

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 “NaTour21”

Autore:

Gruppo INGSW2122_S_03

Aleks Nikolaev Nikolov

N86003002

Docenti:

Prof. Sergio di Martino

Prof. Francesco Cutugno

Indice

Indice	2
Revisioni documentazione.....	5
I - Documento dei Requisiti	6
1.1 - Modello funzionale.....	6
1.1.0 - Glossario.....	6
1.1.1 – Presentazione dell’idea progettuale	7
1.1.2 – Individuazione del target di utenti.....	8
1.1.3 - Requisiti funzionali	9
1.1.4 - Requisiti non funzionali	11
1.1.5 – Use Case Diagram	12
1.1.6 - Descrizione testuale degli use case	13
1.1.7 – Prototipazione visuale	17
1.1.8 – Valutazione dell’usabilità a priori	21
1.1.9 – Prototipazione funzionale.....	24
1.1.10 – Tabella delle funzionalità	25
1.2 - Modelli di dominio	26
1.2.1 – Class diagram di analisi	26
Autenticazione (effettua accesso + crea account)	27
Recupera account	28
Visualizza Home.....	29
Inserisci itinerario	30
Visualizza Itinerario Propri.....	31
Visualizza Dettagli Itinerario	32
Visualizza Profilo.....	33
1.2.2 – Sequence diagram di analisi	34
Inserisci Itinerario Mappa.....	35
Inserisci Itinerario GPX	36
1.2.3 – Activity diagram	37
Recupero Account	37
II - Documento di Design	38
2.1 - Architettura del sistema.....	38
2.1.1 – Architettura del client.....	38
2.1.2 – Architettura del server.....	40
2.1.3 – Gerarchia funzionale.....	43
2.2 – System Design.....	44

2.2.1 – Class diagram di design.....	44
Effettua Accesso	44
Effettua Accesso (Google).....	45
Crea Account	46
Recupero Account	47
Visualizza Home.....	48
Inserisci Itinerario Mappa.....	49
Inserisci Itinerario GPX	50
Visualizza Dettagli Itinerario	51
Visualizza Itinerari Propri.....	52
Visualizza Profilo.....	53
2.2.2 – Sequence diagram di design	54
Crea Account	54
Inserisci Itinerario GPX	55
III - Documento di Testing.....	56
3.1 - Unit testing di 3 metodi con JUnit	56
3.1.1 – Codice JUnit per il metodo validaCompilazioneCampi.....	56
3.1.2 – Codice JUnit per il metodo generaDurataItinerario	63
3.1.3 – Codice JUnit per il metodo generalIntervalloTemporaleInserimento.....	68
3.2 – Valutazione dell’usabilità sul campo.....	72

Revisioni documentazione

Data	Descrizione
13/11/2021	Stesura iniziale, Strutturazione documento
18/11/2021	Presentazione dell'idea progettuale, individuazione target utenti, elenco requisiti, use case diagram
21/11/2021	Descrizione testuale di due use case significativi, mockup, gerarchia funzionale, tabella delle funzionalità
28/11/2021	Prototipazione funzionale, Valutazione dell'usabilità a priori
10/12/2021	Modelli di dominio, sequence diagram di analisi e activity diagram
21/12/2021	Architettura del sistema, client e server
20/01/2022	Object design
12/03/2022	Unit testing
24/03/2022	Valutazione usabilità sul campo

I - Documento dei Requisiti

1.1 - Modello funzionale

1.1.0 - Glossario

Il seguente è un elenco di termini spesso utilizzati nella sezione corrente della documentazione.

Termino	Descrizione
Utente non autenticato	Un utente dell'applicativo che non ha ancora creato un account, oppure non ha effettuato l'autenticazione.
Utente autenticato	Un utente che ha effettuato l'autenticazione mediante le credenziali di email e password, oppure utilizzando un account di una piattaforma esterna, come Google.
Itinerario	Un tracciato geografico dotato di titolo, durata, un livello di difficoltà, una descrizione (opzionale), e un percorso(opzionale)
Percorso	Un percorso/sentiero dotato di un punto d'inizio, un punto di fine e, opzionalmente, da punti di passaggio o tappe. Tali punti vengono individuati dalle coordinate geografiche più comuni: longitudine e latitudine.
Livello di difficoltà	E' una metrica per rappresentare quanto sia difficile da percorrere un itinerario.
File GPX	Uno schema XML progettato per il trasferimento di dati GPS tra applicazioni software.

1.1.1 – Presentazione dell’idea progettuale

Si vuole realizzare un sistema informativo, denominato “NaTour21” e finalizzato ad offrire un moderno social network per appassionati di escursioni.

L’utente finale potrà interagire con il sistema mediante un dispositivo mobile il quale, grazie alla presenza di un sensore GPS, risulta essere la tipologia di dispositivo ideale per un’applicativo di natura geografica.

Una volta autenticato, l’utente potrà inserire nuove escursioni sulla piattaforma, dotandole di varie informazioni volte a descrivere le caratteristiche dell’itinerario, come il nome, la durata, il livello di difficoltà, i punti di inizio e fine, e altri. L’itinerario dovrebbe essere rappresentato anche tramite una mappa geografica interattiva.

Gli itinerari possono essere inseriti dall’utente in vari modi, ad esempio interagendo con una mappa, oppure tramite file in formato standard GPX importati dal loro dispositivo.

Un itinerario dovrebbe essere visualizzabile dall’utente in una schermata di dettaglio, all’interno della quale si trovano tutti i dettagli sull’itinerario. L’utente dovrebbe inoltre poter visualizzare il percorso dell’itinerario su una mappa, così come eventuali recensioni e fotografie inserite da altri utenti.

L’applicativo va progettato in modo tale da facilitarne la manutenibilità e da agevolare la futura espansione mediante l’integrazione di nuove funzionalità.

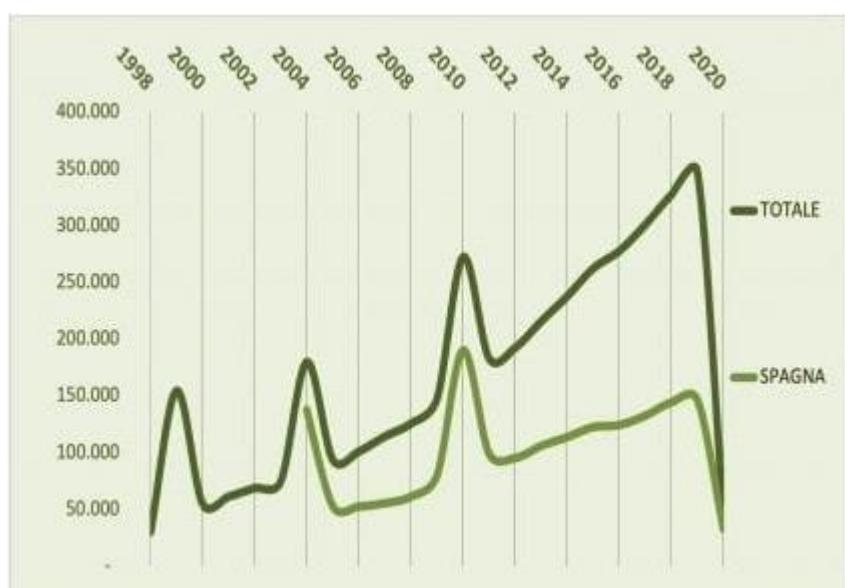
1.1.2 – Individuazione del target di utenti

L'utente finale dell'applicativo NaTour21 è un individuo appassionato di attività sportive come l'escurcionismo, che è interessato a condividere le proprie esperienze con gli altri. E' fondamentale quindi offrire a questa categoria di persone un insieme di funzionalità e servizi che possono rendere facile ed immediata la condivisione di itinerari su una piattaforma dedicata. Per individuare questo target di utenti è stata svolta una ricerca bassandosi su dati e statistiche raccolti dai più grandi leader nell'editoria sportiva legata al trekking.

Sorgenti: OutdoorMagazine, Terre di Mezzo

Il grafico a fianco riporta il risultato di una statistica sul numero di persone che si sono recate a percorrere gli itinerari più famosi in Italia. Si può notare come il Covid-19 ha causato un abbassamento drastico del numero di escursionisti.

Tuttavia, con il potenziale approccio della fine della pandemia, si ipotizza che molte persone si dedicheranno di nuovo all'attività di escursionismo, il che risulterebbe in un target di potenziali utenti più ampio.



Successivamente, è stata fatta una raccolta di dati legati all'età dei camminatori. Dal grafico sottostante si può notare che la maggioranza dei camminatori rientra nella fascia d'età tra i 41 e 70 anni. Ciò può dare spunto alla progettazione di un'interfaccia grafica estremamente intuitiva da utilizzare, che si dirige più verso il minimalismo e la semplicità.



1.1.3 - Requisiti funzionali

Di seguito sono elencati i requisiti funzionali, ovvero le funzionalità che il sistema deve offrire:

Nome	<u>Crea account</u>
Descrizione	Un utente non autenticato deve poter creare un nuovo account con il quale poter accedere all'applicativo.

Nome	<u>Effettua accesso</u>
Descrizione	Un utente non autenticato deve poter effettuare l'accesso, inserendo le proprie credenziali di accesso, oppure tramite un account appartenente ad una piattaforma esterna.

Nome	<u>Recupera account</u>
Descrizione	Il sistema deve consentire ad un utente non autenticato di recuperare il proprio account inserendo la mail collegata all'account e confermando di esserne il proprietario attraverso un codice di verifica. Il recupero consiste nel permettere all'utente di modificare la sua password di accesso.

Nome	<u>Visualizza home</u>
Descrizione	Un utente autenticato deve poter visitare la schermata home dell'applicativo, sulla quale vengono riportati gli itinerari più recenti inseriti da tutti gli utenti.

Nome	<u>Inserisci itinerario mappa</u>
Descrizione	Un utente autenticato deve poter inserire nuovi itinerari tramite una mappa interattiva.

Nome	<u>Inserisci itinerario GPX</u>
Descrizione	Un utente autenticato deve poter inserire nuovi itinerari tramite l'importazione di un file in formato standard GPX.

Nome	<u>Visualizza dettagli itinerario</u>
Descrizione	Un utente autenticato deve poter visualizzare i dettagli di un itinerario, incluse le eventuali recensioni degli utenti e fotografie caricate.

Nome	<u>Visualizza itinerari propri</u>
Descrizione	Un utente autenticato deve poter visualizzare un elenco degli itinerari da esso creati.

Nome	<u>Rimuovi itinerario</u>
Descrizione	Un utente autenticato deve poter rimuovere un itinerario da egli inserito.

Nome	<u>Visualizza profilo</u>
Descrizione	Il sistema deve consentire ad un utente autenticato di visualizzare le informazioni sul suo profilo.

1.1.4 - Requisiti non funzionali

Di seguito sono elencati i requisiti non funzionali, ovvero i vincoli sulle varie qualità del software:

Nome	<u>Performance delle ricerche</u>
Descrizione	Le ricerche effettuate dall'utente devono fornire risultati, almeno parziali, entro 3 secondi

Nome	<u>Apprendibilità</u>
Descrizione	L'utente deve essere in grado di utilizzare il 90% delle funzionalità offerte entro un massimo di 2 ore di utilizzo

Nome	<u>Affidabilità</u>
Descrizione	Il sistema è in grado di fornire la funzione richiesta dall'utente il 99.99% delle volte che essa è richiesta

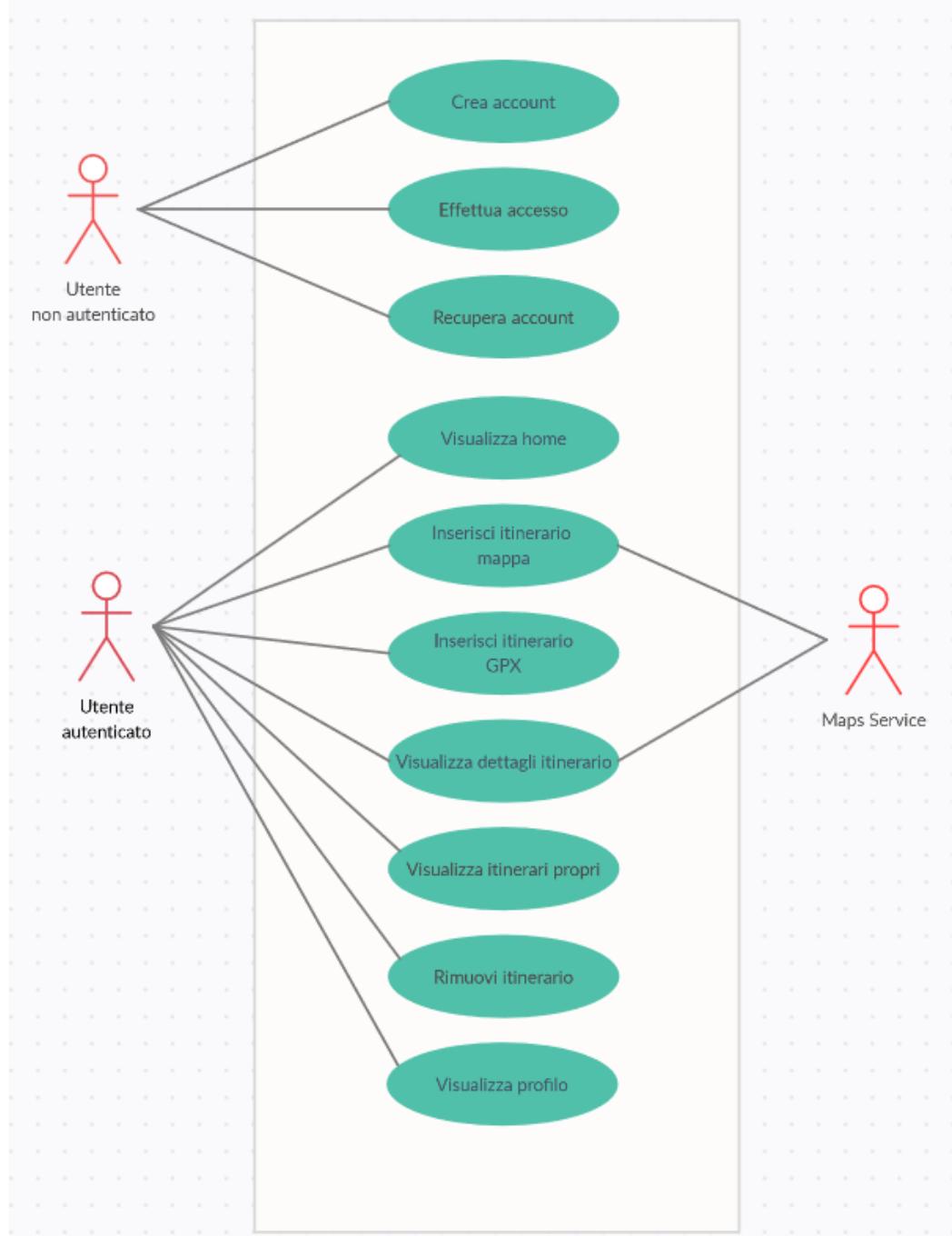
Nome	<u>Manutenibilità</u>
Descrizione	Il sistema deve essere realizzato in modo da poter essere facilmente mantenuto. Eventuali modifiche o evolutive dovrebbero essere apportate in maniera facile e in tempi efficienti.

Nome	<u>Implementazione</u>
Descrizione	Il sistema deve consistere di un lato back end realizzato in un linguaggio object-oriented a scelta e un lato client realizzato sulla piattaforma android

1.1.5 – Use Case Diagram

Di seguito viene riportato il diagramma dei casi d'uso, che riporta in modo semplice e schematico gli attori del sistema e le funzionalità associate a questi. Come si può vedere, la principale distinzione è quella tra utenti autenticati e non autenticati.

Inoltre è presente anche l'attore esterno Maps Service, che permetterà di usufruire di vari servizi legati alla geolocalizzazione e alla visualizzazione dei tracciati su una mappa interattiva.



1.1.6 - Descrizione testuale degli use case

Per la descrizione testuale degli use case è stato utilizzato il template di Alistair Cockburn che permette di offrire, passo per passo, una descrizione del flusso di eventi negli use case.

Di seguito si trovano le tabelle per due use case più significativi. I mockup ai quali viene fatto riferimento all'interno delle tabelle mediante il formato ***mp_xx_schermata*** si trovano nella sezione [1.1.8 – Prototipazione visuale](#).

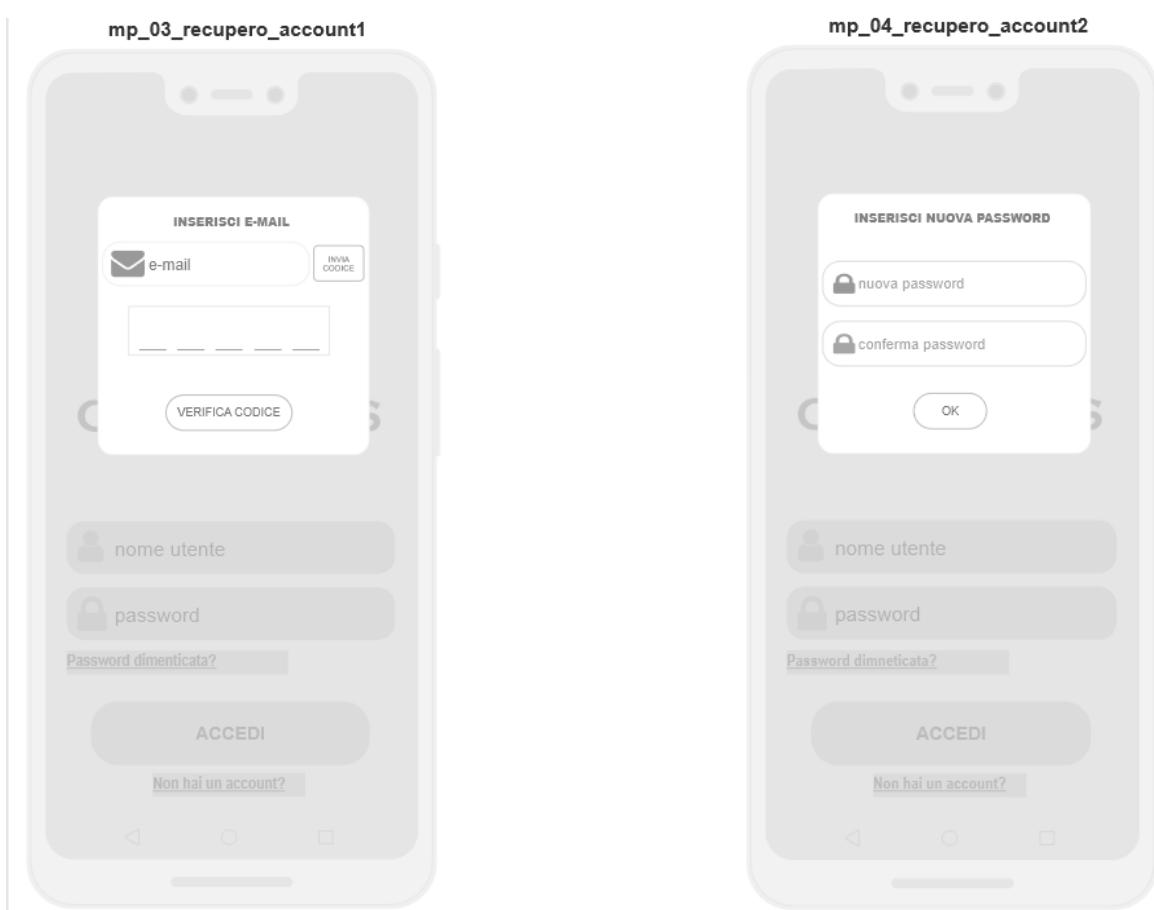
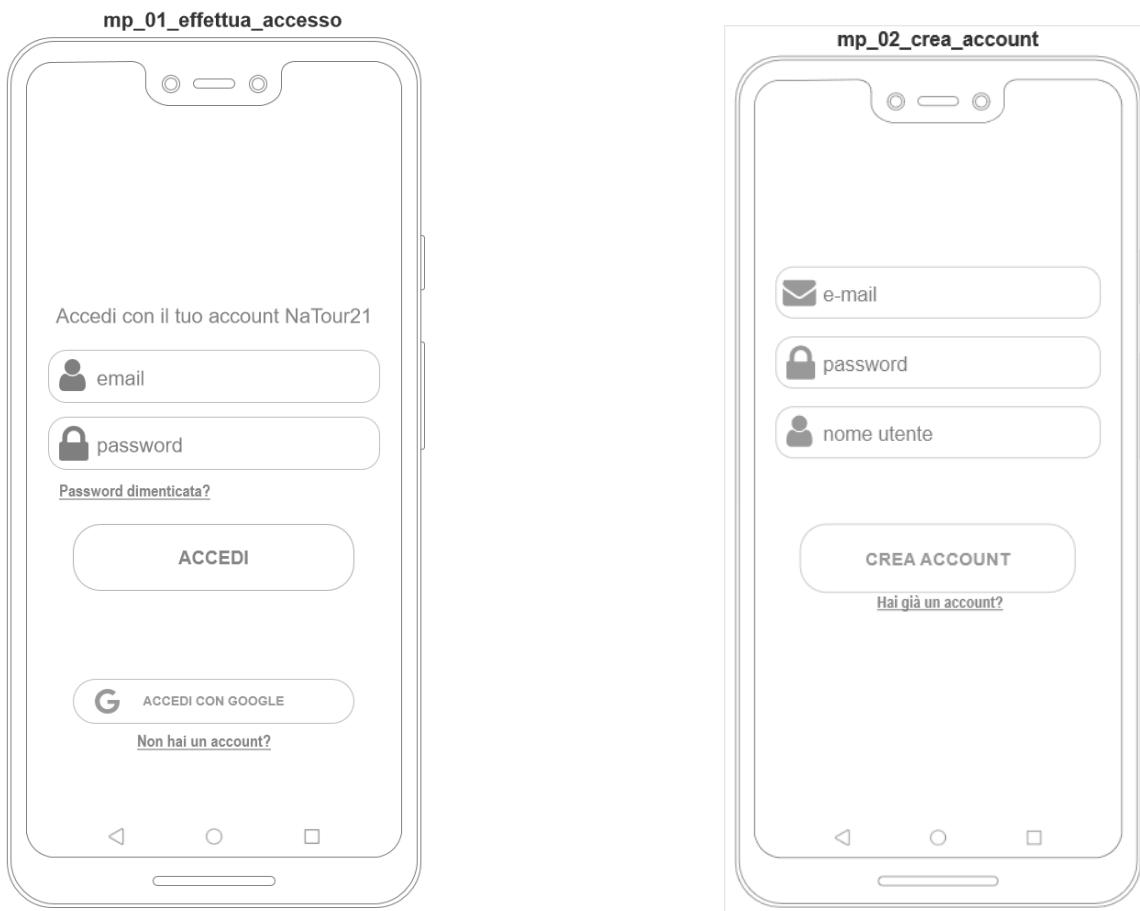
USE CASE	<i>Crea account</i>		
Goal in Context	<i>L'utente non autenticato vuole creare un account</i>		
Preconditions	<i>L'utente non autenticato è fornito di mail non collegata ad alcun account</i>		
Success End Condition	<i>L'utente non autenticato è adesso in possesso di un account</i>		
Failed End Condition	<i>L'utente non autenticato non è in possesso di alcun account</i>		
Primary Actor	<i>Utente non autenticato</i>		
Trigger	<i>Pressione label Non hai un account? di mp_01_effettua_accesso</i>		
MAIN SCENARIO: Tutto si svolge correttamente, l'operazione va a buon fine	Step n°	Attore: <i>Utente non autenticato</i>	Sistema
	1		Mostra <i>mp_02_crea_account</i>
	2	Compila i campi di testo obbligatori email, password, nome utente	
	3	Preme il pulsante CREA ACCOUNT di mp_02_crea_account	
	4		Controlla che i campi di testo obbligatori siano stati compilati correttamente, con esito positivo
	5		Controlla che non esista già account associato a quella mail, con esito positivo
	6		Crea la nuova utenza
	7		Avvisa l'utente dell'avvenuto successo mostrando la dialog <i>dl_01_account_creato</i>

