



A.A. 2021-2022



Progetto NaTour

Università degli studi di Napoli

Federico II



Valentino Bocchetti - N86003405



Mario Gabriele Carofano - N86003228

Indice

1 Revisioni	1
2 Documento dei Requisiti Software	5
2.1 Modello funzionale	5
2.1.1 Modellazione dei Casi d'Uso	5
2.1.2 Tabelle di Cockburn (per 2 casi significativi)	8
2.1.3 Prototipazione visuale via Mock-up (per 2 casi significativi)	12
2.1.4 Presentazione dell'idea progettuale	25
2.1.5 Requisiti funzionali	27
2.1.6 Requisiti non funzionali	29
2.1.7 Individuazione del target degli utenti	30
2.1.8 Valutazione dell'usabilità a priori	32
2.1.9 Prototipazione funzionale via Statechart dell'interfaccia grafica	37
2.1.10 Glossario	39
2.2 Modelli di Dominio	41
2.2.1 Classi, oggetti e relazioni di analisi	41
2.2.2 Diagrammi di sequenza di analisi (per 2 casi significativi)	49
2.2.3 Diagrammi di attività	52
3 Documento di Design del Sistema	66
3.1 Analisi dell'architettura con esplicita definizione dei criteri di design	66
3.1.1 Tecnologie utilizzate	66
3.1.2 Linguaggi e strumenti utilizzati	66

3.1.3	Autenticazione	67
3.1.4	Gestione del salvataggio delle immagini di utenti, admin e itinerari	67
3.1.5	Dati degli utenti e degli admin	67
3.1.6	Gestione dei dati degli utenti e degli admin loggati	68
3.1.7	Gestione dei servizi per utenti e admin	69
3.2	Design pattern utilizzati	70
3.2.1	Backend	70
3.2.2	Frontend	71
3.3	Come abbiamo aumentato la visibilità del progetto	73
3.4	Diagramma delle Classi di Design	74
3.5	Diagrammi di Sequenza di Design per 2 casi d'uso significativi	82
3.6	Definizione delle gerarchie funzionali	85
4	Codice sorgente sviluppato	87
5	Definizione di un piano di testing	88
6	Valutazione sul campo dell'usabilità	88
6.1	Codice jUnit per Unit testing	88
6.1.1	Prima esperienza con il framework JUnit	88
6.1.2	Test dell'applicativo	89
6.2	Valutazione sull'usabilità sul campo, realizzata sul prodotto finito	99
7	Analisi dei tempi di sviluppo	107
8	Contributori	109

1 Revisioni

Versione	Data	Autore	Descrizione
0.1	01-11-2021	Valentino Bocchetti Mario Gabriele Carofano	Creazione della struttura del documento e strutturazione dell'albero di lavoro
0.2	11-11-2021	Valentino Bocchetti Mario Gabriele Carofano	Completamento intestazione e struttura del documento
0.3	25-11-2021	Valentino Bocchetti Mario Gabriele Carofano	Inizio stesura della modellazione dei Casi d'Uso
0.3.1	27-11-2021	Valentino Bocchetti Mario Gabriele Carofano	Completamento stesura della modellazione dei Casi d'Uso. Inizio stesura delle Tabelle di Cockburn
0.3.2	28-11-2021	Valentino Bocchetti Mario Gabriele Carofano	Completamento stesura delle Tabelle di Cockburn Creazione dei mock-up per l'interfaccia Utente
0.4	02-12-2021	Valentino Bocchetti Mario Gabriele Carofano	Inizio stesura della presentazione
0.5	07-12-2021	Valentino Bocchetti Mario Gabriele Carofano	Modifica Documentazione (Aggiunte informazioni sui contributori e modifica bibliografia)
0.5.1	07-12-2021	Valentino Bocchetti	Modifica Documentazione (Aggiunte Tabelle di Cockburn)
0.5.2	12-12-2021	Valentino Bocchetti Mario Gabriele Carofano	Descrizione di Requisiti funzionali e non funzionali. Identificazione del target degli utenti. Valutazione dell'usabilità a priori
0.5.3	17-01-2022	Valentino Bocchetti	Creazione progetto e primi test
0.5.4	18-01-2022	Valentino Bocchetti	Refactoring e modifiche minori del progetto (Aggiunta librerie per AWS e OSM)
0.5.5	24-01-2022	Valentino Bocchetti	Refactoring e creazione del Server per la gestione delle API (Inizializzazione del progetto con SpringBoot)
0.5.6	25-01-2022	Valentino Bocchetti	Primi test con Spring Boot e PostgreSQL
0.5.7	26-01-2022	Valentino Bocchetti	Mappatura parametri (Server) e refactoring
0.5.8	27-01-2022	Valentino Bocchetti	Refactoring + Aggiunta Best Practices per la Dependency Injection (Server)
0.5.9	28-01-2022	Valentino Bocchetti	Introduzione delle classi DTO
0.6.0	31-01-2022	Valentino Bocchetti Mario Gabriele Carofano	Migliorata gestione delle Request (Fix delle Request PUT) Test Applicativo (Backend/Frontend - Spring Boot + Retrofit)
0.6.1	01-02-2022	Valentino Bocchetti	Introduzione della Chat lato Server + Refactoring
0.6.2	02-02-2022	Valentino Bocchetti	Introduzione filtri per la ricerca degli itinerari
0.6.3	04-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Modifiche campi itinerario e sua gestione (Server) Modifiche minori nella documentazione
0.6.4	05-02-2022	Valentino Bocchetti	Aggiunta gestione delle foto per gli itinerari (lato Server)

Versione	Data	Autore	Descrizione
0.6.5	07-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Primi test collegamento Android con Spring Boot. Refactoring Modifica schermata Login (material design)
0.6.6	08-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Migliorie sulla grafica (in accordo con il material design) refactoring
0.6.7	08-02-2022	Valentino Bocchetti	Introduzione di Rx-Java per le request secondo il pattern Observable
0.6.8	09-02-2022	Valentino Bocchetti	Completamento Mappatura Backend/Frontend (Request) Edit Minori
0.6.9	11-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Completata transizione da Amplify ad RxJava-Amplify Testing con Retrofit/SpringBoot sugli itinerari (testing con le CardLayout)
0.7.0	13-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Implementazione del design pattern MVP. Refactoring e modifiche minori
0.7.1	14-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Refactoring e modifiche minori
0.7.2	15-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Gestione delle request e loro implementazione nell'applicativo Bug fixes e refactoring Testing dell'applicativo
0.7.2	15-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Aggiunta istanze per tipo (Retrofit) Creazione costanti e testing dell'applicativo Bug fixes e refactoring
0.7.3	17-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Inizializzazione e implementazione della gestione dell'utente loggato (Applicativo android) e relativo testing Completamento gestione della grafica per gli Itinerari e Mappa
0.7.4	18-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Migliorata la gestione dei dati in locale Implementazione gestione immagini per l'applicativo mobile Aggiunta costanti Bug Fixes
0.7.5	20-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Aggiornamenti e completamento schermata chat singola
0.7.6	24-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Implementazione dell'inserimento interattivo di un nuovo itinerario mediante mappa interattiva Refactoring Migliorata grafica applicativo mobile
0.7.6	26-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Implementazione della gestione dei singoli elementi nella RecyclerView (Applicativo mobile). Refactoring e documentazione

Versione	Data	Autore	Descrizione
0.7.7	27-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Refactoring top-level dell'applicativo mobile. Modifiche minori
0.7.8	27-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Implementazione gestione della vista del singolo itinerario sull'Applicativo mobile Modifiche minori lato Server
0.7.9	28-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Implementazione gestione dei Tag (Lato Server). Edit minori Implementazione gestione dei Tag lato Client
0.7.9	28-02-2022	Valentino Bocchetti Mario Gabriele Carofano	Implementazione gestione delle Chat room (Lato Server). Implementazione gestione delle Chat room lato Client Implementazione gestione delle info di un messaggio
0.8.0	01-03-2022	Valentino Bocchetti Mario Gabriele Carofano	Implementazione gestione invio messaggi Bug fixes
0.8.1	02-03-2022	Valentino Bocchetti Mario Gabriele Carofano	Collegamento bucket S3 - informazioni singolo itinerario Modifiche Profilo (Applicativo android) Gestione della posizione corrente dell'utente Gestione dell'inserimento di un itinerario mediante GPX
0.8.2	04-03-2022	Valentino Bocchetti Mario Gabriele Carofano	Implementazione collegamento tra schermate (Applicativo mobile) Aggiunta di docker file. Bug fixes.
0.8.2	06-03-2022	Valentino Bocchetti Mario Gabriele Carofano	Aggiunta lambda per una mail custom durante la signup Modifiche minori
0.8.2	07-03-2022	Valentino Bocchetti Mario Gabriele Carofano	Aggiunta informazioni lato chat (Applicativo android) Testing sul logout dell'utente Completamento schermata filtri (Applicativo android)
0.8.3	09-03-2022	Valentino Bocchetti	Aggiunta Javadoc (Backend)
0.8.3.1	09-03-2022	Valentino Bocchetti	Creazione applicativo mobile - Lato admin
0.8.3.2	09-03-2022	Valentino Bocchetti	Implementazione componenti applicativo mobile (lato admin)
0.8.3.3	10-03-2022	Valentino Bocchetti	Aggiunta gestione degli admin (lato server)
0.8.4	10-03-2022	Valentino Bocchetti	Implementazione delle segnalazioni - Applicativo mobile (Utente)
0.8.5	10-03-2022	Valentino Bocchetti Mario Gabriele Carofano	Aggiunta degli itinerari nell'applicativo mobile
0.8.6	11-03-2022	Valentino Bocchetti	Implementazione gestione della mancata connessione al Server (Applicativo mobile - Utente e Admin)
0.8.7	11-03-2022	Valentino Bocchetti	Implementazione gestione della visualizzazione della mappa interattiva del percorso. Testing delle segnalazioni.
0.8.8	11-03-2022	Valentino Bocchetti	Implementazione gestione della visualizzazione della mappa interattiva del percorso. Testing delle segnalazioni.
0.8.9	11-03-2022	Valentino Bocchetti	Aggiunta gestione del tipo di utente loggato (tramite mail o con Google)

Versione	Data	Autore	Descrizione
0.8.9.1	12-03-2022	Valentino Bocchetti Mario Gabriele Carofano	Implementazione gestione delle statistiche di utilizzo dei servizi Modifiche minori
0.8.9.2	12-03-2022	Valentino Bocchetti	Testing dell'applicativo mobile (Admin). Aggiunta funzionalità
0.8.9.3	13-03-2022	Valentino Bocchetti	Implementazione della segnalazione sulla foto profilo del singolo itinerario
0.8.9.4	13-03-2022	Valentino Bocchetti	Aggiunta video per la presentazione del progetto
0.8.9.5	14-03-2022	Valentino Bocchetti	Implementazione della visualizzazione delle statistiche (Applicativo mobile - Admin). Aggiunta alla documentazione. Refactoring
0.8.9.6	15-03-2022	Valentino Bocchetti	Modifiche grafica delle statistiche Completamento gestione Applicativo mobile (Admin). Refactoring e testing Aggiunte alla documentazione
0.8.9.7	17-03-2022	Valentino Bocchetti	Implementazione richiesta di aggiunta di una foto aggiuntiva (Applicativo mobile) Inizializzazione del Testing
0.8.9.8	20-03-2022	Valentino Bocchetti Mario Gabriele Carofano	Aggiunte alla documentazione (Usabilità, Gerarchie funzionali) Introduzione del glossario Aggiunta opzioni custom per la gestione degli apk prodotti (release) Modifica opzioni Modifica della Presentazione dell'idea progettuale
0.8.9.9	21-03-2022	Valentino Bocchetti	Aggiunta controlli sulla registrazione dell'utente. Testing accesso con Provider esterno (Google)
0.9.0	22-03-2022	Valentino Bocchetti Mario Gabriele Carofano	Aggiunte alla documentazione. Modifiche minori. Analisi del progetto

2 Documento dei Requisiti Software

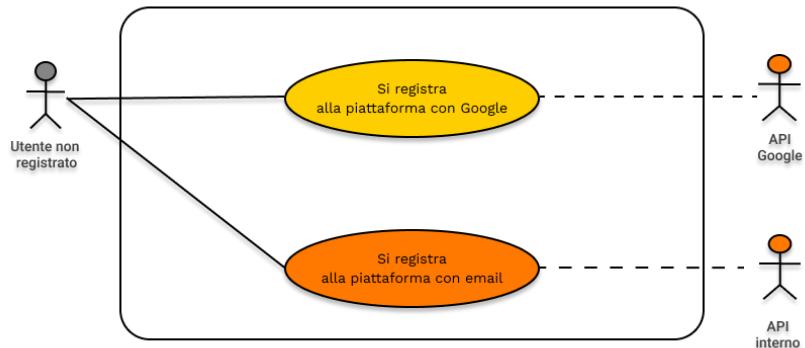
2.1 Modello funzionale

2.1.1 Modellazione dei Casi d'Uso

Per la modellizzazione dei casi d'uso prendiamo in esame:

- ▶ L'operazione per il **login/registrazione**, per utente (registrato o meno) e per gli amministratori del sistema;
- ▶ Operazioni esclusive per utente (registrato) e amministratori.

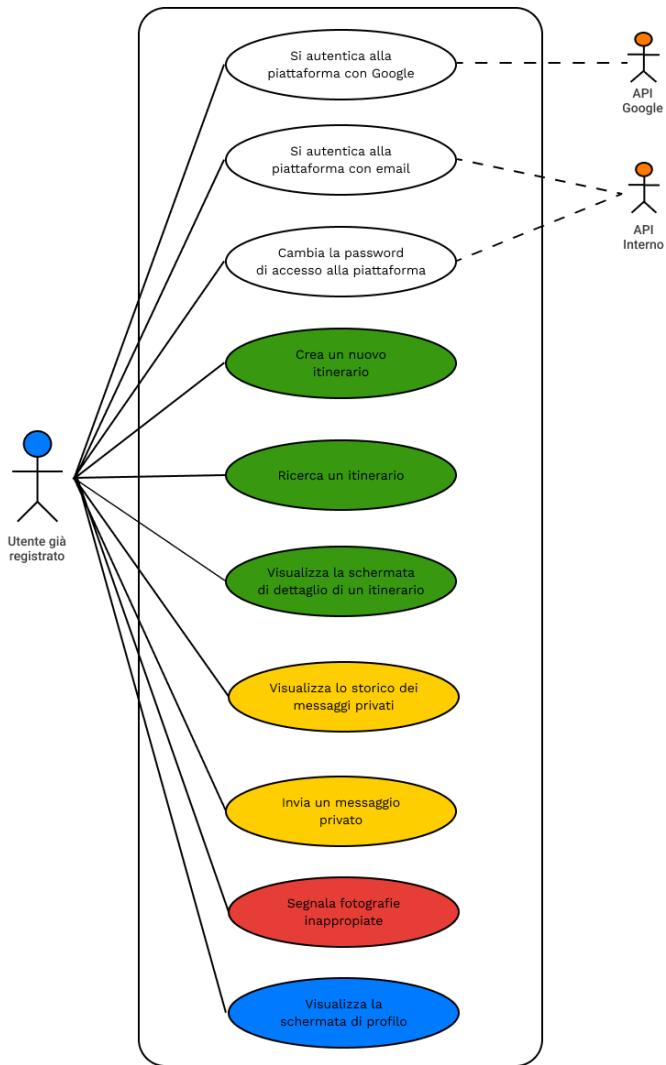
Registrazione per nuovi utenti



NaTour prevede che un utente, per accedere debba essere registrato. Offriamo quindi 2 metodi:

- ▶ Registrazione con Google;
- ▶ Registrazione con email, gestita internamente.

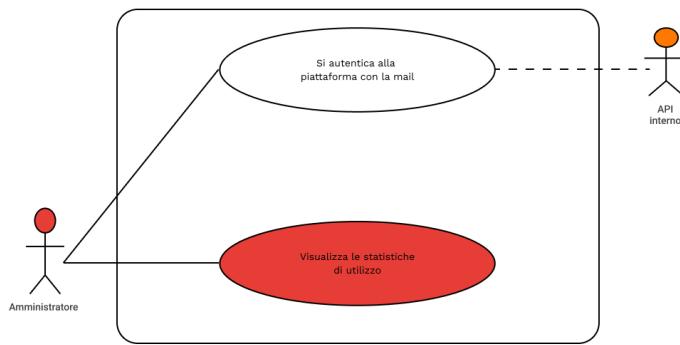
Operazioni per utenti già registrati



L'utente, dopo aver effettuato l'accesso, ha la possibilità di effettuare diverse operazioni:

- ▶ **Creare un nuovo itinerario**, caratterizzato da:
 - ◊ Nome;
 - ◊ Zona geografica (montagna, collina, mare, ...);
 - ◊ Durata minima del percorso (in ore);
 - ◊ Difficoltà, in relazione a dislivello e struttura del percorso;
 - ◊ Punto di inizio del percorso;
 - ◊ Punto di fine del percorso.
- ▶ **Ricercare un itinerario**, secondo parole chiave oppure filtri offerti dalla piattaforma;
- ▶ **Visualizzare lo storico dei messaggi privati**;
- ▶ **Inviare un messaggio privato ad altri utenti**, per chiedere informazioni sui percorsi offerti;
- ▶ **Segnalare fotografie inappropriate**.

Operazioni per amministratori



A differenza degli Utenti, gli account per gli amministratori sono già prefissati. In seguito all'accesso, possono:

- ▶ **Visualizzare le statistiche di utilizzo**, caratterizzate da:
 - ◊ Numero di utenti;
 - ◊ Numero di accessi (mensili, settimanali, giornalieri, ...);
 - ◊ Numero di ricerche (tag più cercati, città più cercate, zone più cercate, ...);

2.1.2 Tabelle di Cockburn (per 2 casi significativi)

Creazione itinerario (Tabella 1)

USE CASE #1	Crea un nuovo itinerario		
Goal in Context	L'utente già registrato vuole creare un nuovo itinerario da aggiungere all'applicazione NaTour		
Scope	System under Development (SuD)		
Level	User-goal		
Preconditions	L'utente già registrato deve essersi registrato ed autenticato all'applicazione NaTour		
Success End Condition	L'utente già registrato riesce a creare correttamente il nuovo itinerario		
Failed End Condition	L'utente già registrato non riesce a creare il nuovo itinerario; il sistema notifica il fallimento dell'operazione mediante pop-up e permette all'utente già registrato di reinserire i dati		
Primary, Secondary actors	Utente già registrato		
Trigger	L'utente già registrato preme il bottone "Nuovo Itinerario Button" in "Profilo"		
DESCRIPTION	Step	Utente già registrato	Sistema
	1	Preme il bottone "Nuovo Itinerario Button" in "Profilo"	
	2		Mostra "Nuovo Itinerario Popup"
	3	Preme il bottone "Seleziona Punti Mappa Button"	
	4		Mostra "Seleziona Punti Mappa"
	5	Seleziona due punti sulla mappa per determinare il percorso dell'itinerario da inserire	
	6	Preme il bottone "Conferma Button"	
	7		Mostra "Nuovo Itinerario"
	8	Inserisce i dati richiesti e obbligatori, ovvero: • Nome; • Città; • Zona geografica; • Lunghezza; • Durata; • Difficoltà	
	9	Preme il bottone "Conferma Button"	
	10		Mostra "Nuovo Itinerario - Inserimento immagine"
	11	Preme il bottone "Si Button"	
	12		Mostra la schermata di selezione delle immagini dalla memoria interna del dispositivo
	13	Seleziona un'immagine dalla memoria interna	
	14		Carica l'itinerario appena creato con tutti i dati inseriti e l'immagine scelta nell'applicazione NaTour
	15		Mostra "Homepage"

EXTENSIONS	Step	Utente già registrato	Sistema
< Il sistema non riesce a caricare la mappa perché l'utente già registrato non ha concesso i permessi necessari >	4.a.1		Mostra "Inserimento Permessi Popup"
	4.a.2	Preme il bottone "Consenti Button"	
Si ricollega allo step 4 del Main Scenario			
< L'utente già registrato seleziona meno di due punti dalla mappa >	7.a.1		Mostra "Popup errore - Errore punti mappa"
	Si ricollega allo step 4 del Main Scenario		
< L'utente già registrato seleziona un file la cui estensione non è .GPX >	7.b.1		Mostra "Popup errore - Estensione non corretta"
	Si ricollega allo step 2 del Main Scenario		
< L'utente già registrato non inserisce tutti i dati obbligatori richiesti >	10.a.1		Mostra "Popup errore - Campi obbligatori"
	Si ricollega allo step 7 del Main Scenario		
< Il sistema non riesce a creare correttamente il nuovo itinerario >	10.b.1		Mostra "Popup errore - Errore creazione"
	Si ricollega allo step 7 del Main Scenario		
< L'utente già registrato seleziona un file che non è un'immagine >	14.a.1		Mostra "Popup errore - Errore inserimento immagine"
	Si ricollega allo step 10 del Main Scenario		
SUB-VARIATIONS	Step	Utente già registrato	Sistema
< L'utente registrato può creare un nuovo itinerario a partire dalle informazioni di un file .GPX >	3	Preme il bottone "Importa File .GPX Button"	
	4		Mostra la schermata di selezione dei file dalla memoria interna del dispositivo
	5	Seleziona un file dalla memoria interna	
	Si ricollega allo step 7 del Main Scenario		
< L'utente registrato decide di inserire anche i dati facoltativi >	8	Inserisce i dati obbligatori e anche quelli facoltativi: • Descrizione; • Tag di ricerca	
	Si ricollega allo step 9 del Main Scenario		
< L'utente registrato decide di non voler caricare alcuna fotografia per il nuovo itinerario >	11	Preme il bottone "No Button"	
	12		Carica l'itinerario appena creato con tutti i dati inseriti e senza alcuna immagine nell'applicazione NaTour
	Si ricollega allo step 15 del Main Scenario		

Registrazione alla piattaforma con email (Tabella 2)

USE CASE #2	Si registra alla piattaforma con email		
Goal in Context	Un utente non registrato desidera registrarsi alla piattaforma NaTour per poter accedere ai servizi offerti		
Scope	System under Development (SuD)		
Level	User-goal		
Preconditions	Nessuna precondition da segnalare		
Success End Condition	L'utente non registrato riesce a registrarsi alla piattaforma con successo		
Failed End Condition	L'utente non riesce a registrarsi alla piattaforma; il sistema notifica il fallimento della registrazione mediante pop-up e permette all'utente non registrato di reinserire i dati		
Primary, Secondary actors	Utente non registrato		
Trigger	L'utente non registrato preme il bottone "Registrati Button" in "Login"		
DESCRIPTION	Step	Utente non registrato	Sistema
	1	Preme il bottone "Registrati Button" in "Login"	
	2		Mostra "Sign-Up"
	3	Inserisce i dati richiesti e obbligatori, ovvero: • Username; • Email; • Password (con conferma)	
	4	Preme il bottone "Registrati Button"	
	5		Mostra "Termini e Condizioni Popup"
	6	Preme il bottone "Conferma Button"	
	7		Invia una email all'utente non registrato all'indirizzo mail specificato nel form, contenente un codice di conferma
	8		Mostra "Codice di Conferma Popup"
	9	Inserisce il codice di conferma ricevuto all'indirizzo mail specificato nel form	
	10	Preme il bottone "Conferma Button"	
	11		Finalizza la registrazione dell'utente con tutti i dati inseriti
	12		Memorizza localmente le credenziali di accesso dell'utente
	13		Mostra "Homepage"

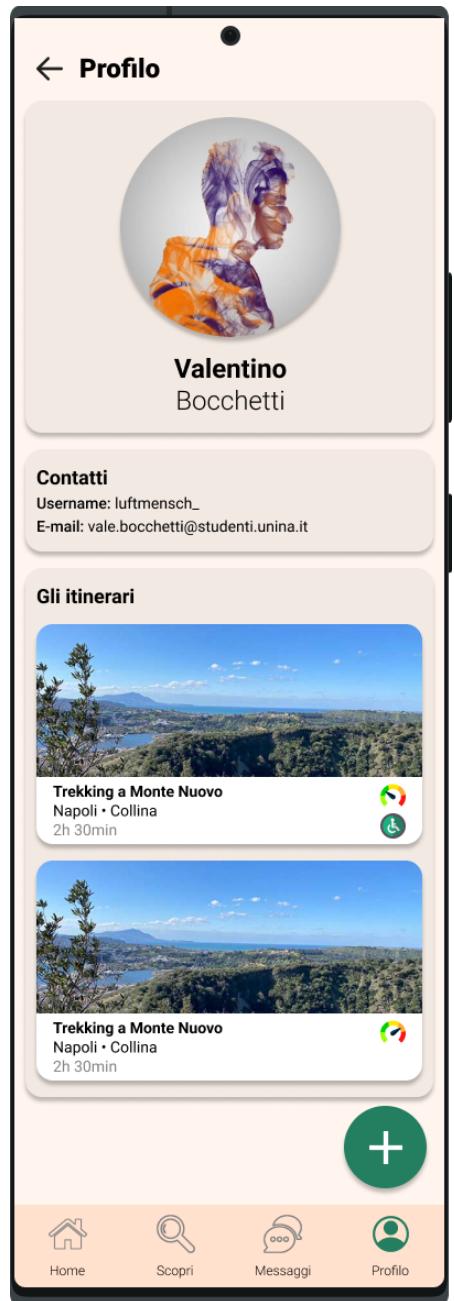
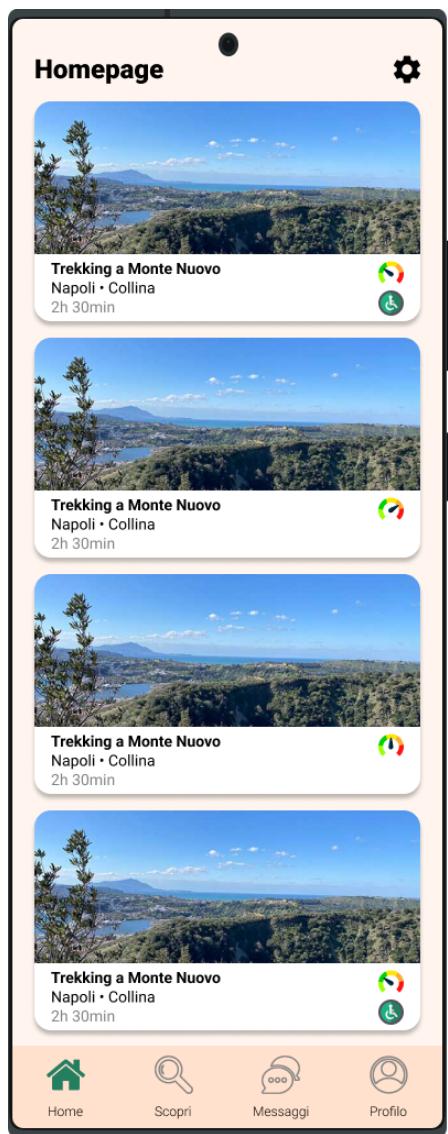
EXTENSIONS	Step	Utente non registrato	Sistema	
< L'utente non registrato non inserisce tutti i dati obbligatori richiesti >	5.a.1		Mostra "Popup errore - Campi obbligatori"	
		Si ricollega allo step 2 del Main Scenario		
< L'utente non registrato inserisce una stringa di testo nel campo "Email" che non ha il formato di un indirizzo mail >	5.b.1		Mostra "Popup errore - Formato email non corretto"	
		Si ricollega allo step 2 del Main Scenario		
< L'utente non registrato inserisce due password diverse nei campi "Password" e "Conferma Password" >	5.c.1		Mostra "Popup errore - Password non uguali"	
		Si ricollega allo step 2 del Main Scenario		
< L'utente non registrato inserisce una password con meno di 8 caratteri >	5.d.1		Mostra "Popup errore - Lunghezza password"	
		Si ricollega allo step 2 del Main Scenario		
< L'utente non registrato inserisce una stringa di testo nel campo "Password" che non ha il formato di una password sicura >	5.e.1		Mostra "Popup errore - Formato password non corretto"	
		Si ricollega allo step 2 del Main Scenario		
< Il sistema non riesce a finalizzare la registrazione dell'utente non registrato >	12.a.1		Mostra "Popup errore - Errore registrazione"	
		Si ricollega allo step 2 del Main Scenario		
SUB-VARIATIONS	Step	Utente non registrato	Sistema	
< Prima di proseguire con la registrazione, l'utente non registrato può visualizzare i termini e condizioni d'uso >	6	Preme il bottone "Visualizza Termini e Condizioni Button"		
		Si ricollega allo step 6 del Main Scenario		

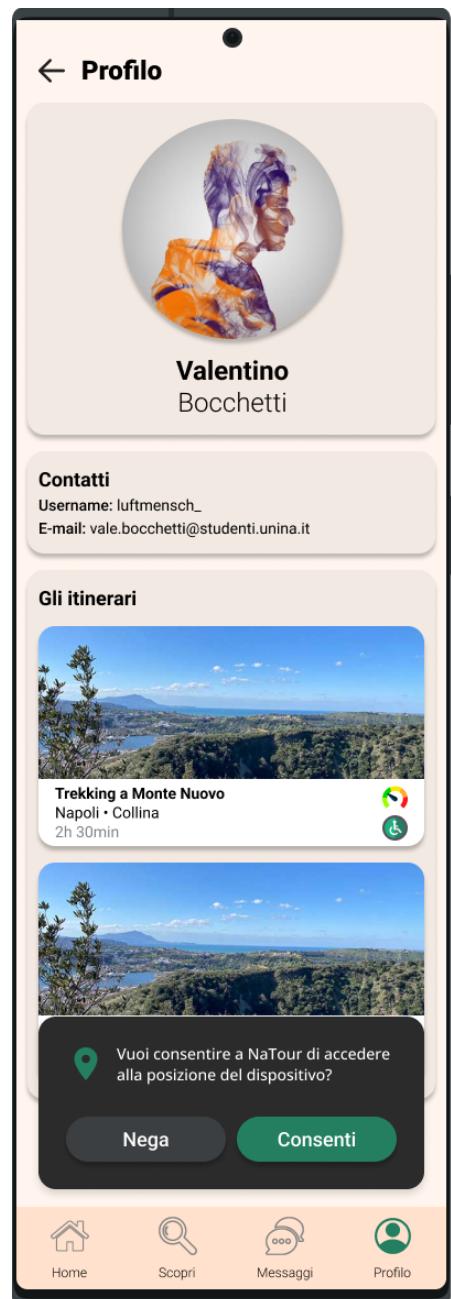
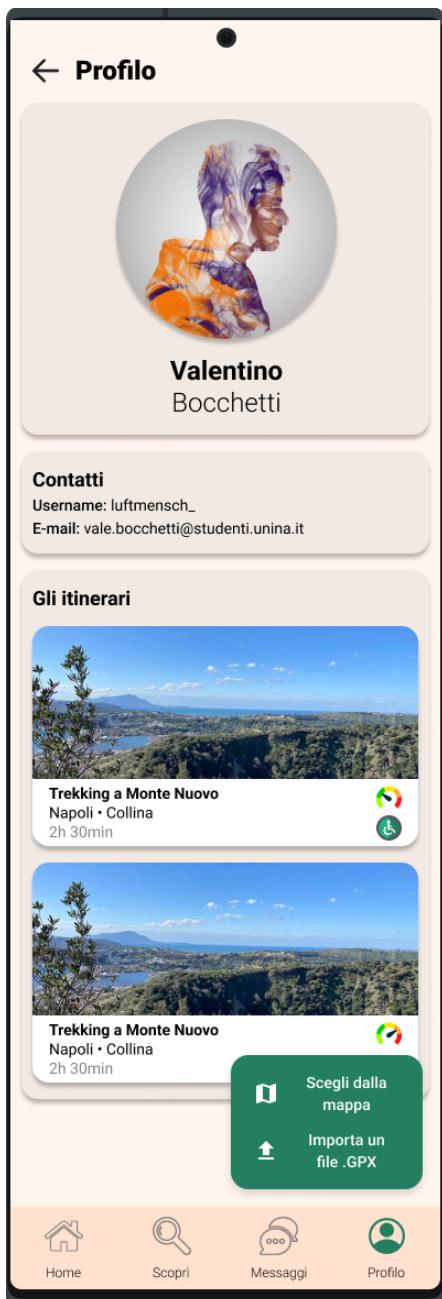
2.1.3 Prototipazione visuale via Mock-up (per 2 casi significativi)

Mockup relativo allo use case n° 1

Di seguito vengono riportati i mockup relativi a:

- ▶ Homepage;
- ▶ Profilo;
- ▶ Popup per creare un nuovo itinerario;
- ▶ Popup per la richiesta dei permessi necessari ad accedere al GPS;
- ▶ Inserimento dei campi necessari per la creazione di un nuovo itinerario;
- ▶ Inserimento di una foto per il nuovo itinerario appena creato;
- ▶ Selezione dei punti di **inizio/fine** dell'itinerario su una mappa interattiva;
- ▶ Avviso di errore nel caso in cui i campi obbligatori per la creazione di un nuovo itinerario o fossero mancanti;
- ▶ Avviso di errore per il fallimento nella creazione di un nuovo itinerario;
- ▶ Avviso di errore per il fallimento nel caricamento dell'immagine di un nuovo itinerario;
- ▶ Avviso di errore nella selezione di un file in un formato non corretto;





← Nuovo itinerario

Nome*

Descrizione

Informazioni percorso

Città*

Zona geografica*

Lunghezza*

Dislivello*

Durata*

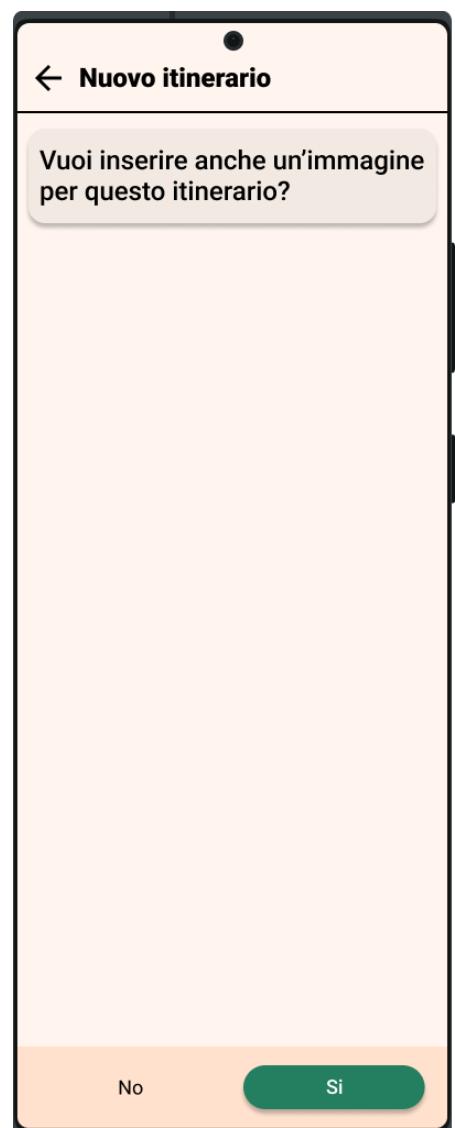
Difficoltà*

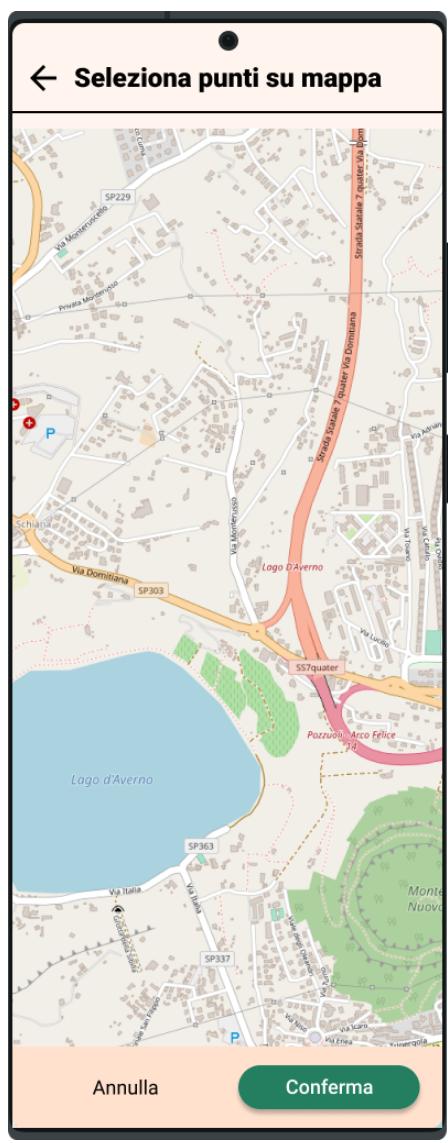
Tag di ricerca

Il percorso è accessibile ai disabili?

