

UNIVERSITÀ DEGLI STUDI DI
NAPOLI FEDERICO II



Corso Di Ingegneria Del Software

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE
(L-31)

NATOUR21

Studenti:

Ciccarelli Francesco – N86003285

Ciacciarella Alex – N86003179

Gruppo 34

ANNO ACCADEMICO 2021/2022

Sommario

Sommario.....	1
1. Introduzione del software.....	3
1.1 Presentazione Idea Progettuale.....	3
2. Modello Funzionale.....	6
2.1 Requisiti funzionali.....	6
2.2 Requisiti non funzionali.....	13
2.3 Requisiti di Dominio	16
2.4 Modellazione dei casi d'uso.....	17
2.5 Tabelle di Cokburn per 2 casi d'uso	22
2.5.1 Tabella di Cockburn per l'inserimento di un itinerario	22
2.5.2 Tabella di Cockburn per la ricerca di un itinerario	26
2.6 Prototipazione via Mock-Up.....	29
2.7 Individuazione del target utenti.....	40
2.8 User Personas	43
2.9 Valutazione Usabilità a Priori	46
2.10 Prototipazione funzionale via Statechart dell'interfaccia grafica	52
2.10.1 Statechart Inserimento percorso.....	52
2.10.2 Statechart Ricerca Percorso	53
3. Modelli di Dominio.....	54
3.1 Classi, oggetti e relazioni d'analisi.....	54
3.1.2 Class Diagram – Registrazione.....	55
3.1.3 Class Diagram – Login.....	56
3.1.4 Class Diagram – Homepage	57
3.1.5 Class Diagram – Inserimento Percorso	58
3.1.6 Class Diagram - Dettagli Percorso.....	59
3.1.7 Class Diagram – E-mail Promozionale	60
3.1.8 Class Diagram – Modifica/Elimina Percorso	61
3.2 Sequence Diagram.....	62
3.2.1 Sequence Diagram – Cerca Percorso	62
3.2.2 Sequence Diagram – Crea Percorso.....	63
3.3 Activity Diagram	64

3.3.1 Registrazione Utente e Log-In.....	64
3.3.2 Crea Itinerario.....	65
3.3.3 Cerca Itinerario	66
3.3.4 Caricamento Foto.....	67
3.3.5 Opinioni Percorso	68
3.3.6 Mail Promozionali	69
3.3.7 Modifica Percorso	70
3.3.8 Elimina Percorso	71
4. Documento Di Design Del sistema.....	72
4.1 Analisi dell'architettura con esplicita definizione dei criteri di design	72
4.2 Diagramma delle Classi di Design	80
4.2.1 Class Diagram – Registrazione.....	80
4.2.2 Class Diagram – Login.....	81
4.2.3 Class Diagram – Homepage	82
4.2.4 Class Diagram – Inserimento Percorso	83
4.2.5 Class Diagram – Dettagli Sentiero.....	84
4.2.6 Class Diagram – E-mail Promozionale	85
4.2.7 Class Diagram – Modifica/Elimina Percorso	86
4.3 Diagrammi di Sequenza di Design per 2 casi d'uso significativi.....	87
4.4 Definizione delle Gerarchie Funzionali.....	89
4.4.1 Gerarchia Funzionale Utente	90
4.4.2 Gerarchia funzionale Amministratore	91
4.4.3 Gerarchia Funzionale Guest	92
5. Codice Sorgente Sviluppato.....	93
6. Definizione di un piano di testing e valutazione sul campo dell'usabilità....	94
6.1 convertHourandMinuteToMinute	94
6.2 isEmailCorrectAndPasswordCorrectandEqual	99
6.3 Data Sentiero To UI.....	104
6.4 Valutazione usabilità sul campo.....	108
6.4.1 Valutazione Euristica	108
6.4.2 Analisi di Logging	110

1. Introduzione del software

1.1 Presentazione Idea Progettuale

NaTour21 è un sistema per dispositivi mobile Android che mira a creare una nuova piattaforma social sia per i più esperti appassionati di escursioni, sia per le persone che hanno intenzione di fare una piccola gita giornaliera.

Nell'applicazione si avranno a disposizione degli strumenti utili per creare, visualizzare e condividere percorsi naturali di qualsiasi tipo e per qualsiasi appassionato di escursioni e non!

L'applicazione è in sviluppo per smartphone Android, al momento, ma si prevedono altre implementazioni per altri sistemi operativi.

L'offerta applicativa può essere usufruita attraverso una registrazione alla piattaforma, dove il sistema richiederà alcuni dati per identificarvi all'interno del social network. Per l'accesso sarà anche possibile richiedere una registrazione attraverso Facebook (Meta) e Google.

La filosofia dell'applicazione volge alla semplicità d'uso e al minimalismo così da non rendere troppo ambiguo e tortuoso l'uso. Infatti, l'uso di UX/UI è molto intuitivo e moderno, che permette in pochi secondi di orientarsi perfettamente all'interno dell'applicazione e goderne l'esperienza.

La home è stata studiata per essere il cuore dell'applicazione; lì troveremo i percorsi, in ordine di pubblicazione, modificati a seconda dei filtri inseriti dall'utente. Sempre nella stessa schermata sarà possibile inserire un nuovo percorso andando ad aggiungere tutti i dettagli possibili, dal nome, inizio e fine del

percorso, alla rappresentazione su mappa bidimensionale dello stesso, con foto ed altro.

Inoltre, la ricerca e l'inserimento prevederà una sezione per disabili così da permettere l'accesso alla piattaforma e alle strutture anche per loro.

L'applicazione sarà totalmente social-like avendo la possibilità di interagire con i creatori di contenuti aggiungendo delle foto al loro stesso sentiero. Inoltre, sarà possibile caricare itinerari in formato mappe per avere un'informazione rapida e veloce sul sentiero e le varie diramazioni che può prendere.

Raccolta Requisiti:

- Il sistema deve permettere agli utenti di registrarsi alla piattaforma, anche con servizi esterni come Google o Facebook. Offre agli utenti registrati la possibilità di inserire percorsi scegliendo un punto di inizio, la durata, la difficoltà, una descrizione, eventuali foto del percorso e se quest'ultimo è adatto a persone con difficoltà motorie.
- Il sistema inoltre permette di effettuare una ricerca per località, livello di difficoltà, durata e accessibilità per disabili.
- Per ogni percorso è possibile visionarne i dettagli, dove vengono raccolte tutte le informazioni inserite nella fase di creazione. Inoltre, è possibile, nel caso in cui l'utente lo ritenga necessario, indicare un diverso punteggio di difficoltà e/o un tempo di percorrenza diverso da quello indicato dall'utente che ha creato il percorso, in questo

caso, la difficoltà e il tempo di percorrenza del sentiero saranno ricalcolati come la media delle difficoltà/dei tempi indicati da tutti gli utenti.

- Sarà possibile anche inserire fotografie per i sentieri percorsi, le quali dovranno essere coerenti con il contesto del sentiero che si sta percorrendo.
- Il sistema deve offrire la possibilità agli amministratori di inviare e-mail promozionali agli utenti su accessori per escursionisti a tutti gli iscritti alla piattaforma.
- Gli amministratori potranno modificare o rimuovere i sentieri che riterranno non consoni alla piattaforma.

Documento dei requisiti del software

2. Modello Funzionale

Il modello funzionale rispetta le caratteristiche del modello FURPS della definizione dei requisiti:

- Funzionalità
- Usabilità
- Affidabilità
- Performance
- Supporto

2.1 Requisiti funzionali

Requisito Funzionale 01	
Nome	Registrazione
Descrizione	Il Sistema deve permettere di Registrarsi alla piattaforma se non si è in possesso di un account

Requisito Funzionale 02	
Nome	Autenticazione
Descrizione	Il sistema deve permettere di accedere alla piattaforma previa creazione dell'account.