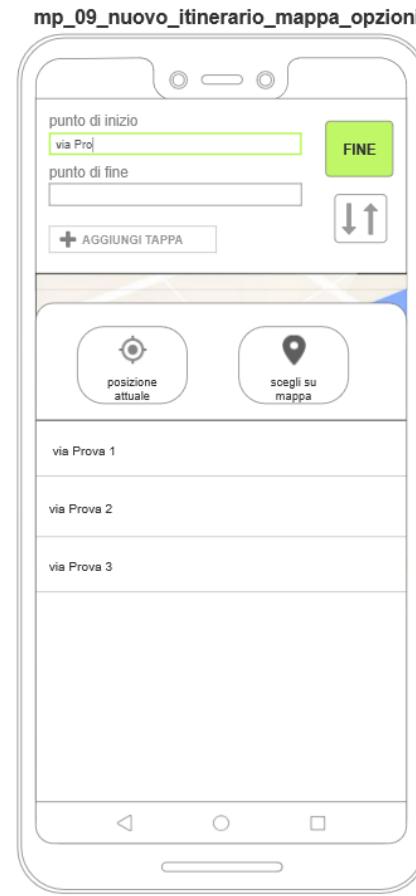
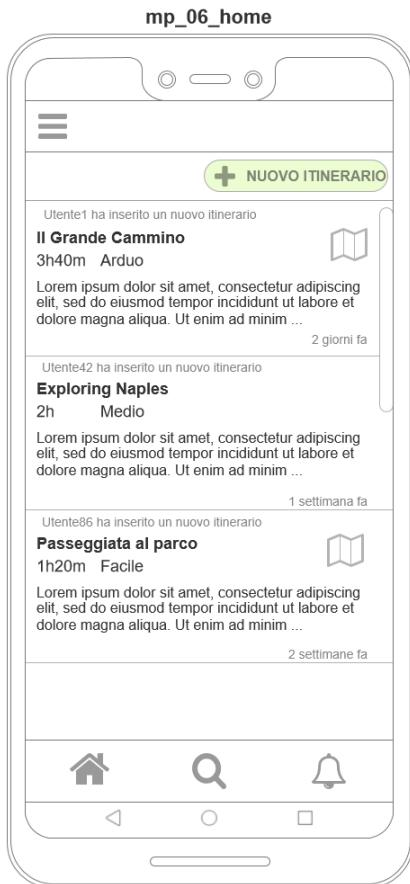
EXTENSION_1: L'utente non compila tutti i campi obbligatori	Step n°	Attore: Utente non autenticato	Sistema
	4.1		Controlla che i campi di testo obbligatori siano stati compilati correttamente, con esito negativo.
	5.1		Avvisa l'utente e richiede la compilazione dei campi.
	6.1	Inserisce le informazioni mancanti	
	7.1	Preme il pulsante CREA ACCOUNT di <i>mp_02_crea_account</i>	
	8.1		Torna allo step 4 del MAIN_SCENARIO
EXTENSION_2: Esiste già un account corrispondente alla mail inserita dall'utente	Step n°	Attore: Utente non autenticato	Sistema
	5.2		Controlla che non esista già account associato a quella mail, con esito negativo.
			Avvisa l'utente e richiede l'inserimento di una mail diversa
	6.2	Inserisce una nuova mail	
	7.2	Preme il pulsante ISCRIVITI di <i>mp_02_crea_account</i>	
	8.2		Torna allo step 4 del MAIN_SCENARIO
EXTENSION_3: L'utente si rende conto che possiede già un account	Step n°	Attore: Utente non autenticato	Sistema
	Fino a step 3	Preme il pulsante hai già un account? di <i>mp_02_crea_account</i>	
	Step successivo		Mostra <i>mp_01_effettua_accesso</i> . Parte lo use case Effettua accesso

USE CASE	Inserisci itinerario mappa		
Goal in Context	<i>L'utente vuole inserire un nuovo itinerario</i>		
Preconditions	<i>L'utente è autenticato</i>		
Success End Condition	<i>L'utente inserisce un nuovo itinerario nel sistema</i>		
Failed End Condition	<i>L'utente non ha inserito un nuovo itinerario</i>		
Primary Actor	<i>Utente autenticato</i>		
Trigger	<i>Pressione del pulsante Nuovo itinerario in mp_06_home</i>		
MAIN SCENARIO: Tutto si svolge correttamente e l'utente inserisce un nuovo itinerario con un percorso	Step n°	Attore Utente autenticato	Sistema
	1		Mostra mp_07_nuovo_itinerario_dati
	2	Inserisce il nome, la durata, la difficoltà e, optionalmente, la descrizione dell'itinerario	
	3	Preme il tasto Apri mappa	
	4		Mostra mp_08_nuovo_itinerario_mappa
	5	Inserisce il punto d'inizio e di fine dell'itinerario come indirizzi fisici oppure come punti sulla mappa	
	6		Interroga Servizio Mappe per ottenere le coordinate geografiche degli indirizzi / punti inseriti
	7		Risponde restituendo le coordinate geografiche richieste
	8		Mostra gli indirizzi inseriti dell'utente sulla mappa e disegna un tracciato che li collega
	9	Preme il tasto Fine	
	10		Mostra mp_07_nuovo_itinerario_dati compilando le informazioni sul percorso con i dati inseriti dall'utente
	11	Preme il tasto Salva	
	12		Salva l'itinerario con percorso

EXTENSION_1: Tutto si svolge correttamente e l'utente inserisce un nuovo itinerario senza percorso	Step n°	Attore Utente autenticato	Sistema	Servizio Mappe
	3.1	Preme il tasto Salva		
	4.1		Mostra la dialog <i>dl_02_avviso_percorso</i>	
	5.1	Preme il tasto SI		
	6.1		Salva l'itinerario senza percorso	
EXTENSION_2: L'utente inserisce le posizioni geografiche interagendo con la mappa	Step n°	Attore Utente autenticato	Sistema	Servizio Mappe
	5.2	Preme su un punto della mappa		
	6.2		Interroga il Servizio Mappe per ottenere l'indirizzo corrispondente alle coordinate della posizione geografica	
	7.2			Risponde restituendo gli indirizzi richiesti
	8.2		Riempe i campi punto di inizio e punto di fine con gli indirizzi ricevuti, torna nel MAIN_SCENARIO	
EXTENSION_3: L'utente compila parzialmente i campi obbligatori nome, durata e difficoltà in <i>mp_07_nuovo_itinerario</i>	Step n°	Attore Utente autenticato	Sistema	Servizio Mappe
	2.3	Compila solo alcuni tra i campi obbligatori nome, durata e difficoltà		
	3.3 – 11.3		Tutto si svolge come nel MAIN_SCENARIO	
	12.3		Avvisa l'utente che i campi nome, durata e difficoltà vanno compilati	
	13.3	Inserisce le informazioni mancanti		
	14.3		Salva l'itinerario	
EXTENSION_4: L'utente inserisce solo un punto di inizio o un punto di fine del percorso	Step n°	Attore Utente autenticato	Sistema	Servizio Mappe
	5.4	Compila solamente uno tra i campi punto di inizio e punto di fine	Di pari passo con l'inserimento degli indirizzi, il sistema interroga Servizio Mappe , mostrando una lista di indirizzi suggeriti	
	6.4 – 9.4		Tutto si svolge come nel MAIN_SCENARIO	
	10.4		Avvisa l'utente che il percorso inserito non sembra essere completo	

1.1.7 – Prototipazione visuale

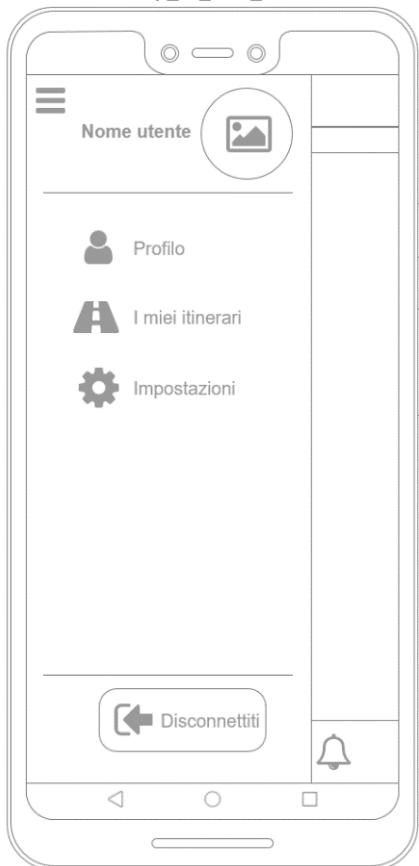




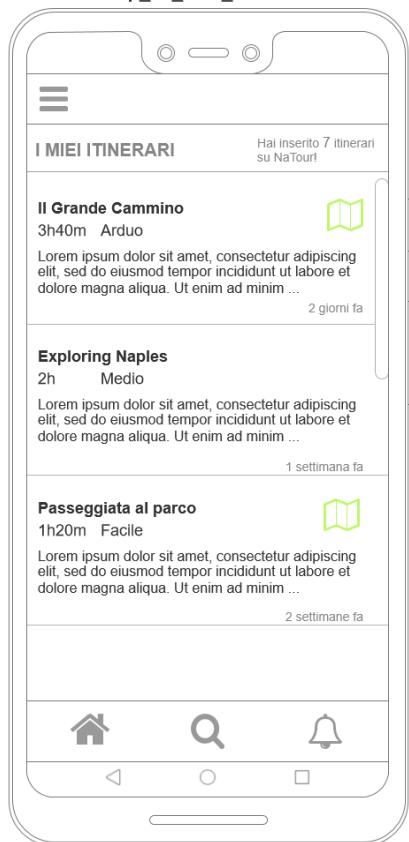
mp_10_dettagli_itinerario



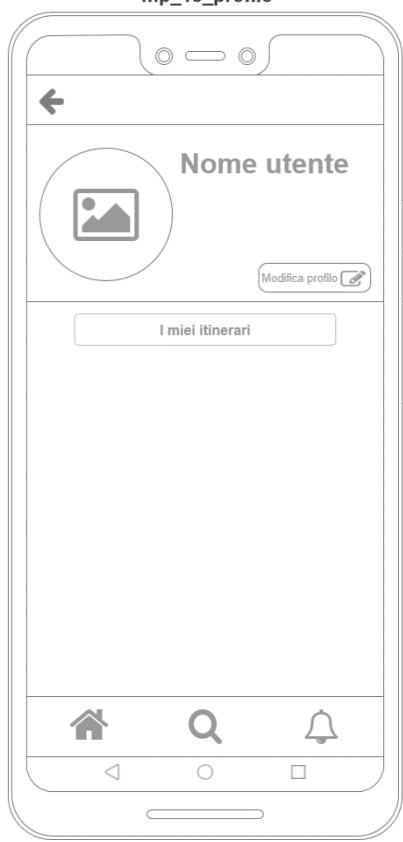
mp_11_side_bar



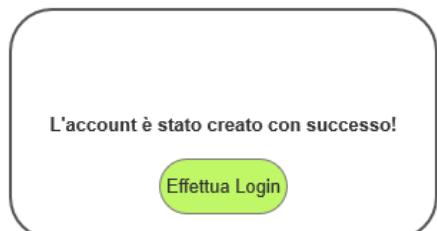
mp_12_miei_itinerari



mp_13_profilo



dl_01_account_creato



dl_02_avviso_percorso

Attenzione, l'itinerario che si vuole creare non
è dotato di un percorso.

Crearlo comunque?

NO

SI

dl_03_durata_itinerario

Inserire la durata dell'itinerario

giorni	ore	minuti
01	09	29
02	10	30
03	11	31

[OK](#)

This block contains a rounded rectangular frame with a thin black border. It includes a header asking to enter the itinerary duration. Below the header is a table with three columns: "giorni", "ore", and "minuti". Each column has three options: 01, 02, or 03 for days; 09, 10, or 11 for hours; and 29, 30, or 31 for minutes. At the bottom of the frame is a green "OK" button.

1.1.8 – Valutazione dell’usabilità a priori

Per valutare l’usabilità dell’applicativo a priori è stata effettuata una sessione di test di compito, dove sono state generate delle task da effettuare e sono stati affidati ad un numero ristretto di utenti. Gli utenti hanno potuto interagire con un prototipo interattivo, composto dai mockup mostrati nella sezione precedente. Le task che gli utenti hanno dovuto svolgere sono:

- **Task #1:** effettuare la registrazione e accedere con il proprio account
- **Task #2:** visualizzare i dettagli di un itinerario dalla schermata home
- **Task #3:** visualizzare uno degli itinerari che hai inserito precedentemente

Di seguito sono mostrati i risultati dei test, eseguiti su tre utenti:

	Task #1	Task #2	Task #3
User #1	✓	✓	✓
User #2	✓	✓	✗
User #3	✓	✓	✗

Tasso di successo = 7 successi / 9 test totali = 77%

Dall’osservazione degli utenti sono risultati evidenti alcuni problemi, per fortuna non bloccanti, con l’interfaccia grafica dell’applicativo. In particolare, per quanto riguarda la **Task#3**, alcuni utenti hanno avuto difficoltà a trovare la schermata **mp_12_miei_itinerari** dove vengono mostrati gli itinerari inseriti dall’utente.

Soluzione al problema citato:

Prima dell'esecuzione dei test di usabilità, il procedimento per arrivare alla schermata **mp_12_miei_itinerari** è stato il seguente:

- a. dalla schermata **mp_06_home** aprire il menu laterale **mp_11_side_bar**
- b. nel menu laterale cliccare sul pulsante profilo
- c. nella schermata **mp_13_profilo** cliccare sul pulsante “I miei itinerari”

Il vecchio menu laterale infatti non permetteva l'accesso diretto agli itinerari dell'utente, per cui è stato aggiunto un pulsante che facilita il raggiungimento di quella schermata in maniera più veloce e comoda:



Inoltre, sono state apportate altre modifiche sull'interfaccia grafica, basandosi su alcuni consigli provvenienti dagli utenti. Ad esempio, nella schermata **mp_07_nuovo_itinerario_dati** è stata modificata la modalità di inserimento della durata dell'itinerario: è stata aggiunta una dialog composta da number picker per un inserimento più comodo e meno prone ad errori, la **dl_03_durata_itinerario**

NUOVO ITINERARIO

Nome itinerario
Durata
Difficoltà
Descrizione (opzionale)

NUOVO ITINERARIO

Nome itinerario
Durata

Difficoltà
Descrizione (opzionale)

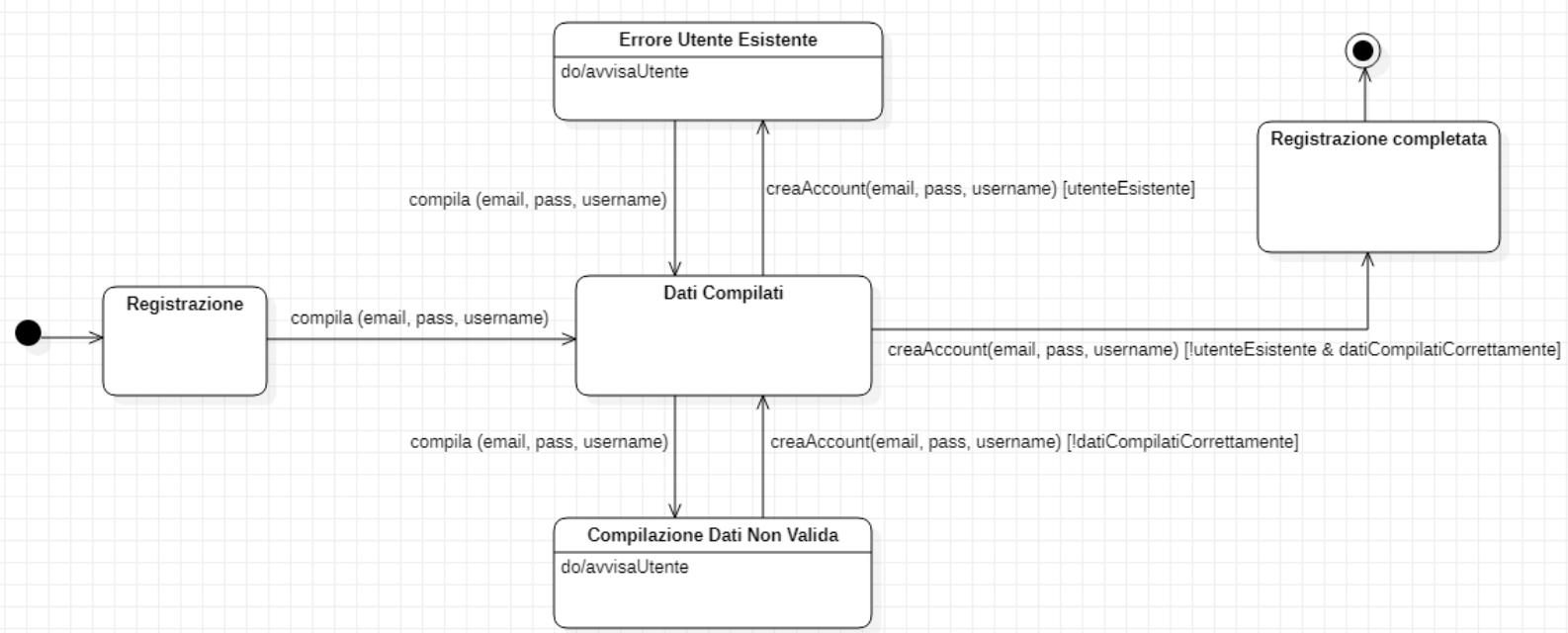
dl_03_durata_itinerario

Inserire la durata dell'itinerario

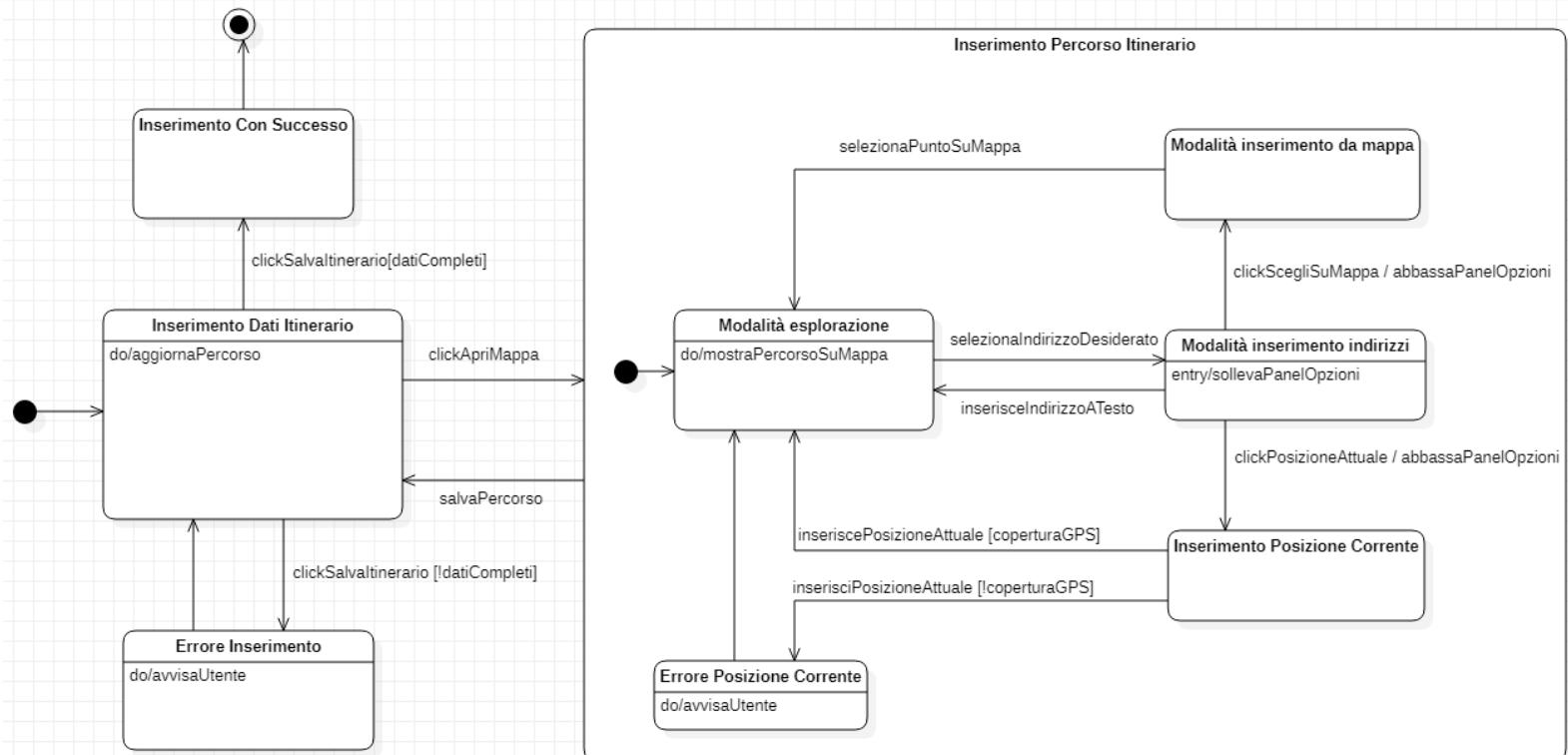
giorni	ore	minuti
01	09	29
02	10	30
03	11	31

1.1.9 – Prototipazione funzionale

1 – Crea Account



2 – Inserisci Itinerario con Mappa



1.1.10 – Tabella delle funzionalità

Di seguito vengono mostrate le tabelle delle funzionalità dell'applicativo mobile. Le tabelle elencano le funzionalità del sistema, indicando per ognuna di essa il relativo livello di completezza. Per motivi di leggibilità e spazio la tabella verrà suddivisa in sottotabelle.

