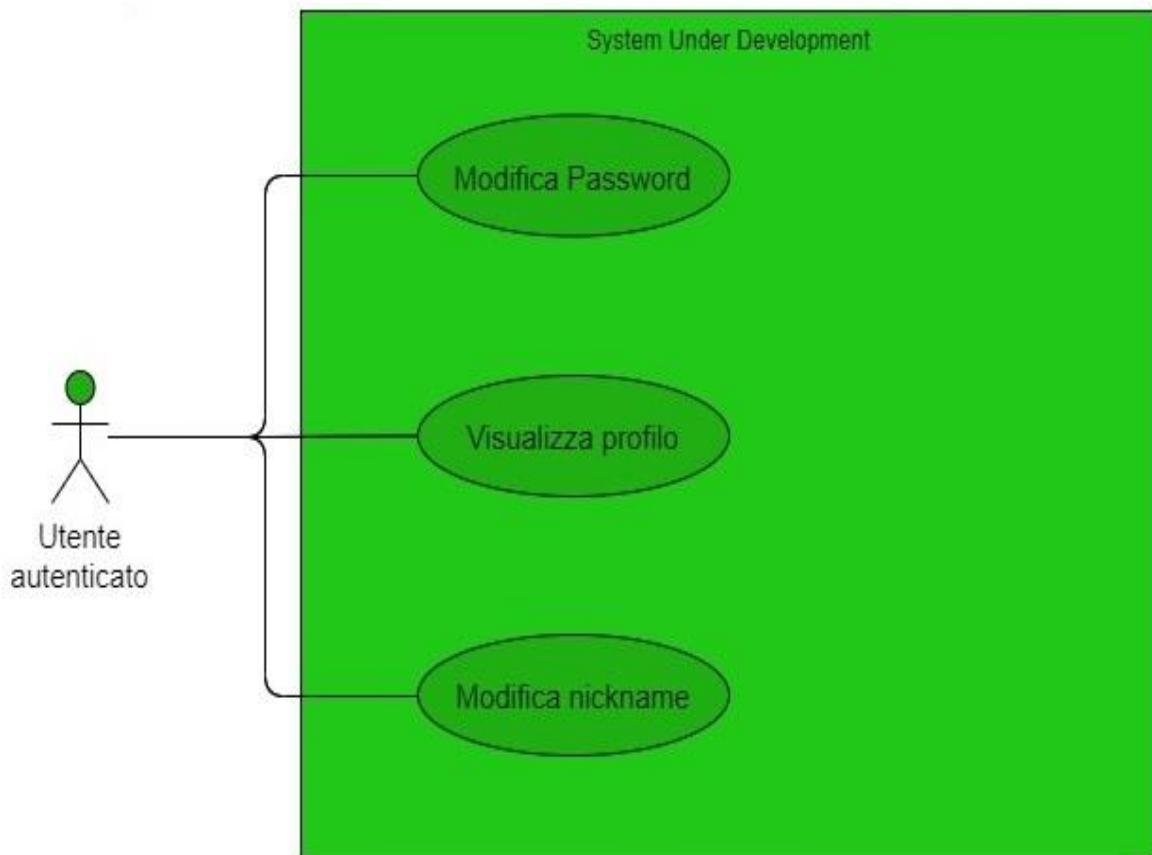
<b>Nome</b>	<b>Descrizione</b>
GDPR	Il sistema deve essere conforme al Regolamento Generale per la Protezione dei Dati (GDPR), in ottemperanza alle regole EU sui dati personali e sulla privacy dell'utente.

## 1.2 Modello Funzionale

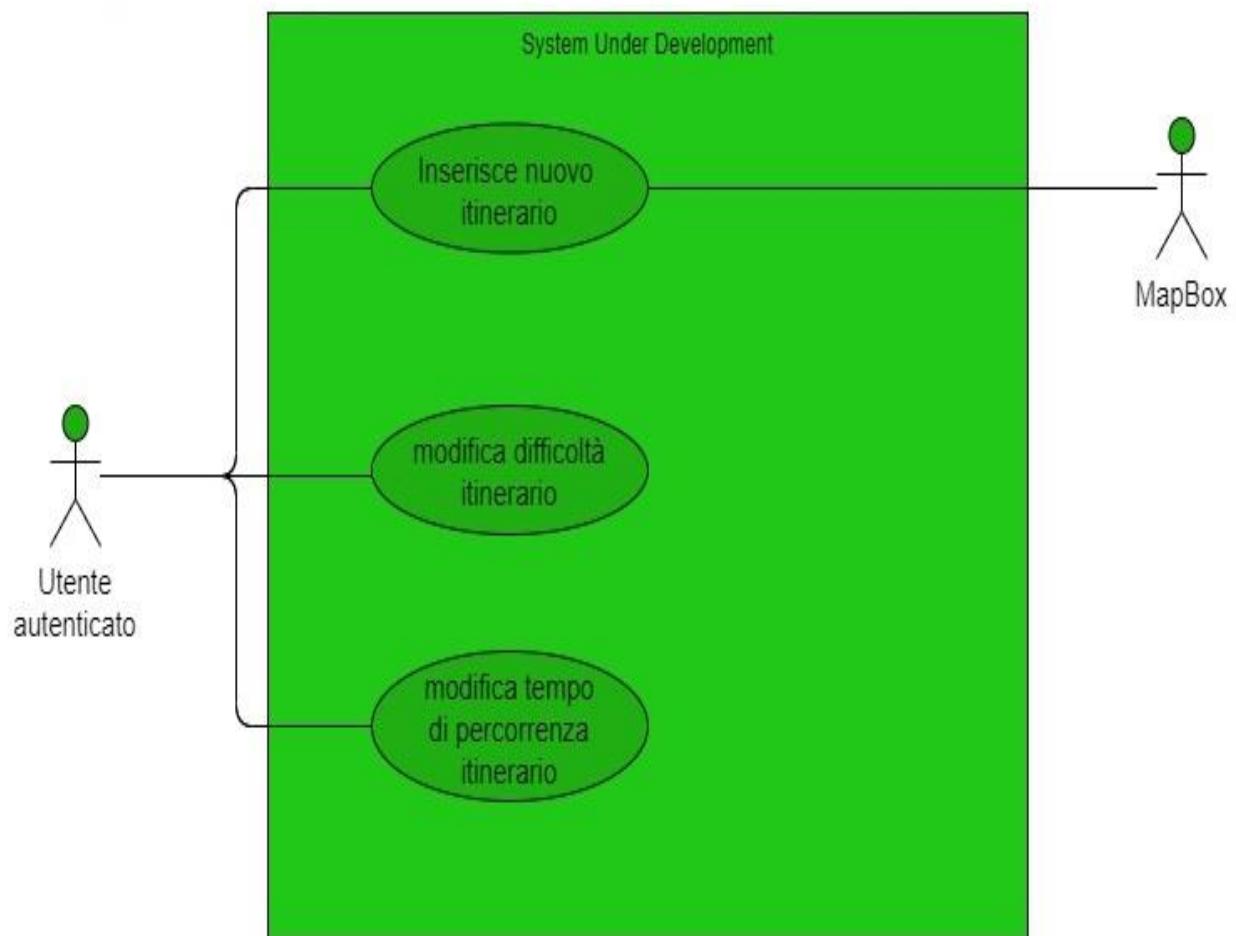
### 1.2.1 Modellazione dei casi d'uso



## Utente registrato gestisce profilo



## Utente autenticato gestisce itinerari



## 1.2.2 Tabelle di Cockburn

In questa sezione vengono elencate le tabelle di Cockburn relative ai casi d'uso dello Use Case Diagram riportato nel paragrafo precedente.

USE CASE #1	Effettua ricerche itinerari		
<b>Goal in Context</b>	L'utente effettua ricerche degli itinerari presenti nella piattaforma.		
<b>Preconditions</b>	L'utente si trova nella home dove è presente la barra di ricerca.		
<b>Succes End Condition</b>	L'utente ha trovato l'itinerario ricercato.		
<b>Failed End Condition</b>	1. L'itinerario non è presente nel database.		
<b>Primary Actor</b>	Utente.		
<b>Trigger</b>	L'utente clicca sulla barra di ricerca.		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>
<b>EXTENSION #1:</b> L'itinerario non è presente nel database.	1	Clicca sulla barra di ricerca in <b>M1</b> e digita il nome dell'itinerario.	
	2		Mostra <b>M3</b> con la lista di itinerari ottenuta.
2.1			Mostra <b>M2</b> con la scritta “Non è stato trovato alcun itinerario”.

USE CASE #2	Filtro ricerche itinerari		
<b>Goal in Context</b>	L'utente filtra le ricerche degli itinerari presenti nella piattaforma.		
<b>Preconditions</b>	L'utente si trova nella home dove è presente la barra di ricerca ed il pulsante filtra.		
<b>Succes End Condition</b>	L'utente riesce a filtrare gli itinerari.		
<b>Failed End Condition</b>	1. I filtri inseriti non hanno prodotto alcun risultato.		
<b>Primary Actor</b>	Utente.		
<b>Trigger</b>	L'utente clicca sul pulsante filtra presente nella home.		
<b>DESCRIPTION</b>	Step n°	Utente	Sistema
	1	Clicca sul pulsante filtra in <b>M1</b> .	
	2		Mostra <b>M4</b> con le opzioni di filtraggio.
	3	Inserisce i filtri desiderati e preme il pulsante “Applica filtri”.	
	4		Mostra <b>M5</b> con la lista degli itinerari filtrati.
<b>EXTENSION #1:</b> I filtri inseriti non hanno prodotto alcun risultato.	4.1		Mostra <b>M2</b> con la scritta “Non è stato trovato alcun itinerario”.
<b>EXTENSION #2:</b> L'utente cancella i filtri inseriti.	3.2	Inserisce i filtri desiderati e preme il pulsante “Cancella filtri”.	
	4.2		Mostra <b>M4</b> con le opzioni di filtraggio.

<b>EXTENSION #3:</b> L'utente esce dalla schermata dei filtri.	3.3	Preme il pulsante “Torna indietro” in alto a sinistra, rappresentato da una freccia con direzione verso sinistra.	
	4.3		Mostra <b>M1</b> .

<b>USE CASE #3</b>	<b>Inserisce nuovo itinerario</b>		
<b>Goal in Context</b>	L'utente inserisce un nuovo itinerario nella piattaforma.		
<b>Preconditions</b>	L'utente che desidera inserire un itinerario deve essere loggato alla piattaforma.		
<b>Succes End Condition</b>	L'utente ha inserito l'itinerario nella piattaforma.		
<b>Failed End Condition</b>	1. L'utente non ha compilato tutti i campi obbligatori.		
<b>Primary Actor</b>	Utente autenticato.		
<b>Trigger</b>	L'utente clicca sul pulsante inserisci itinerario.		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>
	1	Clicca sull'icona in alto a sinistra rappresentato da tre linee orizzontali in <b>M1</b> .	
	2		Mostra <b>M6</b> .
	3	Clicca sul pulsante “Inserisci itinerario”.	
	4		Mostra <b>M7</b> .
	5	Inserisce nome itinerario, tempo totale, difficoltà, punto di inizio e preme il pulsante “Salva itinerario”.	
	6		Mostra <b>M19</b> con la scritta “Itinerario salvato con successo.”
	7	Preme sul pulsante “ok”	
	8		Mostra <b>M1</b> .

<b>EXTENSION #1:</b> L'utente non ha compilato tutti i campi obbligatori.	5.1	Non compila tutti i campi obbligatori e preme il pulsante “Salva itinerario”.	
	6.1		Mostra <b>M18</b> con la scritta “Assicurati di aver inserito tutti i campi obbligatori (indicati con un asterisco).”
	7.1	Preme sul pulsante “ok”.	
	8.1		Mostra <b>M7</b> .
<b>EXTENSION #2:</b> L'utente esce dalla schermata inserisci itinerario.	5.2	Preme il pulsante “Torna indietro” in alto a sinistra, rappresentato da una freccia con direzione verso sinistra.	
	6.2		Mostra <b>M17</b> .
	7.2	Preme sul pulsante “Conferma”.	
	8.2		Mostra <b>M1</b> .

<b>EXTENSION #3:</b> L'utente inserisce anche il campo non obbligatorio “inserisci tracciato” tramite un file GPX.	5.3	Preme sul pulsante “Inserisci tracciato”.	
	6.3		Mostra <b>M9</b> .
	7.3	Preme sul pulsante “Inserisci file GPX”.	
	8.3		Mostra <b>M10</b> .
	9.3	Preme sul pulsante “Seleziona un file dal dispositivo”. Preme sul pulsante “Seleziona”.	
	10.3		Mostra <b>M15</b> con la scritta “Tracciato inserito correttamente”.
	11.3	preme sul pulsante “Salva itinerario”.	
	12.3		Mostra <b>M19</b> con la scritta “Itinerario salvato con successo.”
	13.3	Preme sul pulsante “ok”	
	14.3		Mostra <b>M1</b> .

<b>EXTENSION #4:</b> L'utente inserisce anche il campo non obbligatorio “inserisci tracciato” Attraverso l'interazione con la mappa.	5.4	Preme sul pulsante “Inserisci tracciato”.	
	6.4		Mostra <b>M9.</b>
	7.4	Preme sull'icona, a forma di grafo, in alto sulla mappa.	
	8.4		Mostra <b>M24.</b>
	9.4	Sceglie il tipo di percorso (per auto, per bici, per pedoni) e lo traccia.	
	10.4		Mostra <b>M23.</b>
	11.4	Preme sul pulsante “Salva”.	
	12.4		Mostra <b>M15</b> con la scritta “Tracciato inserito correttamente”.
	13.4	preme sul pulsante “Salva itinerario”.	
	14.4		Mostra <b>M19</b> con la scritta “Itinerario salvato con successo.”
	15.4	Preme sul pulsante “ok”	
	16.4		Mostra <b>M1.</b>

<b>EXTENSION #5:</b> L'utente inserisce anche il campo non obbligatorio “inserisci tracciato” Attraverso l'opzione inizia a registrare.	5.5	Preme sul pulsante “Inserisci tracciato”.	
	6.5		Mostra <b>M9</b> .
	7.5	Preme sul pulsante “Inizia a registrare”.	
	8.5		Mostra <b>M11</b> .
	9.5	Preme sul pulsante in basso a sinistra indicato da un quadratino rosso.	
	10.5		Mostra <b>M12</b> .
	11.5	Preme sul pulsante “Termina”.	
	12.5		Mostra <b>M13</b> .
	13.5	Preme sul pulsante “Termina”.	
	14.5		Mostra <b>M14</b> .
	15.5	Preme sul pulsante “Salva itinerario”.	
	16.5		Mostra <b>M15</b> con la scritta “Tracciato inserito correttamente”.
	17.5	preme sul pulsante “Salva itinerario”.	

	18.5		Mostra <b>M19</b> con la scritta “Itinerario salvato con successo.”
	19.5	Preme sul pulsante “ok”	
	20.5		Mostra <b>M1.</b>
<b>EXTENSION #6:</b> L’utente inserisce anche il campo non obbligatorio “inserisci immagine” .	5.6	Preme sul pulsante “Inserisci Immagine”.	
	6.6		Mostra <b>M21.</b>
	7.6	Preme sul pulsante “Selezione un file dal dispositivo”. Preme sul pulsante seleziona.	
	8.6		Mostra <b>M25</b> con la scritta “Immagine inserita correttamente”.
	9.6	Preme sul pulsante “Salva itinerario”.	
	10.6		Mostra <b>M19</b> con la scritta “Itinerario salvato con successo.”
	11.6	Preme sul pulsante “ok”	
	12.6		Mostra <b>M1.</b>

### 1.2.3 Mock-up

In questa sezione vengono mostrati i Mock-up dell'applicazione

M1



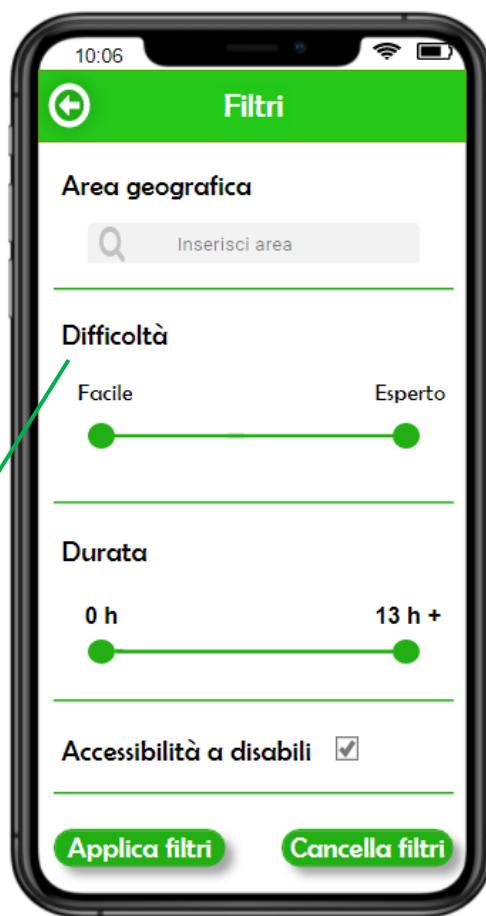
M2



M3



M4



Il filtro "difficolta" può assumere i seguenti 4 valori:

Facile, intermedio, difficile ed esperto.

**M5**



**M6**



M7

10:10

NaTour21

Nome Itinerario \*

Tempo totale \*

Ore Minuti

0 0

Difficoltà \*

Facile  Intermedio

Difficile  Estremo

Punto di inizio \*

Inserisci luogo

I campi segnati con un asterisco, devono essere compilati obbligatoriamente.

M8

10:10

NaTour21

Accessibilità a disabili \*

Descrizione

0 / 200

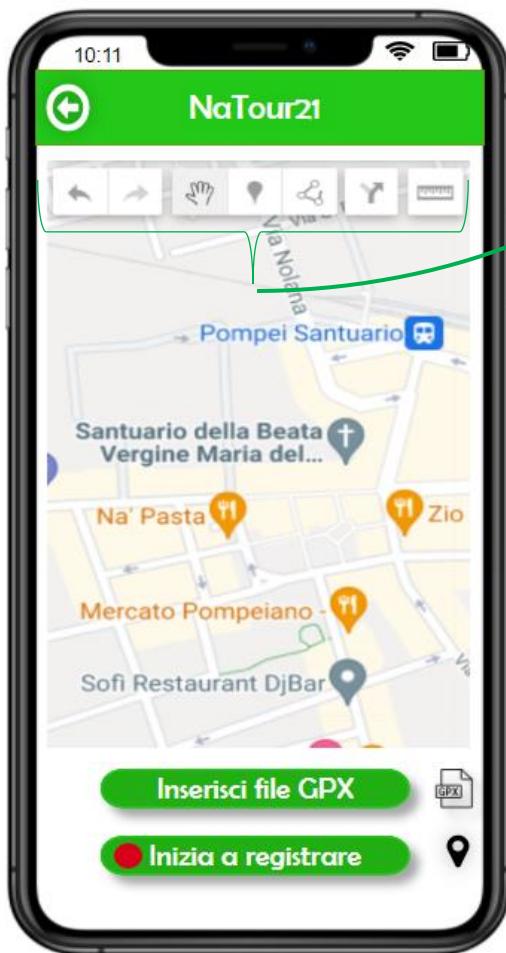
Inserisci tracciato

Inserisci Immagine

Salva itinerario

Il range delle ore va da 0 a 13+.  
Nel caso in cui venga selezionata l'opzione 13+, i minuti diventeranno non selezionabili.

M9

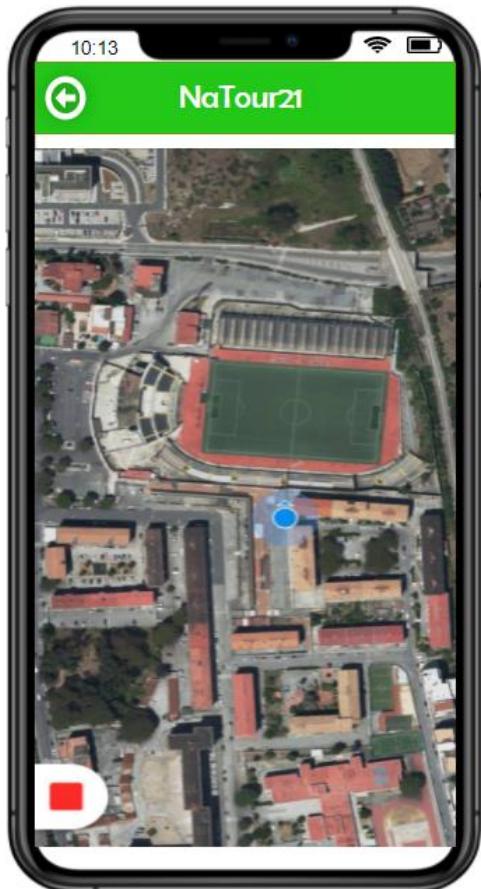


Se l'utente decide di inserire il tracciato manualmente, i pulsanti “Inserisci file GPX” e “Inizia a registrare” diventano non selezionabili.

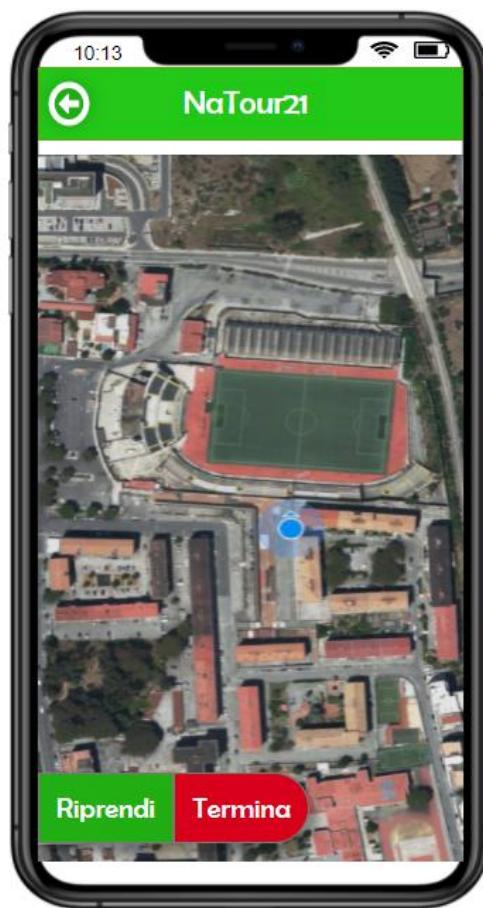
M10



M11



M12



M13



M14



**M15**



**M16**



M17



M18



**M19**



**M20**



**M21**



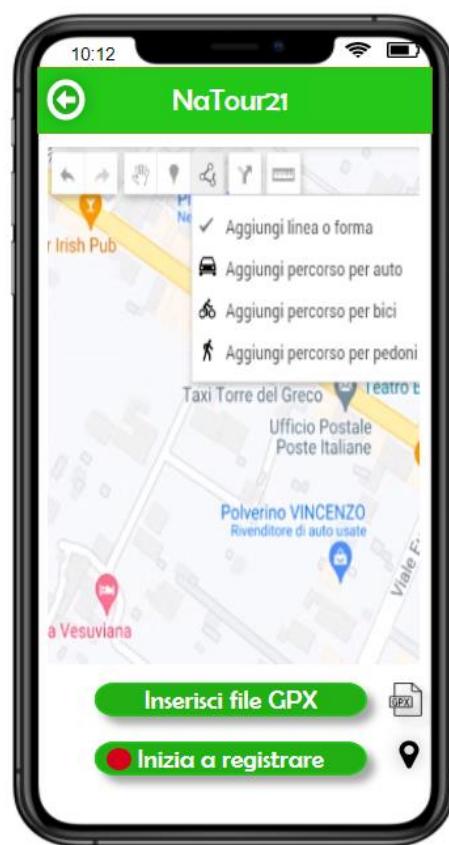
**M22**



M23



M24



## M25



## 1.2.4 Glossario

	<b>Termine</b>	<b>Descrizione</b>
<b>1</b>	GPX	Un file GPX è un formato che permette la memorizzazione e l'elaborazione di dati GPS (geodati).
<b>2</b>	Utente	Con utente si intende in modo indifferente sia un utente non registrato che un utente registrato.
<b>3</b>	GPS	Il GPS è un sistema di posizionamento satellitare che permette in ogni istante di conoscere la longitudine e la latitudine di un oggetto.
<b>4</b>	GDPR	Il regolamento generale sulla protezione dei dati è un regolamento dell'unione europea in materia di trattamento dei dati personali e di privacy dell'UE.

<b>5</b>	Scalabilità	La scalabilità denota in genere la capacità di un sistema di aumentare o diminuire di <i>scala</i> in funzione delle necessità e disponibilità.
<b>6</b>	Profilo spam	Profili che non rispecchiano la persona che si cela dietro.
<b>7</b>	Difficoltà itinerario	La scala dei gradi di difficoltà per itinerari escursionistici tiene conto di 4 parametri oggettivi: dislivello, distanza planimetrica, qualità della segnaletica sul percorso ed in base a questi parametri, la difficoltà può assumere i seguenti 4 valori: Facile, intermedio, difficile ed esperto.
<b>8</b>	Usabilità	l'efficacia, l'efficienza e la soddisfazione con le quali determinati utenti raggiungono determinati obiettivi in determinati contesti. In pratica definisce il grado di facilità e soddisfazione con cui si compie l'interazione tra l'uomo e uno strumento.

## **1.3 Modelli di dominio**

### **1.3.1 Class Diagrams di analisi**

In questa sezione vengono elencati i class diagrams relativi ai casi d'uso dello Use Case Diagram riportato nel paragrafo precedente.

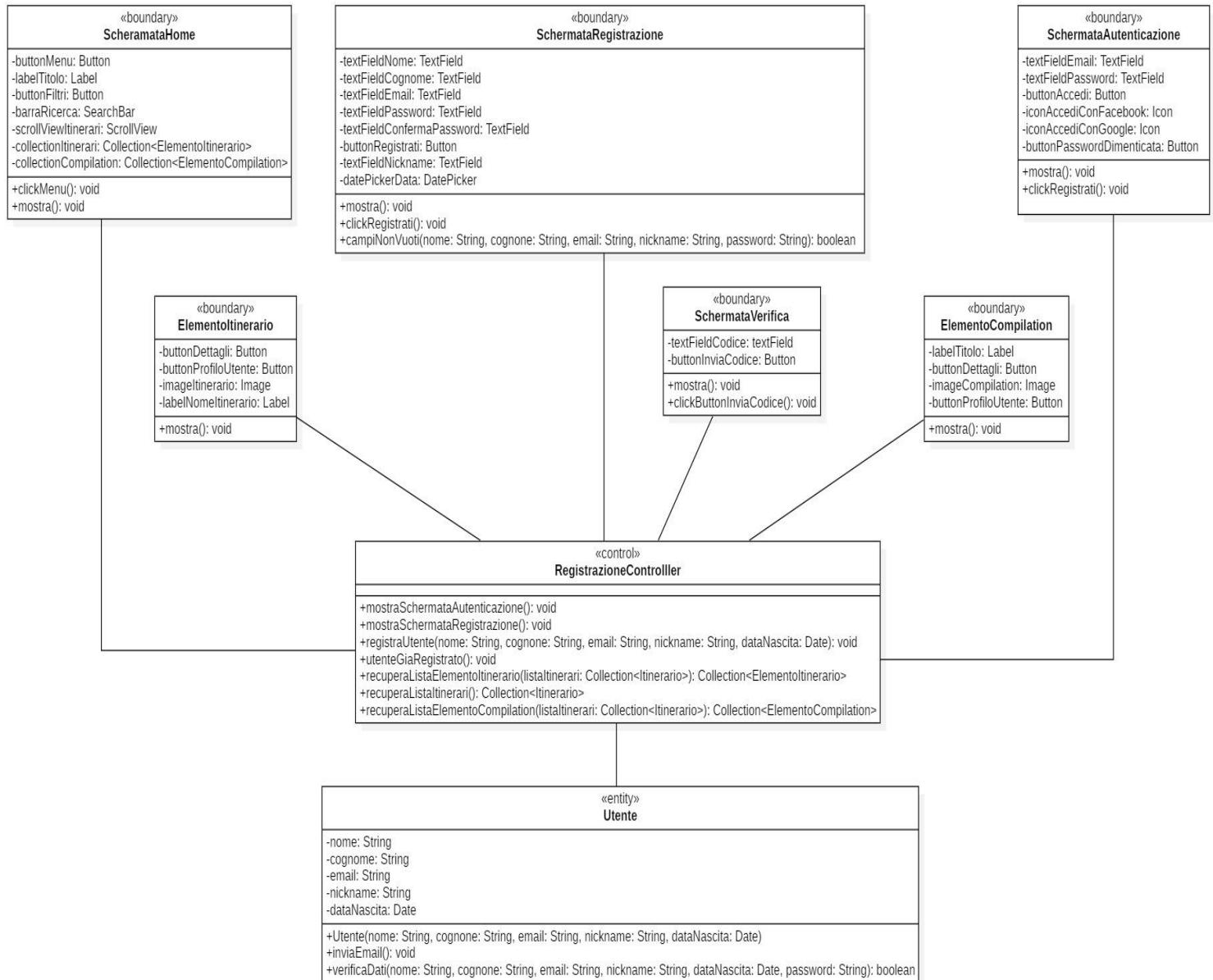
Per motivi di manutenzione e scalabilità si è deciso di utilizzare l'euristica del Three Object Type.

Questa linea guida propone di provare a raggruppare gli oggetti candidati in 3 macro classi:

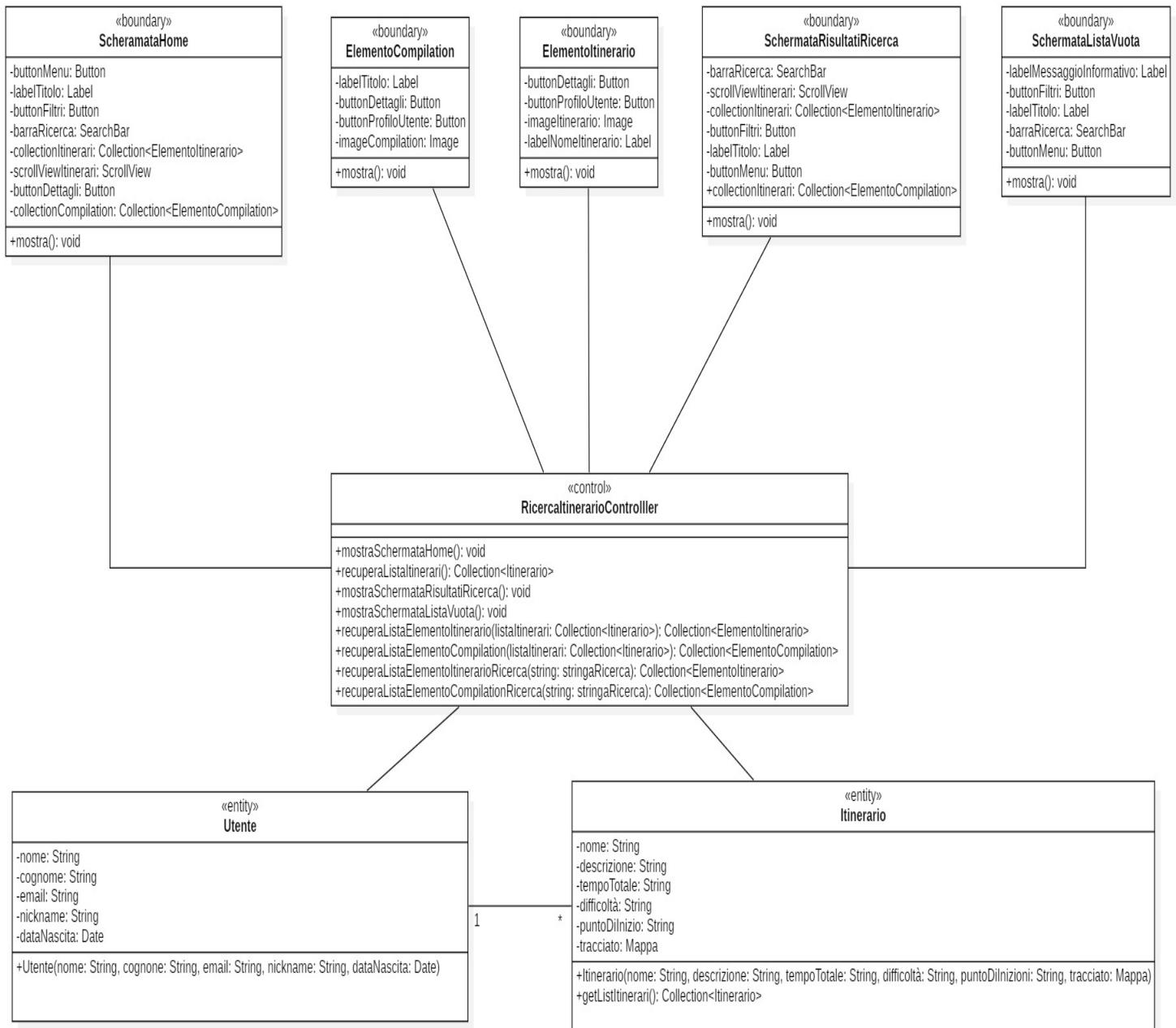
- Classi che si occupano dell'interfaccia utente.
- Classi che si occupano della logica del funzionamento.
- Classi che si occupano della persistenza dei dati.

La linea guida seguita semplifica enormemente la manutenzione nel corso degli anni successivi, quindi si dimostra un grande investimento.

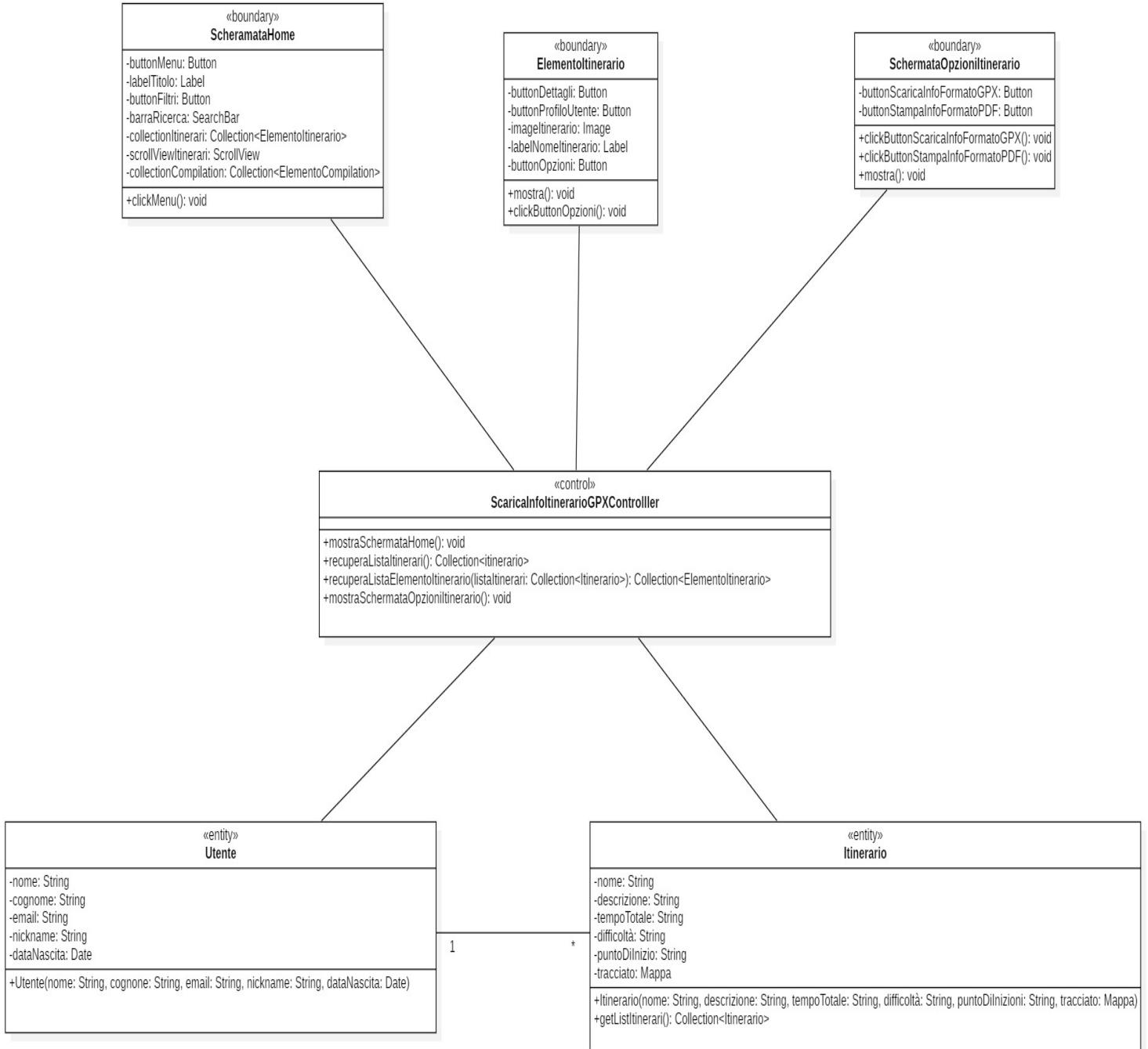
## Utente non registrato effettua registrazione



## Utente effettua ricerche itinerari



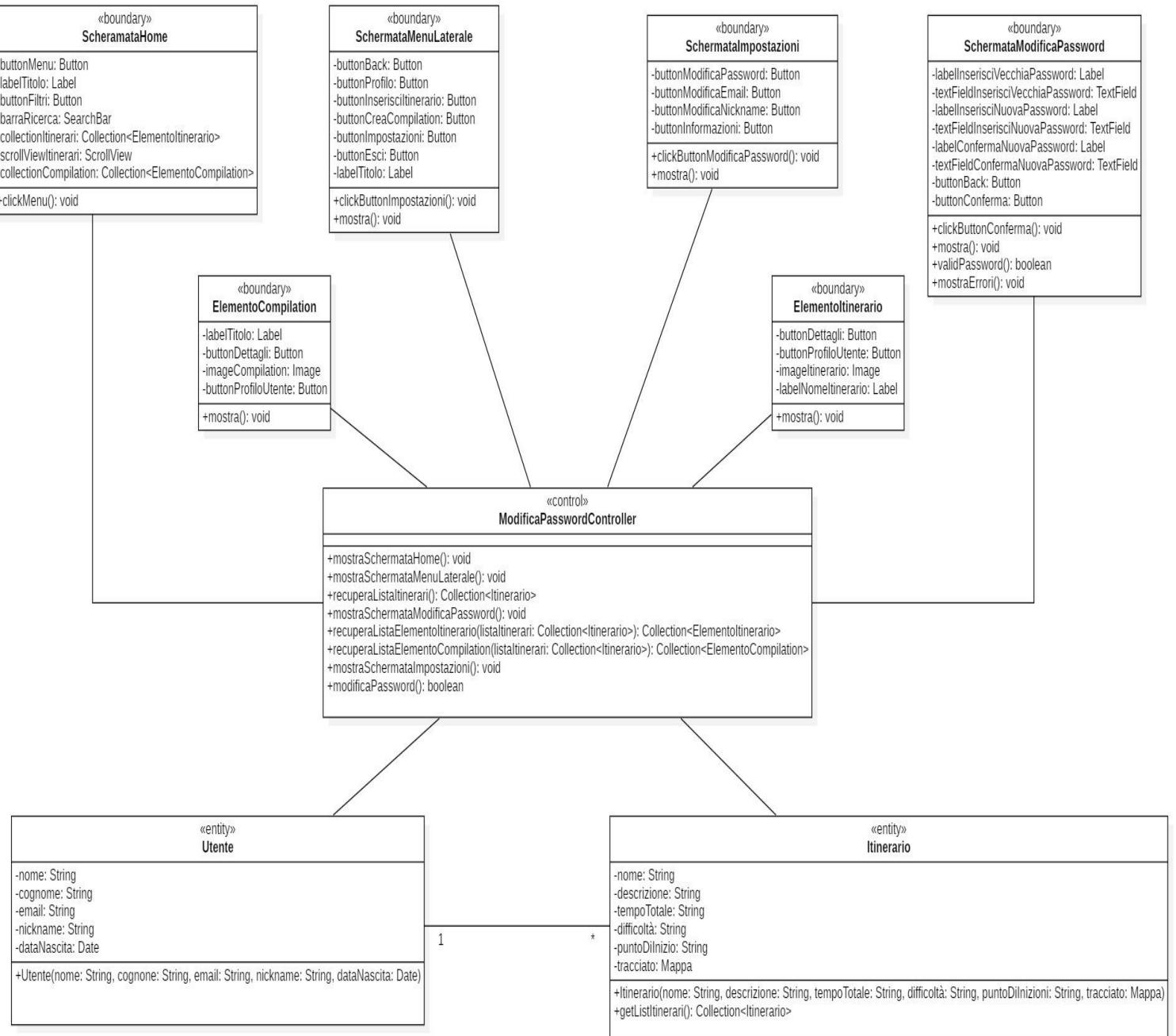
## Utente Scarica Informazioni Itinerario in formato GPX