Requisito Funzionale 03	
Nome	Recupero Password
Descrizione	Il sistema deve permettere di recuperare l'account, attraverso il reset della password.

Requisito Funzionale 04	
Nome	Visualizzare Percorsi nella Homepage
Descrizione	Il sistema deve permettere di visualizzare percorsi visibili nella Homepage

Requisito Funzionale 05	
Nome	Selezione Percorso visibile
Descrizione	Il sistema deve permettere di selezionare percorsi.

Requisito Funzionale 06	
Nome	Visualizzazione della mappa sul percorso selezionato
Descrizione	Il sistema deve permettere di visualizzare su mappa il percorso selezionato.

Requisito Funzionale 07	
Nome	Visualizzazione foto percorso
Descrizione	Il sistema deve permettere di visualizzare tutte le foto caricate del percorso selezionato.

Requisito Funzionale 08	
Nome	Opinione Percorso
Descrizione	Il sistema deve permettere di esprimere un'opinione del percorso.

Requisito Funzionale 09	
<p>Nome</p> <p>Descrizione</p>	<p>Ricerca dei percorsi attraverso dei filtri</p> <p>Il sistema deve permettere di filtrare la ricerca dei percorsi per diversi fattori:</p> <ul style="list-style-type: none"> • Area Geografica • Livello di Difficoltà • Durata • Accessibilità ai disabili

Requisito Funzionale 10	
<p>Nome</p> <p>Descrizione</p>	<p>Inserimento foto nel percorso</p> <p>Il sistema deve permettere di inserire le foto del percorso.</p>

Requisito Funzionale 11	
Nome	Creazione Percorso
Descrizione	<p>Il sistema deve permettere di creare il percorso:</p> <ul style="list-style-type: none"> • Inserimento del nome del percorso • Inserimento della durata del percorso • Inserimento della difficoltà del percorso • Inserimento della descrizione del percorso • Scelta dell'uso del percorso per persone diversamente abili • Inserimento del percorso manuale su mappa. • Inserimento del percorso tramite GPX.

Requisito Funzionale 12	
Nome	Visualizzare percorsi creati
Descrizione	Il sistema deve permettere di visualizzare tutti i percorsi creati

Requisito Funzionale 13	
Nome	Foto sulla mappa
Descrizione	Il sistema deve permettere di visualizzare la fotografia sul marker in mappa

Requisito Funzionale 14	
Nome	Rimozione foto inappropriate
Descrizione	Possibilità di eliminare automaticamente foto inappropriate e non coerenti del percorso.

Requisito Funzionale 15	
Nome	Eliminare un percorso
Descrizione	Il sistema deve permettere agli amministratori la possibilità di eliminare un percorso.

Requisito Funzionale 16	
Nome	E-mail promozionali
Descrizione	Il sistema deve permettere agli amministratori di inviare e-mail promozionali a tutti gli iscritti al programma

Requisito Funzionale 17	
Nome	Modifica del percorso creato
Descrizione	Il sistema deve permettere di modificare il percorso dall'amministratore

2.2 Requisiti non funzionali

Requisito non funzionale 01	
Nome	Scalabilità
Descrizione	Il front end deve essere adattabile a un cambio di database

Requisito non funzionale 02	
Nome	Market share
Descrizione	Sistema va creato per diversi tipi di sistemi operativi, quindi conviene utilizzare un sistema operativo compatibile con quanti più dispositivi ma che abbia quante più funzioni da utilizzare.

Requisito non funzionale 03	
Nome	Sicurezza
Descrizione	La registrazione richiede una password di almeno: 1 carattere minuscolo, 1 carattere maiuscolo, 8 caratteri, 1 numero.

Requisito non funzionale 04	
Nome	Limitazione account
Descrizione	Il sistema deve richiedere la conferma dell'account di un utente mediante mail, al termine della registrazione

Requisito non funzionale 05	
Nome	Scalabilità
Descrizione	In caso di mancata connessione o in caso di malfunzionamento del server va avvisato l'utente attraverso una schermata

Requisito non funzionale 06	
Nome	Usabilità
Descrizione	Nel caricamento delle foto non è possibile caricare video, gif, o audio quindi andare a selezionare solo i formati foto più comuni.

Requisito non funzionale 07	
Nome	Usabilità
Descrizione	L'interfaccia è minimale e completa di tutte le funzionalità che un escursionista ha bisogno.

Requisito non funzionale 08	
Nome	Performance
Descrizione	L'accesso e le richieste al sistema vanno effettuate in pochi secondi dal click, ma non è richiesta una reattività estrema del sistema.

Affidabilità:

- L'applicazione consiste di utenti e amministratori, che sono anch'essi degli utenti ma con delle funzionalità in più.
- Esistono inoltre dei guest che hanno funzionalità in meno rispetto agli utenti poiché non possiedono un account.
- Un utente non può inserire campi vuoti nella creazione del percorso.
- Ogni utente può esprimere un'opinione del percorso più di una volta.
- Il sistema non deve passare in chiaro le informazioni personali degli utenti.
- Vanno eliminate foto e dati di percorso nel tempo di uso dell'applicazione per non sovraccaricare eccessivamente l'uso di RAM della stessa.

2.3 Requisiti di Dominio

-Regolamento del GDPR: GDPR o Regolamento Generale sulla protezione dei Dati, incentrato sulla privacy che sui dati che l'utente ci fornisce.

-Protezione dei dati sul cloud pubblico: SO/IEC 27018:2020 La norma definisce obiettivi di controllo, controlli e linee guida per l'attuazione di misure di protezione dei dati personali comunemente accettate in linea con i principi di privacy raccolti nella ISO/IEC29100 relativamente agli ambienti di cloud pubblico.

2.4 Modellazione dei casi d'uso

Per lo Use Case Diagram abbiamo usato il software gratuito Figma per dare una rappresentazione sommatoria dell'idea progettuale in corso realizzato in UML.