Annulla

Conferma





← Nuovo itinerario

Nome*

Descrizione

Informazioni percorso

Città*

Zona geografica*

Lunghezza*

Dislivello*

Durata*

Difficoltà*

Tag di ricerca

X Attenzione! I campi segnati da * sono obbligatori

Annula Conferma

← Nuovo itinerario

Nome*

Descrizione

Informazioni percorso

Città*

Zona geografica*

Lunghezza*

Dislivello*

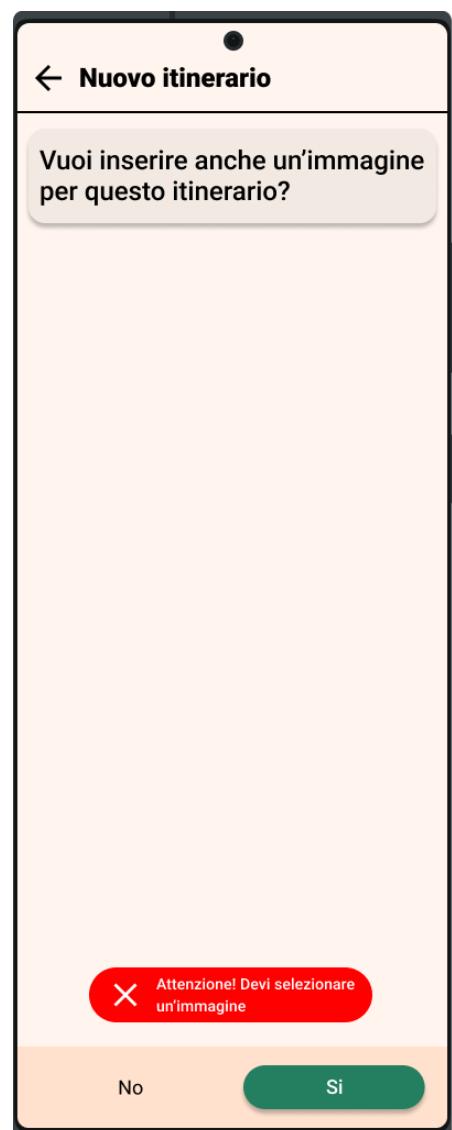
Durata*

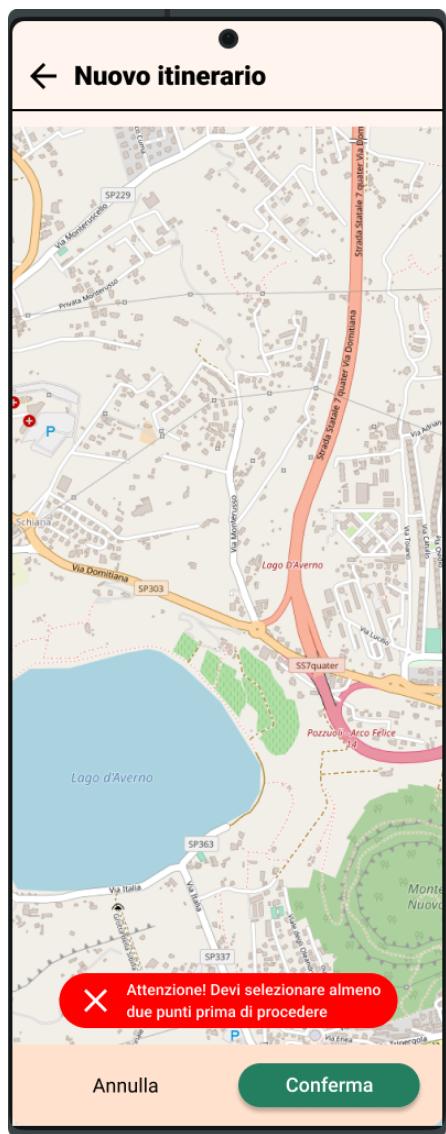
Difficoltà*

Tag di ricerca

Attenzione! Errore durante la creazione del nuovo itinerario

Annula Conferma





← Profilo

Contatti
Username: luftmensch_
E-mail: vale.bocchetti@studenti.unina.it

Gli itinerari

Trekking a Monte Nuovo
Napoli • Collina
2h 30min

Trekking a Monte Nuovo
Napoli • Collina
2h 30min

Scegli dalla mappa

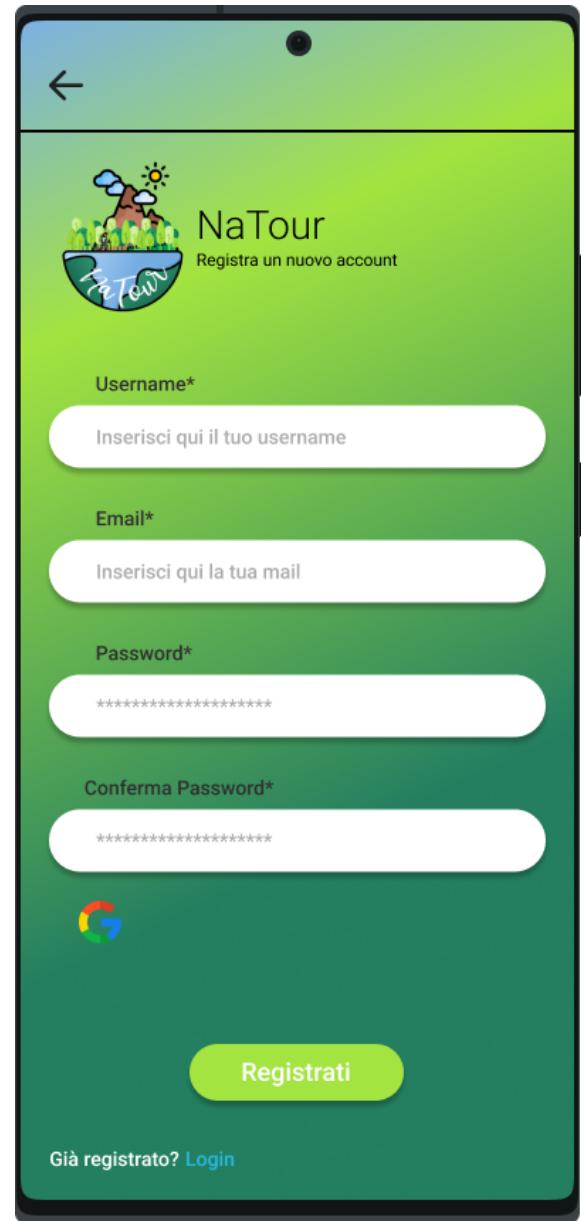
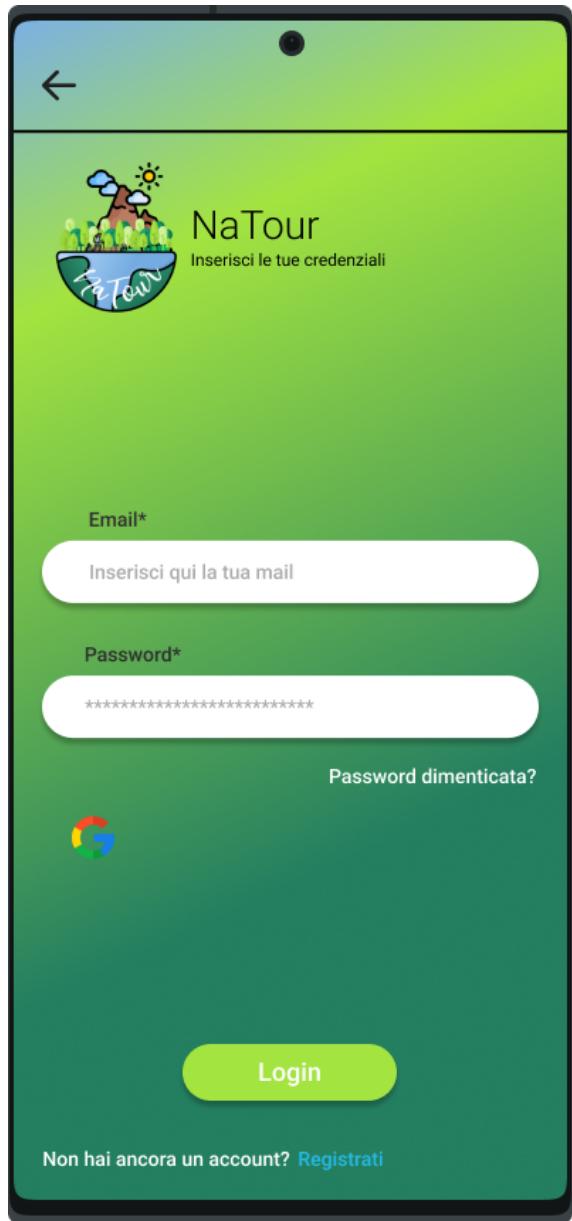
Attenzione! Devi selezionare un file con estensione .GPX

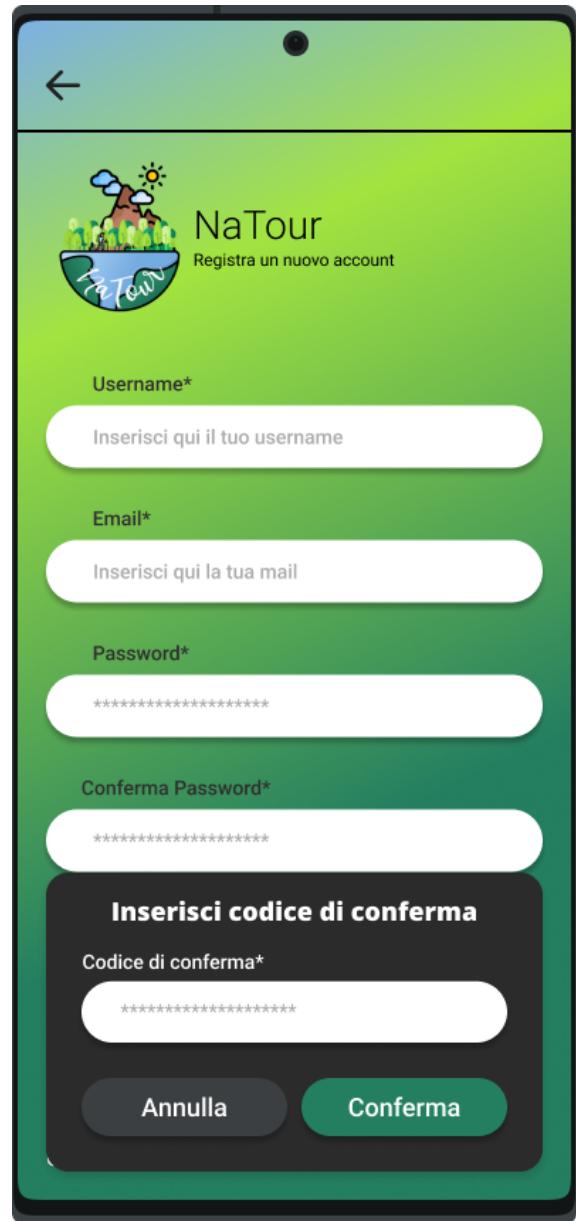
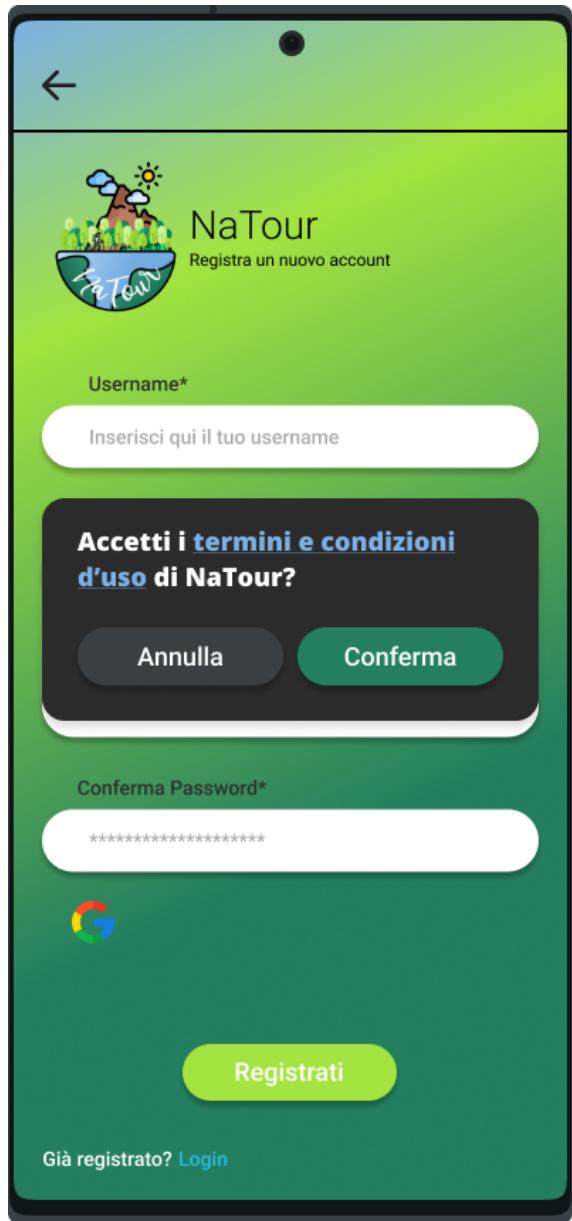
Home **Scopri** **Messaggi** **Profilo**

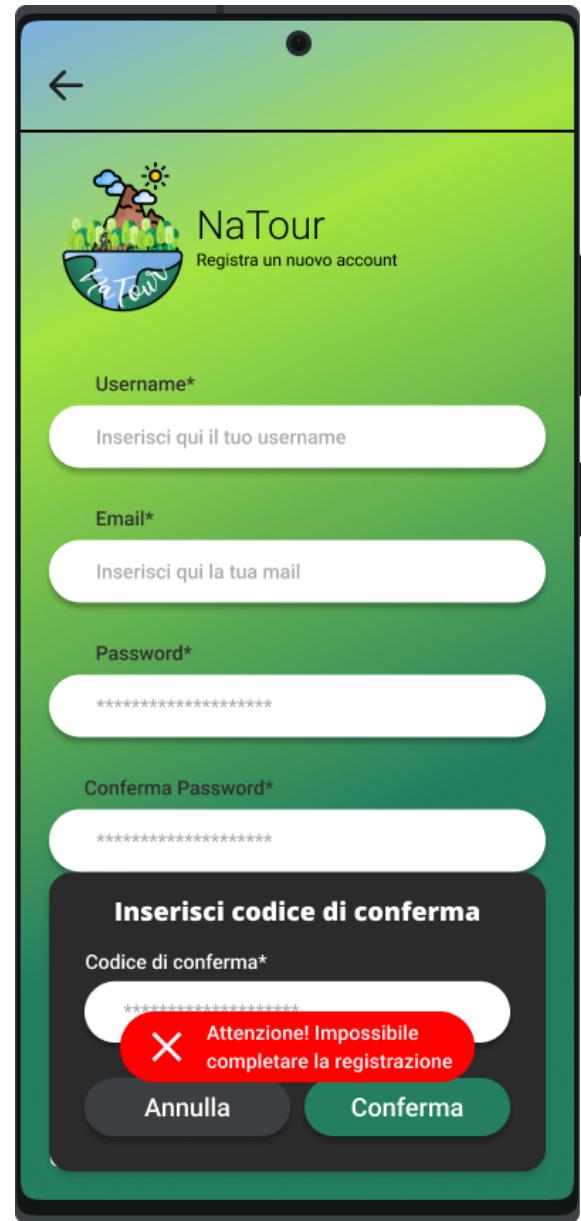
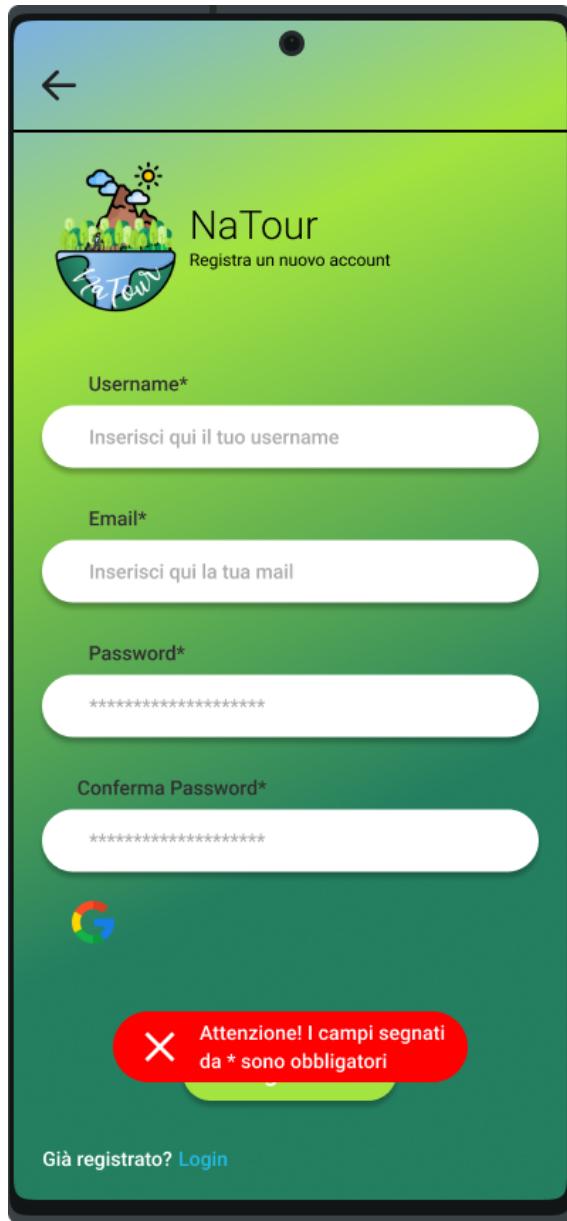
Mockup relativo allo use case n° 2

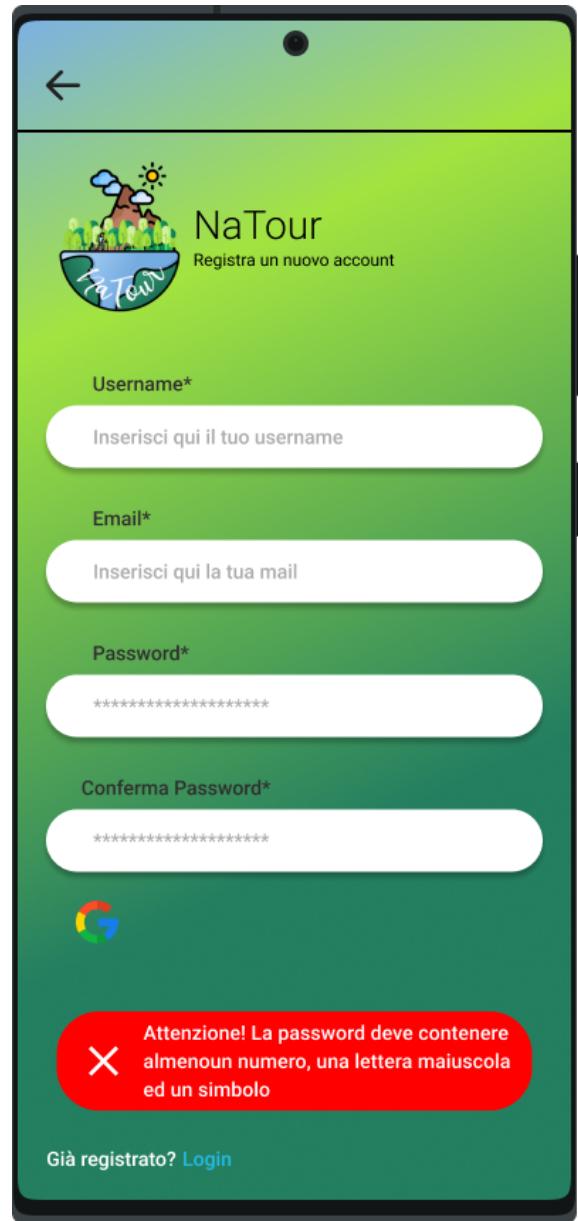
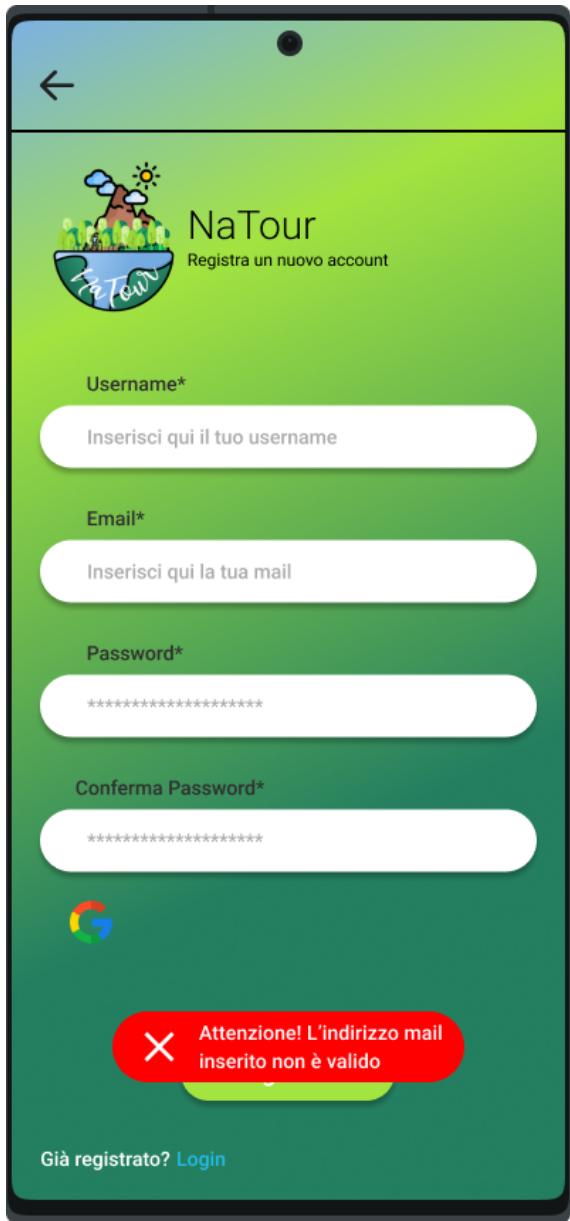
Di seguito vengono riportati i mockup relativi a:

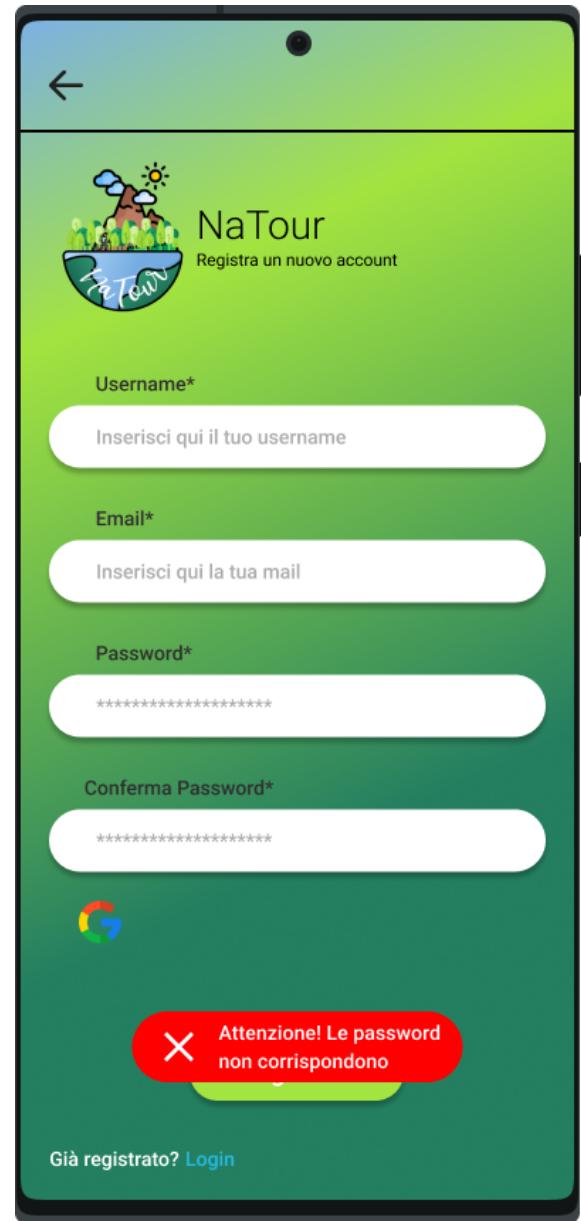
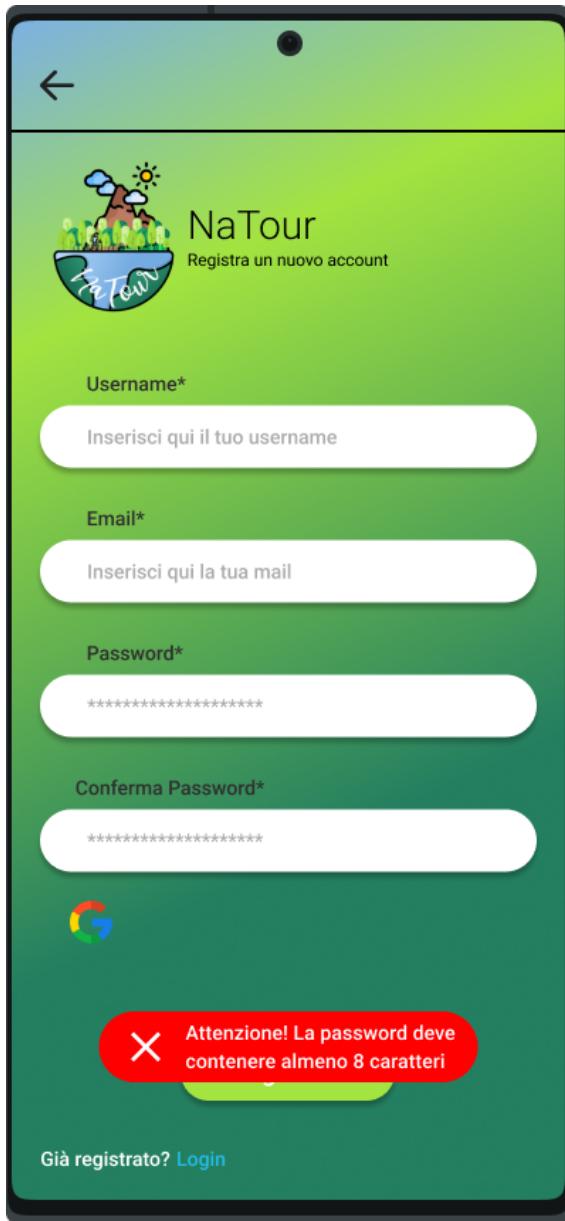
- ▶ Login di un utente già registrato;
- ▶ Registrazione di un nuovo utente;
- ▶ Termini e condizioni per la registrazione;
- ▶ Codice di conferma per la registrazione;
- ▶ Avviso di errore nel caso in cui i campi obbligatori fossero mancanti;
- ▶ Avviso di errore durante la registrazione;
- ▶ Avviso di errore nel caso in cui il formato della email non fosse corretto;
- ▶ Avviso di errore nel caso in cui il formato della password non fosse corretto;
- ▶ Avviso di errore nel caso in cui la lunghezza della password non fosse corretta;
- ▶ Avviso di errore nel caso in cui le 2 password non corrispondessero;











2.1.4 Presentazione dell'idea progettuale



Se vi piacciono l'escursionismo e l'aria aperta, NaTour è l'app che fa per voi.

Vi presentiamo NaTour!

NaTour è un'applicazione **mobile di social networking** dedicata agli **appassionati di trekking** in tutto il mondo, che permette loro di essere guidati nella natura, in un sentiero escursionistico o in altre attività all'aperto come trekking, ciclismo e sci grazie solo all'utilizzo di uno smartphone. NaTour è al momento disponibile solo su Android, ma si prevede lo sviluppo di applicativi anche per iOS e per il Web.

Gli utenti possono accedere ai **servizi offerti** da NaTour **registrandosi alla piattaforma** fornendo il loro *username, email e password* oppure utilizzando il loro account *Google*. Una volta registrati o autenticati alla piattaforma, i Natourers scopriranno quanto sia semplice *cercare itinerari* (anche utilizzando filtri più avanzati) o *crearne di nuovi* (per mettere alla prova sé stessi o semplicemente condividerli con i propri amici) grazie al suo **design UI/UX semplice e intuitivo**. È possibile ricercare itinerari attraverso dei filtri per *nome*, per *possibilità di accesso ai disabili*, per *durata*, per *autore*, per *livello di difficoltà*, per *zona geografica*. I risultati, una volta terminata la ricerca, verranno visualizzati in una lista chiara ed auto esplicativa.

NaTour è un software che **fornisce informazioni nel dettaglio degli itinerari disponibili** in piattaforma come la **lunghezza** del percorso, il **dislivello** massimo, la **durata** dell'escursione organizzata, la **latitudine** e **longitudine** dei punti di inizio e fine e tante altre. Inoltre, **permette agli utenti di inviare messaggi privati** l'un l'altro, ad esempio per chiedere maggiori informazioni al creatore di un itinerario oppure per stringere nuove amicizie in questo campo.

NaTour rappresenta **l'evoluzione di una mappa vecchio stile** e mira a reinventare il modo in cui le persone pianificano i loro percorsi di trekking, indirizzandole attraverso i percorsi più panoramici e più fotogenici. La presenza di una **completa libreria di percorsi online** permetterà loro di scoprire nuovi luoghi e avventure da condividere sui social network, e di creare un'esperienza di viaggio **condivisa, interattiva e sicura**: infatti gli utenti possono **catturare** e condividere momenti nei loro viaggi mediante **l'upload di foto** e, in più, **segnalare eventuali fotografie non opportune** che, durante l'attenta valutazione da parte degli amministratori di NaTour, non potranno essere visualizzate nella schermata di dettaglio di un sentiero.

Per rendere NaTour più competitiva in questo settore, prevediamo di aggiungere nuove funzionalità assai utili come la possibilità di aggiungere più tappe ad un singolo itinerario, la possibilità di condividere un itinerario su altre applicazioni esterne, la possibilità di lasciare recensioni e commenti nella schermata di dettaglio di un sentiero o, addirittura, la possibilità di visualizzare le previsioni meteorologiche della zona geografica dove si tiene un'escursione.

In particolare:

- ▶ Abbiamo deciso di dividere la piattaforma NaTour in due applicazioni:
 - ◊ La prima sarà dedicata solo ai Natourers, e forniremo tutte le funzionalità descritte sopra;
 - ◊ La seconda sarà dedicata solo agli amministratori e permetterà loro di visualizzare una dashboard di statistiche in tempo reale sul sistema, come il numero di itinerari totali, il numero di utenti registrati, il numero di chat room tra utenti e anche il numero di messaggi scambiati.
- Abbiamo pensato di effettuare questa distinzione siccome Natourers e amministratori rappresentano due gruppi di utenti con esigenze troppo diverse.
- ▶ Da un punto di vista progettuale, abbiamo pensato di seguire un modello di processo misto:
 - ◊ Per le fasi di analisi dei requisiti, prototipazione dell'interfaccia utente e progettazione, seguiremo un modello di processo a cascata;
 - ◊ Per le fasi di sviluppo e testing, invece, seguiremo un modello di processo iterativo.

In questo modo, riusciremo ad iniziare la fase di sviluppo con un documento di specifica dei requisiti completo e formale, che ci permetterà di essere molto più produttivi. Invece la fase di sviluppo sarà iterativa perché, così facendo, riusciremo ad ottenere un prodotto software utilizzabile e testabile già a partire dalle prime iterazioni, che verrà progressivamente aggiornato con nuove funzionalità.

- ▶ Da un punto di vista tecnico, ci sembra obbligatorio rendere i nostri prodotti software quanto più comprensibili, leggibili, riusabili e manutenibili possibile. Per far ciò, ci baseremo sui principi di progettazione SOLID:
 - ◊ Abbiamo pensato di progettare le nostre classi e interfacce in modo tale che siano quanto più indipendenti possibile l'una dall'altra, per ottenere un basso accoppiamento, e in modo tale che presentino solo attività e funzionalità strettamente legate tra loro (singola responsabilità), al fine di avere alta coesione;
 - ◊ Abbiamo preso in considerazione l'idea di realizzare classi aperte alle estensioni ma, allo stesso tempo, chiuse alle modifiche: questa importante proprietà ci permetterà, durante le iterazioni della fase di sviluppo, di non riscontrare grandi difficoltà nell'aggiungere nuove funzionalità senza che altre già sviluppate e testate richiedessero nuove modifiche o nuovi test.

Per concludere, vi invitiamo a scaricare l'applicazione oggi stesso e a scoprire

"Un nuovo modo per fare trekking!"

2.1.5 Requisiti funzionali

Di seguito vengono elencati in dettaglio i requisiti funzionali del sistema:¹

ID	NaTour_RF01
Nome	Registrazione alla piattaforma
Descrizione	Il sistema deve consentire ad un utente non registrato di potersi registrare alla piattaforma indicando username email e password oppure utilizzando un proprio account Google

ID	NaTour_RF02
Nome	Autenticazione alla piattaforma
Descrizione	Il sistema deve consentire ad un utente già registrato o un amministratore di poter accedere alla piattaforma indicando il proprio username e la password. In particolare, l'utente può autenticarsi anche utilizzando un proprio account Google

ID	NaTour_RF03
Nome	Cambio password
Descrizione	Il sistema deve consentire ad un utente già registrato di poter modificare la propria password corrente indicando una nuova password

ID	NaTour_RF04
Nome	Effettuare ricerche di itinerari
Descrizione	Il sistema deve consentire ad un utente già registrato di poter effettuare ricerche di itinerari presenti nel sistema attraverso un sistema di filtri per nome, per durata, per zona geografica, per livello di difficoltà e per possibilità di accesso ai disabili

ID	NaTour_RF05
Nome	Visualizzare un itinerario
Descrizione	Il sistema deve consentire ad un utente già registrato di poter visualizzare una schermata di dettaglio di uno specifico itinerario in modo da poterne consultare le relative informazioni

ID	NaTour_RF06
Nome	Creare un nuovo itinerario
Descrizione	Il sistema deve consentire agli utenti già registrati di poter creare nuovi itinerari da aggiungere in piattaforma specificando caratteristiche come città, zona geografica, lunghezza e così via, selezionando i punti di inizio e fine tramite una mappa interattiva o l'inserimento di un file GPX e, eventualmente, inserendo una fotografia.

¹Si intendono quei requisiti che descrivono le funzionalità offerte dal sistema.

ID	NaTour_RF07
Nome	Inviare messaggi privati
Descrizione	Il sistema deve consentire agli utenti già registrati di poter inviare messaggi privati l'un l'altro, ad esempio per chiedere informazioni più dettagliate su un itinerario

ID	NaTour_RF08
Nome	Visualizzare messaggi privati
Descrizione	Il sistema deve consentire ad un utente già registrato di poter accedere all'elenco delle conversazioni aperte con altri utenti già registrati.

ID	NaTour_RF09
Nome	Segnalare fotografie inappropriate
Descrizione	Il sistema deve consentire ad un utente già registrato di poter segnalare fotografie ritenute inappropriate secondo il suo parere. Il sistema, quindi, non deve visualizzare nella schermata di dettaglio di un itinerario le fotografie con una segnalazione in sospeso.

ID	NaTour_RF10
Nome	Visualizzare statistiche in tempo reale
Descrizione	Il sistema deve consentire ad un amministratore di poter visualizzare statistiche in tempo reale sui numeri dell'applicazione come il numero di utenti registrati oppure il numero di itinerari inseriti e così via.

ID	NaTour_RF11
Nome	Visualizzare la schermata di profilo
Descrizione	Il sistema deve consentire ad un utente già registrato di poter visualizzare una schermata di dettaglio di un profilo utente in modo da poterne consultare le relative informazioni.

2.1.6 Requisiti non funzionali

Di seguito vengono elencati i requisiti funzionali del sistema:²

ID	NaTour_RNF01
Nome	Performance di ricerca
Descrizione	Il sistema deve mostrare i risultati di una ricerca entro 3 secondi dall'avvio della stessa, almeno nel 90% dei casi

ID	NaTour_RNF02
Nome	Usabilità dell'applicazione
Descrizione	Un utente deve riuscire ad utilizzare la piattaforma, al massimo delle sue funzionalità, dopo una media di circa 2 ore di utilizzo

ID	NaTour_RNF03
Nome	Limitazione account spam
Descrizione	Il sistema deve richiedere la conferma dell'account di un utente mediante mail, al termine della registrazione

ID	NaTour_RNF04
Nome	Password Policy
Descrizione	Il sistema deve forzare l'utente a inserire una password di almeno 8 caratteri contenenti numeri, almeno una lettera maiuscola ed una minuscola

ID	NaTour_RNF05
Nome	Adattabilità al back-end
Descrizione	Il sistema deve essere adattabile ai possibili cambiamenti del back-end e quindi essere completamente slegato rispetto a quest'ultimo

²Sono quelli che descrivono i vincoli sui servizi offerti dal sistema e sullo stesso processo di sviluppo.

2.1.7 Individuazione del target degli utenti

Il termine *target* è adottato dalla lingua inglese e significa letteralmente *bersaglio*. In questo contesto rappresenta l'insieme dei clienti e delle persone da raggiungere nelle nostre azioni di comunicazione e di marketing, ovvero il mercato di destinazione composto dai potenziali destinatari della nostra applicazione, NaTour.

Per l'individuazione del target degli utenti abbiamo, quindi, effettuato una vera e propria ricerca di mercato, suddivisa nelle fasi di *segmentazione del mercato* e *identificazione dei target group*.

Segmentazione del mercato	Identificazione dei target group
Dividere un mercato in vari segmenti permette di dividere la popolazione in target group che possono essere misurati in base a delle caratteristiche chiave i dati demografici	Un target group è un insieme di persone con caratteristiche simili (come indicato nelle note 3, 4 e 5) sui quali vogliamo porre l'attenzione per lo sviluppo dell'applicazione. I consumatori che rientrano nello stesso target group tendono a valutare gli stessi prodotti e servizi

Per caratteristiche chiave si intendono dati come:

- ▶ **Dati demografici:**³
- ▶ **Geolocalizzazione:**⁴
- ▶ **Interessi ed opinioni.**⁵

Per riuscire in questo compito ci siamo posti le seguenti domande:

- ▶ Chi accede al nostro prodotto?
- ▶ Chi è interessato al suo utilizzo?
- ▶ A quali bisogni del cliente vogliamo rispondere con la nostra applicazione?
- ▶ Quali clienti otterranno benefici dalla nostra applicazione?
- ▶ Ci sono nicchie di mercato a cui far riferimento?
- ▶ Abbiamo dei Competitor?

Per rispondere a queste domande, abbiamo fatto riferimento ai dati esposti sui siti web di condivisione di itinerari ed avventure come [TrekkingItalia](#) (in Italia) e [AllTrails](#) (nel resto del mondo). In particolare ci siamo affidati anche alle esperienze raccontate dai nostri amici di [VisitCampiFlegrei](#), attivi con molte escursioni nella zona di Napoli e Pozzuoli.

³Ovvero età, genere, settore di occupazione, livello di istruzione, redditi, stile di vita.

⁴È importante restringere l'area geografica di utilizzo dell'applicazione per l'individuazione di un target più coerente con la realtà.

⁵Si intendono informazioni sugli interessi, sul tempo libero, sport, hobby, oppure informazioni su opinioni riguardo la sostenibilità e la tutela ambientale.

Quindi, dopo aver intervistato alcuni esperti nel settore del trekking e delle passeggiate outdoor, abbiamo identificato le nostre esigenze in un target group molto ampio, per chiunque ami fare escursioni ed esplorare, dai giovanissimi sino agli over 60, rappresentato dalle seguenti "user personas".⁶

Eliana De Amicis

Professoressa di lingua inglese

"Guardare la bellezza della natura è il primo passo per purificare la mente"

BIOGRAFIA
Una donna grintosa, energica ed altruista. Appassionata della lingua e della cultura britannica, insegna in un istituto superiore ormai da anni. Si occupa dei suoi nipoti, ed il rapporto con loro ed i suoi studenti è il motore che anima le sue giornate.

INFORMAZIONI GENERALI
Sesso: Donna
Età: 56 anni
Residenza: Pisa

HOBBY E INTERESSI
Le piace mantenersi sempre fresca e attiva. Ama fare ginnastica all'aperto, correre nel parco vicino casa ed organizzare pic-nic con parenti ed amici.

OBIETTIVI
È alla ricerca di nuovi itinerari e percorsi naturalistici nei quali coinvolgere i suoi ragazzi, perché il contatto con la natura aiuta ad entrare in connessione con il proprio io più profondo.

Cosimo Imparato

Pensionato (ex capostazione)

"La vita non toglie nulla, dona soltanto"

BIOGRAFIA
Uomo spiritoso e pieno di vita alla ricerca di nuove avventure, da pochi anni si gode la pensione ed il tempo libero. All'inizio non è stato semplice adattarsi alla nuova routine, ma ben presto ha scoperto la bellezza delle lunghe passeggiate all'aria aperta e la voglia di respirare aria pulita e conoscere persone nuove con cui condividere le proprie esperienze.

INFORMAZIONI GENERALI
Sesso: Uomo
Età: 69 anni
Residenza: Napoli

HOBBY E INTERESSI
Alzarsi presto al mattino, godere delle prime luci dell'alba, passeggiare sul lungomare ed osservare la città che si sveglia. Sorreggere un caffè e fare due chiacchiere con gli amici.