Sottotabella 1: funzionalità legate all'autenticazione:

	Crea account	Effettua accesso	Recupera account
Prototipazione visuale (mockup)	✓	✓	✓
Descrizione testuale dell'use case	✓		
Prototipazione funzionale (statechart)	✓		
Design	✓	✓	✓
Implementazione	✓	✓	✓
Unit Testing	✓		
Field testing	✓	✓	✓

Sottotabella 2: funzionalità principali dell'applicativo:

	Visualizza home	Inserisci itinerario mappa	Inserisci itinerario GPX	Visualizza dettagli itinerario	Visualizza itinerari propri	Rimuovi itinerario	Visualizza profilo
Prototipazione visuale (mockup)	✓	✓	✓	✓	✓	✓	✓
Descrizione testuale dell'use case		✓					
Prototipazione funzionale (statechart)		✓					
Design	✓	✓	✓	✓	✓	✓	✓
Implementazione	✓	✓	✓	✓	✓	✓	✓
Unit Testing		✓		✓			
Field testing	✓	✓	✓	✓	✓	✓	✓

1.2 - Modelli di dominio

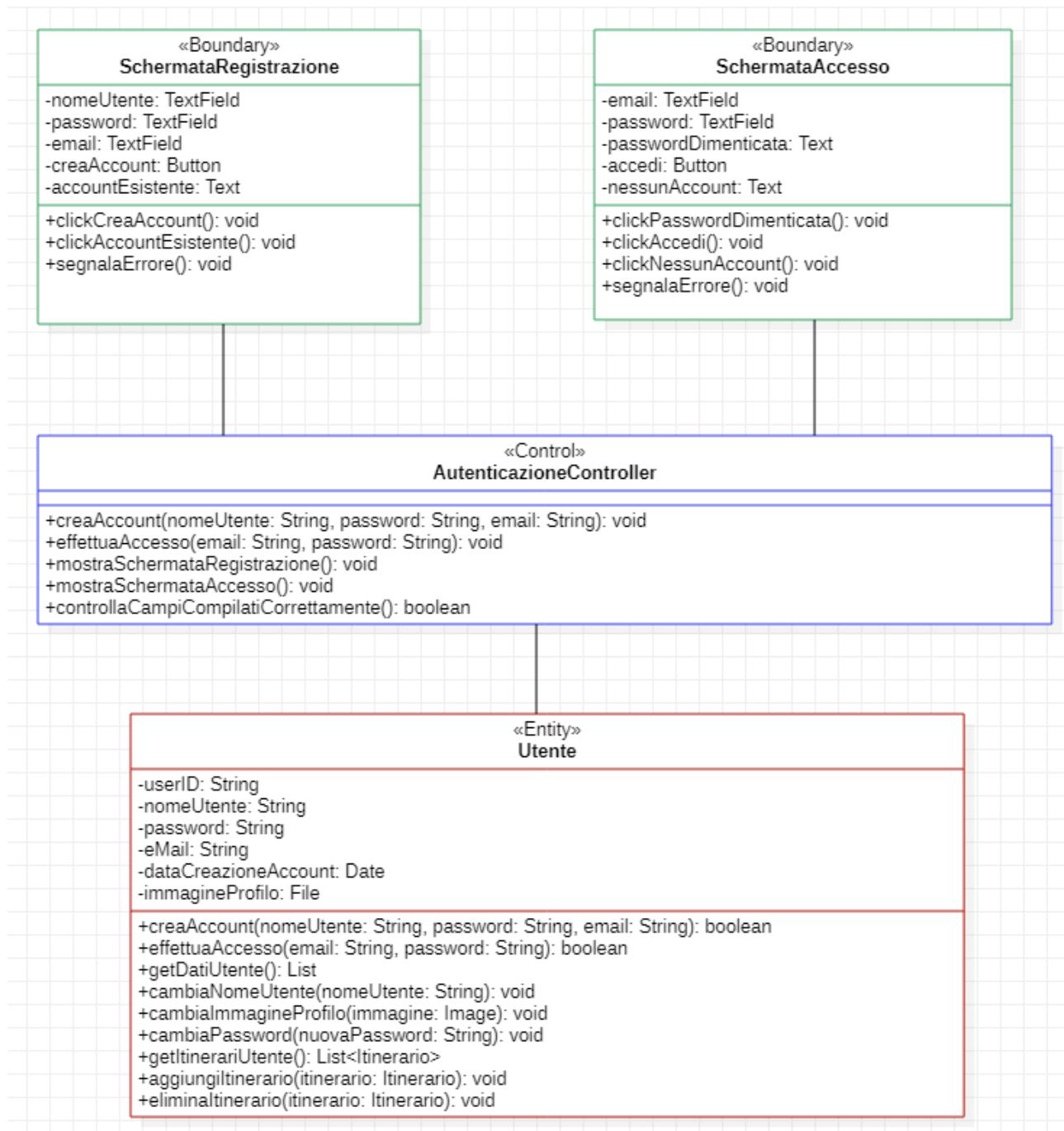
1.2.1 – Class diagram di analisi

Di seguito si trova il modello di dominio, ovvero una rappresentazione visuale delle classi relative al dominio. Prima vengono elencati i class diagram di analisi, che individuano gli oggetti in gioco in base all'euristica Three-Object-Type, dove gli oggetti vanno classificati in tre gruppi in base alla loro funzionalità:

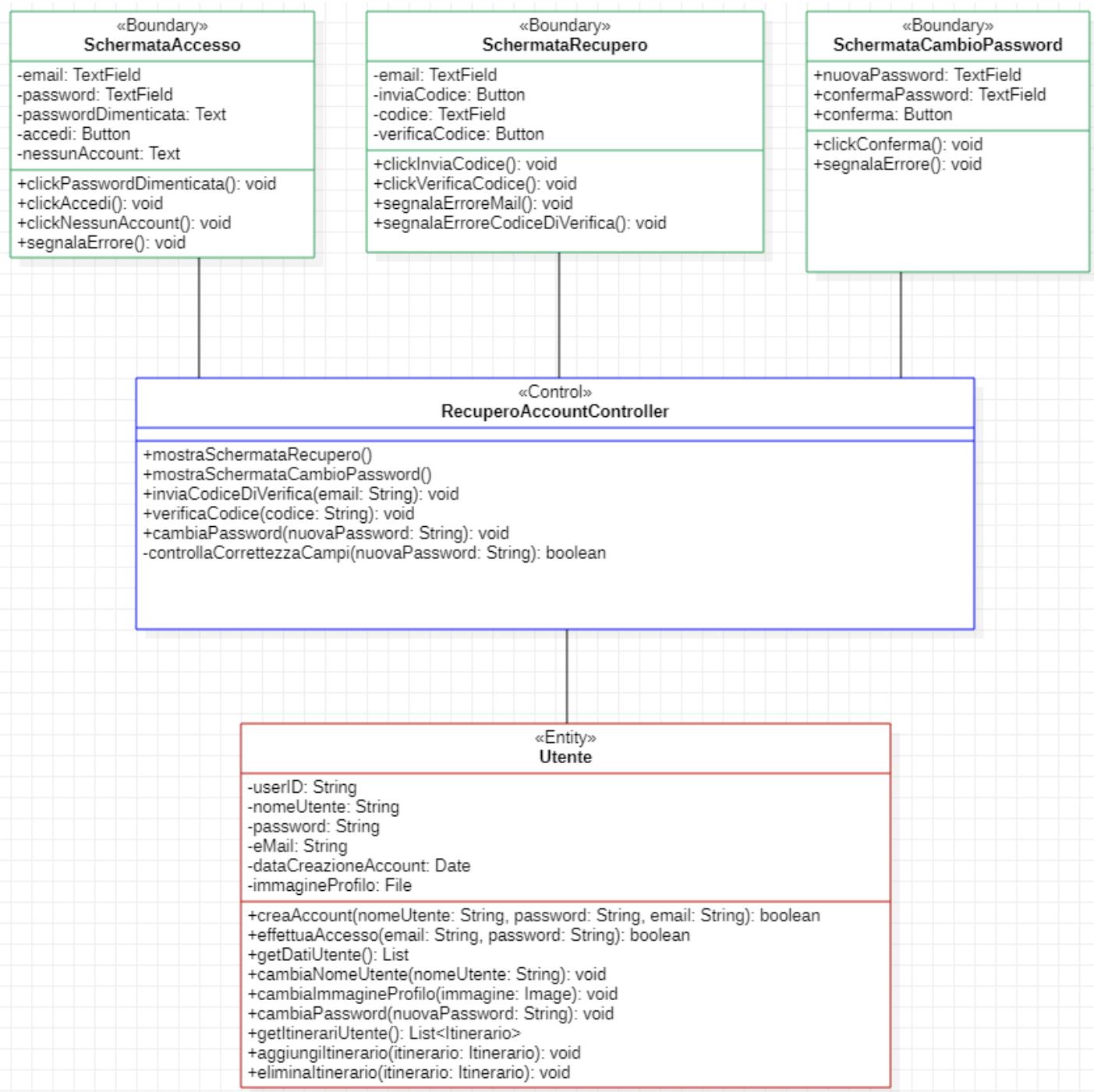
- Boundary – modellano l'interfaccia con la quale andranno ad interagire gli utenti
- Control – modellano la business logic
- Entity – modellano i concetti di dominio e l'informazione persistente

Per motivi di leggibilità le classi sono state raggruppate in singole schermate in base allo use case di appartenanza.

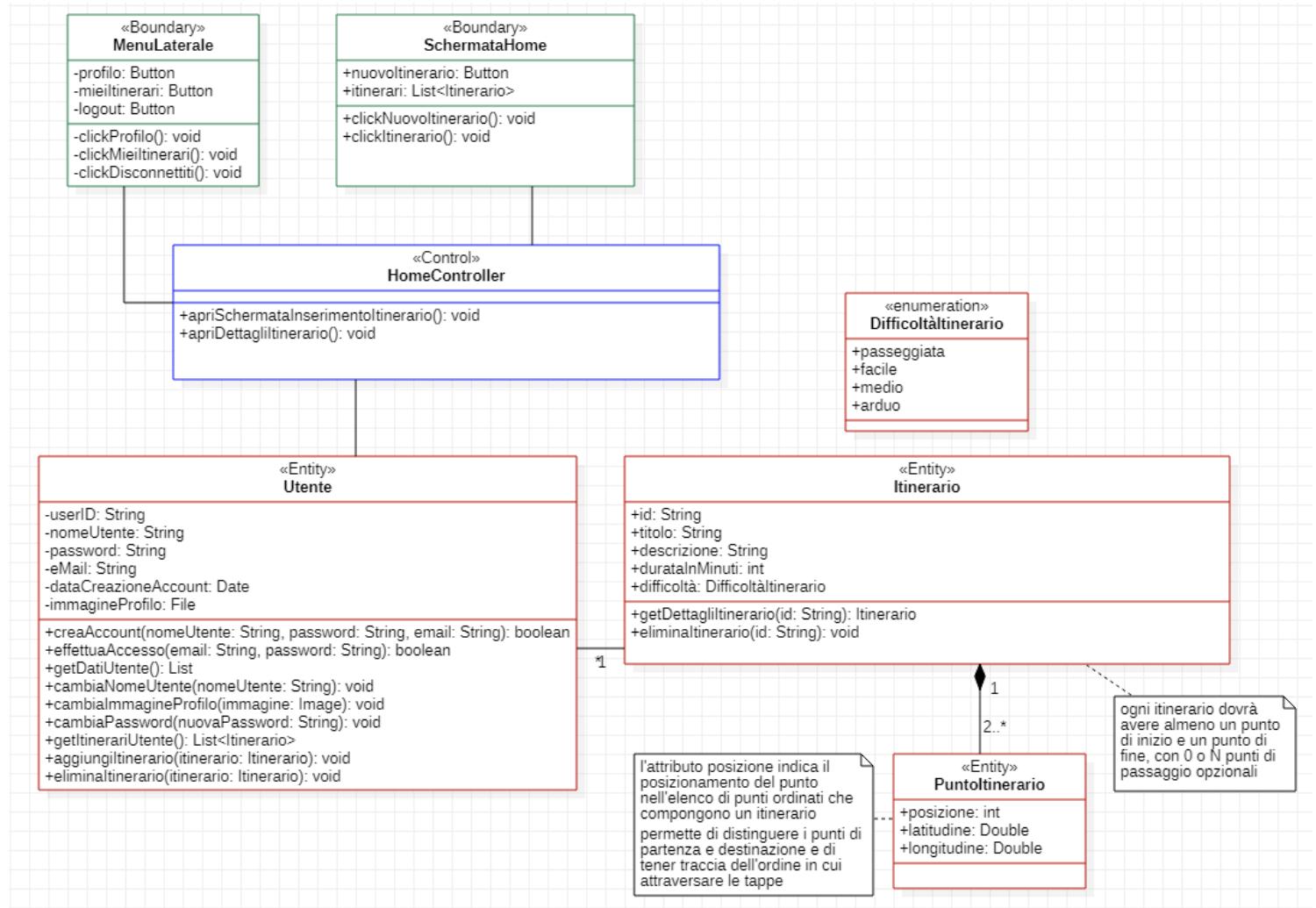
Autenticazione (effettua accesso + crea account)