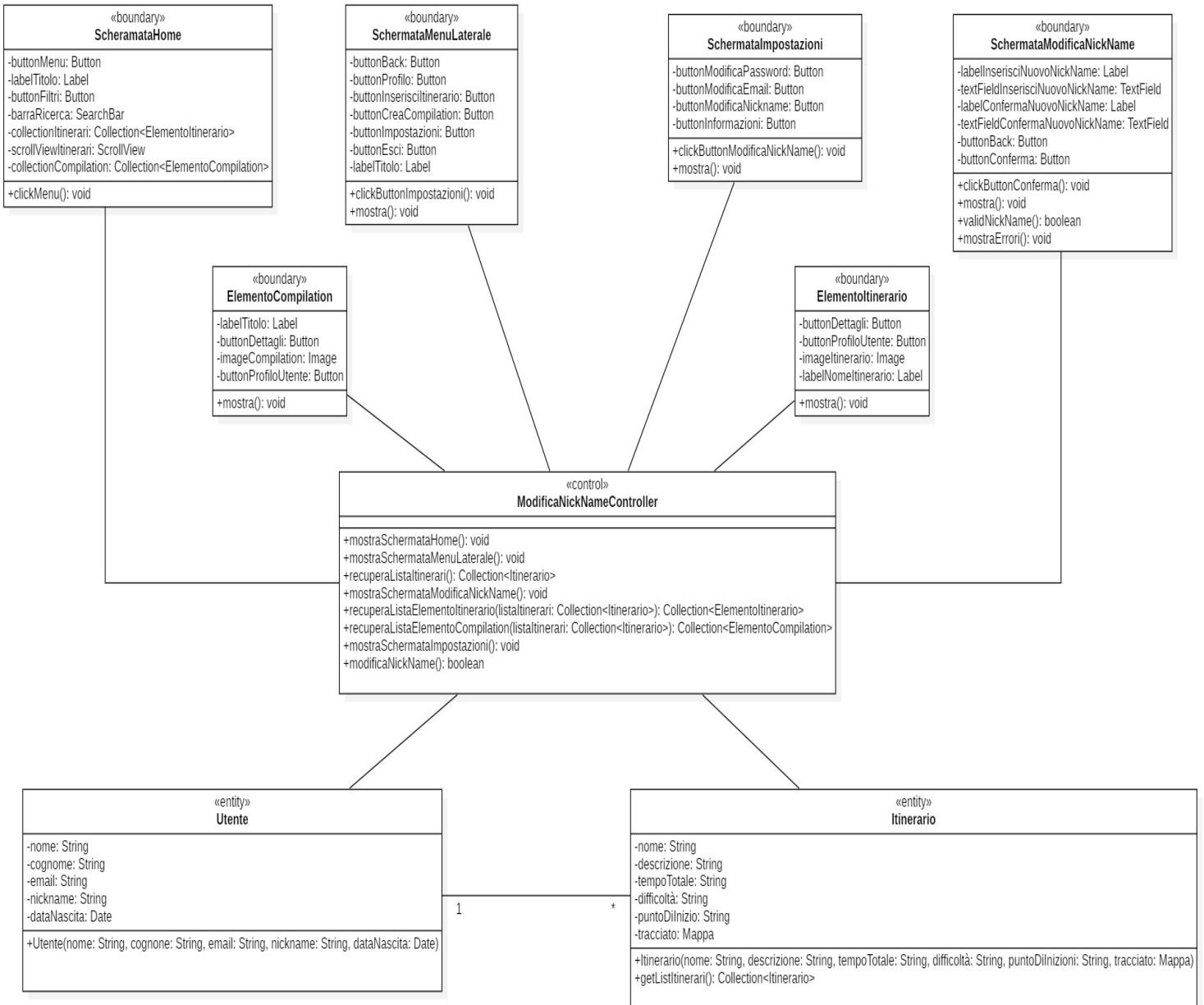
## Utente registrato recupera account



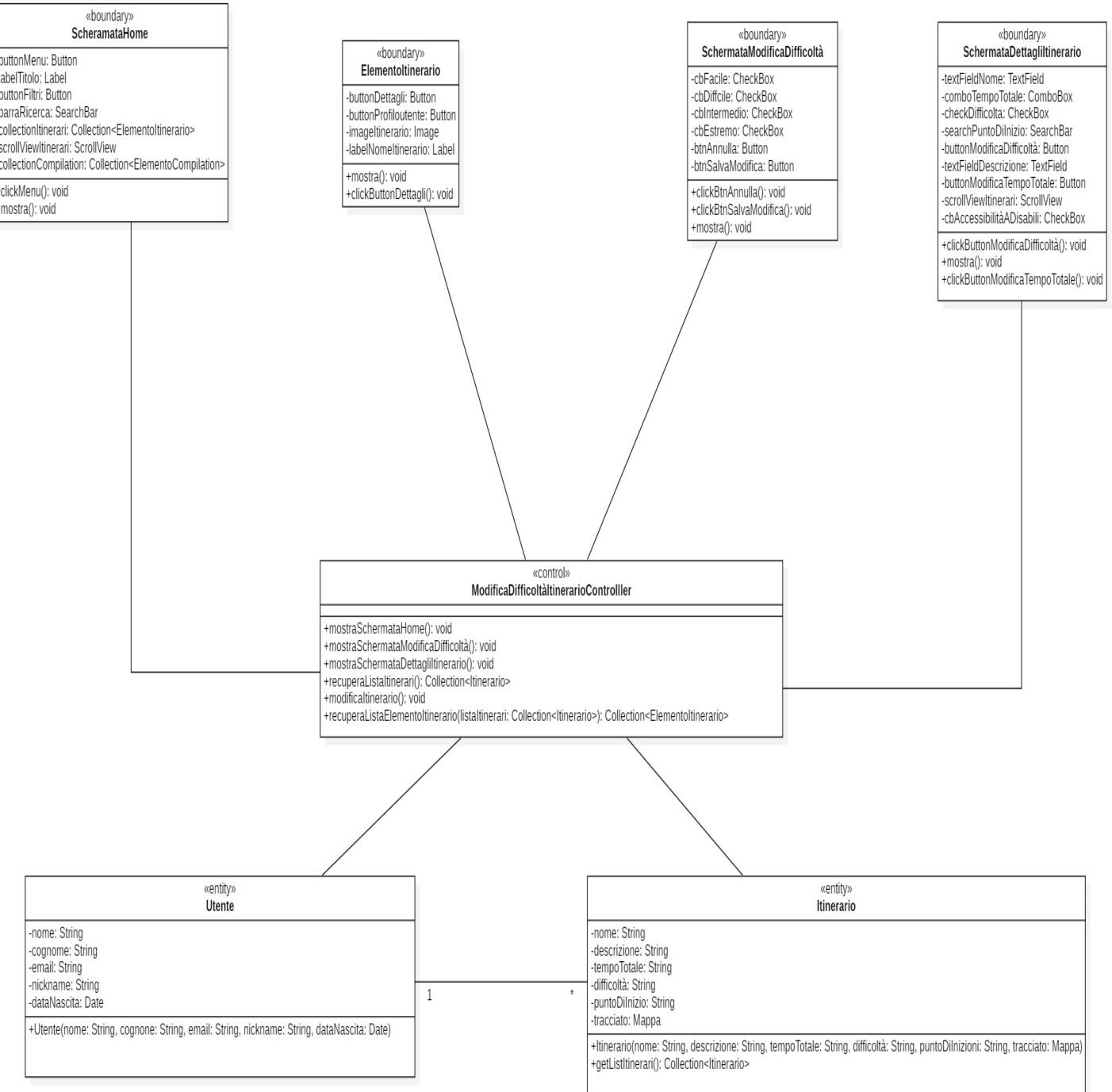
## Utente registrato modifica password



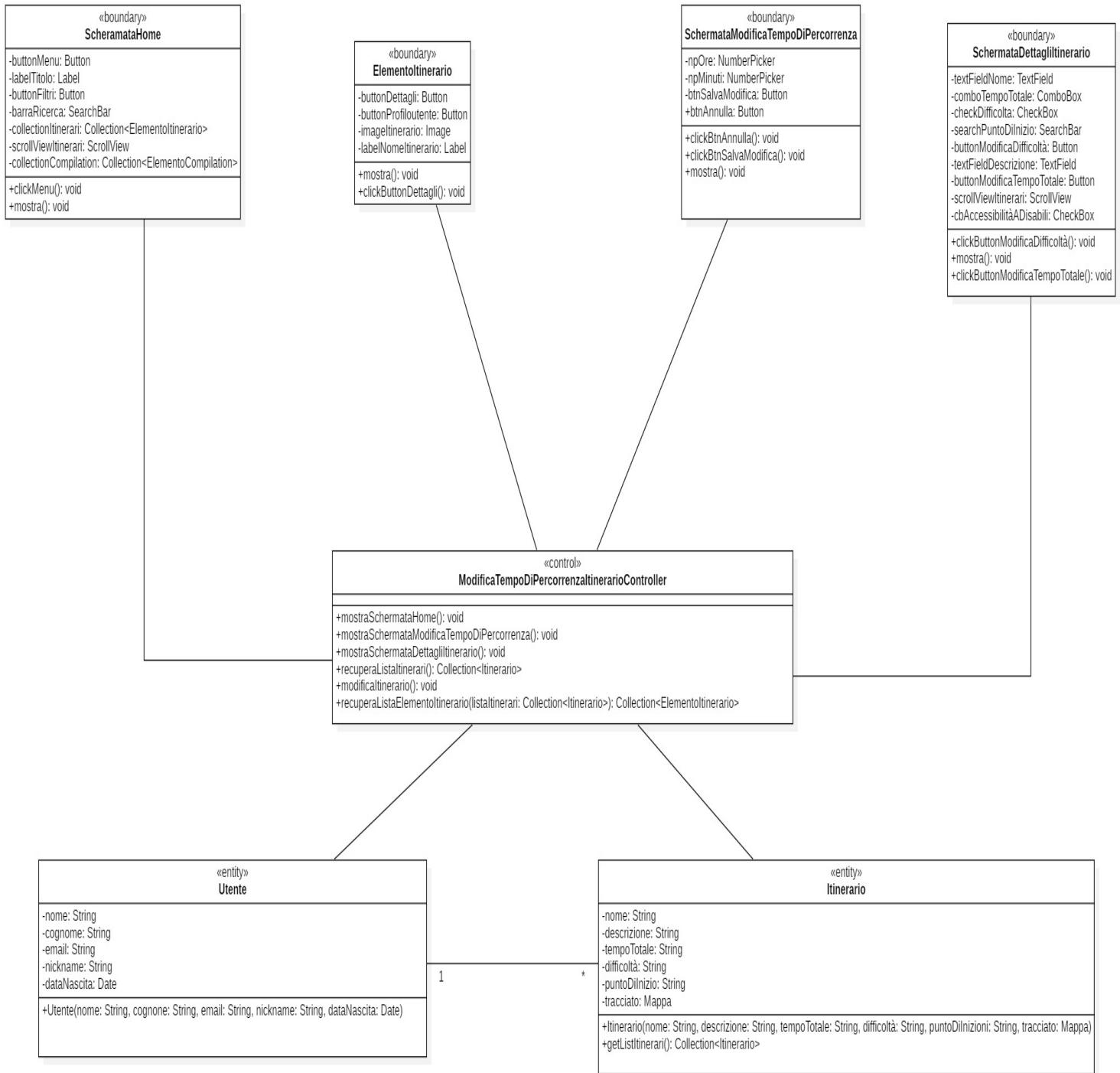
## Utente registrato modifica nickname



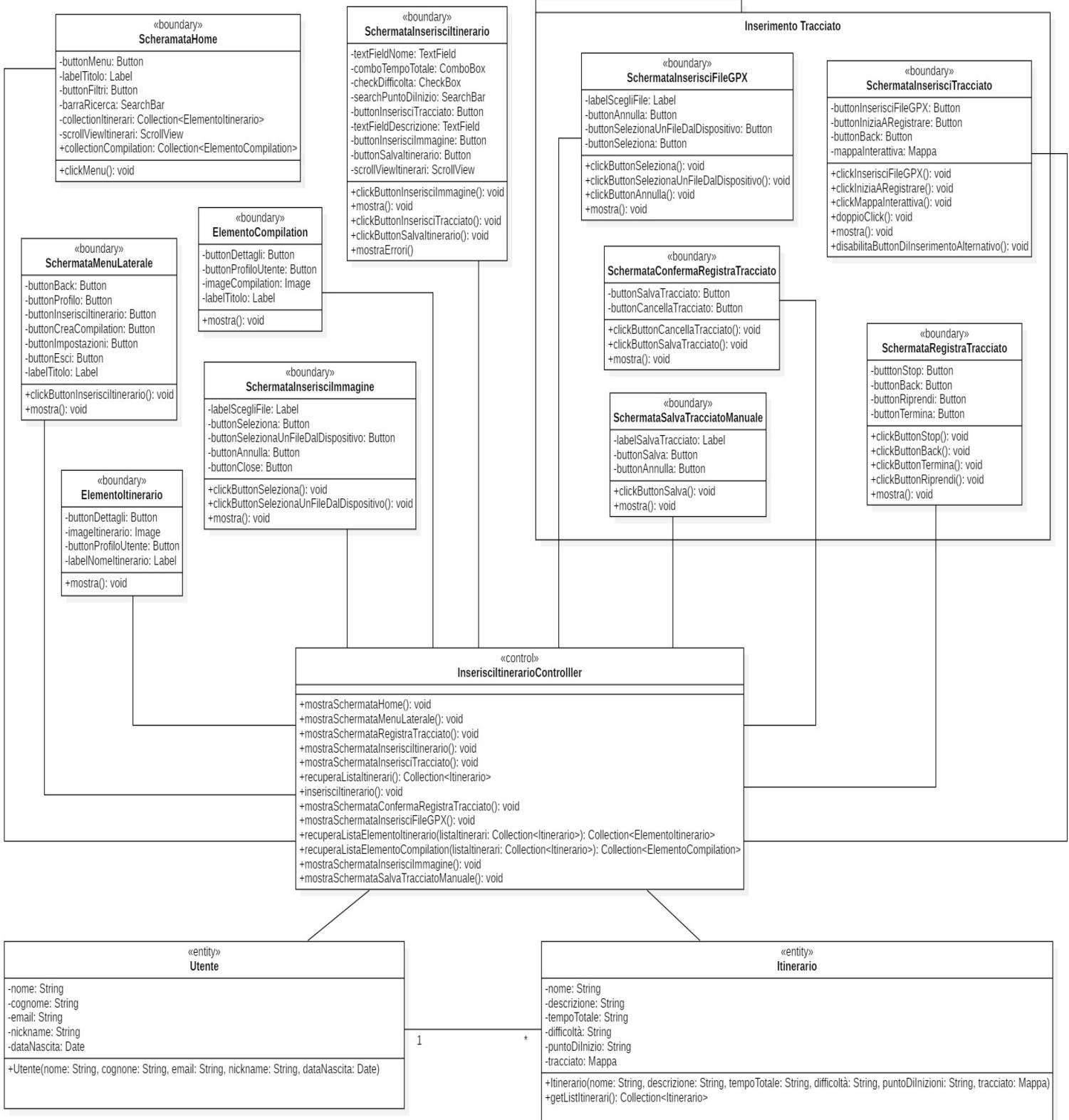
## Utente autenticato modifica difficoltà itinerario



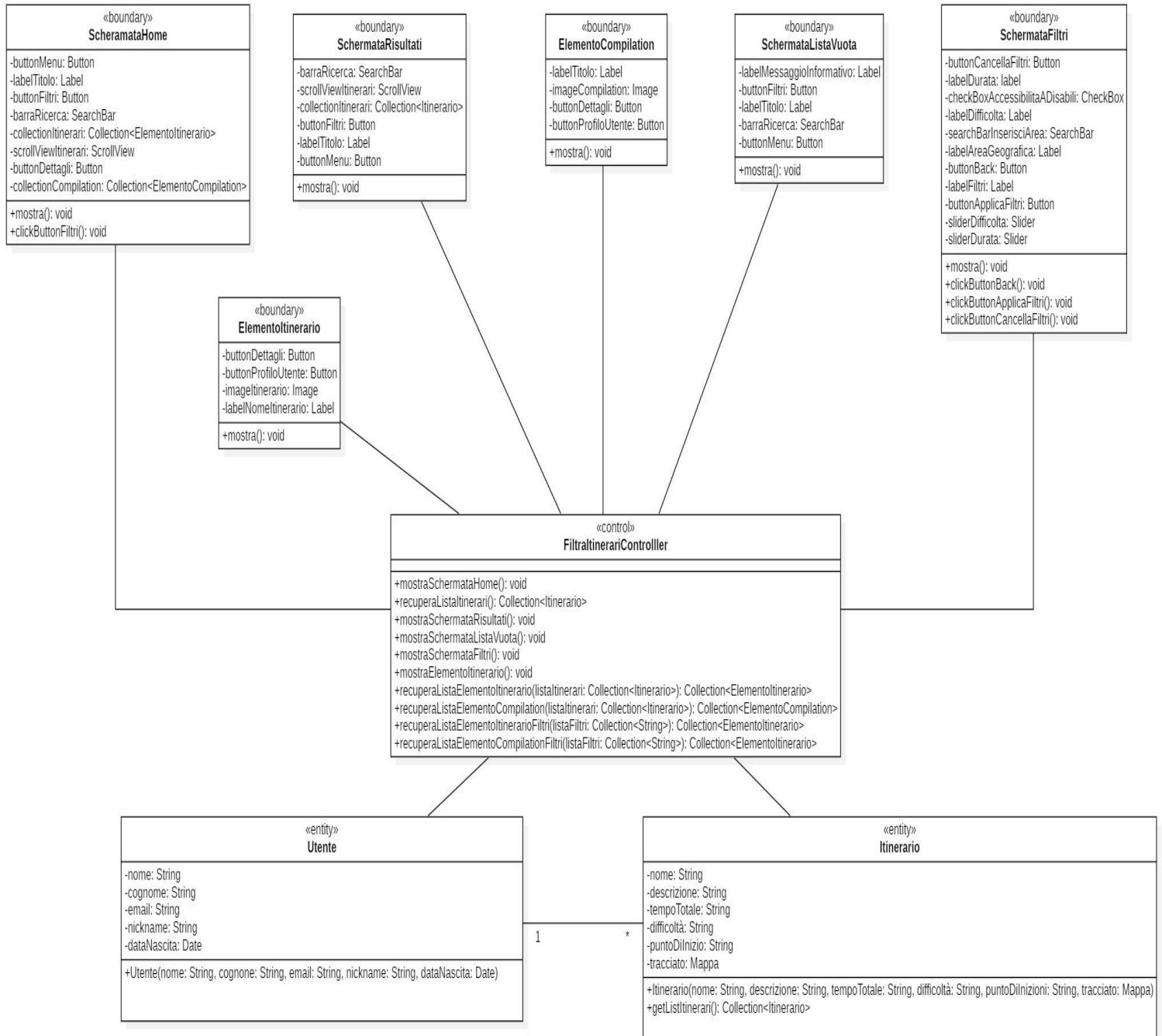
## Utente autenticato modifica tempo di percorrenza itinerario



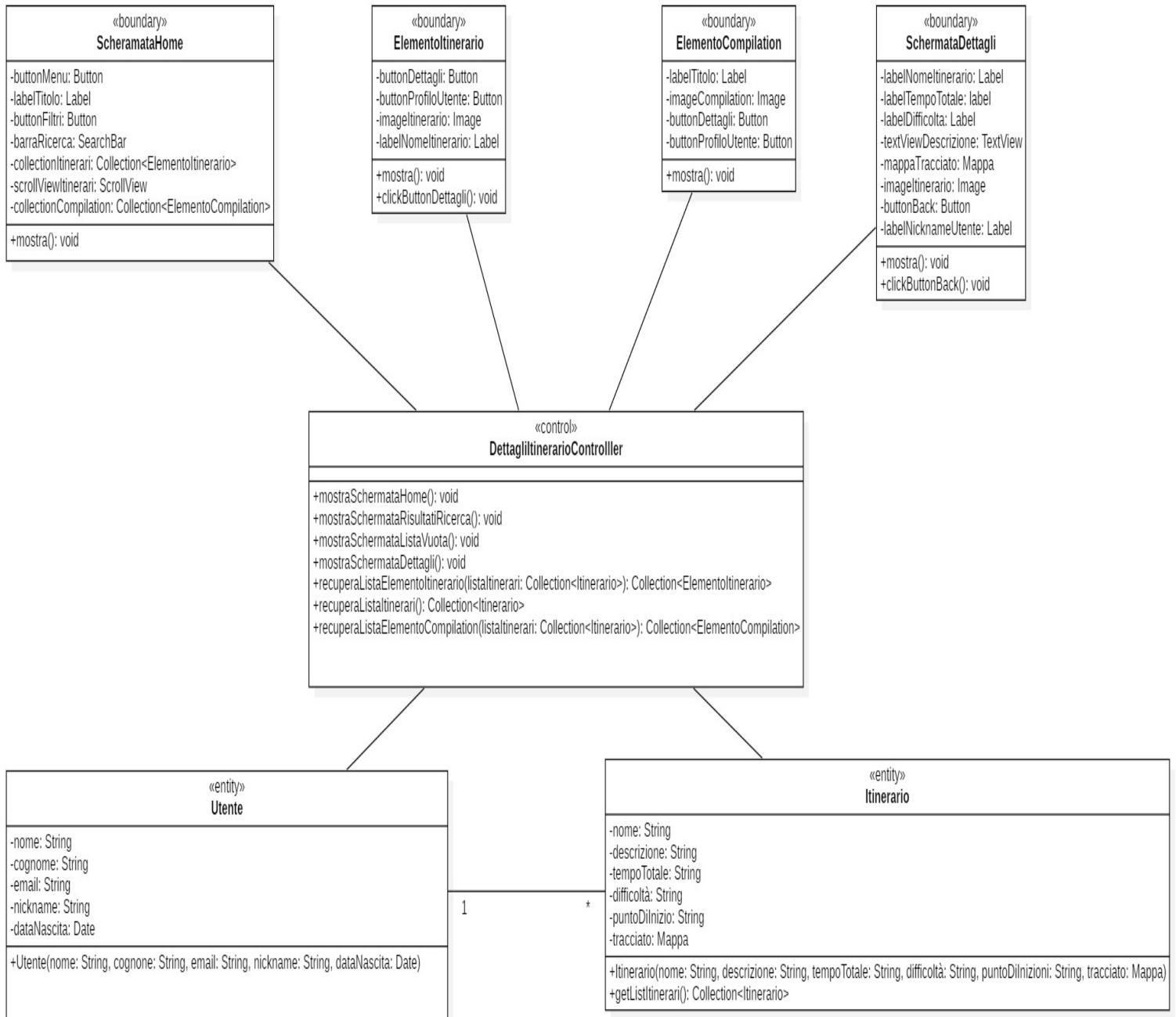
## Utente autenticato inserisce nuovo itinerario



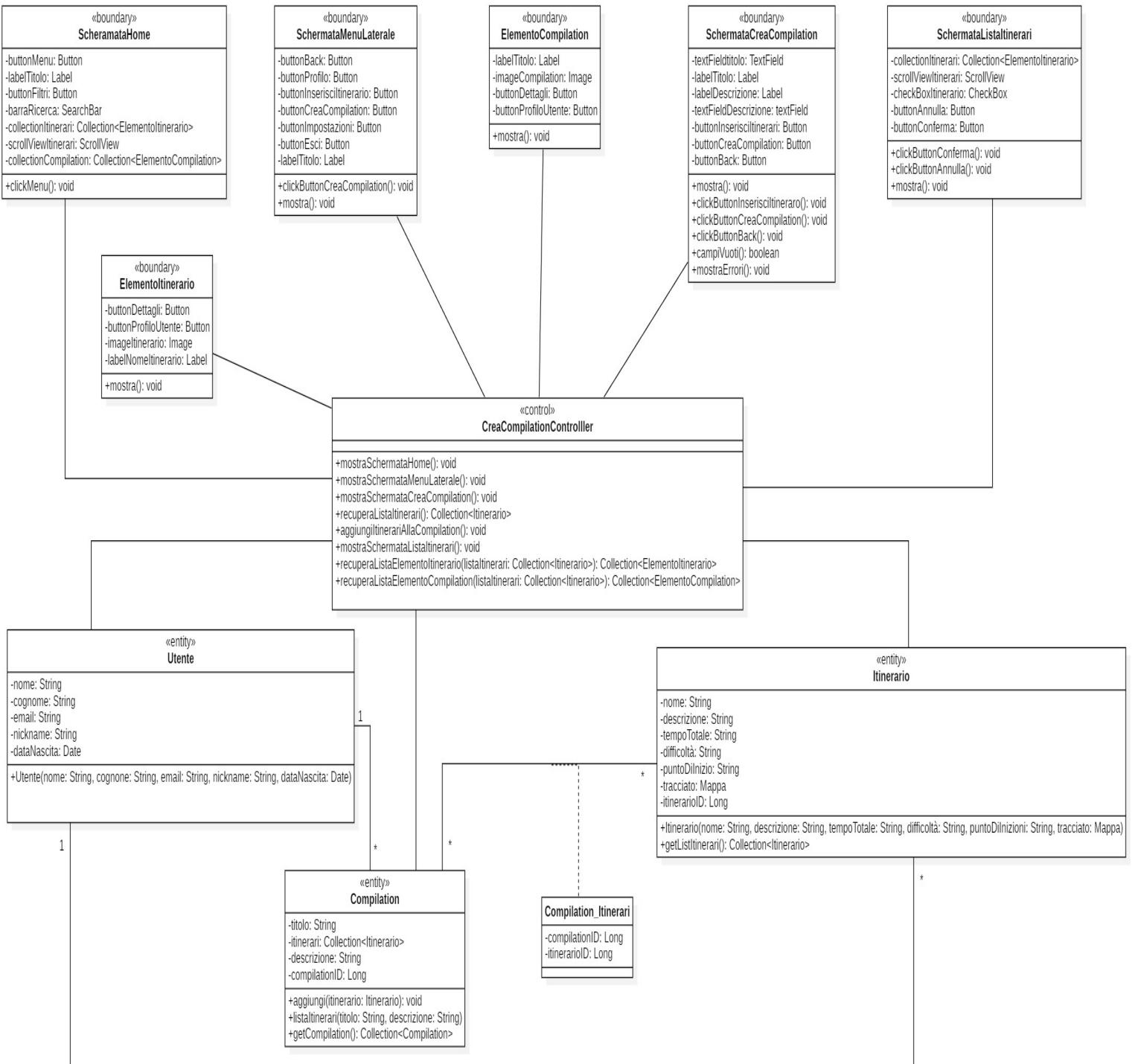
## Utente filtra ricerche itinerari



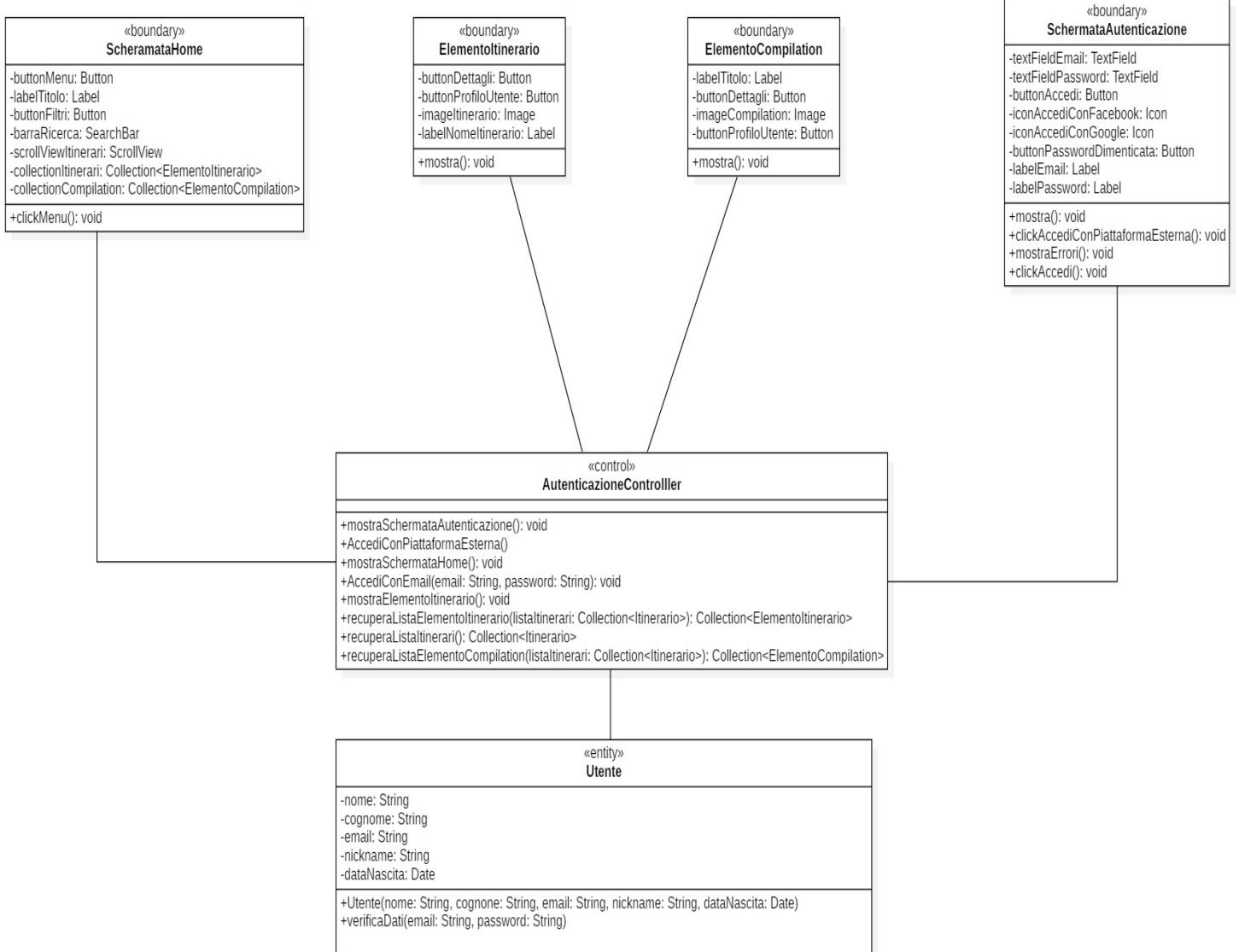
## Utente visualizza schermata di dettaglio



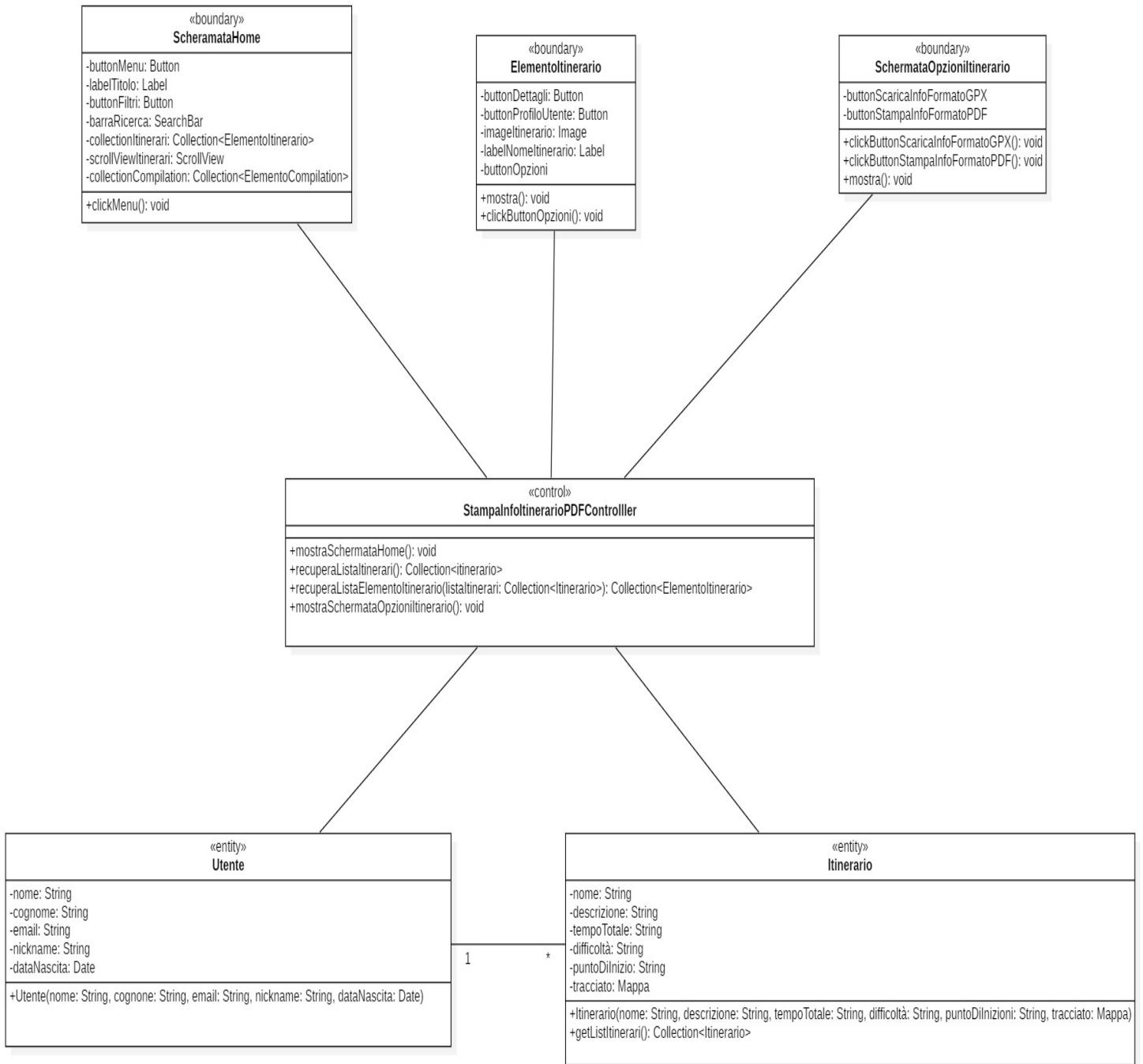
## Utente autenticato crea compilation di sentieri



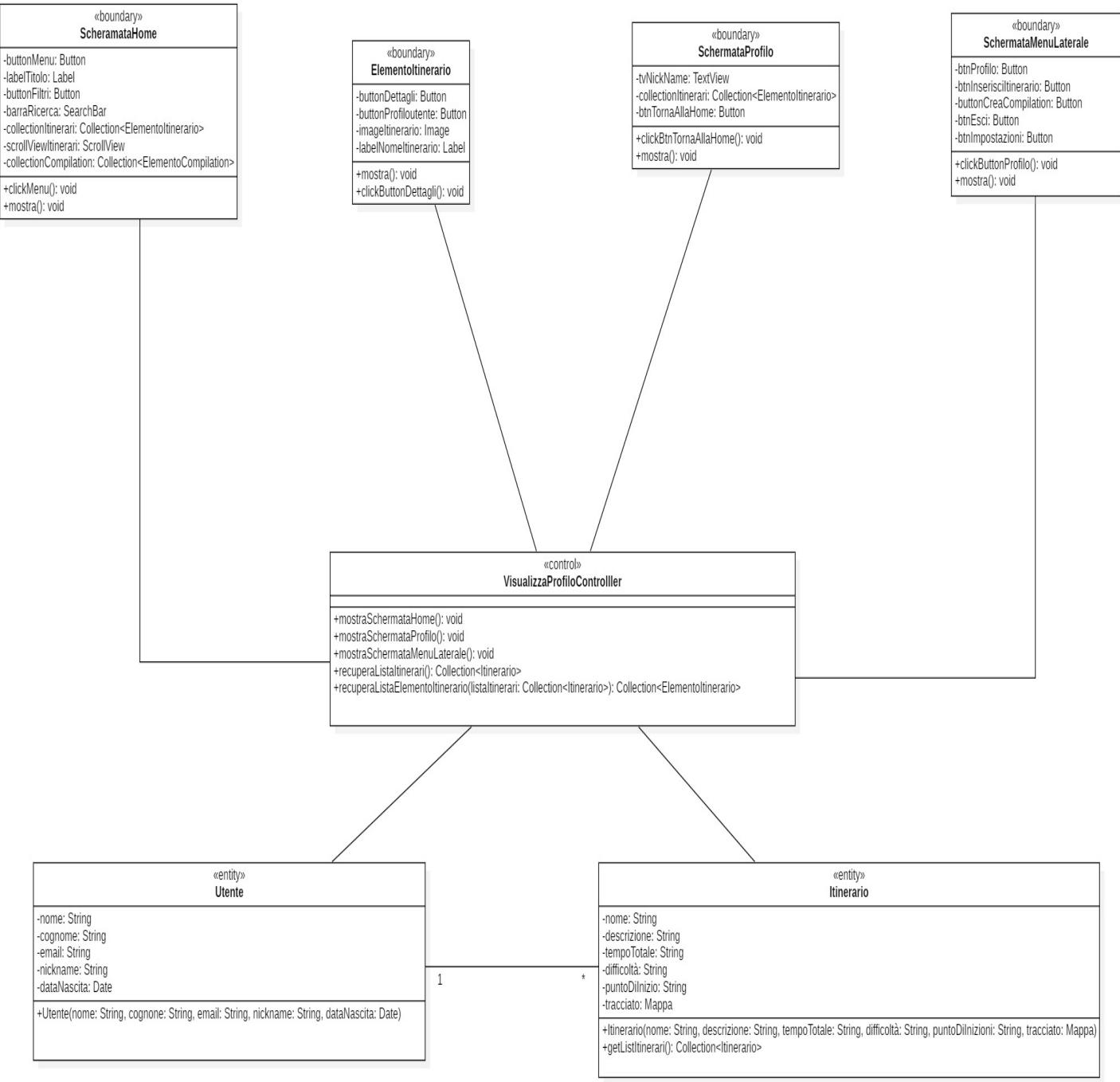
## Utente accede alla piattaforma



## Utente stampa informazioni itinerario in formato PDF



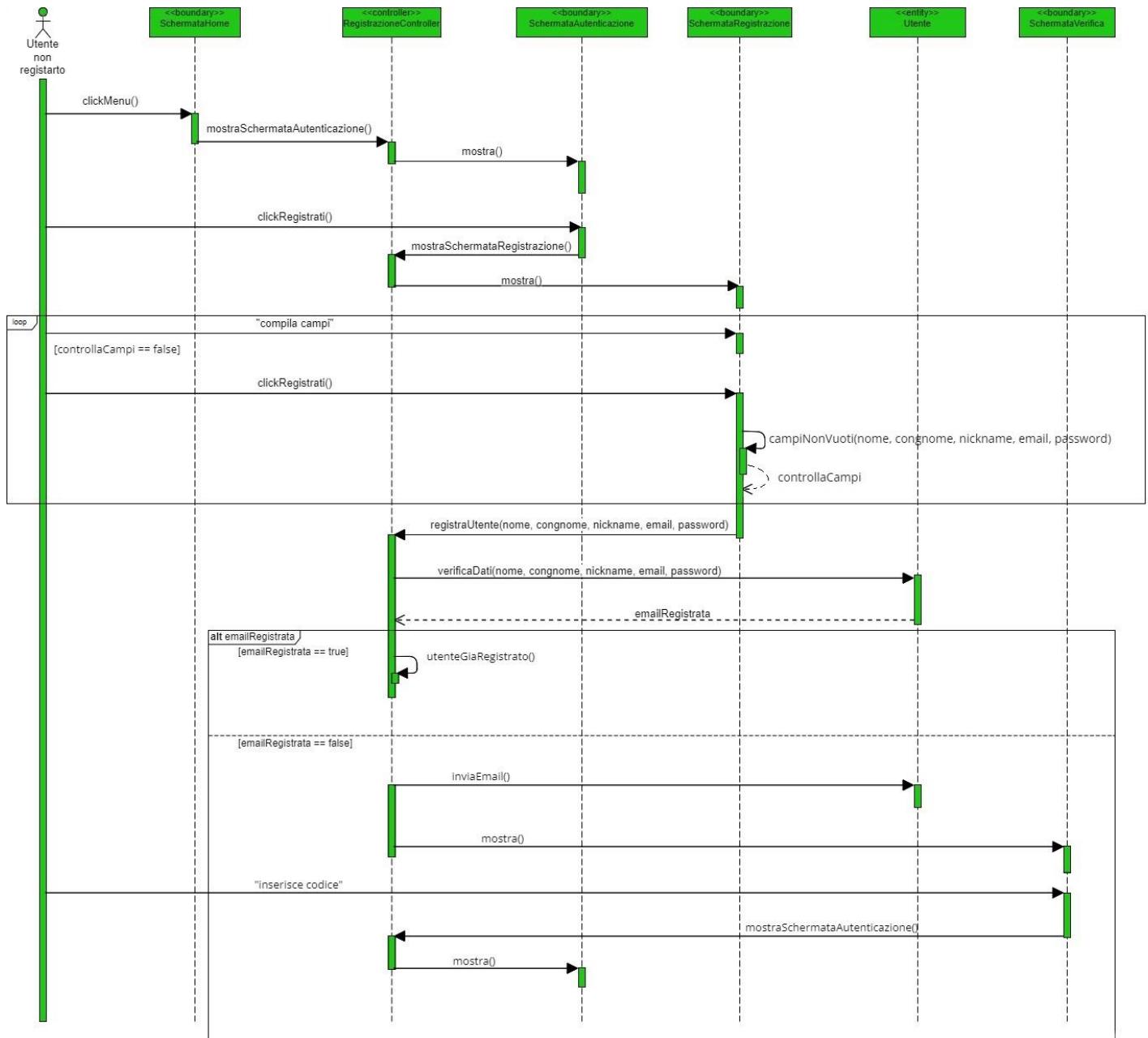
## Utente autenticato visualizza il profilo



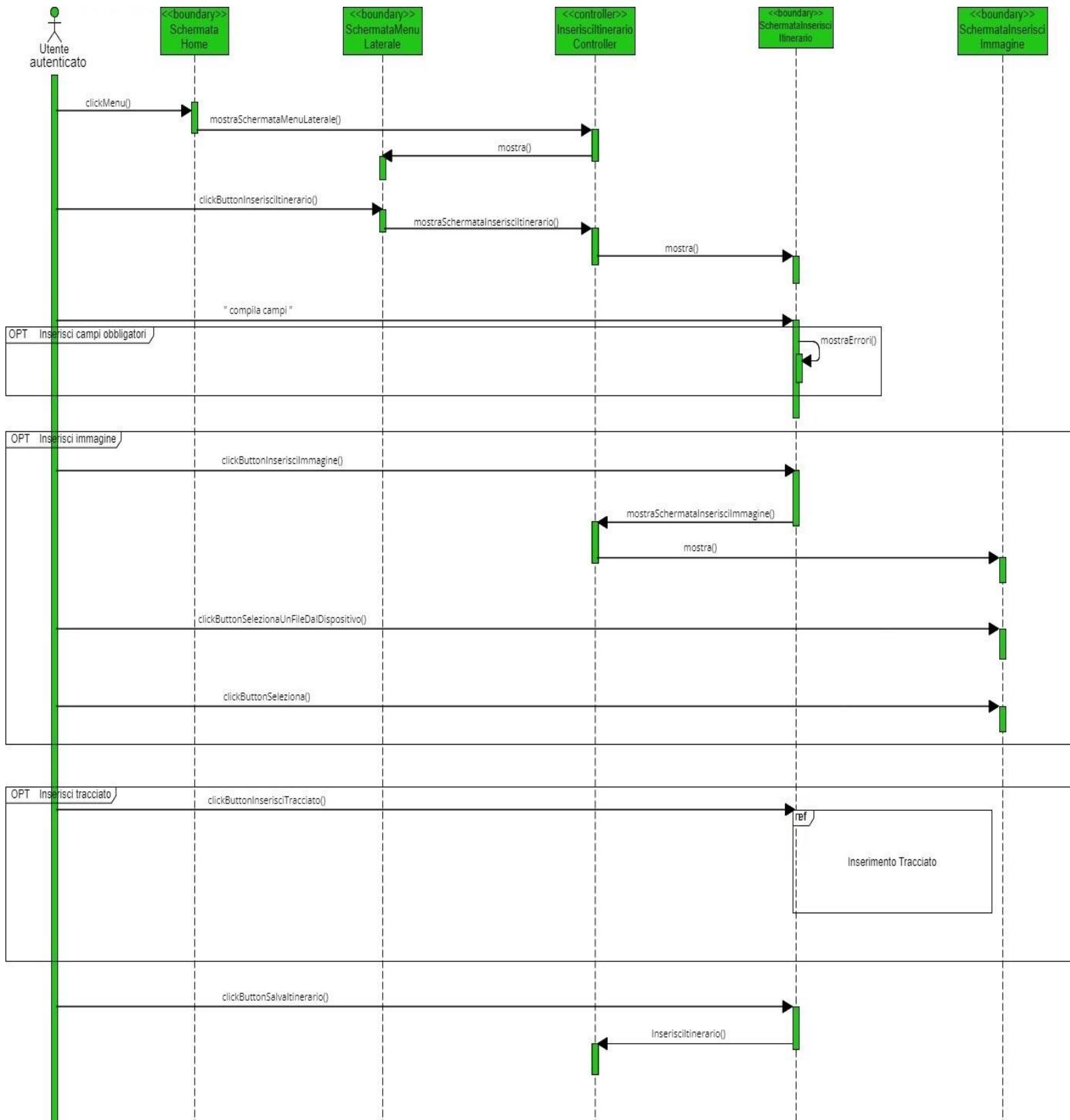
## 1.3.2 Sequence Diagrams di analisi

In questa sezione vengono elencati i sequence diagrams relativi ai casi d'uso di registrazione ed inserimento itinerario dello Use Case Diagram riportato nel paragrafo precedente.