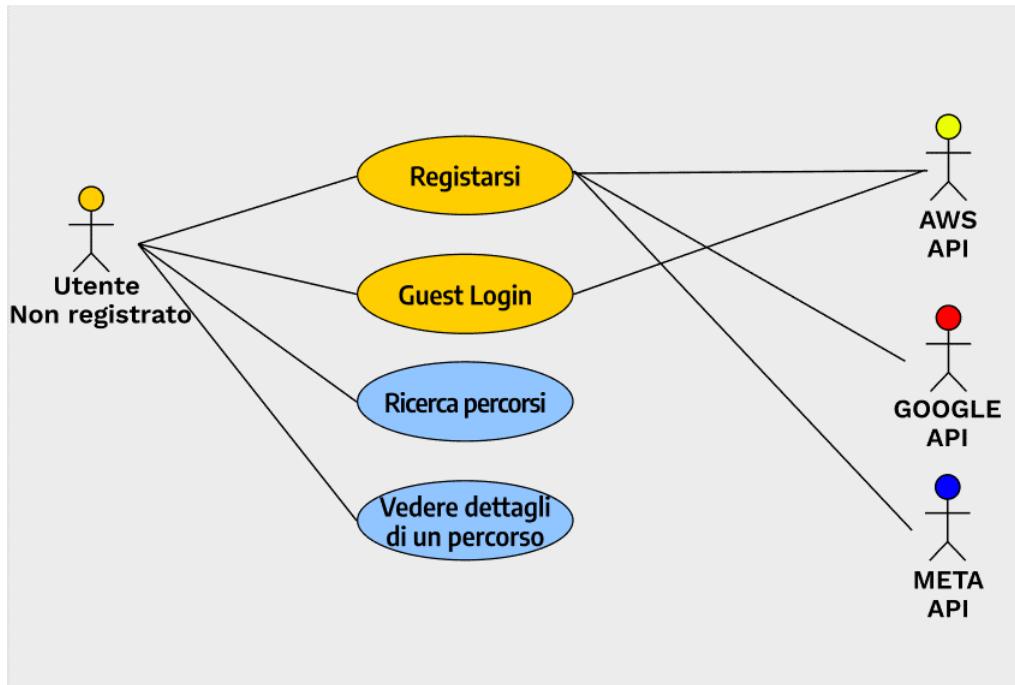
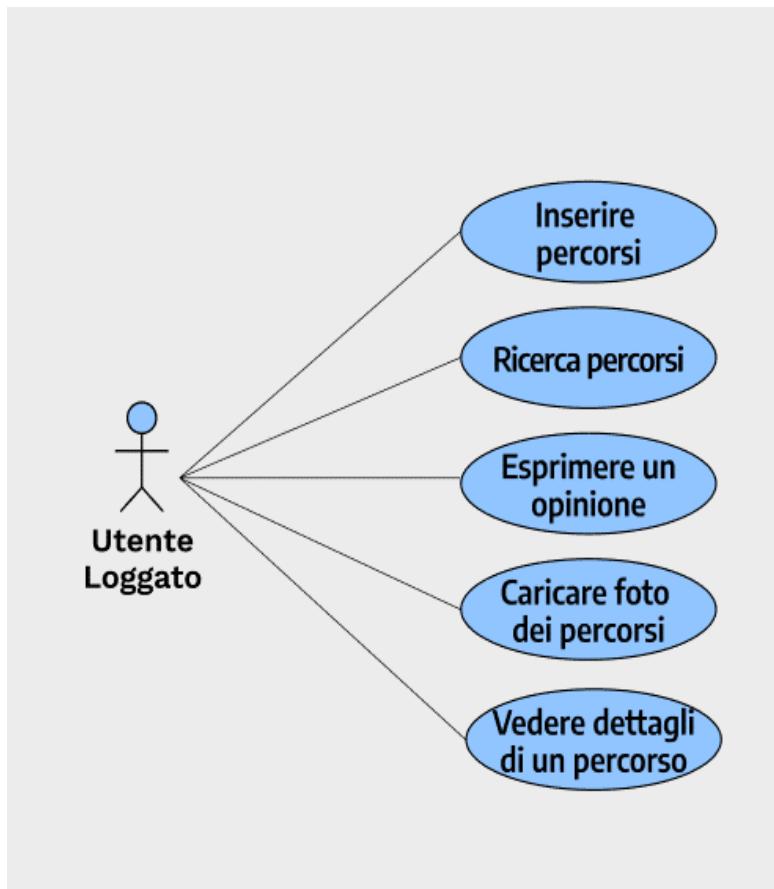


Figura 1 Use Case Diagram della registrazione dell'utente

Lo use case mostra come l'utente può effettuare registrazioni attraverso 3 tipi di piattaforme: Google, Meta e la registrazione tramite email.

Utente già registrato



L'utente registrato e loggato ha a disposizione tutte le possibilità che offre il programma:

- Creazione percorso
- Ricerca percorsi
- Esprimere un'opinione sul percorso
- Caricare delle foto
- Vedere i dettagli

Figura 2 Use Case Diagram di cosa può fare un utente registrato

Log- in Utente

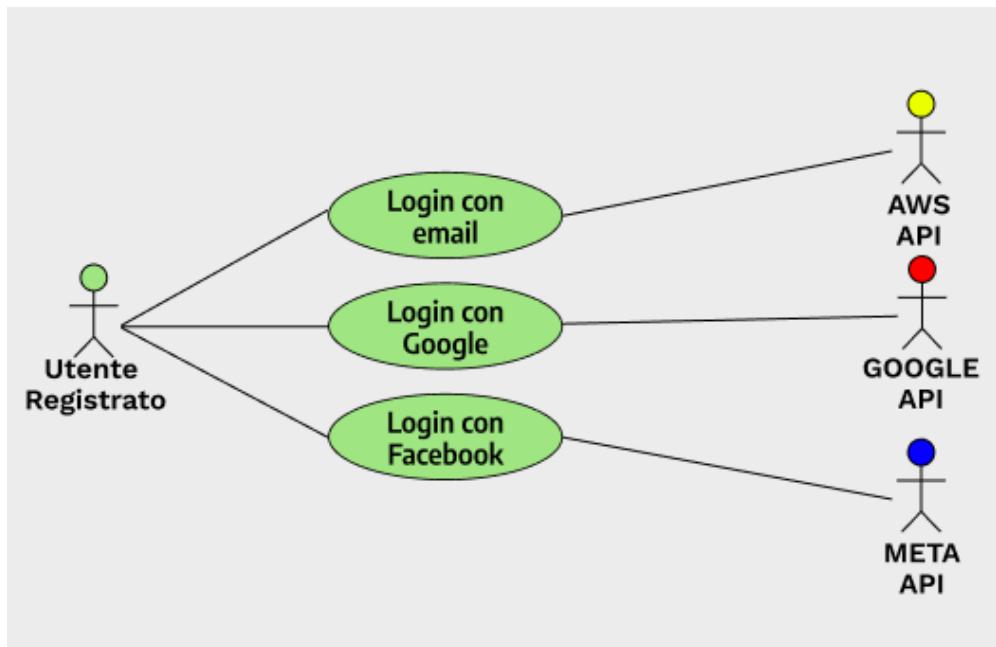


Figura 3 Use Case Diagram di come può entrare un utente registrato

L'utente registrato ma non loggato ha la possibilità di eseguire l'accesso attraverso tre metodi prestabiliti, a seconda di quale abbia scelto lui in partenza.

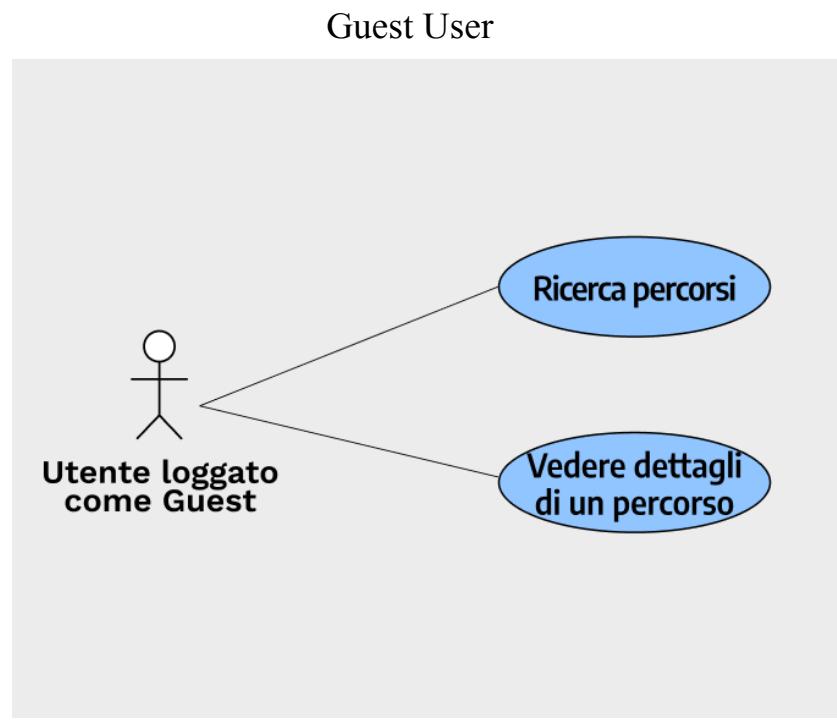


Figura 4 Use Case Diagram di cosa un utente non loggato può fare

Il guest è un utente che decide di non registrarsi ma che vuole solo esplorare l'applicazione.