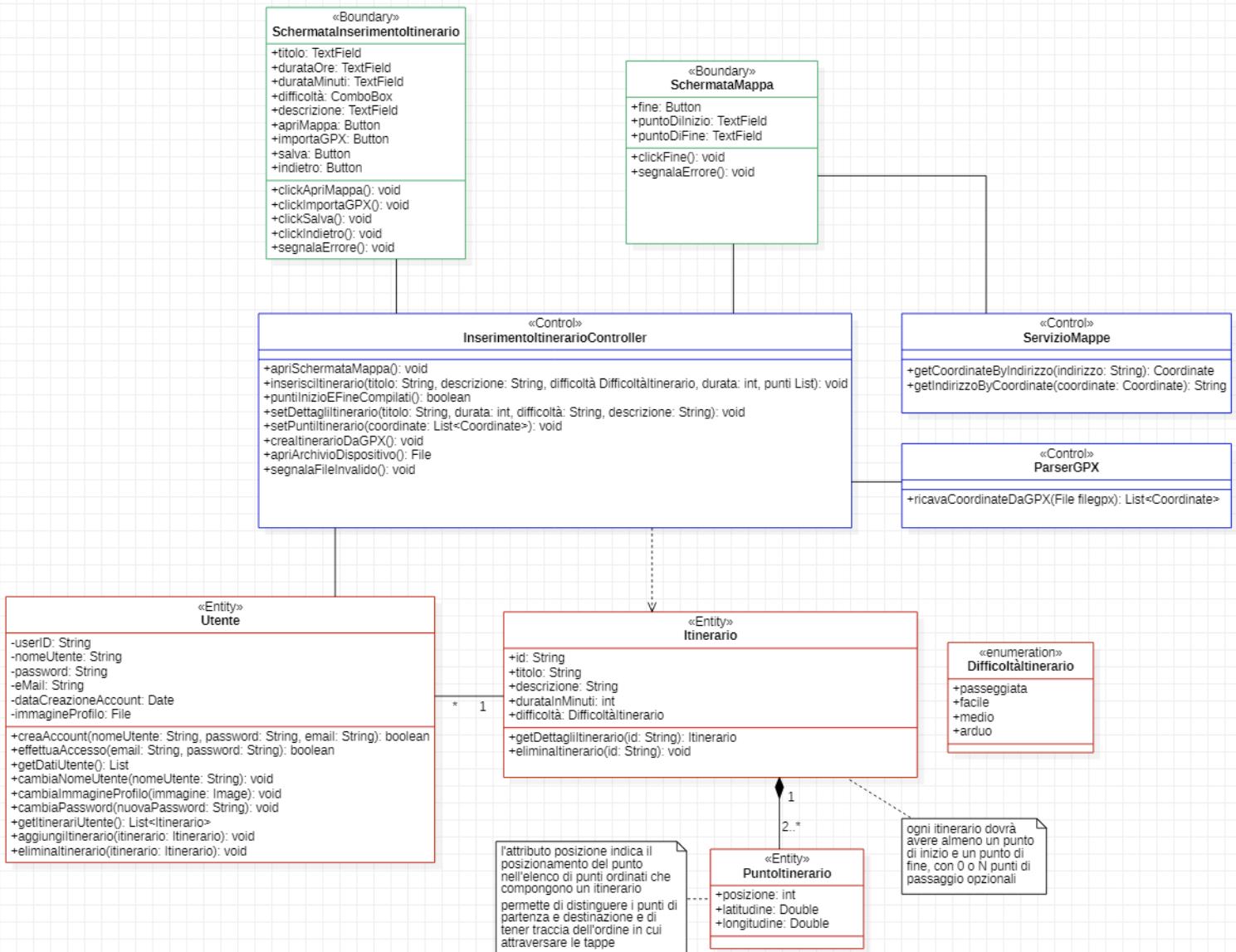
Recupera account



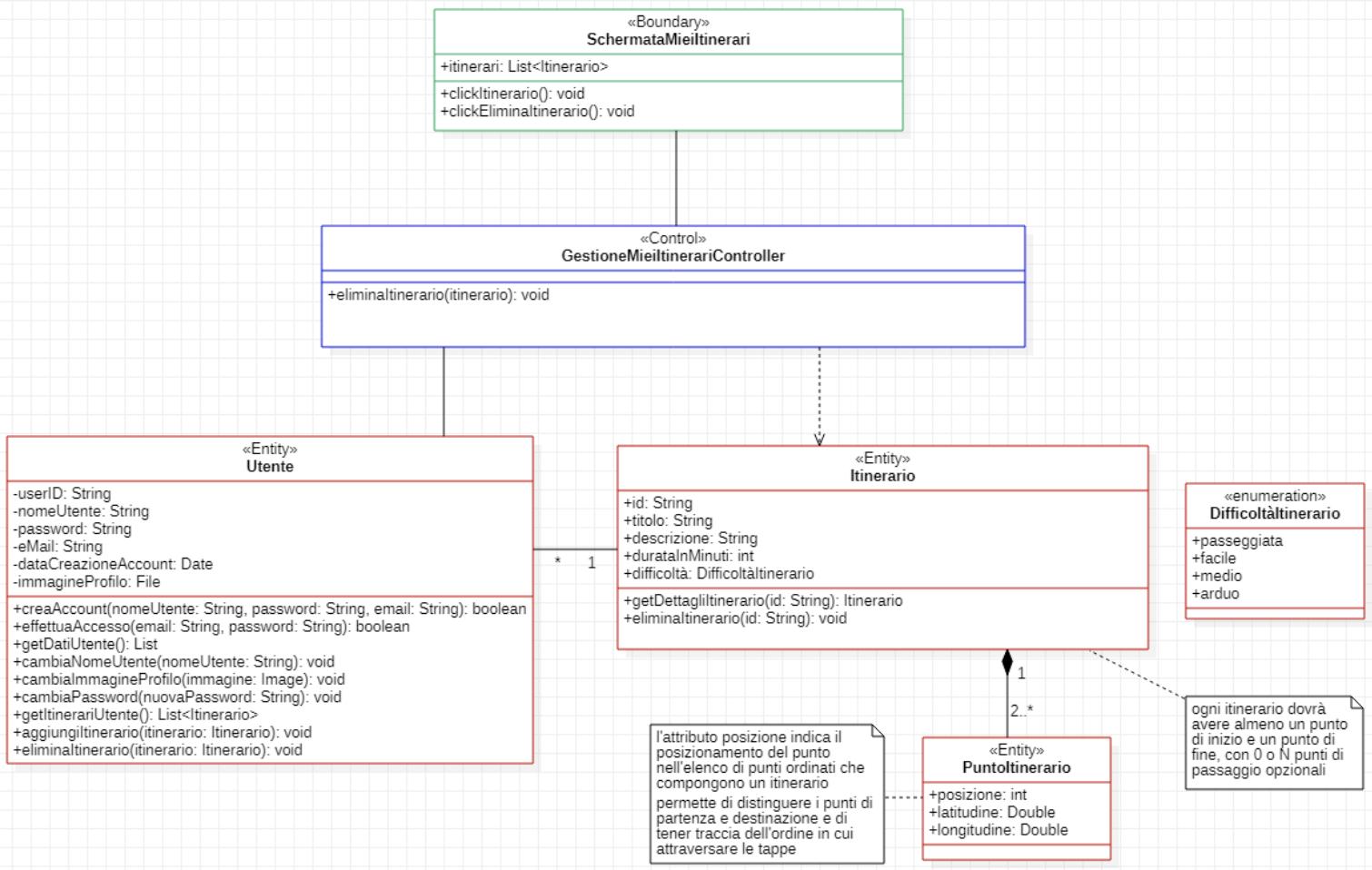
Visualizza Home



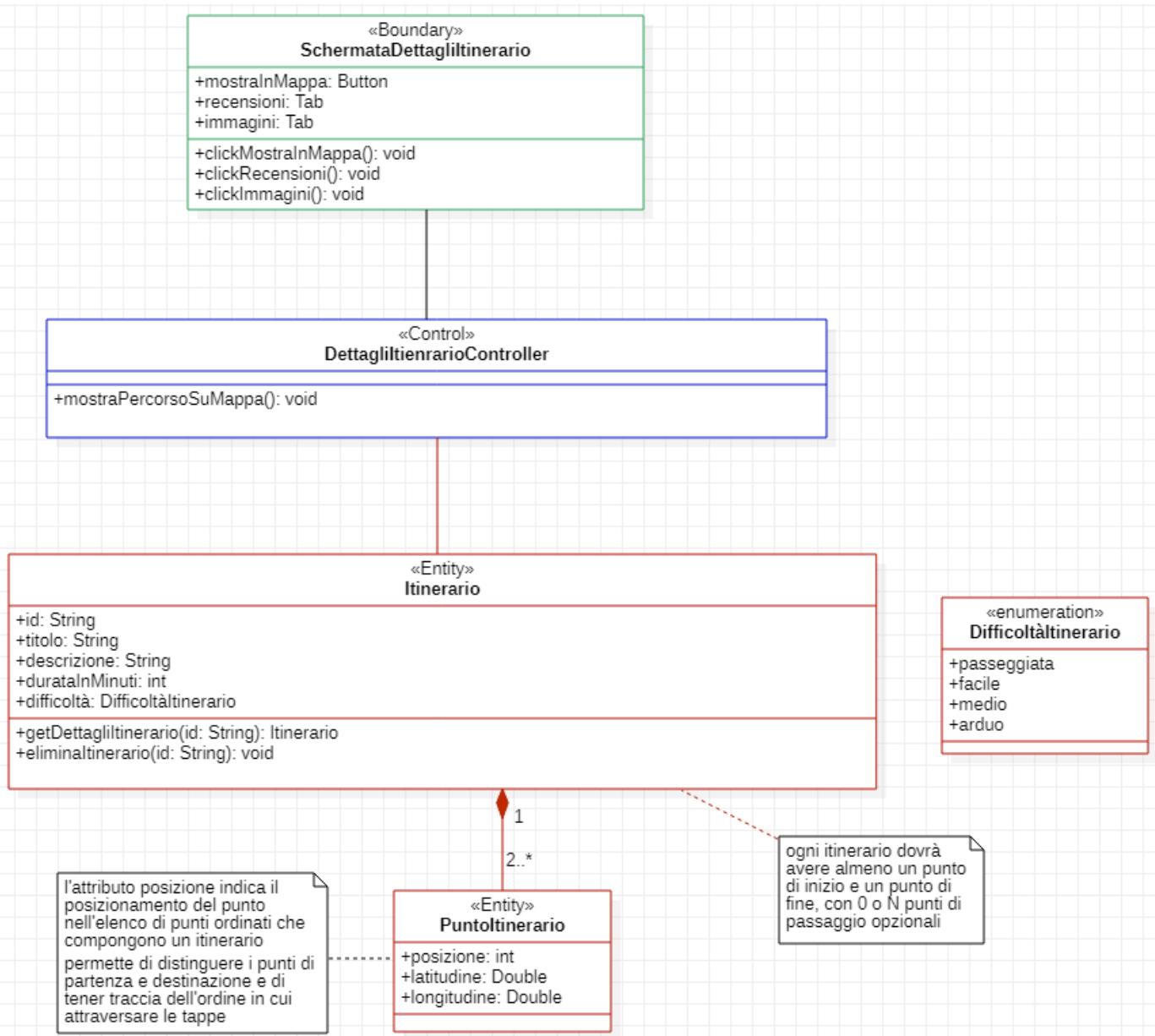
Inserisci itinerario



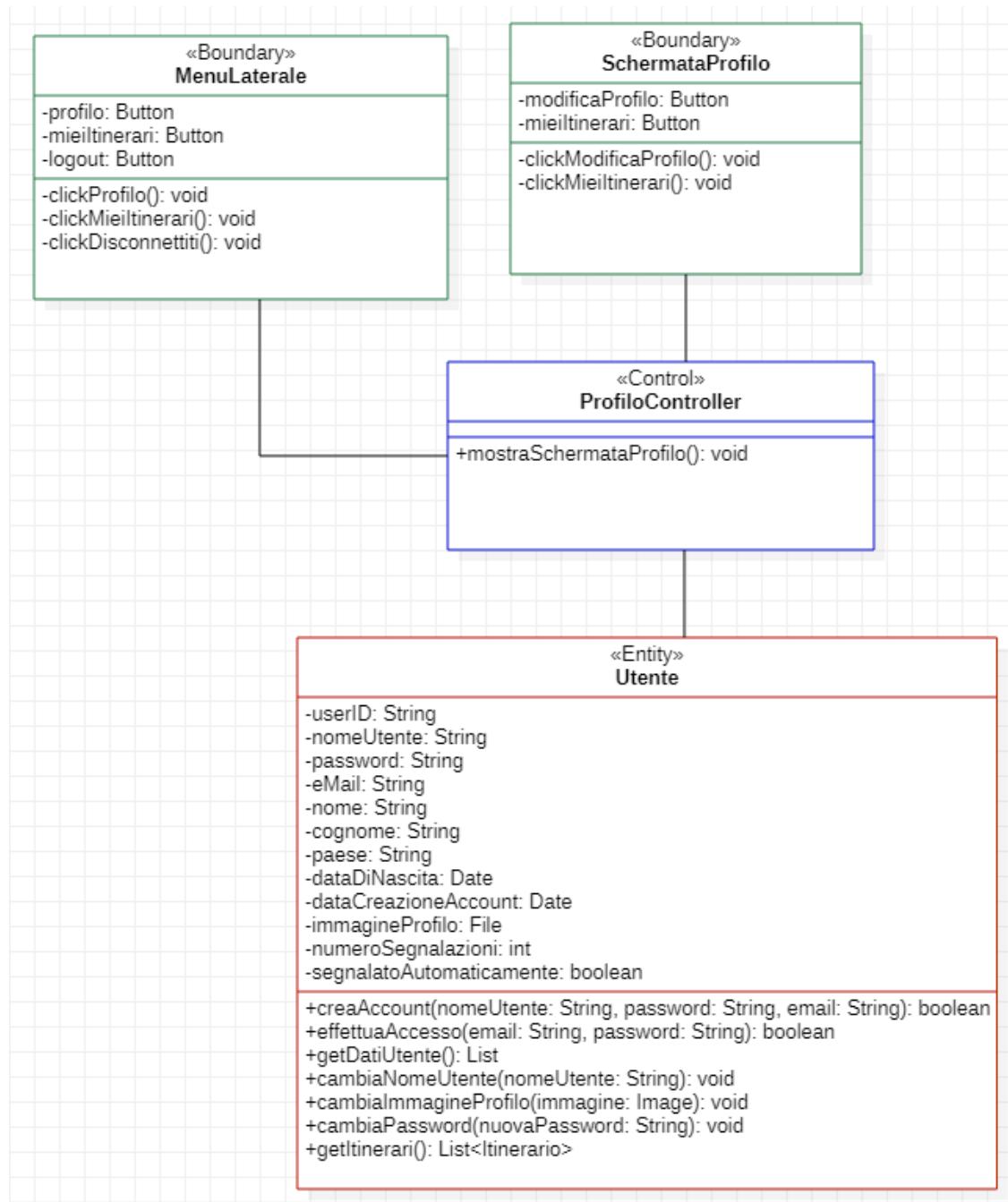
Visualizza Itinerario Propri



Visualizza Dettagli Itinerario



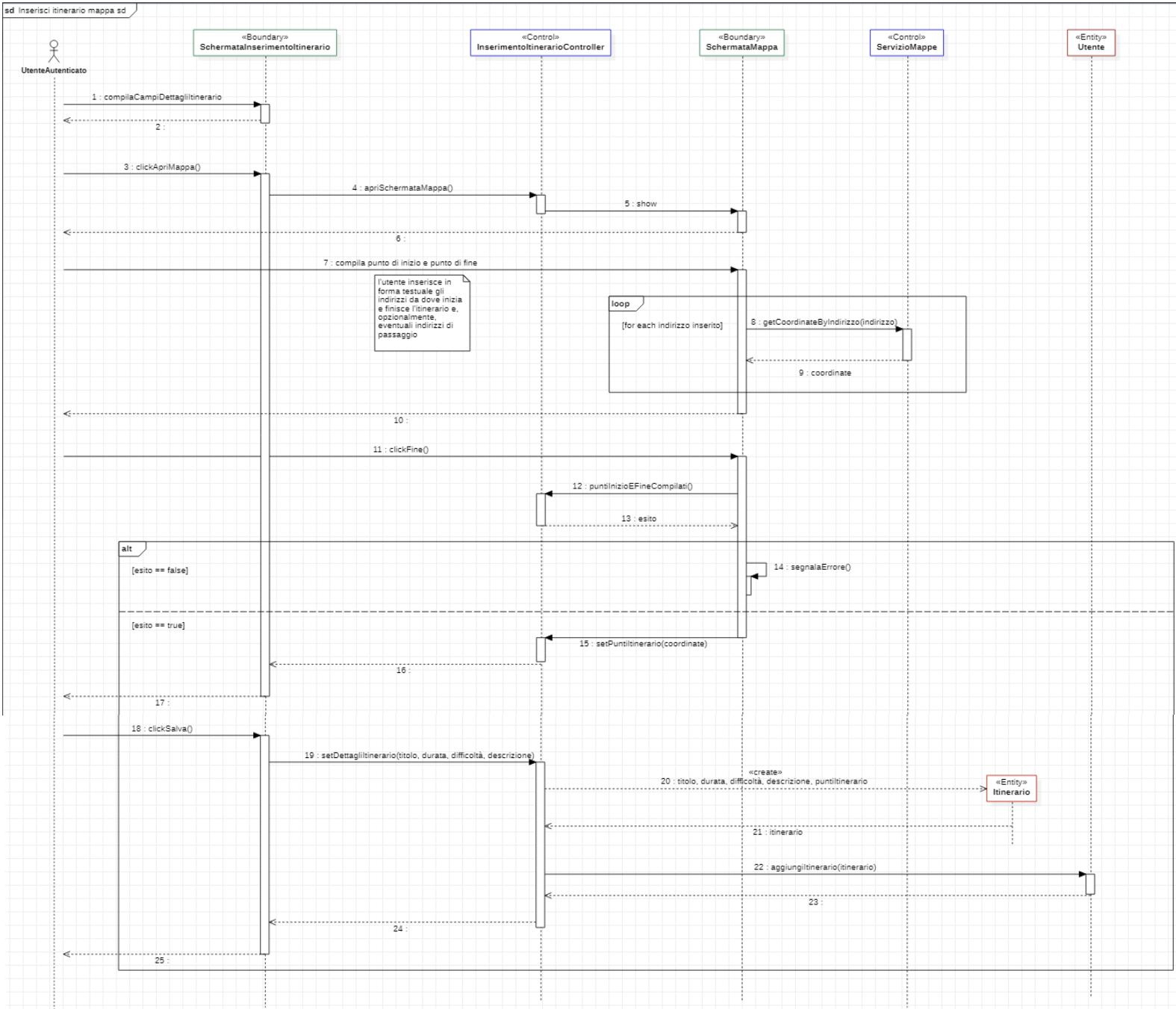
Visualizza Profilo



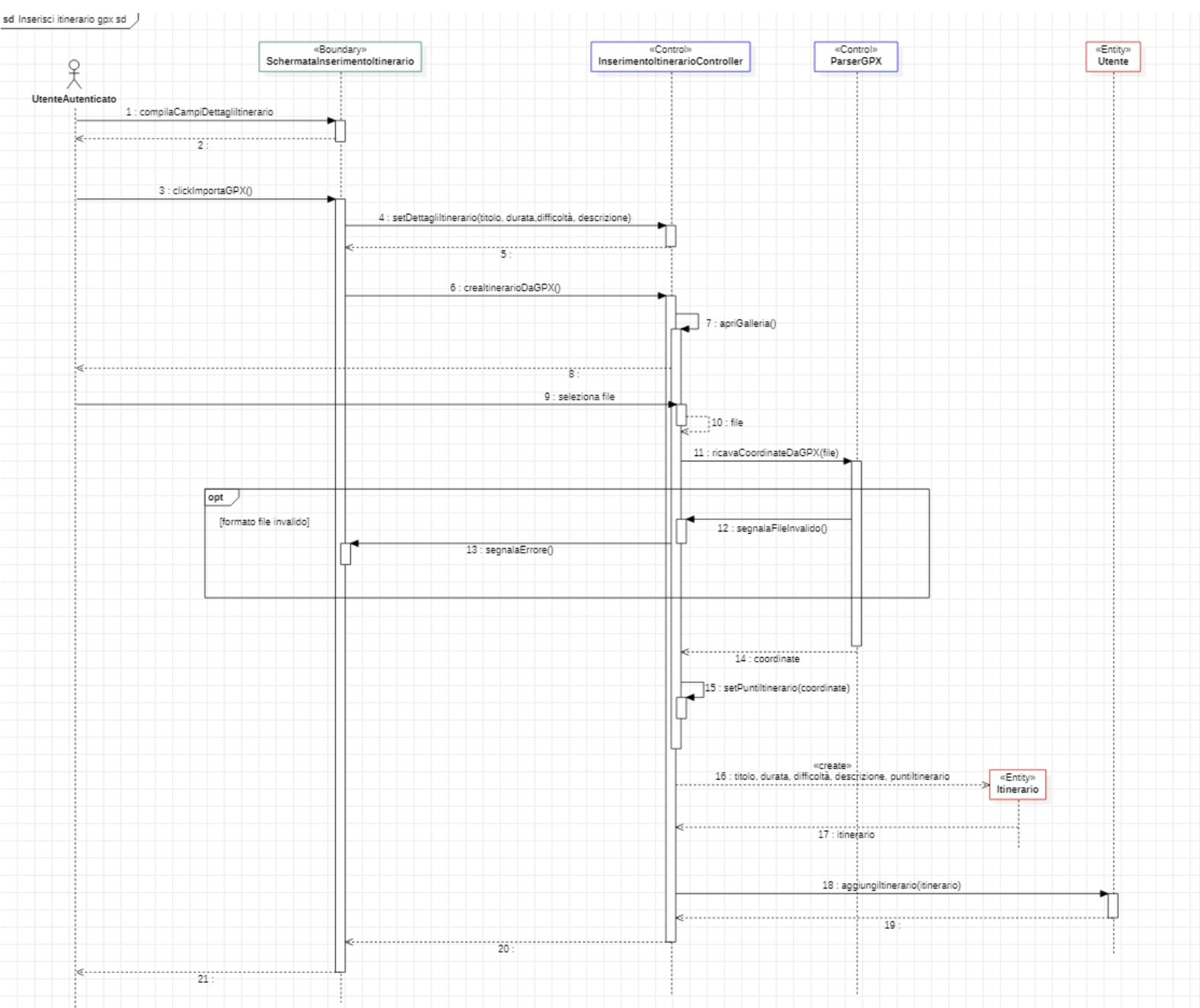
1.2.2 – Sequence diagram di analisi

Di seguito vengono elencati i diagrammi di sequenza per gli use case di inserimento itinerario: il primo tratta il caso in cui il percorso viene inserito dall’utente utilizzando la mappa interattiva, mentre il secondo nel caso in cui viene importato un file gpx contenente le informazioni geografiche del percorso.

Inserisci Itinerario Mappa

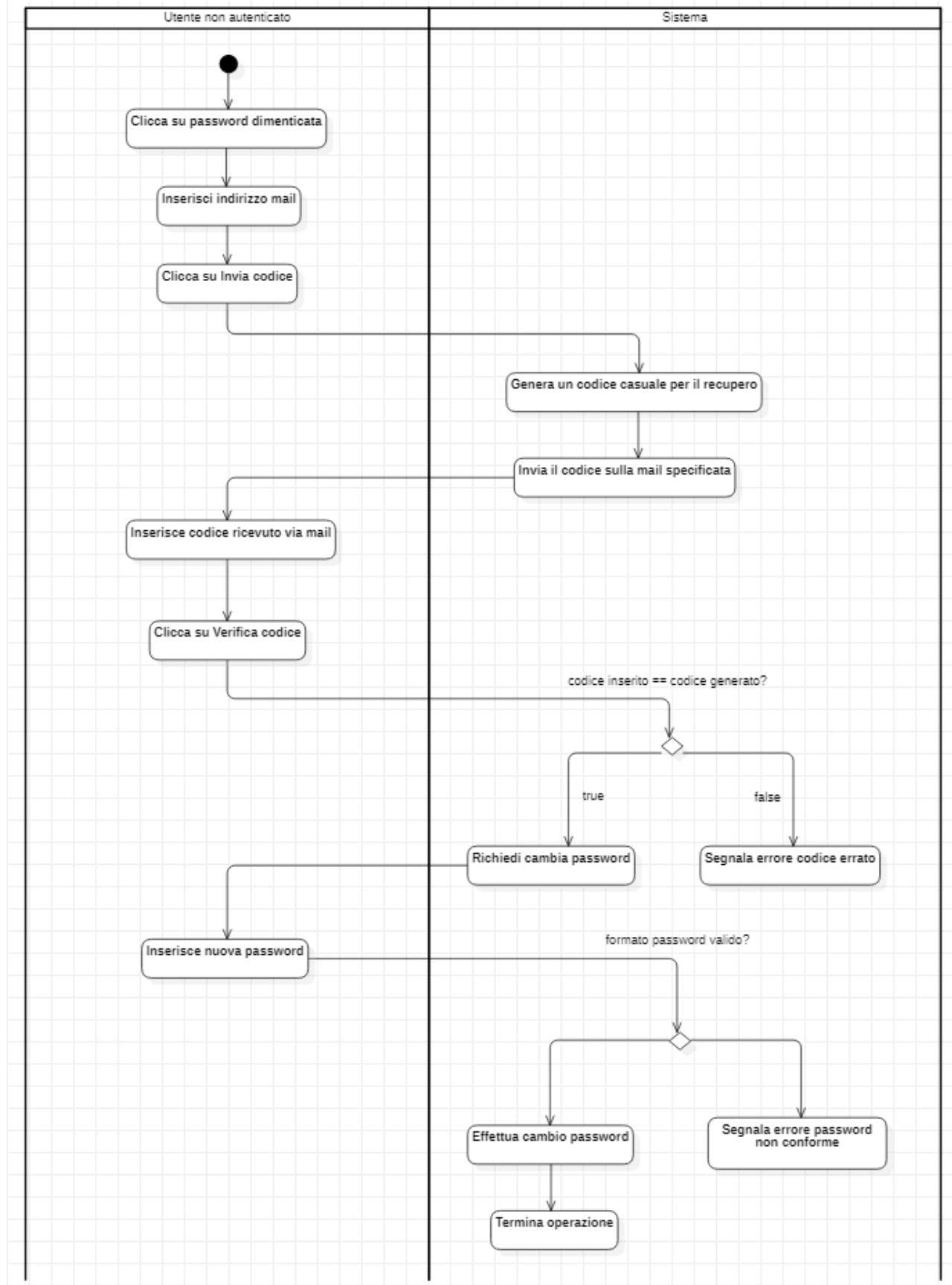


Inserisci Itinerario GPX



1.2.3 – Activity diagram

Recupero Account

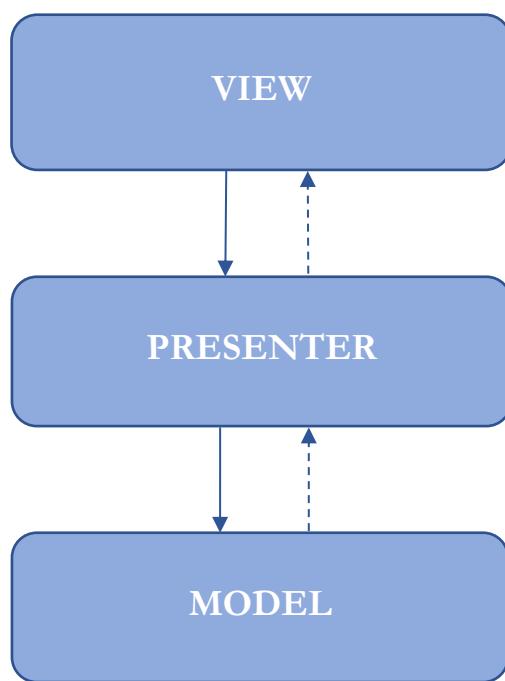


II - Documento di Design

2.1 - Archittettura del sistema

2.1.1 – Architettura del client

Il cliente mobile è stato realizzato su piattaforma Android, con l'ausilio di Android Studio. L'architettura utilizzata per il client mobile è il **MVP (Model-View-Presenter)**. Questo è un tipo di architettura a strati chiusa che promuove la **separation of concerns**, ovvero la distinzione degli oggetti in base al loro ruolo e alle loro responsabilità.



In particolare:

- la **View** è composta dall'insieme di elementi di Android che compongono l'interfaccia utente dell'applicativo: Activity, Fragment e Dialog. Sono soggette al ciclo di vita dell'ambiente Android;
- lo strato **Presenter** è un mediatore tra il View e il Model che si occupa della business logic, ed è composto da tutte le classi con suffisso “Presenter”;
- il **Model** è composto dagli oggetti di dominio sui quali si interagisce nell'interfaccia e dalle classi responsabili per l'interfacciamento con il backend e le api esterne

Di seguito vanno elencati alcuni dei componenti e librerie più notevoli utilizzati nella realizzazione dell'applicativo mobile:

- **Jetpack Navigation**: il componente di navigazione si occupa di reagire alle interazioni dell'utente con l'applicativo. Permette di definire un grafo di navigazione che gestisce le transazioni da una schermata all'altra in modo semplice e veloce. In particolare, è stata utilizzata un'architettura Single-Activity, dove ciascuna schermata è rappresentata da un fragment separato.
- **Volley**: una libreria che facilita l'effettuazione di richieste http ad api esterne. Tra le sue funzionalità principali vi è lo scheduling delle richieste http e la possibilità di effettuare richieste in modo concorrente.
- **osmdroid**: una libreria per l'interazione con mappe geografiche su un dispositivo Android. Contiene una serie di strumenti e utilities per la manipolazione e la rappresentazione dei dati geografici e la creazione di percorsi.
- **JUnit5**: un framework per realizzare unit test di singoli componenti del sistema.