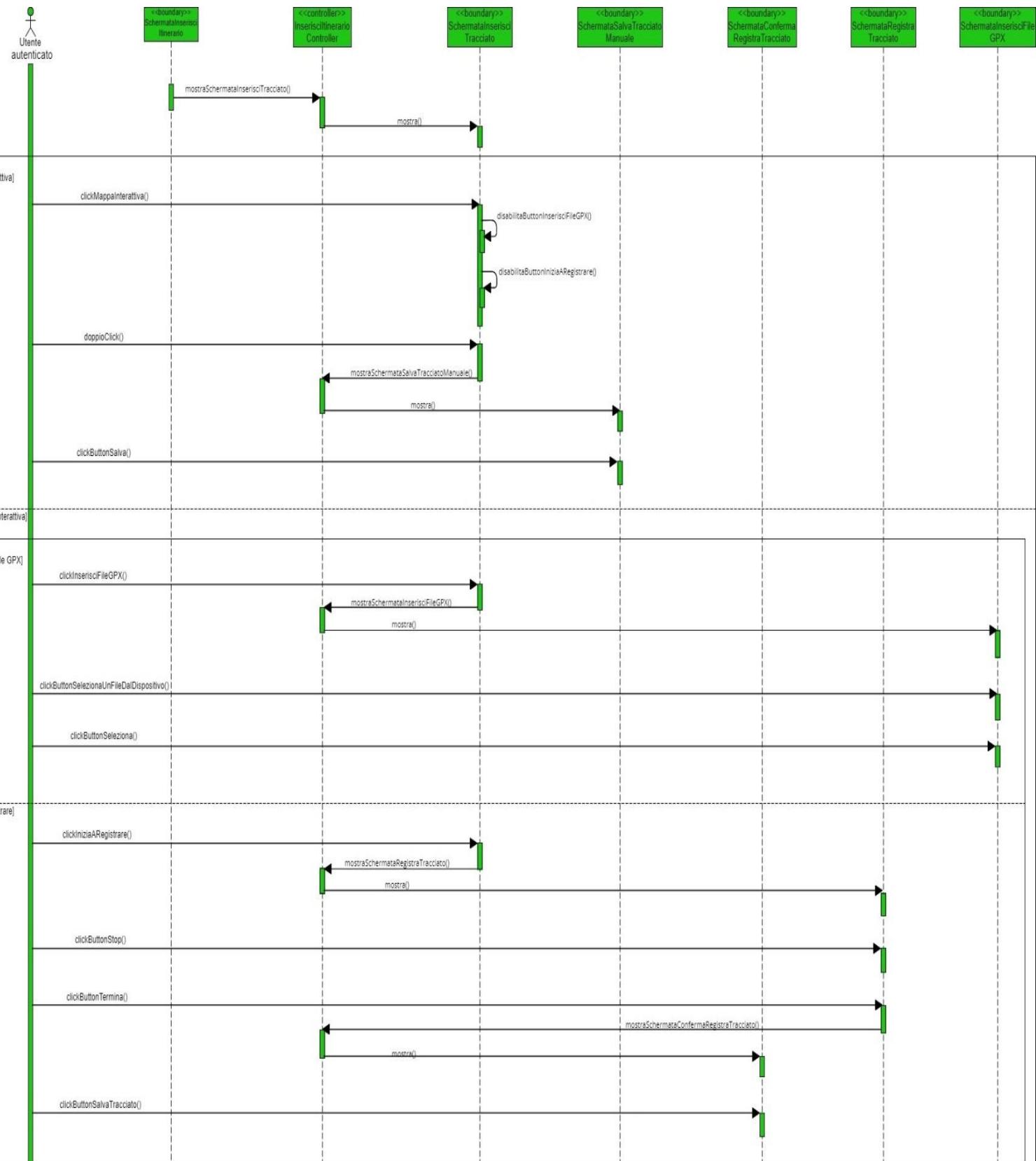
### Utente non registrato effettua registrazione



## Utente autenticato inserisce nuovo itinerario

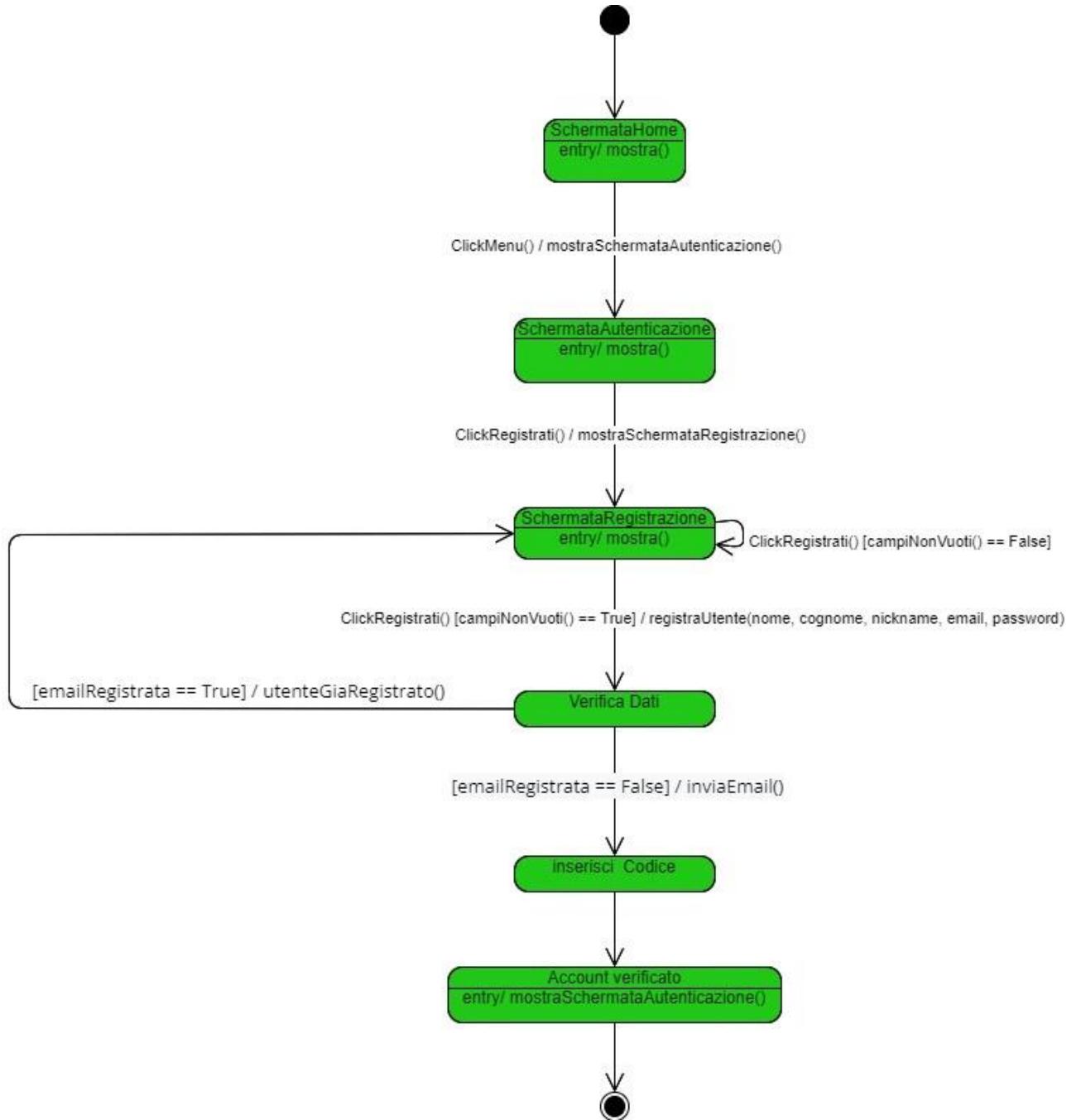


## Continuo “Utente autenticato inserisce nuovo itinerario”

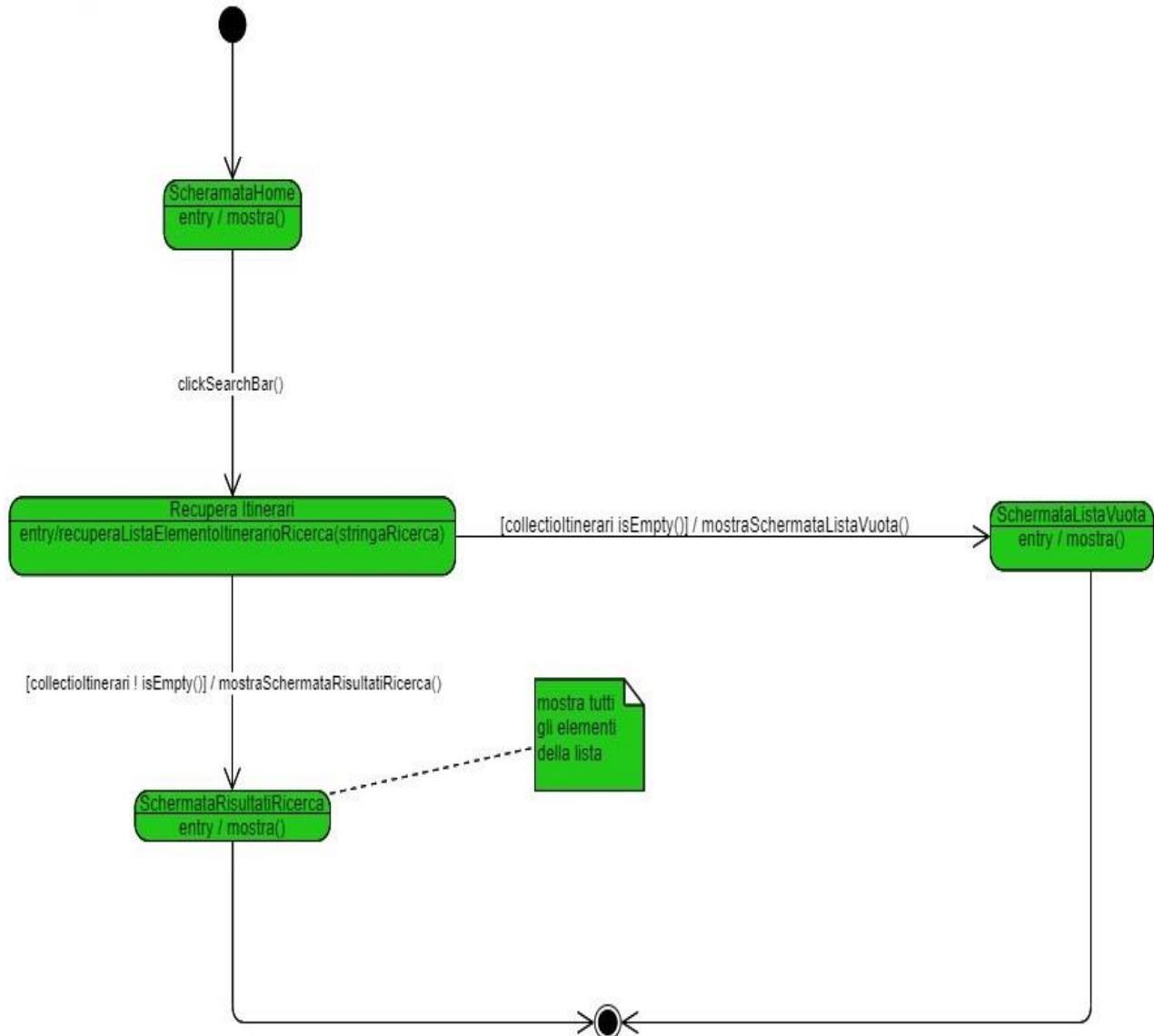


### 1.3.3 State Chart Diagrams di analisi

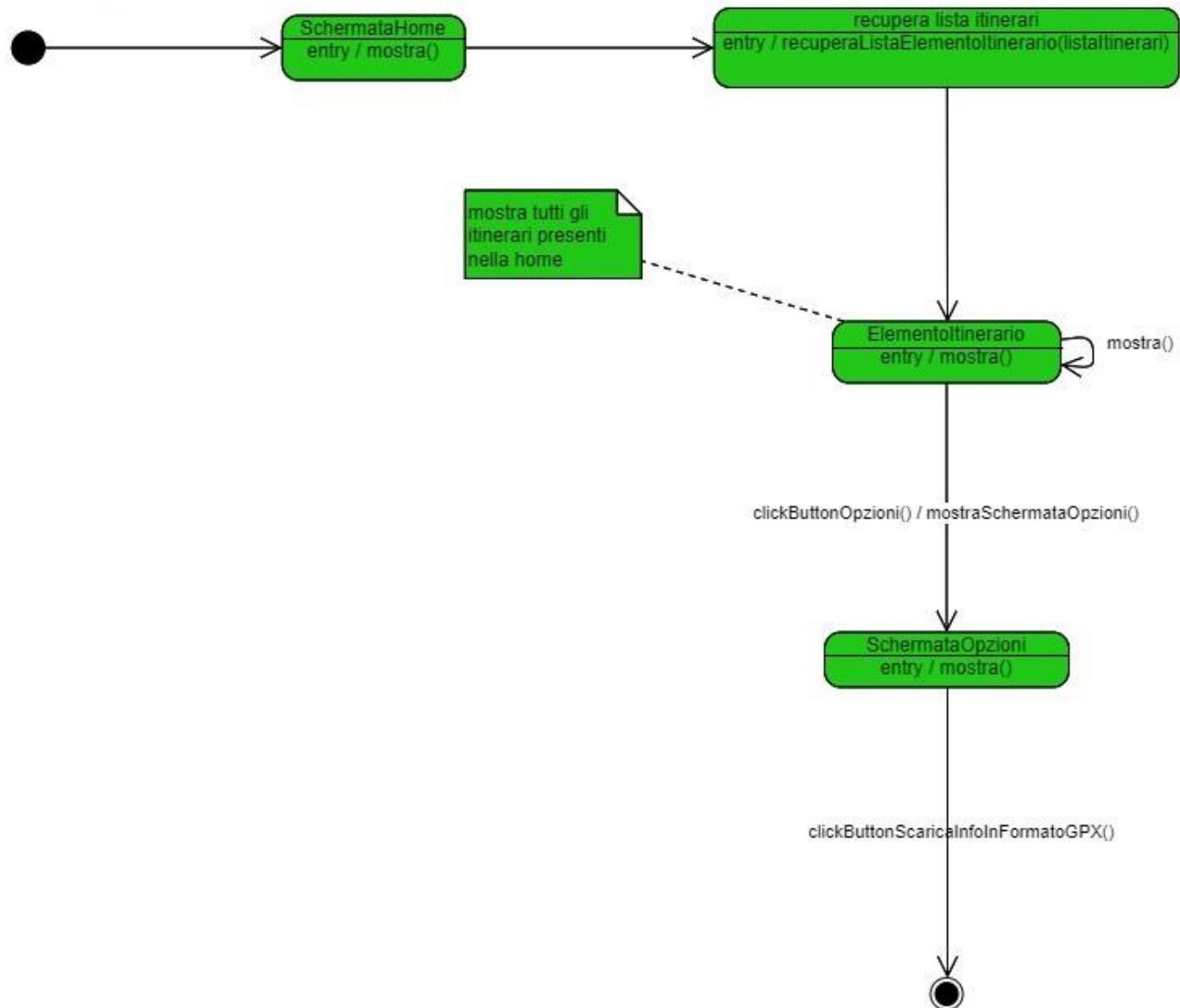
Utente non registrato effettua registrazione



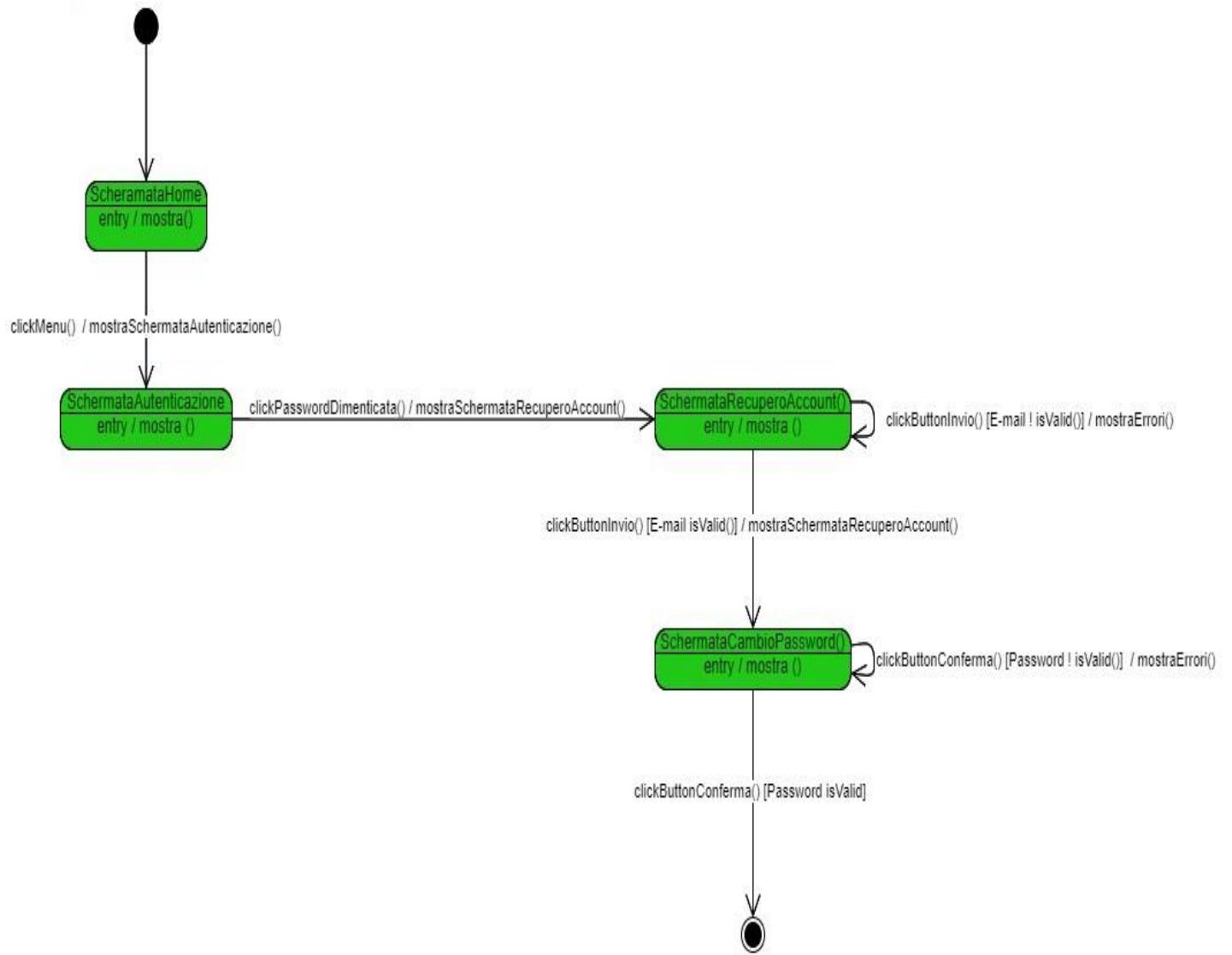
## Utente effettua ricerche itinerari



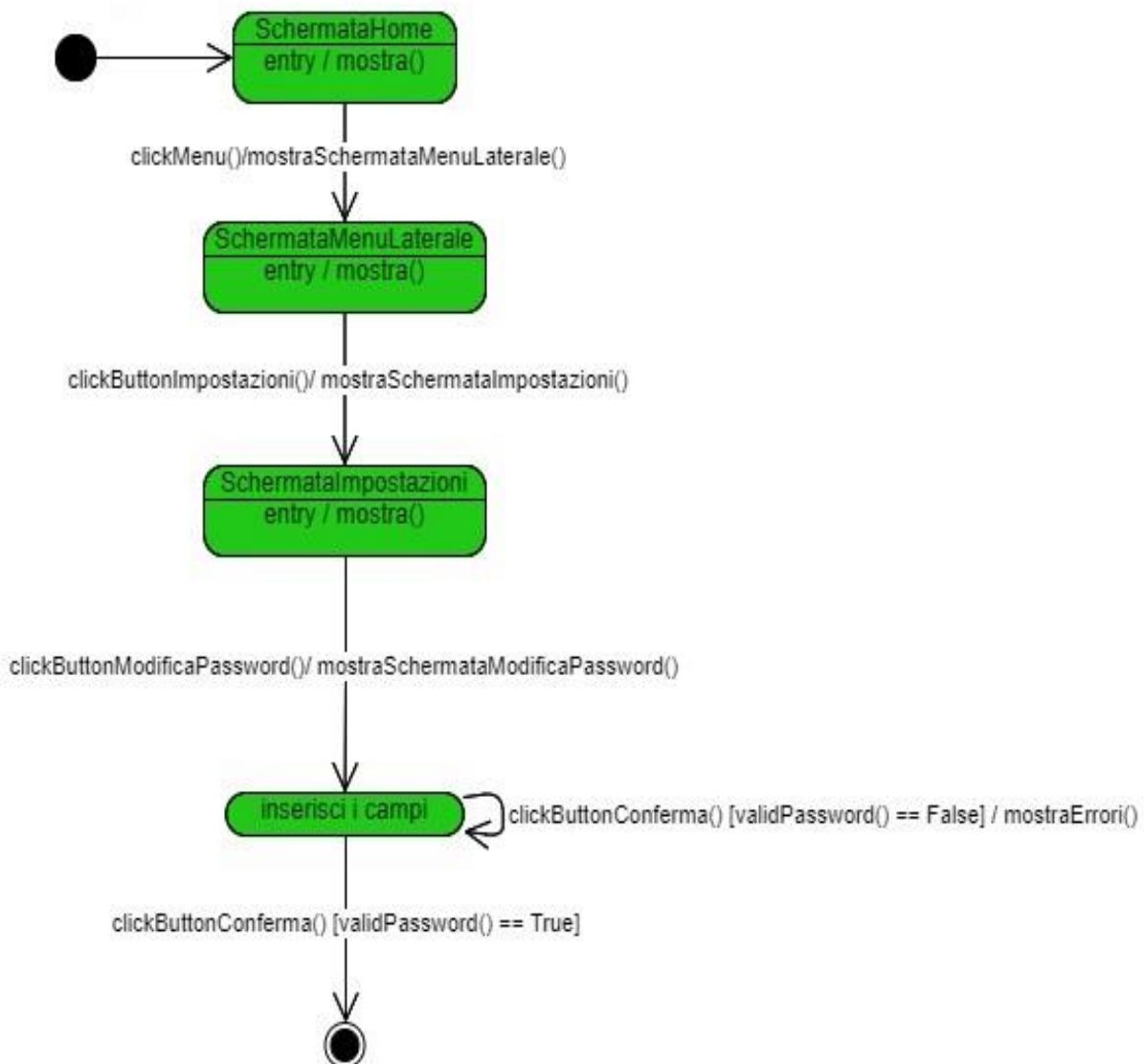
## Utente scarica informazioni itinerario in formato GPX



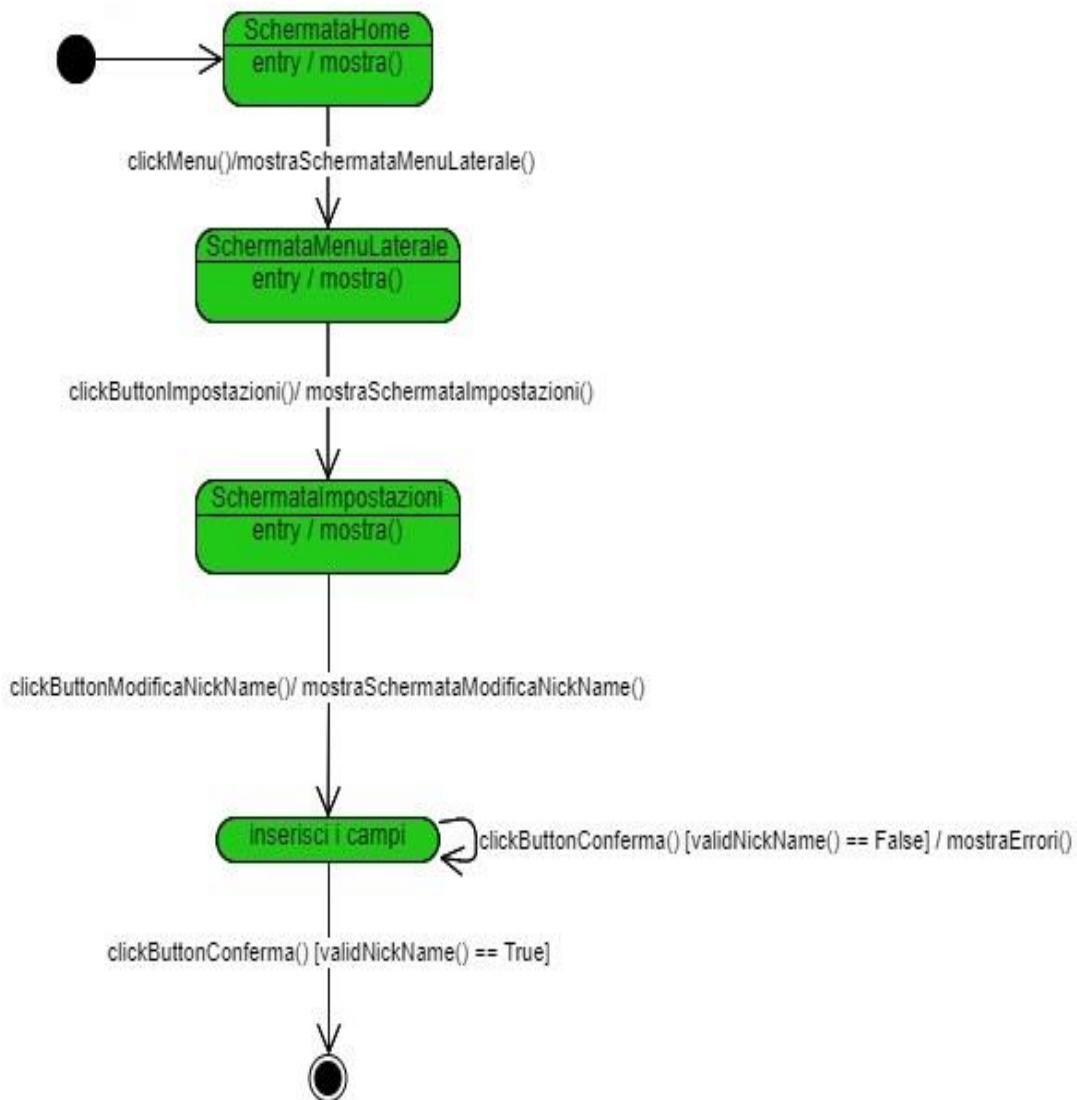
## Utente registrato recupera account



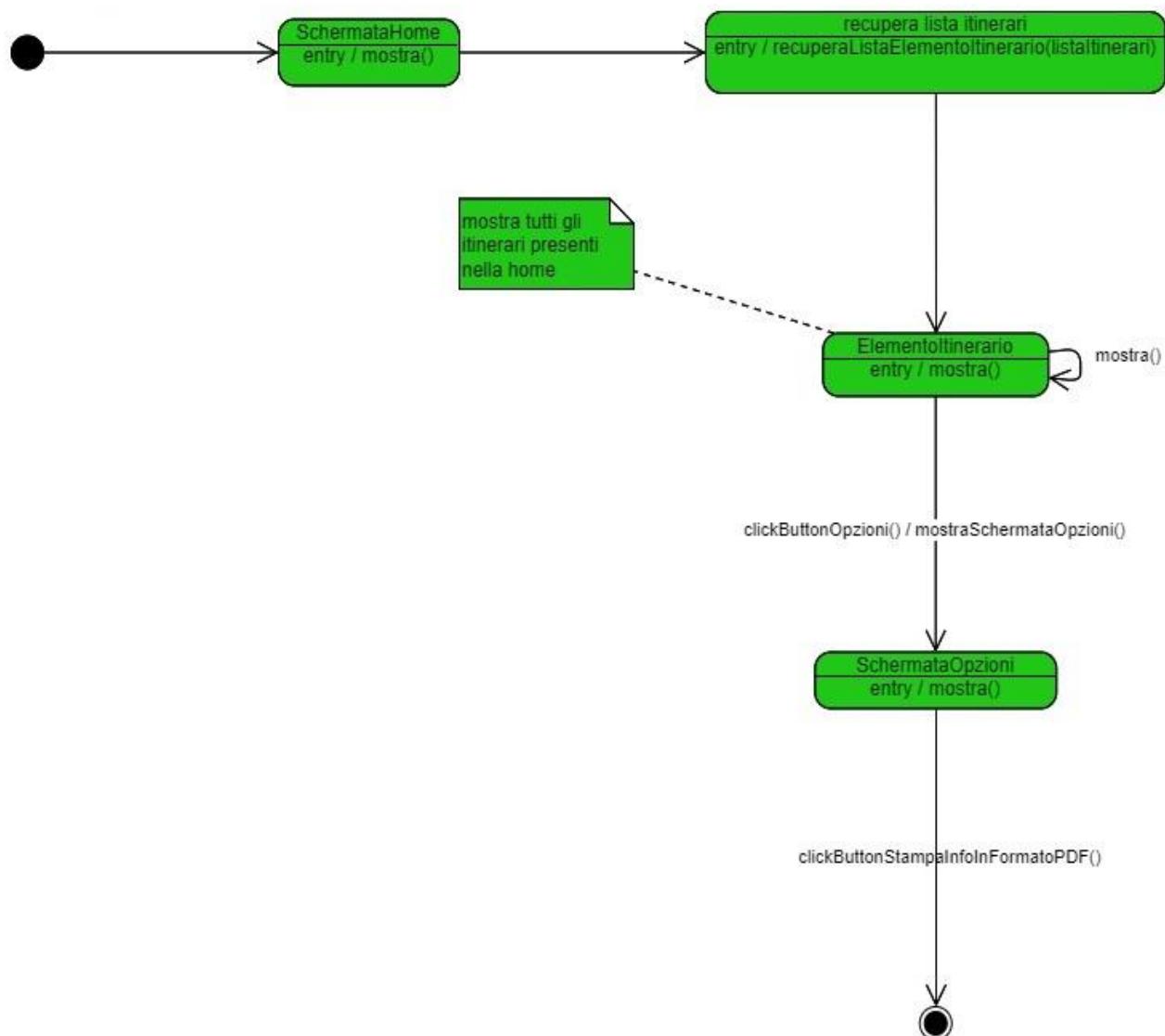
## Utente autenticato modifica password



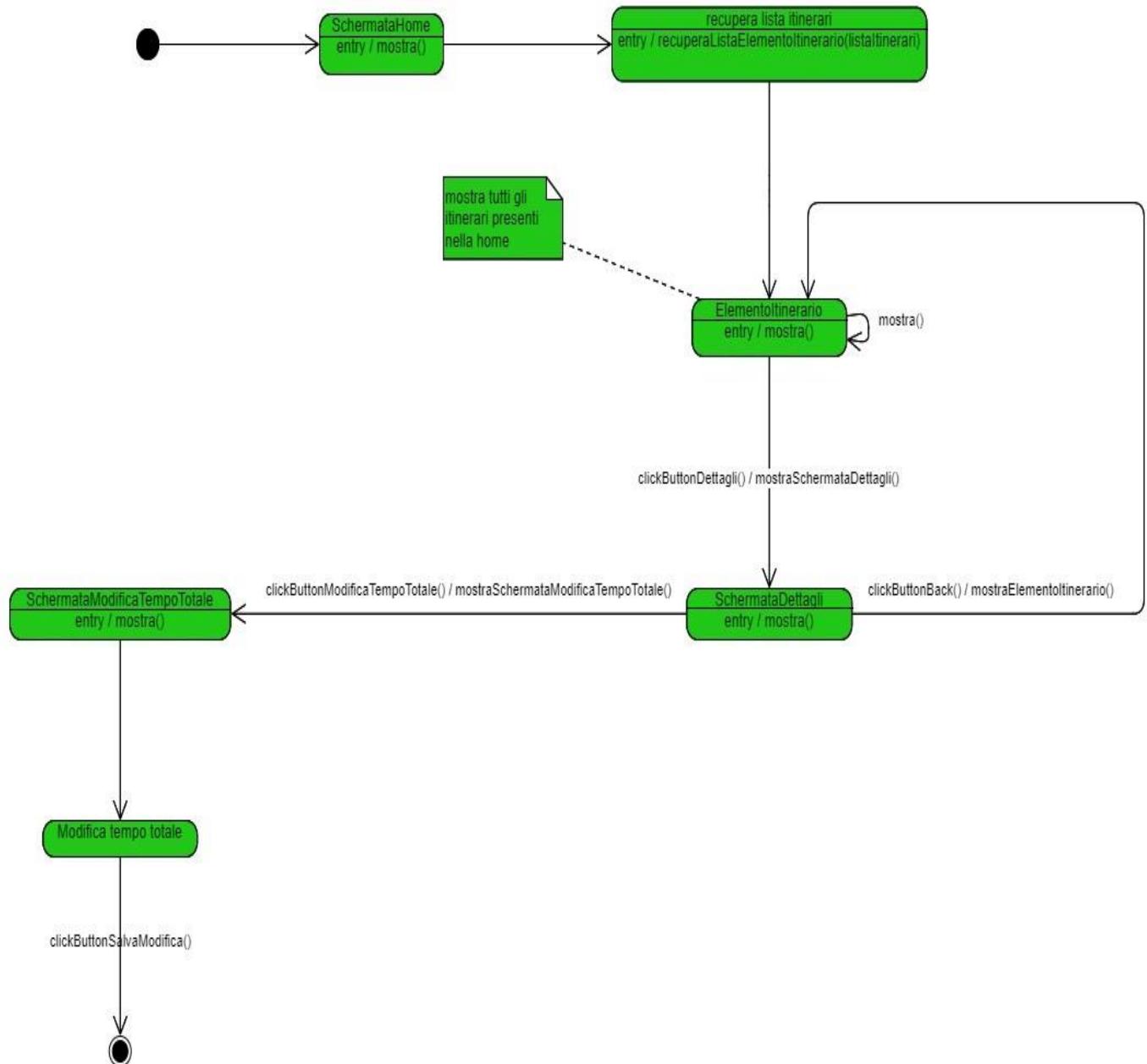
## Utente autenticato modifica nickname



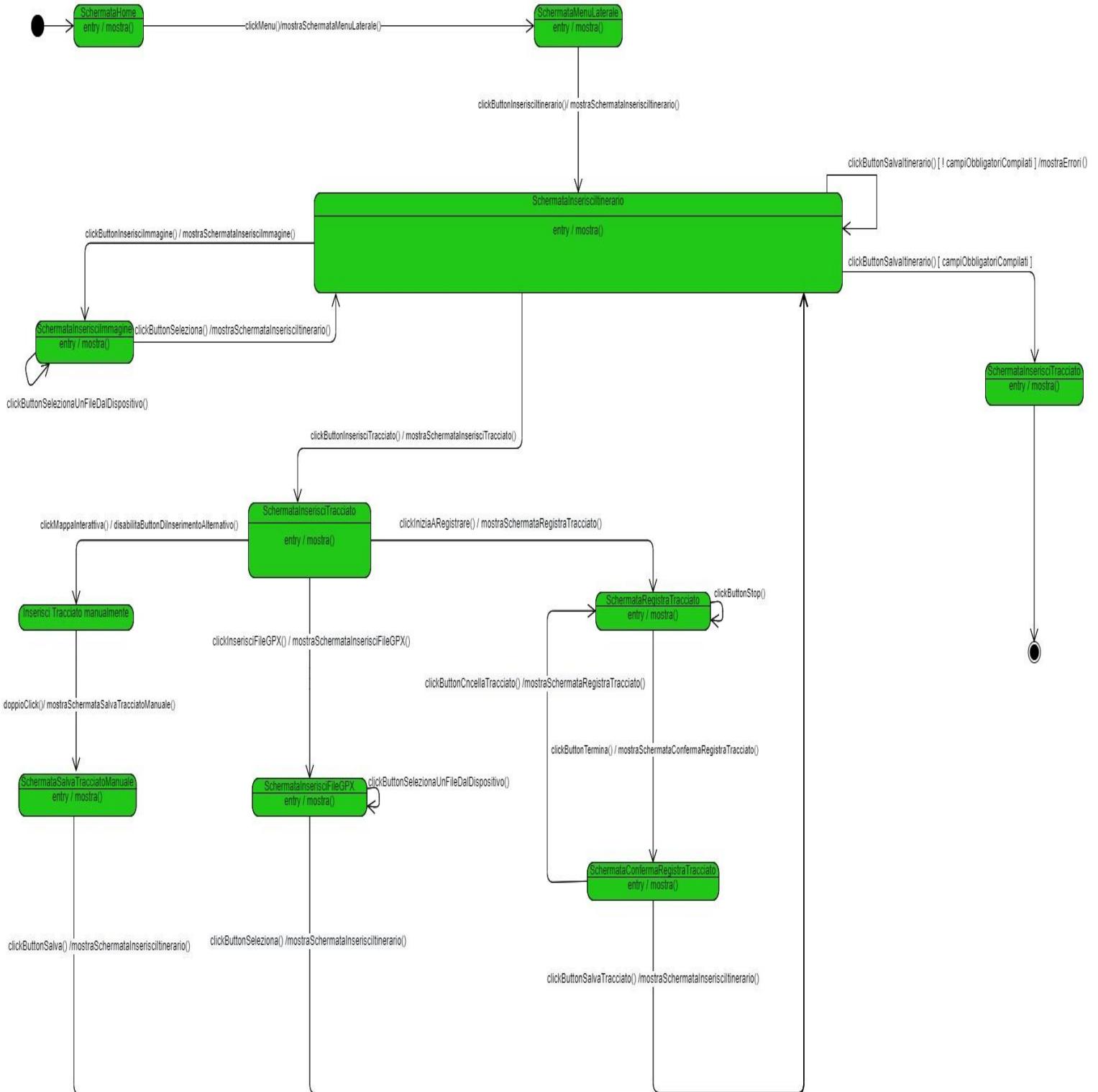
## Utente stampa informazioni itinerario in formato PDF