Rispetto ad un utente normale ha soltanto due funzionalità:

- Ricerca del percorso
- Analizzare un percorso selezionato

Amministrazione

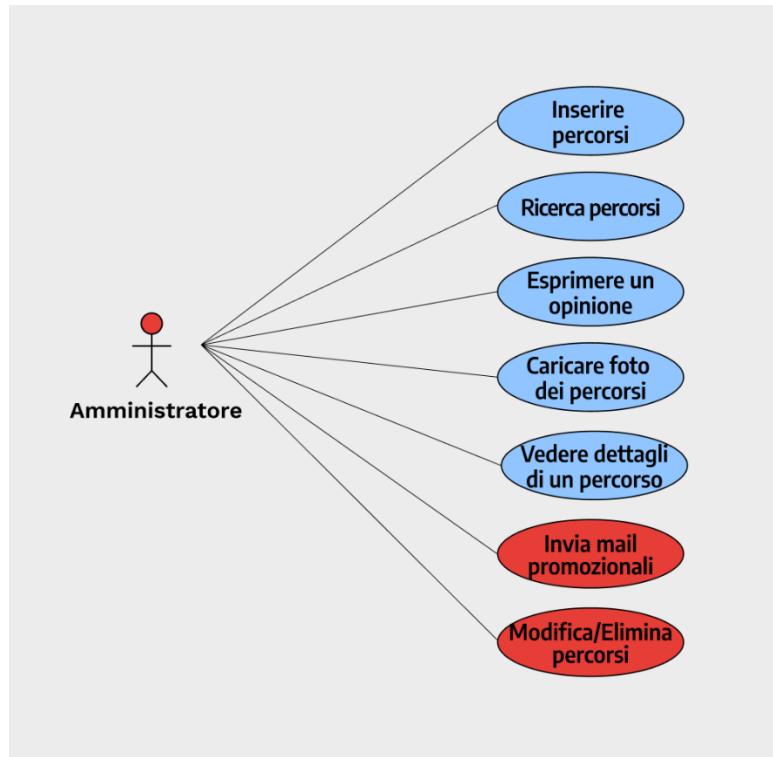


Figura 5 Use Case Diagram di come può agire un utente amministratore.

L'amministratore agisce in modo un po' diverso dagli account utenti normali per la possibilità di inviare mail promozionali agli utenti, cancellare e/o modificare percorsi forzatamente.

2.5 Tabelle di Cockburn per 2 casi d'uso

Ora verranno mostrate le due tavole di Cockburn per due casi d'uso significativi:

2.5.1 Tabella di Cockburn per l'inserimento di un itinerario

Tabella 1

Tabella di Cockburn n°1			
Use Case #1	Inserimento Percorso		
Goal in Context	L'utente vuole creare un nuovo percorso con varie caratteristiche		
Preconditions	L'utente deve essere registrato e loggato nella piattaforma		
Success End Condition	L'utente riesce a inserire l'itinerario nella piattaforma		
Failed End Condition	L'utente non riesce a inserire l'itinerario nella piattaforma		
Primary Actor	Utente loggato, Admin		
Trigger	L'utente clicca sul pulsante per inserire un nuovo percorso		
Description	Step	User Action	System
	1	Clicca sul bottone giallo in	

		basso a destra con la "+"	
2		Mostra mock-up "inserimento"	
3	Selezione la casella "Nome Percorso"		
4		La casella permette di inserire del testo	
5	Selezione la casella "Descrizione Percorso"		
6		Il sistema permette di inserire del testo	
7	Selezione sulla casella "Durata"		
8		Il sistema permette di selezionare la durata tramite un time-picker	
9	L'utente preme su uno dei tre pulsanti "Difficoltà"		
10		Il sistema illumina il pulsante selezionato	
11	L'utente seleziona la casella "Località"		
12		Il sistema mostra una combo box con tutte le località.	
13	L'utente preme sul pulsante "Disabilità"		

	14		Il sistema illumina il pulsante selezionato
	15	L’utente clicca sul tasto per avanzare	
	16		Il sistema apre il mock-up “Scelta”
	17	Selezione sul Tasto GPX	
	18		Il sistema apre l’explorer di sistema
	19	L’utente sceglie il file formato GPX	
	20		Il sistema apre il mock-up “Dialog Inserimento” dove il sistema caricherà in sottofondo il percorso nel database
	21		Il sistema ritorna al mock-up “Homepage”
Extensions	Step	User Action	System
Permessi di archiviazione	18.a.1		Il sistema chiede all’utente i permessi per accedere ai file di sistema
	18.a.2	Preme Consenti	
	Si ricollega al mock up esplora sistema		
L’utente registrato non inserisce tutti i campi disponibili nell’inserimento	15.b.1		Il sistema evidenzia i campi mancanti
	15.b.2	L’utente inserisce i campi mancanti	
	Si ricollega al mock up inserimento		
Subvariations	Step	User Action	System

	17.1	L’utente clicca sul bottone "Mappa" per l’inserimento manuale del percorso	
	17.2		Il sistema mostra il mock-up “Mappa”
	17.3	L’utente inserisce i punti del percorso cliccando sulla mappa	
	17.4	L’utente clicca su “Conferma”	
	17.5		Il sistema mostra il mock-up “Caricamento Percorso” mentre carica il percorso all’interno del database
	17.6	L’utente clicca sul pulsante “OK”	
	17.7		Il sistema mostra il mock-up “Homepage”
Notes			

2.5.2 Tabella di Cockburn per la ricerca di un itinerario

Tabella 2

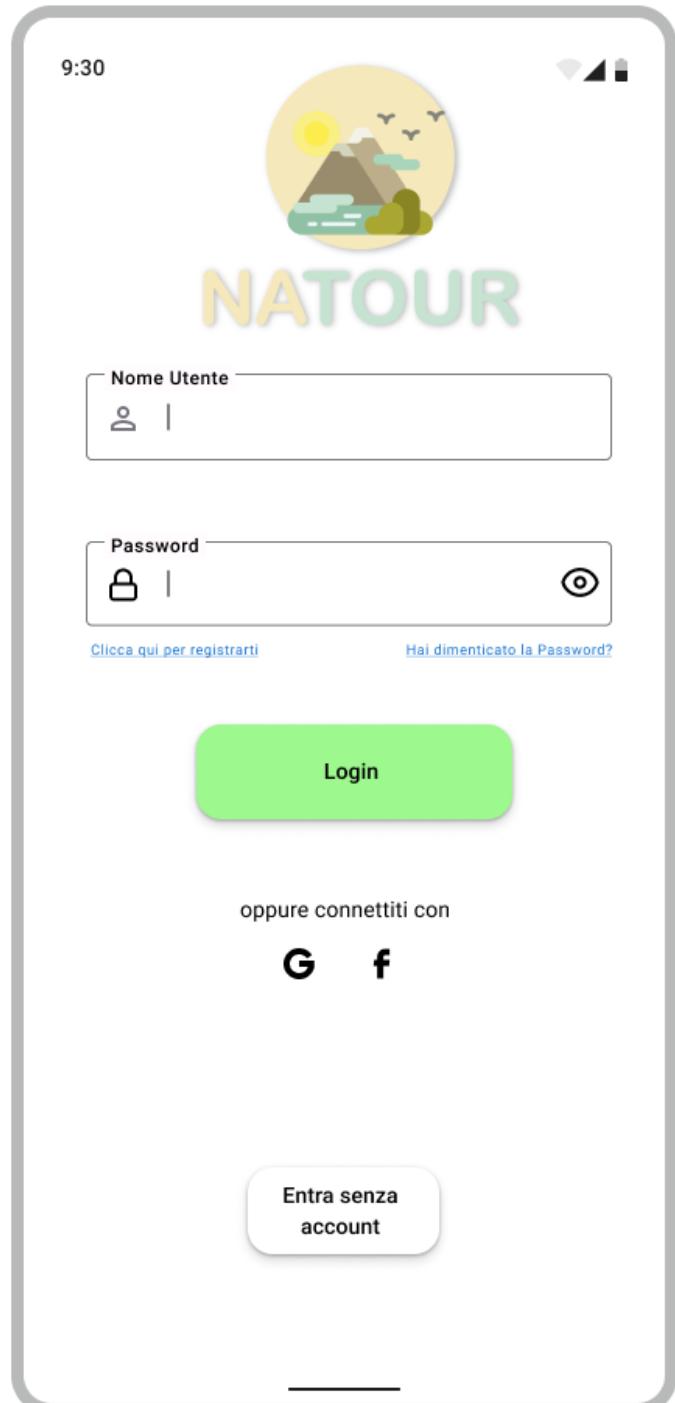
Tabella di Cockburn n°2			
Use Case #2	Ricerca Percorso		
Goal in Context	L'utente vuole cercare un percorso specifico attraverso un sistema di filtraggio		
Preconditions	L'utente non deve essere obbligatoriamente registrato e loggato nella piattaforma		
Success End Condition	L'utente trova il percorso desiderato o più vicino a quello desiderato		
Failed End Condition	L'utente non riesce a trovare il percorso		
Primary Actor	Utente Loggato, Utente non loggato, Admin		
Trigger	L'utente clicca sulla barra di ricerca del programma		
Description	Step	User Action	System
	1	Clicca sulla barra in alto di ricerca	
	2		Mostra mock up Ricerca
	3	Seleziona la casella "Località"	
	4		La casella permette di inserire del testo che suggerisce,