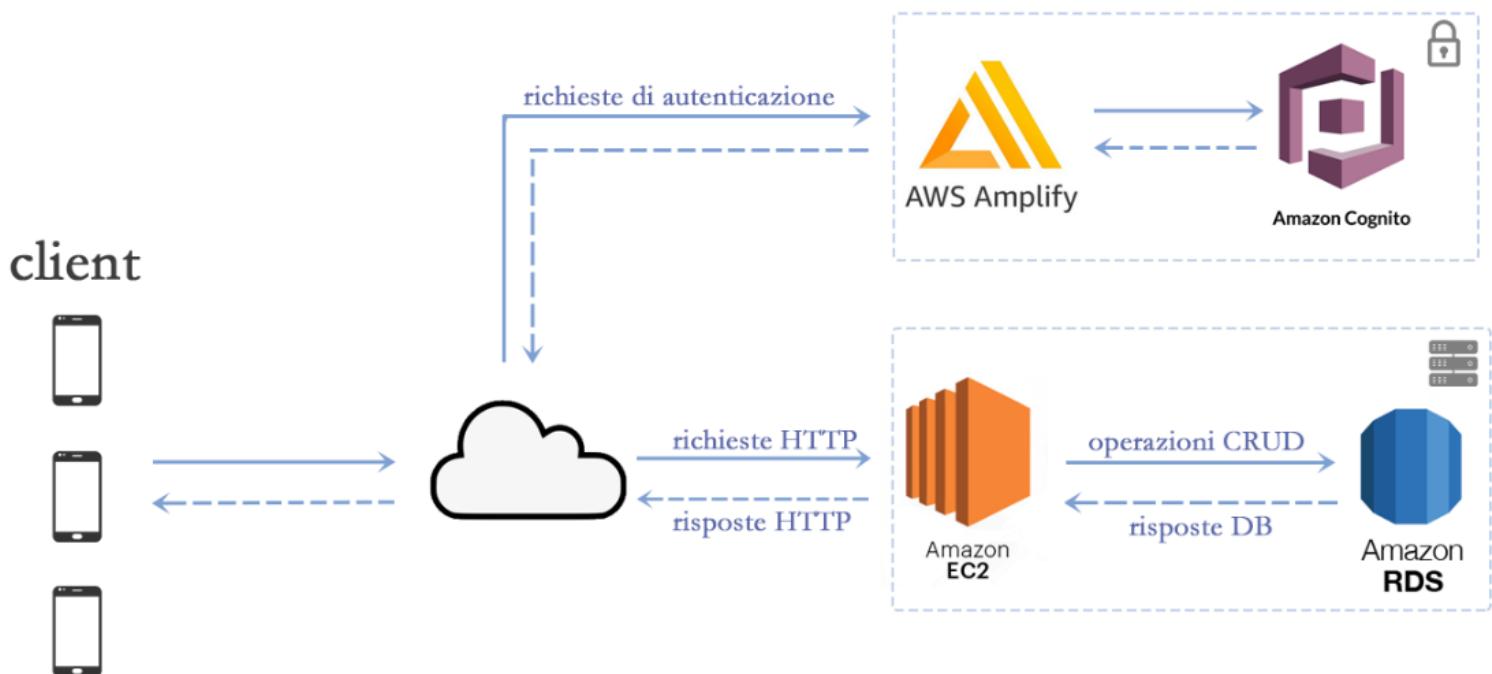
2.1.2 – Architettura del server

L'architettura ad alto livello del sistema è una di tipo **client-server**.

I dispositivi client quindi interagiscono con un back-end, richiedendo dei servizi e risorse mediante una comunicazione di tipo **request-response**.

L'architettura del back-end si fonda sui servizi offerti dalla piattaforma **AWS**, che è stata scelta per i numerosi vantaggi che offre:

- **Flessibilità**: AWS offre una vasta gamma di prodotti e servizi di varia natura. Si ha a disposizione un ambiente virtuale dove è possibile caricare il software e i servizi necessari per il funzionamento dell'applicazione.
- **Sicurezza ed affidabilità**: i servizi offerti dagli Amazon Web Services sono altamente sicuri ed affidabili, sia dal punto di vista fisico e operativo, sia a livello di software.
- **Scalabilità e prestazioni elevate**: AWS offre strumenti come Auto Scaling ed Elastic Load Balancing, che permettono di ridimensionare le risorse di calcolo e di storage in base alle esigenze e alle necessità dell'applicazione.



Come si può osservare nel diagramma architetturale, i client possono effettuare richieste verso quelli che sono i due principali gruppi di servizi back-end:

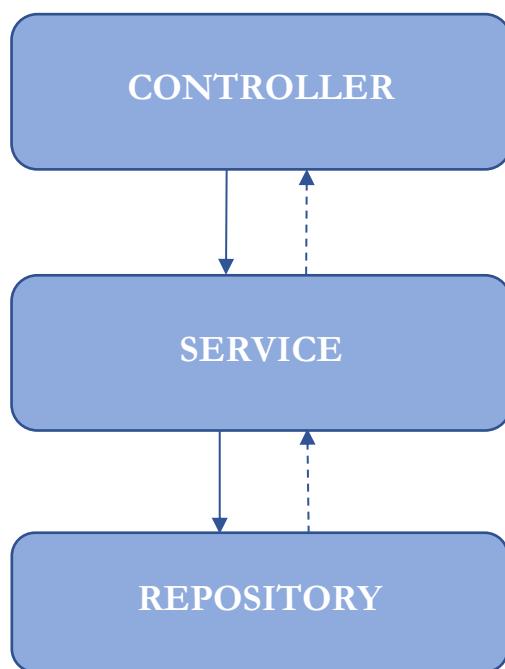
- **servizi di autenticazione:** alla base di questi vi è **Amazon Cognito**, che fornisce strumenti per l'autenticazione e la gestione degli utenti, sia con gli account interni all'applicativo, che mediante account di terze parti, come Google. E' stata utilizzata la libreria **Amazon Amplify** che facilita l'interfacciamento con Cognito e la configurazione delle user e identity pools;
- **server:** per l'hosting del server è stato scelto il servizio **Amazon EC2**, una piattaforma di calcolo che permette di configurare e lanciare istanze di elaboratori sicure ed affidabili sul cloud. Il principale vantaggio delle istanze EC2 è la possibilità di ridimensionarle in base alle necessità del sistema e al carico che si osserva in un determinato periodo, sia in numero, che come prestazioni dell'elaboratore individuale. I dati invece vanno persistiti all'interno di un database MySQL, che si trova su un'istanza di **Amazon RDS**, un servizio di hosting di database relazionali.

Per la realizzazione del server hostato sulle istanze EC2 è stato utilizzato il framework **Spring**, che presenta numerose features volte alla creazione di back-end in Java. Il framework permette di esportare il programma server come un file jar di dimensioni contenute e di lanciarlo nel giro di pochi secondi sulle istanze EC2.

In particolare, è stato usato il modulo **Spring REST**, che permette di realizzare delle API sicure e facilmente manutenibili, rimuovendo la maggioranza del codice boilerplate necessario a gestire le richieste e risposte HTTP, minimizzando così le probabilità di errori di programmazione.



Il server è stato realizzato con un'architettura chiusa fortemente consigliata dagli sviluppatori di Spring:



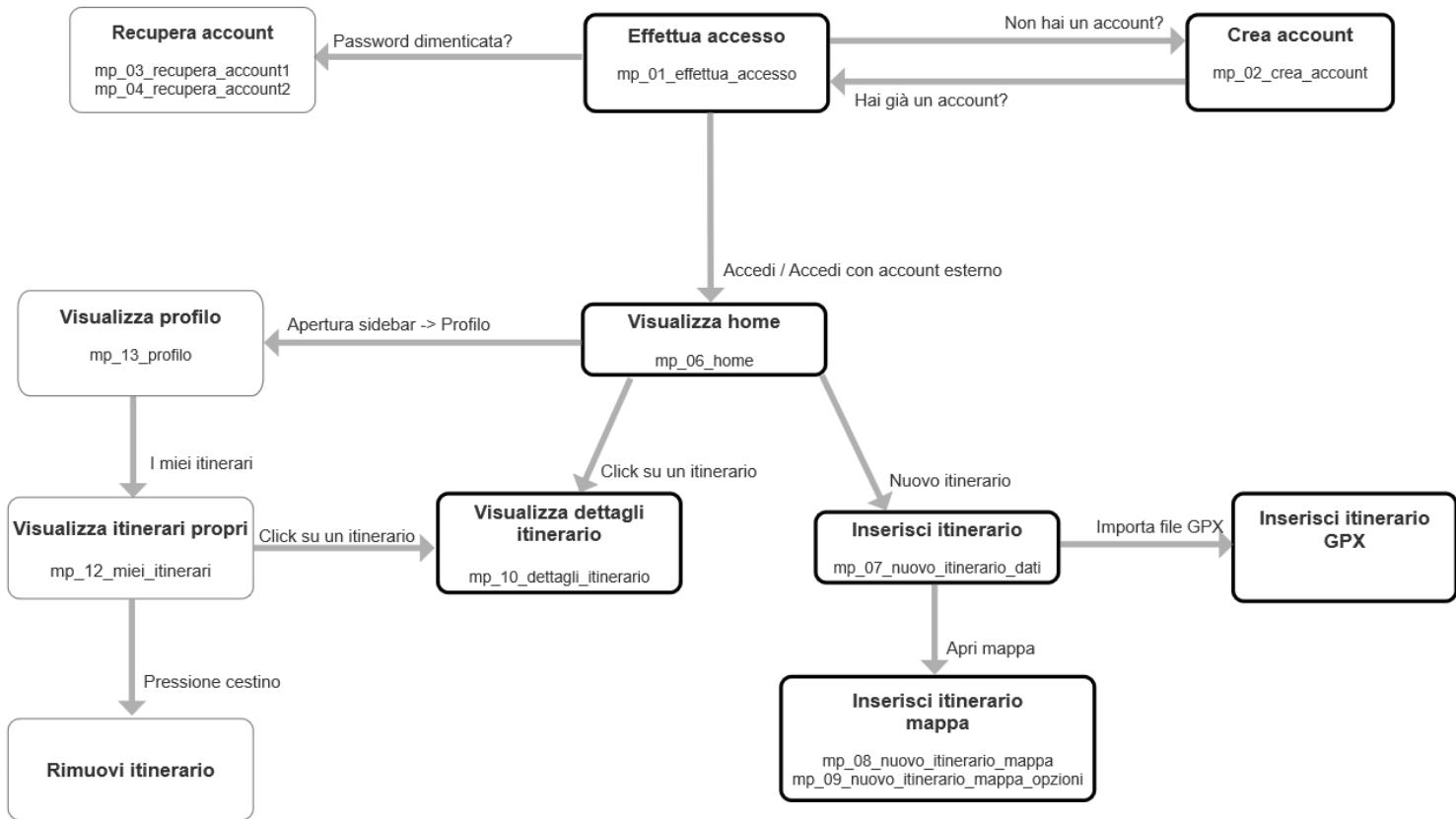
In particolare:

- lo strato **Controller** è composto da classi che fungono da endpoint: accettano e rispondono alle richieste HTTP del client;
- i **Service** rappresentano la business logic del server ed agiscono in base alle richieste provenienti dal controller;
- lo strato **Repository** è composto da tutti gli oggetti di dominio e dalle classi DAO che comunicano con il database sottostante mediante richieste CRUD

2.1.3 – Gerarchia funzionale

Di seguito viene riportata la gerarchia delle funzionalità, che permette di rappresentare la navigazione all'interno dell'applicativo mobile. Ad ogni funzionalità viene associato il mockup ad essa corrispondente. Le frecce rappresentano la transizione da una funzionalità ad un'altra, e sono accompagnate dall'affordance che permette all'utente di interagire con l'applicativo.

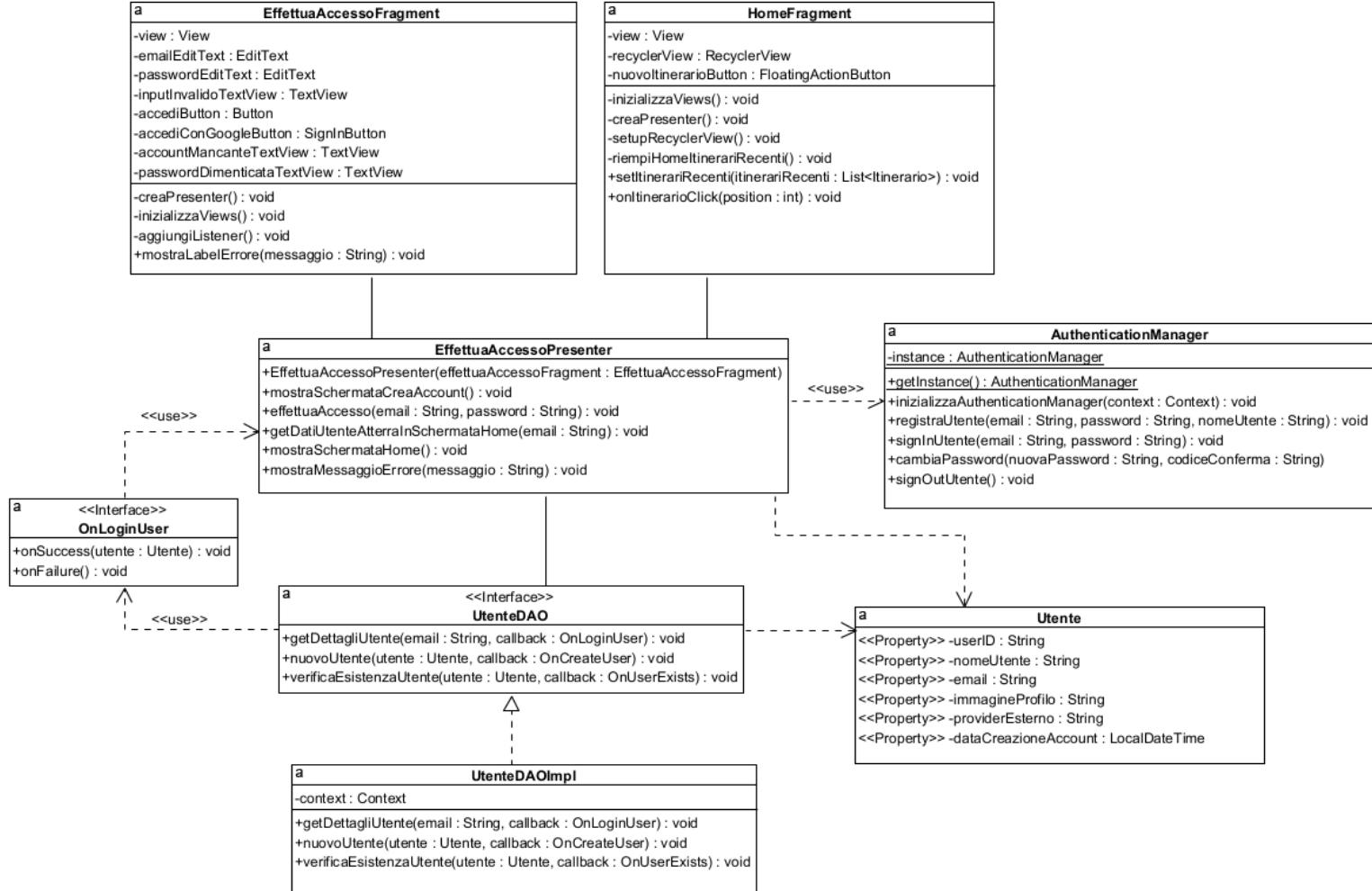
Inoltre, si è effettuata una valutazione sull'importanza delle funzionalità nel contesto dell'applicazione. Prendendo in considerazione le necessità del cliente, ad alcune delle funzionalità è stata data una priorità più alta. Tali funzionalità sono individuabili nel diagramma dal loro bordo più spesso.



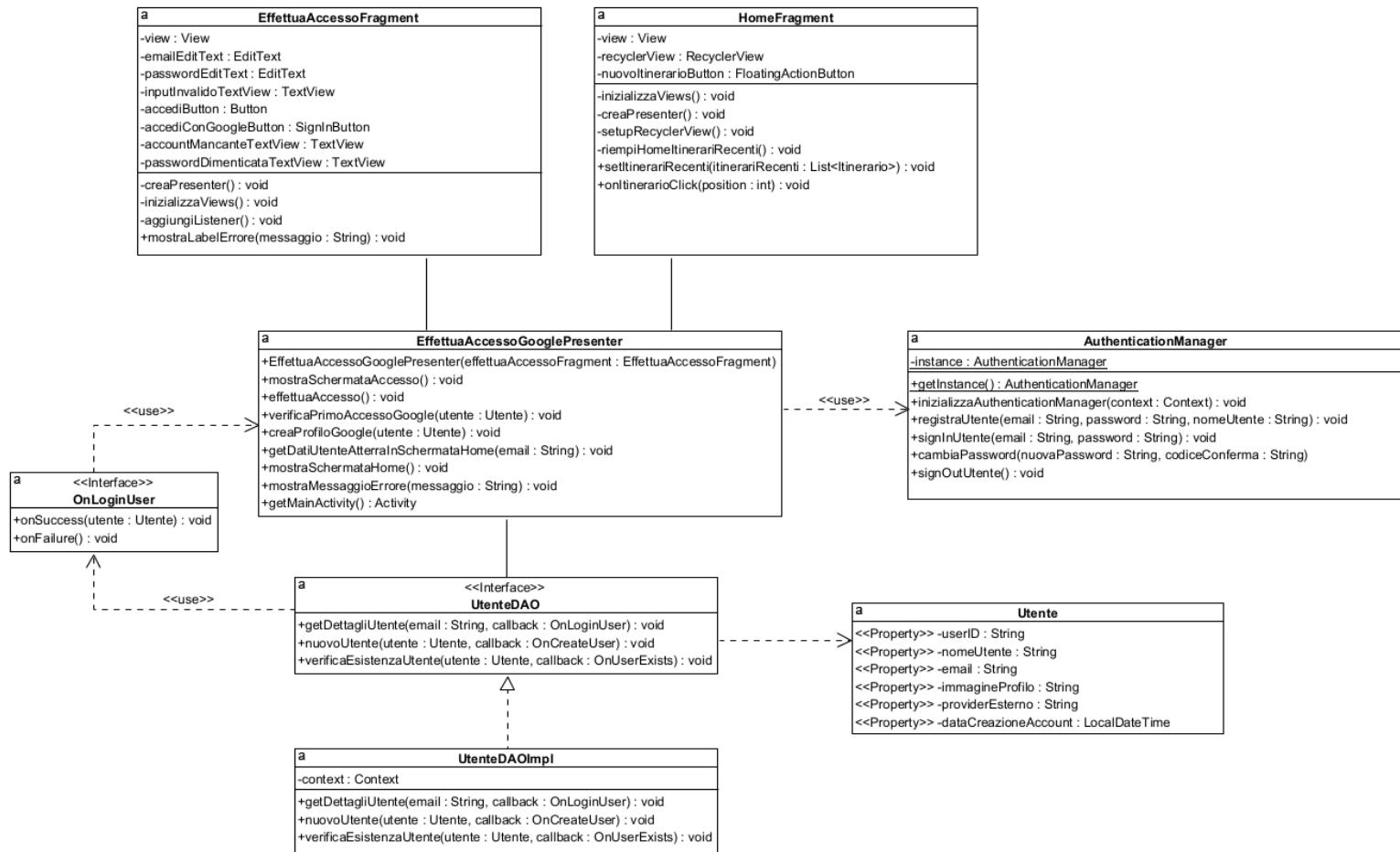
2.2 – System Design

2.2.1 – Class diagram di design

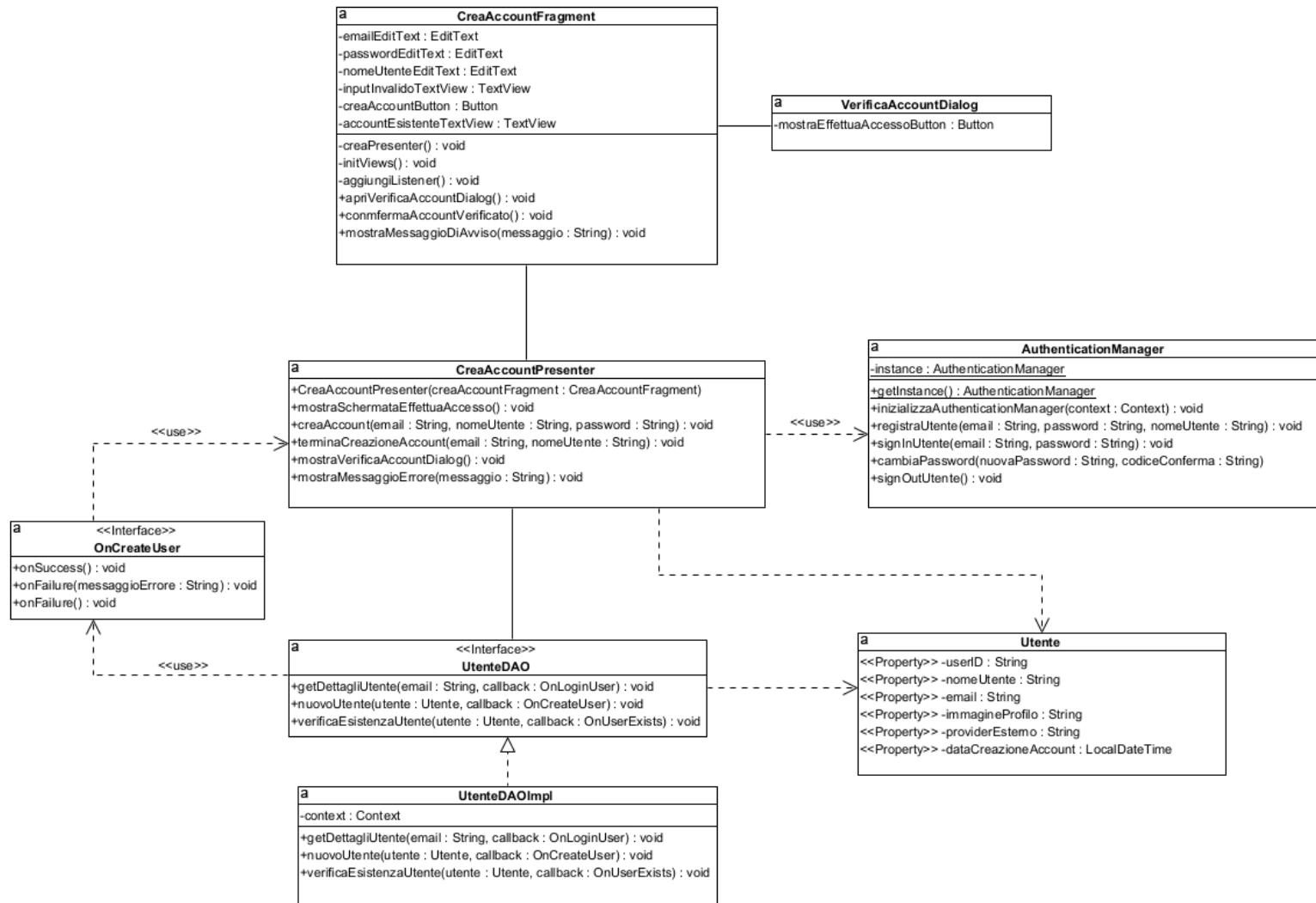
Effettua Accesso



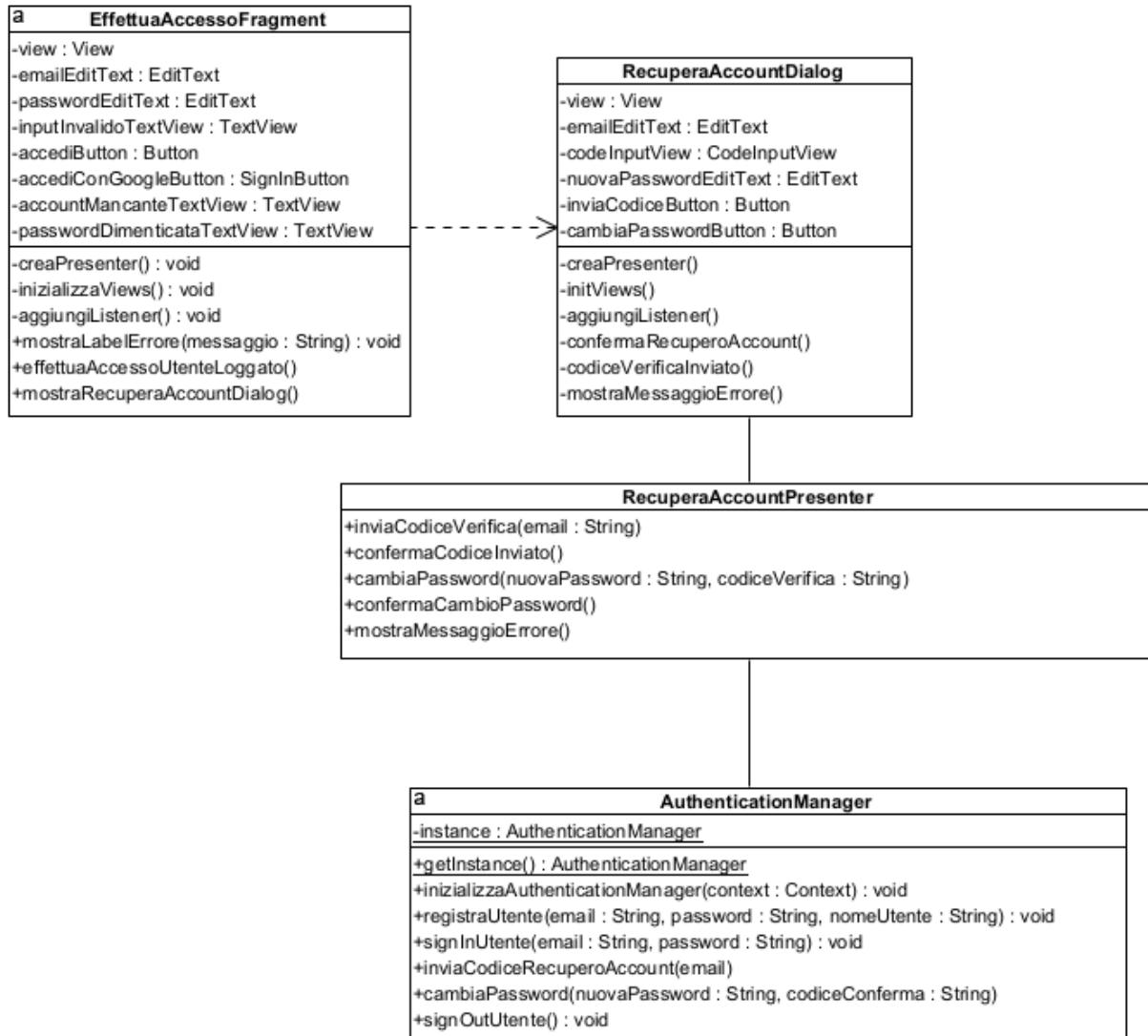
Effettua Accesso (Google)