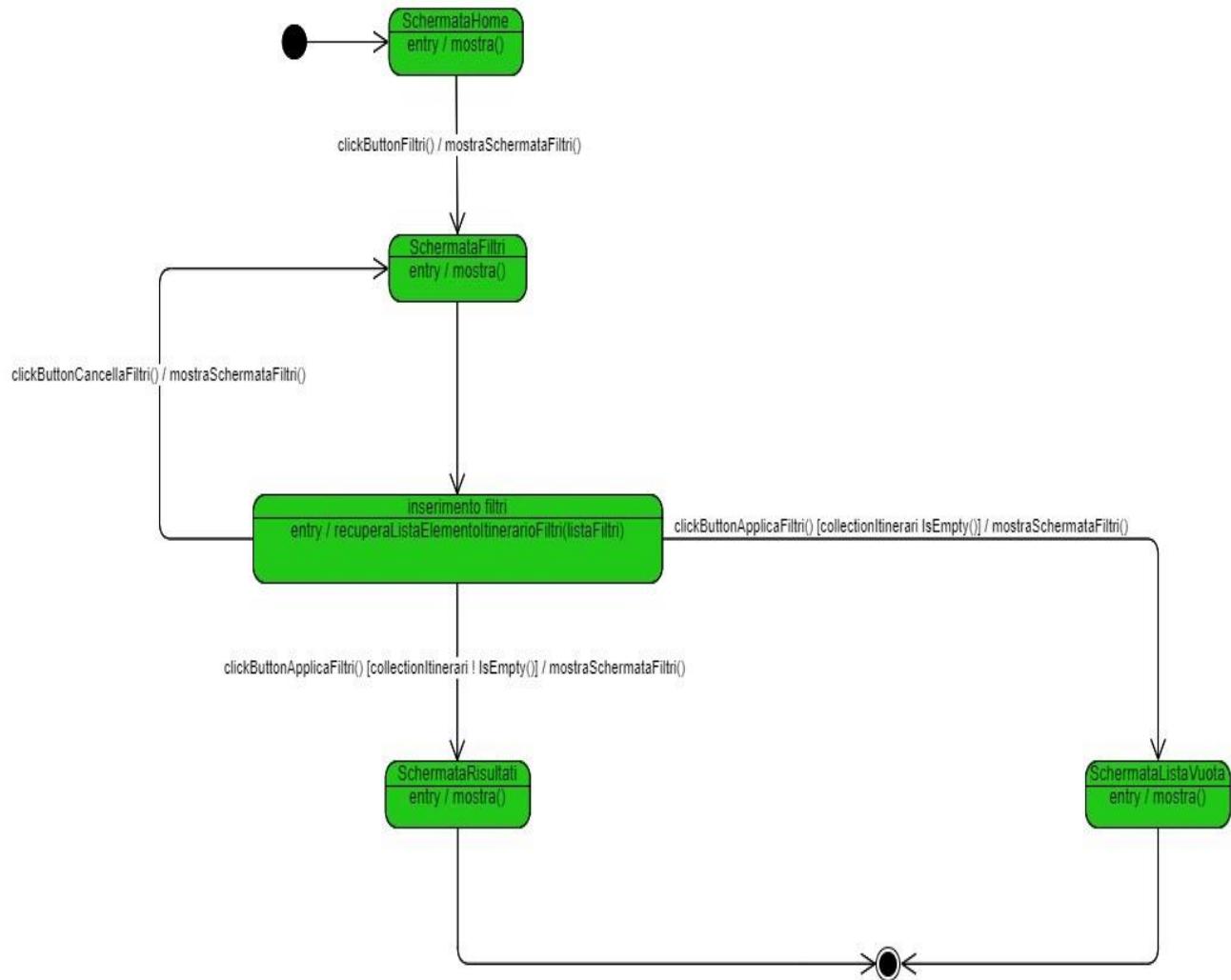
## Utente autenticato modifica tempo di percorrenza itinerario



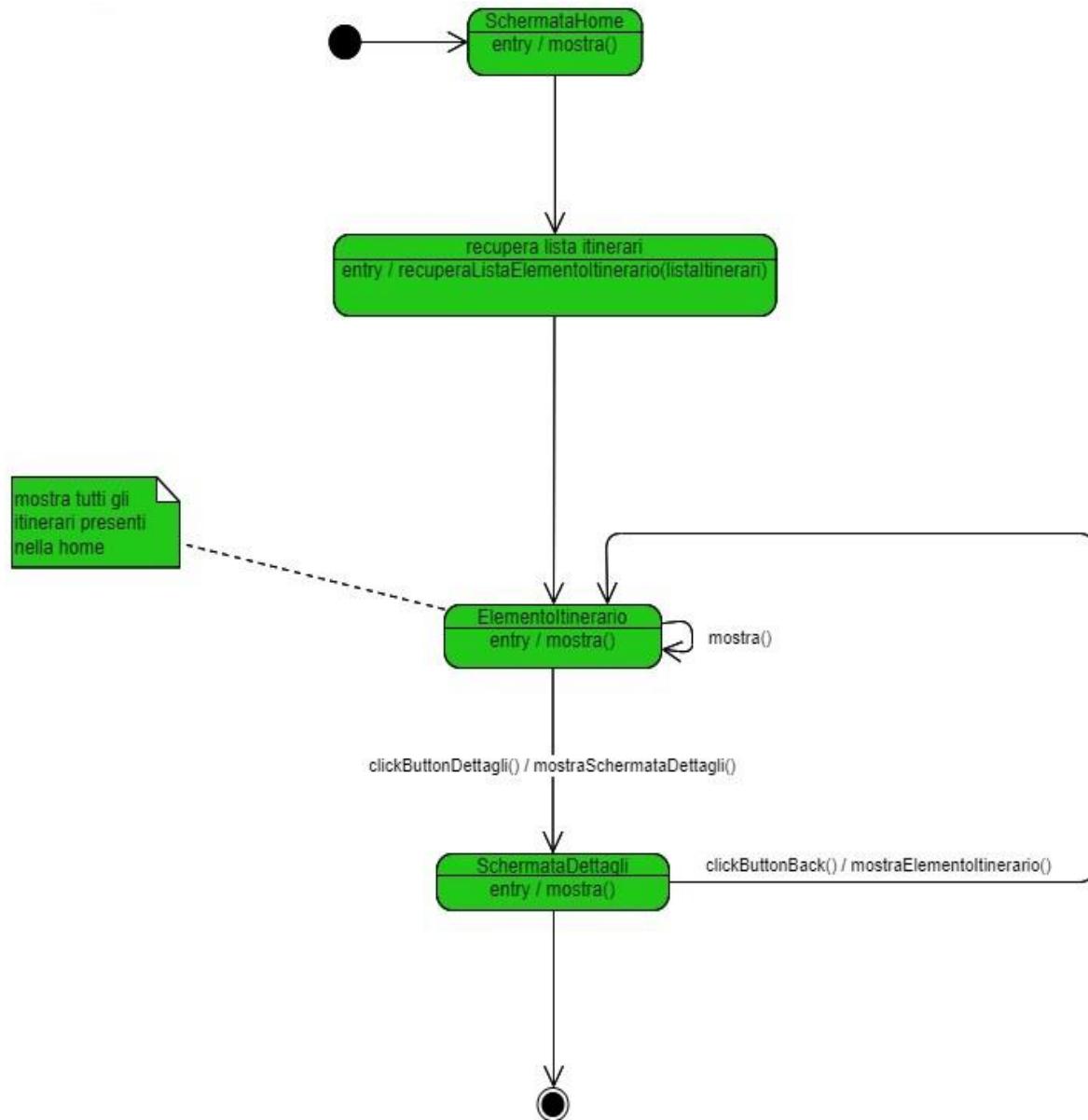
## Utente autenticato inserisce nuovo itinerario



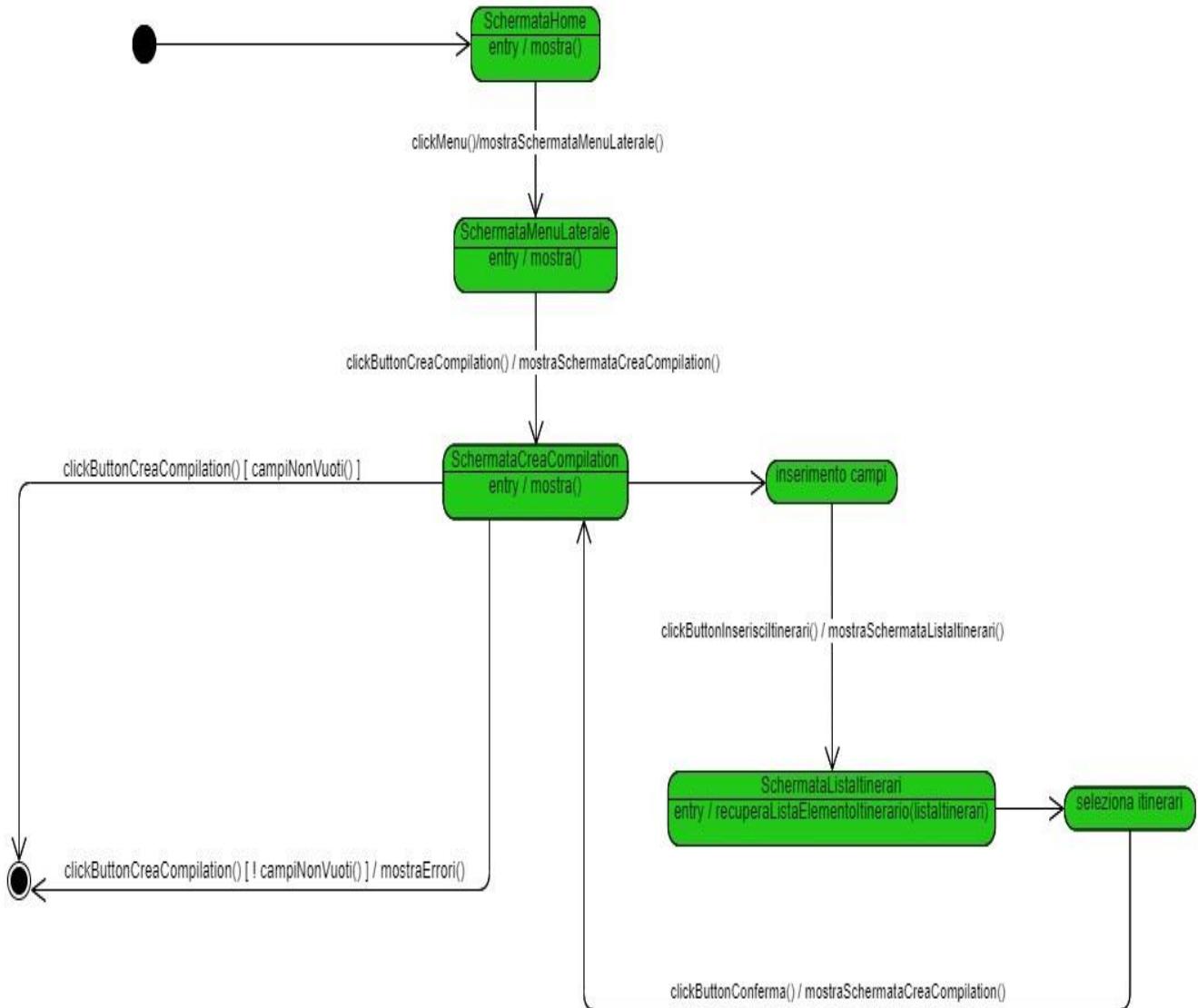
## Utente filtra ricerche itinerari



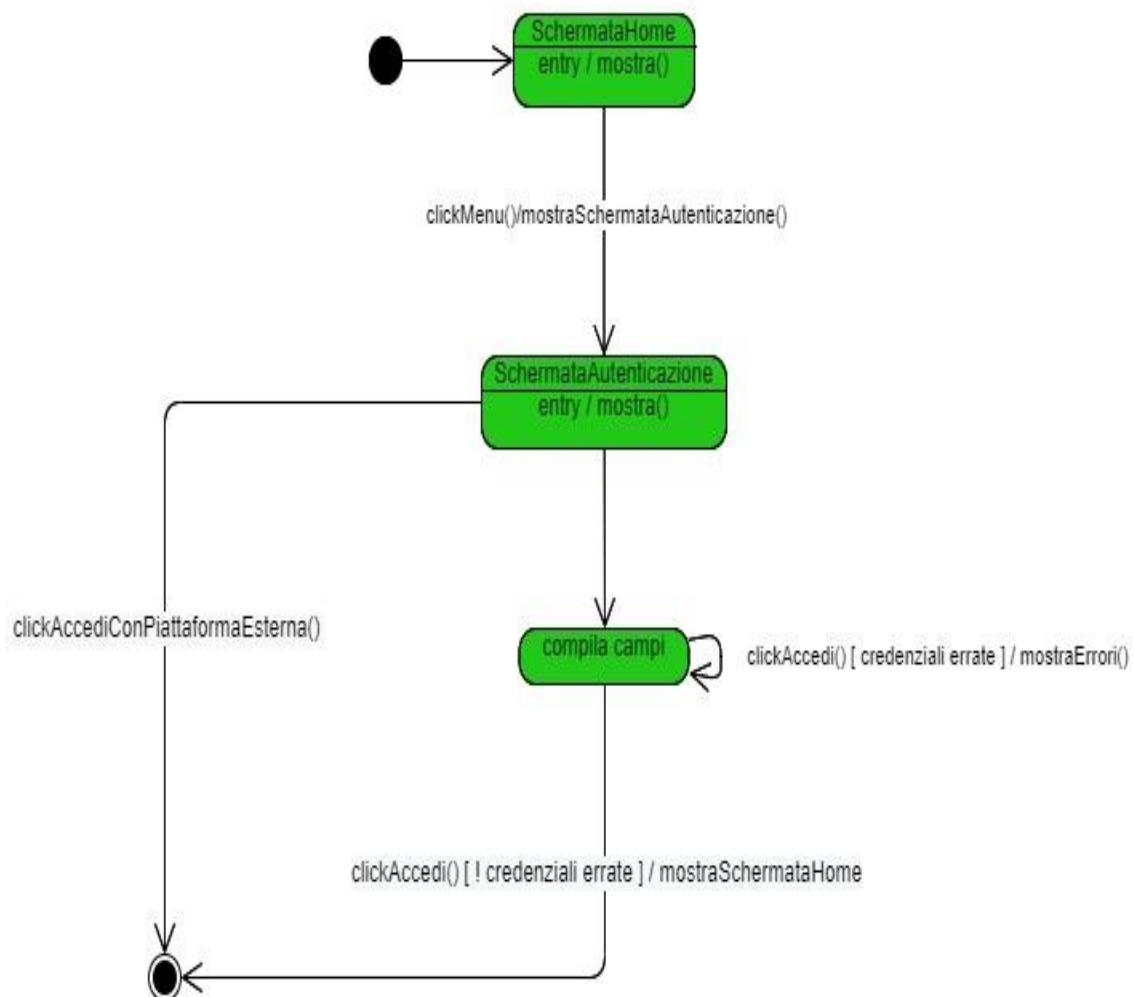
## Utente visualizza schermata di dettaglio



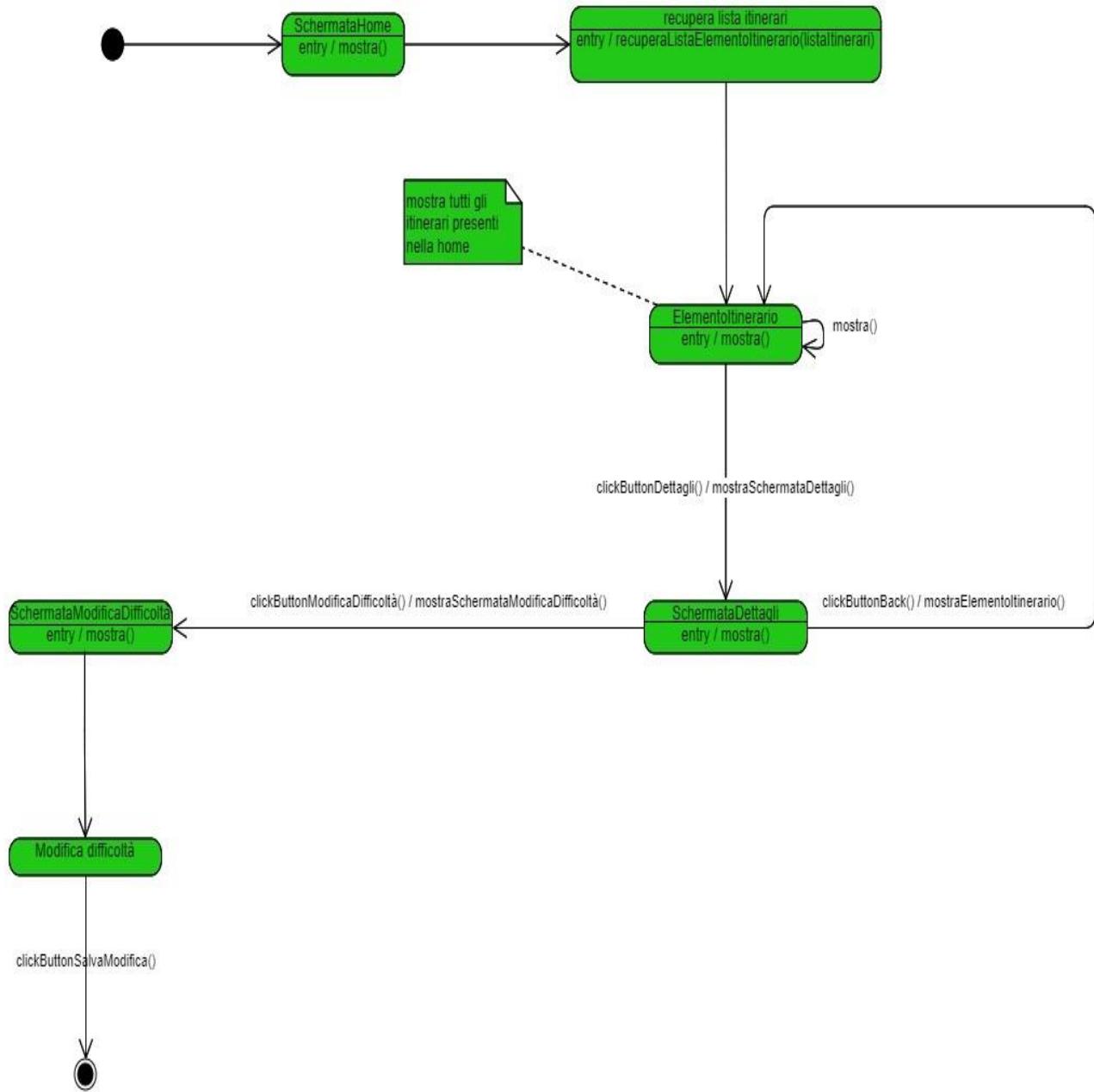
## Utente autenticato crea compilation di sentieri



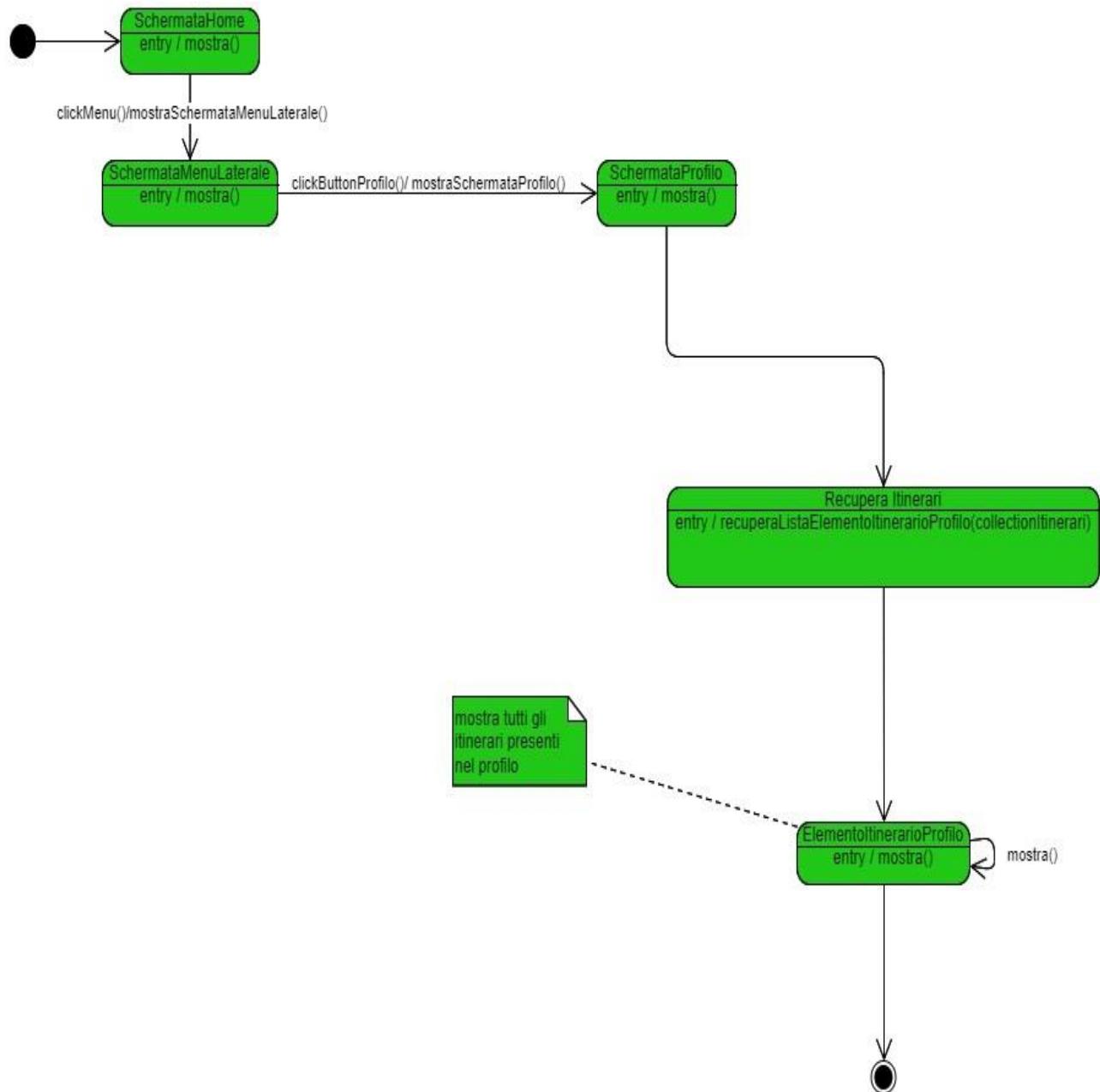
## Utente accede alla piattaforma



## Utente autenticato modifica difficoltà itinerario

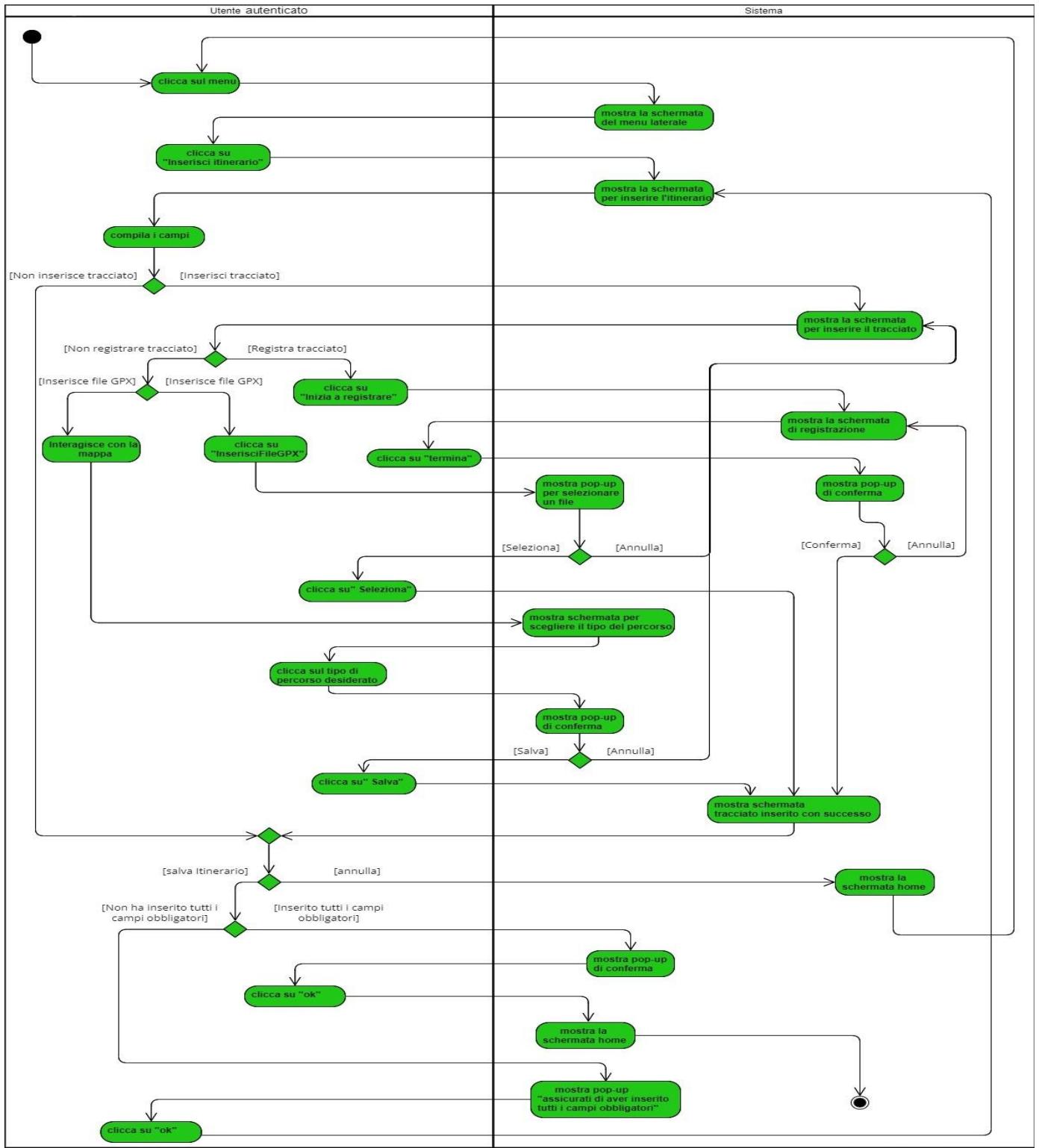


## Utente autenticato visualizza profilo

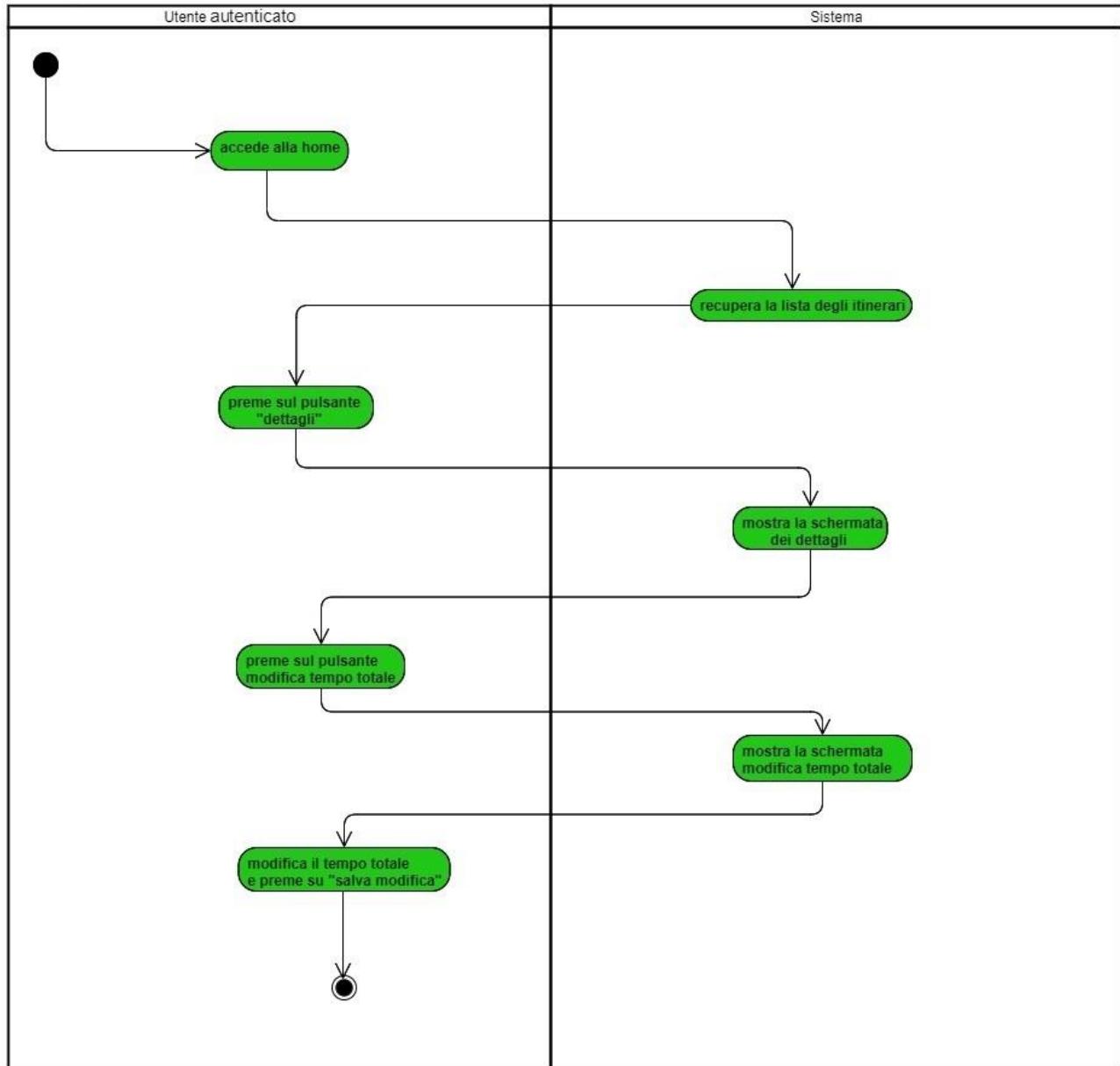


## 1.3.4 Activity Diagrams di analisi

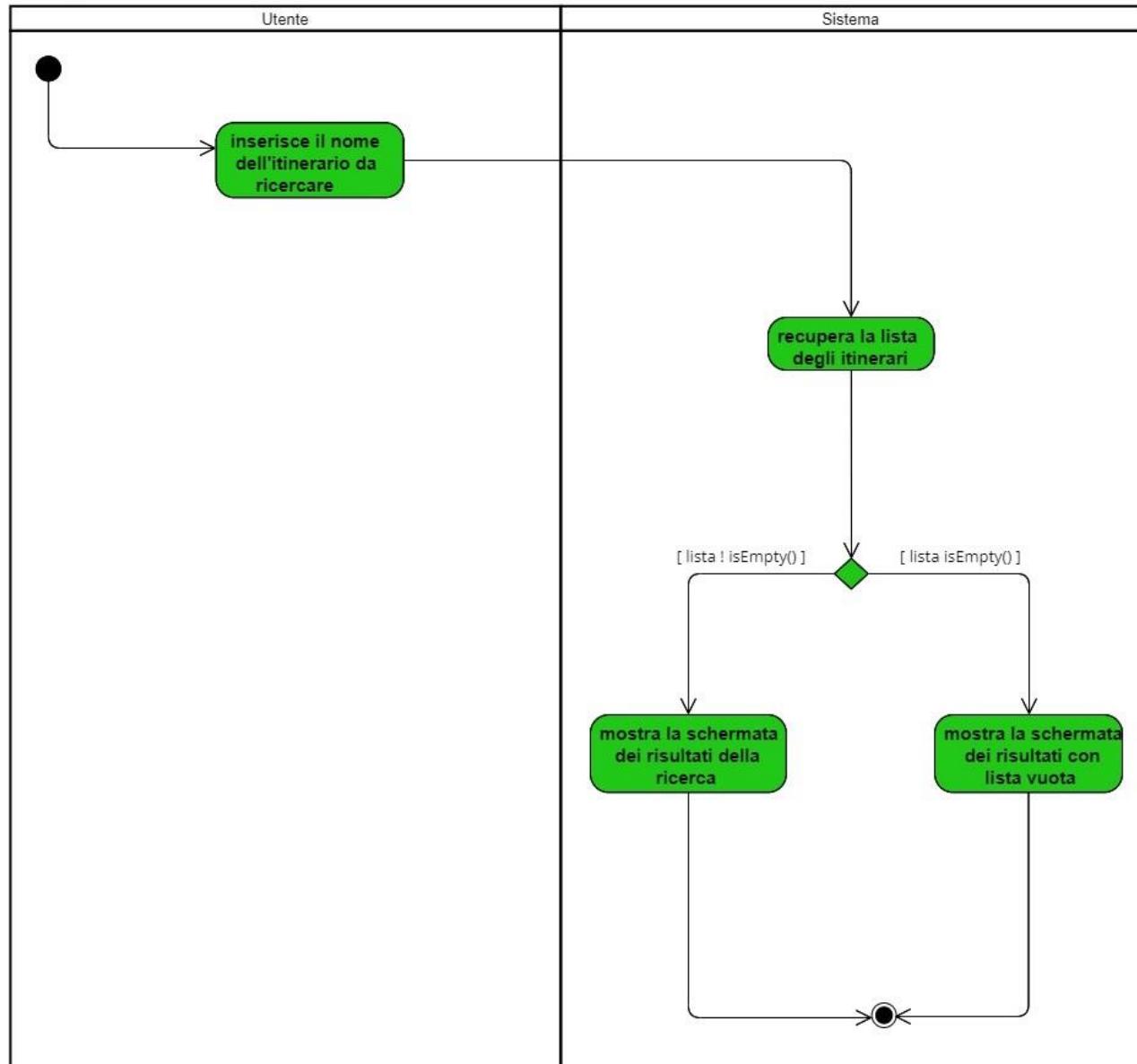
### Utente autenticato inserisce nuovo itinerario



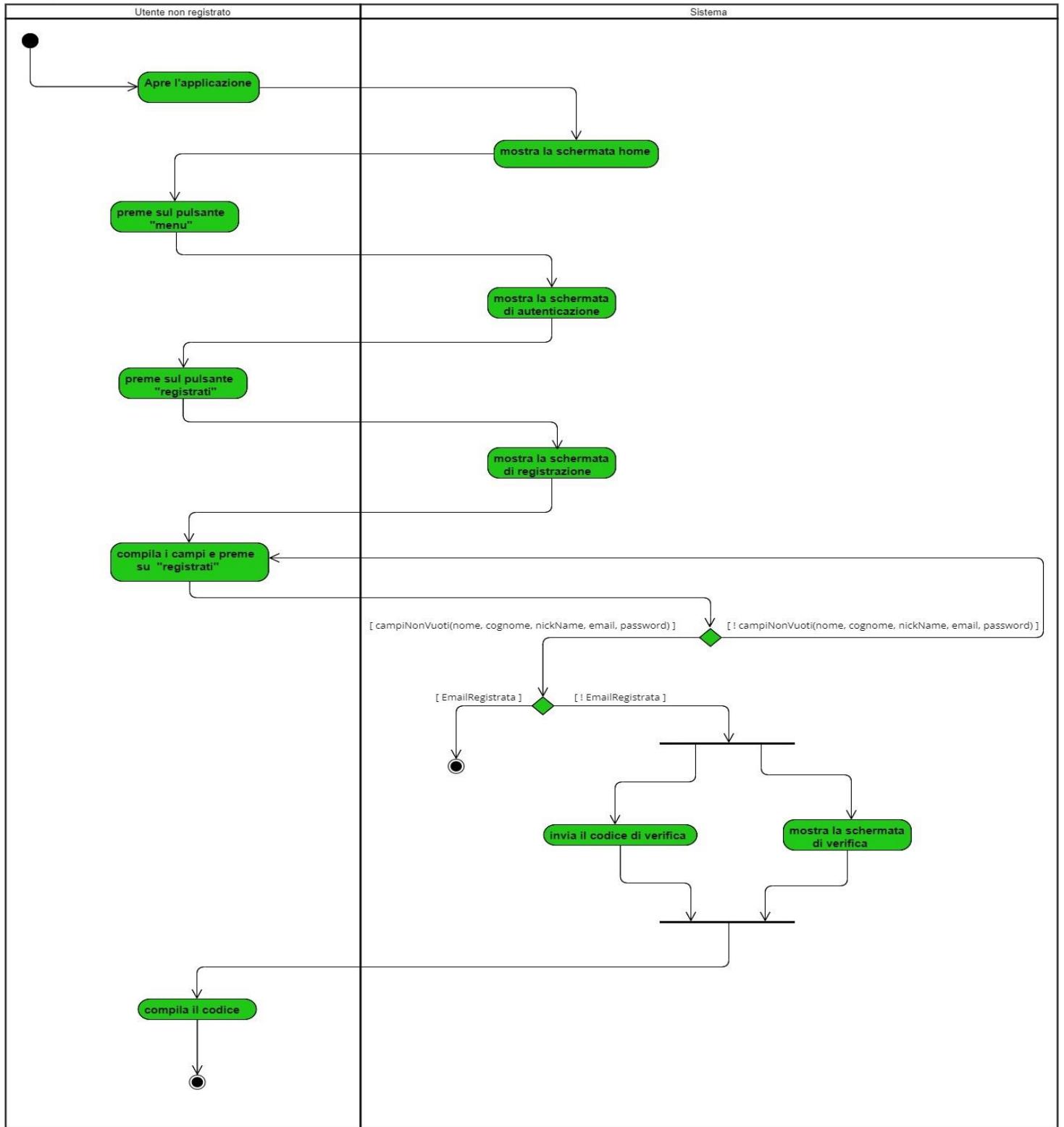
## Utente autenticato modifica tempo di percorrenza itinerario