			attraverso una combo box, le destinazioni più comuni con quel nome
	5	L'utente preme su uno dei 3 pulsanti "Difficoltà"	
	6		Il sistema illumina il pulsante selezionato
	7	Selezione la casella "Durata"	
	8		Il sistema permette di scegliere la durata tramite un time picker
	9	L'utente seleziona la il pulsante del filtro per i disabili	
	10		Il sistema illumina la casella della disabilità
	11	L'utente preme sul pulsante di ricerca	
	12		Il sistema offre la destinazione più coerente ai filtri di ricerca
Subvariations	Step	User Action	System
	3.1	L'utente non inserisce "Località"	Il sistema esclude "Località" dalla ricerca
	5.1	L'utente non inserisce la "Durata"	Il sistema esclude "Durata" dalla ricerca
	7.1	L'utente non inserisce la "Difficoltà"	Il sistema esclude "Difficoltà" dalla ricerca
	9.1	L'utente non inserisce la	Il sistema esclude "Disabile" dalla ricerca

	percorrenza per disabili	
Notes		

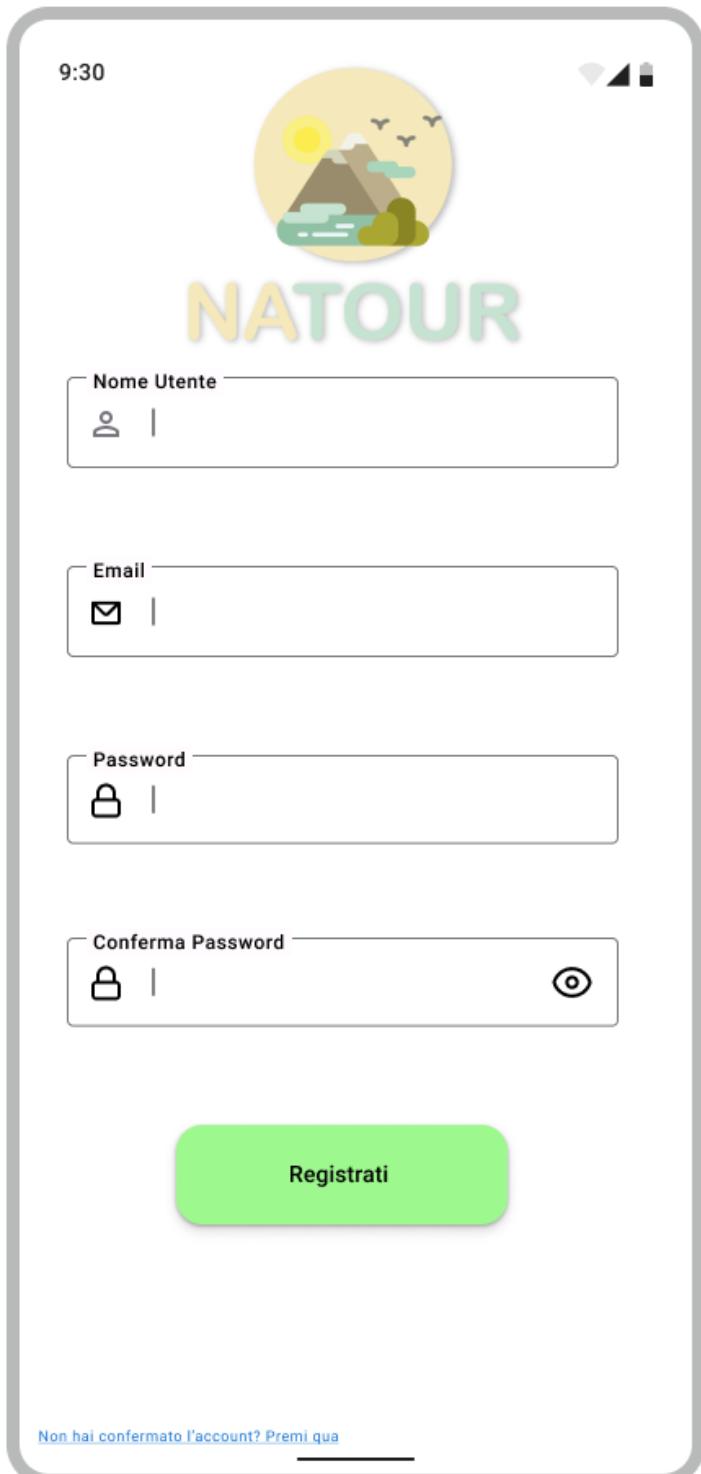
2.6 Prototipazione via Mock-Up

Figura 1



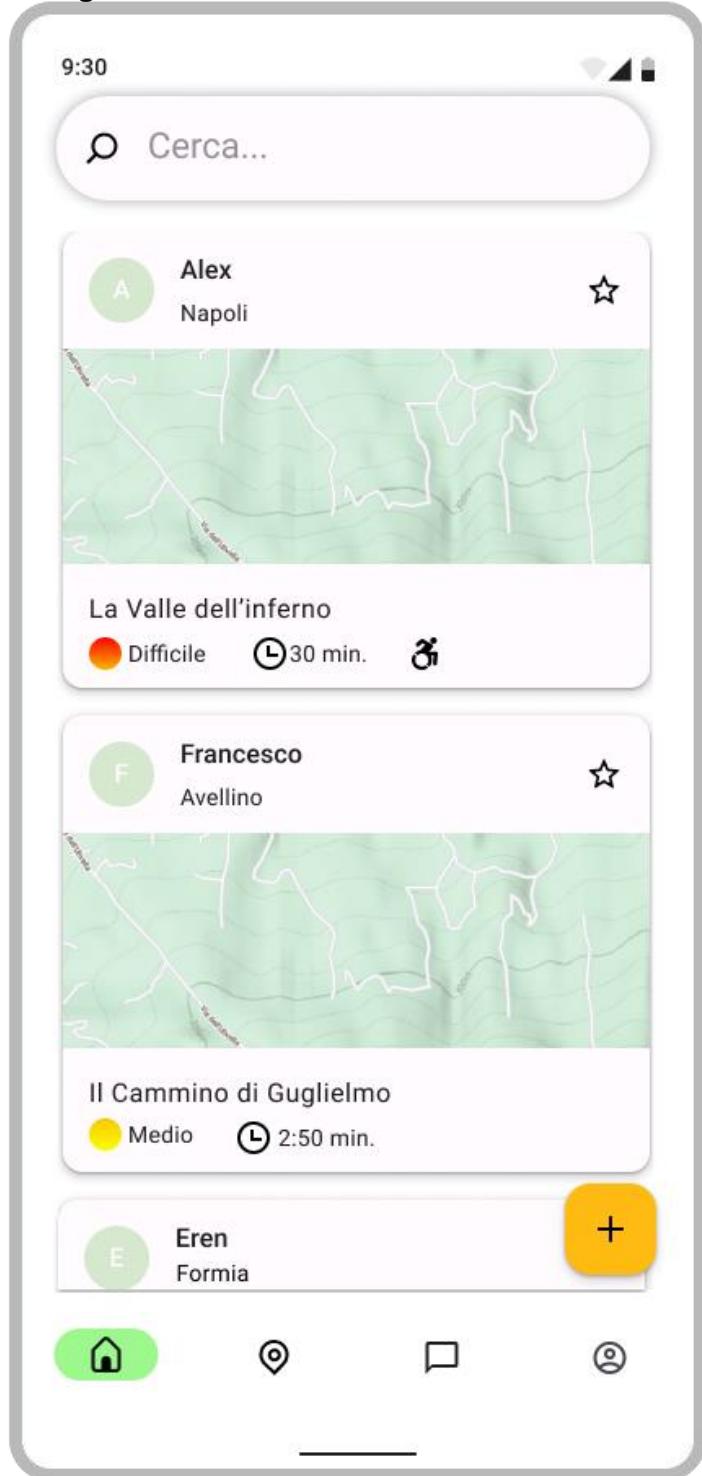
1: Log in page dell'applicazione

Figura 2



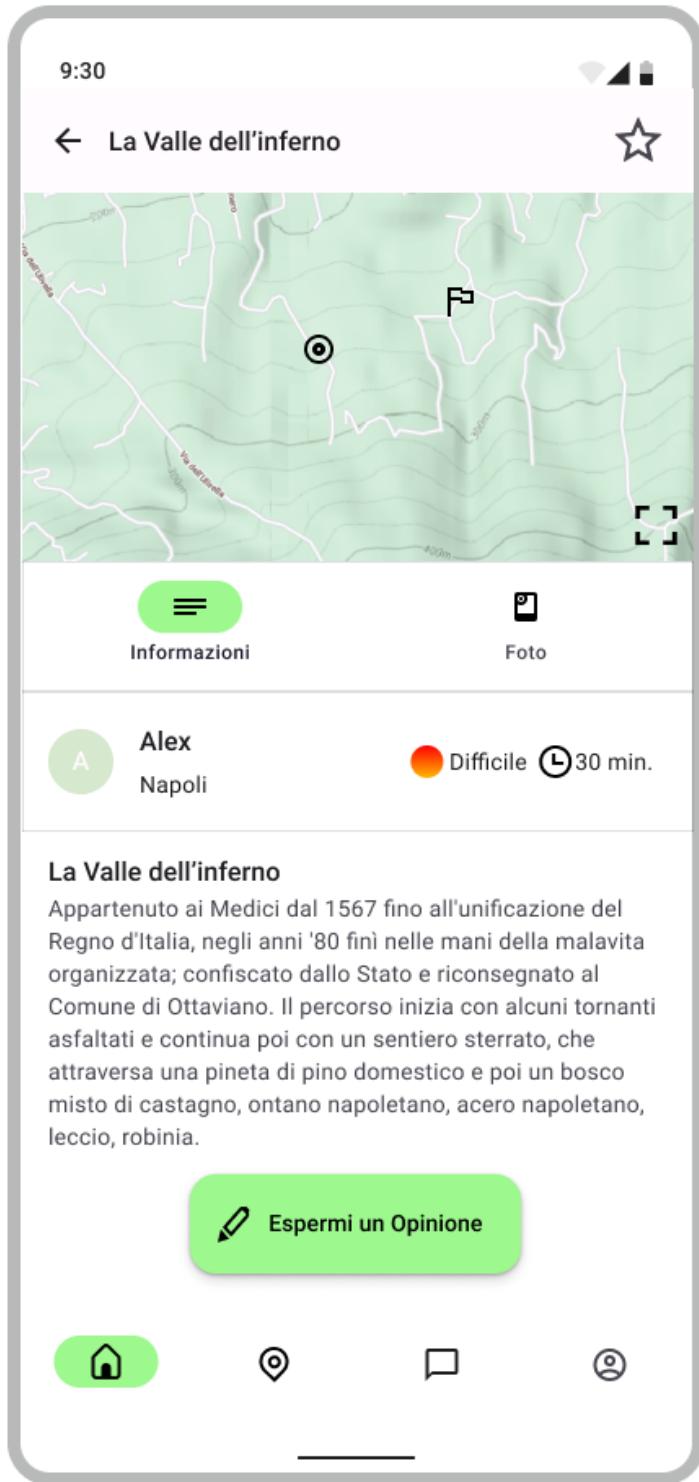
2: Registrazione alla piattaforma

Figura 3



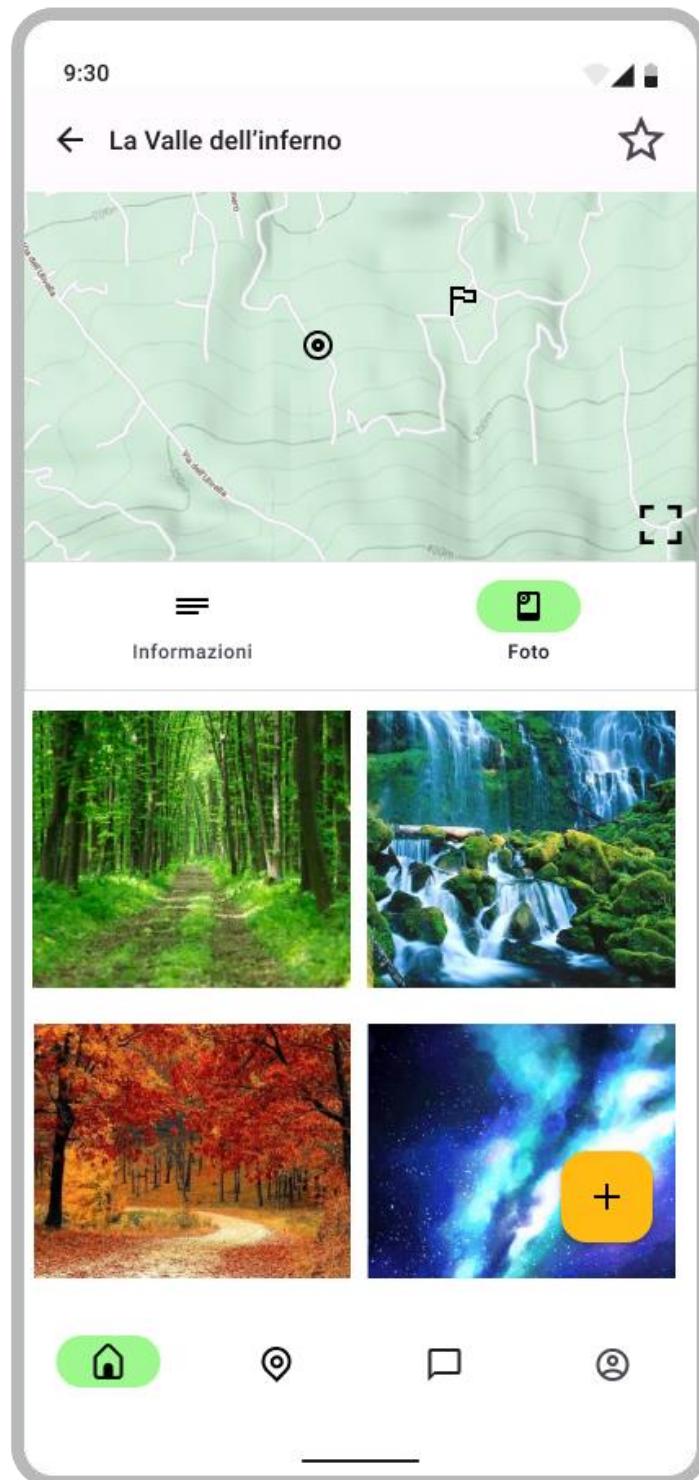
**3: Mock up delle foto del percorso
selezionato**