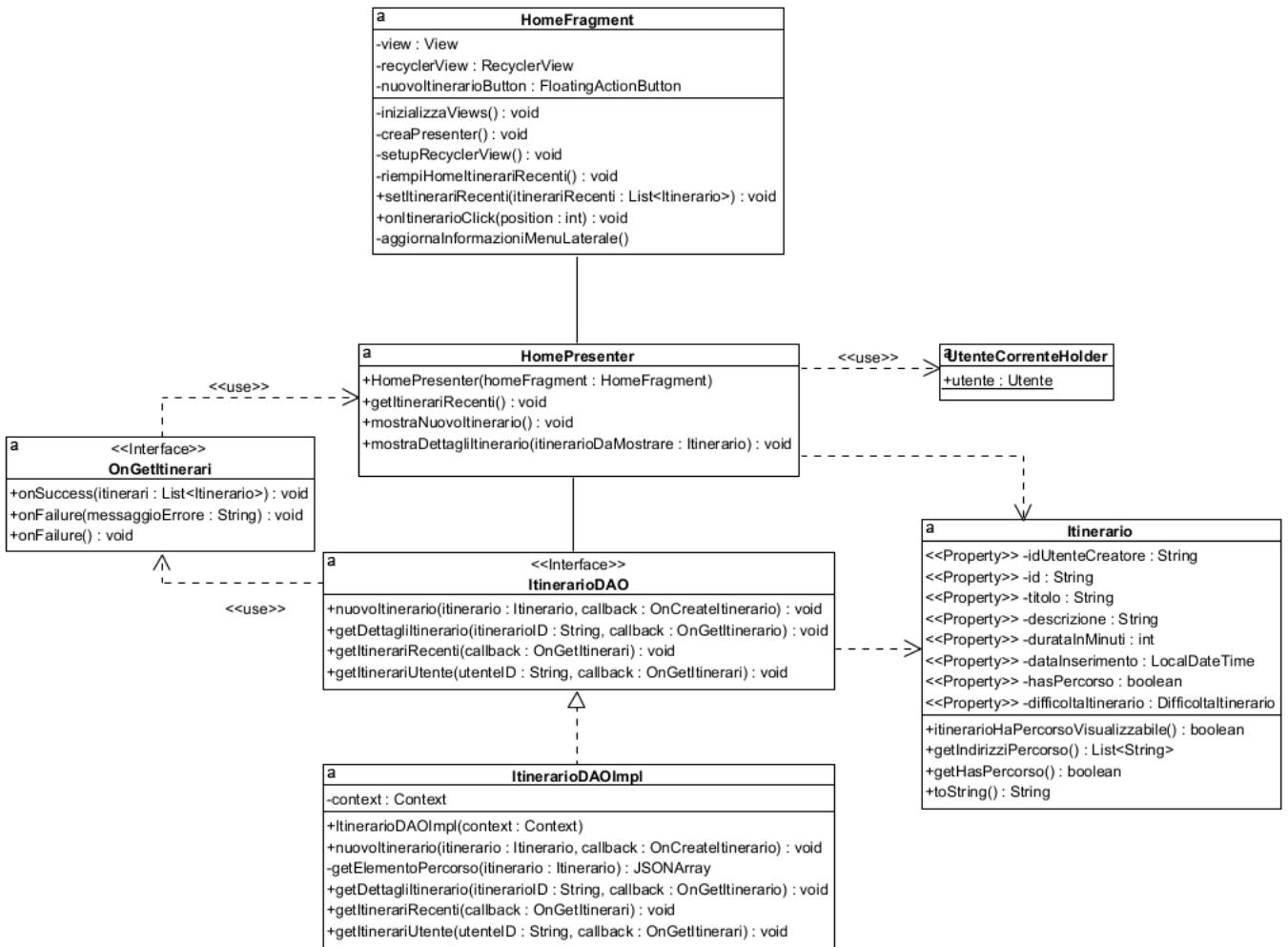
Crea Account



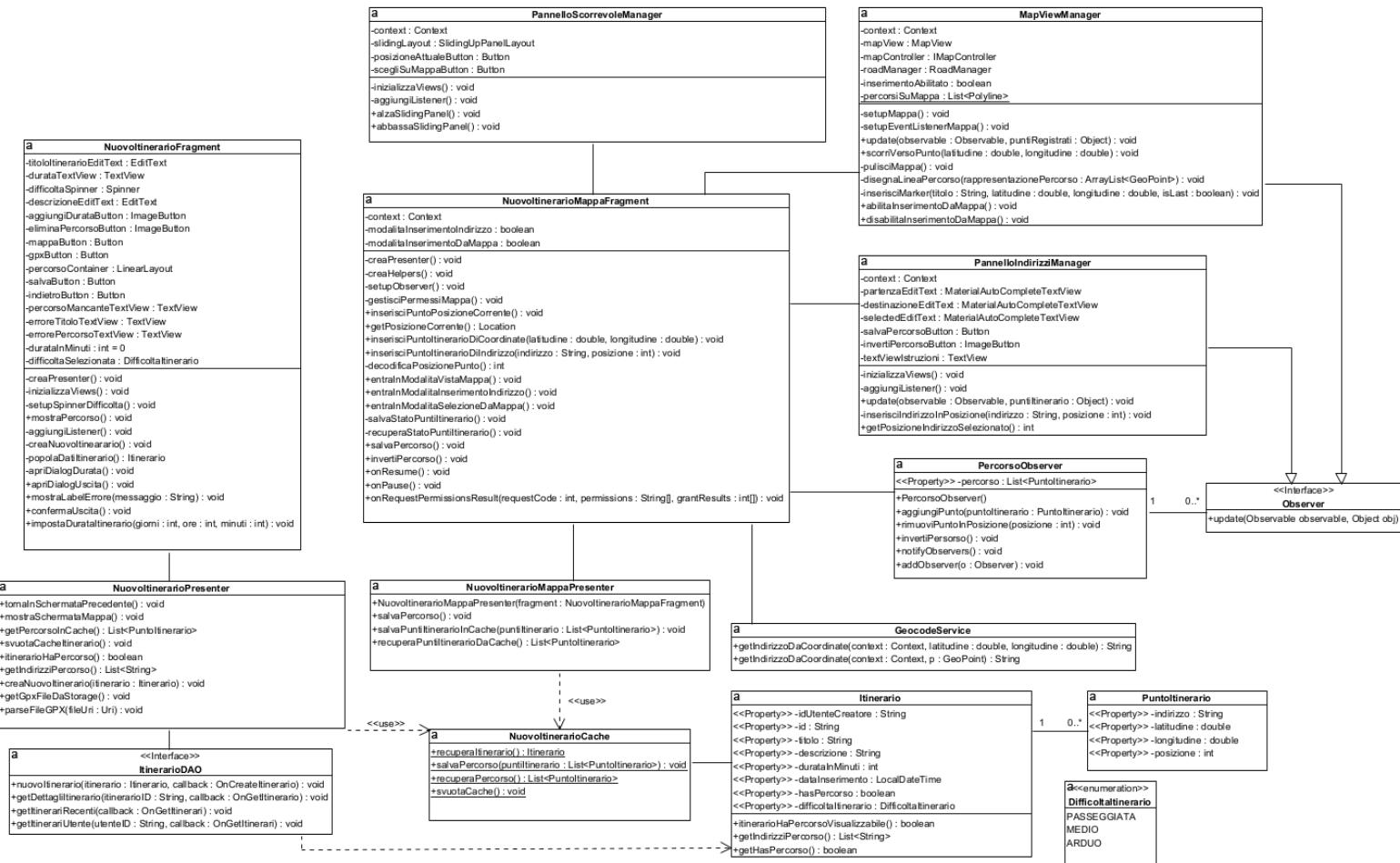
Recupero Account



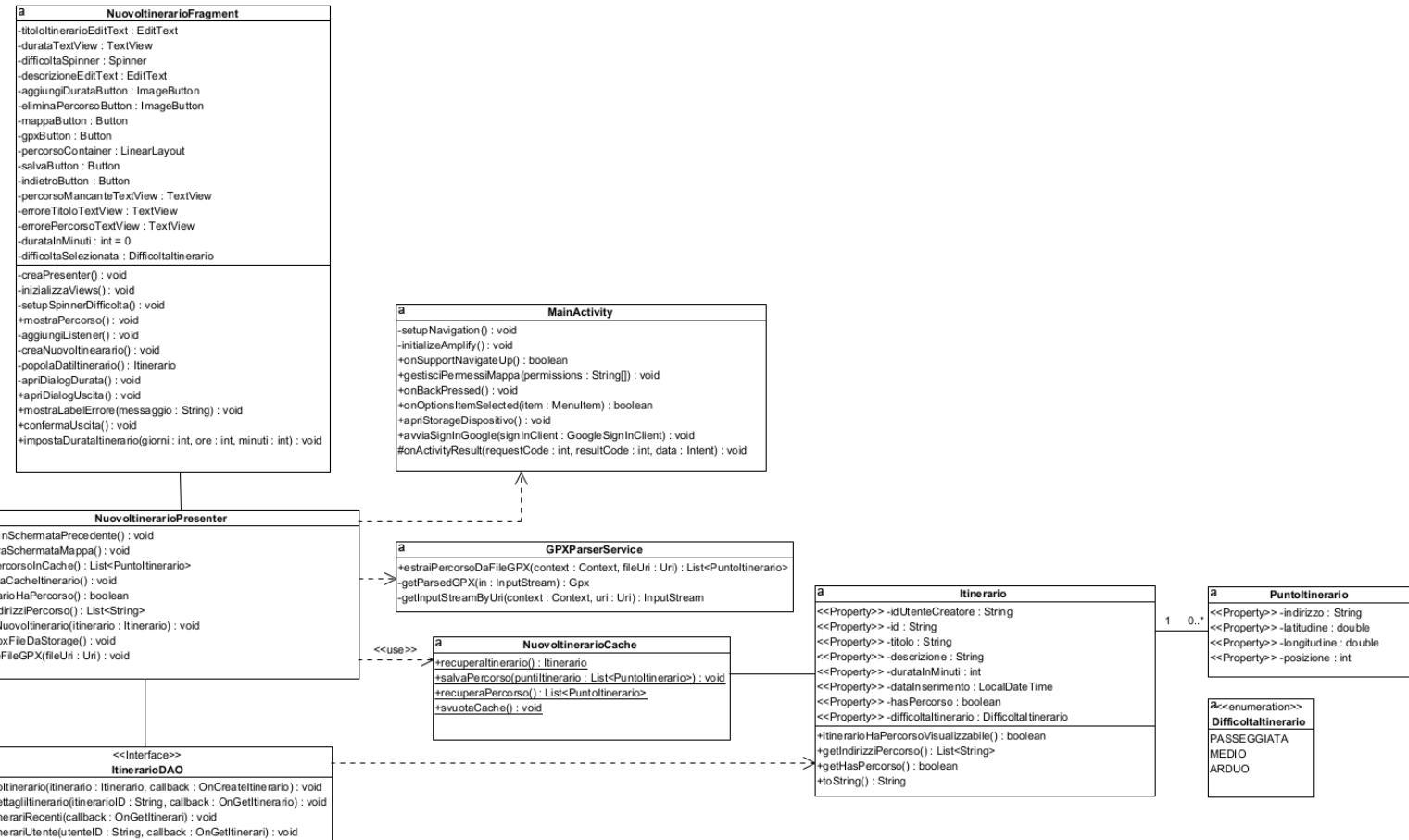
Visualizza Home



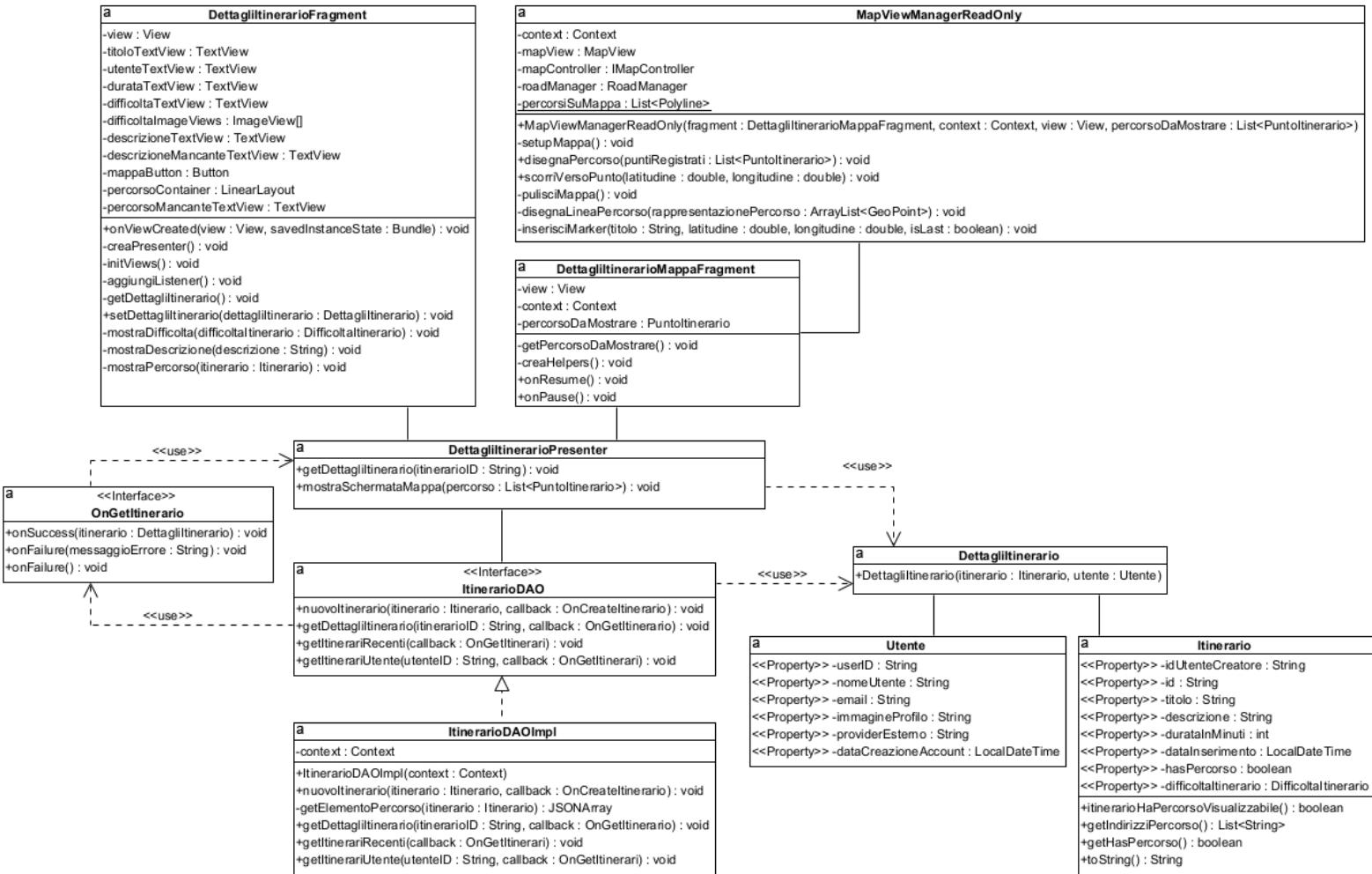
Inserisci Itinerario Mappa



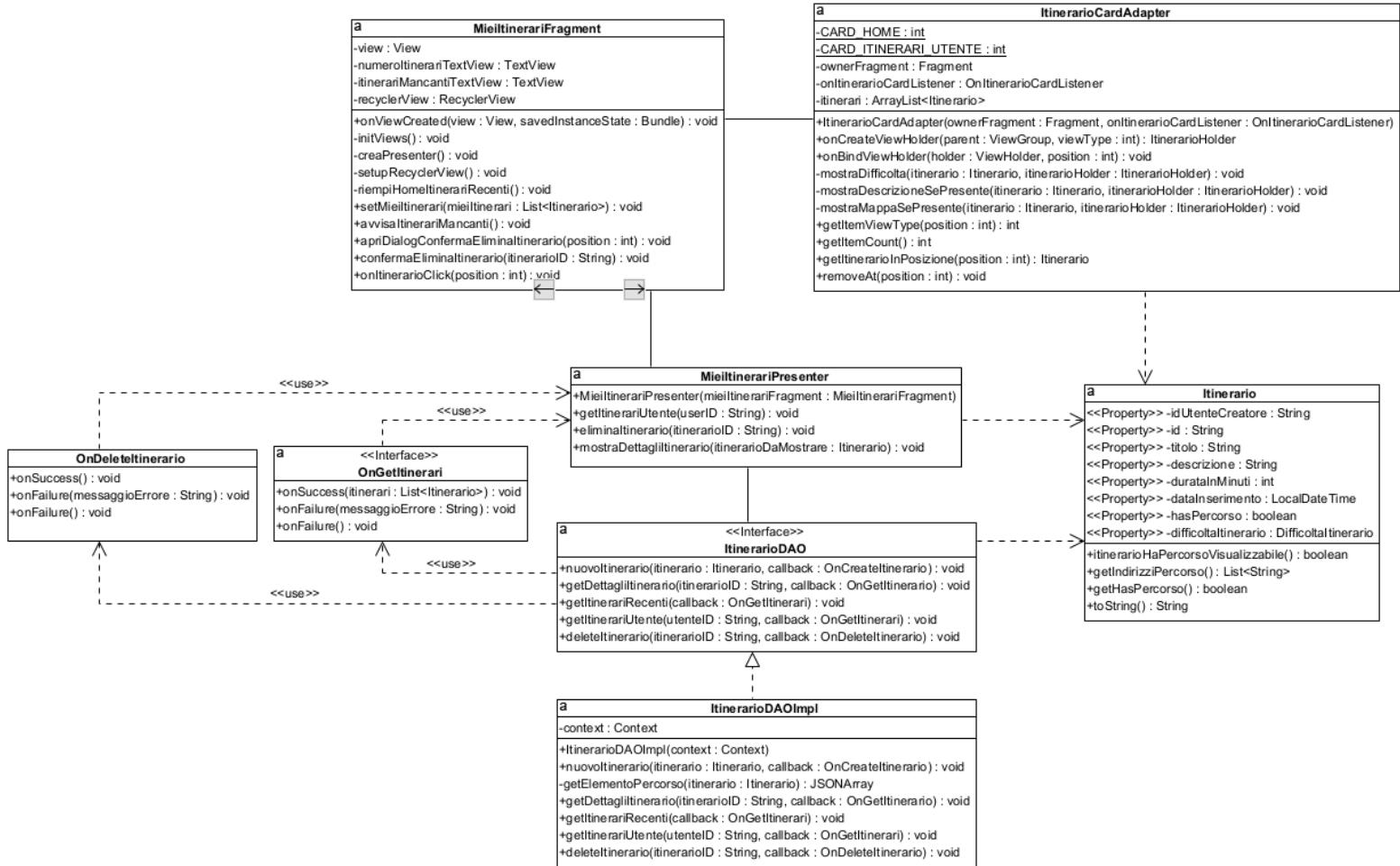
Inserisci Itinerario GPX



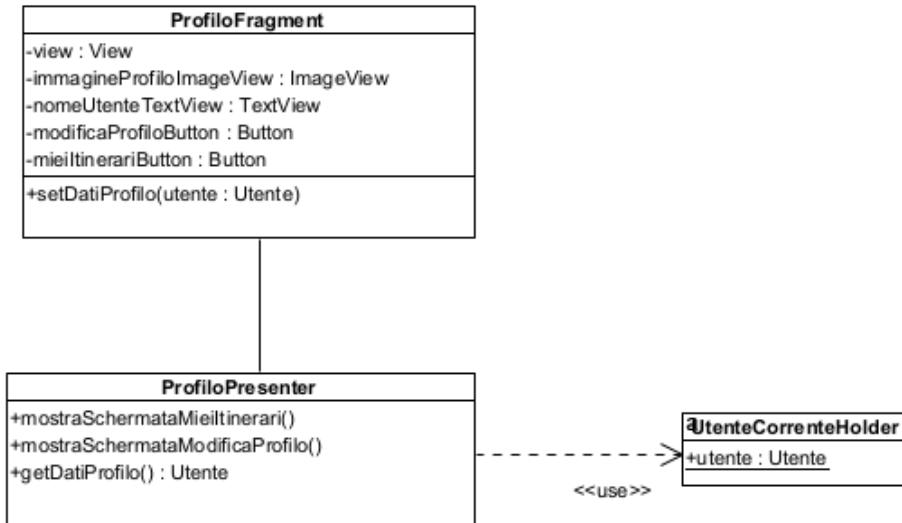
Visualizza Dettagli Itinerario



Visualizza Itinerari Propri

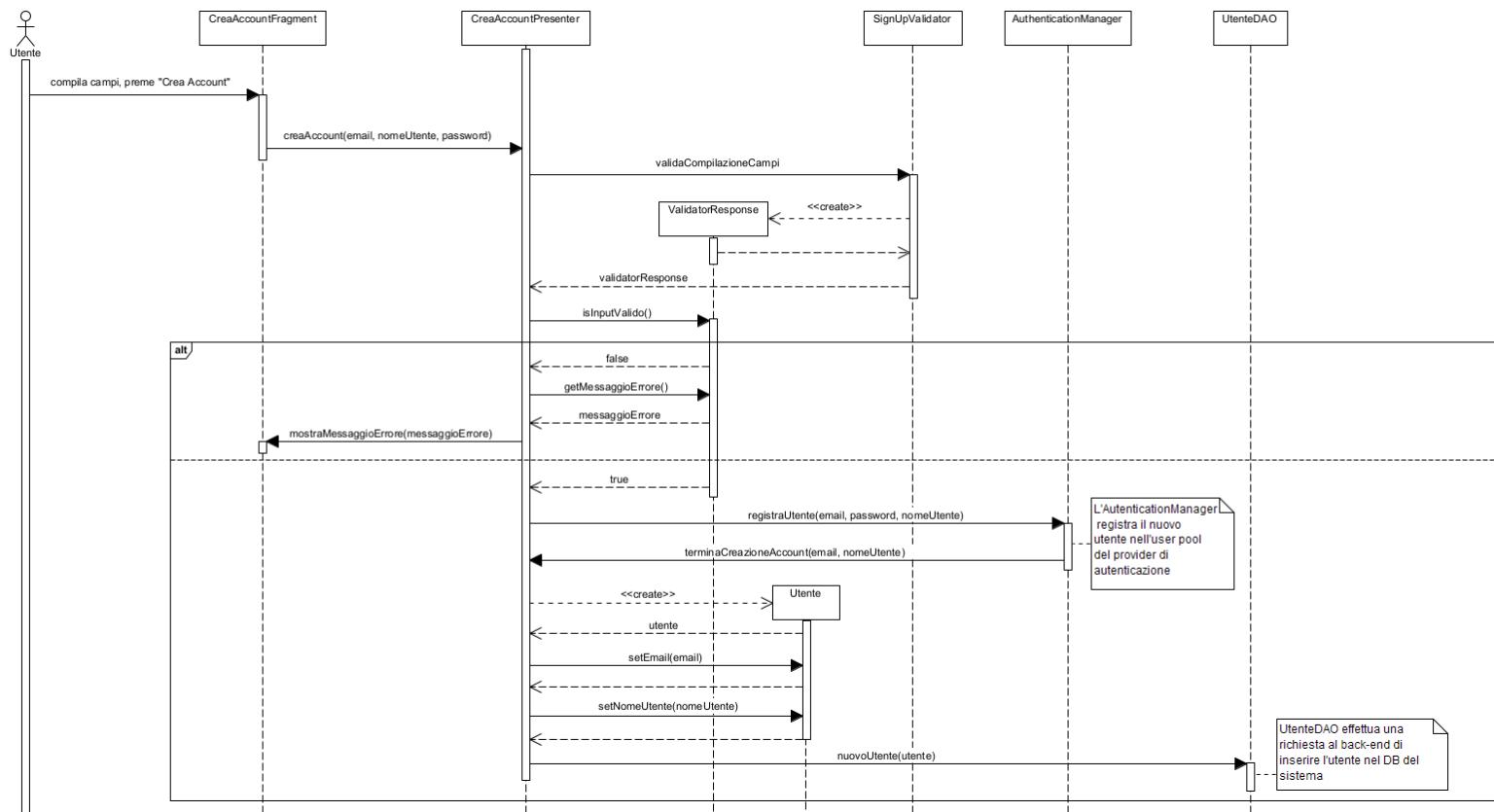


Visualizza Profilo

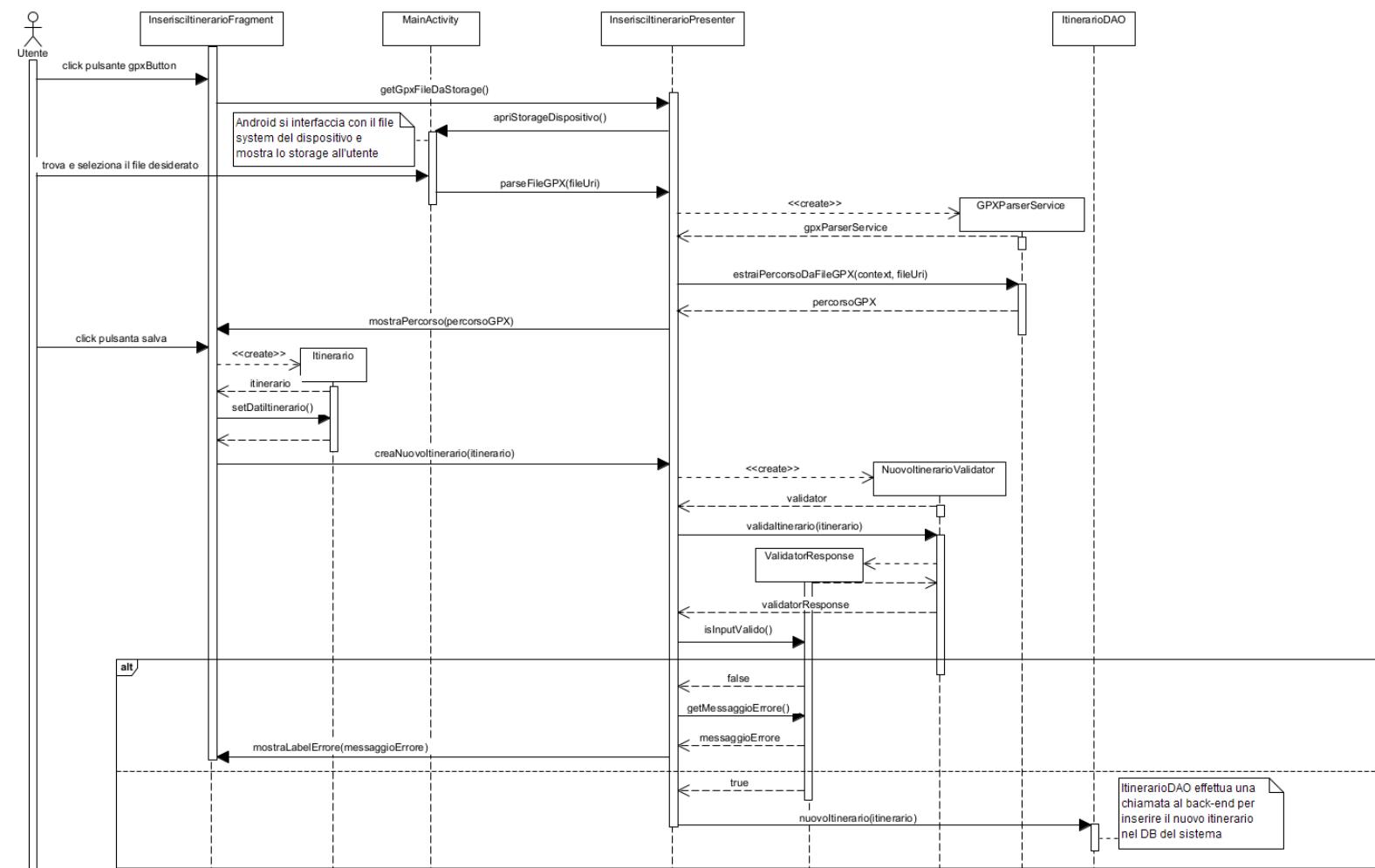


2.2.2 – Sequence diagram di design

Crea Account



Inserisci Itinerario GPX



III - Documento di Testing

3.1 - Unit testing di 3 metodi con JUnit

3.1.1 – Codice JUnit per il metodo validaCompilazioneCampi

Riferimento al metodo, applicativo mobile:

it.ingsw.natour21.control.validators.SignUpInputValidator#validaCompilazioneCampi

Il metodo **validaCompilazioneCampi** ha il compito di validare la compilazione corretta dei campi di testo durante il processo di registrazione dell'utente. Il metodo accetta come parametri le stringhe da validare, effettua i controlli necessari con il supporto della classe ValidationRules, e restituisce un oggetto di tipo ValidatorResponse avente i seguenti attributi:

- il booleano inputValido, valorizzato a true se i campi sono stati correttamente compilati, false altrimenti
- messaggioErrore: quando inputValido è false, descrive il motivo per cui la validazione è fallita

```
public ValidatorResponse validaCompilazioneCampi(String email, String password, String nomeUtente) {  
    boolean esitoValidazione = true;  
    String messaggioErrore = "";  
  
    if (ValidationRules.formatoNomeUtenteInvalido(nomeUtente)) {  
        esitoValidazione = false;  
        messaggioErrore = fragment.getString(R.string.avviso_formato_nome_utente_invalido);  
    }  
  
    if (ValidationRules.formatoPasswordInvalido(password)) {  
        esitoValidazione = false;  
        messaggioErrore = fragment.getString(R.string.avviso_formato_password_invalido);  
    }  
  
    if (ValidationRules.formatoMailInvalido(email)) {  
        esitoValidazione = false;  
        messaggioErrore = fragment.getString(R.string.avviso_formato_mail_invalido);  
    }  
  
    if (ValidationRules.campoVuoto(nomeUtente) ||  
        ValidationRules.campoVuoto(email) ||  
        ValidationRules.campoVuoto(password)) {  
  
        esitoValidazione = false;  
        messaggioErrore = "per favore compilare tutti i campi";  
    }  
  
    return new ValidatorResponse(esitoValidazione, messaggioErrore);  
}
```

Inoltre, viene di seguito mostrato il contenuto della **classe di supporto ValidationRules**. I metodi di questa classe vanno utilizzati dal metodo da testare e sono parte integrante della logica di validazione, ma sono stati estratti in metodi separati per motivi di leggibilità e pulizia del codice:

```
public class ValidationRules {

    private static final int LUNGHEZZA_MINIMA_PASSWORD = 8;
    private static final int LUNGHEZZA_MAXIMA_PASSWORD = 256;

    private static final int LUNGHEZZA_MINIMA_NOME_UTENTE = 4;
    private static final int LUNGHEZZA_MAXIMA_NOME_UTENTE = 25;

    public static boolean campoVuoto(String testo) {
        return testo == null || testo.trim().isEmpty();
    }

    public static boolean formatoMailInvalido(String mail) {
        if (mail != null)
            return mail != null && !android.util.Patterns.EMAIL_ADDRESS.matcher(mail).matches();
        else
            return true;
    }

    public static boolean formatoPasswordInvalido(String password) {
        return password == null ||
               password.trim().length() < LUNGHEZZA_MINIMA_PASSWORD ||
               password.trim().length() > LUNGHEZZA_MAXIMA_PASSWORD;
    }

    public static boolean formatoNomeUtenteInvalido(String nomeUtente) {
        return nomeUtente == null ||
               nomeUtente.trim().length() < LUNGHEZZA_MINIMA_NOME_UTENTE ||
               nomeUtente.trim().length() > LUNGHEZZA_MAXIMA_NOME_UTENTE;
    }
}
```

Strategia Black-Box:

Si effettua l'Input Space Partitioning, ovvero la suddivisione dei parametri del metodo in classi di equivalenza. Poichè le verifiche effettuate su ciascun parametro sono indipendenti da quelle fatte sui parametri rimanenti, si assume che per ogni parametro testato gli altri due siano validi:

Parametro	Valore	Risultato atteso
email	null	inputValido = false messaggioErrore = campo non compilato
email	stringa vuota: ""	inputValido = false messaggioErrore = campo non compilato
email	stringa contenente solo il carattere "@"	inputValido = false messaggioErrore = formato mail invalido
email	stringa contenente solo il carattere .	inputValido = false messaggioErrore = formato mail invalido
email	stringa contentente il carattere "@" preceduto da ":"	inputValido = false messaggioErrore = formato mail invalido
email	stringa contenente il carattere "@" seguito immediatamente da ":"	inputValido = false messaggioErrore = formato mail invalido
email	stringa contentete il carattere "@" seguito da ":" dopo più di un carattere	inputValido = true messaggioErrore = ""
password	null	inputValido = false messaggioErrore = campo non compilato
password	stringa vuota: ""	inputValido = false messaggioErrore = campo non compilato
password	stringa con lunghezza minore di quella consentita	inputValido = false messaggioErrore = formato password invalido
password	stringa con lunghezza maggiore di quella consentita	inputValido = false messaggioErrore = formato password invalido
password	stringa con lunghezza consentita	inputValido = true messaggioErrore = ""
nomeUtente	null	inputValido = false messaggioErrore = campo non compilato
nomeUtente	stringa vuota: ""	inputValido = false messaggioErrore = campo non compilato
nomeUtente	stringa con lunghezza minore di quella consentita	inputValido = false messaggioErrore = formato nome utente invalido
nomeUtente	stringa con lunghezza maggiore di quella consentita	inputValido = false messaggioErrore = formato nome utente invalido
nomeUtente	stringa contente uno spazio	inputValido = false messaggioErrore = formato nome utente invalido
nomeUtente	stringa che inizia con uno spazio	inputValido = false messaggioErrore = formato nome utente invalido
nomeUtente	stringa che termina con uno spazio	inputValido = false messaggioErrore = formato nome utente invalido
nomeUtente	stringa con lunghezza consentita	inputValido = true messaggioErrore = ""

Riferimento alla classe di test:

it.ingsw.natour21.control.ValidaCompilazioneCampiBlackBoxTest

```
public class ValidaCompilazioneCampiTest {

    private final static String messaggioErroreCompilazioneCampi = "campo non compilato";
    private final static String messaggioErroreFormatoMailInvalido = "formato email invalido";
    private final static String messaggioErroreFormatoPasswordInvalido = "formato password invalido";
    private final static String messaggioErroreFormatoNomeUtenteInvalido = "formato nome utente invalido";
    private final static String messaggioErroreSuccesso = ""; // Quando l'input è validato correttamente

    private SignUpInputValidator validator;

    @Before
    public void setupValidator() { validator = new SignUpInputValidator(); }

    @Test
    public void validaEmailNull() {
        String email = null;
        String password = "Password";
        String nomeUtente = "NomeUtente";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreCompilazioneCampi);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }

    @Test
    public void validaEmailVuota() {
        String email = "";
        String password = "Password";
        String nomeUtente = "NomeUtente";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreCompilazioneCampi);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }

    @Test
    public void validaEmailSoloChiocciola() {
        String email = "email@email";
        String password = "Password";
        String nomeUtente = "NomeUtente";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoMailInvalido);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }