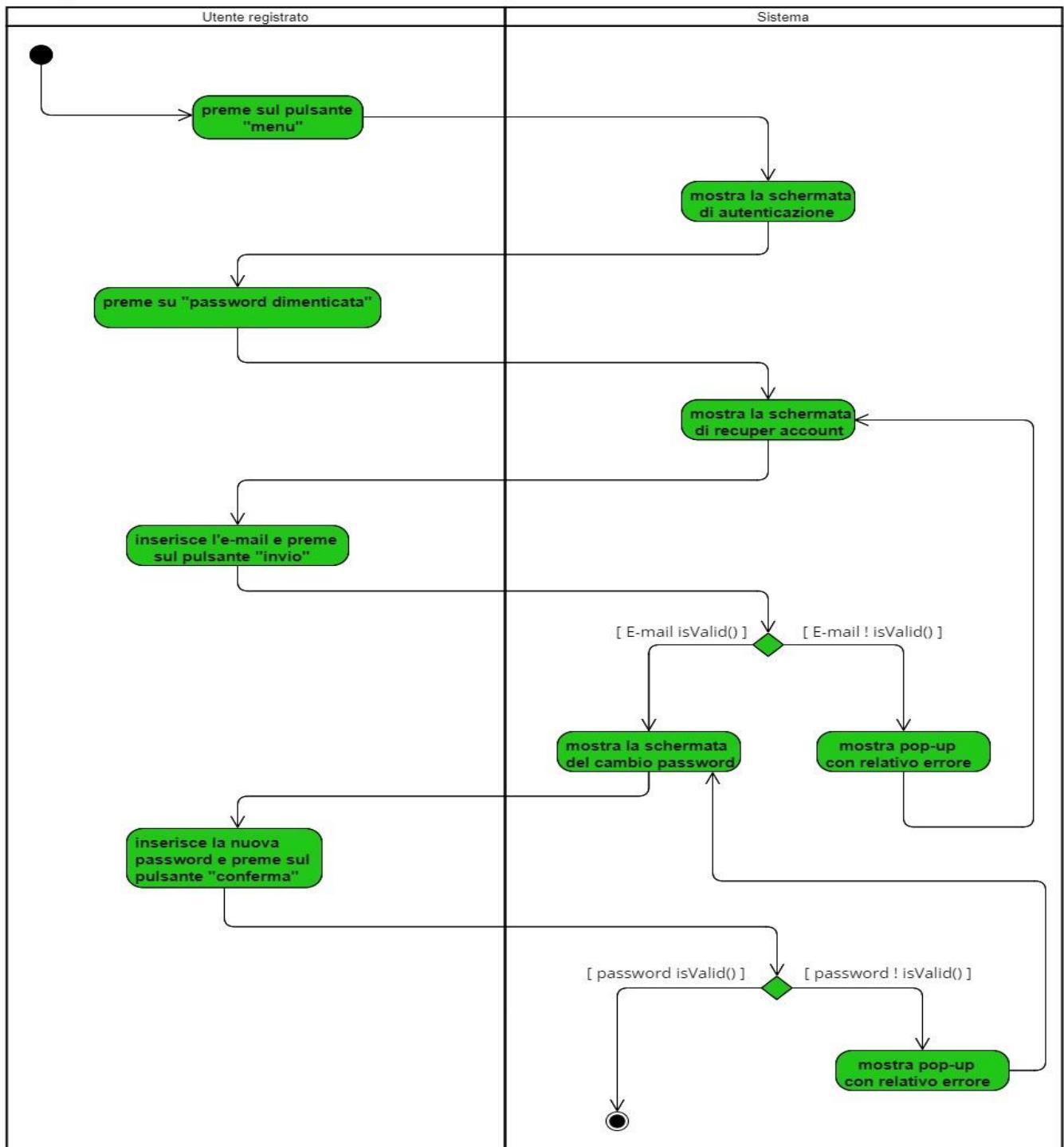
## Utente effettua ricerche itinerari



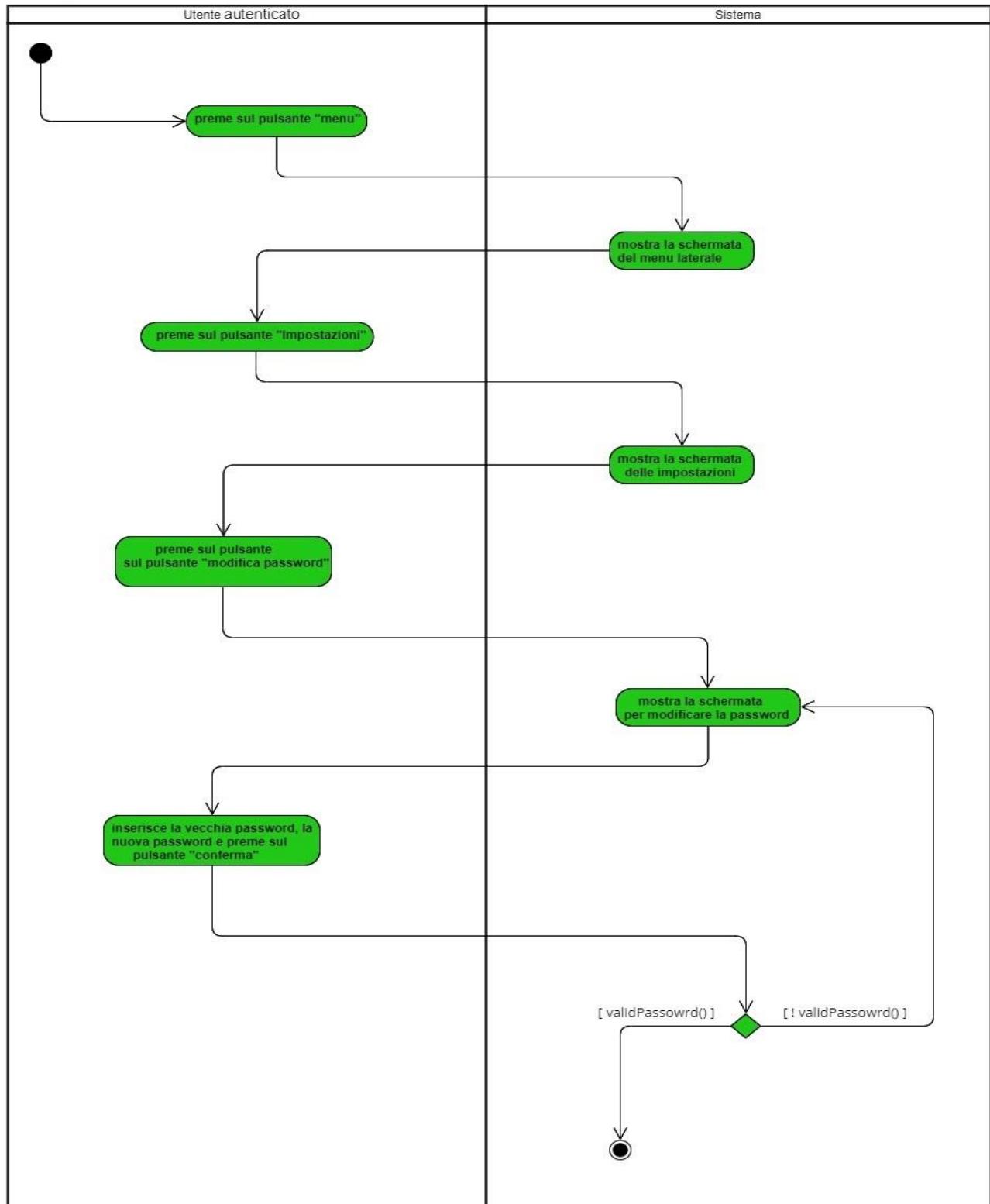
## Utente non registrato effettua registrazione



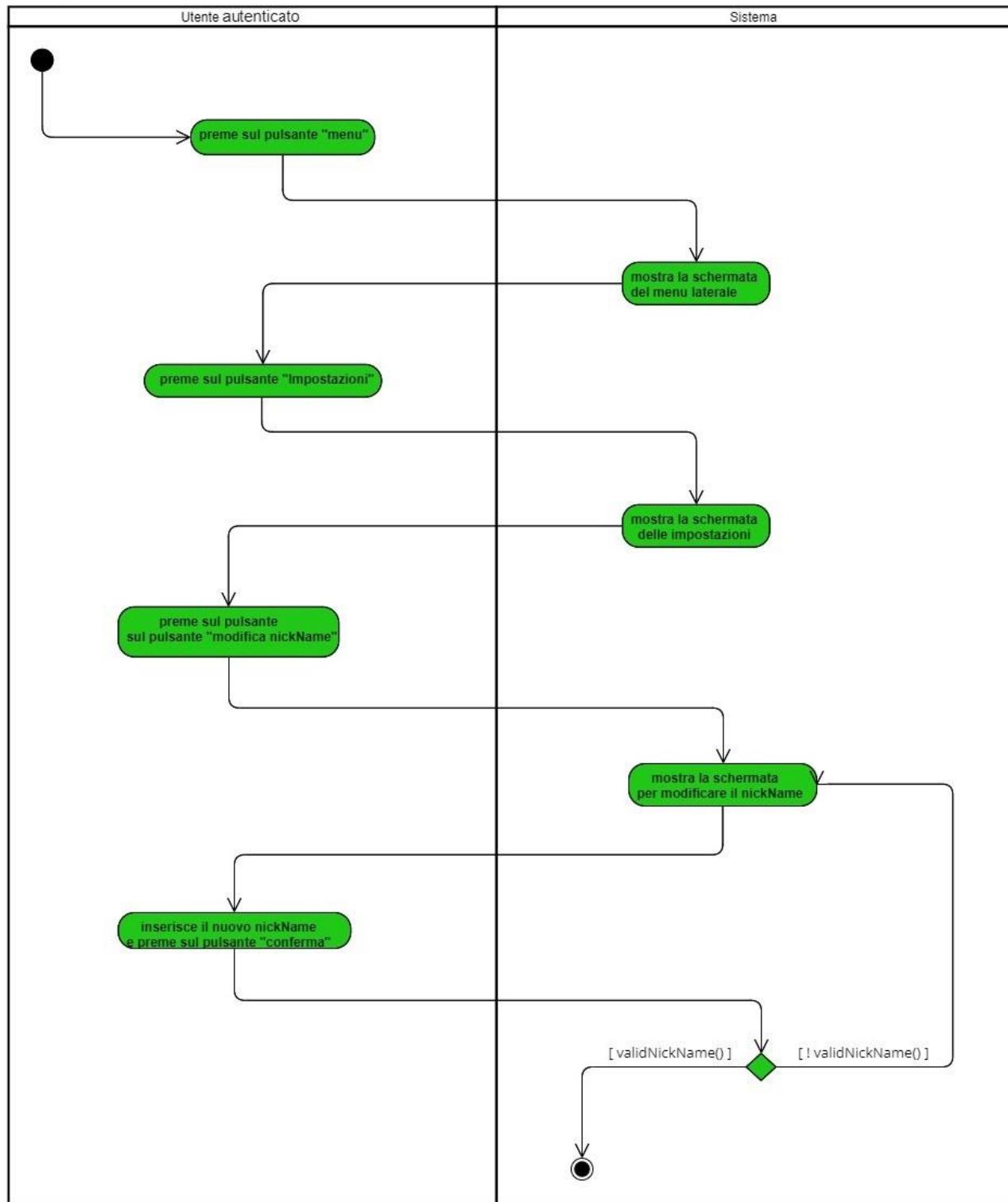
## Utente registrato recupera account



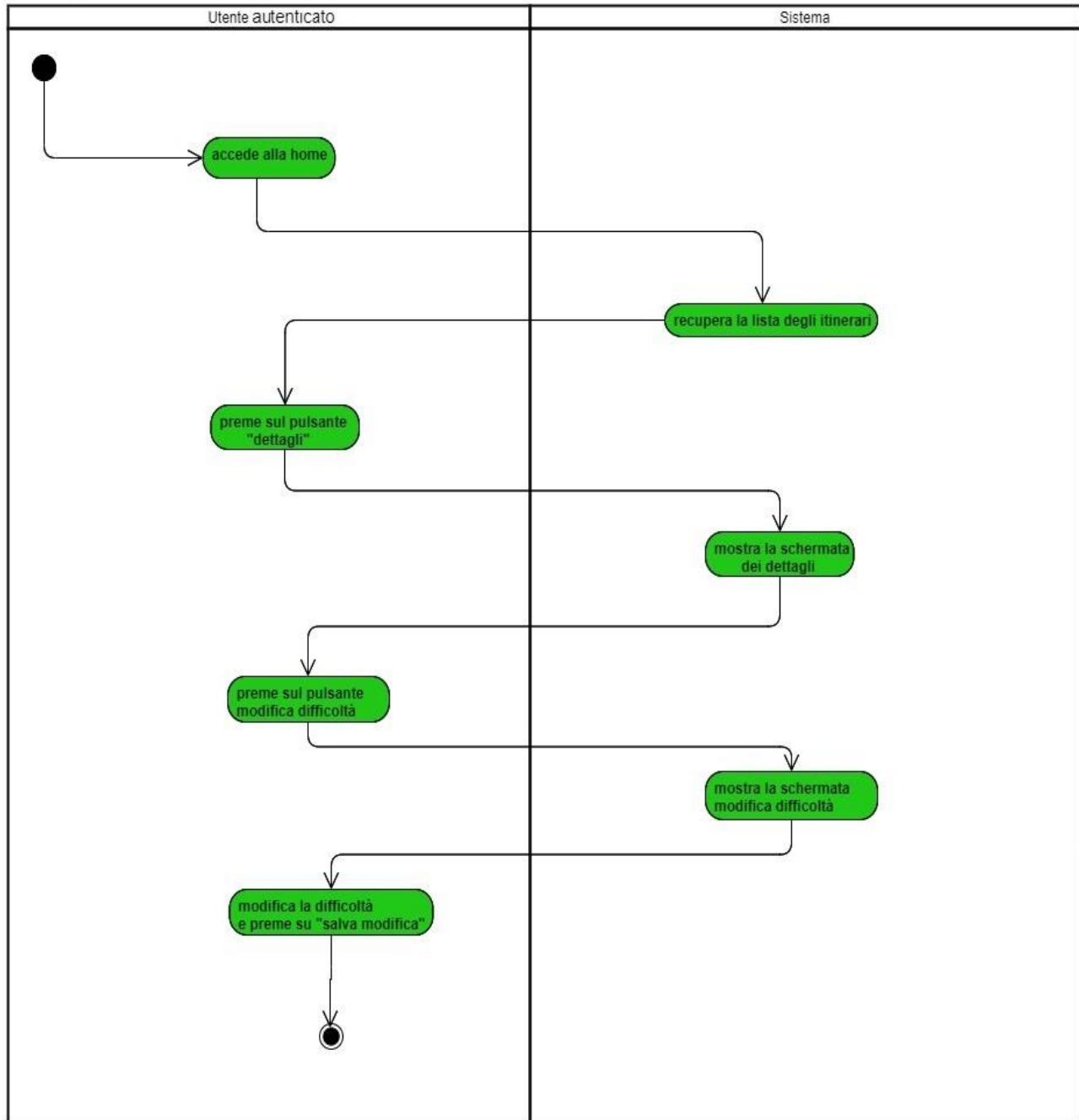
## Utente autenticato modifica password



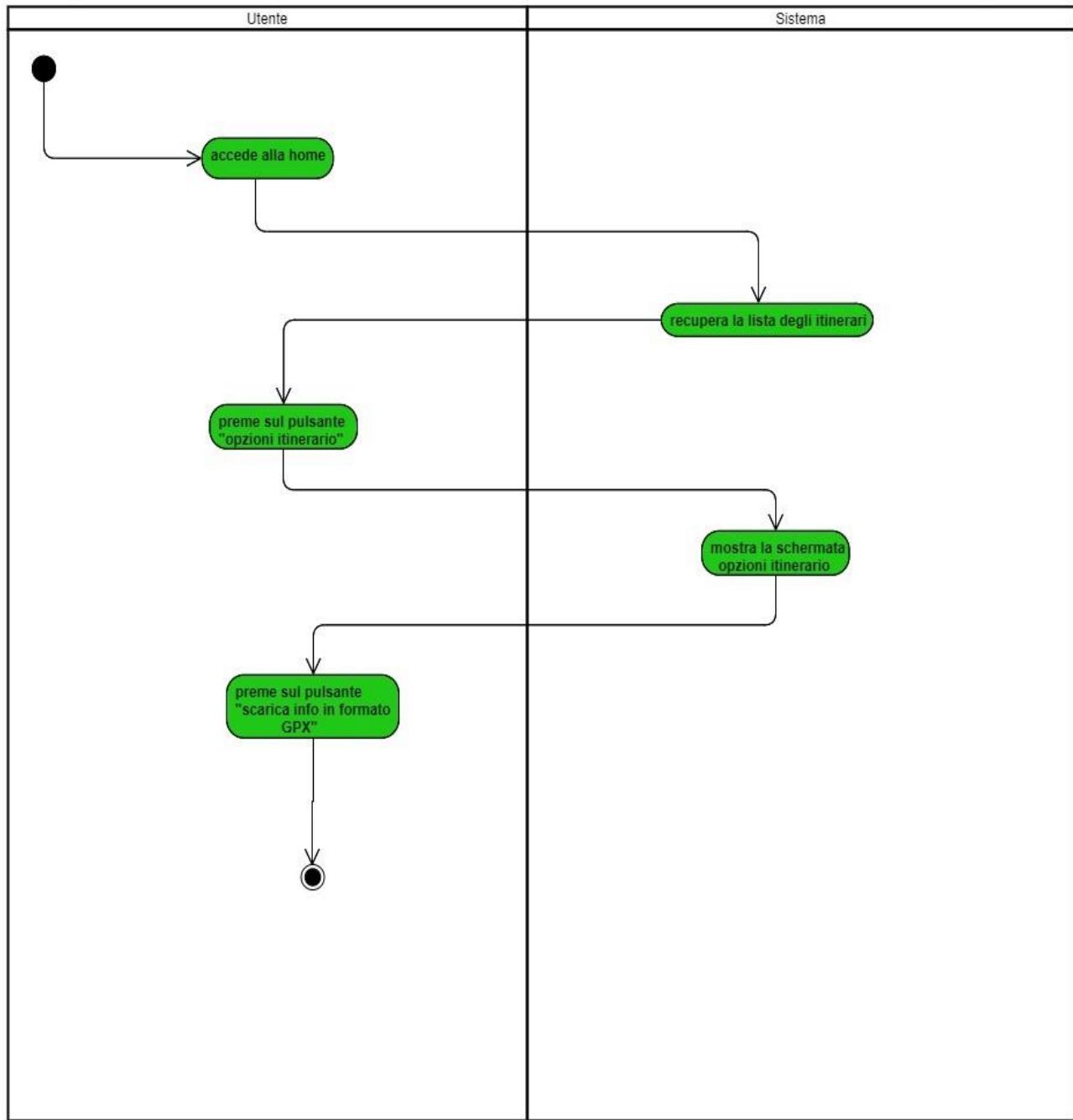
## Utente autenticato modifica NickName



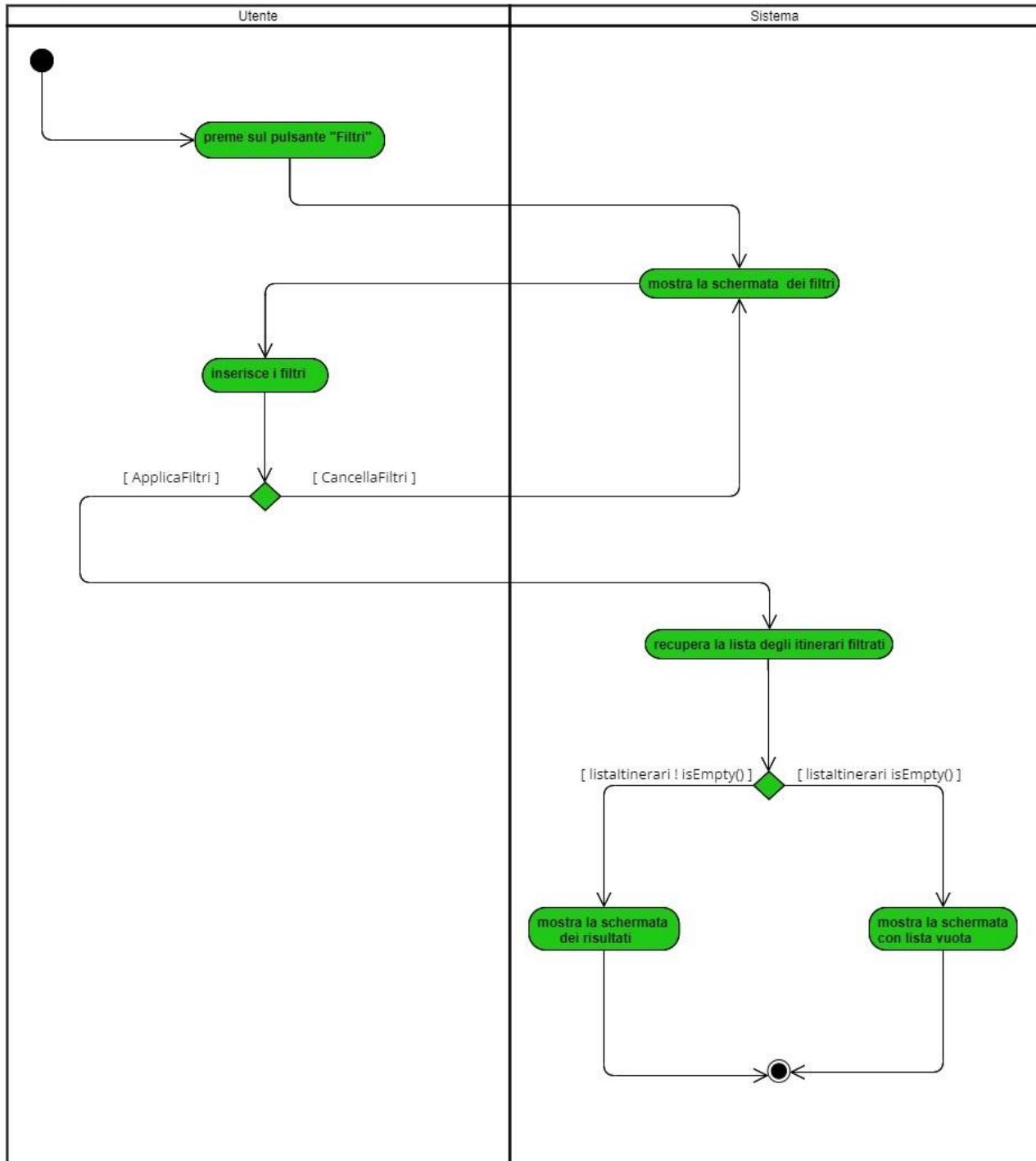
## Utente autenticato modifica difficoltà itinerario



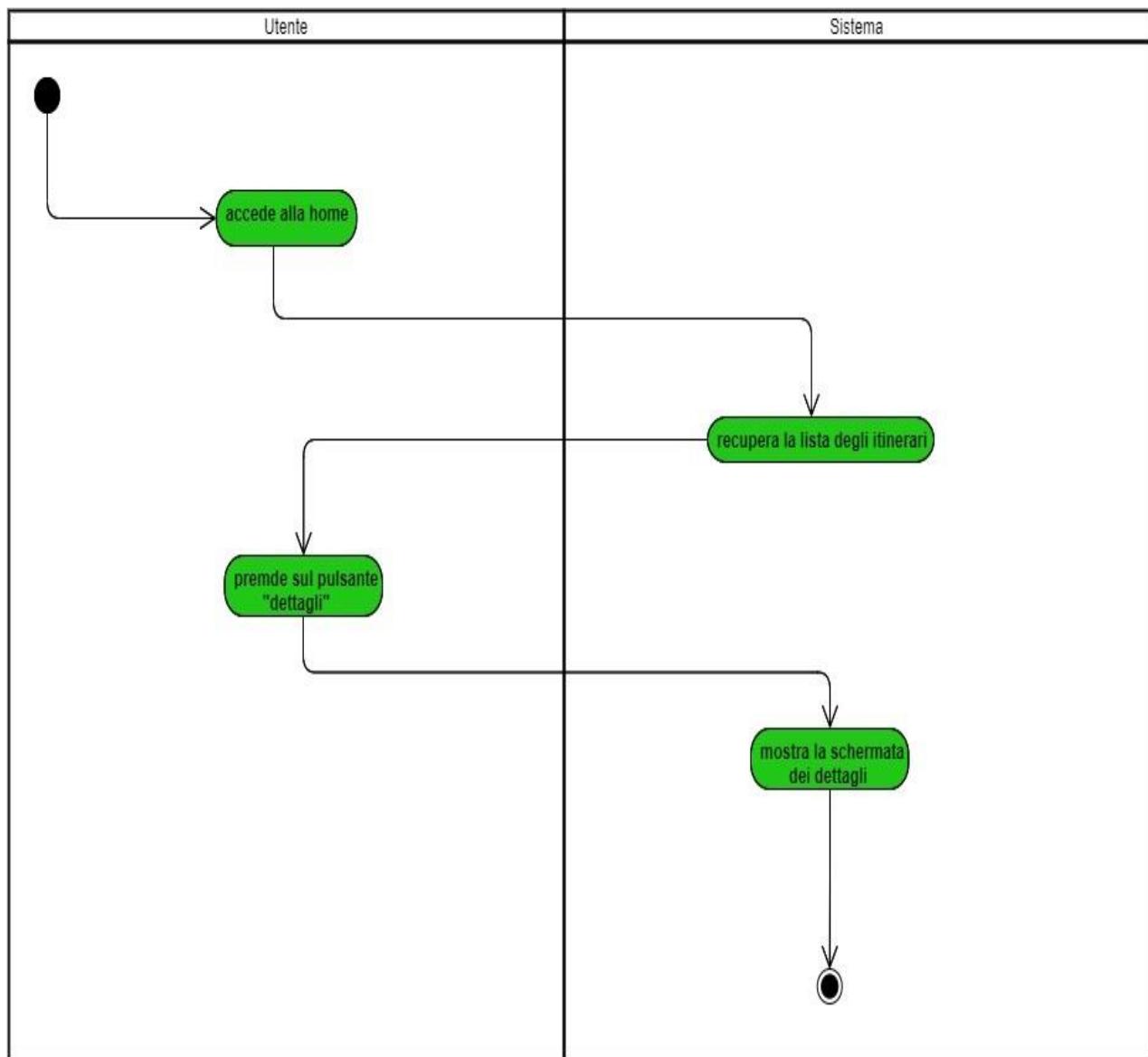
## Utente scarica informazioni sentiero in formato GPX



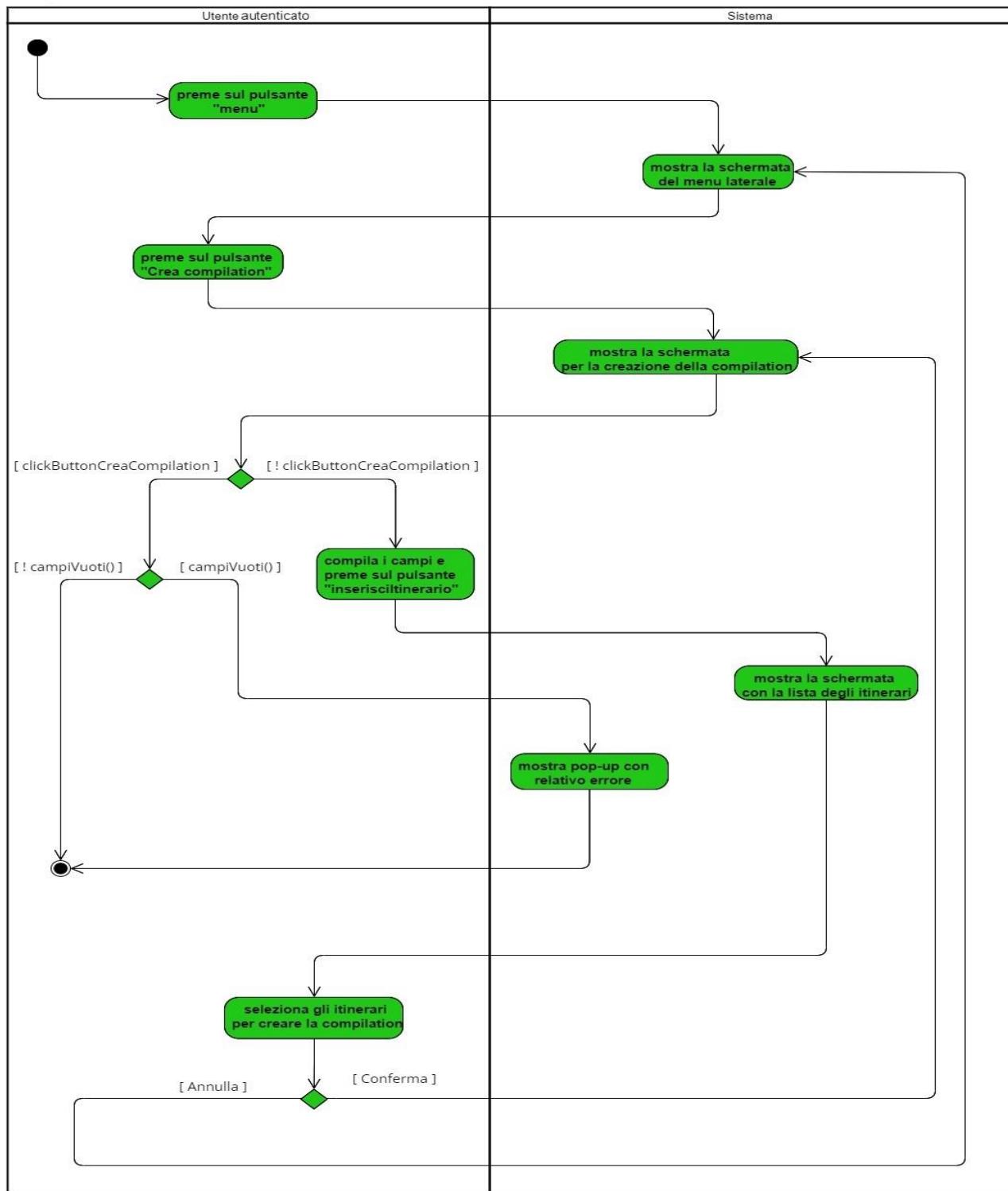
## Utente filtra ricerche itinerari



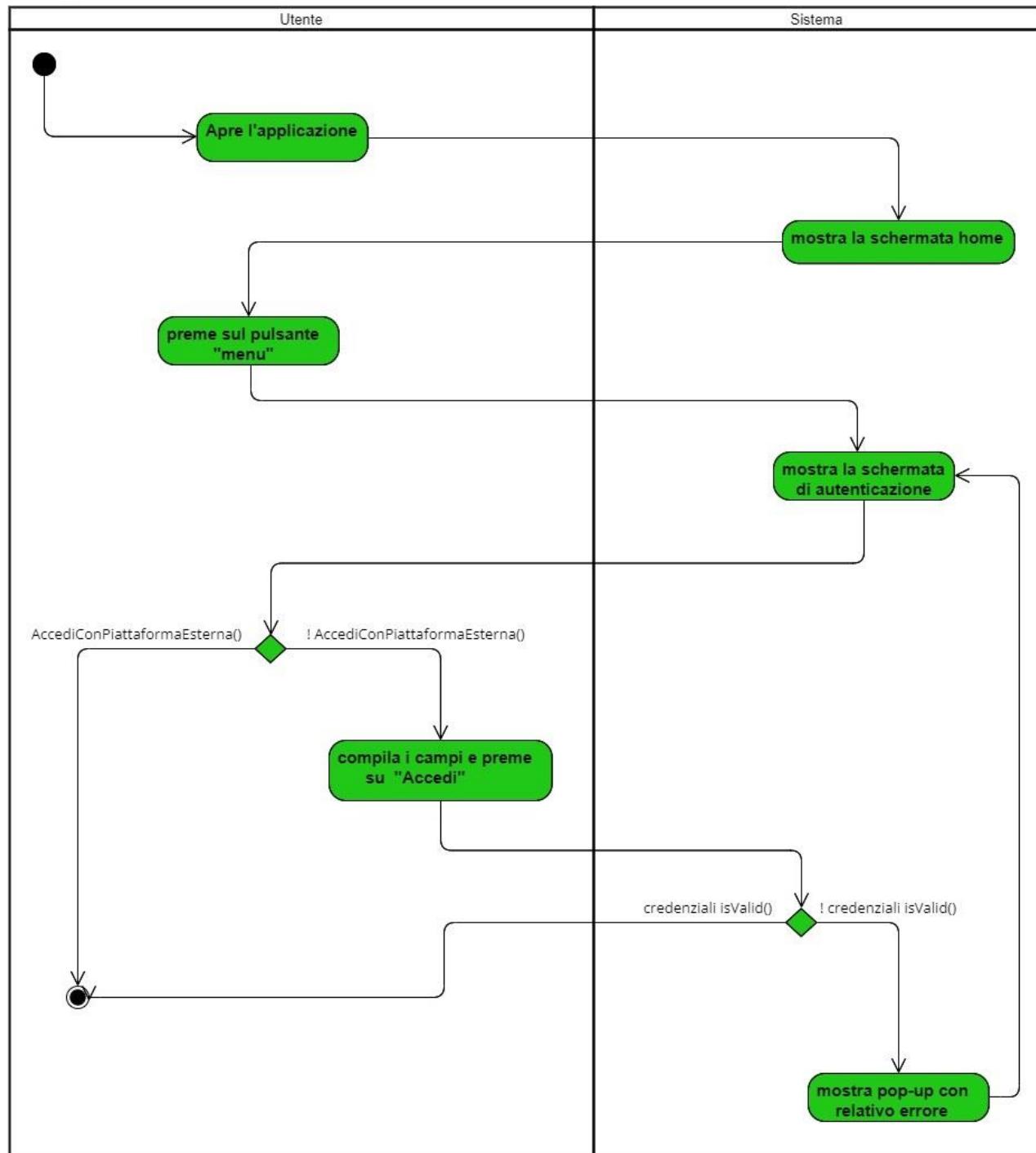
## Utente visualizza schermata di dettaglio



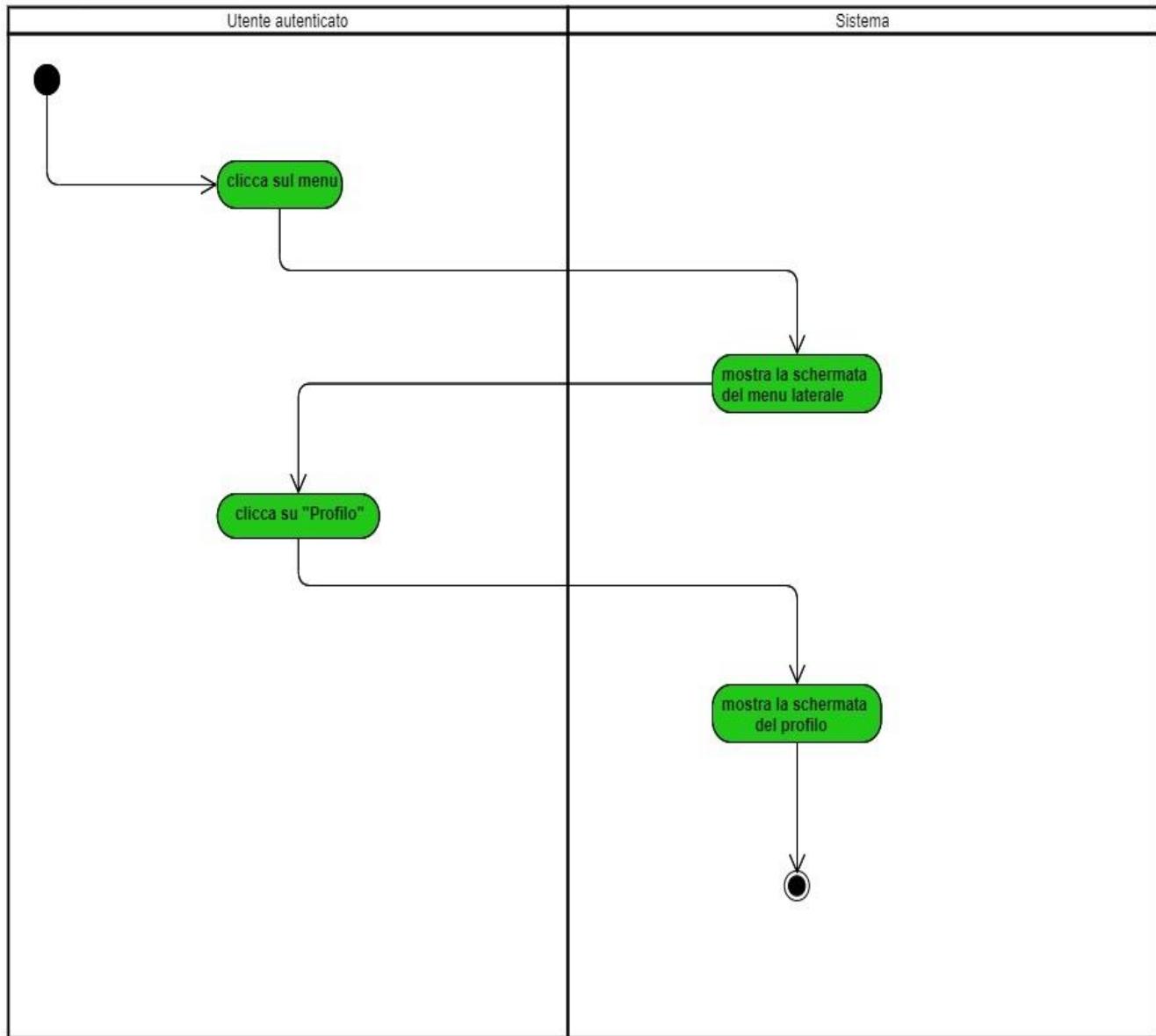
## Utente autenticato crea compilation di sentieri



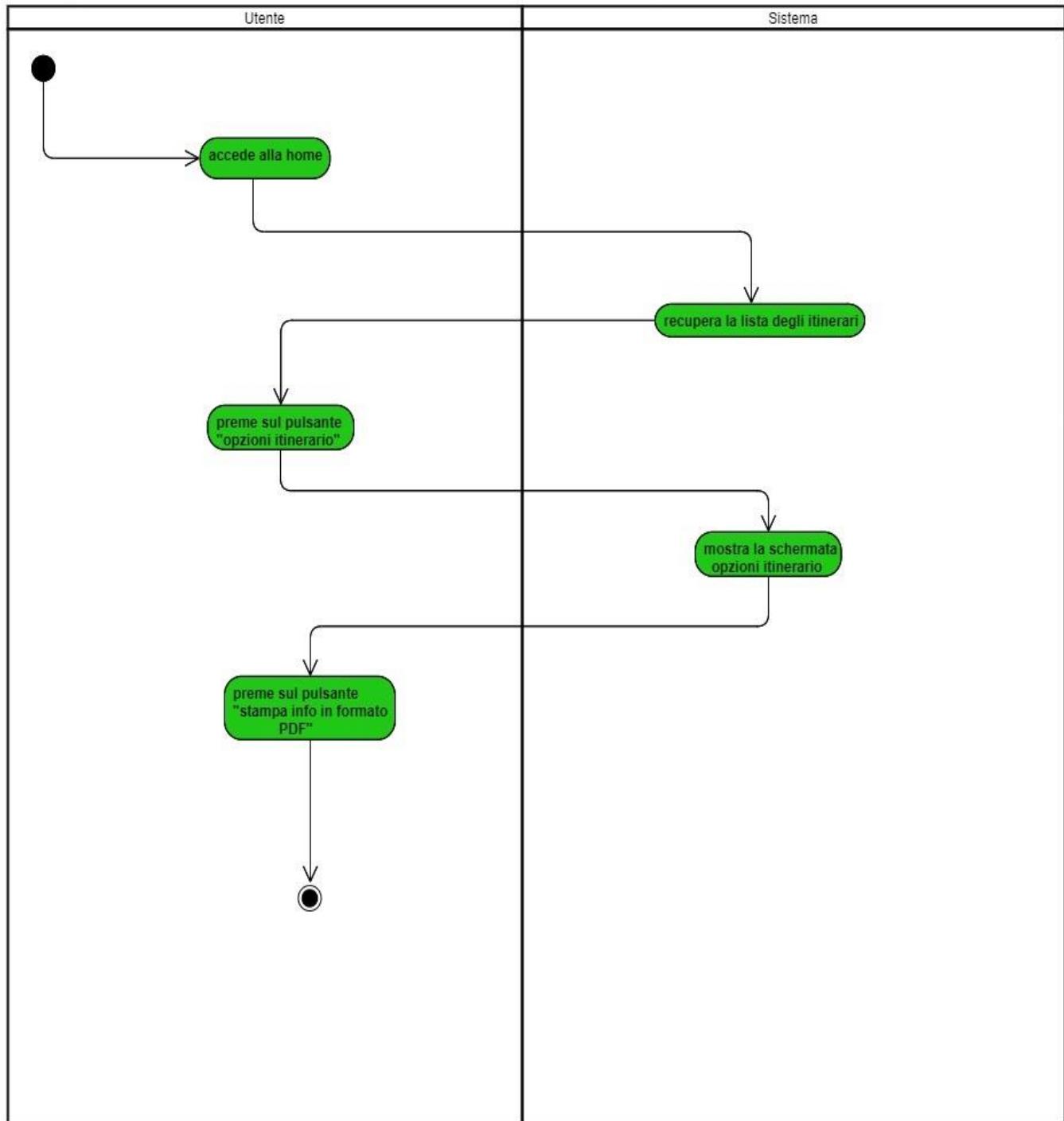
## Utente accede alla piattaforma



## Utente autenticato visualizza profilo



## Utente stampa informazioni itinerario in formato PDF

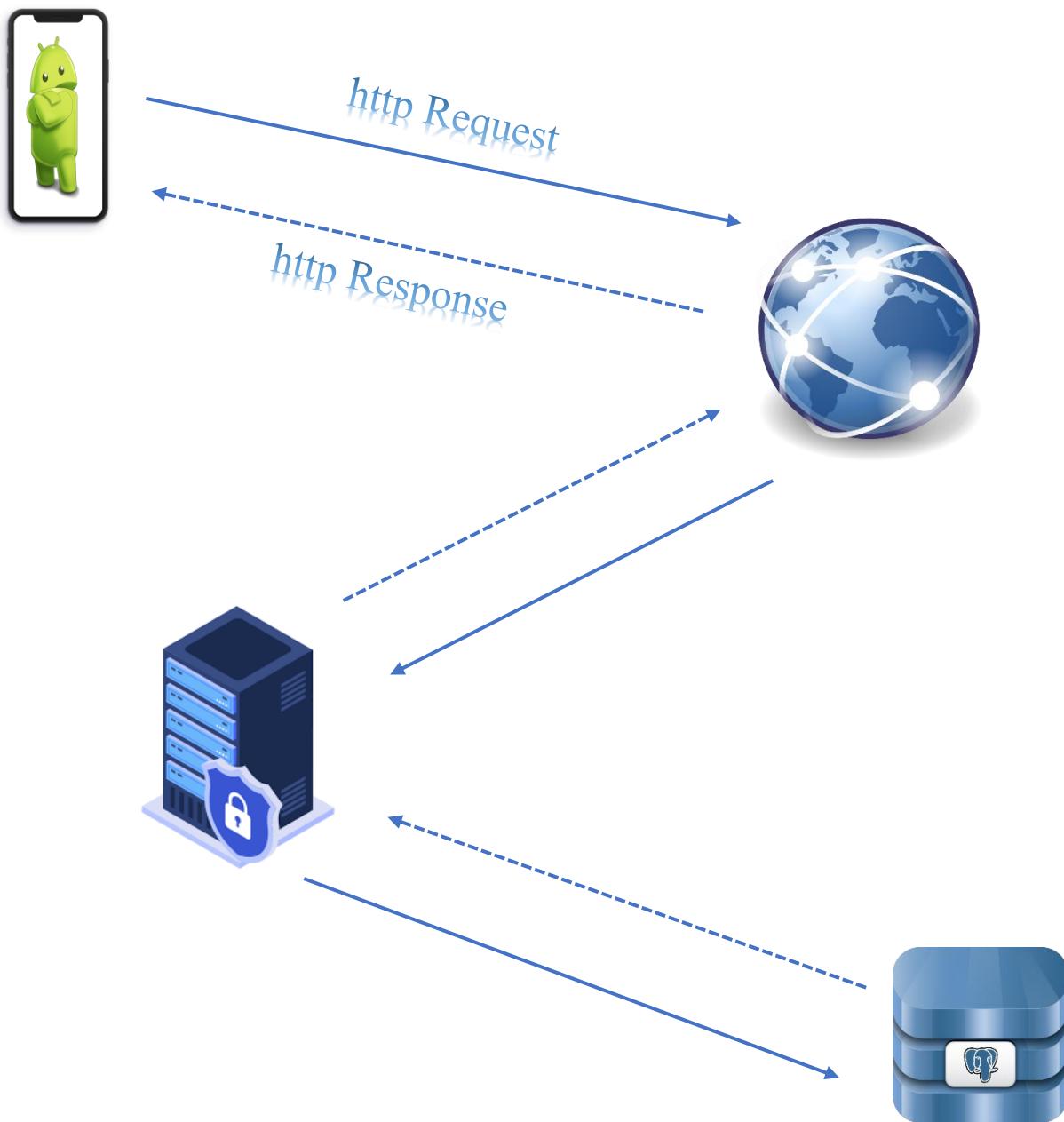


# CAPITOLO II

## DOCUMENTO DI DESIGN DEL SISTEMA

### 2.1 Analisi dell'architettura

*NaTour21* è un sistema composto da due componenti: un server, che funge da tramite per le richieste al database e un front-end mobile (Android).