Figura 4



4: Dettaglio Sentiero

Figura 5



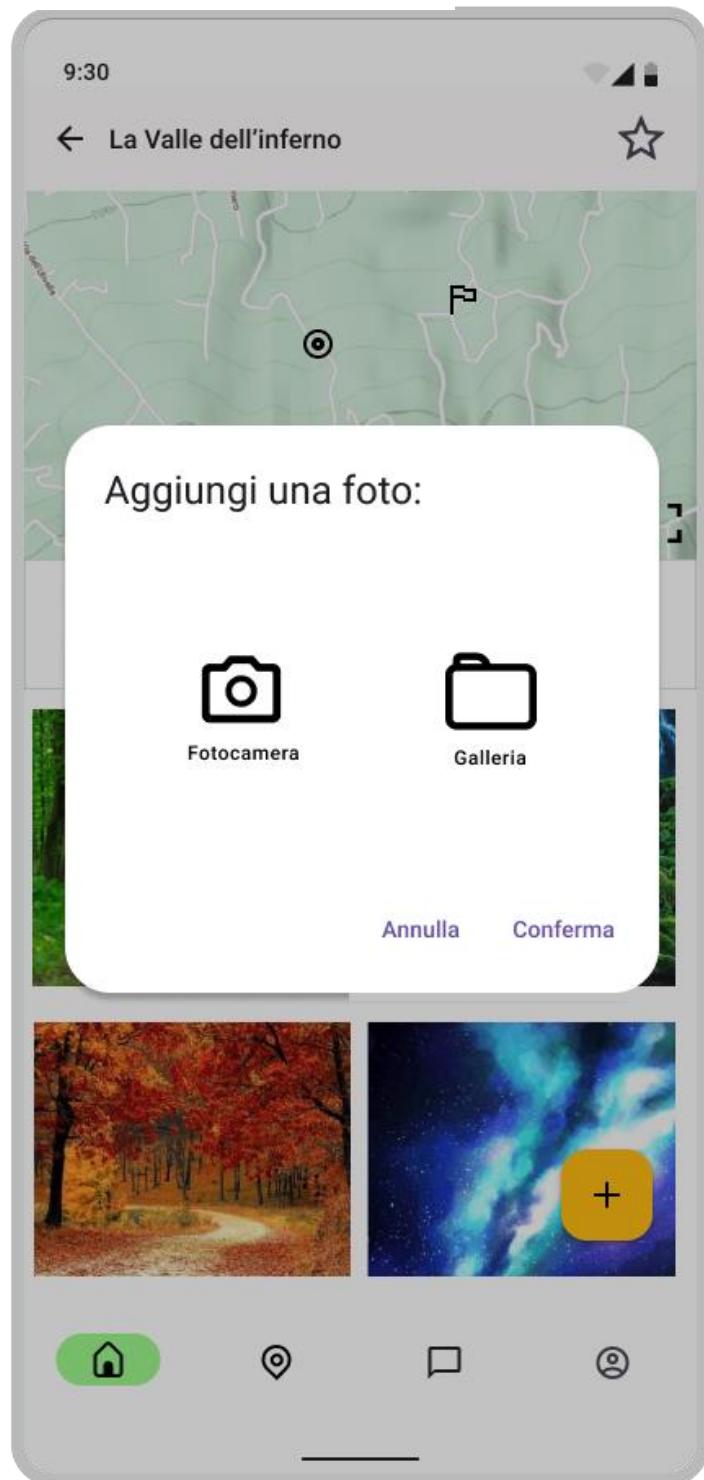
5: Foto di un Percorso

Figura 6



6: Esprimere un'opinione all'interno del percorso

Figura 7



7: Inserimento foto all'interno del percorso

Figura 8



8: Ricerca del percorso

Figura 9



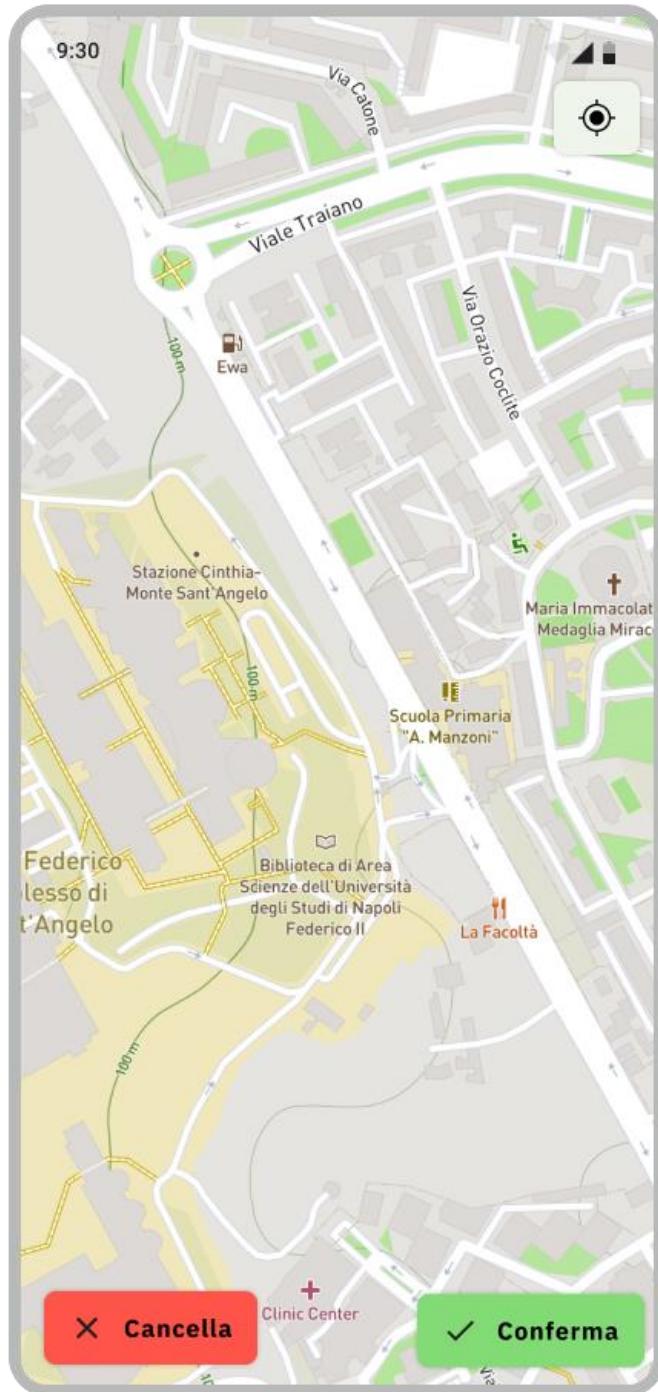
9: Creazione del percorso

Figura 10



10: Scelta del tracciato

Figura 11



11: Visualizzazione della mappa

2.7 Individuazione del target utenti

Le indagini di mercato sono tra i principali strumenti utilizzati da aziende, enti e associazioni per studiare il proprio pubblico di riferimento.