OBIETTIVI
Conoscere nuovi percorsi per entrare in connessione con la natura.

Maurizio Caccavale

Personal Trainer

"Non arrendersi! Rischieresti di farla un'ora prima del traguardo"

BIOGRAFIA
Lo sport è la sua vita, la sua arma, la sua rinascita, la sua rivincita. È stato ciò che lo ha aiutato a superare l'ostacolo di un'infanzia difficile. Disciplina, sacrificio e tenacia lo hanno spronato. Ed anche se non ha ancora vinto alcuna medaglia, oggi è felice di allenare altre persone ad avere un fisico ed un'alimentazione sani. Per lui, sono le chiavi per la felicità.

INFORMAZIONI GENERALI
Sesso: Uomo
Età: 30 anni
Residenza: Pozzuoli

HOBBY E INTERESSI
Ama gli sport estremi, la competizione, darsi e superare sempre nuove sfide. Correre, scalare montagne, ammirare il cielo stellato permettendo all'aria aperta.

OBIETTIVI
Condividere con il resto del mondo la passione per lo sport, per la natura ed avere accesso ad itinerari sul territorio in continuo aggiornamento.

Miriam Acunzo

Laureata in Scienze della comunicazione

"La propria destinazione non è mai un luogo, ma un nuovo modo di vedere le cose"

BIOGRAFIA
È una ragazza creativa, socievole e solare, ma anche molto sensibile. Ha proseguito gli studi a pieni voti e non si ferma davanti a niente per raggiungere il suo obiettivo. Il suo sogno è diventare direttrice creativa di un giornale.

INFORMAZIONI GENERALI
Sesso: Donna
Età: 26 anni
Residenza: Roma

HOBBY E INTERESSI
Non ama molto fare palestra o sport, ma sicuramente ama trascorrere il tempo libero con le amiche. Viaggiare, fare lunghe passeggiate, scoprire nuove tradizioni e paesaggi, assaggiare i piatti tipici locali.

OBIETTIVI
È alla ricerca di nuovi itinerari che mettano in luce le bellezze inesplorate del territorio, e per trascorrere una mattinata in modo diverso in compagnia delle amiche.

⁶Parliamo quindi delle cosiddette Personas, un modello inferenziale, basate su previsioni, che generano meta-persone → Raccolte che clusterizzano una parte del nostro target (generalizzazione di un nostro possibile utente).

2.1.8 Valutazione dell'usabilità a priori

L'ingegneria dell'usabilità è un processo nel quale si specifica quantitativamente (e in anticipo) quali caratteristiche e in quale misura dovrà possedere il prodotto software finale. Questo processo è, poi, seguito dalla realizzazione effettiva del prodotto prototipato e dalla dimostrazione o validazione che esso possiede le caratteristiche pianificate.

Questa scritta sopra è la definizione di ingegneria dell'usabilità:

- ▶ La parola *ingegneria* sottolinea la presenza di un approccio basato su regole standard e fondamenti scientifici che ci permette di realizzare i requisiti richiesti in modo efficace ed efficiente;
- ▶ La parola *usabilità*, invece, indica che dobbiamo focalizzarci sul design dell'interfaccia utente del prodotto software da realizzare.

Quindi, per effettuare una valutazione completa dell'usabilità del nostro prodotto software, seguiremo due fasi complementari e assolutamente non sostituibili l'una con l'altra:

1. Una prima fase, detta di *valutazione euristica* o di *valutazione dell'usabilità a priori*, in cui verranno eseguite valutazioni da parte di esperti di usabilità, appartenenti al team di sviluppo del prodotto software, senza alcun coinvolgimento dei clienti e degli **stakeholders**;
2. Una seconda fase, detta *test dell'usabilità* o di *valutazione dell'usabilità sul campo*, in cui verranno eseguite valutazioni da parte di un campione di utenti, rappresentativo del target identificato al punto 2.1.7, in un ambiente controllato da uno o più osservatori (esperti di usabilità).

La prima fase viene effettuata durante le fasi di progettazione dell'interfaccia utente nell'ambiente di sviluppo, quando ancora non si ha a disposizione il prodotto finito, ma solo dei prototipi. In particolare, è possibile notare dalle seguenti tabelle quali sono le funzionalità che abbiamo implementato nei nostri prototipi e a quale livello di dettaglio.

Classificazione dei prototipi e delle implementazioni rispetto alla completezza funzionale (Tabella 1/2)

Funzionalità implementate						
Livello di dettaglio	Registrazione alla piattaforma	Autenticazione alla piattaforma	Cambio password	Effettuare ricerche di itinerari	Creare un nuovo itinerario	Visualizzare un itinerario
Mockup	X	X		X	X	X
Prototipo interattivo su Figma	X	X		X	X	X
Statecharts	X				X	
Prima implementazione	X	X		X	X	X
Beta testing	X	X		X	X	X
Applicazione finale	X	X	X	X	X	X

Classificazione dei prototipi e delle implementazioni rispetto alla completezza funzionale (Tabella 2/2)

Funzionalità implementate						
Livello di dettaglio	Inviare messaggi privati	Visualizzare messaggi privati	Visualizzare la schermata di profilo	Segnalare fotografie inappropriate	Visualizzare statistiche in tempo reale	Gestire le segnalazioni
Mockup	X	X	X		X	X
Prototipo interattivo su Figma	X	X	X		X	X
Statecharts						
Prima implementazione		X	X		X	
Beta testing	X	X	X		X	
Applicazione finale	X	X	X	X	X	

Classificazione dei prototipi e delle implementazioni rispetto alla completezza funzionale

Gli esperti dell'usabilità, quindi, valuteranno i nostri prototipi forniti in fase di progettazione analizzando il comportamento complessivo del sistema (specificato tramite statecharts al punto 2.1.10) e verificando la loro conformità ad alcune *regole d'oro*, come le famose 8 regole d'oro di Ben Shneiderman o le altrettanto importanti 10 euristiche di Nielsen.

Di seguito verranno elencate alcune di queste regole, seguite da una possibile risposta degli esperti di usabilità (verranno valutati solo i due casi d'uso significativi identificati al punto 2.1.2):

1. Valutazione del primo caso d'uso: l'utente già registrato crea un nuovo itinerario

► **Interfaccia consistente e standard**

La presenza di elementi già visti in altre interfacce utente, come le frecce che puntano verso sinistra per indicare la possibilità all'utente di tornare indietro, rendono l'esperienza generale più semplice. Inoltre, i colori sono utilizzati egregiamente - il verde indica la conferma e il rosso indica i messaggi di errore.

► **Aiutare gli utenti a diagnosticare e correggere gli errori**

I messaggi di errore mostrati a seguito della scelta di un tipo di file con un'estensione non corretta (nelle schermate di selezione di un file GPX o di un'immagine) sono scritti con un linguaggio naturale ed indicano con precisione la soluzione che deve seguire l'utente per procedere correttamente.

► **Prevenzione degli errori**

È ottima la soluzione di impedire all'utente di procedere nella creazione di un nuovo itinerario se alcuni dei campi richiesti non sono stati riempiti. In questo modo, si elimina il problema di un errore di creazione dell'itinerario alla radice.

► **Flessibilità ed efficienza d'uso**

Purtroppo, una grave insufficienza notata durante questa valutazione è la mancanza di shortcuts che facilitino l'utilizzo del prodotto software ad utenti più esperti. Una possibile soluzione potrebbe essere quella di aggiungere campi con auto completamento nel form di creazione dell'itinerario.

2. Valutazione del secondo caso d'uso: l'utente non registrato si registra alla piattaforma tramite email

► **Libertà e controllo da parte degli utenti**

Dato che gli utenti meno esperti potrebbero cliccare o toccare componenti del sistema per errore, è stata molto gradita la scelta di posizionare sia nella schermata di *Login* che nella schermata di *Registrazione* la possibilità di tornare indietro, rispettivamente, alla schermata di *Registrazione* o *Login*.

► **Corrispondenza fra mondo reale e sistema**

I popup (sia di errore che non) che compaiono durante la registrazione dell'utente contengono dei messaggi chiari ed auto esplicativi, con parole, frasi e concetti familiari al sistema. Questo è un requisito non funzionale molto apprezzato in qualsiasi interfaccia utente.

► **Design minimalista ed estetico**

Siccome questo prodotto software è dedicato principalmente a persone appassionate di passeggiate e trekking, la scelta di colori del background rappresentanti la natura, come il celeste o il verde, è conforme all'ambito applicativo dei requisiti del sistema.

► **Visibilità dello stato del sistema**

Il sistema dovrebbe sempre informare gli utenti su ciò che sta accadendo. Il fatto che, facendo un esempio specifico, l'utente non viene avvisato in anticipo che è richiesta una password sicura di 8 caratteri abbassa di molto il grado di valutazione dell'usabilità. Una possibile soluzione potrebbe essere quella di introdurre una lista dei vincoli da rispettare nella scelta della password.

La valutazione dell'usabilità a priori termina con la realizzazione di un diagramma nel quale:

- ▶ Sull'asse verticale sono riportati i valutatori;
- ▶ Sull'asse orizzontale sono riportati i problemi di usabilità riscontrati durante la valutazione, ordinati secondo la facilità di essere individuati (più sono a destra, più sono facili da individuare);
- ▶ L'intersezione tra un punto dell'asse orizzontale e uno dell'asse verticale è un quadratino nero. Indica che un valutatore ha riscontrato un problema.

Diagramma 1. Risultato della valutazione dei prototipi annessi al primo caso d'uso

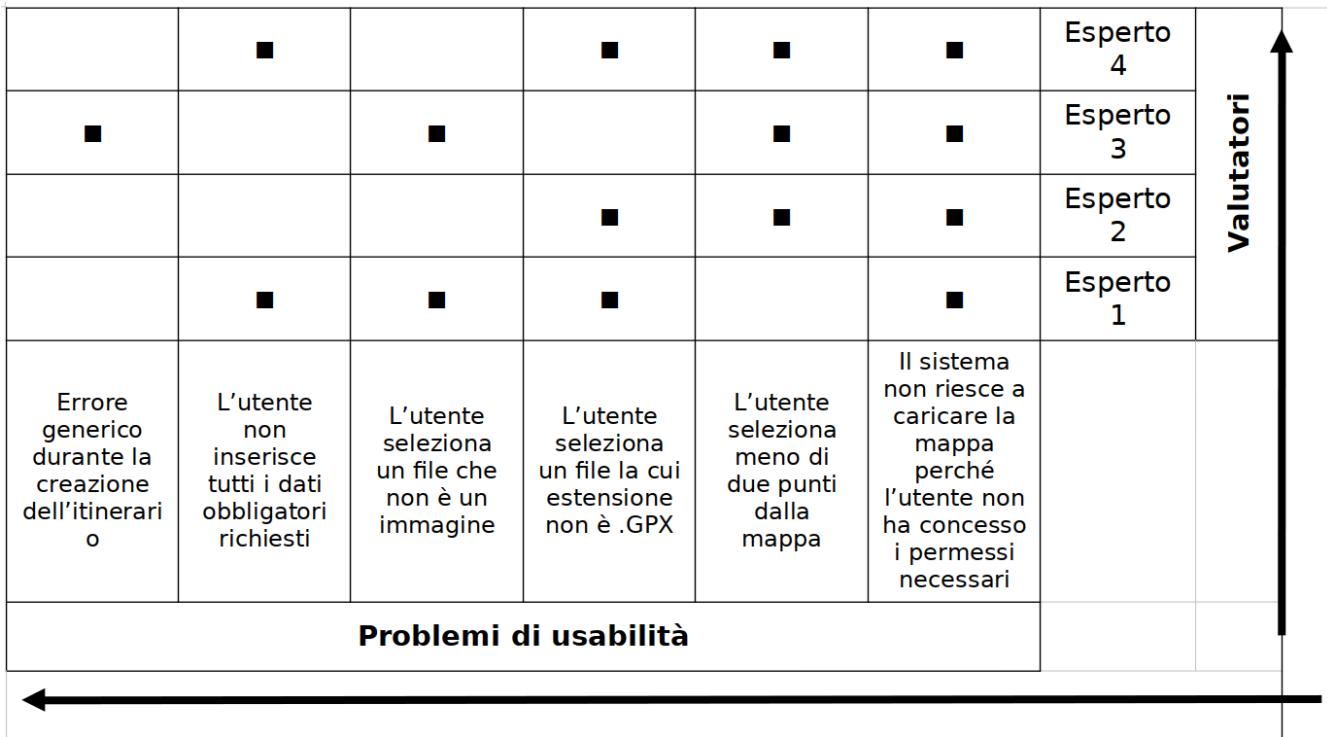
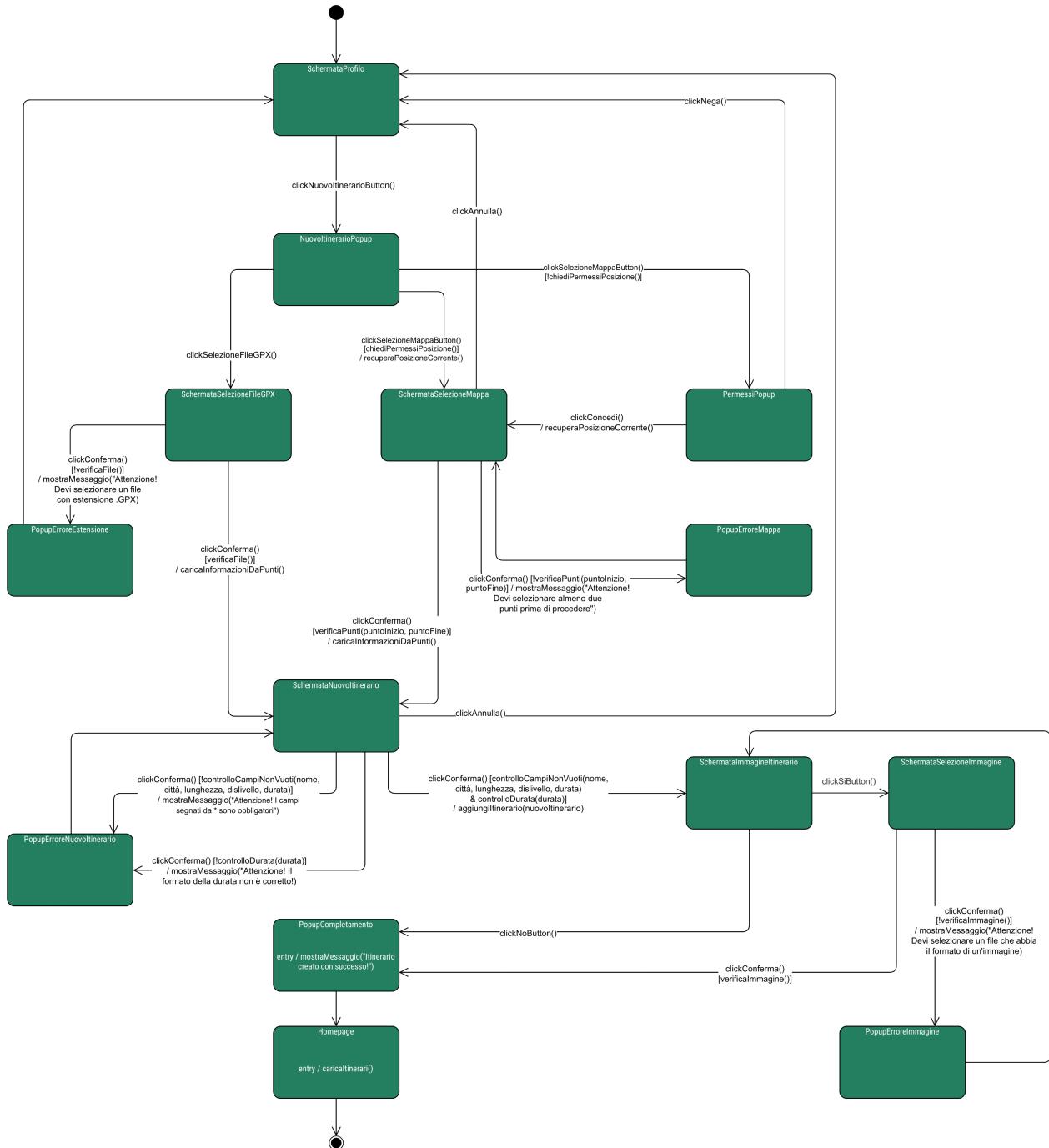


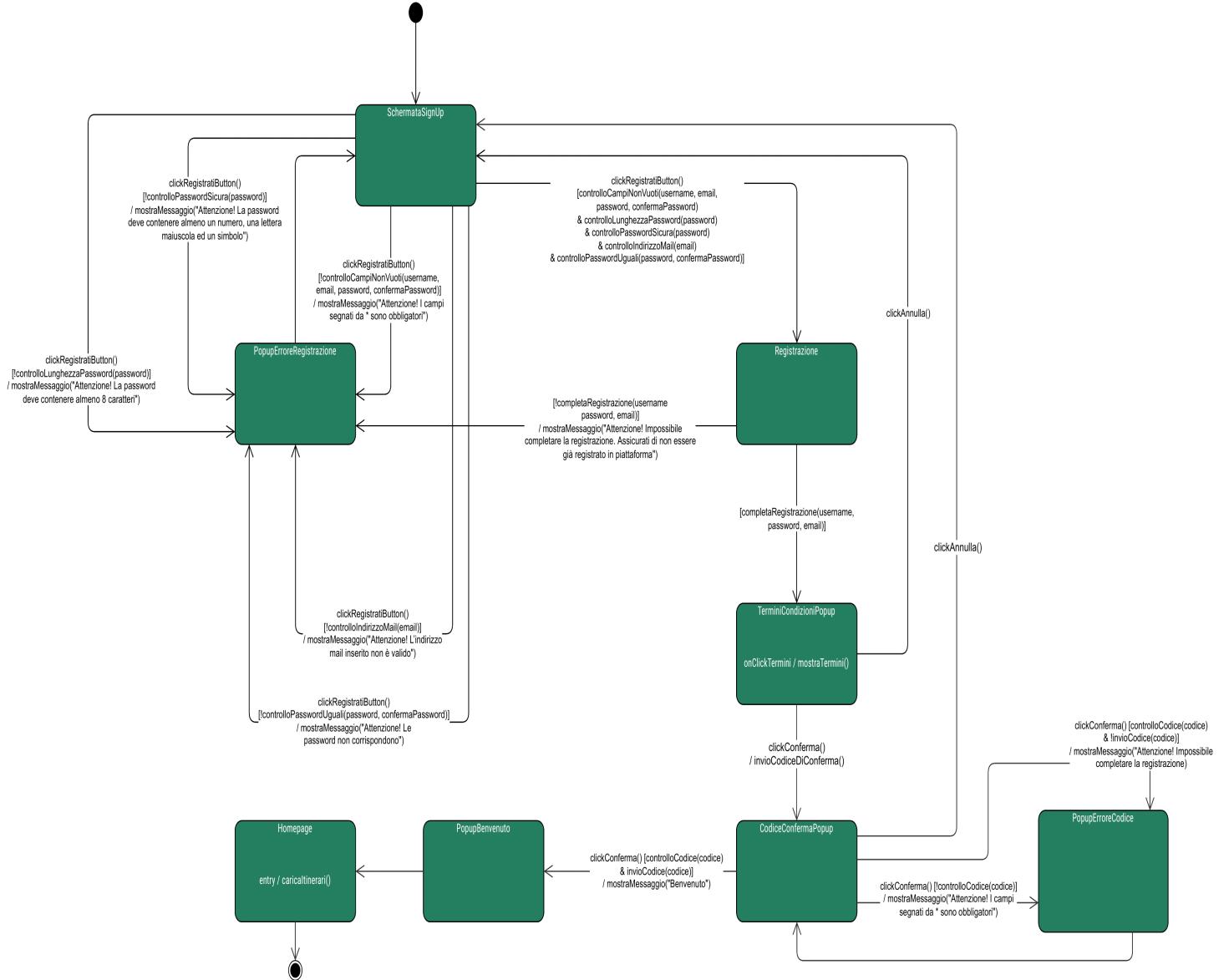
Diagramma 2. Risultato della valutazione dei prototipi annessi al secondo caso d'uso

		■	■	■	■	Esperto 4	Valutatori
■	■		■		■	Esperto 3	
	■		■	■	■	Esperto 2	
		■		■	■	Esperto 1	
Errore generico durante il completamento della registrazione	L'utente non inserisce tutti i dati obbligatori richiesti	L'utente inserisce la password di conferma diversa dalla password	L'utente inserisce un indirizzo mail non valido	L'utente inserisce una password con meno di 8 caratteri	L'utente inserisce una password poco sicura	Problemi di usabilità	

2.1.9 Prototipazione funzionale via Statechart dell'interfaccia grafica

Di seguito vengono riportati i diagrammi **Statechart** prodotti durante lo sviluppo della piattaforma **NaTour**, che rappresentano l'interazione dell'utente con l'interfaccia grafica:





2.1.10 Glossario

- ▶ **Prototipo** → Un prototipo è il progetto astratto di un sistema che realizza una o più funzionalità del prodotto finito e può avere diversi gradi di dettaglio: dagli schizzi su carta, ai mockup statici e mockup funzionali fino all'interfaccia finale pronta per la fase di beta testing e rilascio.
- ▶ **Principi SOLID** → Sono delle linee guida allo sviluppo di software orientati agli oggetti che aumentano il livello della qualità interna di un prodotto software descritte da *Robert C. Martin* agli inizi degli anni 2000.
- ▶ **Target** → Il termine *target* è adottato dalla lingua inglese e significa letteralmente *bersaglio*. In questo contesto rappresenta l'insieme dei clienti e delle persone da raggiungere nelle nostre azioni di comunicazione e di marketing, ovvero il mercato di destinazione composto dai potenziali destinatari della nostra applicazione, NaTour.
- ▶ **Escursionismo** → È un'attività motoria che si svolge all'aperto ed è basata sul camminare. Normalmente viene esercitata lungo sentieri e percorsi di diverso grado di difficoltà. Normalmente l'escursionismo si svolge in montagna o comunque in un ambiente naturale, e può essere accompagnata da attività naturalistiche quali l'osservazione degli uccelli, l'osservazione di specie botaniche o anche la visita di monumenti o fenomeni naturali posti lungo il percorso.
- ▶ **Social Network** → Con l'espressione social network si identifica un servizio Internet che permette la realizzazione e la gestione di reti sociali virtuali. Il social network è un servizio di rete sociale nonché una piattaforma web che consente la comunicazione fra rete e utenti. Attraverso questa comunicazione si possono scambiare informazioni, messaggi, contenuti e tanto altro attraverso vari canali.
- ▶ **Trekking** → Il trekking è uno sport accessibile a tutti, da praticare sia da soli che in compagnia, partecipando ad escursioni e/o visite guidate con gruppi di appassionati. È un'attività sportiva a metà tra l'arrampicata e l'escursionismo, ideale per chi ama passeggiare in mezzo alla natura e molto diffusa sia in Italia che in Europa e nel mondo. Il termine deriva dal verbo inglese to trek, che può essere tradotto in italiano con "camminare lentamente" o "viaggiare a lungo".
- ▶ **Natourers** → Sono gli utenti registrati alla piattaforma NaTour.
- ▶ **Amministratore NaTour** → Nel dominio dell'applicazione, ci si può riferire a questa figura anche come *admin*. Si tratta delle persone incaricate di provvedere alla gestione della piattaforma NaTour.
- ▶ **UX Design (User Experience) / UI Design (User Interface)** → Per *UX Design* si intende l'insieme dei processi rivolti ad aumentare la soddisfazione degli utenti migliorando e rendendo più intuitiva la navigazione in un sistema software. Invece, per *UI Design*, si intende l'insieme dei processi rivolti ad aumentare la soddisfazione degli utenti migliorando il modo in cui un prodotto o servizio si presenta, rendendolo più accattivante e invitante. Entrambe, quindi, riguardano, l'interazione degli utenti con un prodotto o servizio.
- ▶ **Itinerario** → Il percorso che si segue o s'intende seguire in un viaggio, comprendente per lo più un certo numero di tappe. Rappresentazione grafica o descrizione scritta di un percorso. Nel dominio di questa applicazione viene utilizzato come sinonimo del termine *escursione*.
- ▶ **Username** → È un insieme distintivo di caratteri alfabetici e/o numerici usati per identificare un Natourer e ottenere l'accesso sulla piattaforma NaTour.
- ▶ **Email** → Con questo termine si intende l'indirizzo di posta elettronica che l'utente inserisce in fase di registrazione. Viene utilizzato per l'invio di alcuni codici di conferma (ad esempio per la conferma dell'account o il cambio password).
- ▶ **Password** → È una sequenza di caratteri alfanumerici e di simboli che l'utente non registrato associa al proprio username in fase di registrazione alla piattaforma NaTour. Viene utilizzata per identificare ed autenticare in modo esclusivo un utente e permettergli l'accesso alla piattaforma NaTour.

- ▶ **Account Google** → Con account si indica l'insieme di funzionalità, di strumenti e di contenuti attribuiti ad un particolare coppia <username, password> che identifica un singolo utente su una piattaforma. Per account Google, quindi, si intende l'account privato di un utente che viene utilizzato, nel contesto della nostra applicazione, per effettuare il single-sign-on.
- ▶ **Single-Sign-On** → Può essere tradotto in italiano come autenticazione o identificazione unica. Indica la proprietà di una piattaforma di consentire ad un utente di effettuare un'unica autenticazione per accedere alle risorse informatiche di più sistemi software.
- ▶ **Disabilità** → La disabilità può essere definita come la condizione personale di chi ha una ridotta capacità d'interazione con l'ambiente sociale rispetto a ciò che è considerata la norma. Per questo motivo è meno autonomo nello svolgere le attività quotidiane e spesso in condizioni di svantaggio nel partecipare alla vita sociale.
- ▶ **Nome itinerario** → Si indica una sequenza di caratteri che identifica un itinerario nella piattaforma NaTour.
- ▶ **Durata itinerario** → Si indica il tempo di svolgimento di un percorso a più tappe.
- ▶ **Autore itinerario** → Si indica il Natourer che ha ideato, creato e stabilito il punto di inizio e di fine di un percorso e tutte le altre sue caratteristiche.
- ▶ **Livello di difficoltà** → Si intende la valutazione della difficoltà, cioè la scala di difficoltà tecnica indicata nelle nostre escursioni, che prevede cinque livelli di difficoltà: facile (per tutti), non molto facile, medio, difficile e molto difficile (per escursionisti esperti). I diversi livelli di difficoltà tengono conto anche dell'impegno **fisico/atletico**, della lunghezza, del dislivello, della durata e dell'accessibilità per disabili. In questo modo si potrà scegliere il percorso che meglio si adegua all'esperienza e preparazione fisica di ogni persona.
- ▶ **Lunghezza itinerario** → Si intende la lunghezza del percorso di un itinerario. Può essere calcolata come distanza tra le coordinate geografiche del punto di fine e il punto di inizio oppure tracciando il percorso completo.
- ▶ **Dislivello itinerario** → Si intende la differenza delle altitudini calcolate nel punto di fine e nel punto di inizio del percorso di un itinerario.
- ▶ **Zona geografica** → Una zona geografica è una porzione continua della superficie terrestre la cui estensione è caratterizzata da un'ampiezza non inferiore a qualche chilometro quadrato, distinta per caratteristiche proprie che possono variare dalla conformazione del terreno alla posizione geografica, al clima, alla fauna, alla flora, fino a caratteristiche dipendenti dall'azione dell'uomo, cioè cultura, storia e lingua. Nella nostra piattaforma NaTour distinguiamo cinque tipi di zone geografiche: collina, montagna, mare, città e pianura.
- ▶ **Coordinate geografiche** (latitudine e longitudine) → Sono valori misurati in gradi utili per individuare con precisione la posizione di un punto sulla superficie terrestre. La latitudine è la distanza angolare di un punto dall'equatore e la longitudine è la distanza angolare di un punto da un arbitrario meridiano di riferimento lungo lo stesso parallelo del luogo.
- ▶ **Upload** → Operazione consistente nel trasferire un file dal proprio dispositivo a un dispositivo remoto attraverso una rete.

2.2 Modelli di Dominio

2.2.1 Classi, oggetti e relazioni di analisi

Di seguito vengono riportati i diagrammi delle classi di Analisi prodotti durante lo sviluppo della piattaforma NaTour:

Diagramma di classi di Analisi inerente al requisito funzionale n° 1

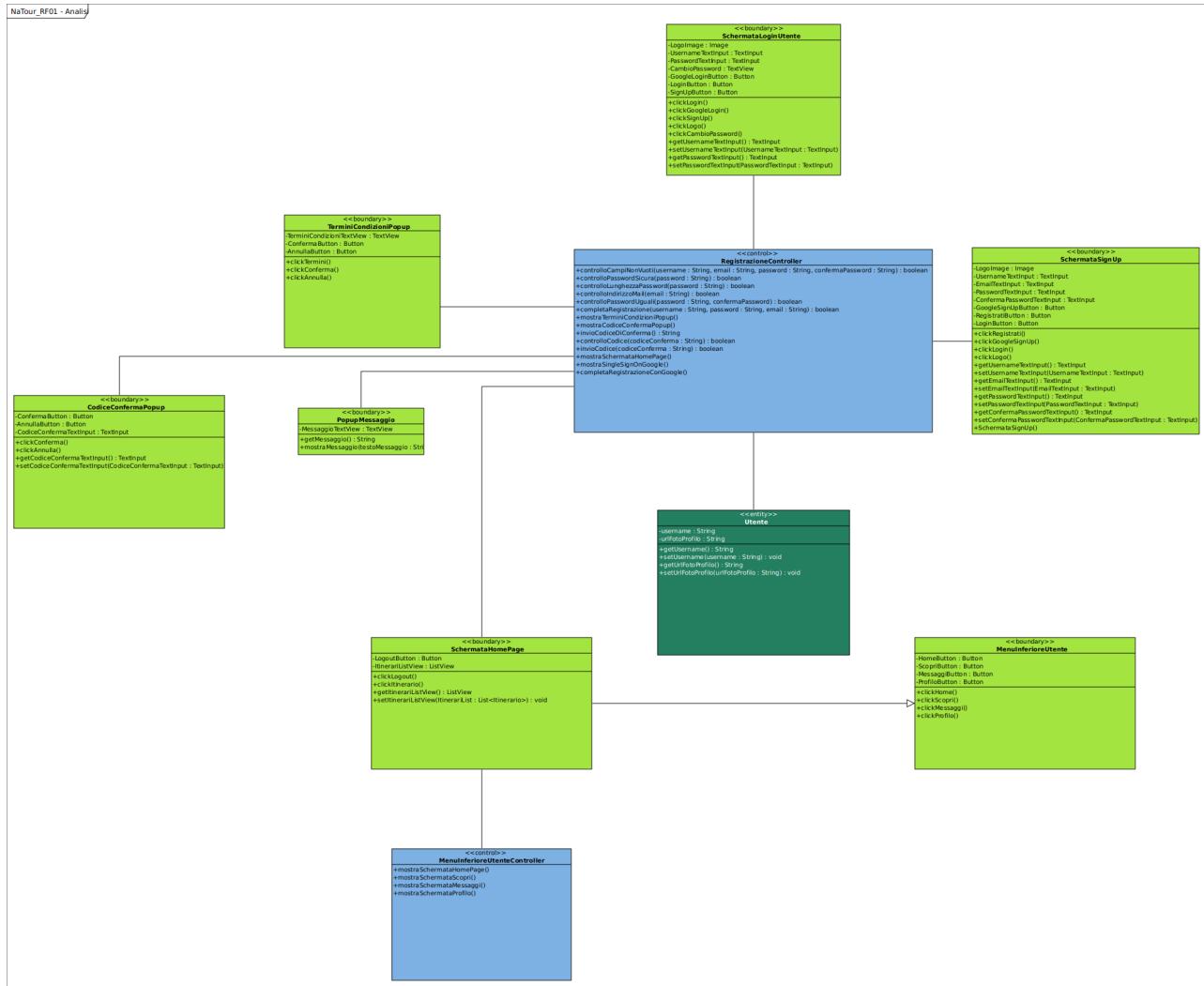


Diagramma di classi di Analisi inerenti ai *requisiti funzionali* n° 2-3

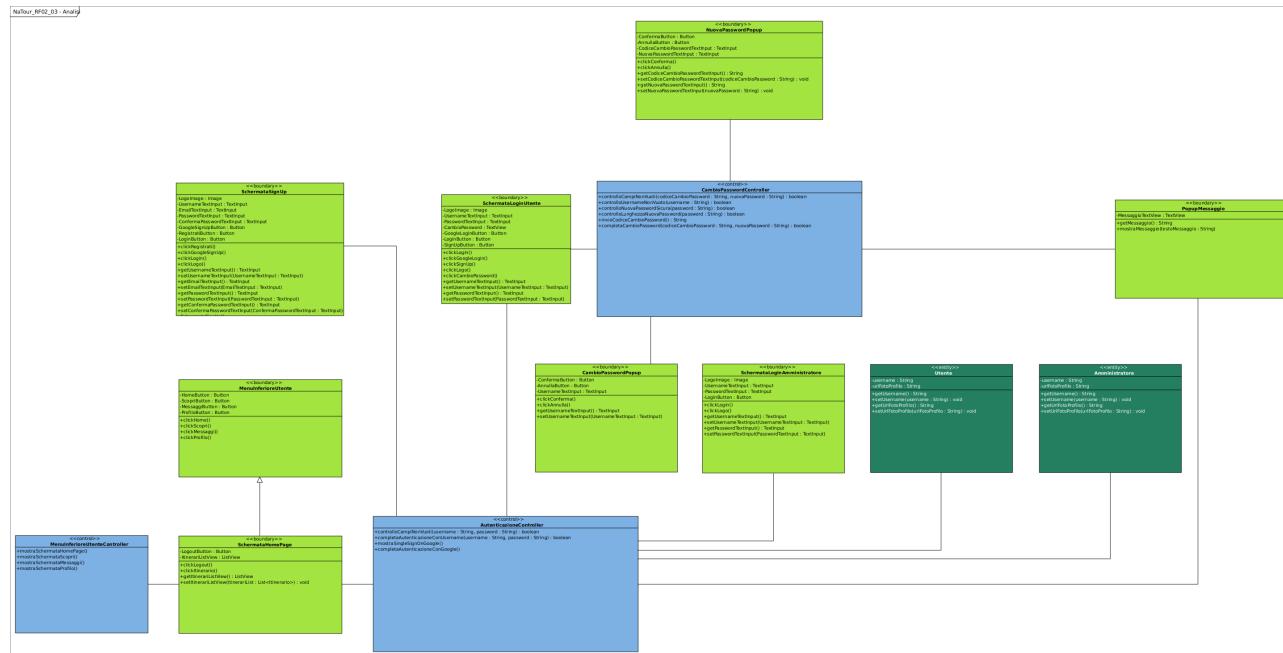


Diagramma di classi di Analisi inerente al *requisito funzionale* n° 4

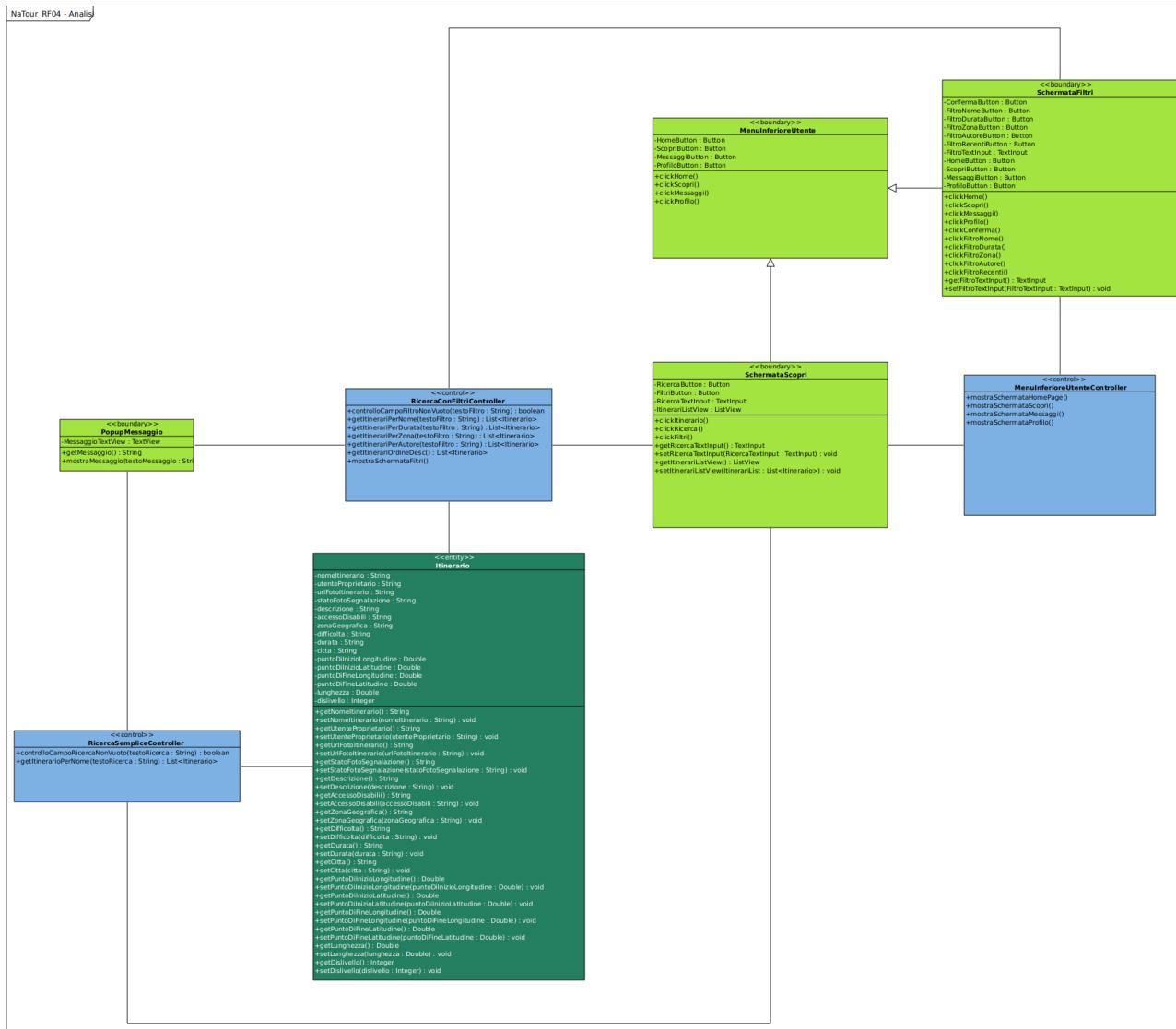


Diagramma di classi di Analisi inerenti ai requisiti funzionali n° 5-9

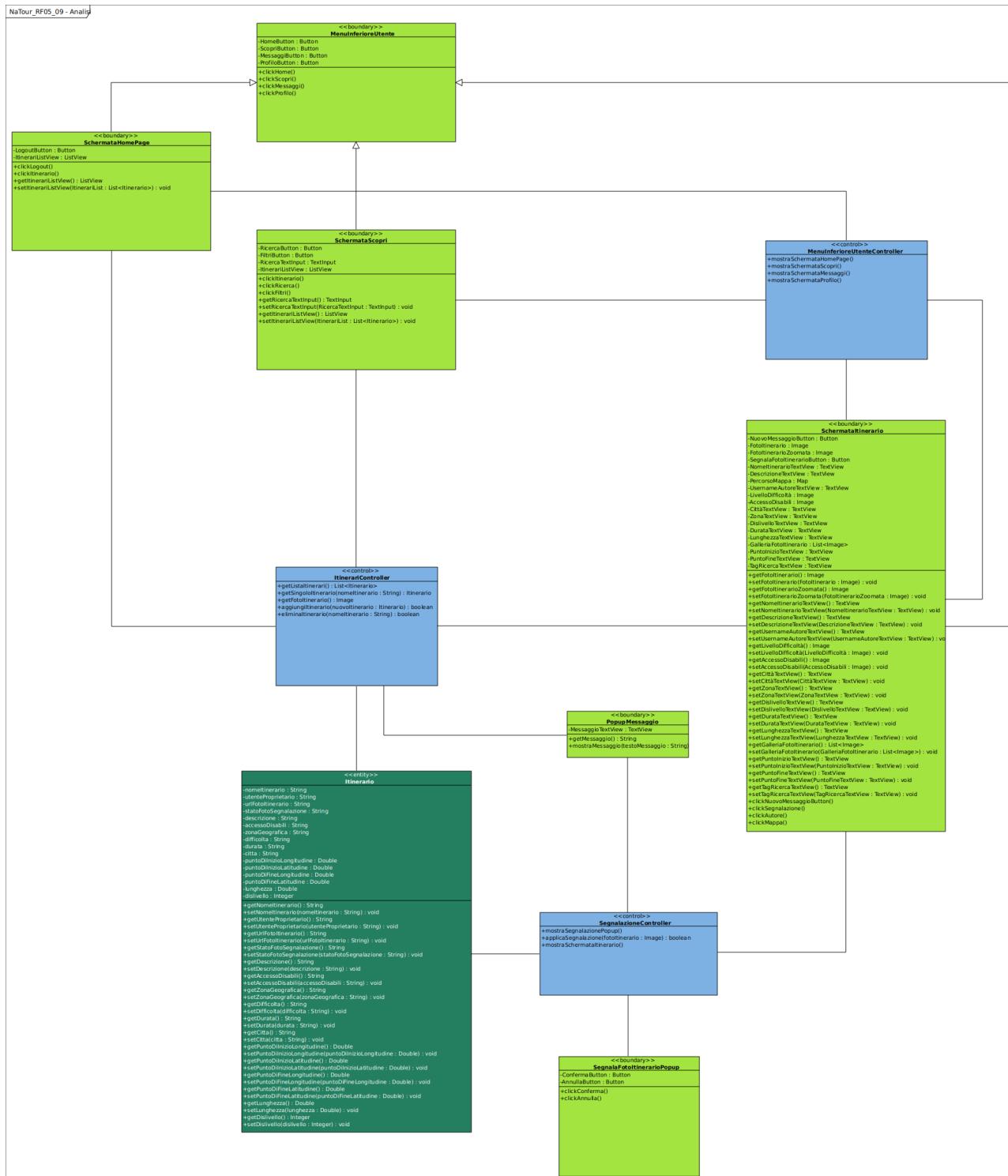


Diagramma di classi di Analisi inerente al *requisito funzionale* n° 6

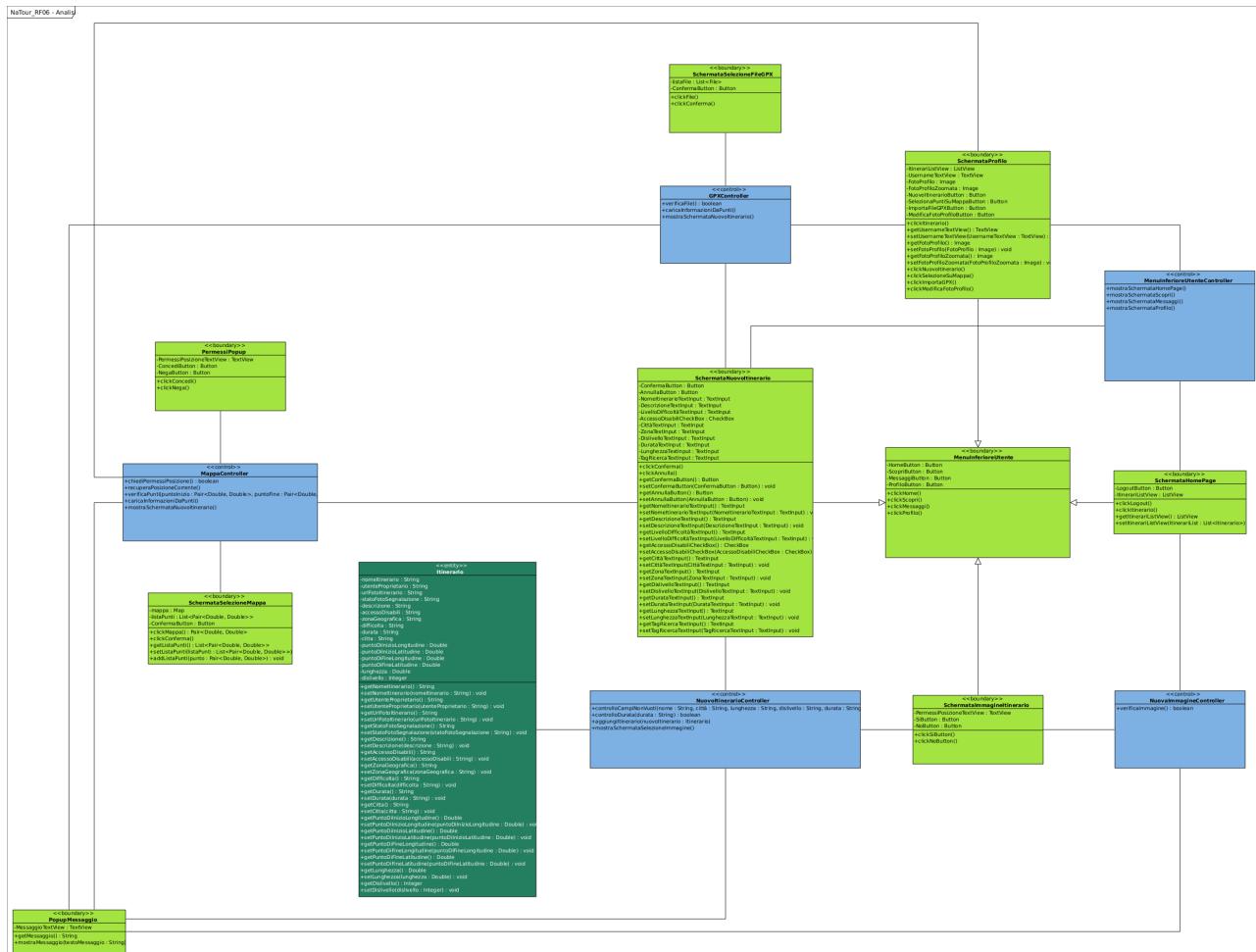


Diagramma di classi di Analisi inerenti al requisiti funzionali n° 7-8

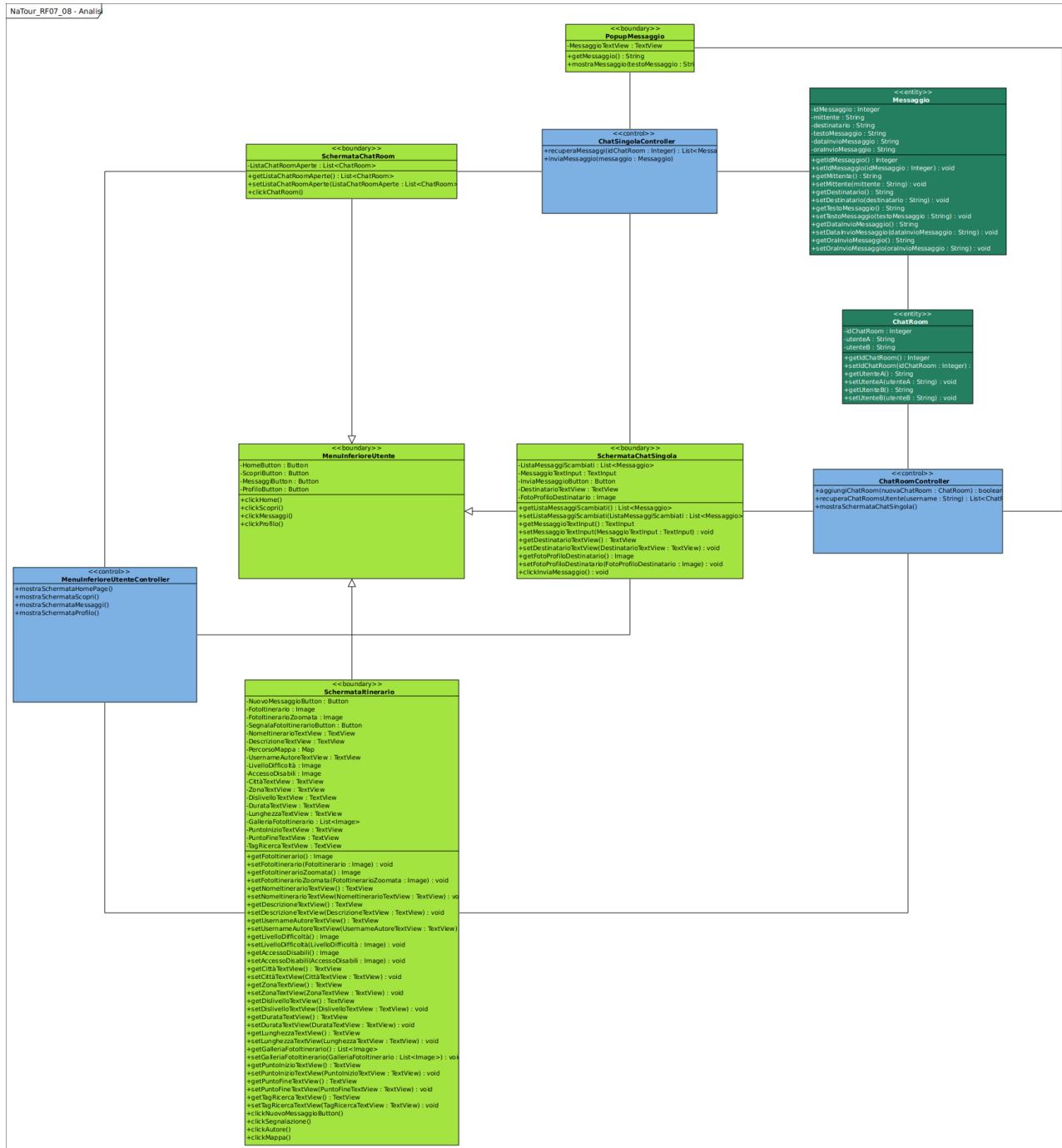


Diagramma di classi di Analisi inerente al *requisito funzionale n° 10*

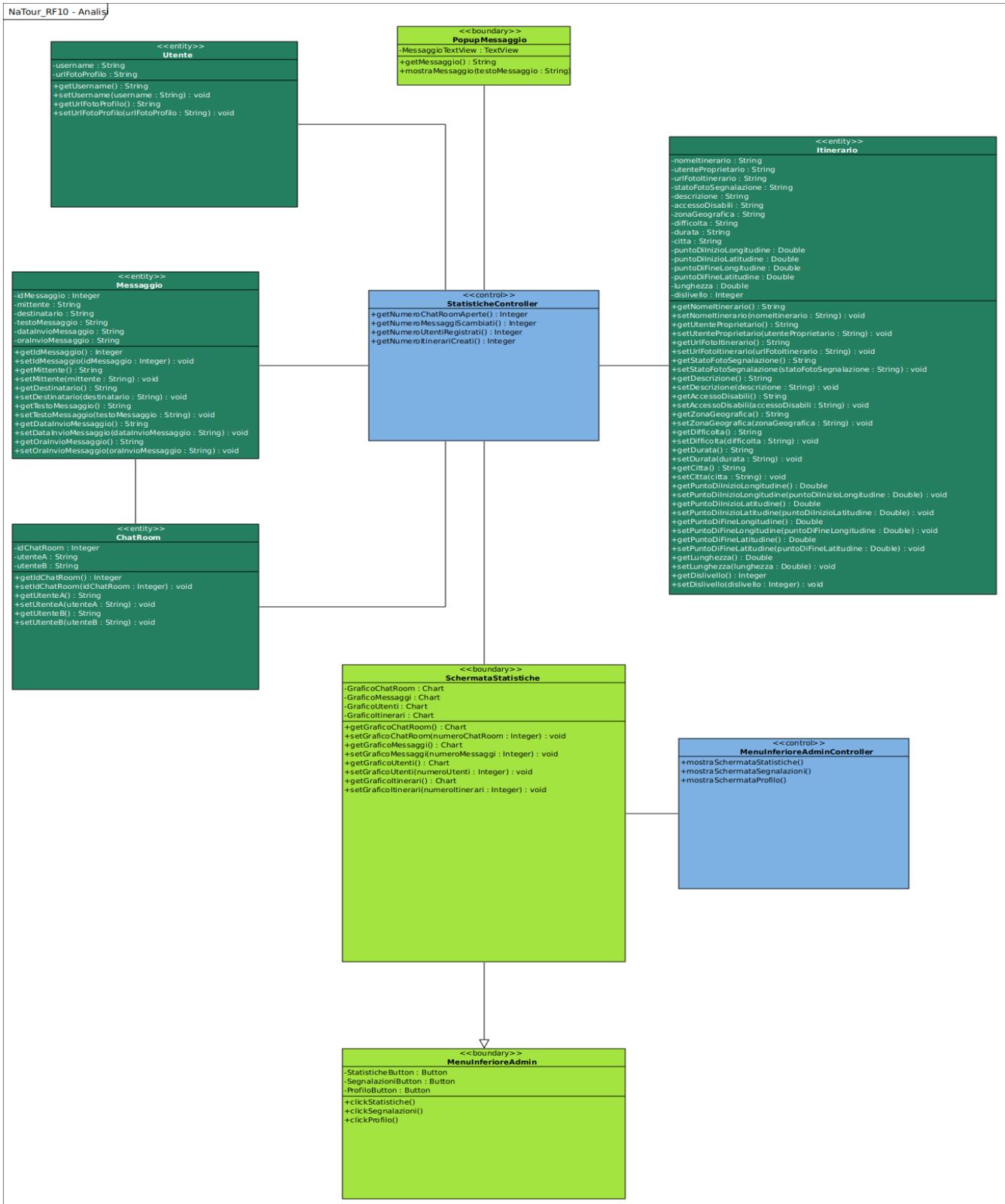
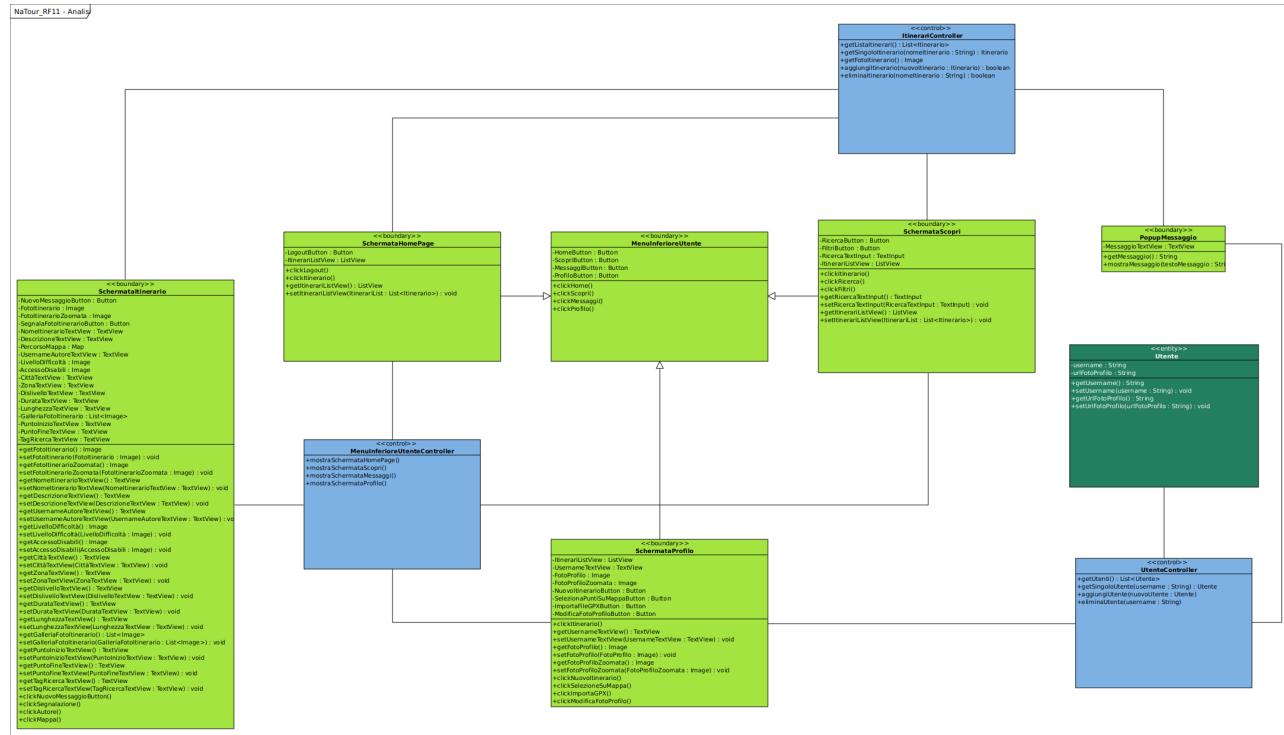


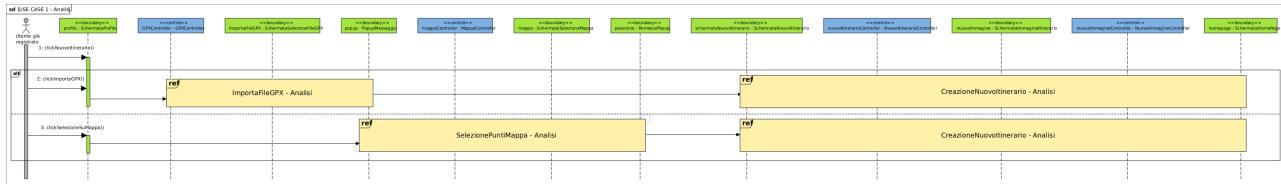
Diagramma di classi di Analisi inerente al *requisito funzionale* n° 11



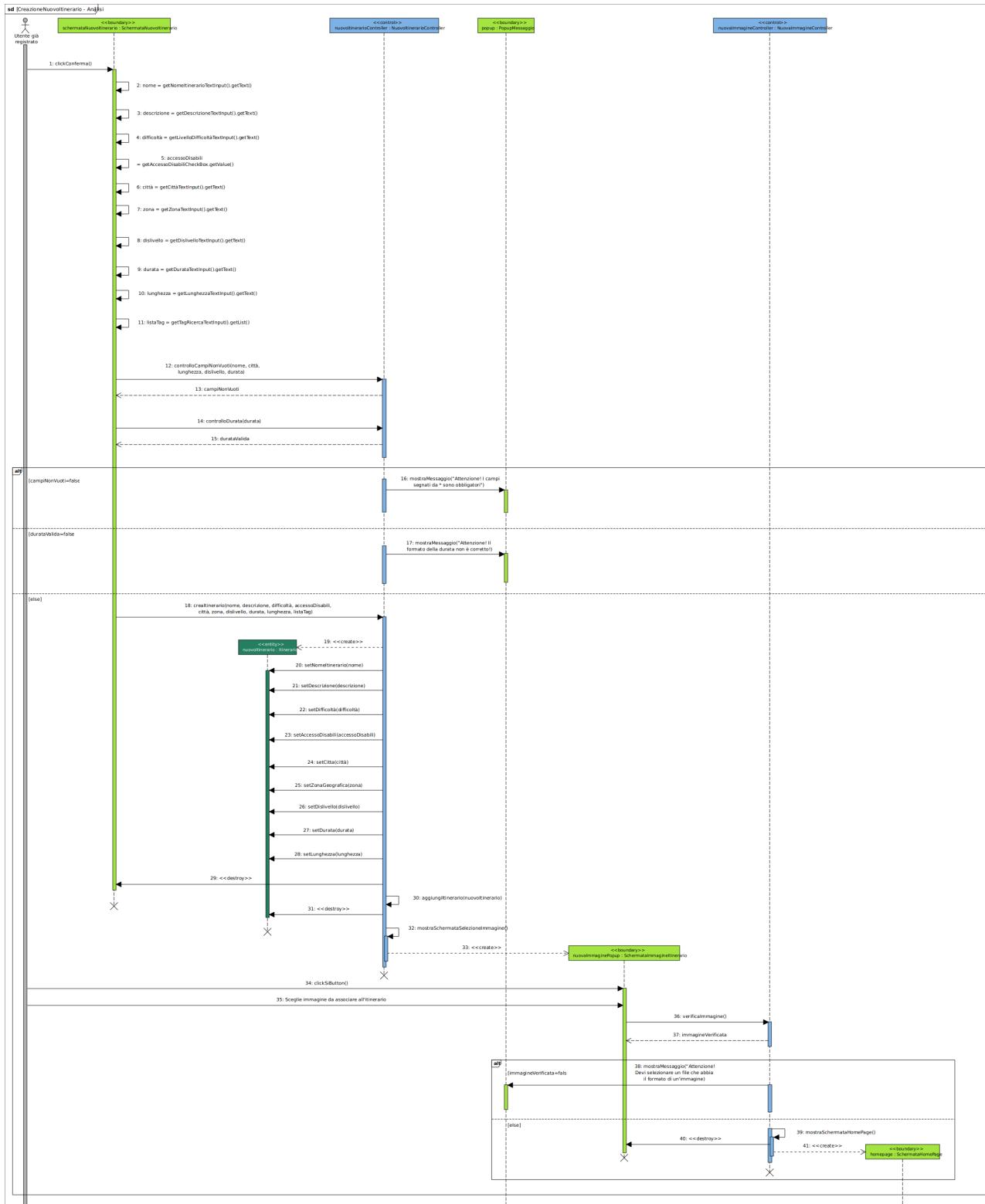
2.2.2 Diagrammi di sequenza di analisi (per 2 casi significativi)

I Sequence Diagram⁷ di seguito riportati sono da supporto alla rappresentazione grafica dei casi d'uso precedentemente descritti

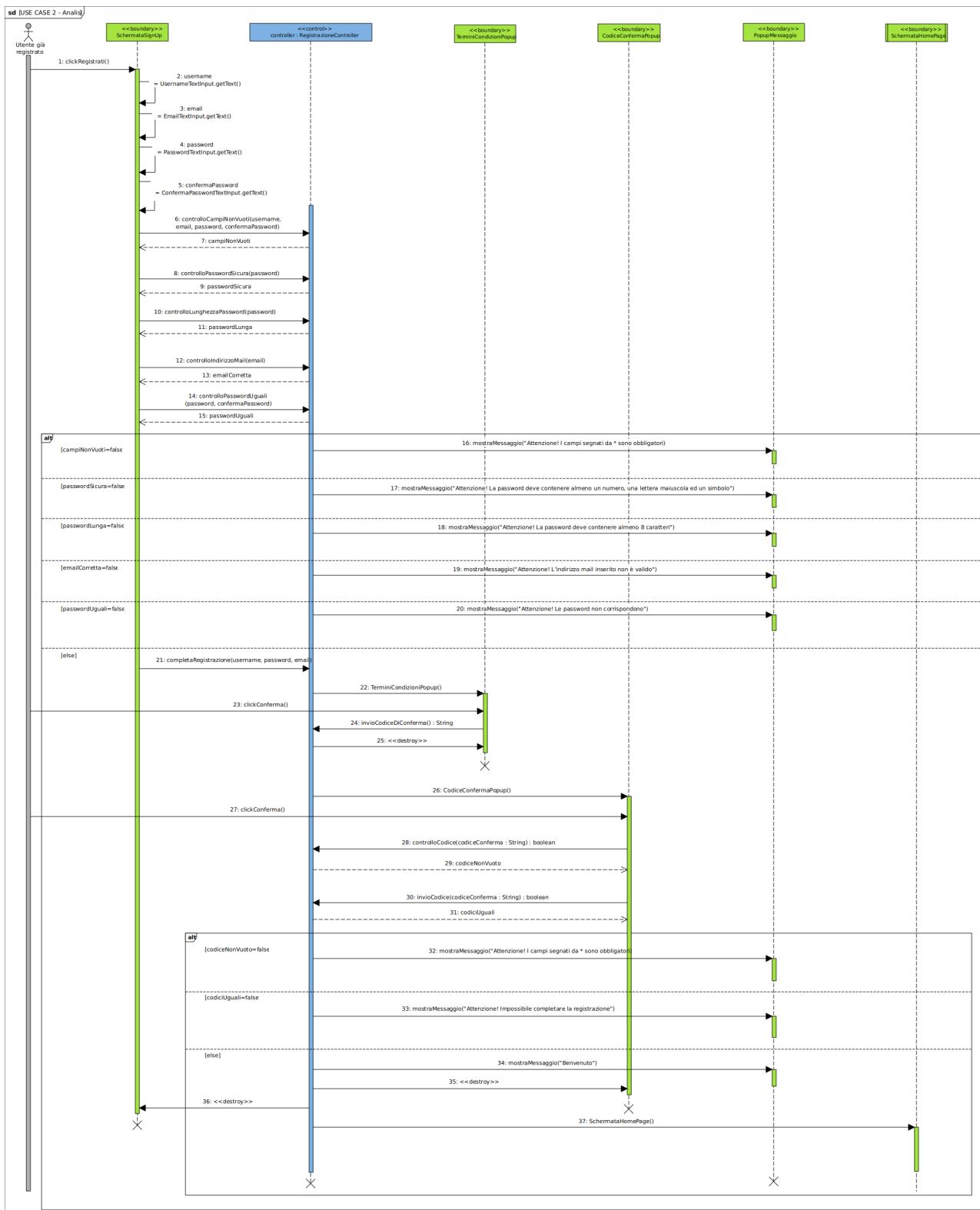
Di seguito vengono quindi riportati i diagrammi di sequenza relativi allo **use case diagram n° 1:**



⁷ Descrivono la collaborazione di un gruppo di oggetti che devono implementare collettivamente un comportamento.



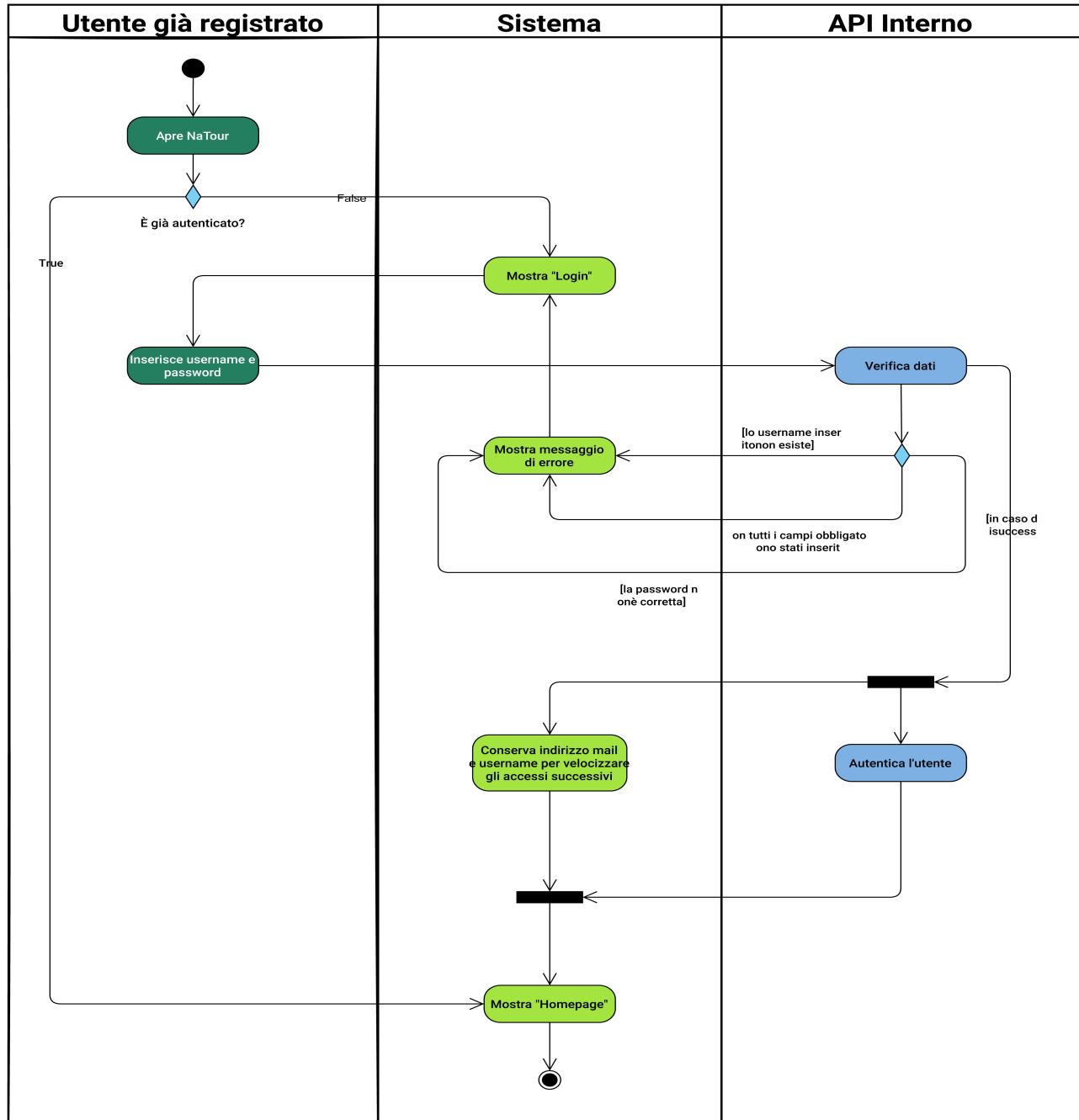
E allo use case diagram n° 2



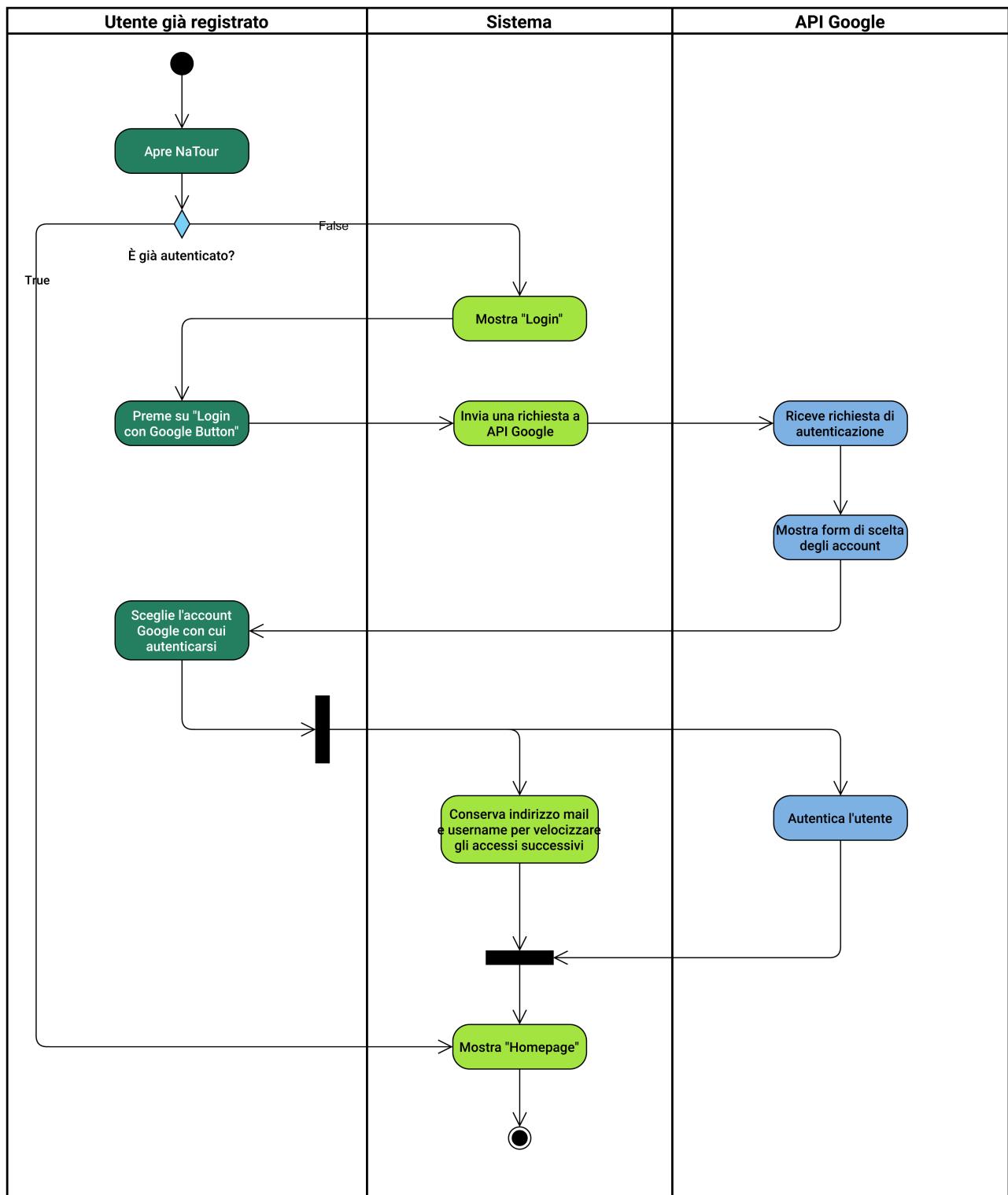
2.2.3 Diagrammi di attività

L'ultimo dei documenti UML di analisi dei requisiti che riportiamo è caratterizzato dai *Diagrammi di attività*,⁸ che hanno lo scopo di far meglio capire il flusso di esecuzione di ogni singola funzionalità offerta dal sistema.

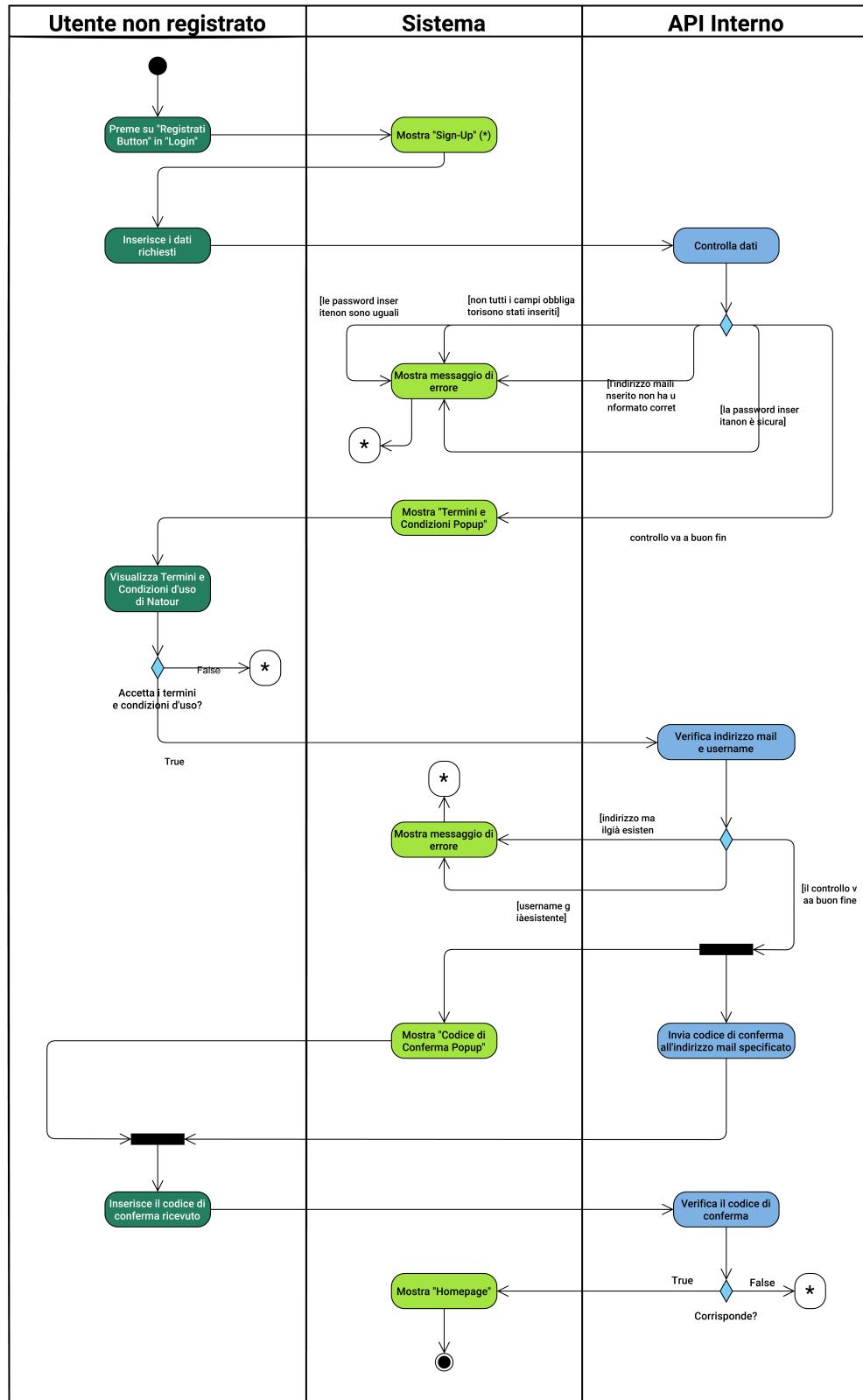
Activity diagram inerenti al *requisito funzionale n° 1*

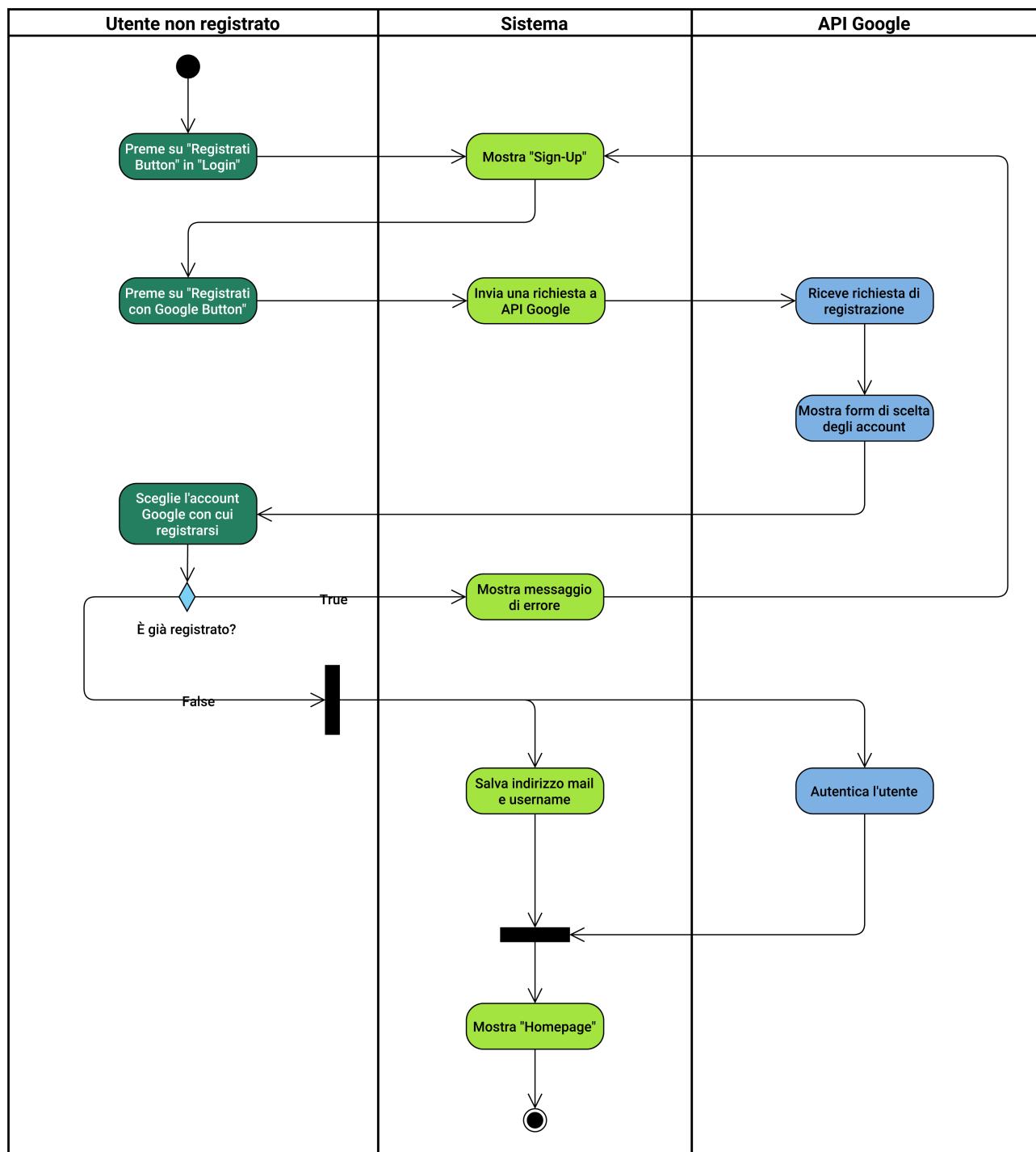


⁸ Mostrano le attività e i cambiamenti da un' attività all' altra con gli eventi che accadono in qualche parte del sistema.