Il client effettua unilateralmente richieste al back-end (che è sempre in ascolto) il quale, le elabora e le propaga al database. Il back-end offre dunque un modo sicuro per indicizzare e recuperare i dati, in quanto non consente chiamate dirette sulla base di dati. Con questo tipo di architettura è possibile, inoltre, modificare il metodo di persistenza dei dati sostituendo il vecchio modulo con quello nuovo, oppure aggiungere in futuro ulteriori client, senza dover riprogettare interamente la struttura attuale. Tutti gli applicativi sono stati scritti in linguaggio Java. Di seguito vengono illustrate le descrizioni delle tecnologie adottate per ciascun componente.

## Architettura Client

L'applicativo mobile è stato realizzato per il sistema operativo Android con API di versione minima 22.

L'architettura Client di *NaTour21* è suddivisa in livelli, per avere una migliore organizzazione e comprensione del funzionamento.

I livelli effettuano richieste in maniera *top-down*, ogni livello si serve esclusivamente di quello successivo, riportando l'esito a quello precedente, in modo tale da avere un'architettura chiusa. Ossia i Client non hanno la possibilità di interagire direttamente con il database, evitando così eventuali richieste dannose per il sistema.

**1. Data Layer** : fa da interfaccia verso i servizi Server di AWS con le opportune API, richiedendo e ottenendo i dati.

**2. Presentation Layer** : si occupa di come presentare i dati ottenuti dal livello inferiore, trasmettendoli all'interfaccia grafica, e inoltra verso il basso le richieste effettuate dall'utente.

**3. UI Layer** : raccoglie tutte le classi deputate all'interfaccia grafica; inoltra verso il basso le richieste effettuate dall'utente.

## Architettura Server

Il server è stato realizzato con il supporto del framework **Spring Boot** che offre la configurazione automatica di tutta una serie di servizi offerti dal framework.

### Dettagli sui singoli moduli che compongono l'applicativo Server:

- **Controller** : è responsabile del controllo della trasmissione dei dati tra il client e il service.
- **Service**: contiene tutta la logica di gestione delle operazioni utilizzate dalle classi controller corrispondenti;
- **ServiceInterface**: contiene le astrazioni delle classi presenti in service
- **Repository**: rappresenta il collegamento messo a disposizione da Spring boot tra l'applicativo java e il database corrispondente;

## Panoramica dei servizi esterni

Segue una breve sintesi dei servizi utilizzati a supporto dell'intera architettura:

- **Facebook API**: per l'autenticazione degli utenti con il loro account Facebook.
- **Google API**: per l'autenticazione degli utenti con il loro account Google.
- **Amazon Cognito**: fornisce autenticazione, autorizzazione e gestione degli utenti
- **Amazon EC2**: per l'hosting del server.
- **Amazon RDS con motore PostgreSQL**: per la persistenza dei dati.
- **Amazon Elastic Ip**: fornisce un indirizzo ip statico.
- **MapBox**: fornisce una mappa globale, localizzazione in tempo reale, ricerca della posizione e navigazione.
- **Retrofit**: per la comunicazione con il web server.
- **Sprint Boot**: per il supporto alla realizzazione del server.

## Design del Sistema

La fase di design di *NaTour21* si è basata sull'utilizzo di diversi **Design Patterns**. I Design Patterns sono delle soluzioni comuni a dei problemi noti e ricorrenti in questa fase del ciclo di vita del software; ogni Design Pattern è costituito da 4 parti:

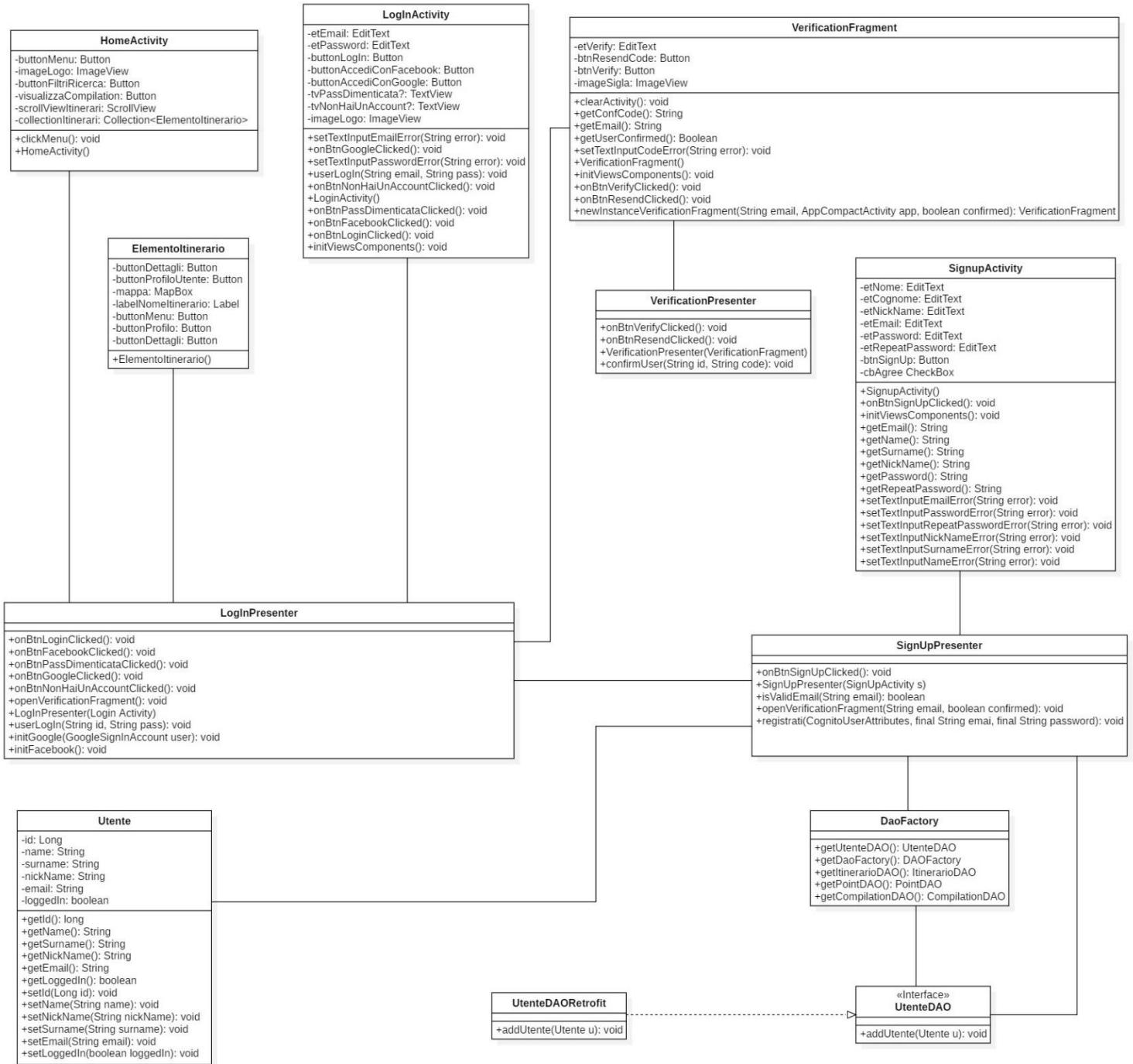
1. **Nome:** identifica univocamente quel Design Pattern.
2. **Problema :** ha dato l'idea per la costruzione di quel Design Pattern.
3. **Soluzione:** la maniera in cui il Design Pattern si comporta per risolvere il problema.
4. **Conseguenze:** vantaggi e svantaggi in System Design conseguenti all'applicazione del Design Pattern.

Per la realizzazione del progetto software sono stati utilizzati tre noti Design Patterns:

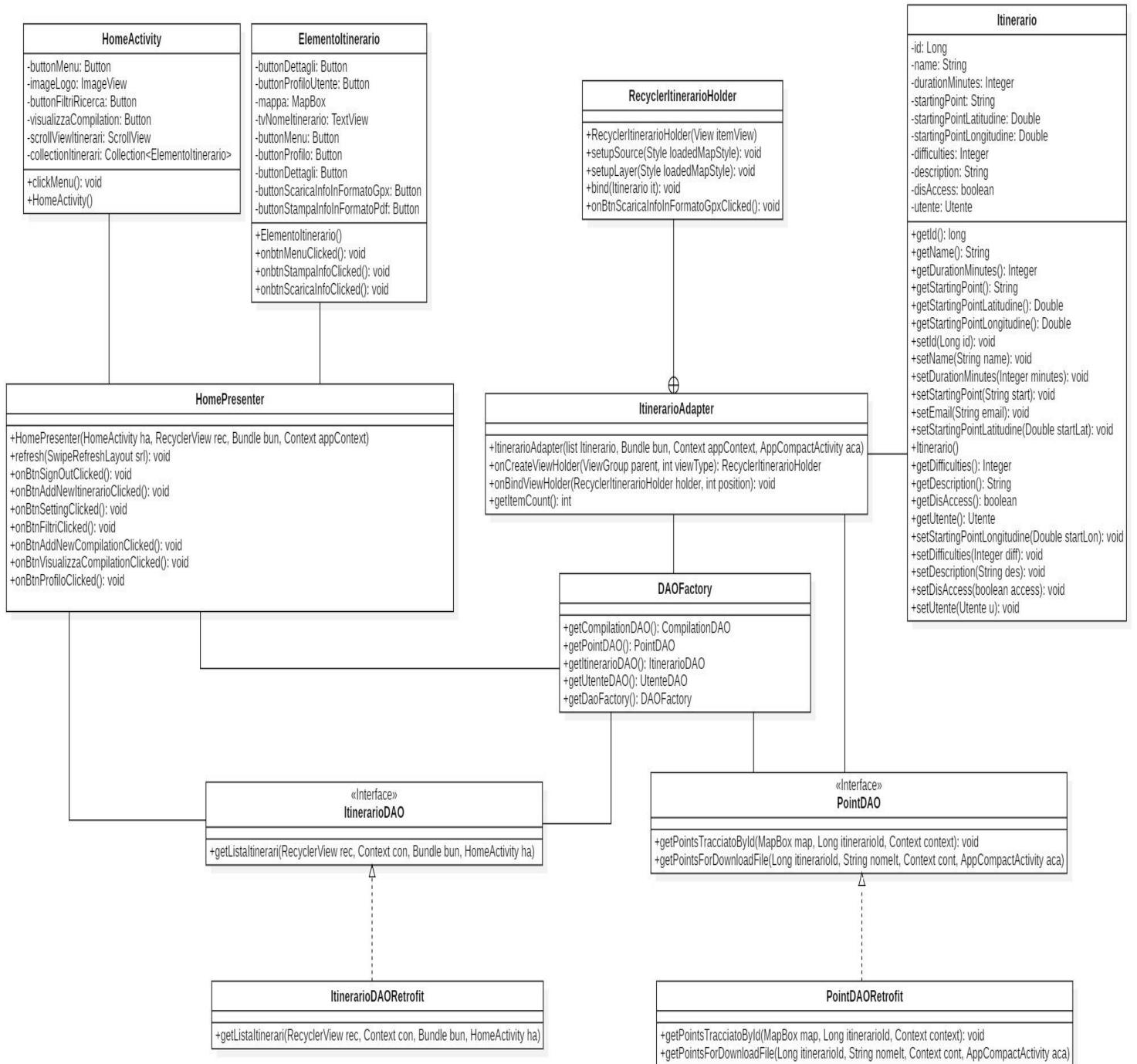
- **Singleton:** la sua applicazione assicura che una determinata classe venga creata una sola volta, fornendo una vista globale di essa all'applicativo.  
È stato utilizzato per la classe *DAOFactory*, essendo una classe orientata a svolgere prettamente operazioni, senza avere uno stato preciso.
- **Factory Method:** si occupa della logica di creazione per alcuni oggetti, non esponendo altre parti dell'applicativo a questa responsabilità. Questi oggetti verranno poi castati ad una interfaccia comune.  
L'utilizzo di questo Design Pattern è stato fondamentale per la gestione delle classi DAO (altro Pattern spiegato in seguito): per garantire una migliore manutenibilità del software, è stata aggiunta la classe *DAOFactory*, la quale si occupa di istanziare la giusta implementazione dell'interfaccia *<entità>DAO*, in base al servizio di recupero dati correntemente usato.
- **DAO (Data Access Object):** l'uso di questo Design Pattern prevede l'implementazione di una classe *<entità>DAO<servizio>*, la quale si interfaccia con la base dati e ha i metodi appropriati per il servizio scelto, essendo essi molto diversi tra loro, e l'entità coinvolta. Questo permette un facile cambio di servizio, qualora ce ne fosse il bisogno, essendo tutte implementazioni dell'interfaccia *<entità>DAO*.

## 2.2 Class Diagrams di design

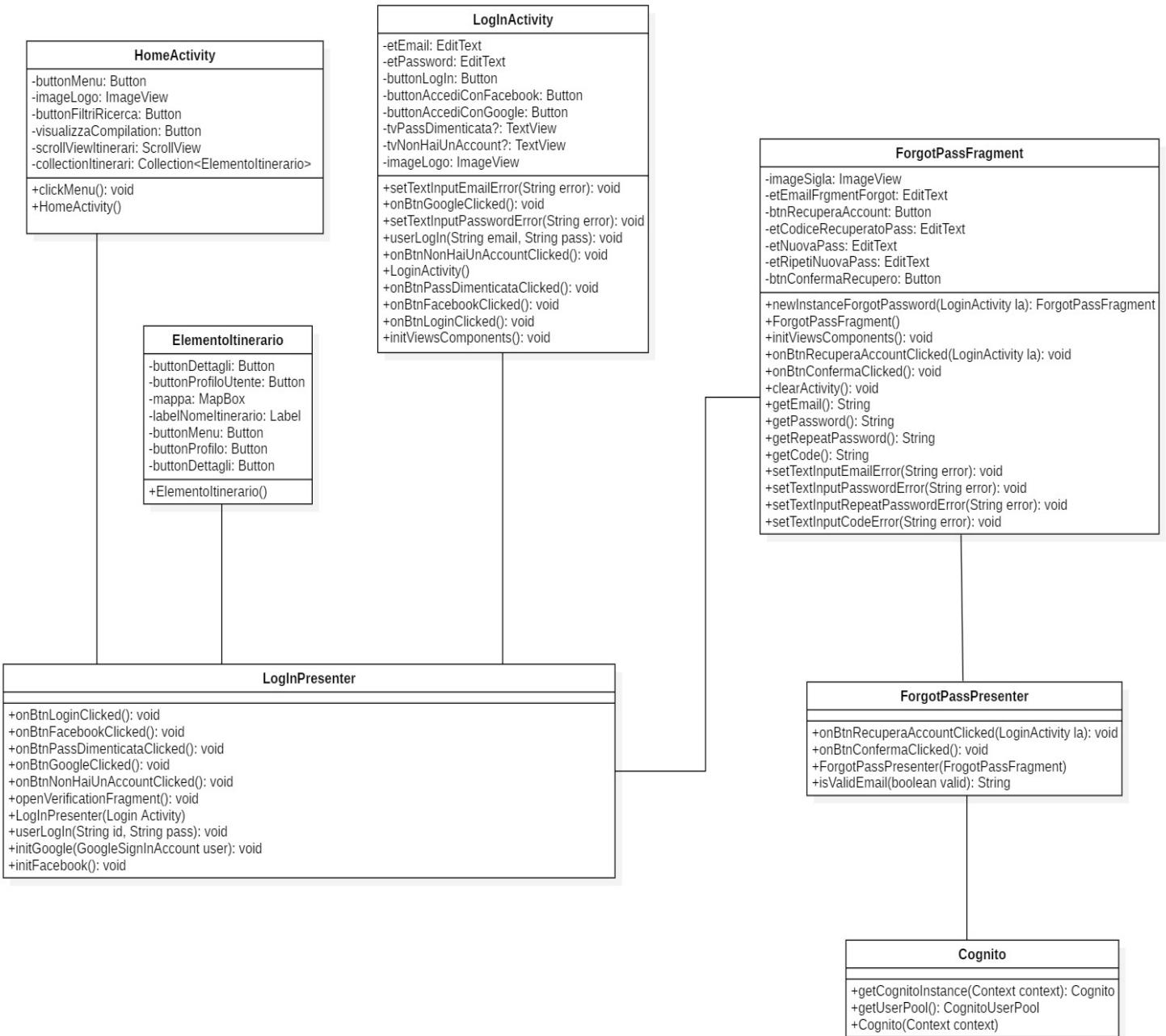
### Utente non registrato effettua registrazione



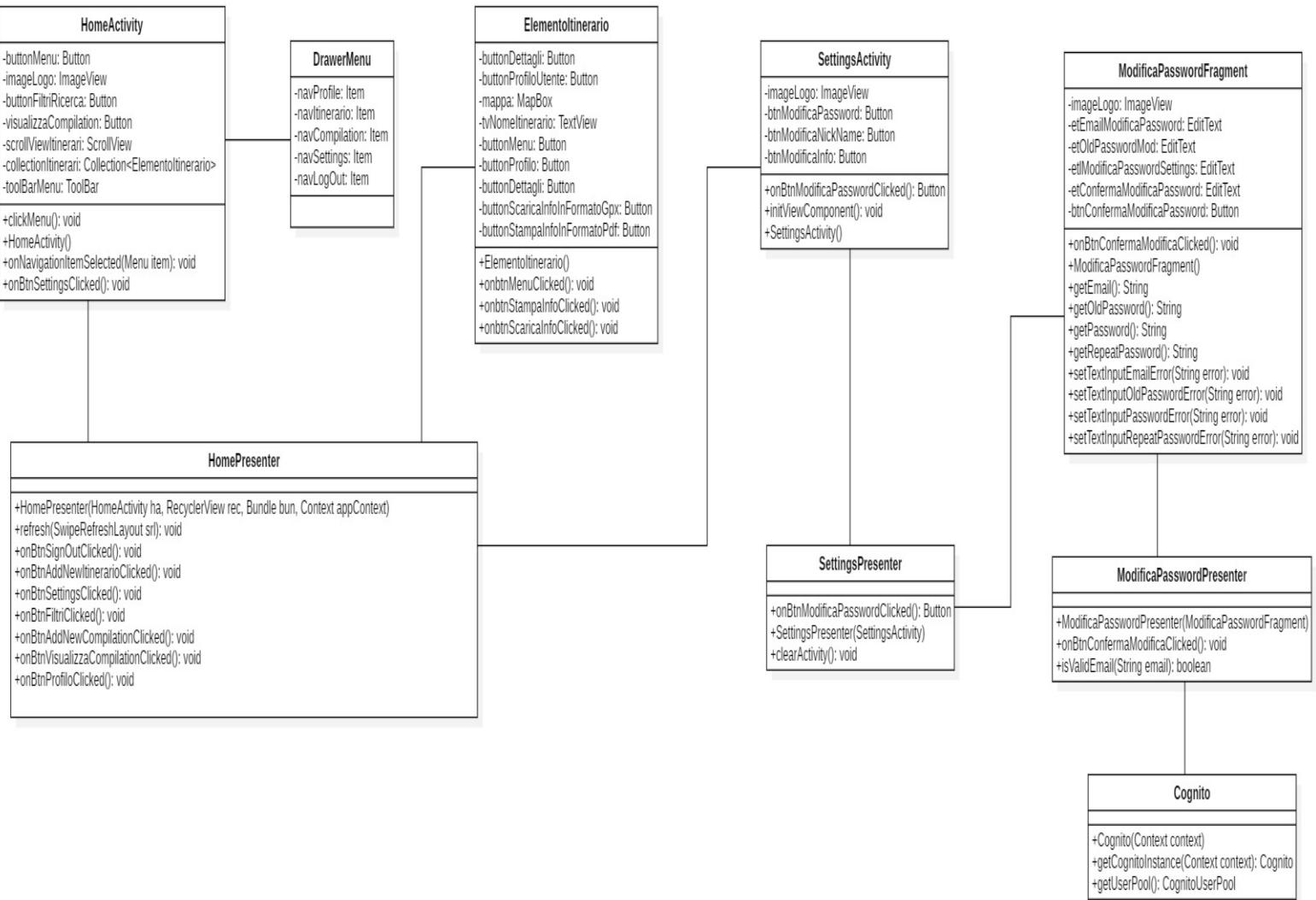
## Utente Scarica Informazioni Itinerario in formato GPX



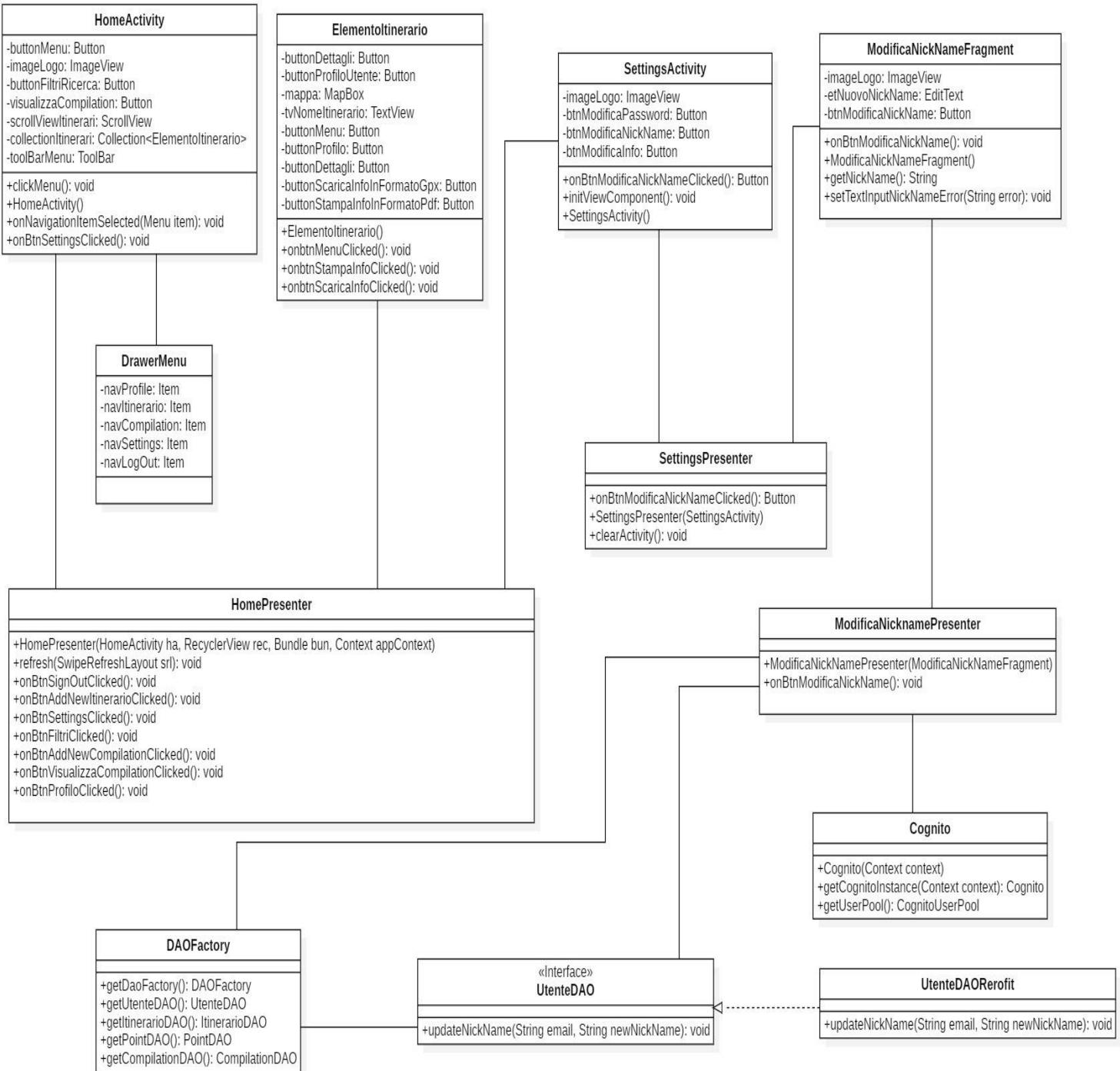
## Utente registrato recupera account



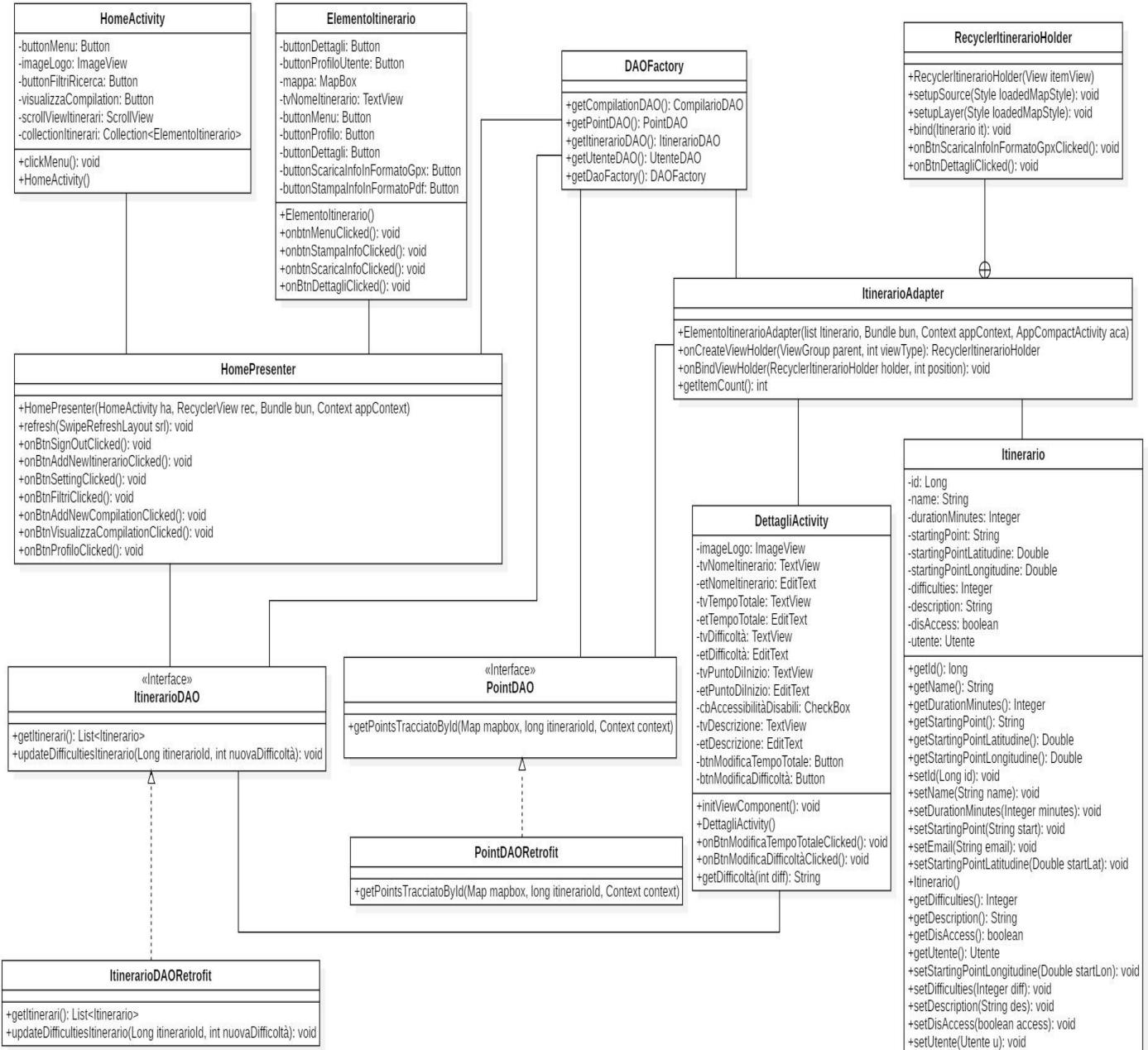
## Utente registrato modifica password



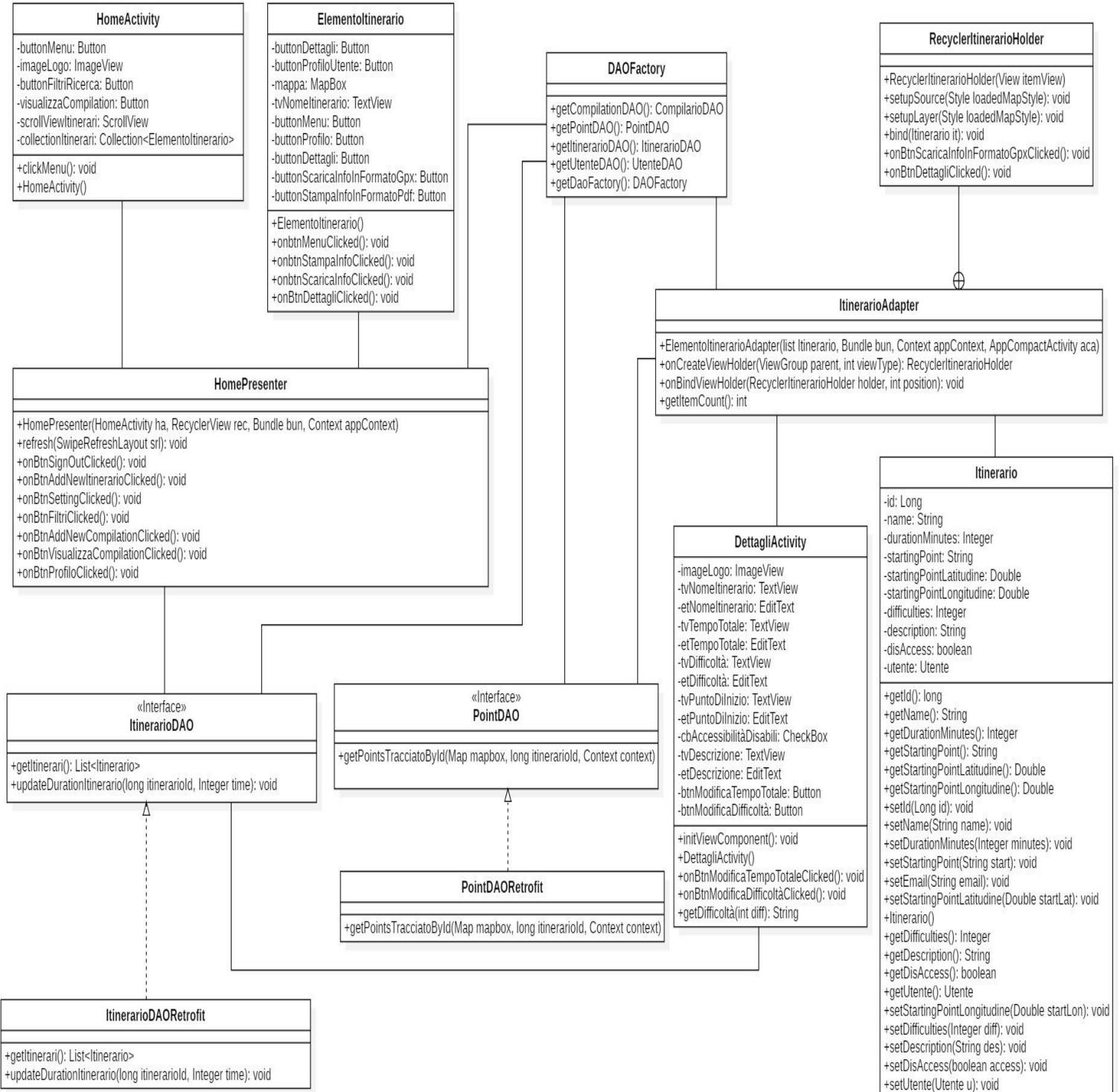
## Utente registrato modifica nickname



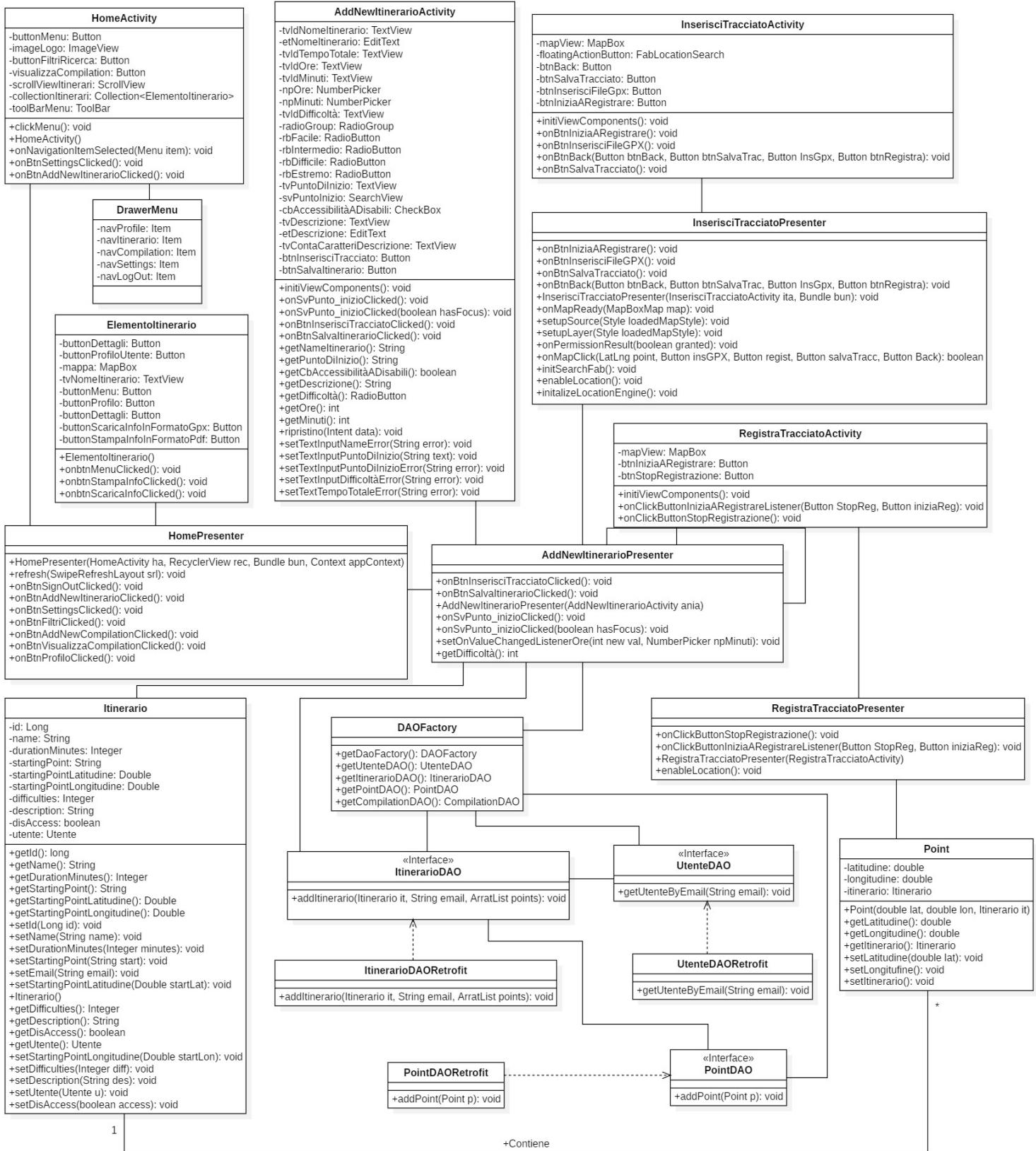
## Utente autenticato modifica difficoltà itinerario