    @Test
    public void validaEmailSoloPunto() {
        String email = "email.email";
        String password = "Password";
        String nomeUtente = "NomeUtente";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoMailInvalido);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }

    @Test
    public void validaEmailChiocciolaPrecedutaDaPunto() {
        String email = "email.email@com";
        String password = "Password";
        String nomeUtente = "NomeUtente";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoMailInvalido);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }
}
```

```

@Test
public void validaEmailChiocciolaImmediatamentePrimaDiPunto() {
    String email = "email@.com";
    String password = "Password";
    String nomeUtente = "NomeUtente";

    ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoMailInvalido);
    ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

    Assert.assertEquals(valoreAtteso, valoreAttuale);
}

@Test
public void validaEmailChiocciolaSeguitaDaPiùCaratteriEPunto() {
    String email = "email@email.com";
    String password = "Password";
    String nomeUtente = "NomeUtente";

    ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: true, messaggioErroreSuccesso);
    ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

    Assert.assertEquals(valoreAtteso, valoreAttuale);
}

@Test
public void validaPasswordNull() {
    String email = "email@email.com";
    String password = null;
    String nomeUtente = "NomeUtente";

    ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreCompilazioneCampi);
    ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

    Assert.assertEquals(valoreAtteso, valoreAttuale);
}
@Test
public void validaPasswordVuota() {
    String email = "email@email.com";
    String password = "";
    String nomeUtente = "NomeUtente";

    ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreCompilazioneCampi);
    ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

    Assert.assertEquals(valoreAtteso, valoreAttuale);
}

@Test
public void validaPasswordMinoreDiLunghezzaConsentita() {
    String email = "email@email.com";
    String password = "Pass";
    String nomeUtente = "NomeUtente";

    ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoPasswordInvalido);
    ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

    Assert.assertEquals(valoreAtteso, valoreAttuale);
}

@Test
public void validaPasswordMaggioreDiLunghezzaConsentita() {
    String email = "email@email.com";
    String password = new String(new char[300]).replace( oldChar: '\0', newChar: 'A'); // crea una stringa più lunga del limite di caratteri per la password
    String nomeUtente = "NomeUtente";

    ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoPasswordInvalido);
    ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

    Assert.assertEquals(valoreAtteso, valoreAttuale);
}

```

```

    @Test
    public void validaPasswordDiLunghezzaConsentita() {
        String email = "email@email.com";
        String password = "Password";
        String nomeUtente = "NomeUtente";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: true, messaggioErroreSuccesso);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }

    @Test
    public void validaNomeUtenteNull() {
        String email = "email@email.com";
        String password = "Password";
        String nomeUtente = null;

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreCompilazioneCampi);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }

    @Test
    public void validaNomeUtenteVuoto() {
        String email = "email@email.com";
        String password = "Password";
        String nomeUtente = "";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreCompilazioneCampi);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);
    }

    @Test
    public void validaNomeUtenteMinoreDiLunghezzaConsentita() {
        String email = "email@email.com";
        String password = "Password";
        String nomeUtente = "Nom";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoNomeUtenteInvalido);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }

    @Test
    public void validaNomeUtenteMaggioreDiLunghezzaConsentita() {
        String email = "email@email.com";
        String password = "Password";
        String nomeUtente = "NomeUtenteNomeUtenteNomeUtente";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoNomeUtenteInvalido);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }

    @Test
    public void validaNomeUtenteContenteSpazio() {
        String email = "email@email.com";
        String password = "Password";
        String nomeUtente = "Nome Utente";

        ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoNomeUtenteInvalido);
        ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

        Assert.assertEquals(valoreAtteso, valoreAttuale);
    }

```

```
@Test
public void validaNomeUtenteIniziaConSpazio() {
    String email = "email@email.com";
    String password = "Password";
    String nomeUtente = " NomeUtente";

    ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoNomeUtenteInvalido);
    ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

    Assert.assertEquals(valoreAtteso, valoreAttuale);
}

@Test
public void validaNomeUtenteFinisceConSpazio() {
    String email = "email@email.com";
    String password = "Password";
    String nomeUtente = "NomeUtente ";

    ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: false, messaggioErroreFormatoNomeUtenteInvalido);
    ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

    Assert.assertEquals(valoreAtteso, valoreAttuale);
}

@Test
public void validaNomeUtenteDiLunghezzaConsentita() {
    String email = "email@email.com";
    String password = "Password";
    String nomeUtente = "NomeUtente";

    ValidatorResponse valoreAtteso = new ValidatorResponse( inputValido: true, messaggioErroreSuccesso);
    ValidatorResponse valoreAttuale = validator.validaCompilazioneCampi(email, password, nomeUtente);

    Assert.assertEquals(valoreAtteso, valoreAttuale);
}
```

3.1.2 – Codice JUnit per il metodo generaDurataItinerario

Riferimento al metodo, applicativo mobile:

it.ingsw.natour21.control.Utils#generaDurataItinerario

Il metodo generaDurataItinerario ha il compito di generare e restituire una stringa che rappresenti la durata temporale di un itinerario tramite il formato ‘Xg Yh Zm’ (giorni, ore, minuti). Il metodo accetta come parametri la durata di un itinerario espressa in minuti.

```
51 @  
52     public static String generaDurataItinerario(int durataInMinuti) {  
53         if (durataInMinuti > 0 && durataInMinuti < MAX_DURATA_ITINERARIO) {  
54             int giorni = durataInMinuti / 1440;  
55             int ore = (durataInMinuti % 1440) / 60;  
56             int minuti = (durataInMinuti % 1440) % 60;  
57  
58             String giorniString = (giorni != 0) ? giorni + "g " : "";  
59             String oreString = ((ore != 0) || (giorni != 0 && minuti != 0)) ? ore + "h " : "";  
60             String minutiString = (minuti != 0) ? minuti + "m" : "";  
61  
62             return giorniString + oreString + minutiString;  
63         } else {  
64             return " - ";  
65         }  
}
```

Strategia Black-Box:

Si effettua l'Input Space Partitioning, ovvero la suddivisione dei valori possibili assunti dal parametro in classi di equivalenza:

- $\text{durataInMinuti} \leq 0$ - il metodo restituisce la stringa “ - ”
- $0 < \text{durataInMinuti} \leq \text{MAX_DURATA_MINUTI}$ - il metodo genera e restituisce la stringa rappresentante la durata in giorni, ore e minuti
- $\text{durataInMinuti} > \text{MAX_DURATA_MINUTI}$ - il metodo restituisce la stringa “ - ”

La stringa rappresentante la durata viene generata in maniera tale da omettere i giorni/ore/minuti quando questi sono uguali a 0. Unica eccezione di questa regola sono le ore, che possono essere omesse anche nel caso in cui i giorni e i minuti sono entrambi diversi da 0. Questo controllo viene effettuato dalle righe 57-59 in immagine.