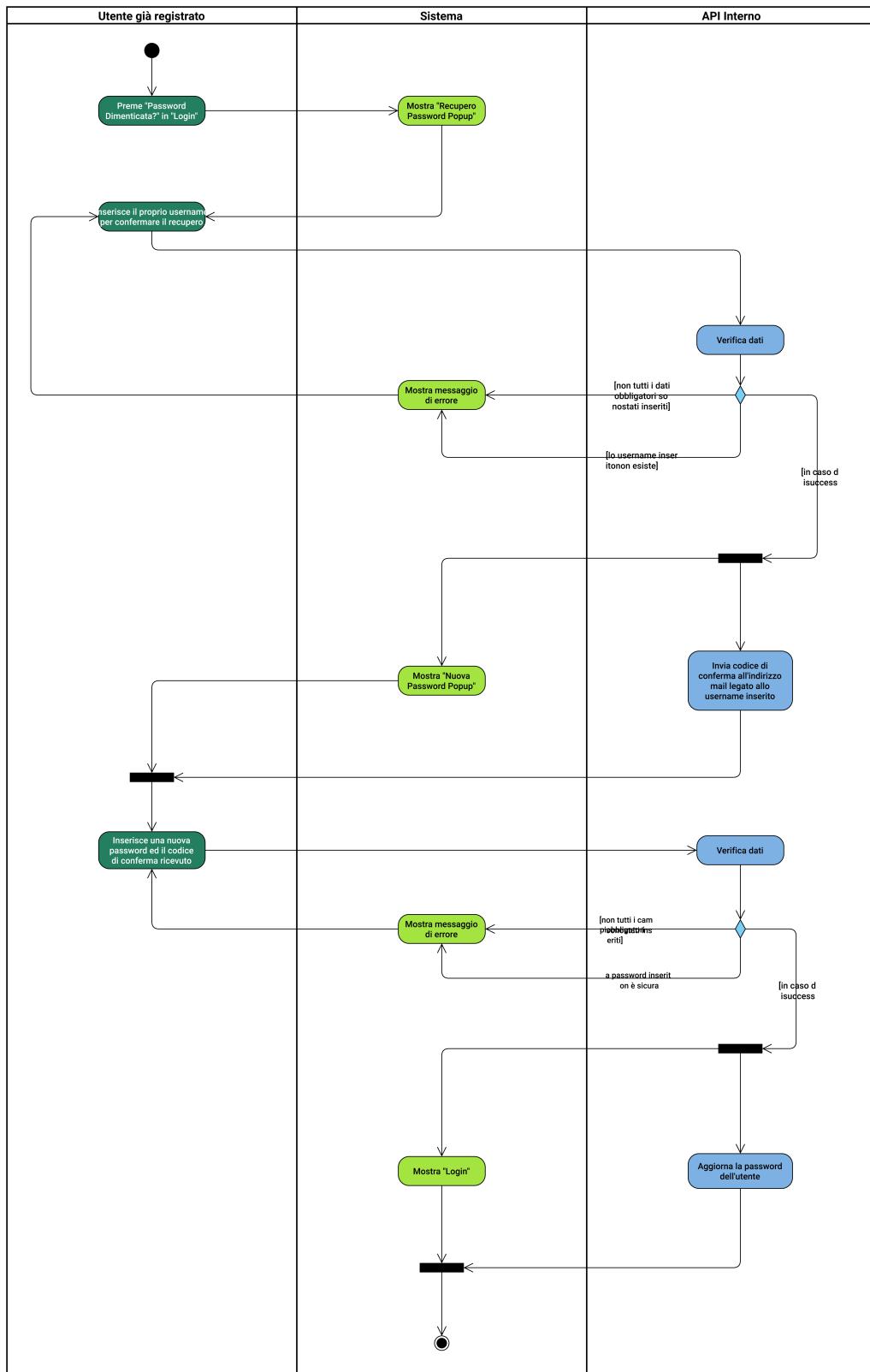


Activity diagram inerenti al *requisito funzionale* n° 2

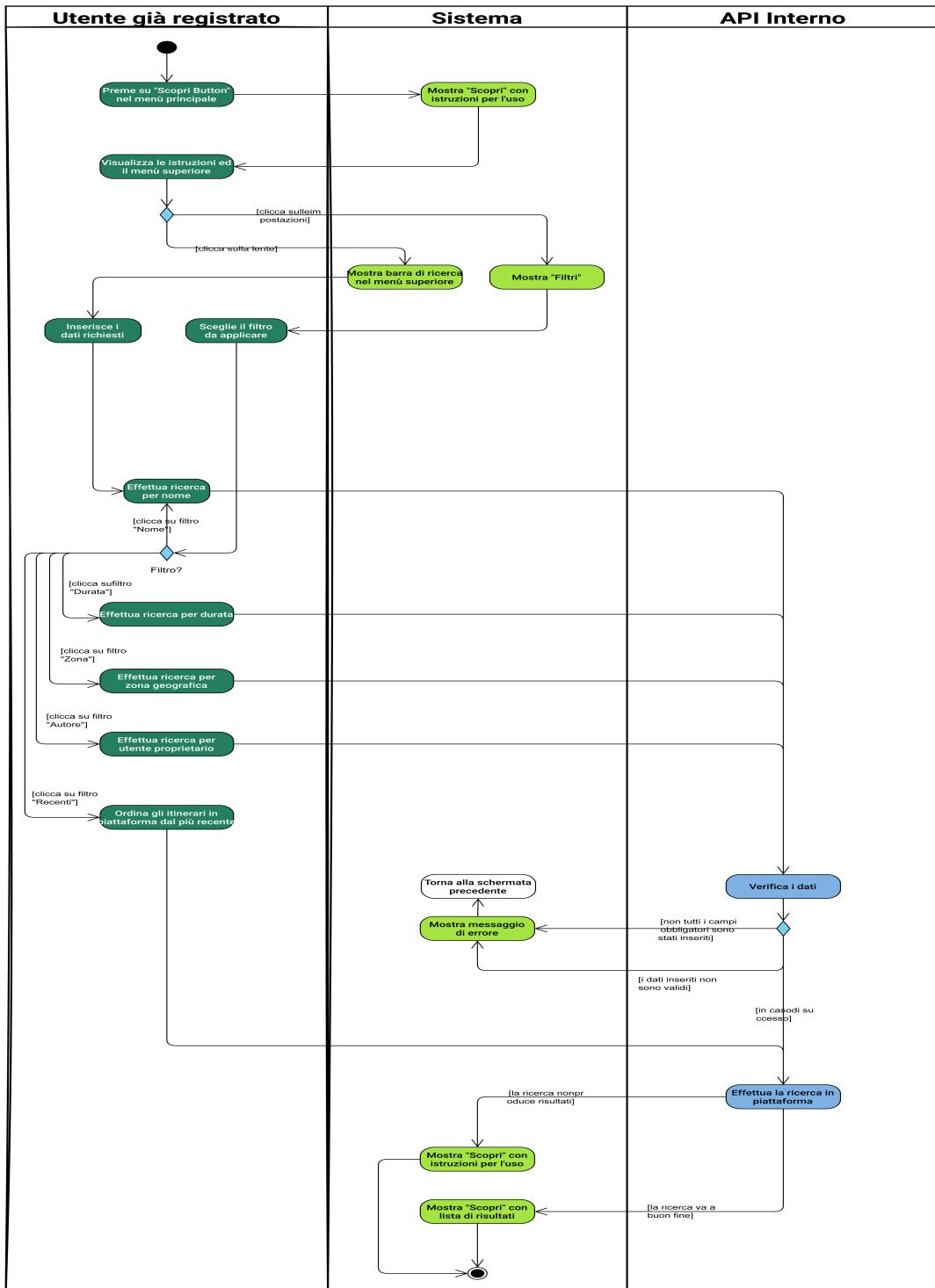




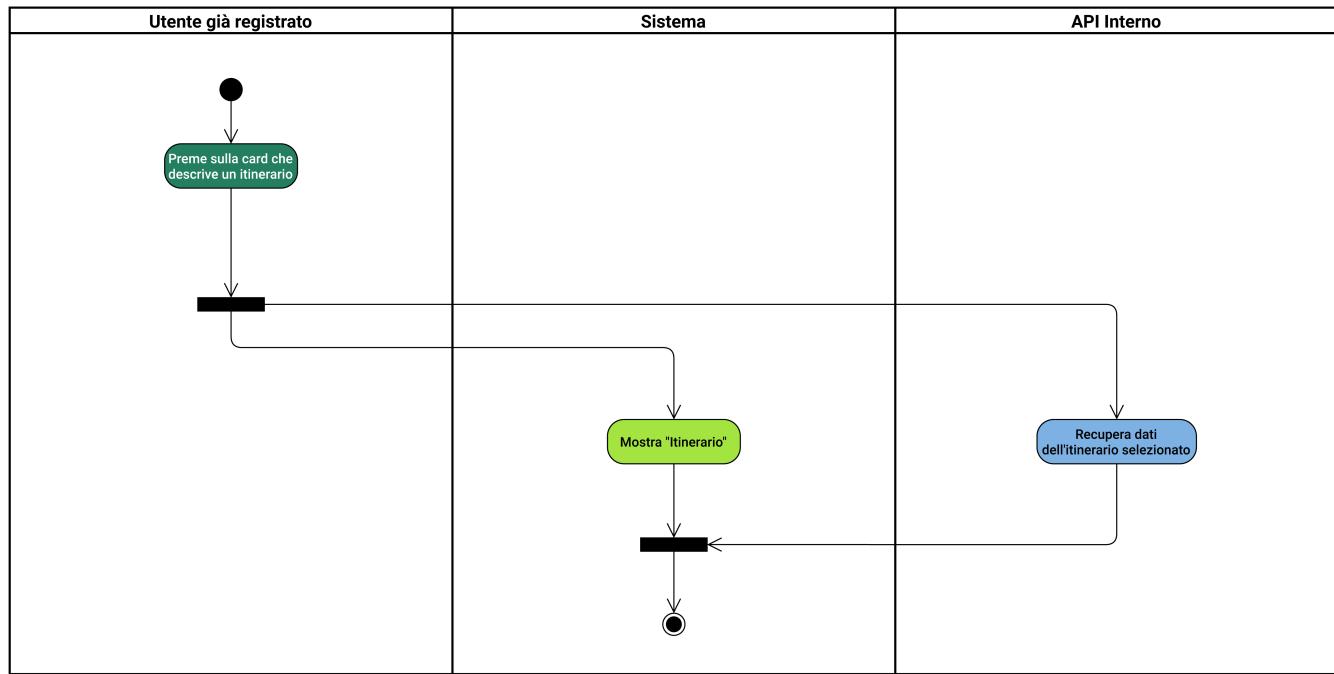
Activity diagram inerente al *requisito funzionale n° 3*



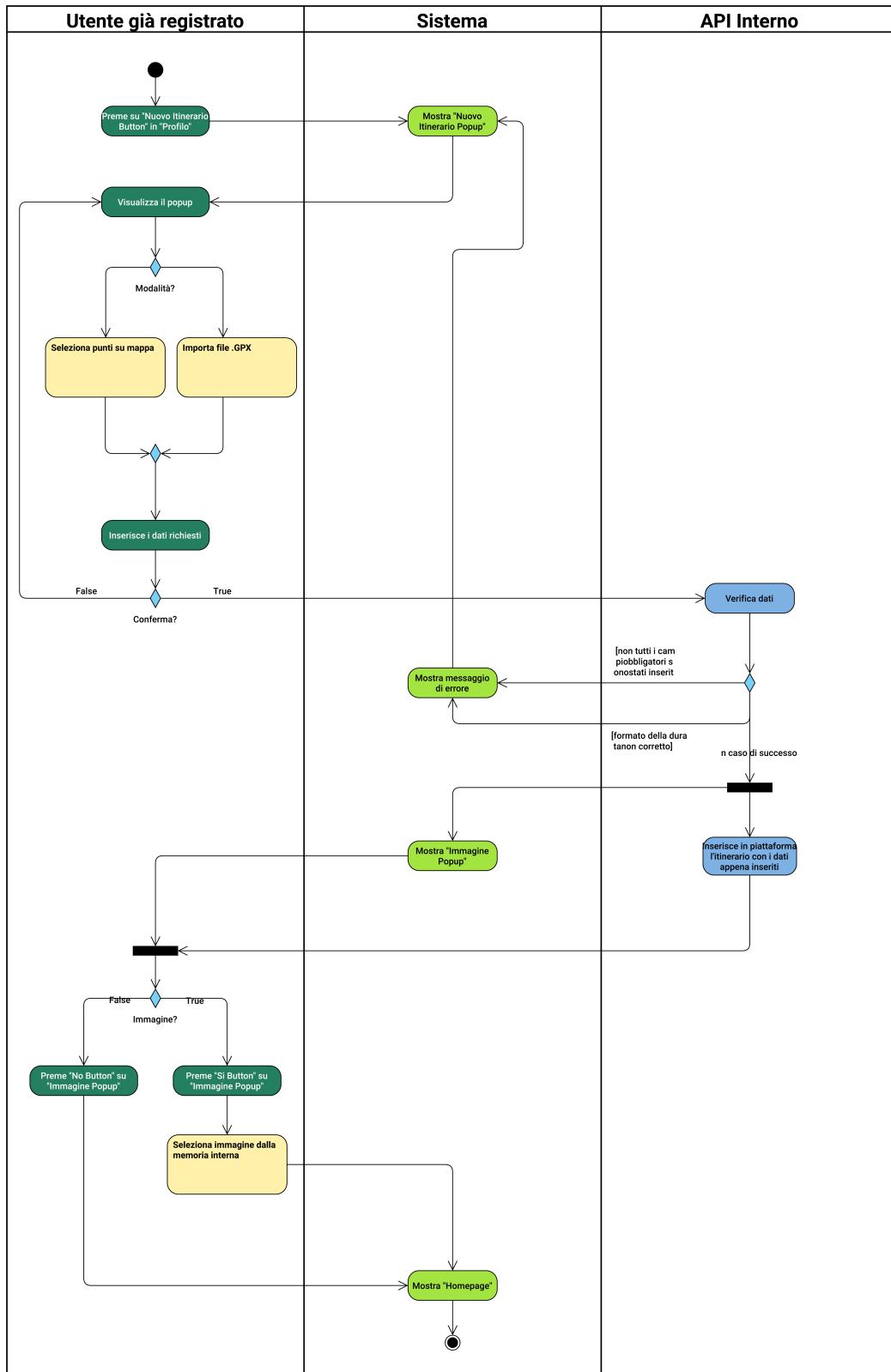
Activity diagram inerente al *requisito funzionale n° 4*

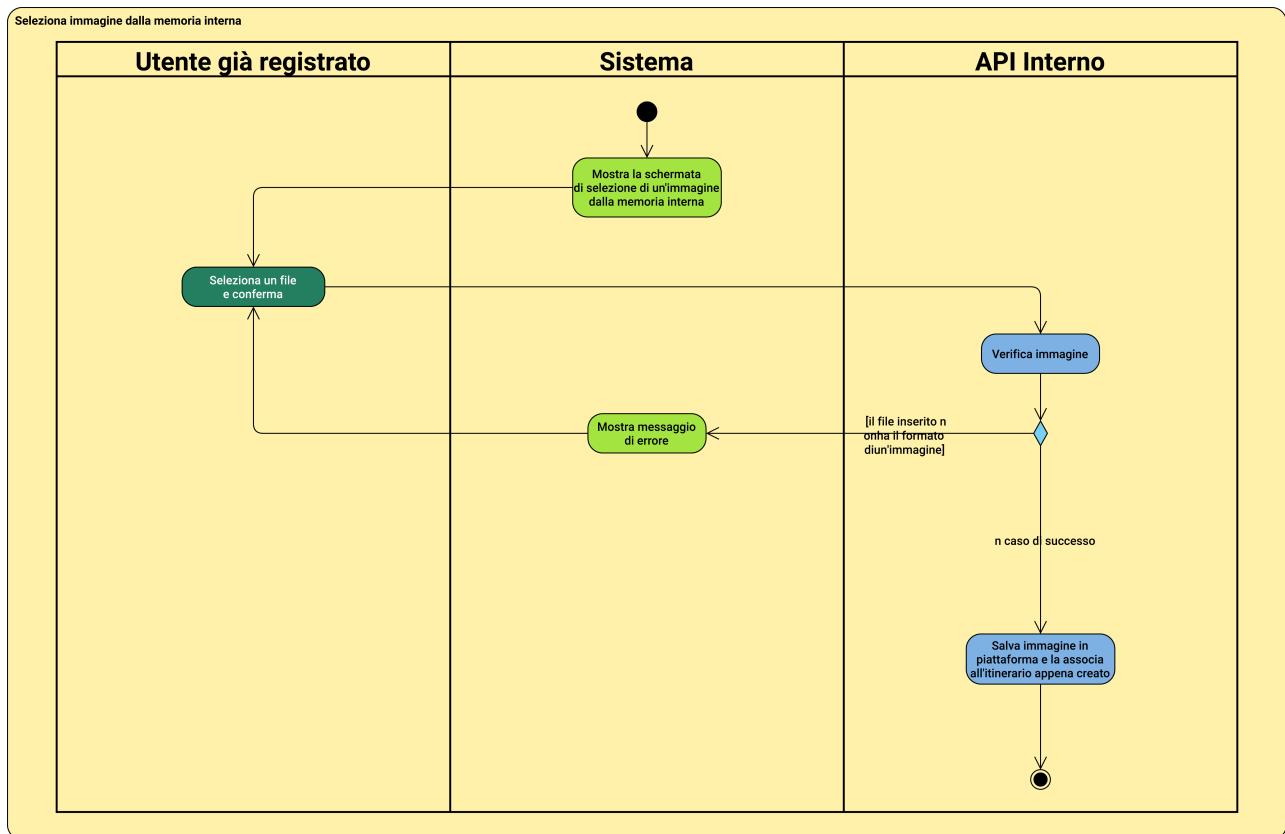
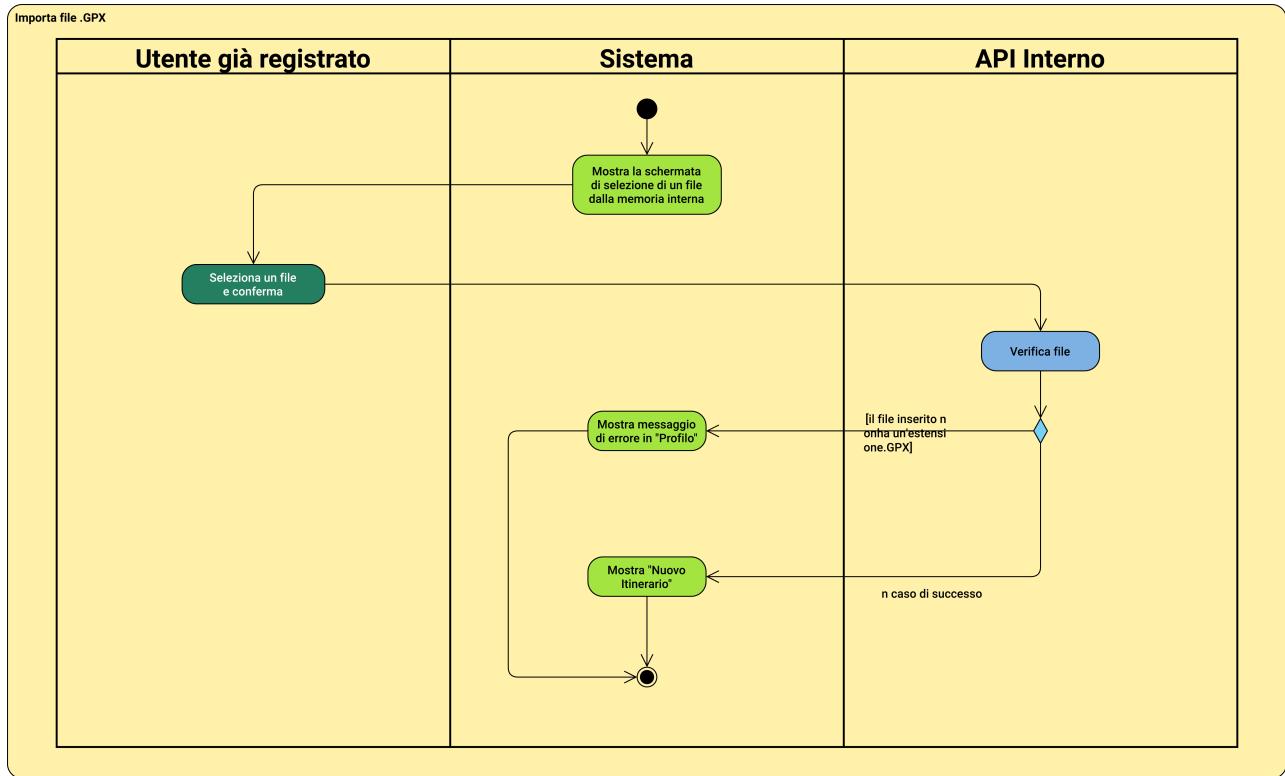


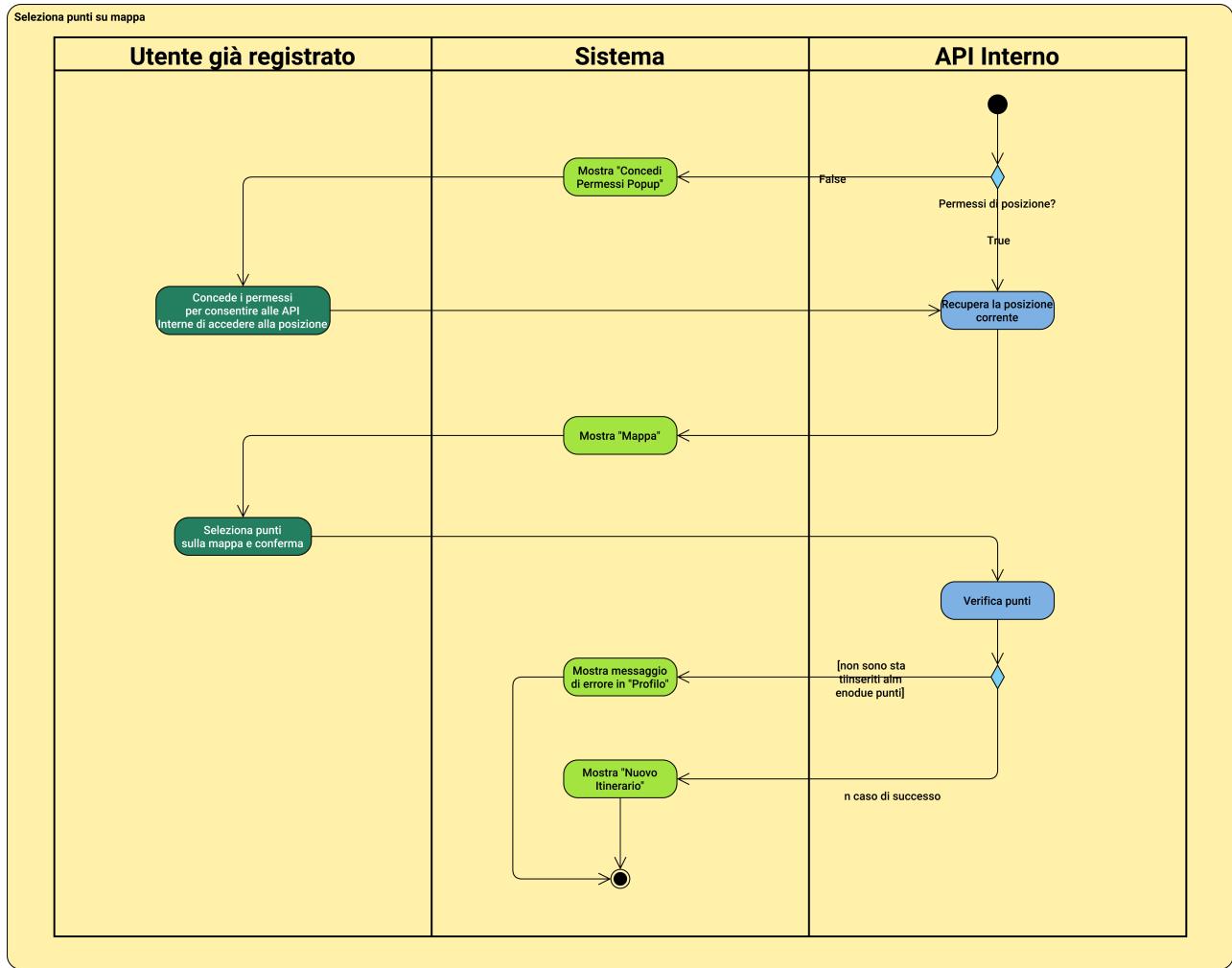
Activity diagram inerente al *requisito funzionale n° 5*



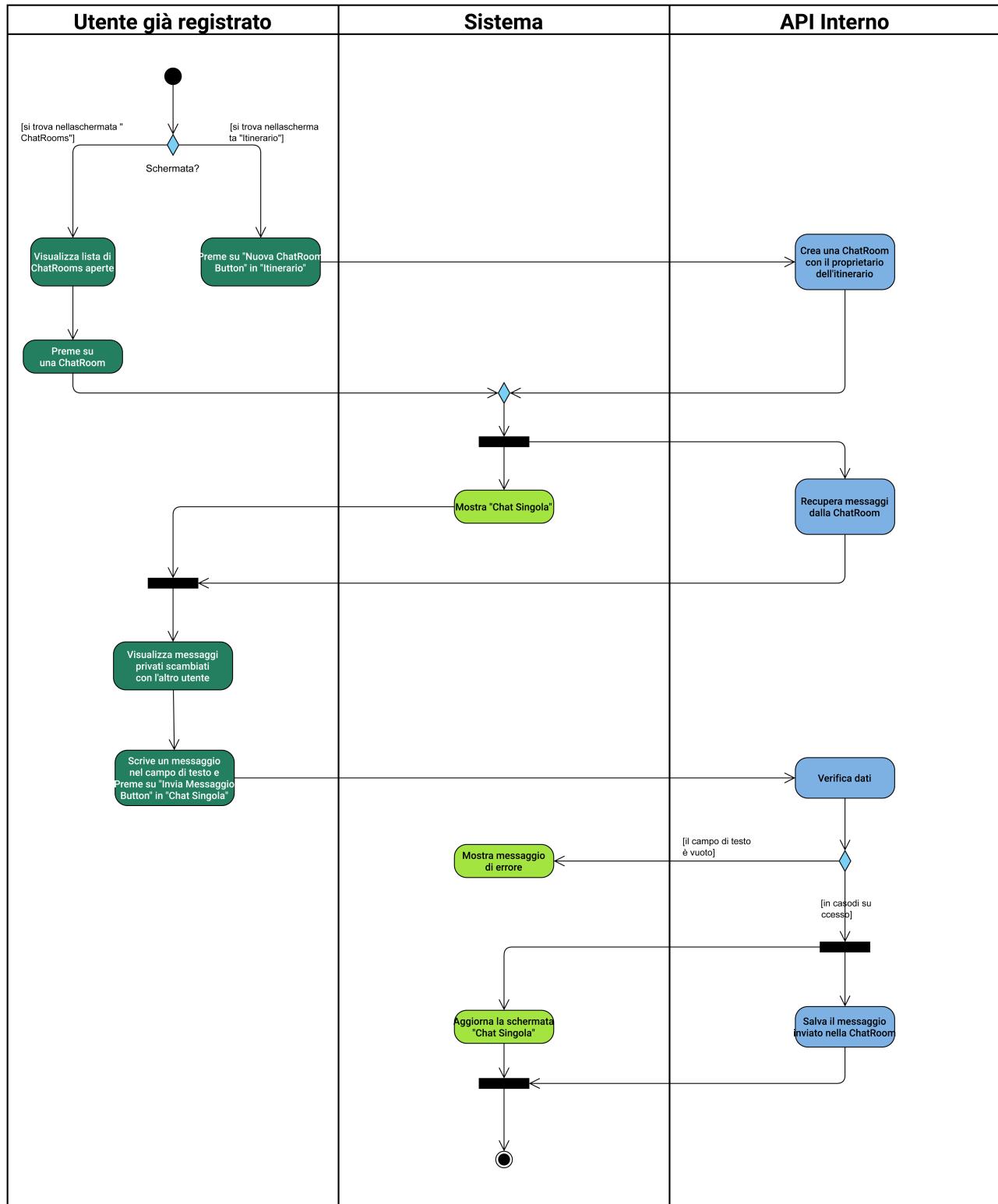
Activity diagram inerenti al *requisito funzionale* n° 6



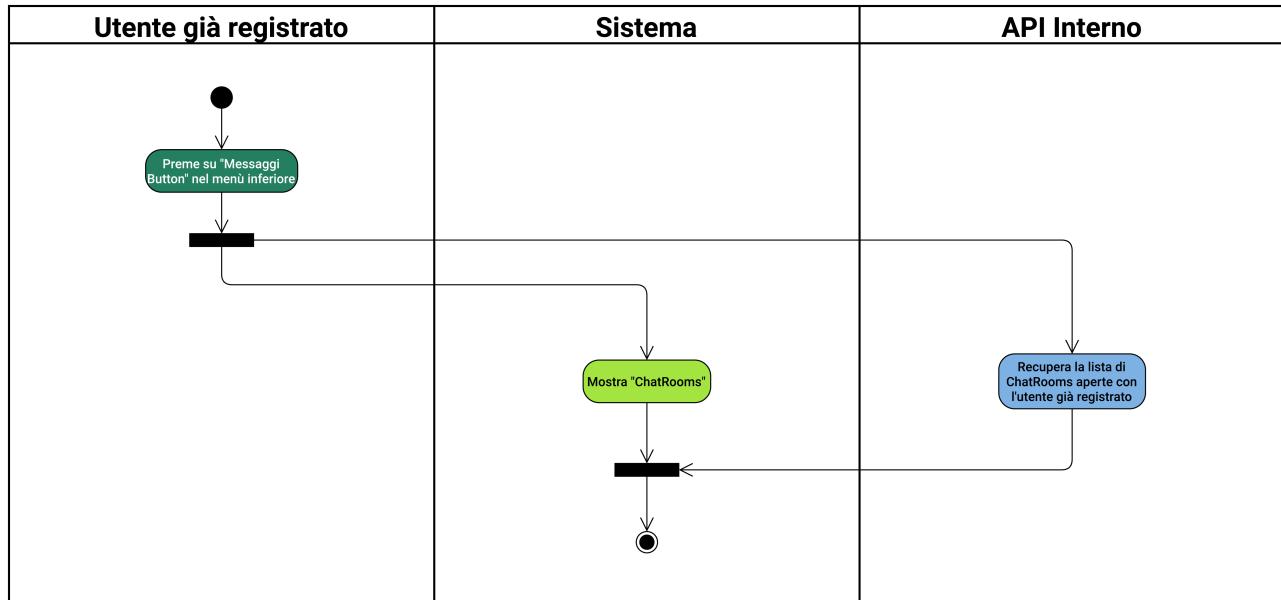




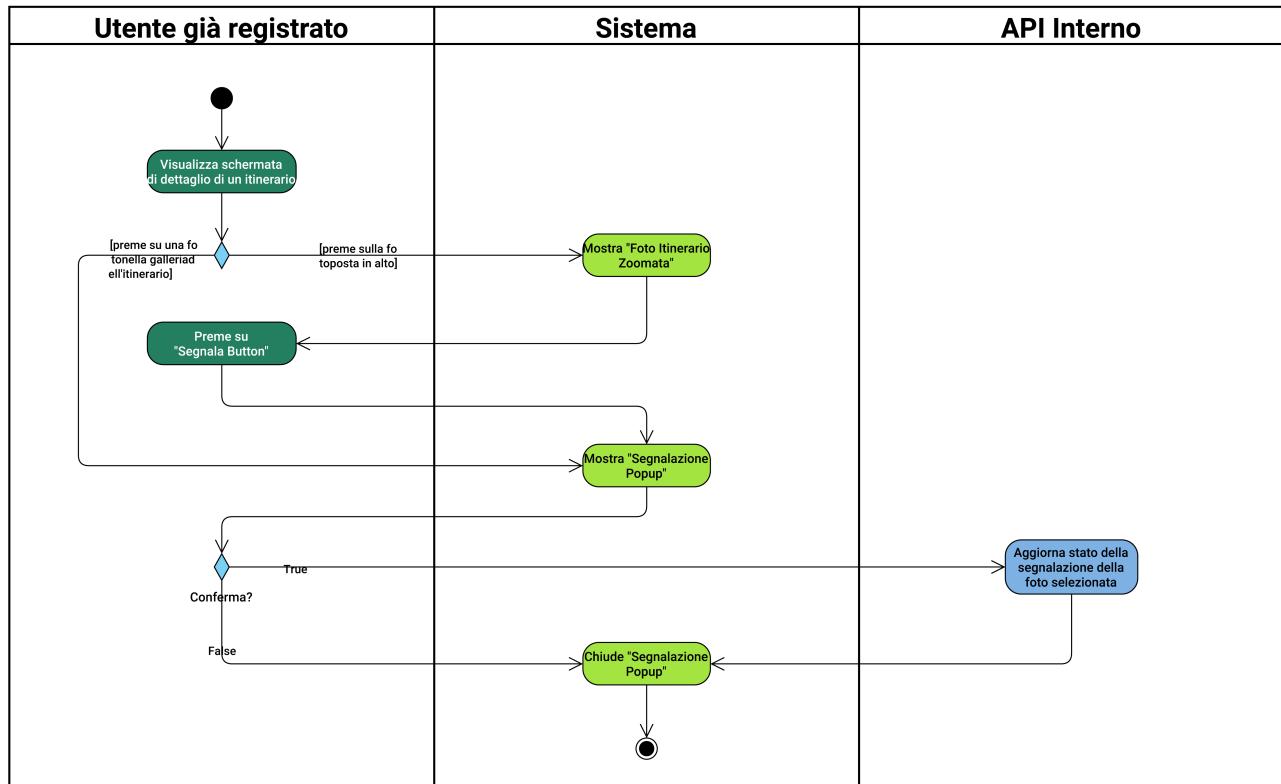
Activity diagram inerente al *requisito funzionale n° 7*



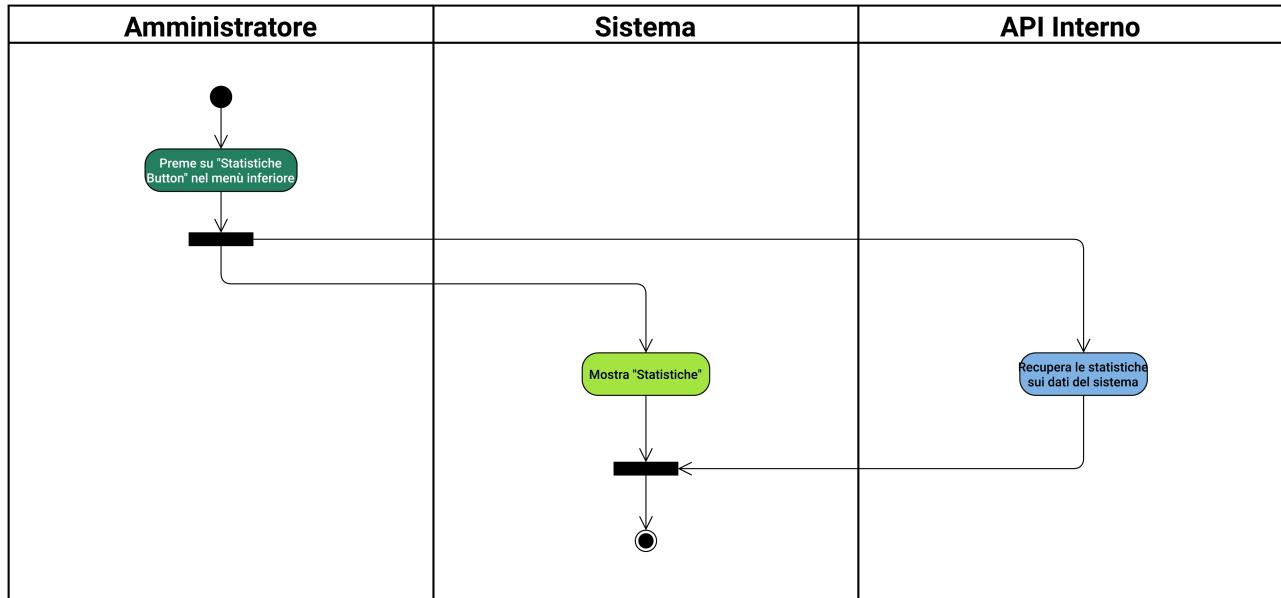
Activity diagram inerente al *requisito funzionale n° 8*