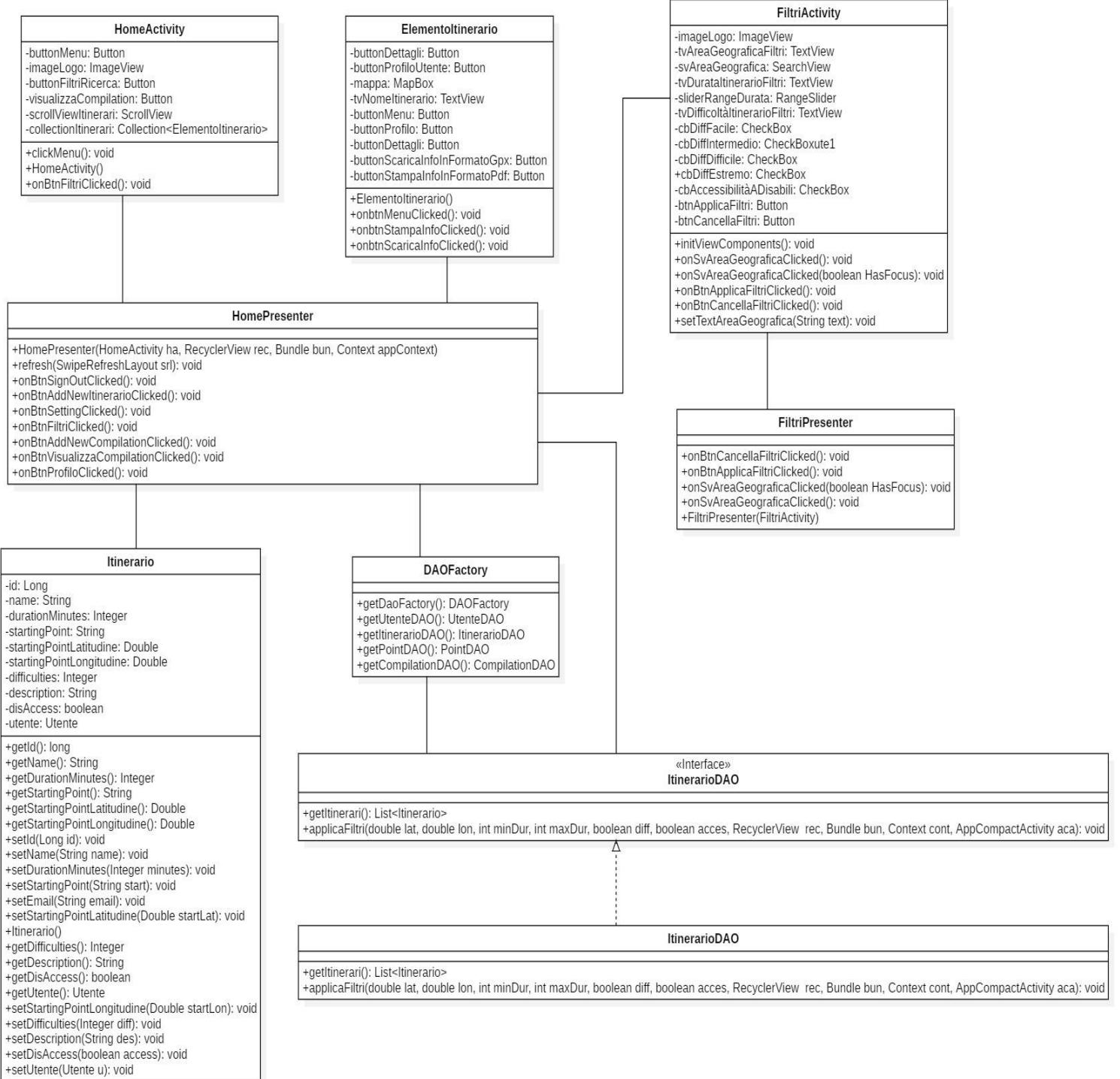
## Utente autenticato modifica tempo di percorrenza itinerario



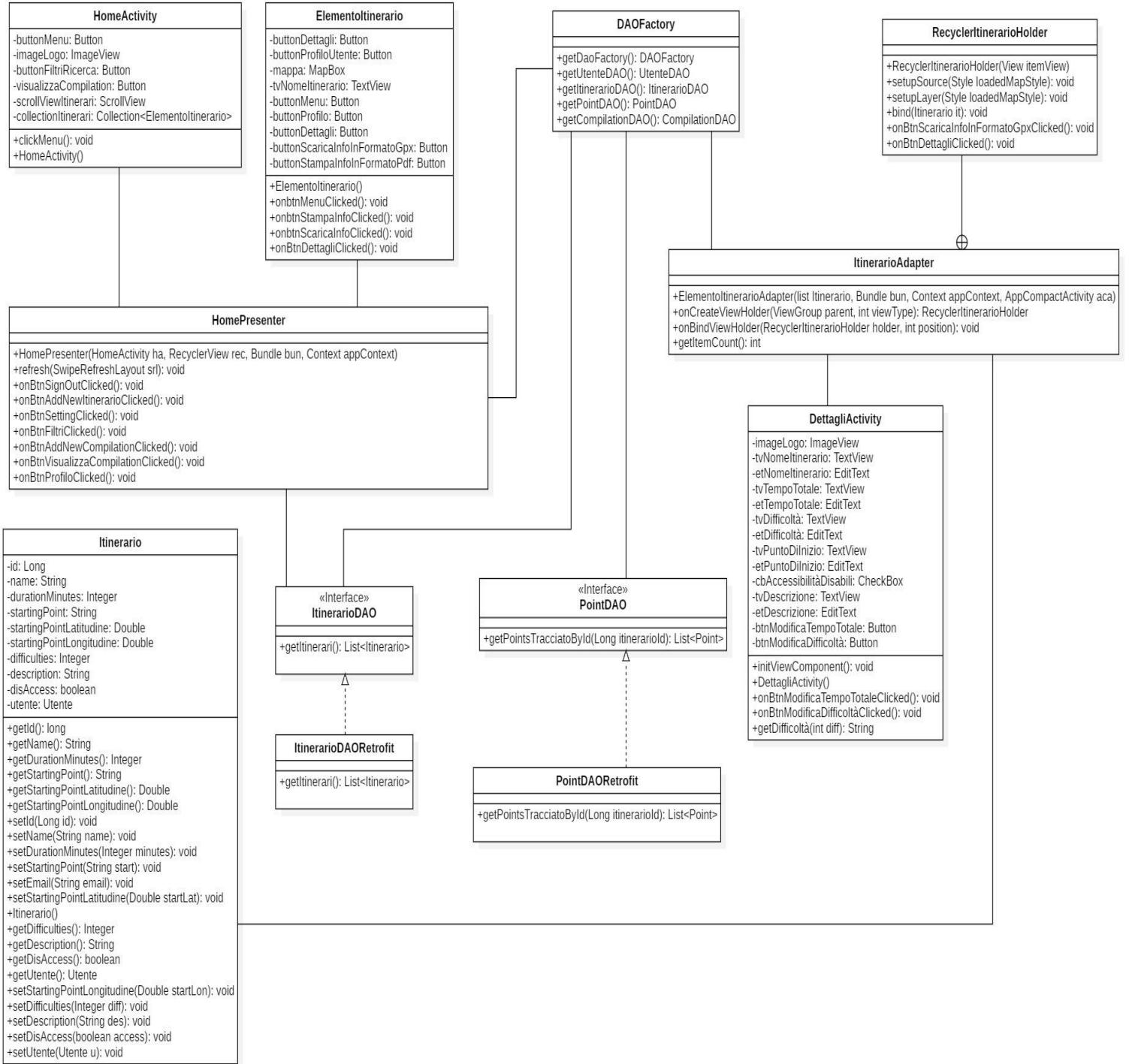
## Utente autenticato inserisce nuovo itinerario



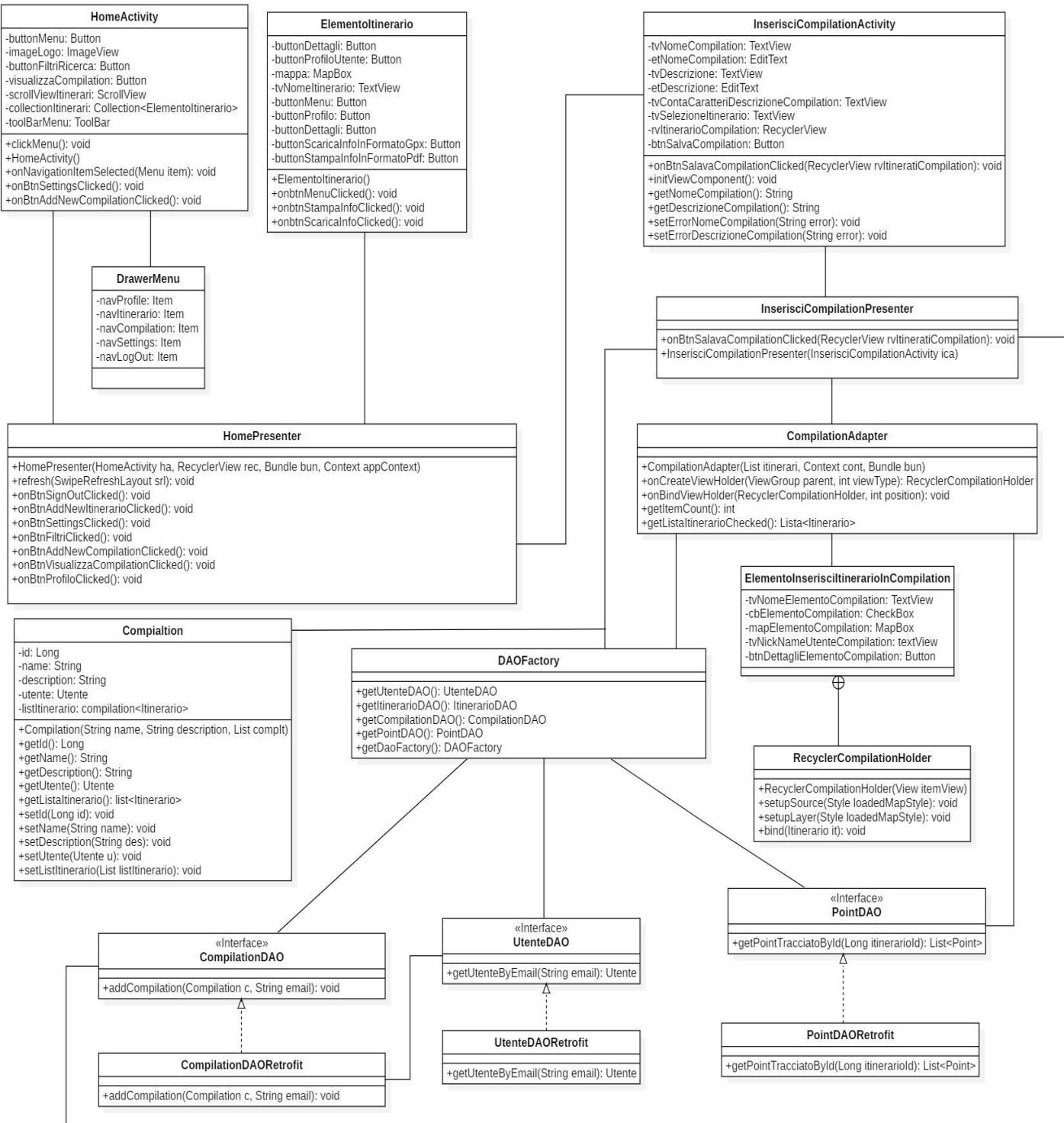
## Utente filtra ricerche itinerari



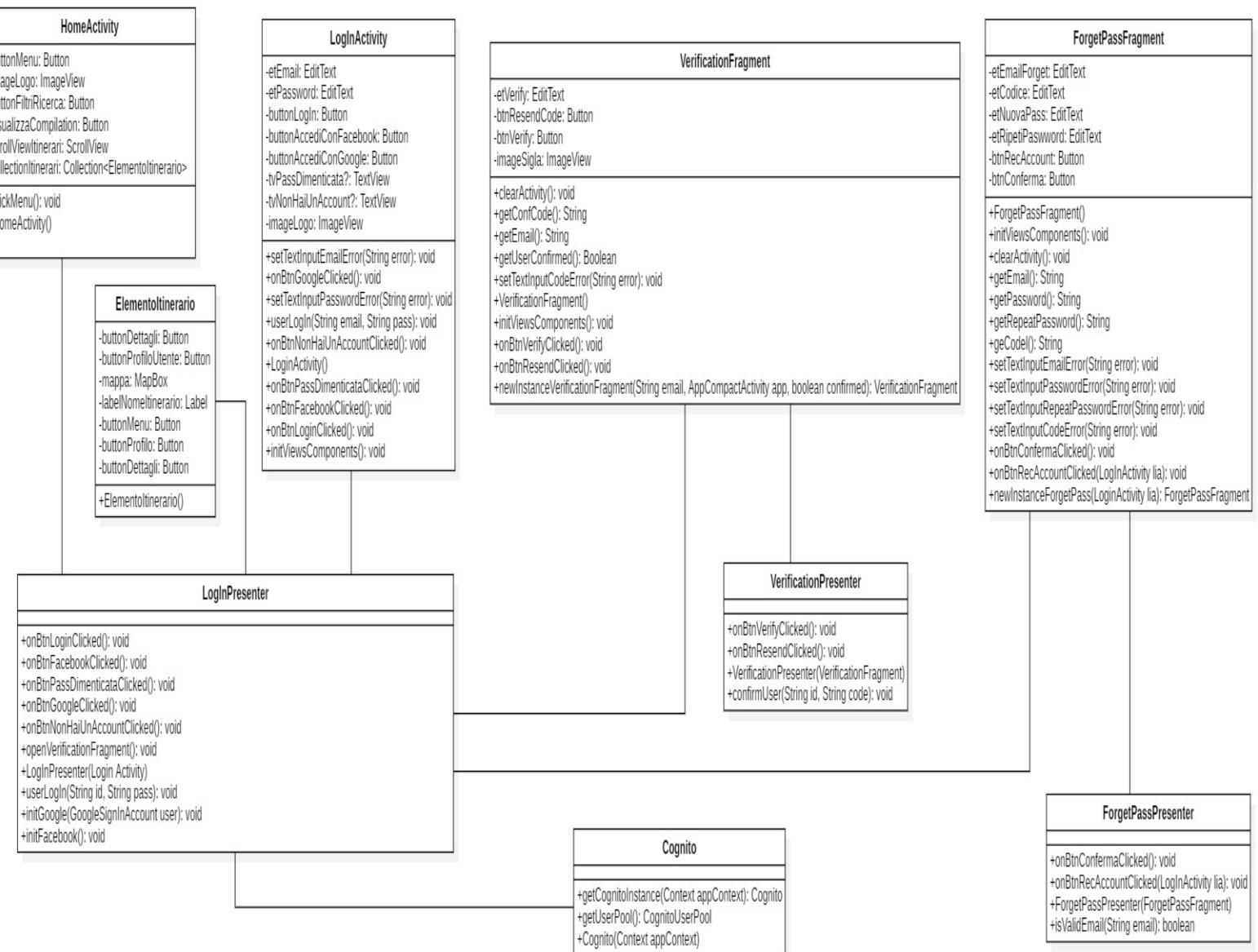
## Utente visualizza schermata di dettaglio



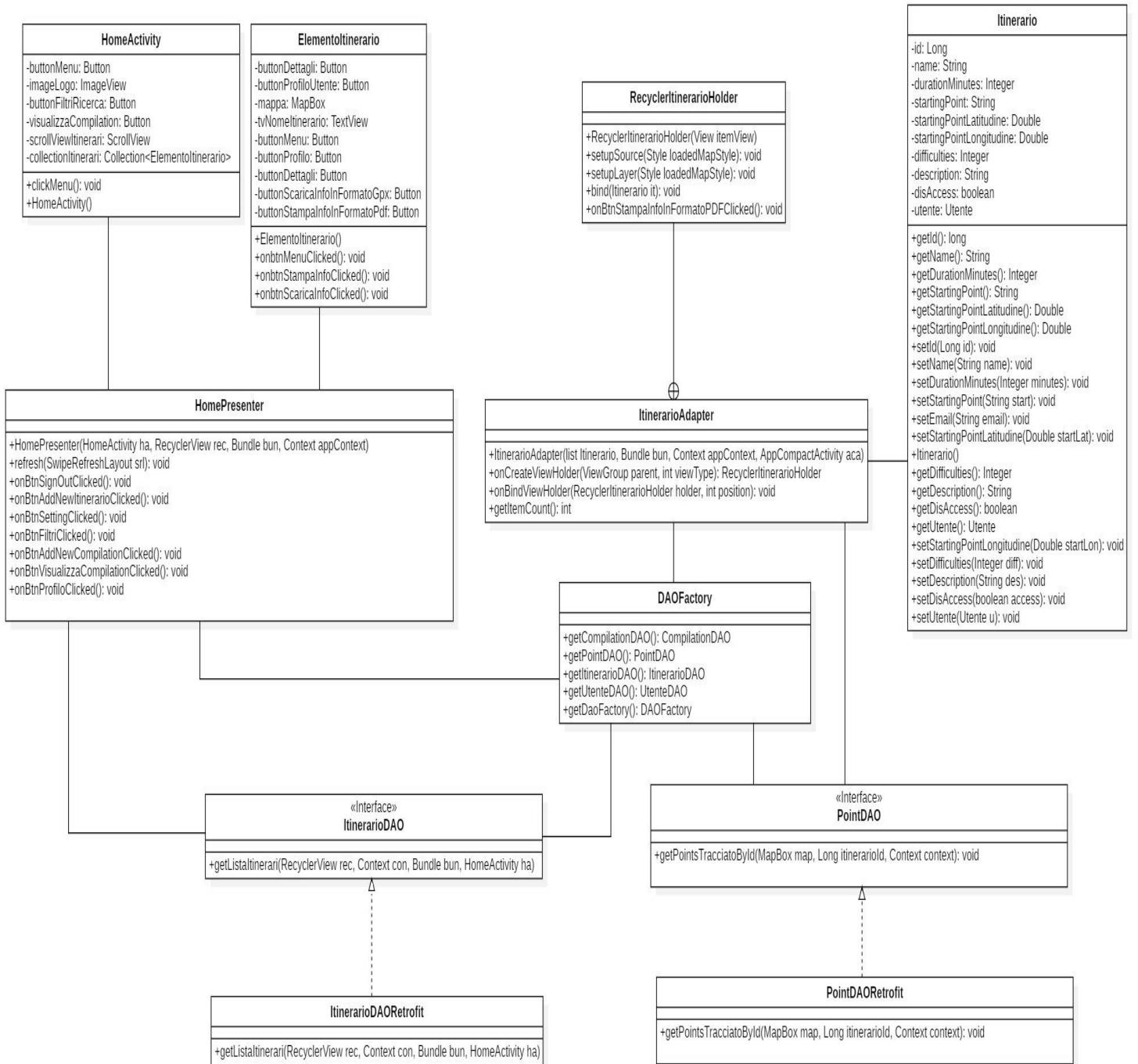
## Utente autenticato crea compilation di sentieri



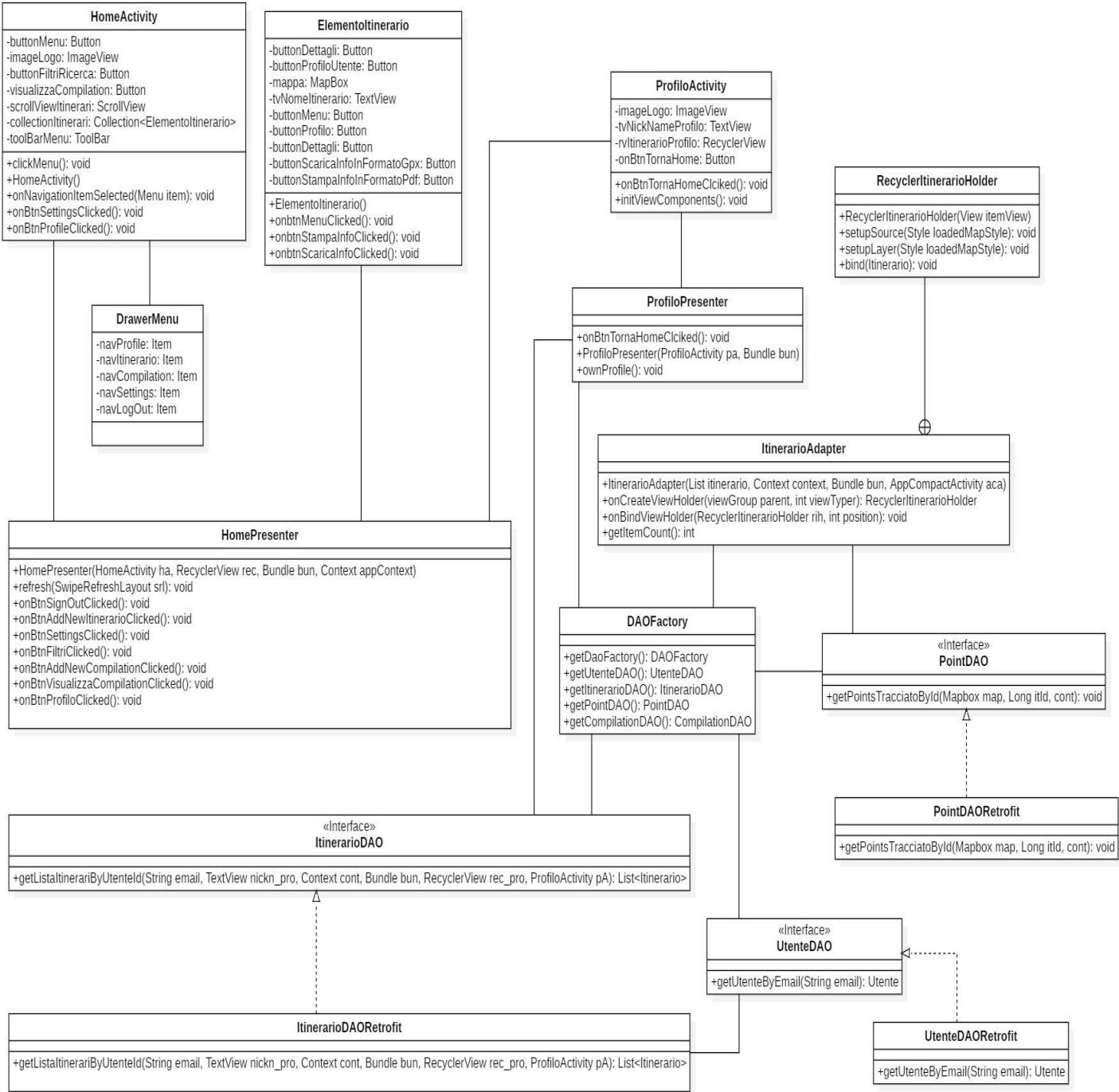
## Utente accede alla piattaforma



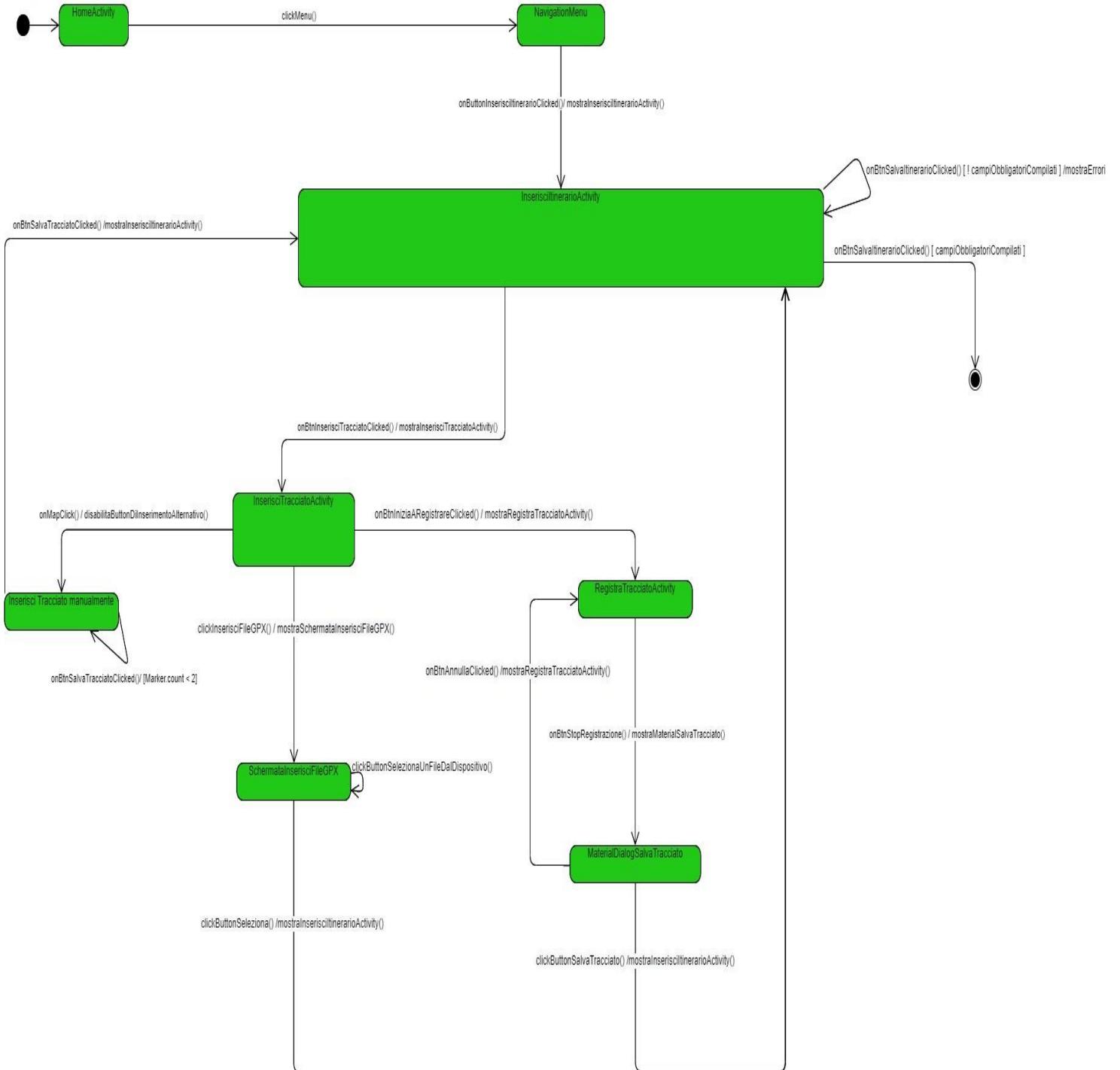
## Utente stampa informazioni itinerario in formato PDF



## Utente autenticato visualizza il profilo

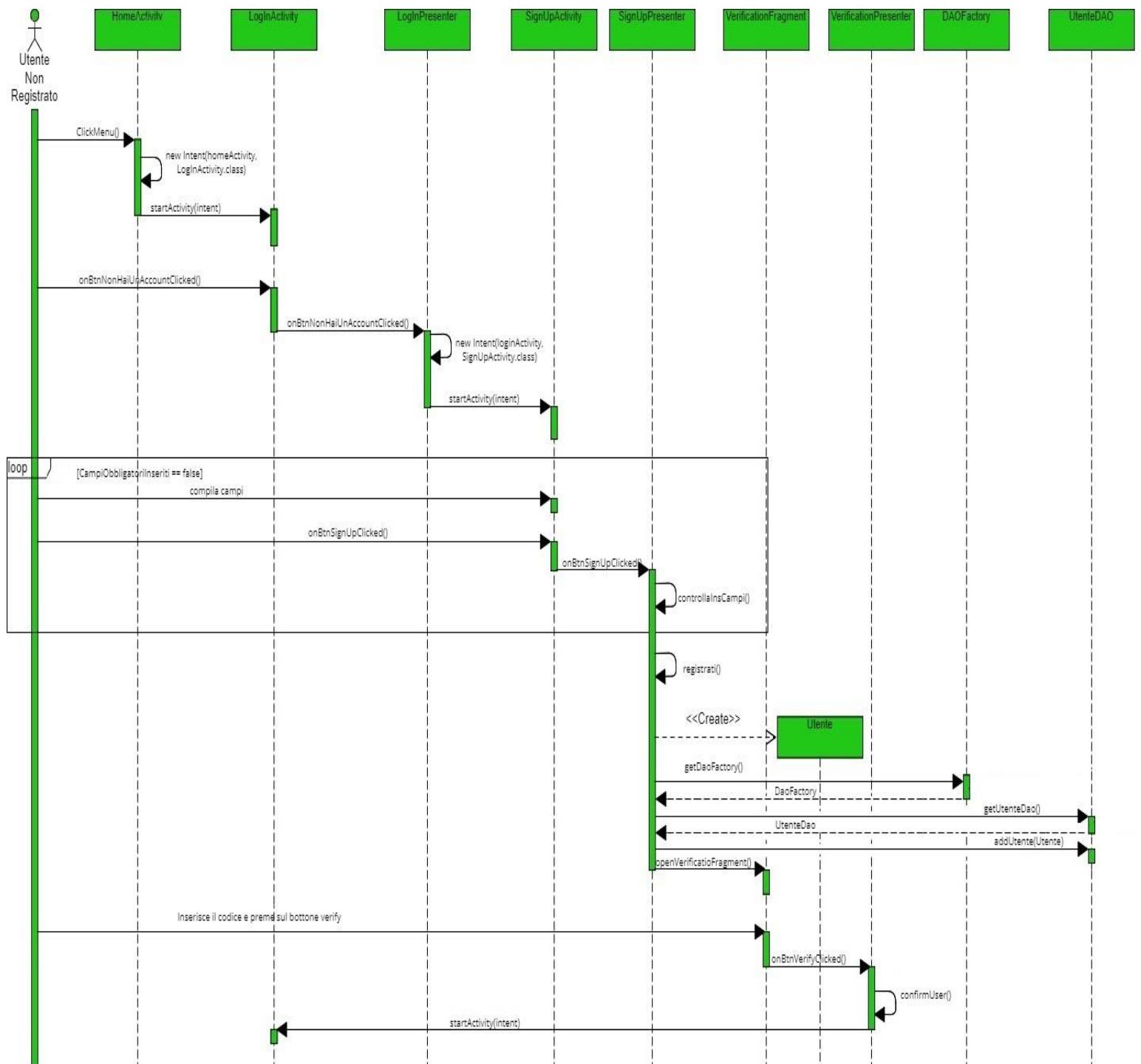


## 2.3 State Chart Diagram di design

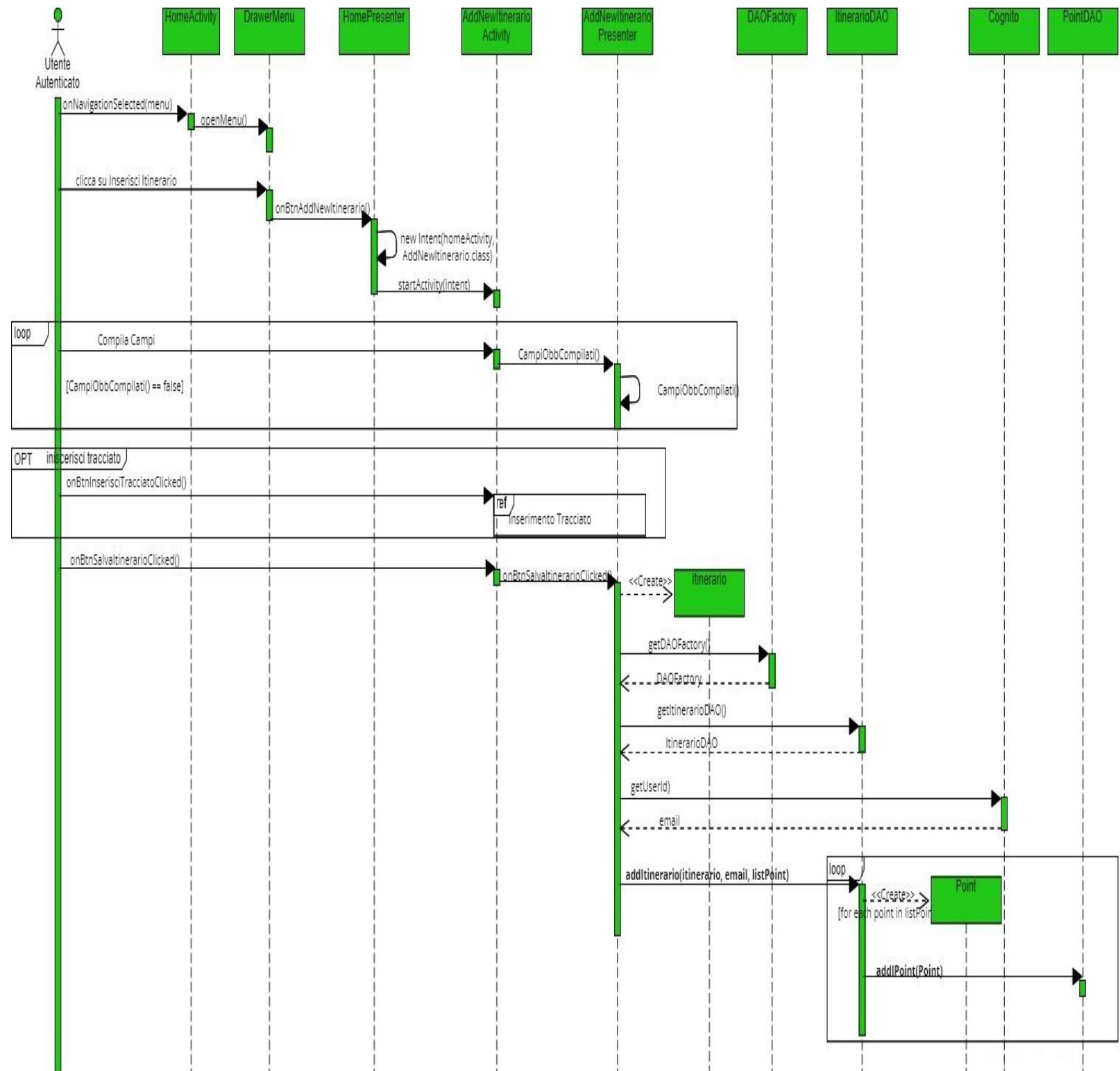


## 2.4 Sequence Diagrams di design

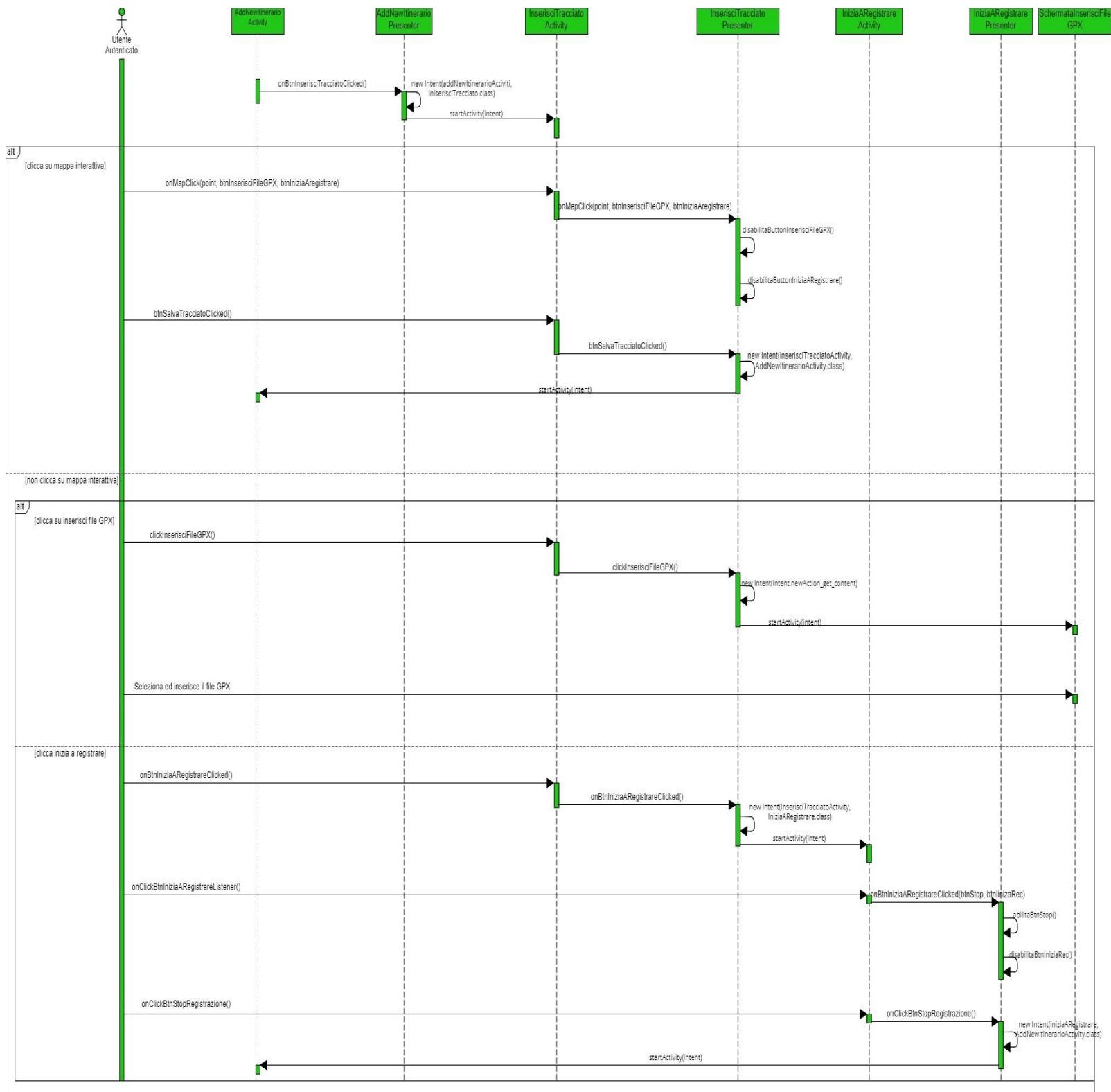
Utente non registrato effettua registrazione



## Utente autenticato inserisce nuovo itinerario



## Continuo “Utente autenticato inserisce nuovo itinerario”



# CAPITOLO III

## DOCUMENTO DI TESTING DEL SISTEMA

---

### 3.1 Test Plan per System Testing

ID Test case	Test_UC_1	
Nome	Utente non registrato effettua registrazione.	
Descrizione	L'obiettivo di questo test case è di verificare "Utente non registrato effettua registrazione".	
Input	Oracolo	Esito
L'utente inserisce tutte le informazioni richieste rispettando tutti i vincoli.	La registrazione va a buon fine.	
L'utente inserisce una mail già in uso.	La registrazione fallisce. Mostra il messaggio: “La mail inserita è già registrata alla piattaforma”.	
L'utente inserisce una password che non rispetta i vincoli (almeno 8 caratteri, tra cui un numero, una lettera minuscola, una lettera maiuscola, un numero ed un carattere speciale).	La registrazione fallisce. Mostra il messaggio: “La password deve contenere almeno 8 caratteri, tra cui un numero, una lettere minuscola, una lettera maiuscola, un numero ed un carattere speciale”.	

L'utente inserisce una password di conferma che non corrisponde con la prima password inserita.	La registrazione fallisce. Mostra il messaggio: “Le due password non coincidono”.	
L'utente lascia uno o più campi incompleti.	La registrazione fallisce, con apposito messaggio di inserimento del campo obbligatorio lasciato vuoto.	
L'utente inserisce una mail non valida.	La registrazione fallisce. Mostra il messaggio: “Inserisci una E-mail valida”.	

ID Test case	Test_UC_2	
Nome	Utente scarica informazioni itinerario in formato GPX.	
Descrizione	L'obiettivo di questo test case è di verificare “Utente scarica informazioni itinerario in formato GPX” .	
Input	Oracolo	Esito
L'utente preme sul pulsante “scarica info in formato gpx” presente nel menù relativo agli itinerari.	Il file gpx è presente nella cartella download del dispositivo dell'utente.	

<b>ID Test case</b>	<b>Test_UC_3</b>	
<b>Nome</b>	Utente registrato recupera account.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente registrato recupera account.” .	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente inserisce un' email valida e preme sul pulsante recupera account.	Invia un codice di verifica all'email specificata e abilita le EditText (per inserire il codice e le password) ed il pulsante “conferma”	
Dopo aver inserito un'email valida e premuto sul pulsante recupera account, inserisce il codice corretto ricevuto via email, la nuova password rispettando i vincoli, ripeti nuova password (correttamente) e preme sul pulsante “conferma”.	Il recupero va a buon fine. Mostra il messaggio: “La password è stata cambiata correttamente”.	
L'utente inserisce una mail non valida e preme sul pulsante recupera account.	Il recupero fallisce. Mostra il messaggio: “Inserisci una E-mail valida”.	
L'utente non inserisce una mail e preme sul pulsante recupera account.	Il recupero fallisce. Mostra il messaggio: “Inserisci la tua E-mail”.	
Dopo aver inserito un'email valida e premuto sul pulsante recupera account, inserisce un codice errato e preme sul pulsante “conferma”.	Il recupero fallisce. Mostra il messaggio: “Errore: Il codice inserito non è corretto”.	