Il valore MAX_DURATA_MINUTI rappresenta la massima durata che un itinerario può avere, ed è al momento impostata al valore 14400, che equivale a 10 giorni

Parametro	Valore assunto	Risultato atteso
durataInMinuti	-100	-
durataInMinuti	0	-
durataInMinuti	59	59m
durataInMinuti	60	60m
durataInMinuti	90	1h 30m
durataInMinuti	120	2h
durataInMinuti	1439	23h 59m
durataInMinuti	1440	1g
durataInMinuti	2160	1h 12h
durataInMinuti	2880	2g
durataInMinuti	2900	2g 20m
durataInMinuti	14399	9g 23h 59m
durataInMinuti	14400	10g
durataInMinuti	14401	-

Riferimento alla classe Test:

it.ingsw.natour21.control.GeneraDurataItinerarioBlackBoxTest

```
6  class GeneraDurataItinerarioBlackBoxTest {
7
8      @Test
9      public void generaDurataNegativa() {
10         int durataInMinuti = -100;
11         String valoreAtteso = " - ";
12         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
13
14         assertEquals(valoreAtteso, valoreAttuale);
15     }
16
17     @Test
18     public void generaDurataNulla() {
19         int durataInMinuti = 0;
20         String valoreAtteso = " - ";
21         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
22
23         assertEquals(valoreAtteso, valoreAttuale);
24     }
25
26     @Test
27     public void generaDurataUnMinutoMenoDiUnOra() {
28         int durataInMinuti = 59;
29         String valoreAtteso = "59m";
30         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
31
32         assertEquals(valoreAtteso, valoreAttuale);
33     }
34
35     @Test
36     public void generaDurataUnOraEsatta() {
37         int durataInMinuti = 60;
38         String valoreAtteso = "1h ";
39         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
40
41         assertEquals(valoreAtteso, valoreAttuale);
42     }
43
44     @Test
45     public void generaDurataUnOraEMezza() {
46         int durataInMinuti = 90;
47         String valoreAtteso = "1h 30m";
48         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
49
50         assertEquals(valoreAtteso, valoreAttuale);
51     }
52
53     @Test
54     public void generaDurataDueOre() {
55         int durataInMinuti = 120;
56         String valoreAtteso = "2h ";
57         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
58
59         assertEquals(valoreAtteso, valoreAttuale);
60     }
61
62     @Test
63     public void generaDurataUnMinutoMenoDiUnGiorno() {
64         int durataInMinuti = 1439;
65         String valoreAtteso = "23h 59m";
66         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
67
68         assertEquals(valoreAtteso, valoreAttuale);
69     }
```

```
71     @Test
72     public void generaDurataUnGiorno() {
73         int durataInMinuti = 1440;
74         String valoreAtteso = "1g ";
75         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
76
77         assertEquals(valoreAtteso, valoreAttuale);
78     }
79
80     @Test
81     public void generaDurataUnGiornoEMezzo() {
82         int durataInMinuti = 2160;
83         String valoreAtteso = "1g 12h ";
84         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
85
86         assertEquals(valoreAtteso, valoreAttuale);
87     }
88
89     @Test
90     public void generaDurataDueGiorni() {
91         int durataInMinuti = 2880;
92         String valoreAtteso = "2g ";
93         String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
94
95         assertEquals(valoreAtteso, valoreAttuale);
96     }
97
98     @Test
99     public void generaDurataGiorniEMinutiDiversiDaZero() {
100        int durataInMinuti = 2900;
101        String valoreAtteso = "2g 20m";
102        String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
103
104        assertEquals(valoreAtteso, valoreAttuale);
105    }
106
107    @Test
108    public void generaDurataUnMinutoMenoDiDurataMassima() {
109        int durataInMinuti = 14399;
110        String valoreAtteso = "9g 23h 59m";
111        String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
112
113        assertEquals(valoreAtteso, valoreAttuale);
114    }
115
116    @Test
117    public void generaDurataMassima() {
118        int durataInMinuti = 14400;
119        String valoreAtteso = "10g ";
120        String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
121
122        assertEquals(valoreAtteso, valoreAttuale);
123    }
124
125    @Test
126    public void generaDurataMaggioreDiDurataMassima() {
127        int durataInMinuti = 14401;
128        String valoreAtteso = " - ";
129        String valoreAttuale = Utils.generaDurataItinerario(durataInMinuti);
130
131        assertEquals(valoreAtteso, valoreAttuale);
132    }
133
134 }
```

3.1.3 – Codice JUnit per il metodo generaIntervalloTemporaleInserimento

Riferimento al metodo, applicativo mobile:

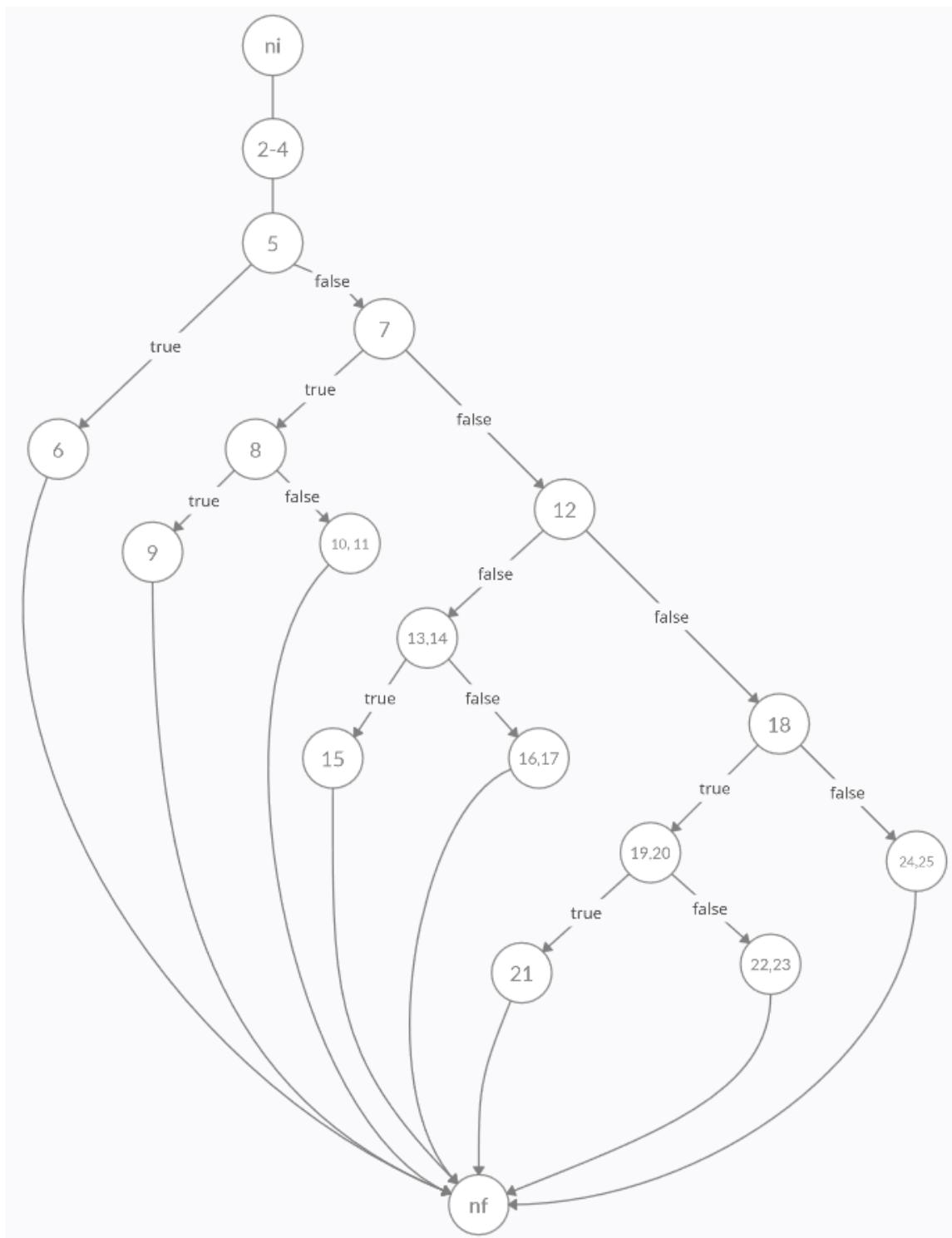
it.ingsw.natour21.control.Utils#generaIntervalloTemporaleInserimento

Il metodo generaIntervalloTemporaleInserimento ha il compito di generare una stringa che indica quanti giorni/settimane/mesi fa è stato inserito un itinerario in riferimento alla data odierna. Accetta come parametro il DateTime di inserimento dell'itinerario.

```
1  @ public static String generaIntervalloTemporaleInserimento(LocalDateTime dateTimeInserimento) {  
2      LocalDateTime now = LocalDateTime.now();  
3      long giorni = Duration.between(dateTimeInserimento, now).toDays();  
4  
5      if (giorni == 0) {  
6          return "oggi";  
7      } else if (giorni > 0 && giorni < 7) {  
8          if (giorni == 1)  
9              return giorni + " giorno fa";  
10         else  
11             return giorni + " giorni fa";  
12     } else if (giorni >= 7 && giorni < 30) {  
13         long settimane = giorni / 7;  
14         if (settimane == 1)  
15             return settimane + " settimana fa";  
16         else  
17             return settimane + " settimane fa";  
18     } else if (giorni >= 30) {  
19         long mesi = giorni / 30;  
20         if (mesi == 1)  
21             return mesi + " mese fa";  
22         else  
23             return mesi + " mesi fa";  
24     } else  
25         return "";  
26 }
```

Strategia strutturale (White-Box):

Data la struttura complessa del metodo, composto da diversi conditional statements anche innestati tra di loro, è stato deciso di utilizzare una strategia di testing strutturale con copertura branch-coverage. Facendo riferimento al seguente grafo del flusso di controllo (GFC) sono stati scritti i seguenti test case:



Riferimento alla classe Test:

it.ingsw.natour21.control.GeneralIntervalloTemporaleInserimentoBlackBoxTest

```
8  public class GeneraIntervalloTemporaleInserimentoTest {  
9  
10     @Test  
11    public void testGeneraIntervallo5_6() {  
12        LocalDateTime dataInserimento = LocalDateTime.now();  
13        String valoreAtteso = "oggi";  
14  
15        String valoreAttuale = Utils.generaIntervalloTemporaleInserimento(dataInserimento);  
16  
17        assertEquals(valoreAtteso, valoreAttuale);  
18    }  
19  
20    @Test  
21    public void testGeneraIntervallo5_7_8_9() {  
22        LocalDateTime dataInserimento = LocalDateTime.now().minusDays(1);  
23        String valoreAtteso = "1 giorno fa";  
24  
25        String valoreAttuale = Utils.generaIntervalloTemporaleInserimento(dataInserimento);  
26  
27        assertEquals(valoreAtteso, valoreAttuale);  
28    }  
29  
30    @Test  
31    public void testGeneraIntervallo5_7_8_10_11() {  
32        LocalDateTime dataInserimento = LocalDateTime.now().minusDays(2);  
33        String valoreAtteso = "2 giorni fa";  
34  
35        String valoreAttuale = Utils.generaIntervalloTemporaleInserimento(dataInserimento);  
36  
37        assertEquals(valoreAtteso, valoreAttuale);  
38    }  
39  
40    @Test  
41    public void testGeneraIntervallo5_7_12_13_14_15() {  
42        LocalDateTime dataInserimento = LocalDateTime.now().minusWeeks(1);  
43        String valoreAtteso = "1 settimana fa";  
44  
45        String valoreAttuale = Utils.generaIntervalloTemporaleInserimento(dataInserimento);  
46  
47        assertEquals(valoreAtteso, valoreAttuale);  
48    }  
49  
50    @Test  
51    public void testGeneraIntervallo5_7_12_13_14_16_17() {  
52        LocalDateTime dataInserimento = LocalDateTime.now().minusWeeks(3);  
53        String valoreAtteso = "3 settimane fa";  
54  
55        String valoreAttuale = Utils.generaIntervalloTemporaleInserimento(dataInserimento);  
56  
57        assertEquals(valoreAtteso, valoreAttuale);  
58    }
```

```
60  @Test
61  public void testGeneraIntervallo5_7_12_18_19_20_21() {
62      LocalDateTime dataInserimento = LocalDateTime.now().minusMonths(1);
63      String valoreAtteso = "1 mese fa";
64
65      String valoreAttuale = Utils.generaIntervalloTemporaleInserimento(dataInserimento);
66
67      assertEquals(valoreAtteso, valoreAttuale);
68  }
69
70  @Test
71  public void testGeneraIntervallo5_7_12_18_19_20_22_23() {
72      LocalDateTime dataInserimento = LocalDateTime.now().minusMonths(4);
73      String valoreAtteso = "4 mesi fa";
74
75      String valoreAttuale = Utils.generaIntervalloTemporaleInserimento(dataInserimento);
76
77      assertEquals(valoreAtteso, valoreAttuale);
78  }
79
80  @Test
81  public void testGeneraIntervallo5_7_12_18_24_25() {
82      LocalDateTime dataInserimento = LocalDateTime.now().plusDays(1);
83      String valoreAtteso = "";
84
85      String valoreAttuale = Utils.generaIntervalloTemporaleInserimento(dataInserimento);
86
87      assertEquals(valoreAtteso, valoreAttuale);
88  }
89
90 }
```

3.2 – Valutazione dell’usabilità sul campo

Per l’usabilità dell’applicativo sul campo è stata effettuata una sessione di test sugli stessi utenti che hanno partecipato nella valutazione dell’usabilità a priori. Questa volta gli utenti hanno interagito con la versione finale dell’applicativo, ed è stato pensato un nuovo elenco di task da svolgere:

- **Task #1:** effettuare la registrazione e accedere con il proprio account
- **Task #2:** visualizzare i dettagli di un itinerario dalla schermata home
- **Task #3:** inserire un nuovo itinerario utilizzando la mappa
- **Task #4:** visualizzare uno degli itinerari che hai inserito precedentemente

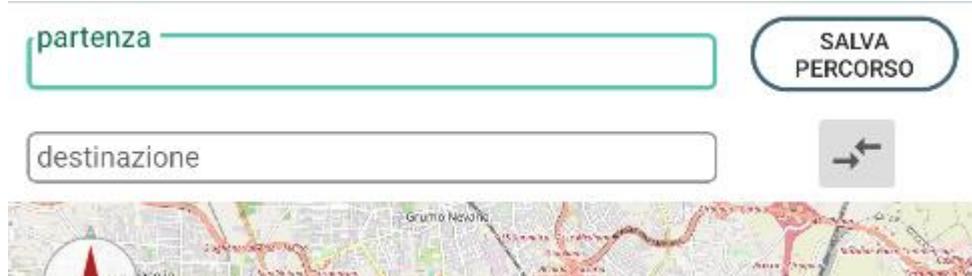
Di seguito sono mostrati i risultati dei test eseguiti sui tre utenti:

	Task #1	Task #2	Task #3	Task #4
User #1	✓	✓	✓	✓
User #2	✓	✓	✗	✓
User #3	✗	✓	✓	✓

Tasso di successo = 10 successi /12 test totali = 83%

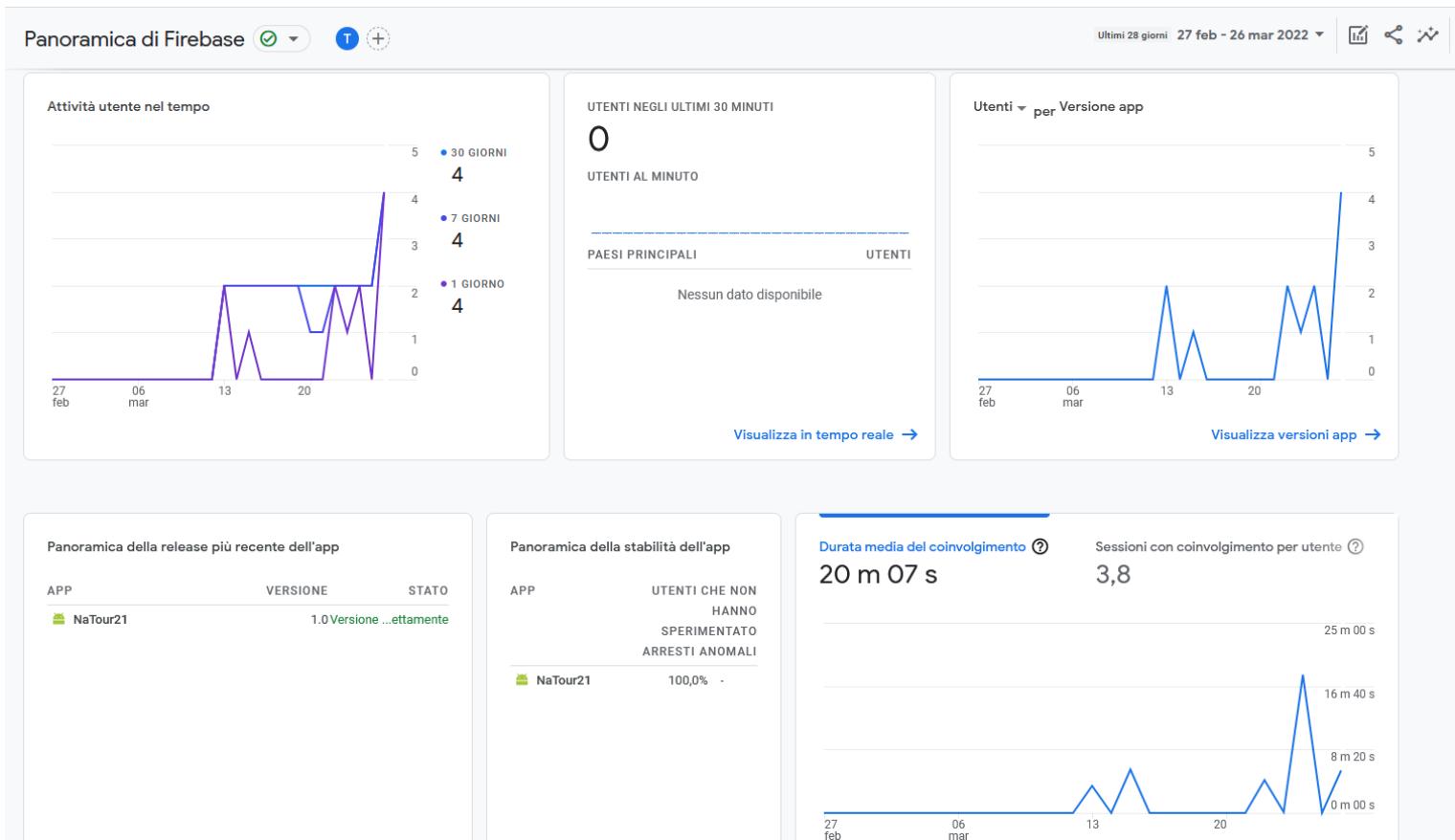
Dall’interazione con l’applicativo da parte degli utenti sono risultati evidenti alcuni problemi sia tecnici che di design:

- uno degli utenti ha riscontrato un problema durante il processo di autenticazione, dove non era capace a confermare la propria mail per poter poi effettuare l’accesso a causa di un bug del sistema.
- un altro utente invece ha avuto difficoltà con l’utilizzo dell’interfaccia di inserimento del percorso geografico di un itinerario **mp_08_nuovo_itinerario_mappa**. In dettaglio, l’interfaccia non offre un’indicazione grafica su quale indirizzo si sta inserendo, la partenza o la destinazione. Di conseguenza, l’interfaccia è stata aggiornata appositamente per evidenziare l’indirizzo che l’utente seleziona:



La valutazione sul campo è stata anche una buona opportunità per mettere alla prova i vari strumenti di logging ed analytics che sono stati integrati nell'applicativo. Il sistema ha più modi di generare file di log e registrare gli eventi innescati dall'utente:

- log nella console di sviluppo, quello meno utile in quanto si ha la visibilità solo in ambiente di sviluppo
- file di log generati sul dispositivo di ogni utente
- Firebase Analytics: una piattaforma di analisi sviluppata da Google che mette a disposizione numerosi strumenti tra i quali:
 - una **dashboard** di riepilogo che offre numerose informazioni sugli utenti che hanno interagito con l'applicativo, come il loro numero, la loro regione geografica e il tempo di coinvolgimento medio



- **Events:** un sistema che permette di registrare le azioni dell'utente, così come eventi di sistema ed errori

Titolo pagina e classe schermata ▾	+	↓Visualizzazioni	Utenti	Nuovi utenti	Visualizzazioni per utente	Durata media del coinvolgimento	Scorrimenti utenti unici	Conteggio eventi		Conversioni Tutti gli eventi	Entrate totali
								Tutti gli eventi	100% del totale		
Totali		567 100% del totale	4 100% del totale	4 100% del totale	141,75 Uguale alla media	20 m 07 s Uguale alla media	0	1.114 100% del totale	4,00 100% del totale		0,00 \$
1 HomeFragment		183	4	0	45,75	3 m 23 s	0	320	0,00		0,00 \$
2 MainActivity		138	4	0	34,50	3 m 18 s	0	302	0,00		0,00 \$
3 NuovoltinerarioFragment		63	4	0	15,75	3 m 39 s	0	146	0,00		0,00 \$
4 DettagliitinerarioFragment		60	4	0	15,00	1 m 41 s	0	90	0,00		0,00 \$
5 EffettuaAccessoFragment		38	4	0	9,50	3 m 15 s	0	81	0,00		0,00 \$
6 MieitinerariFragment		28	4	0	7,00	1 m 42 s	0	63	0,00		0,00 \$
7 NuovoltinerarioMappaFragment		24	4	0	6,00	1 m 47 s	0	25	0,00		0,00 \$
8 SignInHubActivity		16	4	0	4,00	0 m 00 s	0	16	0,00		0,00 \$
9 CreaAccountFragment		13	4	0	3,25	1 m 19 s	0	42	0,00		0,00 \$
10 ProfiloFragment		4	1	0	4,00	0 m 05 s	0	8	0,00		0,00 \$

Eventi esistenti					
Nome dell'evento ↑	Conteggio	Variazione %	Utenti	Variazione %	
first_open	4	-	4	-	
inserimento_itinerario	3	-	3	-	
login	92	-	4	-	
rimozione_itinerario	3	-	3	-	
screen_view	567	-	4	-	
select_content	336	-	4	-	
session_start	17	-	4	-	
sign_up	7	-	3	-	
visualizza_dettagli_itinerario	22	-	3	-	
visualizza_itinerari_propri	16	-	3	-	
visualizza_profilo	4	-	1	-	

- **DebugView:** una schermata che permette di monitorare gli eventi di uno specifico dispositivo in tempo reale, quando l'applicativo viene avviato nella modalità di debug di Analytics