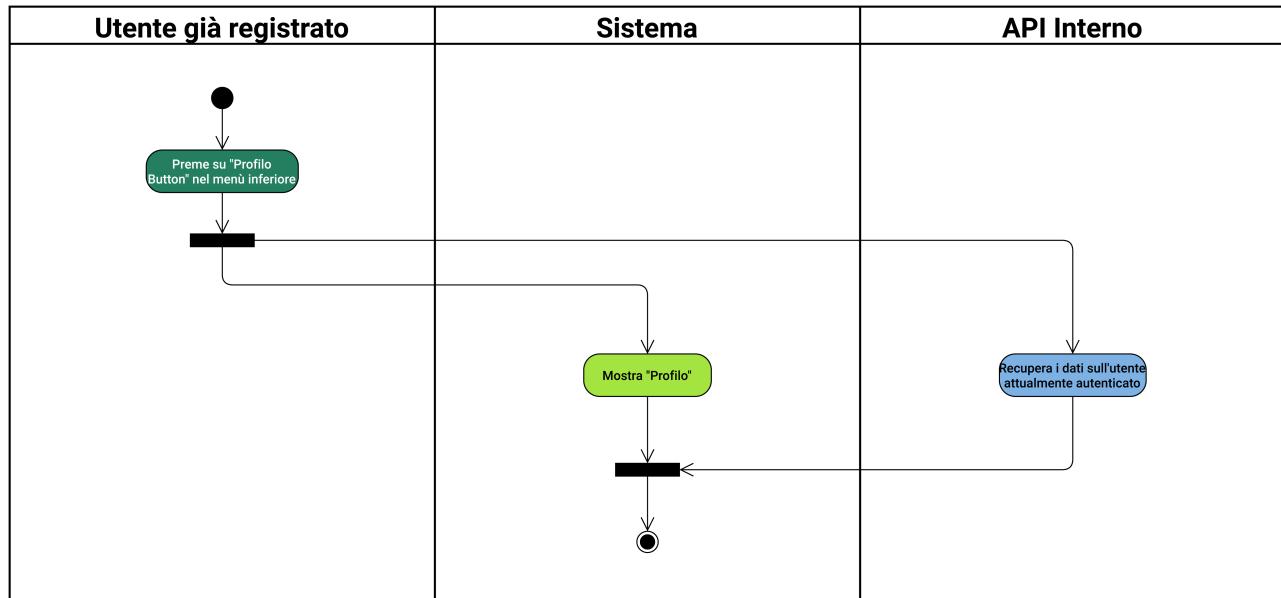
Activity diagram inerente al *requisito funzionale n° 9*

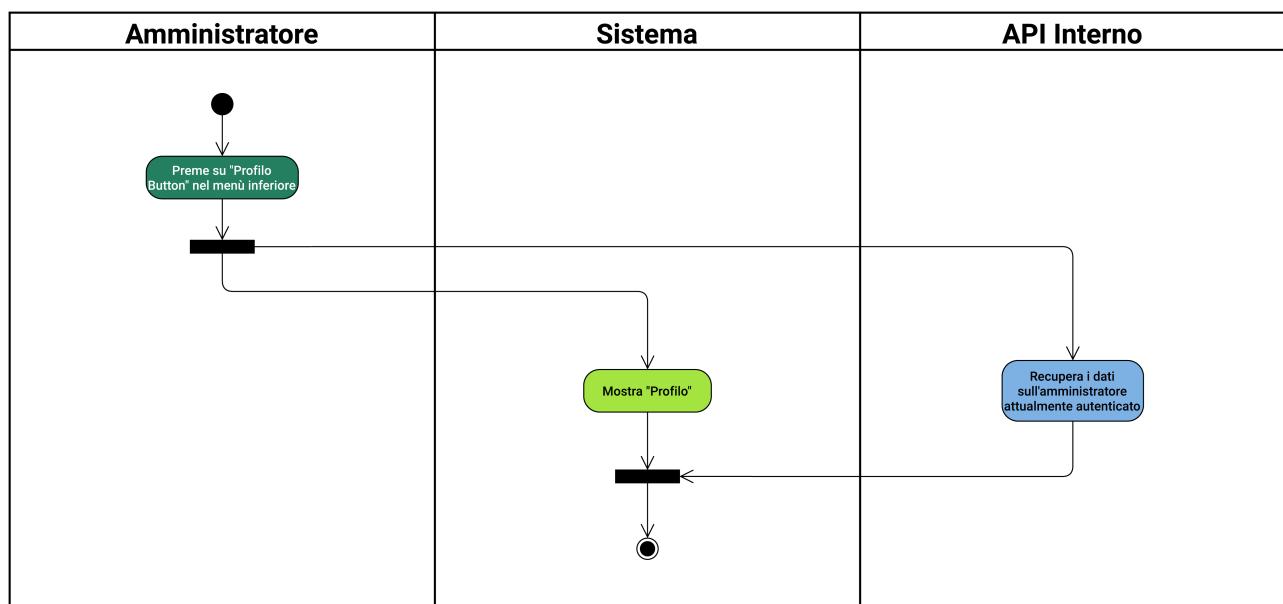
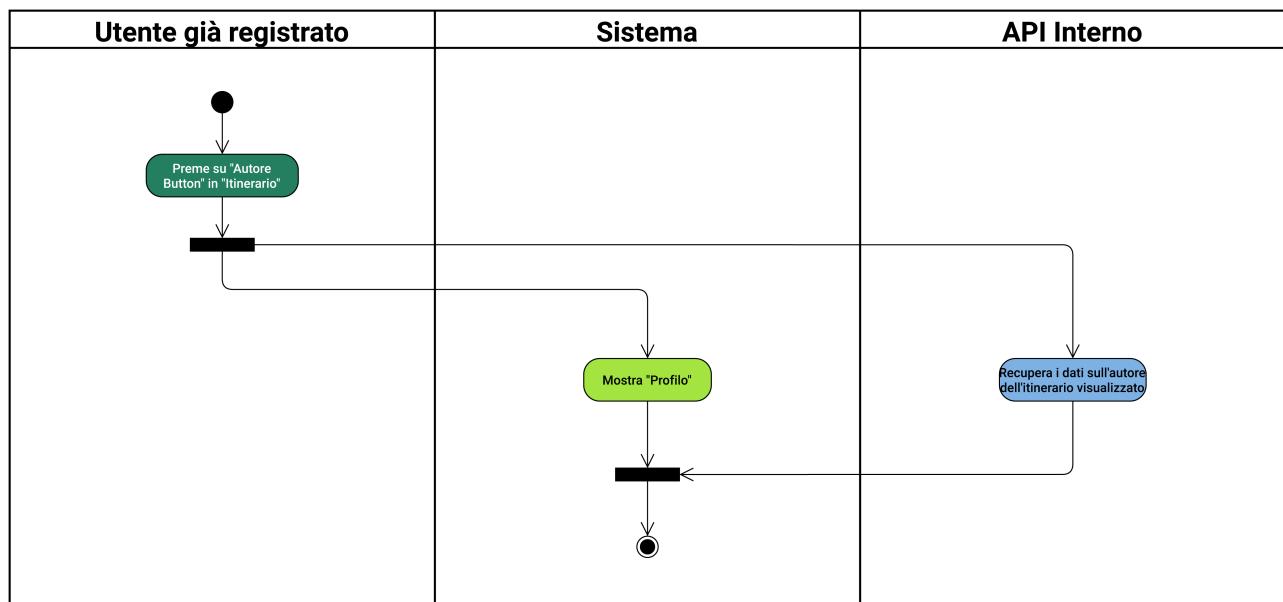


Activity diagram inerente al *requisito funzionale* n° 10



Activity diagram inerenti al *requisito funzionale* n° 11





3 Documento di Design del Sistema

3.1 Analisi dell'architettura con esplicita definizione dei criteri di design

3.1.1 Tecnologie utilizzate

Con lo scopo di rendere sempre affidabile e sicura la piattaforma si è scelto di mettere in campo tutte le tecnologie conosciute durante le nostre esperienze lavorative. In particolare abbiamo fatto uso di:

- ▶ Cloud services, offerto dal noto service provider AWS, di proprietà *Amazon Inc.*, per la gestione dei dati degli utenti e admin iscritti alla piattaforma e delle immagini inerenti agli itinerari
 - ❖ La gestione di tutti questi servizi è gestito attraverso il framework [Amplify](#)⁹
- ▶ Il noto framework Spring Boot, che offre una soluzione *convention over configuration* per il framework Spring di Java, che è stata preferita ad una gestione manuale che non offre web service come [Apache Tomcat](#), necessari al corretto funzionamento della piattaforma;
- ▶ Container¹⁰ docker, per la gestione dei dati;

3.1.2 Linguaggi e strumenti utilizzati

In merito ai linguaggi adoperati per lo sviluppo di entrambi i moduli (Applicazione mobile rivolta ad Admin e agli Utenti, e il server che gestisce le richieste dei primi), la scelta è ricaduta su [Java](#)¹¹(versione 17), che ci ha permesso in tempi brevi di rendere pronta all'uso entrambi i moduli.

La scelta per gli ambienti di sviluppo¹² su 2 prodotti sviluppati dalla Software House [JetBrains](#)

- ▶ [Android Studio](#), per la parte di Android (utilizzata per l'applicativo mobile);
- ▶ [IntelliJ IDEA](#), per la parte di Server (necessaria al corretto funzionamento dell'applicativo mobile).

⁹ Consiste in un set di strumenti e caratteristiche appositamente progettati per consentire agli sviluppatori front-end di applicazioni Web e per dispositivi mobili di costruire rapidamente e facilmente applicazioni full-stack in AWS, con la flessibilità di sfruttare i vari servizi AWS man mano che i casi d'uso si evolvono.

¹⁰ Per una certezza, quasi matematica, di una corretta esecuzione, indipendente dalla macchina che fa girare il programma, utilizziamo un docker container → Un'unità di software che pacchettizza codice e tutte le sue dipendenze per una deploy sicuro e affidabile.

¹¹ Linguaggio Object Oriented che ha permesso un rapido sviluppo degli applicativi richiesti.

¹² Conosciuti anche come IDE. Sono software che, in fase di programmazione, supporta i programmatore nello sviluppo e debugging del codice sorgente di un programma.

Di seguito viene riportata, per ciascuna funzionalità, lo specifico modulo software che si è utilizzato per implementarla.

3.1.3 Autenticazione

Per l'autenticazione (sia degli *admin* che degli *utenti*) si è fatto uso del modulo di Amazon AWS *Cognito*¹³ che mette a disposizione una pool¹⁴ on demand per il salvataggio delle informazioni dei singoli utenti (previa una configurazione iniziale).

Per mantenere una netta suddivisione degli admin e degli utenti si è preferito utilizzare due pool separate (non ha senso infatti tenere i dati di entrambi nello stesso luogo).

Vincoli di unicità dei profili (sia degli utenti che degli admin) e validità delle loro credenziali sono affidate a questo software.

Dai punti sopra citati si deduce che questa scelta permette di mantenere una forte scalabilità dei dati (infatti automaticamente AWS gestisce lo spazio utilizzato).

3.1.4 Gestione del salvataggio delle immagini di utenti, admin e itinerari

Per una forte scalabilità ci si è affidati al noto *Cloud Object Storage S3*, sempre offerto da Amazon AWS, che previa autenticazione, permette:

- ▶ Retrieval di immagini;
- ▶ Salvataggio e caricamento di immagini da parte degli utenti (Definizione di una immagine di profilo e di immagini appartenenti ai singoli itinerari);

3.1.5 Dati degli utenti e degli admin

Per una forte ridondanza dei dati si è scelto di tener traccia dei dati anche sul Server (in modo da poter permettere in un futuro ipotetico cambi repentina delle tecnologie, evitando in questo modo il problema del **vendor lock-in**¹⁵). Ha anche il compito di gestire i dati anche degli itinerari e tutti i riferimenti (foto, tappe, informazioni).

Per rendere ciò possibile si è fatto affidamento al concetto di container sopra citato, che contiene una soluzione embedded¹⁶ del noto DBMS [PostgreSQL](#), che comunicando con Spring boot, ne permette il salvataggio.

¹³Servizio che mette a disposizione delle API e un infrastruttura per la gestione delle chiavi utente (come autenticazione, autorizzazione e registrazione).

¹⁴Si intende una collezione di risorse che vengono conservate in modo da essere utilizzate nell'immediato. Nel nostro caso questa collezione corrisponde ad un bacino di dati di utenti e admin.

¹⁵È la situazione in cui il costo del passaggio a un fornitore diverso è così elevato che il cliente è essenzialmente bloccato con il fornitore originale. A causa delle pressioni finanziarie, di una forza lavoro insufficiente o della necessità di evitare interruzioni delle operazioni aziendali, il cliente è bloccato in quello che potrebbe essere un prodotto o un servizio inferiore.

¹⁶Identifica genericamente un sistema progettato appositamente per un determinato utilizzo (special purpose), spesso con una piattaforma hardware ad hoc, integrati nel sistema che controllano e in grado di gestirne tutte o parte delle funzionalità richieste.

Di seguito una breve spiegazione di come è stato possibile il deployment rapido, attraverso l'utilizzo del programma docker-compose, che sfruttando la definizione del container nel file sotto citato, gestisce automaticamente le dipendenze:

```
1 # Di seguito viene riportato il contenuto del file necessario a docker-compose per il deploy del
2   container
3
4 # Definiamo '3' in quanto rappresenta l'ultima versione disponibile
5 version: '3'
6
7 # Docker Compose lavora con il concetto di `services` dove
8 #           1 service = 1 container
9 # Utilizziamo la keyword 'services' per la creazione del servizio necessario al deploy di un
10    database Postgres
11
12 services:
13   # Il nome del nostro service sarà "database" (è possibile ovviamente modificarlo)
14   database:
15     # Selezionamo l'immagine docker ufficiale di Postgres da DockerHub (useremo l'ultima versione
16     # disponibile)
17     image: 'postgres:latest'
18
19   # Definiamo un nome per il container (risulta comodo in situazioni in cui si vada a deployare `n
20   # ` container diversi)
21   container_name: 'pg_db'
22
23   # Definiamo un volume necessario al container per il salvataggio dei dati
24   volumes:
25     - ./db-data:/var/lib/postgresql/data/
26
27   # Come default Postgres utilizza la porta 5432.
28   # Volendo accedere al database dall'esterno dobbiamo definire una porta di condivisione del tipo
29   # [porta che si vuole usare sulla macchina]:[porta su cui vogliamo comunicare con il container]
30   ports:
31     - 5432:5432
32
33   # Di seguito definiamo delle variabili di ambiente necessarie al database e
34   # alla connessione che viene eseguita dal server (Spring boot)
35   environment:
36     POSTGRES_USER: # Nome dell'utente PostgreSQL
37     POSTGRES_PASSWORD: # Password dell'utente sopra definito
38     POSTGRES_DB: # Il nome del database PostgreSQL
```

3.1.6 Gestione dei dati degli utenti e degli admin loggati

Per rendere l'applicazione sempre reattiva, abbiamo pensato di fare uso di una tecnologia built-in di Android, ovvero il database interno **SQLite**, che andiamo ad utilizzare per la gestione dell'identificativo dell'utente loggato, necessario alla fruizione dell'applicativo.

Abbiamo constato che, sfruttando questa tecnologia, i tempi necessari al caricamento dei dati vengono dimezzati, rendendo rapida la richiesta di dati, necessari alla visualizzazione degli itinerari nell'applicazione.

3.1.7 Gestione dei servizi per utenti e admin

Come abbiamo detto in precedenza, al server è demandato il compito della gestione dei dati e la loro fruizione. Proprio per questo abbiamo deciso di gestire la richiesta dei dati delle rispettive applicazioni con la combinazioni dei 2 framework:

SpringBoot e Retrofit

Il primo utilizzato dal Server, il secondo invece utilizzato da entrambe le applicazioni (rivolte agli utenti e agli admin).

Questi 2 framework comunicano attraverso l'uso delle REST API,¹⁷ in modo tale che il Server faccia da produttore, mentre l'applicazione (e quindi di conseguenza l'admin o l'utente) da consumatore.

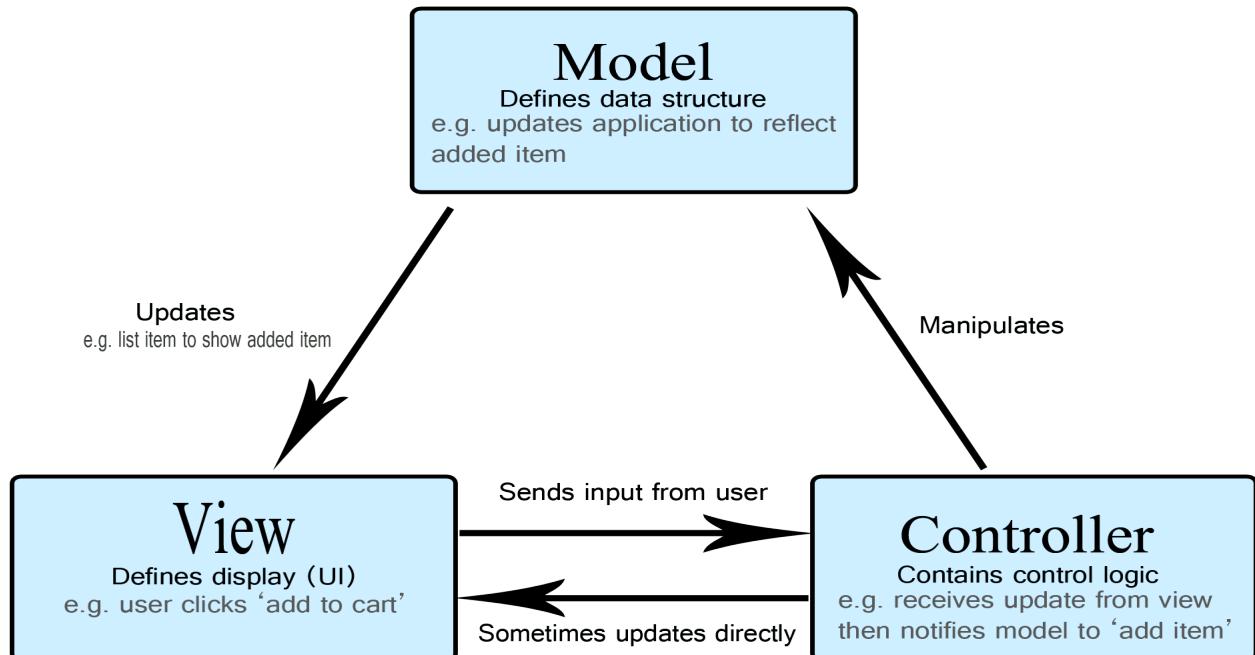
In questo modo l'applicazione non deve far altro che inviare una richiesta HTTP (in maniera trasparente all'utilizzatore) alla suddetta API per ricevere tutti i dati da visualizzare in tempo reale.

¹⁷ Nota anche come API RESTful, rappresenta uno scambio di risorse nel loro stato rappresentativo.

3.2 Design pattern utilizzati

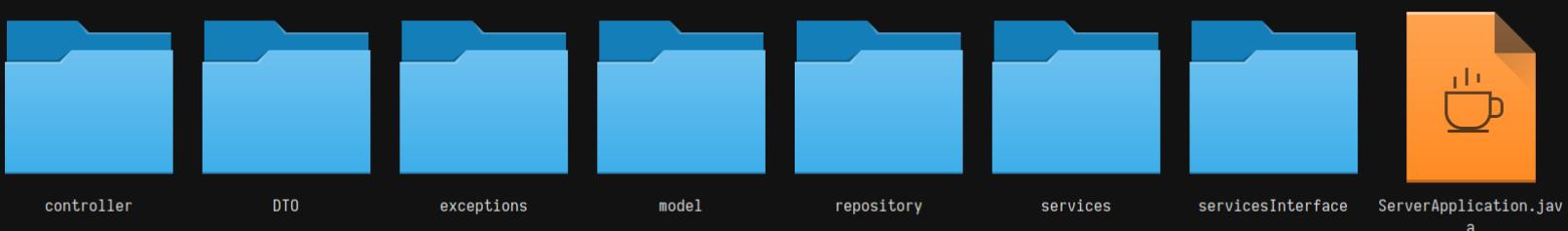
3.2.1 Backend

Prendendo ispirazione dall'approccio architetturale a micro-servizi, ci è sembrato fin da subito fare affidamento, per la gestione del Backend (che come detto in precedenza si basa sul framework Springboot), al pattern architetturale:



un design pattern, che ha permesso in tempi brevi una suddivisione netta e chiara dei componenti e dei loro compiti.

Per maggiore chiarezza, riportiamo di seguito lo stato del progetto,¹⁸ al momento della stesura del documento :



¹⁸Per una visione più dettagliata si faccia riferimento alla javadoc prodotta in allegato al documento.

Dove:

- ▶ **Controller** → Package che mette a disposizione le rotte da utilizzare per le chiamate HTTP del client (l'applicativo mobile);
- ▶ **DTO** → Package che mette a disposizione le classi che verranno usate per la mappatura con i Model utilizzati;
- ▶ **exceptions** → Package che mette a disposizione delle eccezioni custom, che risultano più mirate in caso di errore;
- ▶ **model** → Package che mette a disposizione i modelli che verranno utilizzati per la descrizione dei dati che si vuole andare a rappresentare, ovvero come sono caratterizzati
 - ◊ Gli utenti;
 - ◊ Gli admin;
 - ◊ Gli itinerari e i loro attributi;
 - ◊ Le chatroom e i messaggi a loro corrispondenti;
- ▶ **repository** → Package che rappresenta il collegamento messo a disposizione da Springboot tra l'applicativo java e il database corrispondente;
- ▶ **service** → Package che contiene tutta la logica di gestione delle operazioni utilizzate dalle classi controller corrispondenti;
- ▶ **ServiceInterface** → Package che contiene le astrazioni delle classi presenti in **service**
- ▶ **ServiceApplication** → Come si può intuire dal nome, è il cuore dell'applicativo.

3.2.2 Frontend

Per le applicazioni mobile (sia per utenti che per admin), si è deciso di prendere il meglio di 2 design pattern architetturali:

- ▶ MVC (Model-View-Controller), già utilizzato per la gestione del Server;
- ▶ MVP (Model-View-Presenter), un design pattern utilizzato nel campo mobile development.

Analogamente a come fatto per il backend andiamo a definire lo stato del progetto, al momento della stesura del documento, per l'applicativo ad uso utente e ad uso admin.¹⁹

¹⁹ Si noti che la vista risulta diversa, per la diversa implementazione del design pattern architettonico.

```
► └ model
  └ presenter
    ► └ adapter
    ► └ amplify
    ► └ api
    ► └ callbackInterface
    ► └ request
    ► └ retrofit
  └ utils
    ► └ constants
    ► └ handler
    ► └ persistence
  └ view
    ► └ activity
    ► └ fragment
```

Dove:

- **model** → Package che mette a disposizione i modelli che verranno utilizzati per la mappatura con quelli presenti sul Server;
- **presenter** → Package che contiene i seguenti componenti:
 - ◊ **adapter** → Package che mette a disposizione la logica delle viste dei dati dei singoli elementi all'interno della grafica dell'applicazione;
 - ◊ **amplify** → Package che contiene tutti i metodi utilizzati per il collegamento con la piattaforma Amplify (utilizzata per la gestione delle credenziali di Cognito);
 - ◊ **api** → Package che contiene tutti i metodi necessari alle chiamate HTTP al Server;
 - ◊ **callbackinterface** → Package che contiene tutte le callback utilizzate durante le chiamate HTTP, necessarie all'ottenimento dei dati dal Server;
 - ◊ **request** → Package che ha in sé l'implementazione delle gestione dei dati in seguito alle request HTTP;
 - ◊ **retrofit** → Package che contiene tutti i singleton²⁰ necessari al framework **Retrofit** per effettuare chiamate di tipo

²⁰Ha lo scopo di garantire che di una determinata classe venga creata una e una sola istanza, e di fornire un punto di accesso globale a tale istanza.

HTTP;

- ▶ **utils** → Package che contiene i seguenti componenti:

- ◊ **constants** → Contiene tutte le costanti necessarie al corretto funzionamento dell'applicativo;
- ◊ **handler** → Contiene gli handler dello stato dell'applicazione in caso di errore. Ha il compito di avvisare l'utente di un problema nella richiesta appena effettuata;
- ◊ **persistence** → Contiene la gestione del database locale necessario ad una gestione più veloce delle richieste;

- ▶ **view** → Package che contiene i seguenti componenti:

- ◊ **activity** → Contiene le activity fondamentali dell'applicazione;
- ◊ **fragment** → Contiene le fragment fondamentali dell'applicazione;

3.3 Come abbiamo aumentato la visibilità del progetto

Per rendere più visibile e competitivo il prodotto finale, abbiamo deciso di mettere in campo tutte le competenze a nostra disposizione, per produrre un sito web di presentazione (raggiungibile dal seguente [link](#)), a cui tutti i possibili nuovi utenti possano fare riferimento.

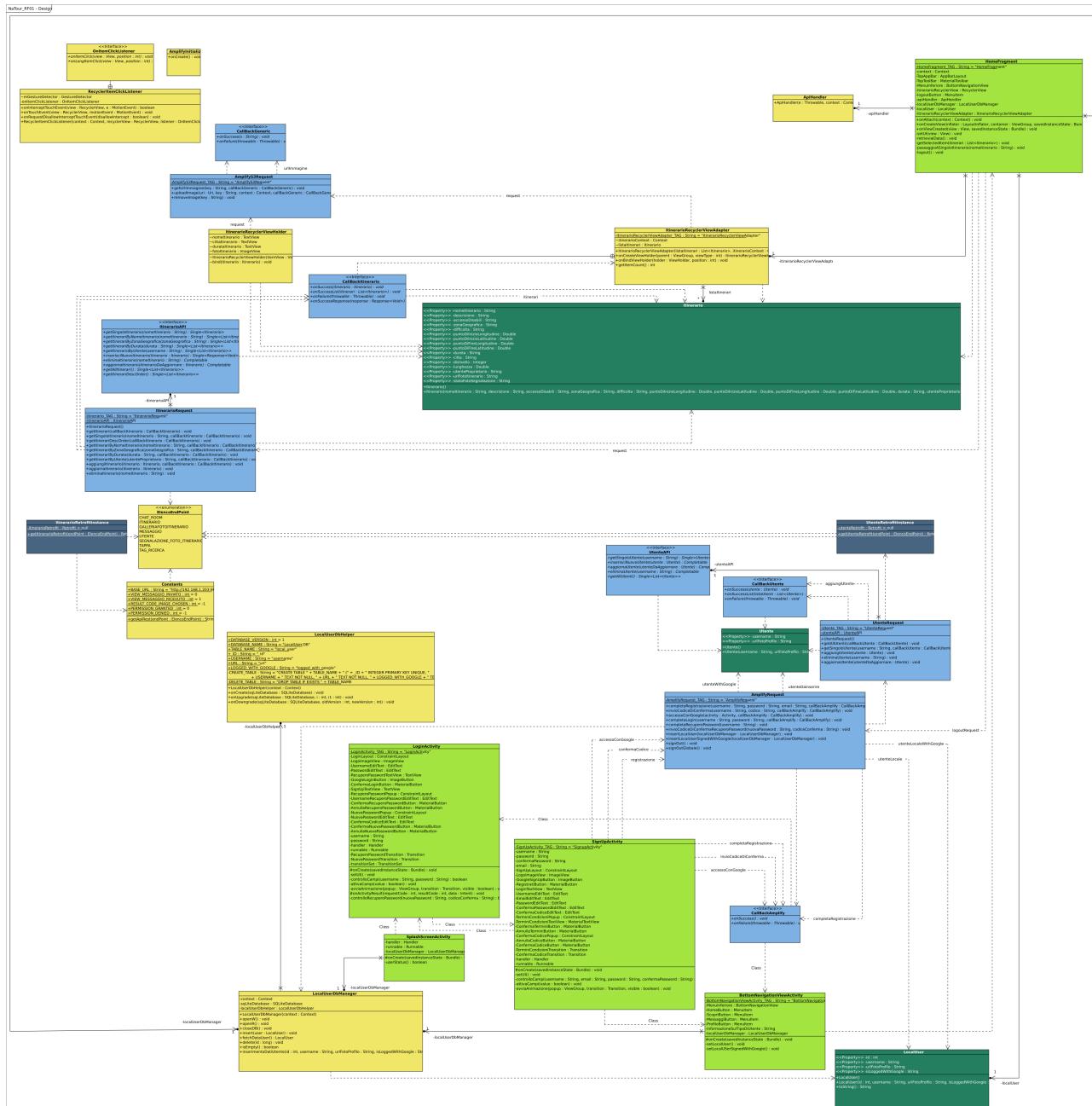
Abbiamo messo a disposizione un portale, che permetta, a chi si avvicina al progetto, di mettersi in comunicazione con noi per ricevere assistenza o per chiarimenti, con un form anonimo (avendo in mente il rispetto per la privacy) raggiungibile al seguente [link](#).

Infine è stata aggiunta anche la possibilità di poter scaricare, in anteprima, l'applicazione, in attesa che venga rilasciata sul play store.

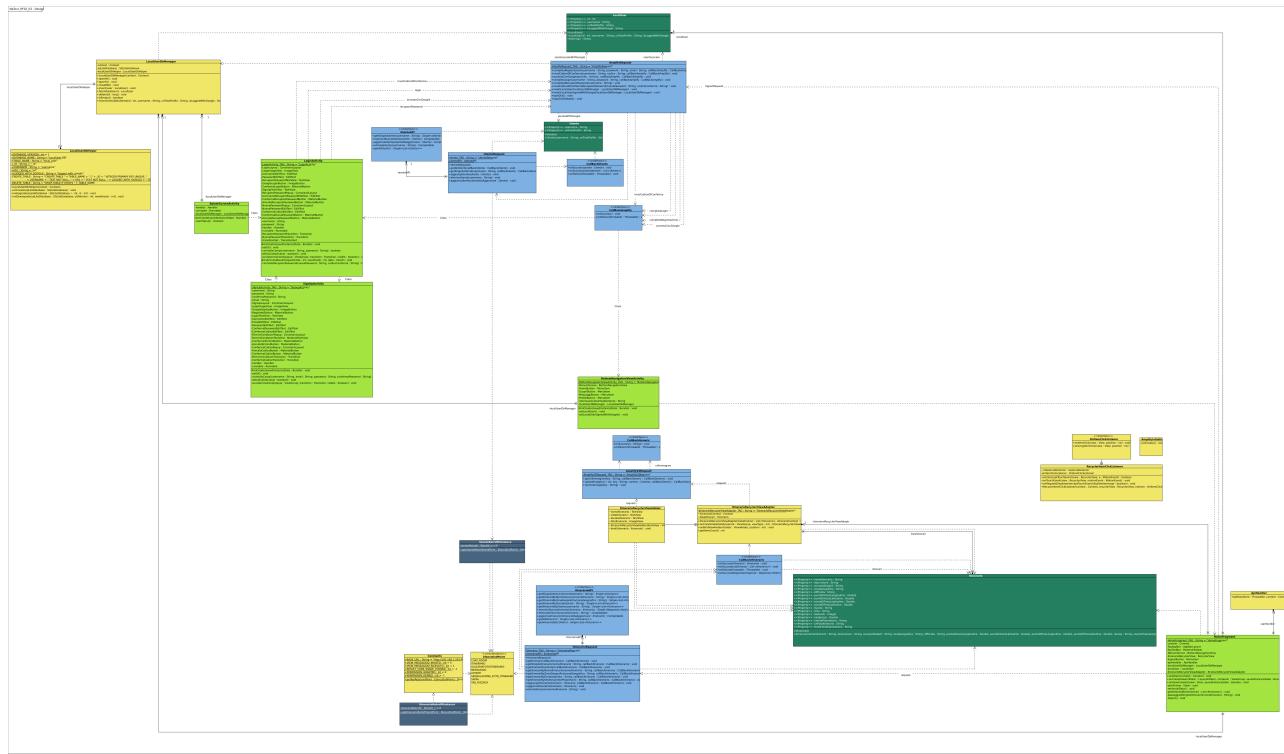
3.4 Diagramma delle Classi di Design

Riportiamo, come da titolo, i diagrammi delle classi di *Design* che sono stati prodotti durante lo sviluppo della piattaforma *NaTour*:

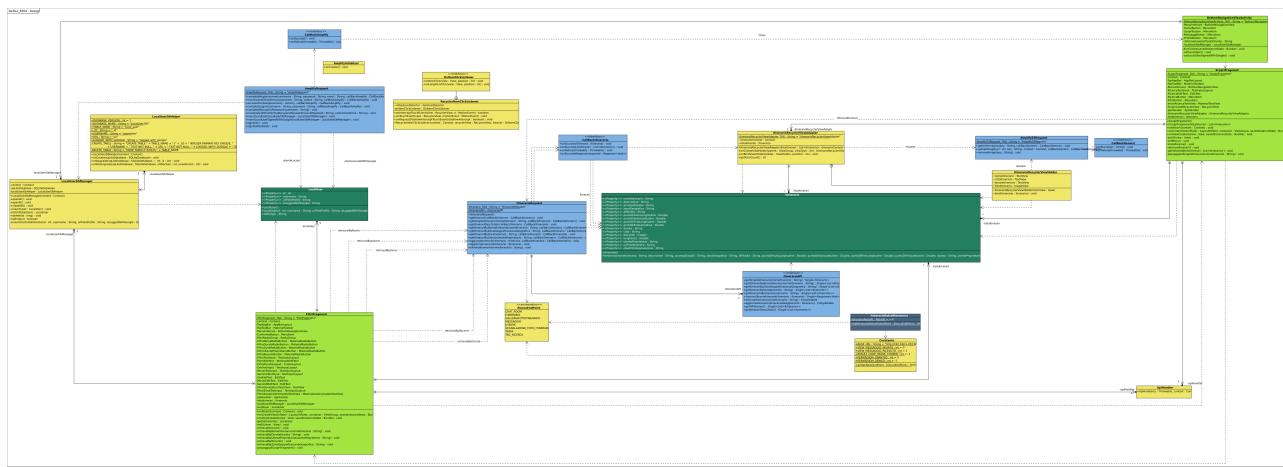
Class diagram inerente al *requisito funzionale n° 1*



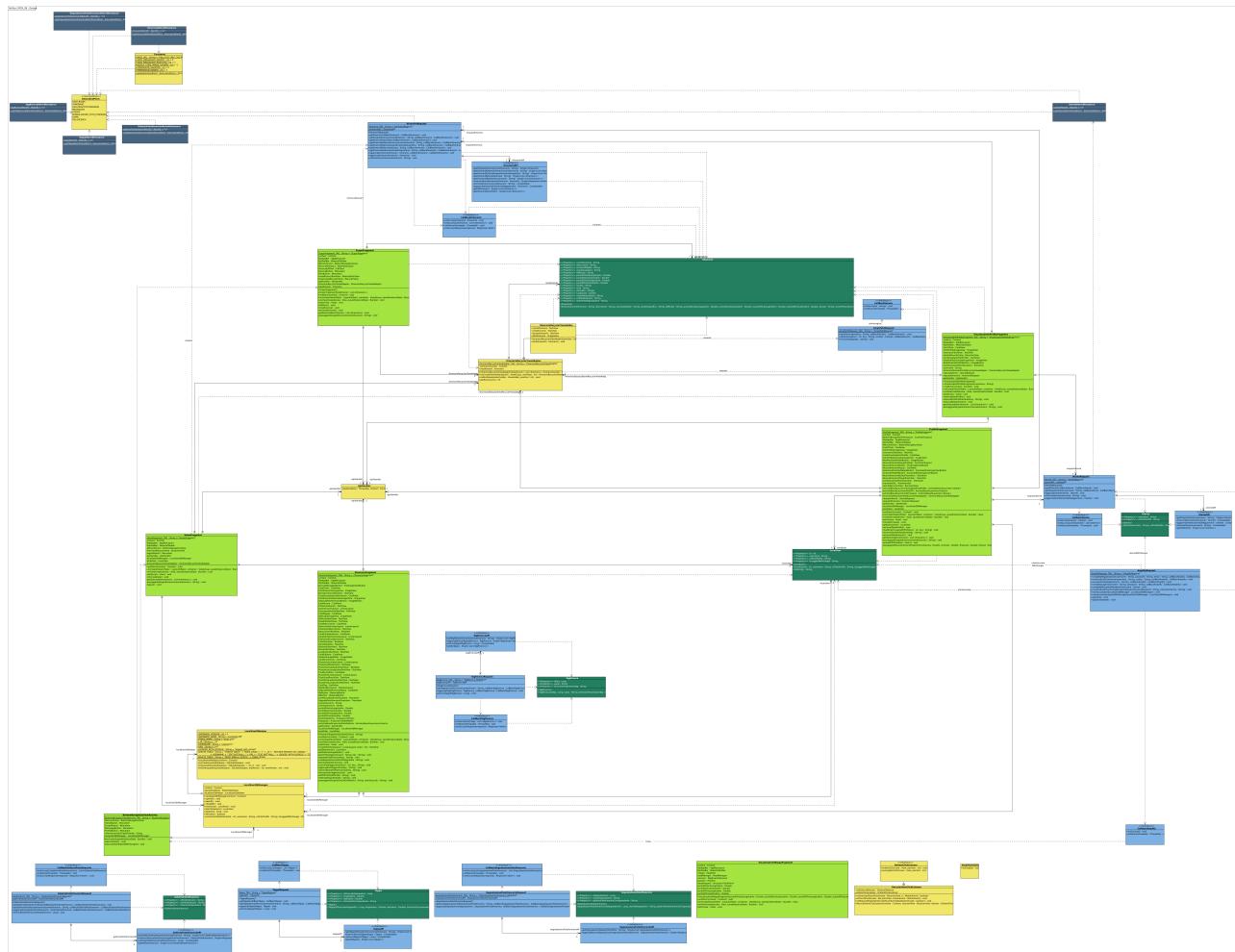
Class diagram inerenti ai *requisiti funzionali* n° 2-3