<p>Dopo aver inserito un'email valida e premuto sul pulsante recupera account, inserisce il codice corretto e non inserisce la nuova password” e preme sul pulsante “conferma”.</p>	<p>Il recupero fallisce. Mostra il messaggio: “Inserisci la nuova password”.</p>	
<p>Dopo aver inserito un'email valida e premuto sul pulsante recupera account, inserisce il codice corretto, inserisce la nuova password e compila il campo “ripeti nuova password” in modo non conforme e preme sul pulsante “conferma”.</p>	<p>Il recupero fallisce. Mostra il messaggio: “Le due password non coincidono”.</p>	
<p>Dopo aver inserito un'email valida e premuto sul pulsante recupera account, inserisce il codice corretto, inserisce la nuova password non rispettando i vincoli e preme sul pulsante “conferma”.</p>	<p>Il recupero fallisce. Mostra il messaggio: “La password deve contenere almeno 8 caratteri, tra cui un numero, una lettera minuscola, una lettera maiuscola, un numero ed un carattere speciale”.</p>	
<p>Dopo aver inserito un'email valida e premuto sul pulsante recupera account, inserisce il codice corretto ma dopo 24 h e preme sul pulsante “conferma”.</p>	<p>Il recupero fallisce. Mostra il messaggio: Errore: Il codice inserito è scaduto, richiedi un nuovo codice</p>	

<b>ID Test case</b>	<b>Test_UC_4</b>	
<b>Nome</b>	Utente registrato modifica password.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente registrato modifica password”.	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente inserisce un' email valida, la vecchia password corretta, la nuova password rispettando i vincoli, ripete la nuova password correttamente e preme sul pulsante “conferma la modifica”.	La modifica della password va a buon fine. Mostra il messaggio: “La password è stata cambiata correttamente”.	
L'utente inserisce un' email non valida e preme sul pulsante “conferma la modifica”.	La modifica della password fallisce. Mostra il messaggio: “Inserisci una E-mail valida”.	
L'utente inserisce un' email valida, la vecchia password non corretta e preme sul pulsante “conferma la modifica”.	Il recupero fallisce. Mostra il messaggio: “Email o Password errata”.	
L'utente inserisce un' email valida ma non presente nel database e preme sul pulsante “conferma la modifica”.	Il recupero fallisce. Mostra il messaggio: “Email o Password errata”.	
L'utente inserisce un' email valida, la vecchia password corretta, la nuova password non rispettando i vincoli, e preme sul pulsante “conferma la modifica”.	Il recupero fallisce. Mostra il messaggio: “La password deve contenere almeno 8 caratteri, tra cui un numero, una lettera minuscola, una lettera maiuscola, un numero ed un carattere speciale”.	

L'utente non inserisce un'email e preme sul pulsante "conferma la modifica".	Il recupero fallisce. Mostra il messaggio: "Inserisci la tua E-mail".	
L'utente non inserisce la vecchia password e preme sul pulsante "conferma la modifica".	Il recupero fallisce. Mostra il messaggio: "Inserisci vecchia password".	
L'utente non inserisce la nuova password e preme sul pulsante "conferma la modifica".	Il recupero fallisce. Mostra il messaggio: "Inserisci la tua password".	
L'utente inserisce la nuova password e compila il campo "ripeti nuova password" in modo non conforme e preme sul pulsante "conferma la modifica".	Il recupero fallisce. Mostra il messaggio: "Le due password non coincidono".	

<b>ID Test case</b>	<b>Test_UC_5</b>	
<b>Nome</b>	Utente registrato modifica nickname.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente registrato modifica nickname”.	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente inserisce il nuovo nickname e preme sul pulsante “modifica nickname”.	<p>La modifica del nickname va a buon fine.</p> <p>Mostra il messaggio:</p> <p>“Nickname cambiato correttamente”.</p>	
L'utente inserisce non inserisce il nuovo nickname e preme sul pulsante “modifica nickname”.	<p>La modifica del nickname fallisce.</p> <p>Mostra il messaggio:</p> <p>“Inserisci un nickname”.</p>	

<b>ID Test case</b>	<b>Test_UC_6</b>	
<b>Nome</b>	Utente autenticato modifica difficoltà itinerario.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente autenticato modifica difficoltà itinerario”.	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente inserisce una nuova difficoltà per l'itinerario selezionato e preme sul pulsante “Salva modifica”.	<p>La modifica della difficoltà va a buon fine.</p> <p>Mostra il messaggio:</p> <p>“Difficoltà cambiata con successo”.</p>	
L'utente non inserisce una nuova difficoltà per l'itinerario selezionato e preme sul pulsante “Salva modifica”.	<p>La modifica della difficoltà fallisce.</p> <p>Mostra il messaggio:</p> <p>“Per effettuare la modifica inserisci la difficoltà dell'itinerario”.</p>	

<b>ID Test case</b>	<b>Test_UC_7</b>	
<b>Nome</b>	Utente autenticato modifica tempo di percorrenza itinerario.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare "Utente autenticato modifica tempo di percorrenza itinerario".	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente inserisce un nuovo tempo di percorrenza per l'itinerario selezionato e preme sul pulsante "Salva modifica".	La modifica del tempo di percorrenza va a buon fine. Mostra il messaggio: "Tempo totale cambiato con successo".	
L'utente non inserisce un nuovo tempo di percorrenza per l'itinerario selezionato e preme sul pulsante "Salva modifica".	La modifica del tempo di percorrenza fallisce. Mostra il messaggio: "Per effettuare la modifica inserisci il tempo totale dell'itinerario".	

<b>ID Test case</b>	<b>Test_UC_8</b>	
<b>Nome</b>	Utente autenticato inserisce nuovo itinerario.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente autenticato inserisce nuovo itinerario”.	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L’utente inserisce tutti i campi obbligatori in modo corretto e preme sul pulsante “Salva itinerario”.	L’inserimento dell’itinerario va a buon fine. Mostra il messaggio: “Itinerario salvato correttamente”.	
L’utente non inserisce il nome dell’ itinerario e preme sul pulsante “Salva itinerario”.	L’inserimento dell’itinerario fallisce. Mostra il messaggio: “Inserisci il nome dell’itinerario”.	
L’utente non inserisce il tempo di percorrenza dell’ itinerario e preme sul pulsante “Salva itinerario”.	L’inserimento dell’itinerario fallisce. Mostra il messaggio: “Inserisci tempo di percorrenza dell’itinerario”.	
L’utente non inserisce la difficoltà dell’ itinerario e preme sul pulsante “Salva itinerario”.	L’inserimento dell’itinerario fallisce. Mostra il messaggio: “Inserisci la difficoltà dell’itinerario”.	
L’utente non inserisce il punto d’inizio dell’ itinerario e preme sul pulsante “Salva itinerario”.	L’inserimento dell’itinerario fallisce. Mostra il messaggio: “Inserisci il punto di inizio dell’itinerario”.	

<b>ID Test case</b>	<b>Test_UC_9</b>	
<b>Nome</b>	Utente filtra ricerche itinerari.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare "Utente filtra ricerche itinerari".	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente inserisce i filtri correttamente e preme sul pulsante “Applica Filtri”.	La ricerca va a buon fine. Mostra gli itinerari filtrati.	
L'utente inserisce i filtri correttamente (che non producono risultati) e preme sul pulsante “Applica Filtri”.	La ricerca fallisce. Mostra il messaggio: “I filtri inseriti non hanno prodotto alcun risultato”.	
L'utente inserisce i filtri impostando max durata e min durata su 0 e preme sul pulsante “Applica Filtri”.	La ricerca fallisce. Mostra il messaggio: “Inserire una durata diversa da 0”.	

<b>ID Test case</b>	<b>Test_UC_10</b>	
<b>Nome</b>	Utente visualizza schermata di dettaglio.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente visualizza schermata di dettaglio”.	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente preme sul pulsante dettagli di un selezionato itinerario.	Mostra correttamente la schermata con i dettagli.	

<b>ID Test case</b>	<b>Test_UC_11</b>	
<b>Nome</b>	Utente stampa informazioni itinerario in formato PDF.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente stampa informazioni itinerario in formato PDF” .	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente preme sul pulsante “stampa informazioni in formato pdf” presente nel menù relativo agli itinerari.	Il file pdf è presente nella cartella download del dispositivo dell'utente pronto per essere stampato.	

<b>ID Test case</b>	<b>Test_UC_12</b>	
<b>Nome</b>	Utente autenticato crea compilation di sentieri.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente autenticato crea compilation di sentieri” .	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L’utente inserisce il nome della compilation, la descrizione, seleziona gli itinerari da inserire nella compilation correttamente e preme sul pulsante “Salva Compilation”.	La creazione della compilation va a buon fine. Mostra il messaggio: “Compilation salvata correttamente”	
L’utente non inserisce il nome della compilation e preme sul pulsante “Salva Compilation”.	La creazione della compilation fallisce Mostra il messaggio: “Inserire il nome della compilation”	
L’utente non inserisce la descrizione della compilation e preme sul pulsante “Salva Compilation”.	La creazione della compilation fallisce Mostra il messaggio: “Inserire la descrizione della compilation”	
L’utente inserisce il nome della compilation, la descrizione, non seleziona gli itinerari da inserire nella compilation correttamente e preme sul pulsante “Salva Compilation”.	La creazione della compilation fallisce Mostra il messaggio: “Per creare una compilation devi inserire almeno un itinerario”	

<b>ID Test case</b>	<b>Test_UC_13</b>	
<b>Nome</b>	Utente accede alla piattaforma.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente accede alla piattaforma” .	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L’utente inserisce la mail corretta, la password corretta e preme sul pulsante “Login”	L’accesso va a buon fine, mostra la schermata home e mostra il messaggio: “Autenticazione effettuata correttamente”	
L’utente non inserisce l’email e preme sul pulsante Login”.	L’accesso fallisce Mostra il messaggio: “Inserisci la tua E-mail”	
L’utente non inserisce la password e preme sul pulsante Login”.	L’accesso fallisce Mostra il messaggio: “Inserisci la tua password”	
L’utente inserisce la password o la mail errata e preme sul pulsante Login”.	L’accesso fallisce Mostra il messaggio: “Autenticazione Fallita”	
L’utente inserisce la mail corretta, la password corretta e preme sul pulsante “Login” ( ma l’account risulta registrato ma non verificato)	L’accesso fallisce Mostra la schermata di verifica dell’account e mostra il messaggio: “La tua mail risulta registrata, inserisci il codice che hai ricevuto via e-mail”	

<b>ID Test case</b>	<b>Test_UC_14</b>	
<b>Nome</b>	Utente autenticato visualizza il profilo.	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare “Utente autenticato visualizza il profilo” .	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente autenticato preme sul pulsante profilo.	Mostra correttamente la schermata del profilo.	
L'utente non autenticato preme sul pulsante profilo.	La visualizzazione del profilo fallisce e mostra la schermata di login.	

## 3.2 Unit Testing

### Testing del metodo updateNickname

Il metodo appartiene alla classe UtenteService, presente all'interno dell'applicativo server. Il suo scopo è, dati in input l'email dell'utente ed il nickname, aggiorna il vecchio nickname associato all'utente, con il nuovo nickname passato come secondo argomento al metodo.

Nel caso in cui l'email o il nickname non sono validi, il metodo lancerà un'eccezione.

#### Parametri di input:

- **String email:** Stringa che rappresenta l'email di un utente registrato nel sistema che desidera modificare il proprio nickname.
- **String nickname:** Stringa che rappresenta il nuovo nickname che l'utente desidera avere.

#### Metodo

```
public void updateNickname(String email, String nickname) {  
    Utente utente = utenteRepository.findUtenteByEmail(email).  
       orElseThrow(() -> new EmailNotExistException("Utente con email " + email + " non è presente nel DB"));  
  
    if (nickname == null || nickname.length() < 1) {  
        throw new NickNameNotValidException("Il NickName Inserito non è valido");  
    }  
    if ( !utente.getNickname().equals(nickname)) {  
        utente.setNickname(nickname);  
    }  
}
```

## Testing Black Box

### String email

Classi di equivalenza	Dominio	Validità
C. E. 1	{null}	Non valida
C. E. 2	{"“”"}	Non valida
C. E. 3	{email registrate}	Valida
C. E. 4	{String \ C. E. 3}	Non valida

### String nickname

Cassie di equivalenza	Dominio	Validità
C. E. 5	{null}	Non valida
C. E. 6	{"“”"}	Non valida
C. E. 7	{x: x è una stringa ^  x  > 0}	Valida

La strategia di testing adottata è WECT (Weak Equivalence Class Testing). Le classi di equivalenza coperte dai metodi di seguito illustrati sono le seguenti:

- checkEmailIsNullAndNickNameIsValid(): copre CE1 e CE7
- checkEmailIsBlankAndNickNameIsValid(): copre CE2 e CE7
- checkEmailIsValidAndNickNameIsValid(): copre CE3 e CE7
- checkEmailIsNotValidAndNickNameIsValid(): copre CE4 e CE7
- checkEmailIsValidAndNickNameIsNull(): copre CE3 e CE5
- checkEmailIsValidAndNickNameIsNotNull(): copre CE3 e CE6

# Codice jUnit

```
@SpringBootTest
class UtenteServiceTest {

    @Autowired private UtenteRepository utenteRepositoryTest;
    @Autowired private UtenteService utenteServiceTest;
    private Utente utente;

    @BeforeEach
    void setup() {
        utente = new Utente( name: "Carmine", surname: "DiMa", nickname: "CarDiMa313", email: "test@test.it", loggedin: false);
        utenteRepositoryTest.save(utente);
    }

    @AfterEach
    void tearDown() {
        utenteRepositoryTest.delete(utente);
    }

    @Test
    void checkEmailIsNullAndNickNameIsValid() {
        Assertions.assertThrows(EmailNotExistException.class, ()-> utenteServiceTest.updateNickname( email: null, nickname: "nickname"));
    }

    @Test
    void checkEmailIsBlankAndNickNameIsValid() {
        Assertions.assertThrows(EmailNotExistException.class, ()-> utenteServiceTest.updateNickname( email: "", nickname: "nickname"));
    }

    @Test
    void checkEmailIsValidAndNickNameIsValid() {
        utenteServiceTest.updateNickname( email: "test@test.it", nickname: "CarDiMa313");
        Assertions.assertEquals( expected: "CarDiMa313", utenteRepositoryTest.findUtenteByEmail("test@test.it").get().getNickname());
    }

    @Test
    void checkEmailIsNotValidAndNickNameIsValid() {
        Assertions.assertThrows(EmailNotExistException.class, ()-> utenteServiceTest.updateNickname( email: "emailNotRegistered", nickname: "nickname"));
    }

    @Test
    void checkEmailIsValidAndNickNameIsNull() {
        Assertions.assertThrows(NickNameNotValidException.class, ()-> utenteServiceTest.updateNickname( email: "carmine.7@hotmail.it", nickname: null));
    }

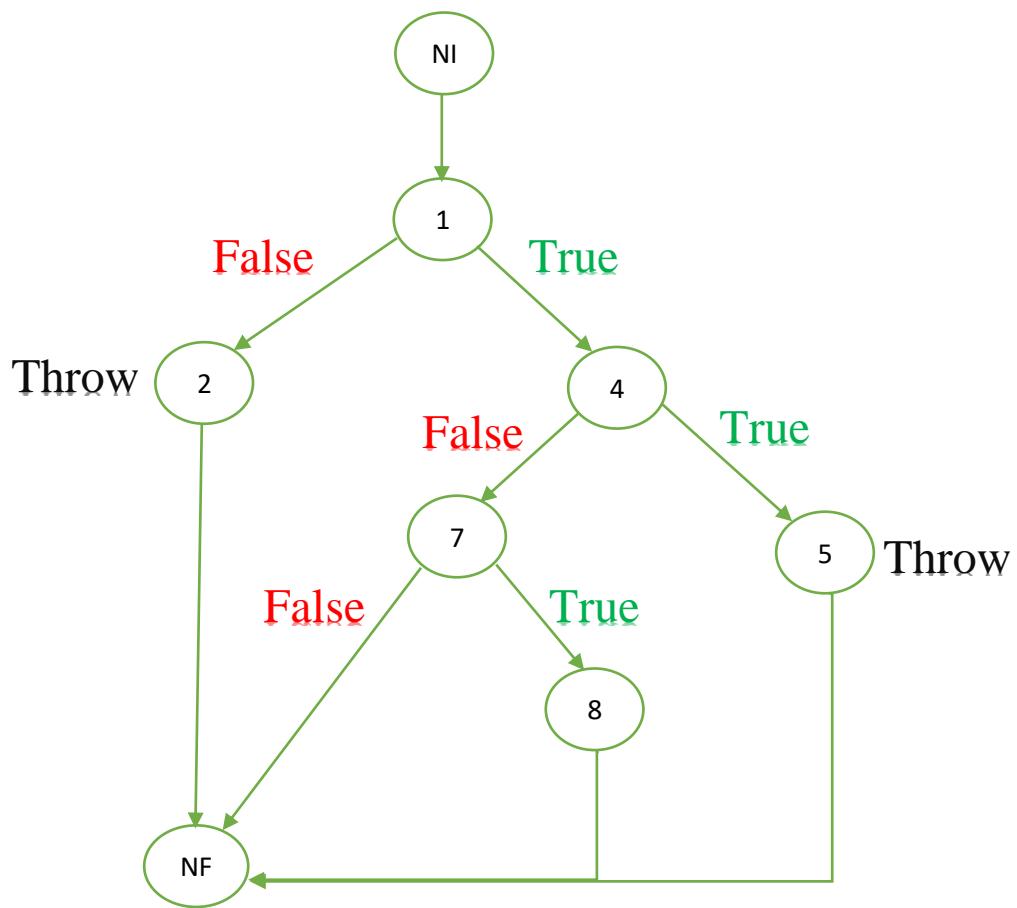
    @Test
    void checkEmailIsValidAndNickNameIsNotValid() {
        Assertions.assertThrows(NickNameNotValidException.class, ()-> utenteServiceTest.updateNickname( email: "carmine.7@hotmail.it", nickname: ""));
    }
}
```

The screenshot shows the UtenteServiceTest class with six test methods. The first four tests pass, while the last two fail with a NickNameNotValidException. The test results are displayed in a table at the bottom of the code editor.

Test	Status	Time	Message
Test Results	PASSED	4 sec 117 ms	"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" ...
UtenteServiceTest	PASSED	4 sec 117 ms	12:16:55.166 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDe
checkEmailIsBlankAndNickNameIsValid	PASSED	1 sec 226 ms	12:16:55.183 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using cor
checkEmailIsNullAndNickNameIsValid	PASSED	614 ms	12:16:55.271 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper
checkEmailIsValidAndNickNameIsNotNull	PASSED	526 ms	12:16:55.311 [main] INFO org.springframework.boot.test.context.SpringBootTestBootstrapper - Neither @ContextD
checkEmailIsValidAndNickNameIsNotValid	FAILED	517 ms	12:16:55.323 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default re
checkEmailIsValidAndNickNameIsNull	FAILED	523 ms	12:16:55.324 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default re
checkEmailIsValidAndNickNameIsNotNull	FAILED	71 ms	

## Testing White Box

Il Grafo del Flusso di Controllo ottenuto dal codice è il seguente:



Di seguito vengono illustrati i metodi necessari ai fini di una Branch Coverage totale del **GFC** precedentemente illustrato.

```
@Test
void check_path_1_2() {
    Assertions.assertThrows(EmailNotExistException.class, ()-> utenteServiceTest.updateNickname( email: "emailNotRegistered", nickname: "nickname"));
}

@Test
void check_path_1_4_5() {
    Assertions.assertThrows(NickNameNotFoundException.class, ()-> utenteServiceTest.updateNickname( email: "test@test.it", nickname: null));
}

@Test
void check_path_1_4_7() {
    utenteServiceTest.updateNickname( email: "test@test.it", nickname: "CarDiMa31");
    Assertions.assertEquals( expected: "CarDiMa31", utenteRepositoryTest.findUtenteByEmail("test@test.it").get().getNickname());
}

@Test
void check_path_1_4_7_8() {
    utenteServiceTest.updateNickname( email: "test@test.it", nickname: "Curiosity");
    Assertions.assertEquals( expected: "Curiosity", utenteRepositoryTest.findUtenteByEmail("test@test.it").get().getNickname());
}
```

utenteServiceTest x  
↓ ↴ ↵ ↻ ↑ ↓ ⌂ » ✅ Tests passed: 10 of 10 tests – 6 sec 91 ms

## Testing del metodo updateDifficultiesItinerario

Il metodo appartiene alla classe ItinerarioService, presente all'interno dell'applicativo server. Il suo scopo è, dati in input l'id dell'itinerario e la difficoltà, aggiorna la vecchia difficoltà associata all'itinerario, con la nuova difficoltà passata come secondo argomento al metodo.

Nel caso in cui l'id o la nuova difficoltà non sono validi, il metodo lancerà un'eccezione.

### Parametri di input:

- **Long itinerarioID:** Long che rappresenta l'id di un itinerario registrato nel sistema del quale si desidera modificare la difficoltà.
- **Integer difficoltà:** Integer che rappresenta la nuova difficoltà impostata da un utente per l'itinerario.

## Metodo

```
public void updateDifficultiesItinerario(Long itinerarioId, Integer difficulties) {  
    if (itinerarioId == null) {  
        throw new IllegalArgumentException("L'id dell'itinerario non può essere null");  
    }  
    Itinerario itinerario = itinerarioRepository.findById(itinerarioId).  
       orElseThrow(() -> new IdNotExistException("Itinerario con id " + itinerarioId + " non è presente nel DB"));  
    if (difficulties == null) {  
        throw new IllegalArgumentException("La difficoltà dell'itinerario non può essere null");  
    }  
    else if (difficulties > -1 && difficulties < 4) {  
        itinerario.setDifficulties(difficulties);  
    } else {  
        throw new DifficultiesIsNotValidException("La difficoltà inserita per l'itinerario non è valida");  
    }  
}
```

## Testing Black Box

### Long itinerarioId

Classi di equivalenza	Dominio	Validità
C. E. 1	[MIN_LONG, 0]	Non valida
C. E. 2	{null}	Non valida
C. E. 3	{x: x è Long ^ x è presente nel sistema}	Valida
C. E. 4	{Long \ C. E. 3}	Non valida

### Integer difficulties

Cass di equivalenza	Dominio	Validità
C. E. 5	[MIN_INT, -1]	Non valida
C. E. 6	[0, 3]	valida
C. E. 7	[4, MAX_INT]	Non valida
C. E. 8	{null}	Non valida

La strategia di testing adottata è WECT (Weak Equivalence Class Testing). Le classi di equivalenza coperte dai metodi di seguito illustrati sono le seguenti:

- checkItinerarioIDIsNotPositiveAndDifficultiesIsValid (): copre CE1 e CE6
- checkItinerarioIDIsNullAndDifficultiesIsValid (): copre CE2 e CE6
- checkItinerarioIDIsValidAndDifficultiesIsValid (): copre CE3 e CE6
- checkItinerarioIDIsNotValidAndDifficultiesIsValid (): copre CE4 e CE6
- checkItinerarioIDIsValidAndDifficultiesIsNegative (): copre CE3 e CE5
- checkItinerarioIDIsValidAndDifficultiesIsNotValid(): copre CE3 e CE7
- checkItinerarioIDIsValidAndDifficultiesIsNull(): copre CE3 e CE8

## Codice jUnit

```
@SpringBootTest
class ItinerarioServiceTest {

    @Autowired private ItinerarioRepository itinerarioRepositoryTest;
    @Autowired private ItinerarioService itinerarioServiceTest;
    private Itinerario itinerarioTest;

    @BeforeEach
    void setup() {
        itinerarioTest = new Itinerario( name: "Test", durationMinutes: 300, startingPoint: "T.A.", difficulties: 1, description: "test", disAccess: true);
        itinerarioRepositoryTest.save(itinerarioTest);
    }

    @AfterEach
    void tearDown() {
        itinerarioRepositoryTest.delete(itinerarioTest);
    }

    @Test
    void checkItinerarioIDIsNotPositiveAndDifficultiesIsValid() {
        Assertions.assertThrows(IdNotExistException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario( itinerarioId: -1L, difficulties: 1));
    }

    @Test
    void checkItinerarioIDIsNullAndDifficultiesIsValid() {
        Assertions.assertThrows(IllegalArgumentException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario( itinerarioId: null, difficulties: 1));
    }

    @Test
    void checkItinerarioIDIsNotValidAndDifficultiesIsValid() {
        Assertions.assertThrows(IdNotExistException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario( itinerarioId: 500L, difficulties: 1));
    }

    @Test
    void checkItinerarioIDisValidAndDifficultiesIsValid() {
        itinerarioServiceTest.updateDifficultiesItinerario(itinerarioRepositoryTest.getIdByItinerario(itinerarioTest), difficulties: 3);
        Assertions.assertEquals( expected: 3, itinerarioRepositoryTest.getItinerarioById(itinerarioRepositoryTest.getIdByItinerario(itinerarioTest)).getDifficulties());
    }

    @Test
    void checkItinerarioIDisValidAndDifficultiesIsNegative() {
        Assertions.assertThrows(DifficultiesIsNotValidException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario( itinerarioId: 35L, difficulties: -1));
    }

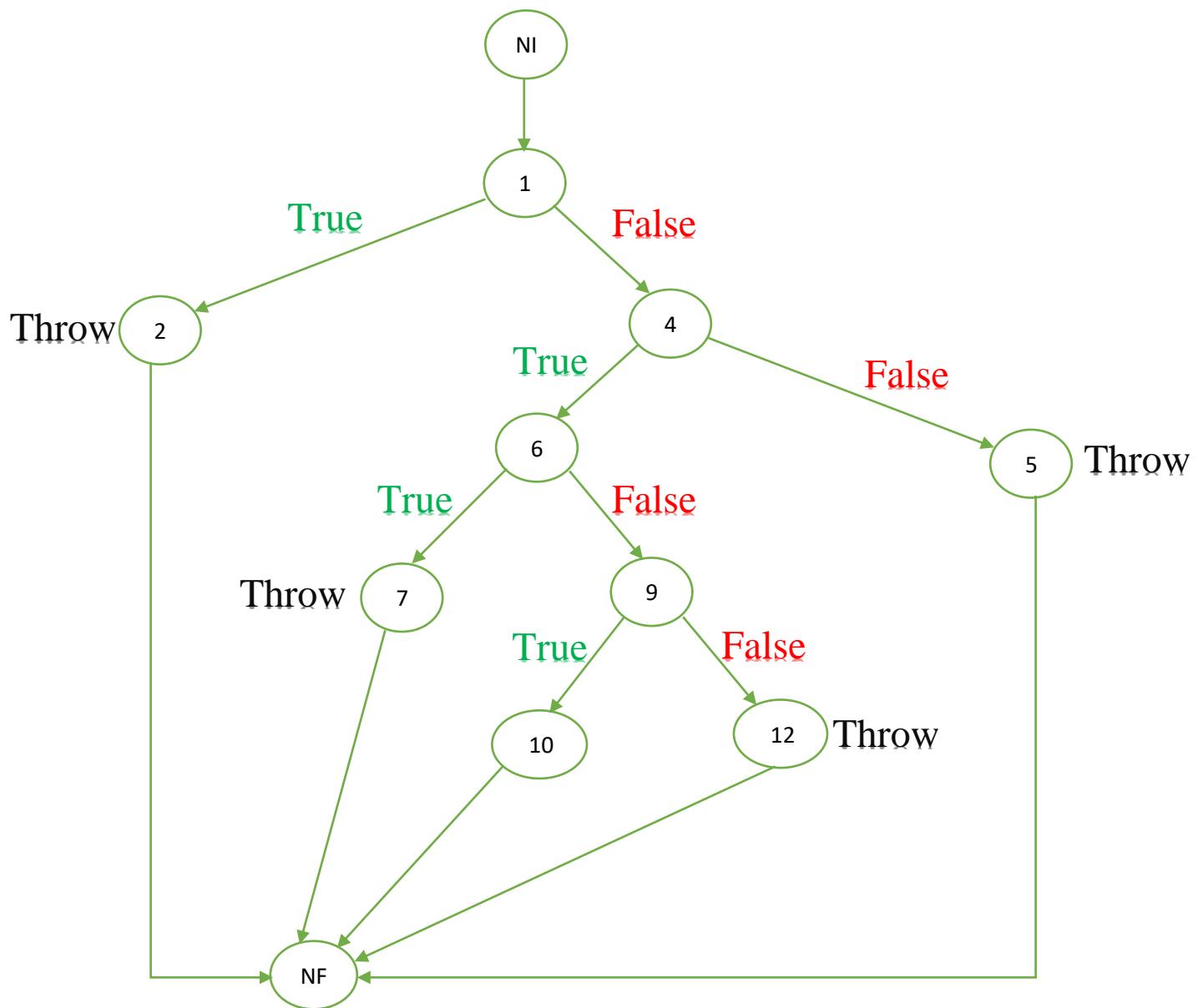
    @Test
    void checkItinerarioIDisValidAndDifficultiesIsNotValid() {
        Assertions.assertThrows(DifficultiesIsNotValidException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario( itinerarioId: 35L, difficulties: 7));
    }

    @Test
    void checkItinerarioIDisValidAndDifficultiesIsNull() {
        Assertions.assertThrows(IllegalArgumentException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario( itinerarioId: 35L, difficulties: null));
    }
}
```

ItinerarioServiceTest x 4 sec 266 ms C:\Program Files\Java\jdk-17.0.2\bin\java.exe ...

## Testing White Box

Il Grafo del Flusso di Controllo ottenuto dal codice è il seguente:



Di seguito vengono illustrati i metodi necessari ai fini di una Branch Coverage totale del **GFC** precedentemente illustrato.

```
@Test
void check_path_1_2() {
    Assertions.assertThrows(IllegalArgumentException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario(itinerarioId: null, difficulties: 1));
}

@Test
void check_path_1_4_5() {
    Assertions.assertThrows(NotFoundException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario(itinerarioId: 500L, difficulties: 1));
}

@Test
void check_path_1_4_6_7() {
    Assertions.assertThrows(IllegalArgumentException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario(itinerarioId: 35L, difficulties: null));
}

@Test
void check_path_1_4_6_9_10() {
    itinerarioServiceTest.updateDifficultiesItinerario(itinerarioRepositoryTest.getItinerarioIdByItinerario(itinerarioTest), difficulties: 3);
    Assertions.assertEquals(expected: 3, itinerarioRepositoryTest.getItinerarioById(itinerarioRepositoryTest.getItinerarioIdByItinerario(itinerarioTest)).getDifficulties());
}

@Test
void check_path_1_4_6_9_12() {
    Assertions.assertThrows(DifficultiesIsNotValidException.class, ()-> itinerarioServiceTest.updateDifficultiesItinerario(itinerarioId: 35L, difficulties: 7));
}
```

itinerarioServiceTest X ⚙ -

↓ ↴ ⏪ ⏵ ⏹ ⏵ ⏹ ↴ ⏵ ↴ ⏹ ↴ ↵ ⓘ » ✓ Tests passed: 18 of 18 tests - 9 sec 521 ms

## Testing del metodo updateDurationItinerario

Il metodo appartiene alla classe ItinerarioService, presente all'interno dell'applicativo server. Il suo scopo è, dati in input l'id dell'itinerario e il tempo di percorrenza (durata), aggiorna la vecchia durata associata all'itinerario, con la nuova durata passata come secondo argomento al metodo.

Nel caso in cui l'id o la nuova durata non sono validi, il metodo lancerà un'eccezione.

### Parametri di input:

- **Long itinerarioID:** Long che rappresenta l'id di un itinerario registrato nel sistema del quale si desidera modificare la difficoltà.
- **Integer durationMinutes:** Integer che rappresenta il nuovo tempo di percorrenza impostato da un utente per l'itinerario.

## Metodo

```
public void updateDurationItinerario(Long itinerarioId, Integer durationMinutes) {  
    if (itinerarioId == null) {  
        throw new IllegalArgumentException("L'id dell'itinerario non può essere null");  
    }  
    Itinerario itinerario = itinerarioRepository.findById(itinerarioId).  
       orElseThrow(() -> new IdNotExistException("Itinerario con id " + itinerarioId + " non è presente nel DB"));  
  
    if (durationMinutes == null) {  
        throw new IllegalArgumentException("La durata dell'itinerario non può essere null");  
    }  
    else if (durationMinutes < 1) {  
        throw new DurationIsNotValidException("La durata di percorrenza inserita per l'itinerario non è valida");  
    }  
    itinerario.setDurationMinutes(durationMinutes);  
}
```

## Testing Black Box

### Long itinerarioId

Classi di equivalenza	Dominio	Validità
C. E. 1	[MIN_LONG, 0]	Non valida
C. E. 2	{null}	Non valida
C. E. 3	{x: x è Long ^ x è presente nel sistema}	Valida
C. E. 4	{Long \ C. E. 3}	Non valida

### Integer durationMinutes

Cassie di equivalenza	Dominio	Validità
C. E. 5	[MIN_INT, 0]	Non valida
C. E. 6	[1, MAX_INT]	valida
C. E. 7	{null}	Non valida

La strategia di testing adottata è WECT (Weak Equivalence Class Testing). Le classi di equivalenza coperte dai metodi di seguito illustrati sono le seguenti:

- `checkItinerarioIDIsNotPositiveAndDurationIsValid ()`: copre CE1 e CE6
- `checkItinerarioIDIsNullAndDurationIsValid ()`: copre CE2 e CE6
- `checkItinerarioIDIsValidAndDurationIsValid ()`: copre CE3 e CE6
- `checkItinerarioID IsNotValidAndDurationIsValid ()`: copre CE4 e CE6
- `checkItinerarioIDIsValidAndDurationIsNotPositive ()`: copre CE3 e CE5
- `checkItinerarioIDIsValidAndDurationIsNull ()`: copre CE3 e CE7

## Codice jUnit

```
@Test
void checkItinerarioIDIsNotPositiveAndDurationIsValid() {
    Assertions.assertThrows(NotExistException.class, ()-> itinerarioServiceTest.updateDurationItinerario(itinerarioId: -1L, durationMinutes: 10));
}

@Test
void checkItinerarioIDIsNullAndDurationIsValid() {
    Assertions.assertThrows(IllegalArgumentException.class, ()-> itinerarioServiceTest.updateDurationItinerario(itinerarioId: null, durationMinutes: 10));
}

@Test
void checkItinerarioIDIsValidAndDurationIsValid() {
    itinerarioServiceTest.updateDurationItinerario(itinerarioRepositoryTest.getItinerarioIdByItinerario(itinerarioTest), durationMinutes: 100);
    Assertions.assertEquals(expected: 100, itinerarioRepositoryTest.getItinerarioById(itinerarioRepositoryTest.getItinerarioIdByItinerario(itinerarioTest))
        .getDurationMinutes());
}

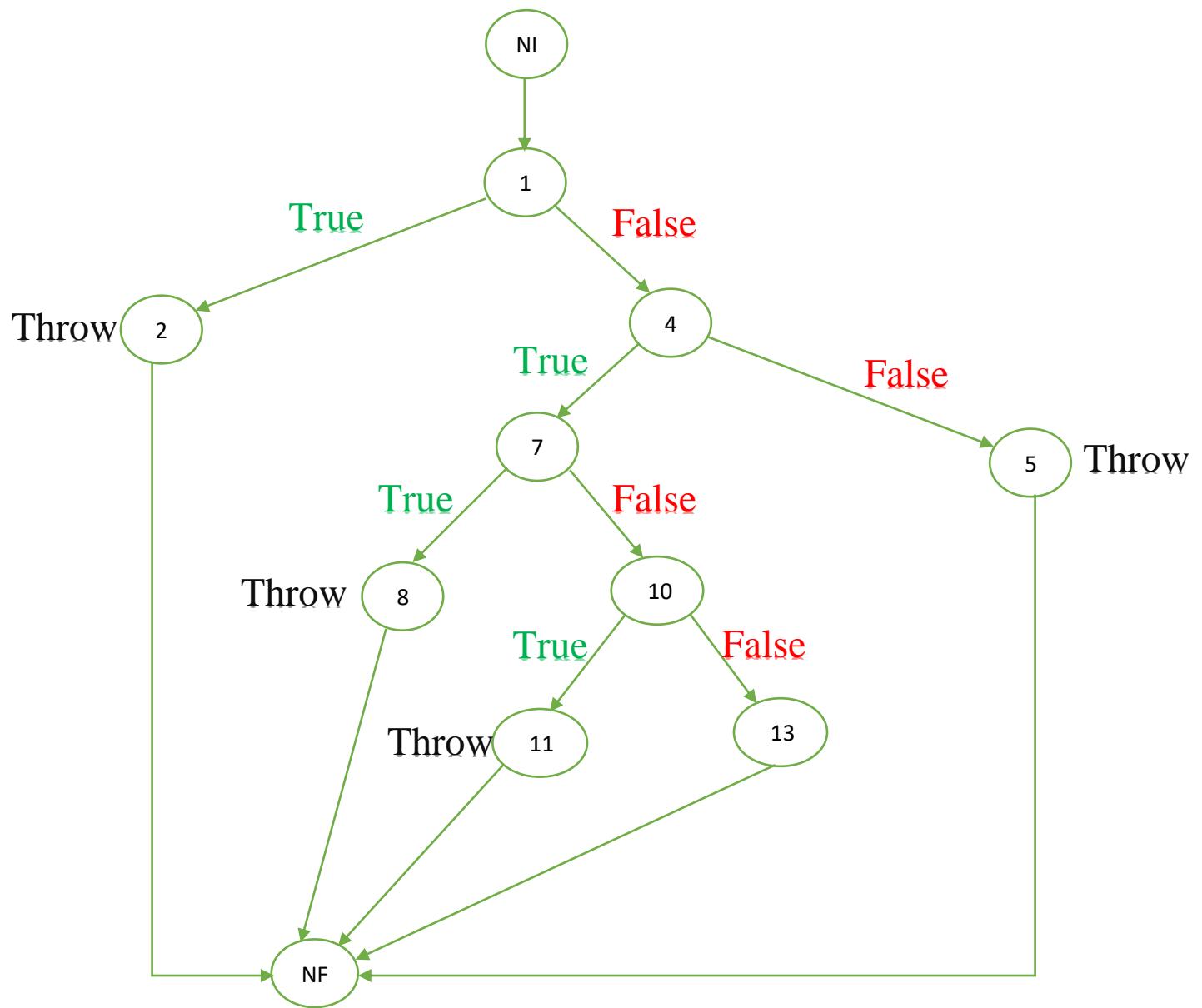
@Test
void checkItinerarioIDIsNotValidAndDurationIsValid() {
    Assertions.assertThrows(NotExistException.class, ()-> itinerarioServiceTest.updateDurationItinerario(itinerarioId: 500L, durationMinutes: 10));
}

@Test
void checkItinerarioIDIsValidAndDurationIsNotPositive() {
    Assertions.assertThrows(DurationIsNotValidException.class, ()-> itinerarioServiceTest.updateDurationItinerario(itinerarioId: 35L, durationMinutes: -1));
}

@Test
void checkItinerarioIDIsValidAndDurationIsNull() {
    Assertions.assertThrows(IllegalArgumentException.class, ()-> itinerarioServiceTest.updateDurationItinerario(itinerarioId: 35L, durationMinutes: null));
}
```

## Testing White Box

Il Grafo del Flusso di Controllo ottenuto dal codice è il seguente:



Di seguito vengono illustrati i metodi necessari ai fini di una Branch Coverage totale del **GFC** precedentemente illustrato.

```
@Test
void check_path_1_2b() {
    Assertions.assertThrows(IllegalArgumentException.class, ()-> itinerarioServiceTest.updateDurationItinerario( itinerarioId: null, durationMinutes: 10));
}

@Test
void check_path_1_4_5b() {
    Assertions.assertThrows(IdNotExistException.class, ()-> itinerarioServiceTest.updateDurationItinerario( itinerarioId: 500L, durationMinutes: 10));
}

@Test
void check_path_1_4_7_8b() {
    Assertions.assertThrows(IllegalArgumentException.class, ()-> itinerarioServiceTest.updateDurationItinerario( itinerarioId: 35L, durationMinutes: null));
}

@Test
void check_path_1_4_7_10_11b() {
    Assertions.assertThrows(DurationIsNotValidException.class, ()-> itinerarioServiceTest.updateDurationItinerario( itinerarioId: 35L, durationMinutes: -1));
}

@Test
void check_path_1_4_7_10_13b() {
    itinerarioServiceTest.updateDurationItinerario(itinerarioRepositoryTest.getIdByItinerario(itinerarioTest), durationMinutes: 100);
    Assertions.assertEquals( expected: 100, itinerarioRepositoryTest.getIdByItinerario(itinerarioRepositoryTest.getIdByItinerario(itinerarioTest))
        .getDurationMinutes());
}
```

ioServiceTest X  
Tests passed: 23 of 23 tests – 12 sec 867 ms