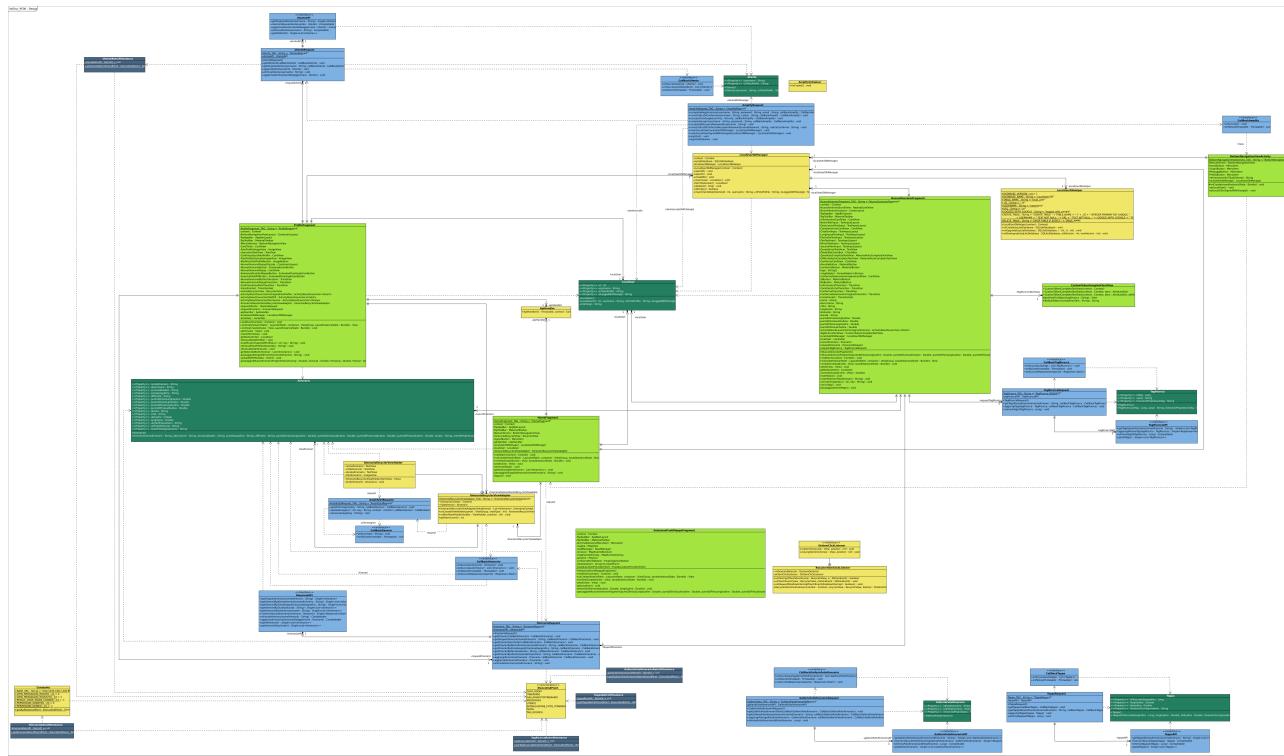
Class diagram inerente al *requisito funzionale* n° 4



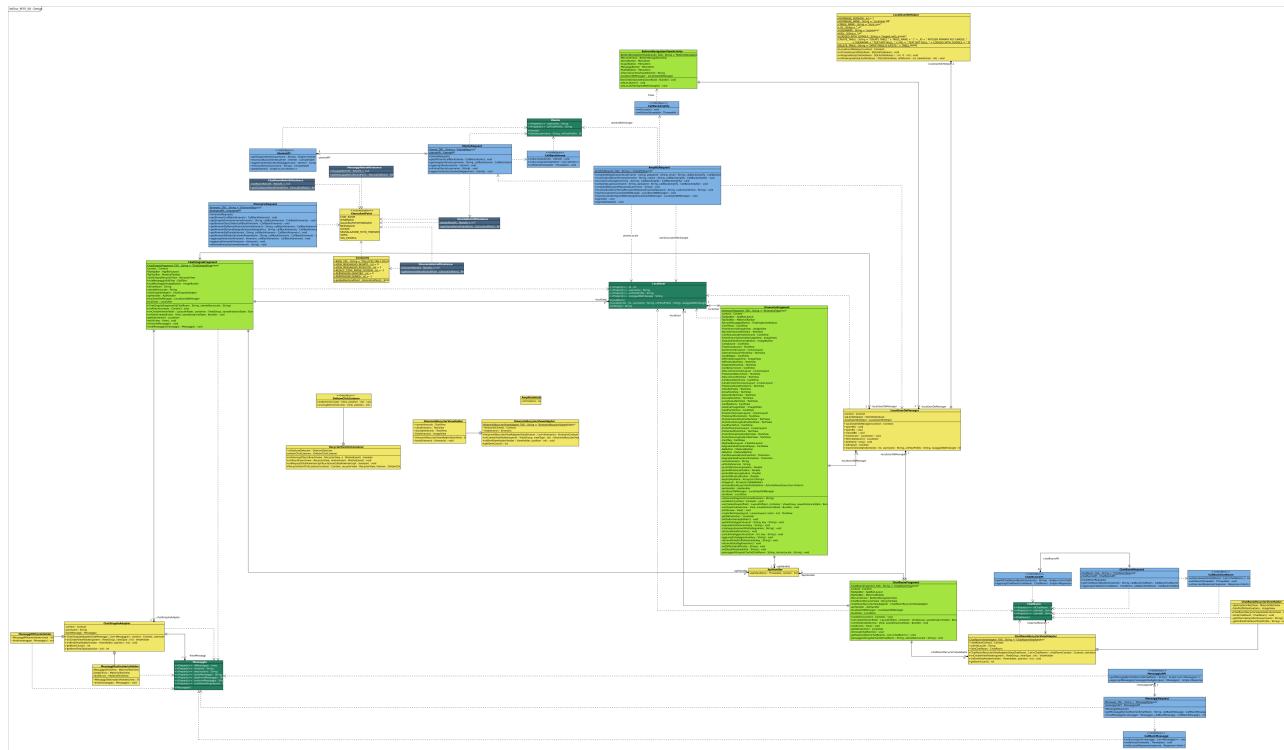
Class diagram inerenti ai *requisiti funzionali* n° 5-9



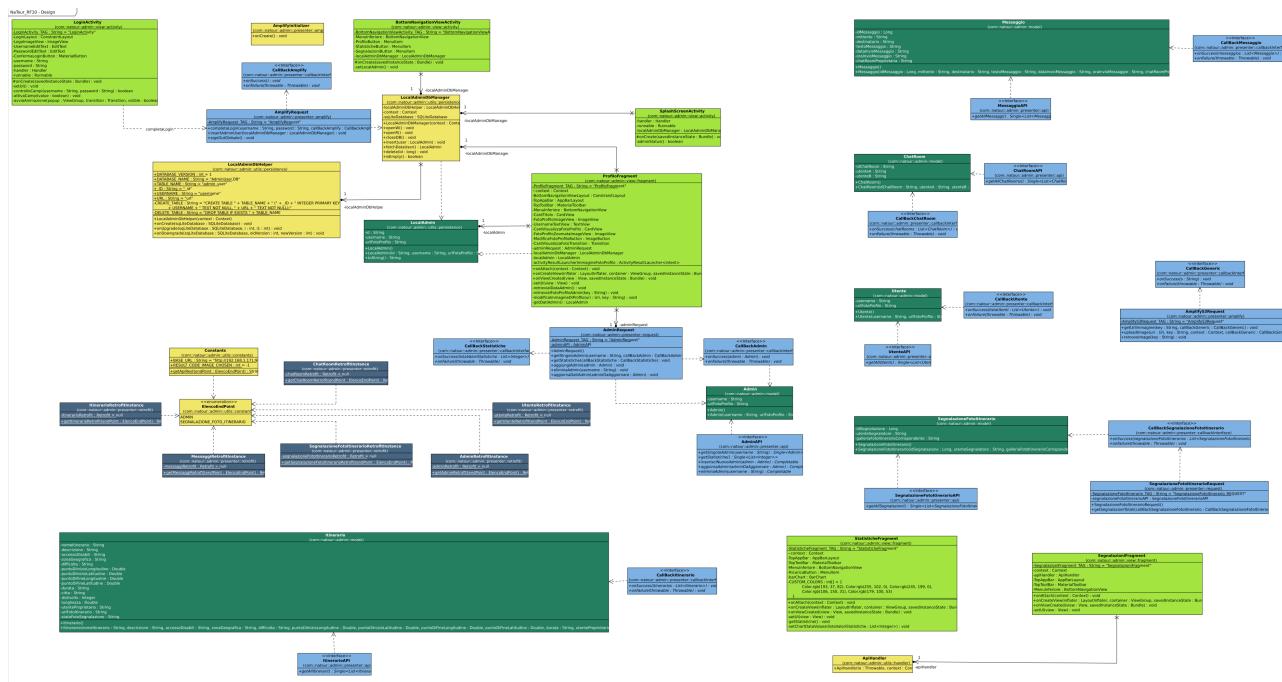
Class diagram inerente al *requisito funzionale n° 6*



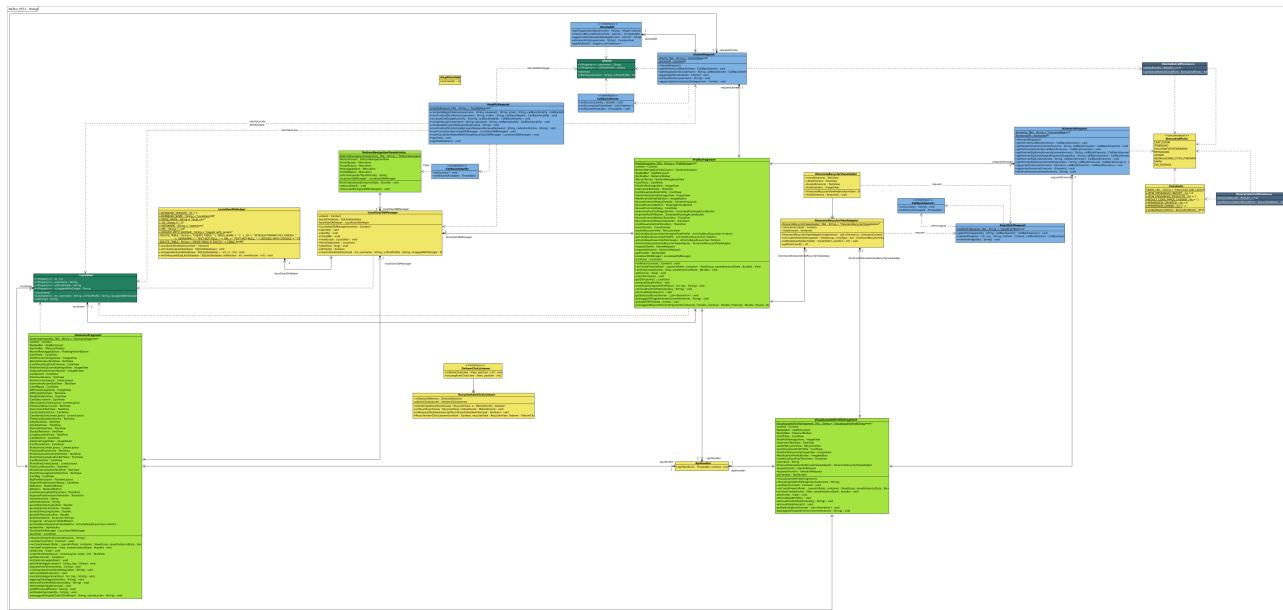
Class diagram inerenti ai *requisiti funzionali* n° 7-8



Class diagram inerente al *requisito funzionale* n° 10

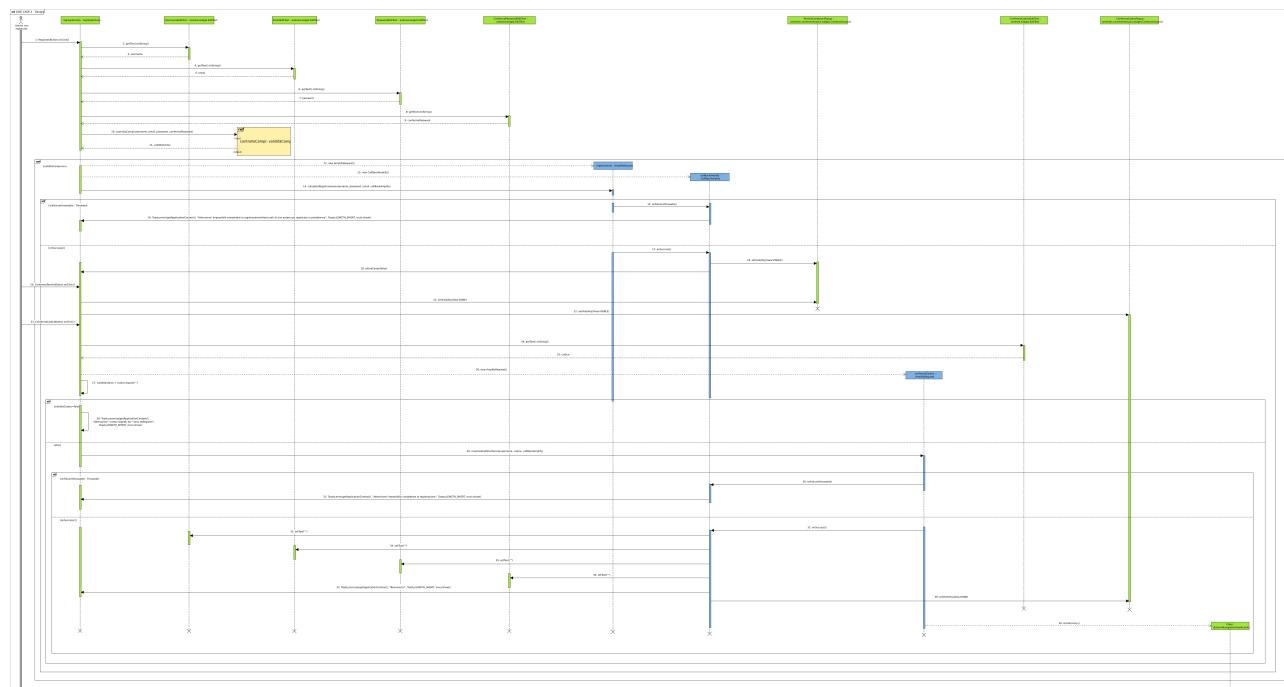
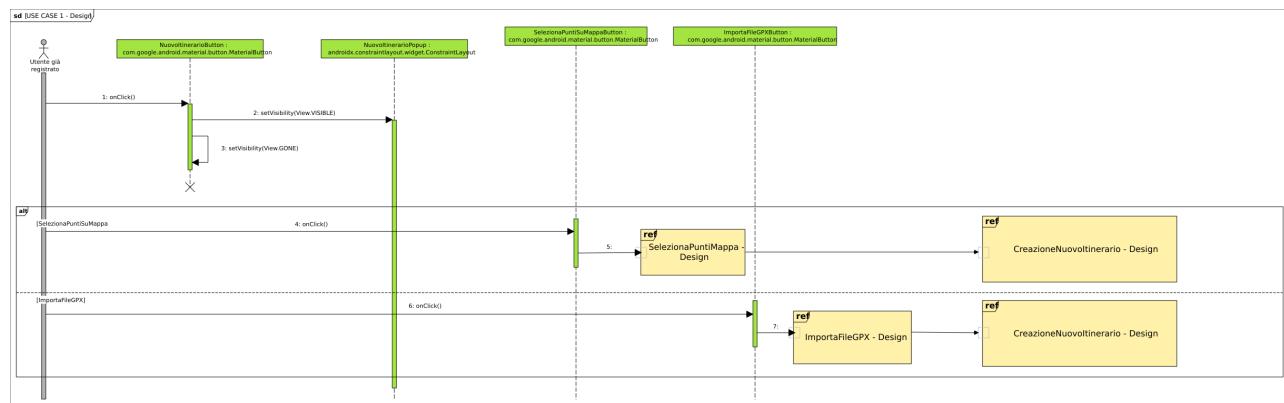


Class diagram inerente al *requisito funzionale* n° 11

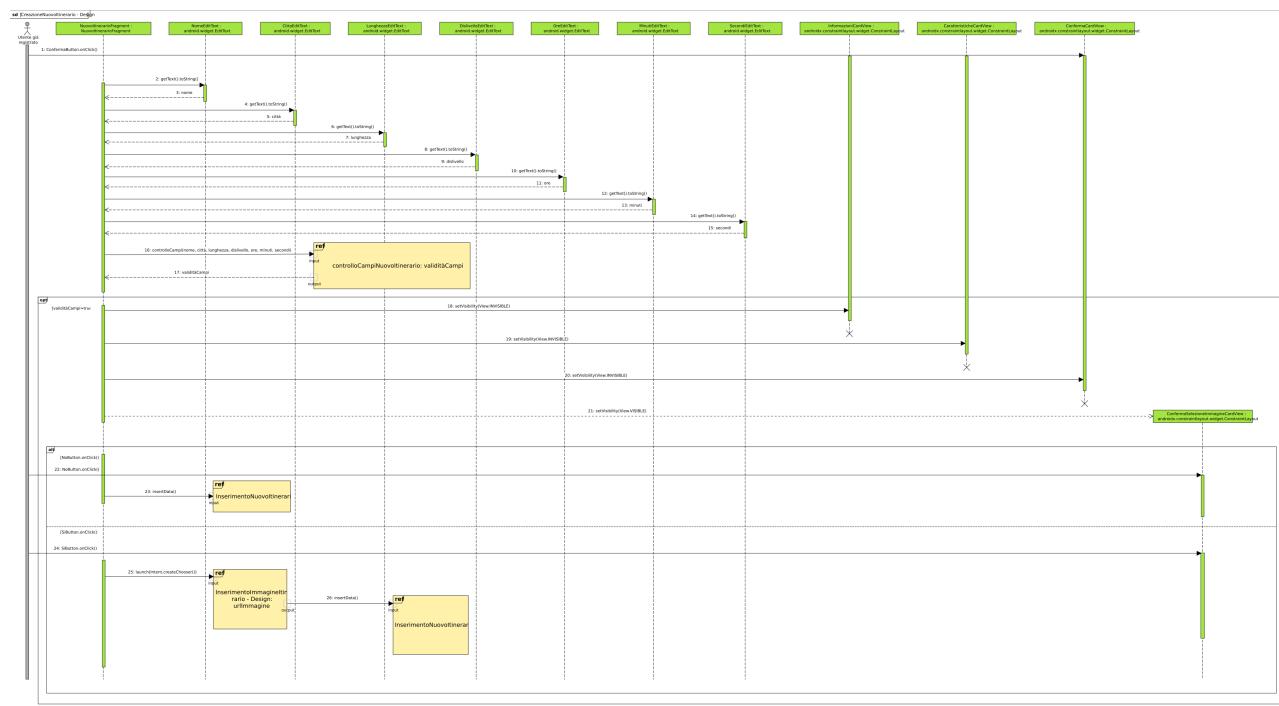


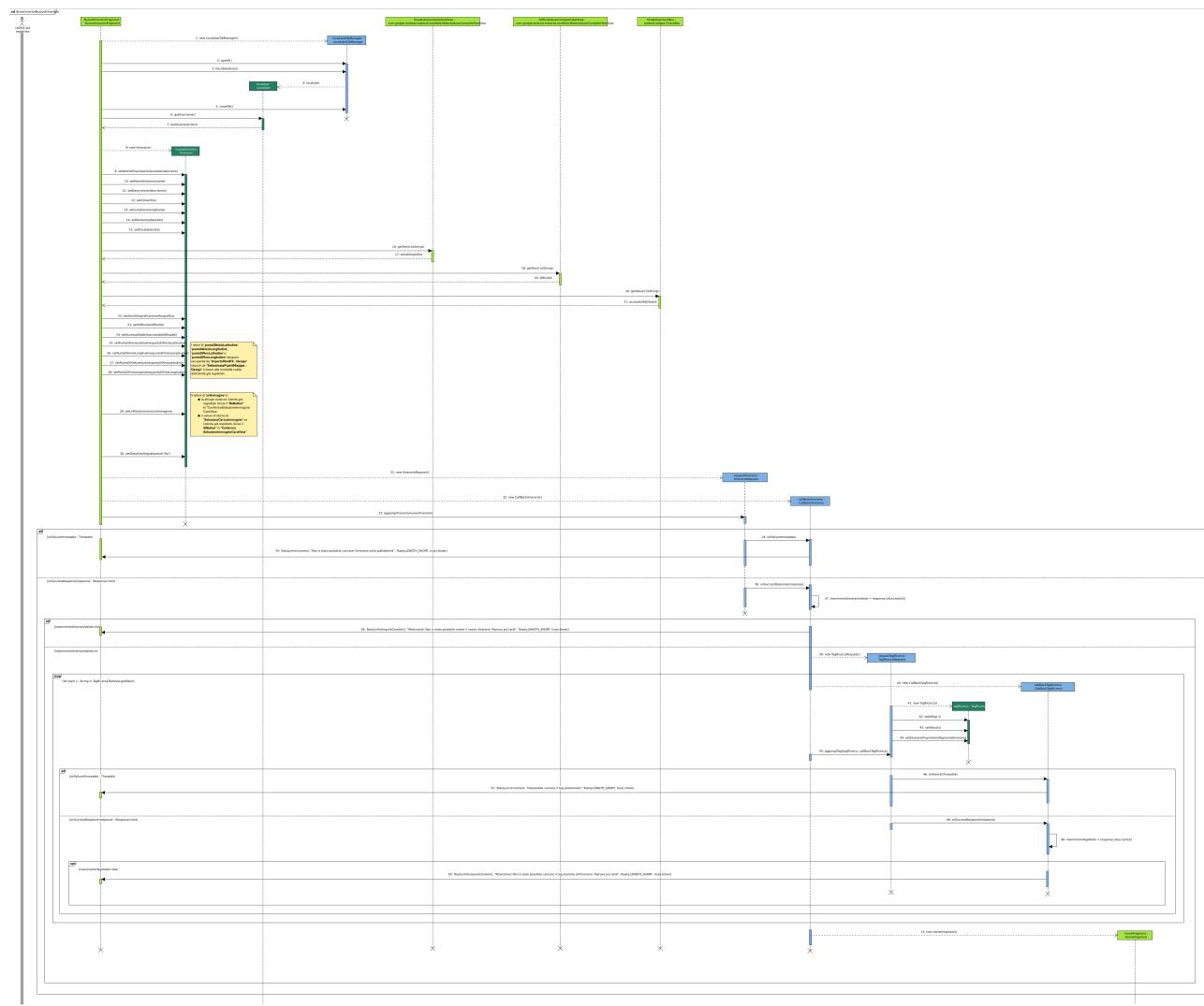
3.5 Diagrammi di Sequenza di Design per 2 casi d'uso significativi

Di seguito vengono riportati i diagrammi di sequenza per i 2 casi d'uso in precedenza presentati:



e inerenti alla gestione di **creazione/inserimento** di un nuovo itinerario:





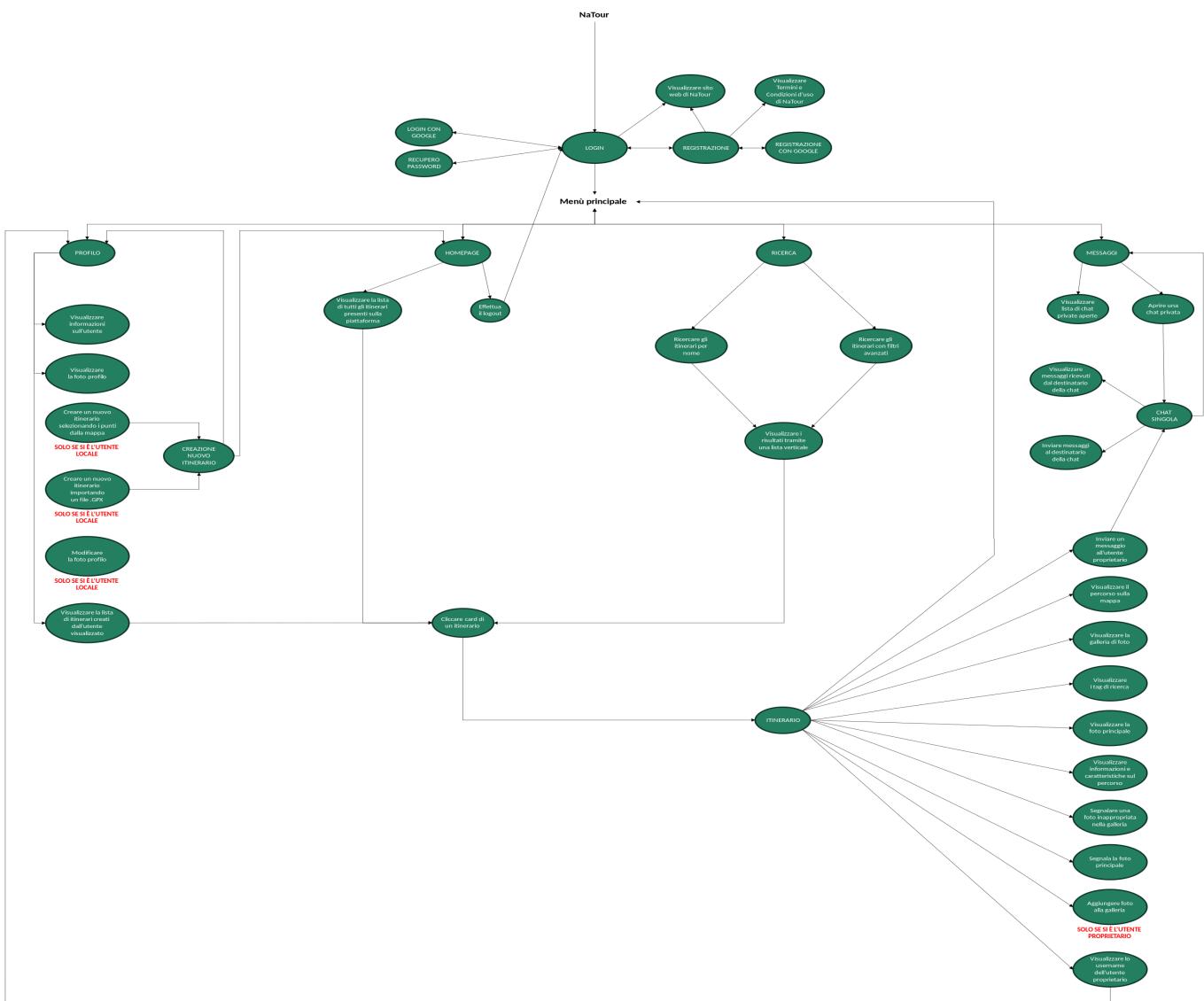
3.6 Definizione delle gerarchie funzionali

Da una attenta analisi svoltasi durante lo sviluppo della piattaforma NaTour è risultato fondamentale tenere traccia degli spostamenti che un utente già registrato o un amministratore dell'app compie durante il suo utilizzo.

Risulta ovvio, quindi, che devono essere evitati casi di inconsistenza in cui l'utente può trovarsi su percorsi errati.

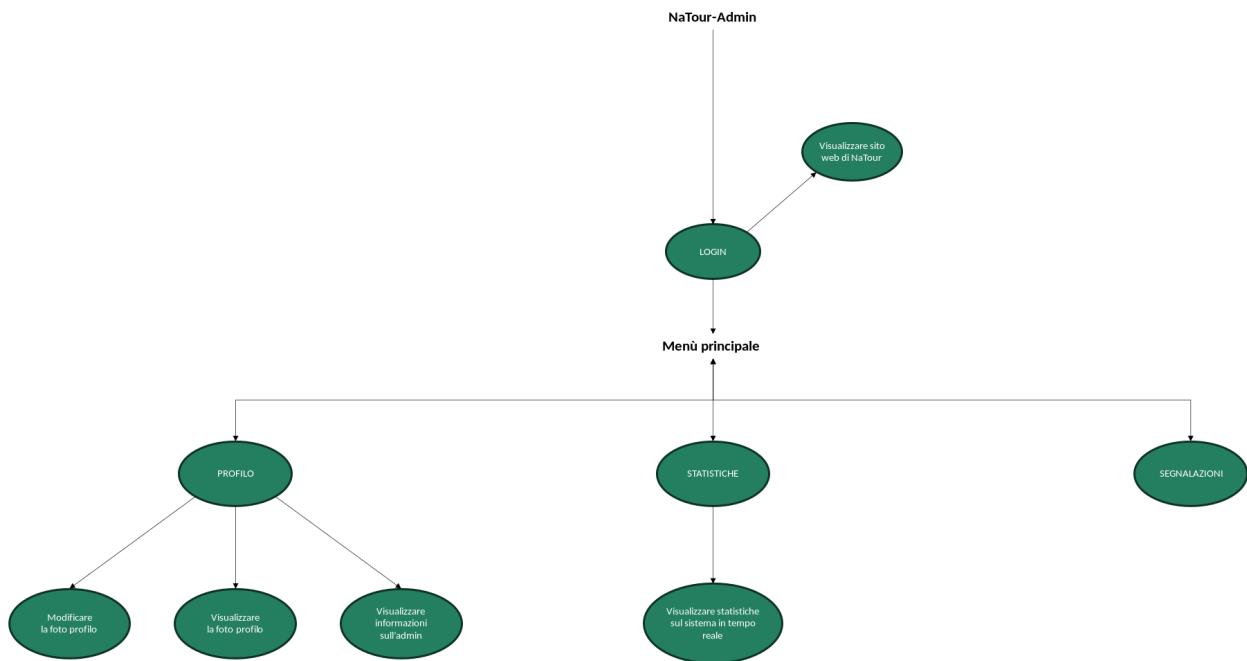
In questo documento riportiamo le gerarchie funzionali dei 2 applicativi sviluppati²¹ :

Gerarchia di NaTour (lato utente)



²¹ Per una visualizzazione migliore delle immagini è possibile accedere alle immagini originali, dal codice sorgente hostato su github (Si faccia riferimento al paragrafo 4 - Codice sorgente sviluppato).

Gerarchia di NaTour-Admin (lato admin)



4 Codice sorgente sviluppato

Il codice sorgente prodotto durante lo sviluppo di *NaTour*[©] è disponibile sulla piattaforma [GitHub](#), che ne ha permesso anche il versionamento.

Essendo molto vicini al concetto di open-source, si è scelto fin da subito un tipo di licenza che fosse in linea con questo principio. È per questo che tutti i programmi necessari alla piattaforma *NaTour* sono sotto licenza [GPL 3.0](#) (che è possibile consultare anche nella repository di NaTour).

Di seguito riportiamo un link per il download: <https://github.com/luftmensch-luftmensch/Progetto-NaTour>²²

²²Potrebbe non essere accessibile a tutti (il repository è per privacy privata).

5 Definizione di un piano di testing

Come si è potuto evincere dai vari seminari durante il corso, definire un piano di testing è una scelta fondamentale per una buona riuscita del software.²³

Durante la fase di implementazione si è deciso quindi di testare funzionalità appartenenti all'applicazione android, funzionalità che ai nostri occhi sono state ritenute essere le più esemplificative.

6 Valutazione sul campo dell'usabilità

6.1 Codice jUnit per Unit testing

Per avere una maggiore certezza della validità e stabilità del nostro applicativo, abbiamo deciso di testare alcuni metodi, che ritieniamo essere punti chiave del suo corretto funzionamento.

Per fare ciò ci siamo affidati al noto framework [JUnit](#),²⁴ arrivato alla versione 5 al tempo della stesura del documento.

6.1.1 Prima esperienza con il framework JUnit

Per essere in grado di eseguire testing nel miglior modo possibile abbiamo deciso di effettuare un primo testing abbastanza semplice, che esegua un paragone tra il package definito e una stringa corrispondente. Di seguito riportiamo la parte rilevante.

```
1 package com.natour;
2
3 import android.content.Context;
4 import android.util.Log;
5
6 import androidx.test.platform.app.InstrumentationRegistry;
7 import androidx.test.ext.junit.runners.AndroidJUnit4;
8
9 import org.junit.Test;
10 import org.junit.runner.RunWith;
11
12 import static org.junit.Assert.*;
13 @RunWith(AndroidJUnit4.class)
14 public class AndroidTesting {
15     ...
16
17     // Controllo che il nome del package corrisponda alla stringa `com.natour`
18     @Test
19     public void useApplicationContext() {
20         Context applicationContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
```

²³ Si vuole a tutti i costi evitare di lanciare sul mercato prodotti non funzionanti o con grosse problematicità.

²⁴ Framework di unit testing per il linguaggio di programmazione Java.

```

21     // Ci aspettiamo di ricevere true come risposta del test
22     assertEquals("com.natour", appContext.getPackageName());
23 }
24
25 ...
26
27 }
```

6.1.2 Test dell'applicativo

Per la scelta dei metodi da testare abbiamo optato per un bacino variegato di funzionalità.

In particolare, andremo a testare:

1. Controllo e test di alcune funzioni (funzioni che sono trasparenti all'utente), in particolare
 - I. Due metodi per la gestione del Database locale [SQLite](#), utilizzato per gestione dei dati dell'utente loggato;
2. Test della funzione che ha lo scopo di controllare la validità dei dati inseriti dall'utente durante la fase di registrazione alla piattaforma (È chiaro che questa fase risulta particolarmente critica, in quanto l'inserimento dei dati in maniera non automatica è fortemente pronata ad errori);
3. Test della funzione che ha lo scopo di controllare la validità dei dati inseriti dall'utente durante la fase di recupero (e quindi cambio) password (È chiaro che questa fase risulta particolarmente critica, in quanto l'inserimento dei dati in maniera non automatica è fortemente pronata ad errori);

Di seguito riportiamo il testing della funzionalità descritta nel punto 1.a:

```

1 package com.natour;
2
3 import android.Manifest;
4 import android.content.Context;
5 import android.util.Log;
6
7 import androidx.core.app.ActivityCompat;
8 import androidx.test.platform.app.InstrumentationRegistry;
9 import androidx.test.ext.junit.runners.AndroidJUnit4;
10
11 import org.junit.Test;
12 import org.junit.runner.RunWith;
13
14 import static org.junit.Assert.*;
15
16 import com.natour.utils.constants.Constants;
17 import com.natour.utils.persistence.LocalUser;
```

```

18 import com.natour.utils.persistence.LocalUserDbManager;
19
20 @RunWith(AndroidJUnit4.class)
21 public class AndroidTesting {
22
23     // Necessari al corretto testing del metodo `inserimentoEControllo`
24     LocalUser localUser;
25     LocalUserDbManager dbManager;
26
27     ...
28
29     ...
30
31 /*
32      Testing dell'utilizzo del database locale.
33
34 Testiamo il metodo eseguendo i seguenti passaggi:
35     1. Inseriamo quindi un utente;
36     2. Controlliamo se il database risulti vuoto (Ci aspettiamo ovviamente che la `response` sia falsa)
37
38 */
39
40 public boolean inserimentoEControllo(int id, String username, String urlFotoProfilo, String
41 isLoggedInWithGoogle){
42     dbManager = new LocalUserDbManager(InstrumentationRegistry.getInstrumentation().
43         getTargetContext());
44     dbManager.openW();
45
46     // Per avere un feedback immediato durante il testing abbiamo deciso di fare uso di Log (che
47     // vengono mostrati durante l'esecuzione del testing)
48     Log.d("Android Testing", "Inizio inserimento dati");
49     dbManager.inserimentoDatiUtente(id, username, urlFotoProfilo, isLoggedInWithGoogle);
50     boolean response = dbManager.isEmpty();
51     Log.d("Android Testing", "Stato del db in seguito all'inserimento: " + response);
52     dbManager.closeDB();
53     return response;
54 }
55
56 // Testiamo quindi il metodo precedentemente definito
57 @Test
58 public void testDatabaseManager(){
59     inserimentoEControllo(1, "AndroidTesting", "AndroidTesting", "No");
60 }

```

Per il testing delle funzionalità descritte nel punto 2 e 3 è necessario fare affidamento ad una classe di mockup:²⁵

```

1 package com.natour;
2
3 import java.util.regex.Pattern;
4
5 public class ControlloCampiMock {
6     public ControlloCampiMock() { }
7
8     // Visto che i Toast non sono mockuppabili andiamo a modificare leggermente il metodo della
9     // SignUpActivity eliminando le chiamate con i Toast
10
11    // Per questo metodo utilizzo black box wect
12    public boolean controlloCampi(String username, String email, String password, String
13        confermaPassword) throws IllegalArgumentException{
14        String regex = "^(\\w+([-+.']\\w+)*@[\\w+([\\.-]*[\\w+])\\w+]+([\\.-][\\w+([\\.-]*[\\w+])\\w+]+)*@([\\w+([\\.-]*[\\w+])\\w+]+([\\.-][\\w+([\\.-]*[\\w+])\\w+]+)*([a-zA-Z0
15        -9-]+\\.)+[a-zA-Z]{2,6}$";
16        Pattern pattern = Pattern.compile(regex);
17
18        if((username == null) || (email == null) || (password == null) || (confermaPassword == null))
19            {
20                throw new IllegalArgumentException();
21            }
22        if((username.isEmpty()) || (email.isEmpty()) || (password.isEmpty()) || (confermaPassword.
23            isEmpty())){
24                return false;
25            }
26        if(!pattern.matcher(email).matches()){
27            return false;
28        }
29
30        if (password.length() < 8) {
31            return false;
32        }
33
34        if(!password.matches(".*[0-9].*") | !password.matches(".*[A-Z].*") | !password.matches("
35            ^(?=.*[_().$%&@]).*$")){
36            return false;
37        }
38    }

```

²⁵Gli unit test si concentrano su un particolare pezzo di codice che deve essere esercitato. Nella maggior parte dei casi, questo codice si basa su dipendenze esterne. Tali dipendenze devono essere controllate, in modo tale che solo il codice sotto testing sia effettivamente testato. La rimozione delle dipendenze viene eseguita con l'introduzione delle classi Mock, che le simulano.

```

33     if(!password.equals(confermaPassword)){
34         return false;
35     }
36     return true;
37 }

38

39 // Visto che il Toast non sono mockuppabili ricopio il metodo della LoginActivity eliminando le
40 // chiamate con i Toast
41 // Metodo che controlla che la nuova password sia valida (utilizziamo sempre la black box wect)
42 public boolean controlloRecuperoPassword(String nuovaPassword, String codiceConferma){
43     if ((nuovaPassword.isEmpty()) || (codiceConferma.isEmpty())) {
44         return false;
45     }
46     if (nuovaPassword.length() < 8) {
47         return false;
48     }
49     if (!nuovaPassword.matches(".*[0-9].*") | !nuovaPassword.matches(".*[A-Z].*") | !
50         nuovaPassword.matches("^(?=.*[_().$&@]).*\$")) {
51         return false;
52     }
53     return true;
54 }

```

Per il testing di questo metodo si è deciso di optare per il testing **black box**.²⁶

Il metodo in esame prende in input 4 parametri di tipo **stringa** (username, email, password, confermaPassword). Sono state quindi individuate le seguenti classi di equivalenza:

- ▶ CEU1 → Parametro **username** valido;
- ▶ CEU2 → Parametro **username** vuoto;
- ▶ CEU3 → Parametro **username** null;
- ▶ CEE1 → Parametro **email** valida;
- ▶ CEE2 → Parametro **email** vuota;
- ▶ CEE3 → Parametro **email** null;
- ▶ CEE4 → Parametro **email** che non contiene il carattere @;
- ▶ CEE5 → Parametro **email** che non contiene il dominio;
- ▶ CEP1 → Parametro **password** valida;
- ▶ CEP2 → Parametro **password** vuota;

²⁶Testing basato sulle specifiche del programma e sulla conoscenza delle sole funzionalità e sulla ricerca dell'affidabilità e dell'efficienza.

- ▶ CEP3 → Parametro `password` null;
- ▶ CEP4 → Parametro `password` più corta di 8 caratteri;
- ▶ CEP5 → Parametro `password` che non contiene numeri;
- ▶ CEP6 → Parametro `password` che non contiene caratteri speciali;
- ▶ CEP7 → Parametro `password` che non contiene una lettera maiuscola;
- ▶ CECP1 → Parametro `confermaPassword` che combacia con il parametro `password`;
- ▶ CECP2 → Parametro `confermaPassword` che non combacia con il parametro `password`;
- ▶ CECP3 → Parametro `confermaPassword` vuota;
- ▶ CECP4 → Parametro `confermaPassword` nulla;

Come già scritto nel commento a riga 10 della classe `ControlloCampiMock`, procederemo con una selezione di casi di test, basata su una combinazione delle classi di equivalenza individuate, adottando il criterio di copertura WECT (*Weak Equivalence Class Testing*). Con questo criterio, ci limiteremo a scrivere precisamente un caso di test per ogni classe di equivalenza non valida per ogni parametro in input, ed un caso di test per ogni classe di equivalenza valida del parametro che ha il massimo numero di classi di equivalenza valide tra tutti gli altri. Nel nostro caso, abbiamo scelto:

- ▶ Il caso di test `public void testRegistrazioneValida()`, che verifichi contemporaneamente la validità delle classi `CEU1`, `CEE1`, `CEP1`, `CECP1`;
- ▶ Il caso di test `public void testUsernameVuoto()` che verifichi la validità della sola classe `CEU2`;
- ▶ Il caso di test `public void testUsernameNull()` che verifichi la validità della sola classe `CEU3`;
- ▶ Il caso di test `public void testEmailVuota()` che verifichi la validità della sola classe `CEE2`;
- ▶ Il caso di test `public void testEmailNull()` che verifichi la validità della sola classe `CEE3`;
- ▶ Il caso di test `public void testEmailSenzaChiocciola()` che verifichi la validità della sola classe `CEE4`;
- ▶ Il caso di test `public void testEmailSenzaDominio()` che verifichi la validità della sola classe `CEE5`;
- ▶ Il caso di test `public void testPasswordVuota()` che verifichi la validità della sola classe `CEP2`;
- ▶ Il caso di test `public void testPasswordNulla()` che verifichi la validità della sola classe `CEP3`;
- ▶ Il caso di test `public void testPasswordBreve()` che verifichi la validità della sola classe `CEP4`;
- ▶ Il caso di test `public void testPasswordSenzaNumeri()` che verifichi la validità della sola classe `CEP5`;
- ▶ Il caso di test `public void testPasswordSenzaCaratteriSpeciali()` che verifichi la validità della sola classe `CEP6`;
- ▶ Il caso di test `public void testPasswordSenzaLettereMaiuscole()` che verifichi la validità della sola classe `CEP7`;
- ▶ Il caso di test `public void testPasswordMatchNonValido()` che verifichi la validità della sola classe `CECP2`;
- ▶ Il caso di test `public void testConfermaPasswordVuota()` che verifichi la validità della sola classe `CECP2`;
- ▶ Il caso di test `public void testConfermaPasswordNulla()` che verifichi la validità della sola classe `CECP2`;

Otteniamo, quindi, i seguenti 14 casi di test. Abbiamo preferito l'utilizzo di questa strategia rispetto ad altre molto più robuste e sicure come SECT (*Strong Equivalence Class Testing*) o WCT (*Worst Case Testing*) per non ottenere un numero di casi di test esponenziale sul numero di parametri.

```
1 package com.natour;
2
3 import org.junit.Before;
4 import org.junit.Test;
5
6 import static org.junit.Assert.*;
7
8 /**
9  * Classe di test per il controllo della validità dei campi
10 */
11 public class ControlloCampiTest {
12     ControlloCampiMock controlloCampiMock;
13
14     @Before
15     public void setup(){
16         controlloCampiMock = new ControlloCampiMock();
17     }
18
19     // Test sullo username
20     @Test
21     public void testRegistrazioneValida(){
22         assertTrue(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", "Prova123@", "Prova123@"));
23     }
24
25     @Test
26     public void testUsernameVuoto(){
27         assertFalse(controlloCampiMock.controlloCampi("", "valentinobocchetti59@gmail.com", "Prova123@", "Prova123@"));
28     }
29
30     @Test(expected = IllegalArgumentException.class)
31     public void testUsernameNull(){
32         controlloCampiMock.controlloCampi(null, "valentinobocchetti59@gmail.com", "Prova123@", "Prova123@");
33     }
34
35     // Test sulla mail
36     @Test
37     public void testEmailVuota(){
38         assertFalse(controlloCampiMock.controlloCampi("", "", "Prova123@", "Prova123@"));
39     }
```

```

40
41     @Test(expected = IllegalArgumentException.class)
42     public void testEmailNull(){
43         controlloCampiMock.controlloCampi("test123", null, "Prova123@", "Prova123@");
44     }
45
46     @Test
47     public void testEmailSenzaChiocciola(){
48         assertFalse(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", "
49             Prova123@", "Prova123@"));
50     }
51
52     @Test
53     public void testEmailSenzaDominio(){
54         assertFalse(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@", "Prova123@", "
55             "Prova123@"));
56     }
57
58     // Test sulla password
59     @Test
60     public void testPasswordVuota(){
61         assertFalse(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", "", "
62             "Prova123@"));
63     }
64
65     @Test(expected = IllegalArgumentException.class)
66     public void testPasswordNulla(){
67         controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", null, " "
68             Prova123@");
69     }
70
71     @Test
72     public void testPasswordBreve(){
73         assertFalse(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", " "
74             P123@", "Prova123@"));
75     }
76
77     @Test
78     public void testPasswordSenzaNumeri(){
79         assertFalse(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", " "
80             prova", "Prova123@"));
81     }
82
83     @Test
84     public void testPasswordSenzaCaratteriSpeciali(){
85         assertFalse(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", " "
86             prova!@#$%^&*()_+=~`{|}"));
87     }

```

```

    prova12", "Prova123@"));
}

81
82 @Test
83 public void testPasswordSenzaLettereMaiuscole(){
84     assertFalse(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", "
85         prova123@", "Prova123@"));
86 }
87
88 // Test su confermaPassword
89 @Test
90 public void testPasswordMatchNonValido(){
91     assertFalse(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", "
92         Prova123@", "Prova1234@"));
93 }
94
95 @Test
96 public void testConfermaPasswordVuota(){
97     assertFalse(controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", "
98         Prova123@", ""));
99 }
100
101 @Test(expected = IllegalArgumentException.class)
102 public void testConfermaPasswordNulla(){
103     controlloCampiMock.controlloCampi("test123", "valentinobocchetti59@gmail.com", "Prova123@", "
104         null");
105 }
106 }
```

Passiamo adesso invece all'analisi dell'ultimo metodo che si è scelto di testare (anche questo mediante testing black box).

Il metodo prende in input 2 parametri di tipo **stringa** (nuovaPassword e codiceConferma). Sono state quindi individuate le seguenti classi di equivalenza:

- ▶ CEP1 → Parametro **nuovaPassword** valida;
- ▶ CEP2 → Parametro **nuovaPassword** vuota;
- ▶ CEP3 → Parametro **nuovaPassword** che ha una lunghezza inferiore agli 8 caratteri;
- ▶ CEP4 → Parametro **nuovaPassword** che non contiene numeri;
- ▶ CEP5 → Parametro **nuovaPassword** che non contiene maiuscole;
- ▶ CEP6 → Parametro **nuovaPassword** che non contiene caratteri speciali;
- ▶ CECC1 → Parametro **codiceConferma** valido;
- ▶ CECC2 → Parametro **codiceConferma** non valido;

Da queste classi di equivalenza andremo a testare i seguenti metodi:

- ▶ Il caso di test `public void testPasswordValid()` che verifichi la validità delle classi CEP1 e CECC1;
- ▶ Il caso di test `public void testPasswordVuotaControlloRecuperoPassword()` che verifichi la validità della sola classe CEP2;
- ▶ Il caso di test `public void testCodiceConfermaVuotoControlloRecuperoPassword()` che verifichi la validità della sola classe CECC2;
- ▶ Il caso di test `public void testPasswordBreveControlloRecuperoPassword()` che verifichi la validità della sola classe CEP3;
- ▶ Il caso di test `public void testPasswordSenzaNumeriControlloRecuperoPassword()` che verifichi la validità della sola classe CEP4;
- ▶ Il caso di test `public void testPasswordSenzaMaiuscoleControlloRecuperoPassword()` che verifichi la validità della sola classe CEP5;
- ▶ Il caso di test `public void testPasswordSenzaCaratteriSpecialiControlloRecuperoPassword()` che verifichi la validità della sola classe CEP6.

Otteniamo quindi, i seguenti 7 casi di test. Come per il testing precedente abbiamo preferito l'utilizzo di questa strategia rispetto ad altre molto più robuste e sicure come SECT (*Strong Equivalence Class Testing*) o WCT (*Worst Case Testing*) per non ottenere un numero di casi di test esponenziale sul numero di parametri.

```
1 package com.natour;
2
3 import org.junit.Before;
4 import org.junit.Test;
5
6 import static org.junit.Assert.*;
7
8 /**
9  * Classe di test per il controllo della validità dei campi
10 */
11 public class ControlloCampiTest {
12     ControlloCampiMock controlloCampiMock;
13
14     @Before
15     public void setup(){
16         controlloCampiMock = new ControlloCampiMock();
17     }
18
19     ...
20
21     ...
22 }
```

```

23     // Testing del metodo controlloRecuperoPassword
24
25     @Test
26     public void testPasswordValida(){
27         assertTrue(controlloCampiMock.controlloRecuperoPassword("Prova123@", "12345"));
28     }
29
30     // Test del caso in cui la password sia vuota
31
32     @Test
33     public void testPasswordVuotaControlloRecuperoPassword(){
34         assertFalse(controlloCampiMock.controlloRecuperoPassword("", "12345"));
35     }
36
37     // Test del caso in cui il codice di conferma sia vuoto
38
39     @Test
40     public void testCodiceConfermaVuotoControlloRecuperoPassword(){
41         assertFalse(controlloCampiMock.controlloRecuperoPassword("Prova123@", ""));
42     }
43
44     // Test del caso in cui la password sia troppo corta
45
46     @Test
47     public void testPasswordBreveControlloRecuperoPassword(){
48         assertFalse(controlloCampiMock.controlloRecuperoPassword("Pro123@", "12345"));
49     }
50
51     // Test del caso in cui la password non contenga numeri
52
53     @Test
54     public void testPasswordSenzaNumeriControlloRecuperoPassword(){
55         assertFalse(controlloCampiMock.controlloRecuperoPassword("Provatest@", "12345"));
56     }
57
58     // Test del caso in cui la password non contenga lettere maiuscole
59
60     @Test
61     public void testPasswordSenzaMaiuscoleControlloRecuperoPassword(){
62         assertFalse(controlloCampiMock.controlloRecuperoPassword("prova123@", "12345"));
63     }
64

```

6.2 Valutazione sull'usabilità sul campo, realizzata sul prodotto finito

Sommario

Come già detto nel punto 2.1.9 di questo documento, la valutazione sul campo dell'usabilità consiste nel far eseguire ad un campione di utenti, rappresentativo del target identificato al punto 2.1.7, un insieme di compiti tipici di utilizzo del sistema in un ambiente controllato da uno o più osservatori (esperti di usabilità).

Metodologia utilizzata

L'interazione tra sistema e utente, in questa situazione, può avvenire attraverso varie tecniche:

- ▶ **Utilizzo di prototipi di carta con tecnica del mago di Oz**

È necessaria una terza persona, ad esempio un progettista, che conosca egregiamente il sistema e ne simuli il comportamento "di nascosto".

- ▶ **Utilizzo di prototipi interattivi**

Si utilizzano applicativi software come Axure o Figma. Questi permettono la realizzazione di prototipi precisi e di alta qualità connessi tra loro tramite una serie di transizioni che consentono agli utenti di valutare il sistema.

- ▶ **Utilizzo del prodotto software in fase di beta testing**

Dopo aver terminato le fasi di verifica con Unit Testing, Integration Testing e System Testing, si entra nelle fasi di validazione composte da Acceptance Testing, Usability Testing e Installation Testing.

Per questa fase di valutazione sul campo dell'usabilità, abbiamo pensato di chiedere ai nostri valutatori di valutare direttamente il prodotto finito in fase di validazione attraverso uno Usability Test composto solo da test sommativi, in modo che vengano rilevati i punti deboli e vengano fornite indicazioni concrete per il miglioramento dell'interfaccia utente del sistema prima del rilascio finale.

Scelta dei valutatori e dell'ambiente

- ▶ Abbiamo scelto i valutatori in modo che coprano quanti più tipi di utenti del nostro target, basandoci fortemente sul livello di conoscenza del dominio del sistema (nel nostro caso trekking, passeggiate outdoor, escursioni, ecc.) ma anche sul livello di familiarità con la tecnologia (es. applicazioni per smartphone, social networking, ...). La scelta finale è rappresentata nella seguente tabella



Dimensioni del profilo degli utenti

► Compiti assegnati

Per effettuare il test di usabilità, abbiamo preso in considerazione le seguenti attività da far svolgere ai valutatori. Nella seguente lista sono indicate l'attività svolta, un tempo massimo preventivo per lo svolgimento dell'attività ed un esempio delle loro risposte:

- Compito 1. Registrati alla piattaforma NaTour [10 minuti]

Eliana De Amicis, nonostante abbia un basso livello di familiarità con le tecnologie mobile, dice di essersi sentita guidata durante tutta l'operazione, soprattutto grazie ai popup auto esplicativi e a dei colori di sfondo piacevoli. Hanno completato tutti l'attività con successo.

- Compito 2. Ricerca un itinerario in collina [5 minuti]

Miriam Acunzo non è riuscita a completare l'operazione nel tempo previsto siccome, inizialmente, non ha capito come attivare la ricerca con filtri avanzati. Per questo motivo, l'attività viene considerata un successo parziale. Successivamente, le è stato spiegato che il menù in alto a sinistra poteva essere "tenuto premuto" per mostrare un feedback che ne spiegasse il funzionamento.

- Compito 3. Apri la mappa del percorso di un itinerario [10 minuti]

Gli osservatori hanno notato che Eliana De Amicis durante lo svolgimento di questo compito ha riscontrato alcune difficoltà nell'apertura della schermata di dettaglio di un itinerario siccome provava a tener premuto sulla card del singolo itinerario.

- Compito 4. Crea un nuovo itinerario che duri più di 3 ore [15 minuti]

Cosimo Imparato ritiene che la mappa dove selezionare i punti non fornisca il giusto supporto perché alcune delle caratteristiche del percorso, come lunghezza o dislivello, le ha dovute calcolare personalmente attraverso delle ricerche più approfondite.

- Compito 5. Visualizza il numero di utenti totali registrati alla piattaforma NaTour [5 minuti]

Tutti i valutatori sono stati in grado di completare quest'attività con successo nel tempo preventivato.

- Compito 6. Invia un messaggio ad un utente a cui non ha mai inviato un messaggio [15 minuti]

Miriam Acunzo era inizialmente perplessa sul motivo per cui non fosse presente un pulsante per inviare un nuovo messaggio direttamente nella schermata delle chat room. Successivamente, i valutatori le hanno spiegato che NaTour non è un software dedicato alla messaggistica e che, quindi, la possibilità di creare nuovi messaggi è presente solo nella schermata di dettaglio di un itinerario e, da lì, mettersi in contatto direttamente con il creatore dell'itinerario.

Sintesi delle misure e analisi dei risultati

Oltre ad un'osservazione qualitativa nel quale i valutatori hanno espresso il loro parere personale riguardo le attività proposte, durante il nostro test di usabilità ci siamo concentrati soprattutto anche sull'identificazione ed il raccoglimento di alcune misure oggettive. Durante la fase di progettazione, quando abbiamo pensato come svolgere gli Usability Test, abbiamo identificato le seguenti misure:

- ▶ Il tasso di successo, cioè la percentuale di attività che sono state portate a termine con successo, tenendo conto anche del tempo impiegato da ogni valutatore per l'esecuzione di una singola attività;
- ▶ L'analisi dei file di log ottenuti durante lo svolgimento delle attività di test.

Il calcolo del tasso di successo è molto utile per validare la semplicità e la comprensibilità dell'interfaccia utente del sistema. È una misura percentuale che può tener conto anche dei compiti eseguiti solo parzialmente oppure dei compiti eseguiti in un tempo troppo lungo rispetto al tempo preventivato. I risultati del calcolo del tasso di successo sono mostrati nella seguente tabella:

	Compito 1	Compito 2	Compito 3	Compito 4	Compito 5	Compito 6
Cosimo Imparato	S	S	S	P	S	S
Maurizio Caccavale	S	S	S	S	S	S
Eliana De Amicis	S	S	P	P	S	P
Miriam Acunzo	S	P	S	S	S	P

Legenda: (S) = Successo, (P) = Successo parziale, (F) = Fallimento

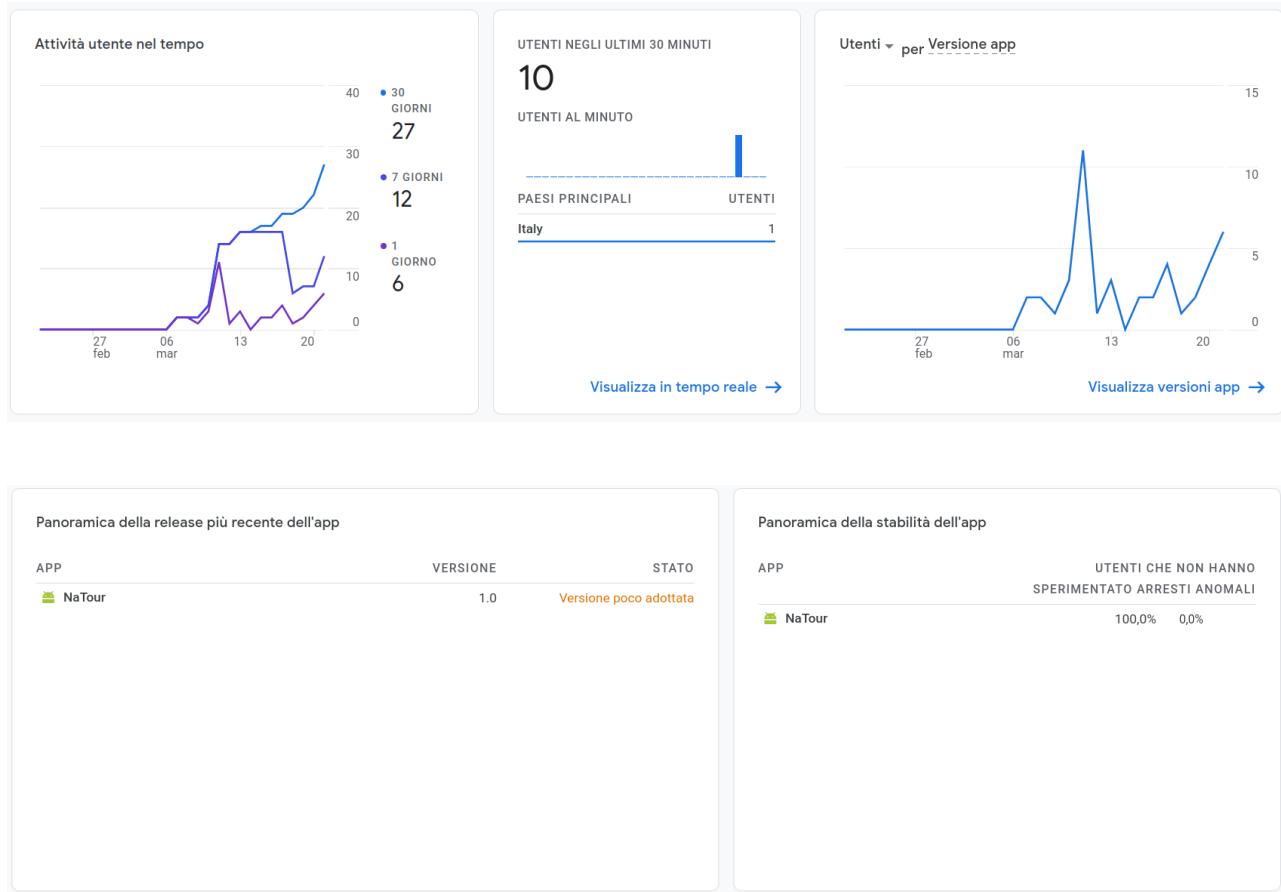
dei risultati di un test di usabilità

Sintesi

Assegnando il valore 1 per le attività svolte con successo (S) e il valore 0.5 per le attività svolte parzialmente o completate oltre il tempo previsto (P), il tasso di successo può essere calcolato come il rapporto tra la somma del valore assegnato alle attività S e il valore assegnato alle attività P, e il numero totale di attività eseguite, cioè:

$$\text{Tasso di successo} = \frac{(19 * 1 + 5 * 0.5)}{24} = \approx 90\%$$

Come detto prima, abbiamo accompagnato l'esecuzione dei test di usabilità dalla registrazione (logging) automatica degli eventi attivati dalle funzionalità di *Amplify* e dalle richieste *HTTP* utilizzando il framework *Google Analytics for Mobile Apps*, che fornisce informazioni dettagliate sull'utilizzo del prodotto software da parte degli utenti. Di seguito, vi riportiamo alcuni estratti presi dalle statistiche di utilizzo, accessibili dalla console di *FireBase*:



Di seguito invece vengono riportati i report che contengono informazioni su:

1. Fidelizzazione;
2. Modelli di smartphone che i nostri utenti utilizzano;
3. Le schermate su cui i nostri utenti passano la maggior parte del tempo.

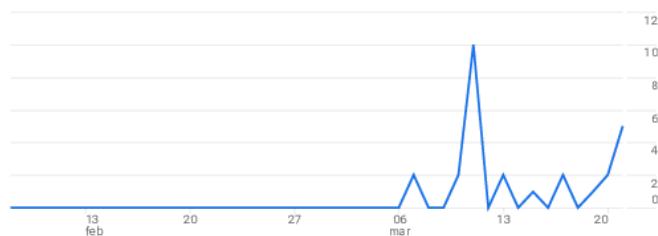
Panoramica fidelizzazione ()

Personalizzato 8 feb - 21 mar 2022

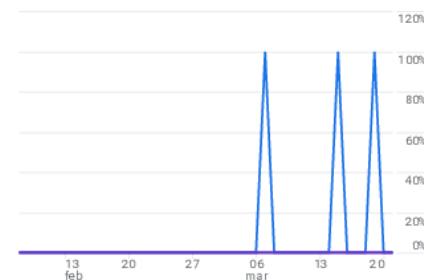
 Tutti gli utenti Aggiungi confronto

Nuovi utenti
27

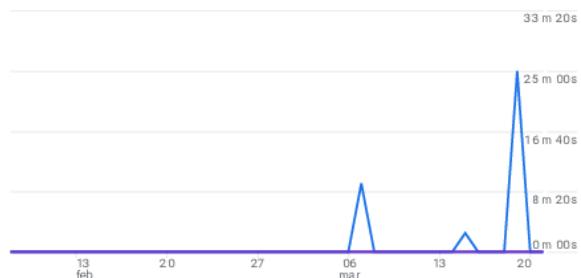
Utenti di ritorno
11



Fidelizzazione utenti per coorte

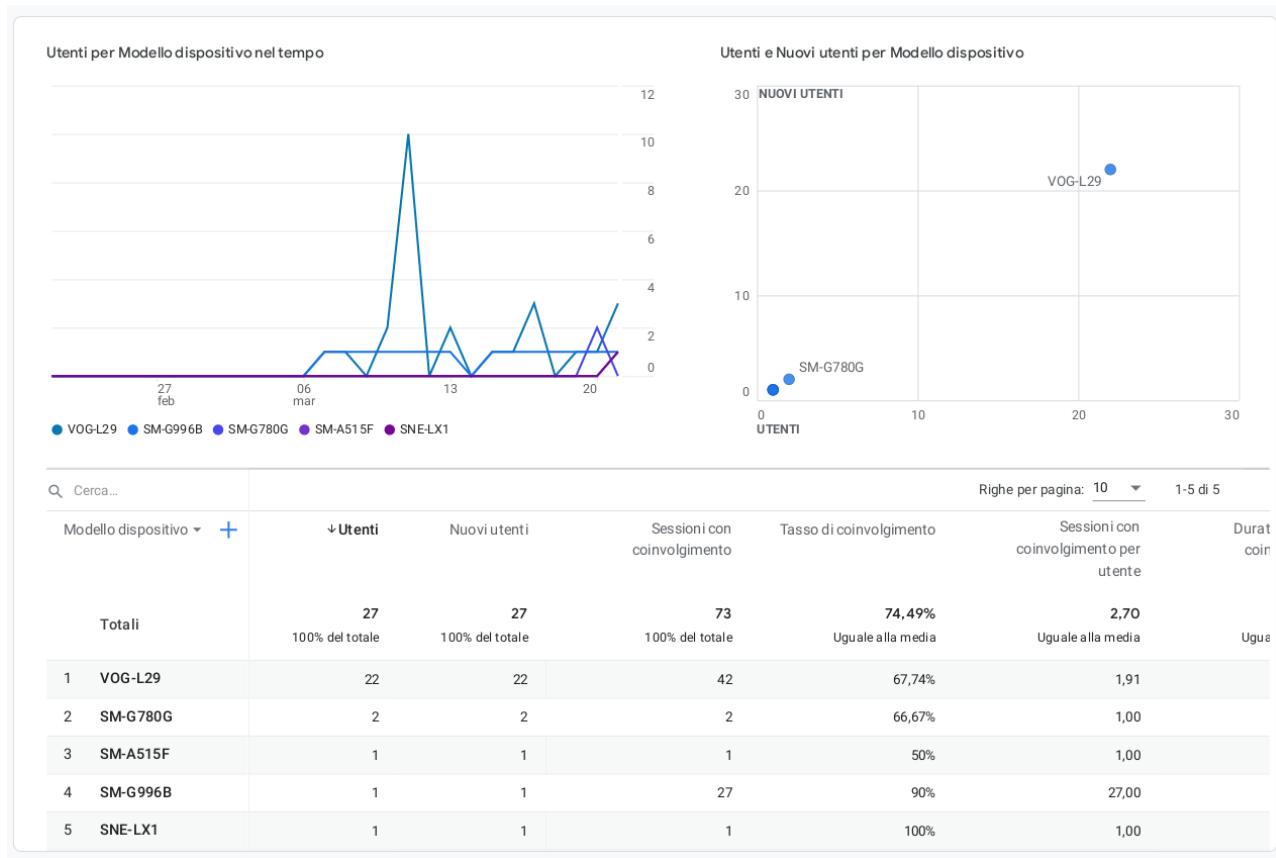
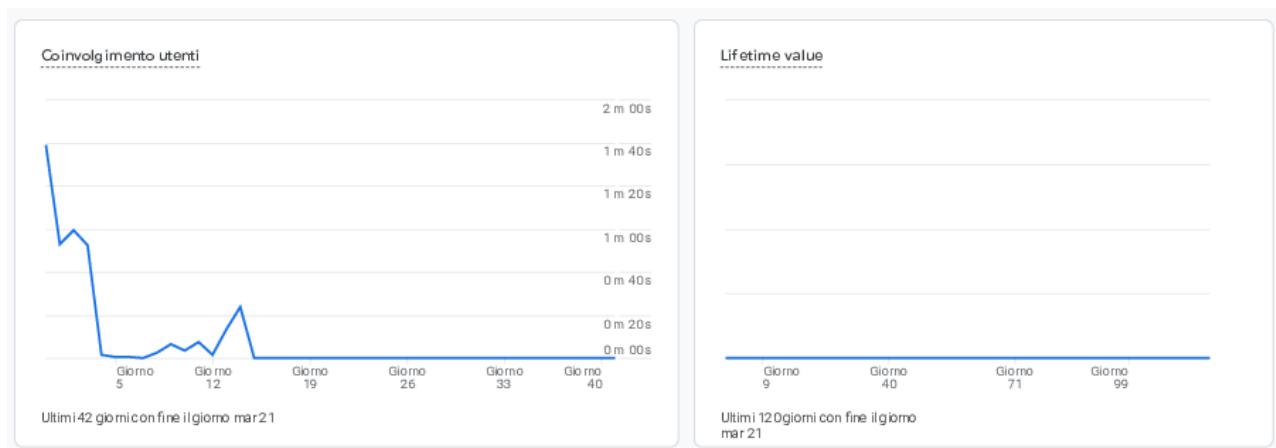


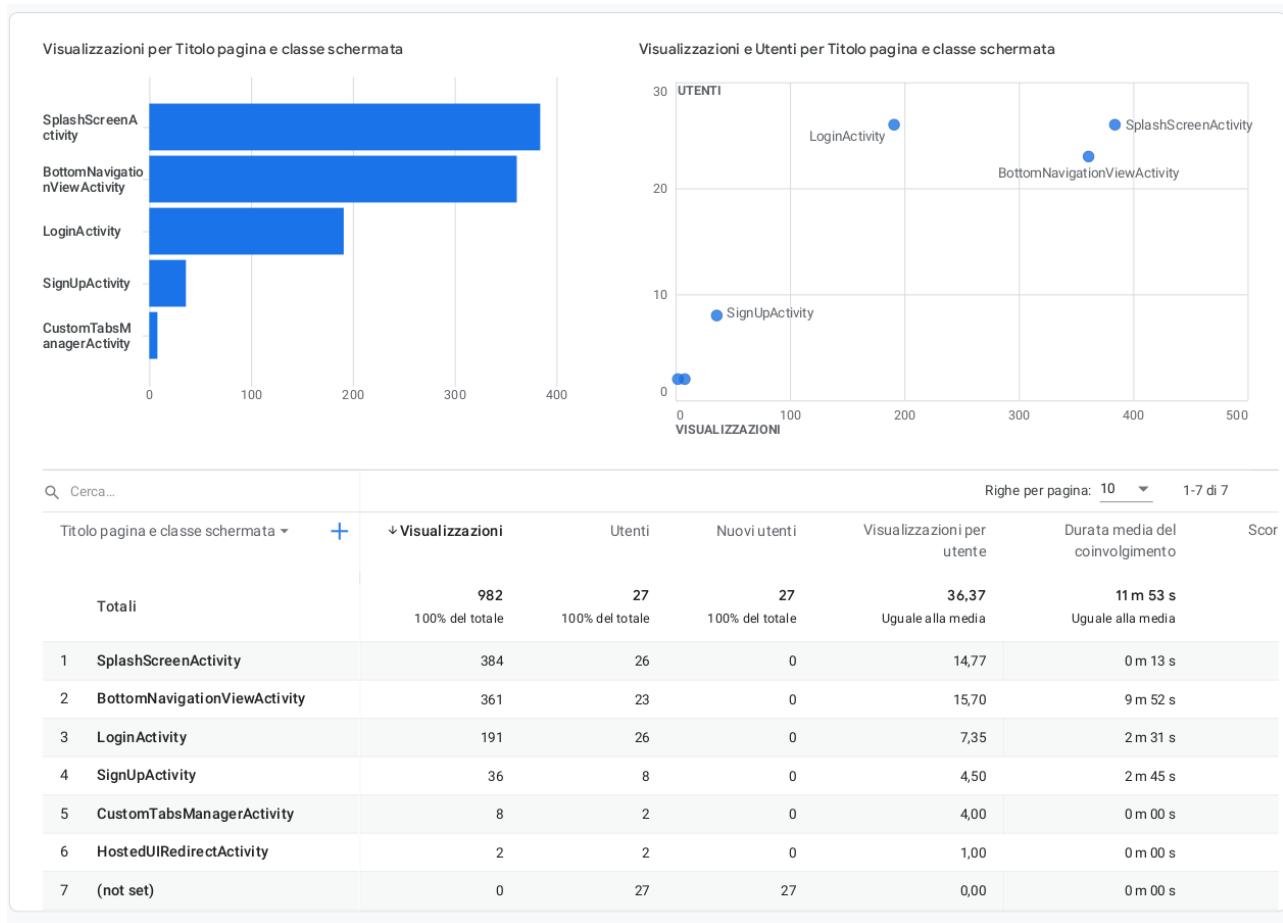
Coinvoltimento utenti per coorte



Fidelizzazione utenti



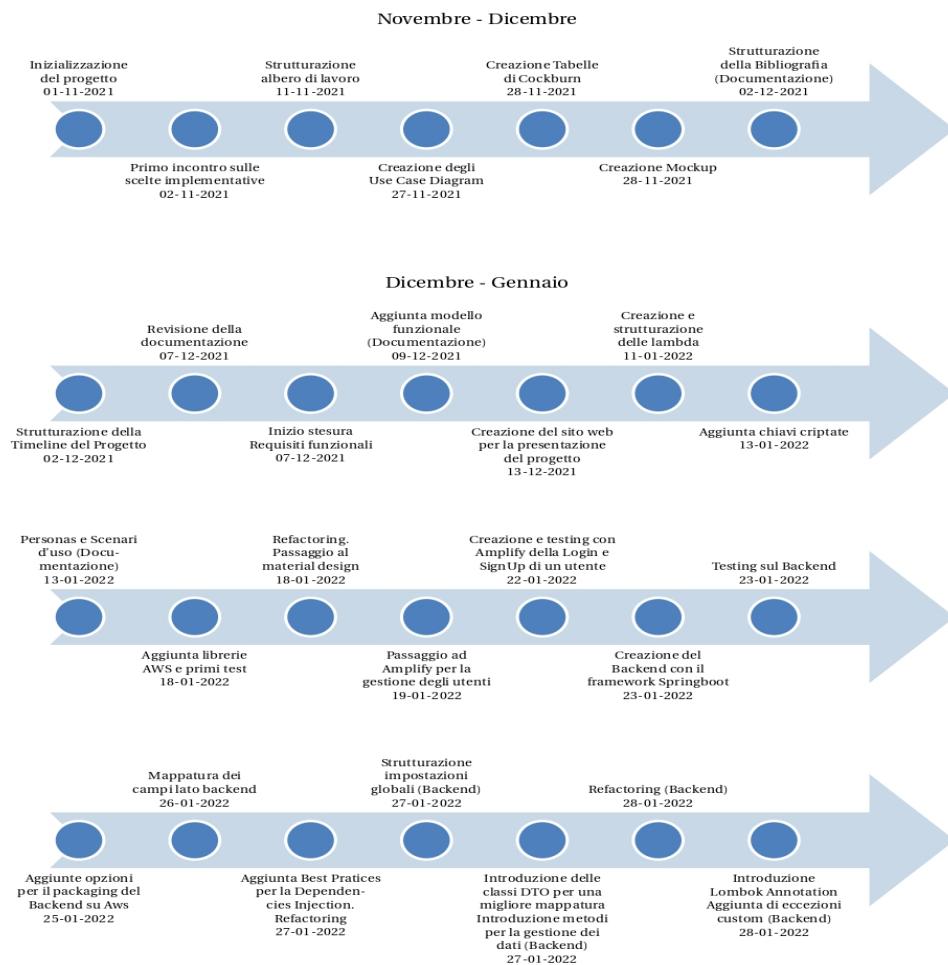




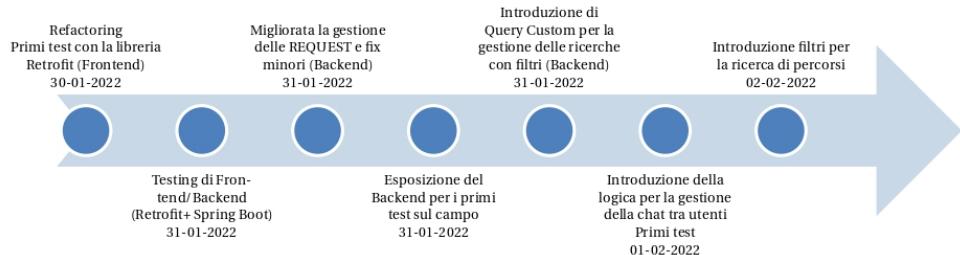
Per avere una visione più dettagliata di file di logging dell'applicazione durante la sua esecuzione su un dispositivo reale durante questa fase di beta testing del prodotto software si faccia riferimento al file di log in allegato alla documentazione.

7 Analisi dei tempi di sviluppo

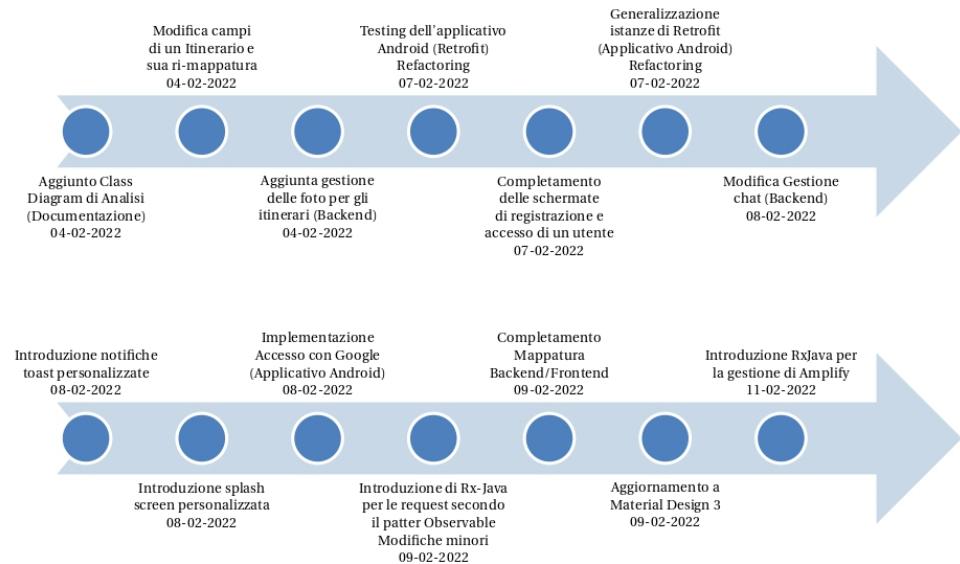
Di seguito viene riportata lo storico dei tempi di sviluppo sotto forma di timeline con una vista mese per mese i passaggi chiave dello sviluppo di *NaTour*[®]



Gennaio - Febbraio



Febbraio -



Introduzione MVP
Refactoring
Modifiche minori
13-02-2022

Completamento transizione da Amplify ad RxJava-Amplify Passaggio da Activity a Fragment 11-02-2022

Creazione popup per la gestione dei termini e delle condizioni 13-02-2022

8 Contributori



Informazioni:

- ▶ Valentino Bocchetti
- ▶ vale.bocchetti@studenti.unina.it
- ▶ N86003405



Informazioni:

- ▶ Mario Gabriele Carofano
- ▶ m.carofano@studenti.unina.it
- ▶ N86003228

9 Riferimenti

- ▶ Identificare i target per le ricerche di mercato;
- ▶ Definizione di target.