Inoltre, l'indagine di mercato viene utilizzata anche per individuare nuovi prodotti e servizi da offrire.

Target è un termine adottato dall'inglese e letteralmente significa bersaglio. Nel campo del marketing aziendale è l'obiettivo da raggiungere con le azioni di comunicazione.

Identificare i clienti target vuol dire individuare i potenziali destinatari dei propri prodotti o servizi.

Per identificare i clienti target si ha bisogno di questi elementi base fondamentali per una ricerca efficiente:

- Dati demografici
- Localizzazione
- Interessi e opinioni

Infine, c'è bisogno di effettuare cinque domande per l'identificazione del target:

1. Qual è l'immagine del tuo cliente tipo?
2. A quali problemi/bisogni del cliente vuoi rispondere con i tuoi prodotti/servizi?
3. Quali clienti otterranno benefici dai tuoi prodotti?
4. Ci sono nicchie di mercato a cui puoi far riferimento?
5. Chi sono i tuoi competitor?

Facendo riferimento a vari dati da fonti quali ISTAT, Trekking Italia e All Trails (sito globale sul trekking), abbiamo ricavato

informazioni e dati sull'andamento dell'escursionismo in Italia e nel mondo.

Seleziona periodo		
2014		77 353
2015		67 168
2016		74 133
2017		69 972
2018		83 803
2019		80 021
2020		41 194
2021		36 154

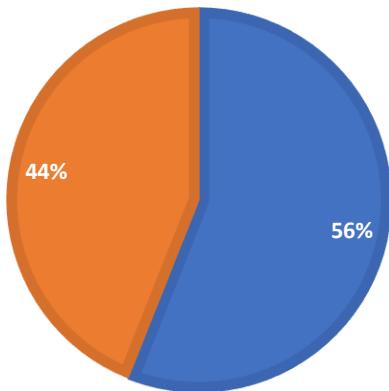
Secondo l'ISTAT la tendenza di escursionismo è continuato a crescere negli anni a venire, ma notiamo un'improvvisa diminuzione quasi del 50% nel 2020 causa pandemia, e un conseguente trend negativo l'anno successivo. Nonostante ciò, secondo l'ISNART la tendenza di attività degli italiani si è spostata decisamente verso l'escursionismo¹. A confermarlo è la rivista Outdoor Magazine, dove nel suo speciale trekking evidenzia come il popolo italiano proprio nel 2020 ha iniziato a fare escursioni giornaliere sempre più frequenti.

Analizzando un sondaggio italiano creato dalla stessa rivista, evinciamo le caratteristiche degli escursionisti italiani, su una base di risposte di circa 3300 persone.

¹ Fonte : [Italiani popolo di camminatori: è il trekking l'attività più praticata nelle vacanze 2020 - Il Sole 24 ORE](#)

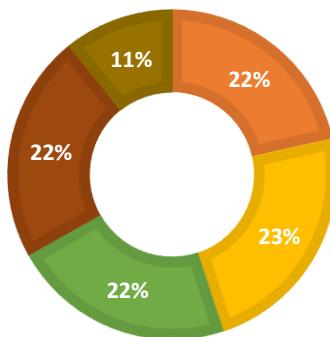
GENERE ESCURSIONISTI

■ Uomo ■ Donna



FASCE D'ETÀ

■ Under 30 ■ 31-40 ■ 41-50 ■ 51-60 ■ Over 61



Da questi grafici basati su risposte date da quel campione di persone, possiamo notare come non c'è una predominanza di genere o di età per l'escursionismo. I dati ci incoraggiano sulla quantità e varietà di persone che possono essere interessate all'applicazione.

2.8 User Personas

Capire bene qual è il target audience è fondamentale per creare un prodotto eccezionale. Le persone qui intervistate sono fondamentali per aiutarci a capire chi è più o meno interessato al nostro prodotto. Infatti, a tal proposito abbiamo intervistato delle persone

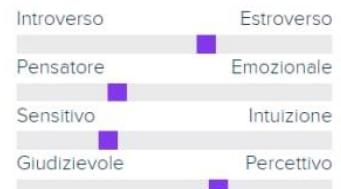
Amalia De Stefano



Obiettivi

- Cambiare vita spesso
- Cucinare
- Seguire la moda
- Amante di Film e Serie TV

Personalità



"Libera Professionista"

Età:30

Lavoro: Consulente Marketing

Luogo: Vienna

Sesso: Femmina

Biografia

Sono una creatrice di contenuti online su Instagram e Tik-Tok, ho da poco iniziato ad apprezzare le gite fuori porta. Riesco finalmente a staccare dalla vita quotidiana e riprendere quelle emozioni che sotterro per dedicarmi ai miei lavori.

"Vivere la vita al 100%"

Maria Carrese



"Escursionista Esperto"

Età: 25

Lavoro: Pilota D'aerei

Luogo: Milano

Sesso: Femmina

Obiettivi

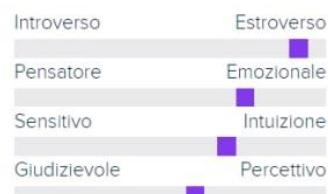
- Essere libera
- Viaggiare Molto
- Curare la natura e all'ambiente

Biografia

Ho preso il brevetto di pilota d'aerei perché amo osservare il mondo che mi circonda da un'altra e alta prospettiva. Sono interessata a nuove avventure che il sentiero della vita ha da propormi.

"Freedom"

Personalità



Luigi Tadoni



"Studente liceale"

Età: 20

Lavoro: Studente

Luogo: Kyoto

Sesso: Maschio

Obiettivi

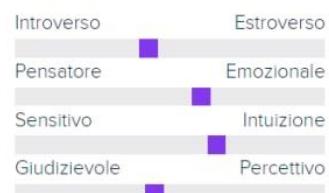
- Escursioni all'aperto
- Baseball
- Barbecue

Biografia

Sono alla ricerca di nuovi posti naturalistici da esplorare e da vivere nei quali coinvolgere i miei amici e familiari, per riuscire a scoprire meglio me stesso e le mie relazioni.

"Lascio i problemi di domani al me di domani"

Personality



Jonathan Borgheresi



"Imprenditore"

Età: 40
Lavoro: Imprenditore
Luogo: Napoli
Sesso: Maschio

Obbiettivi

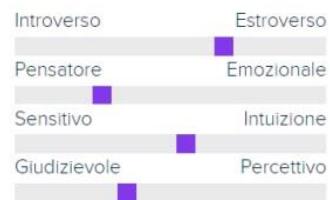
- Modellismo
- Macchine vintage
- Fotografia
- Chiacchierare

Biografia

Fondatore della GnG Corporation, una persona sempre in cerca di nuove esperienze, in questo periodo si sta focalizzando sulle escursioni naturalistiche. Con non molto tempo a disposizione prova comunque a dedicare il proprio cuore alle proprie passioni.

"Keep moving forward"

Personalità



Queste user Personas sono un modello referenziale, basate su previsioni della realtà.

2.9 Valutazione Usabilità a Priori

Per effettuare una valutazione completa dell’usabilità del nostro software, prendiamo ispirazione dal modello Norman che ci spiega quanto sono articolati, non semplici e intuitivi i processi che coinvolgono la nostra interazione col mondo. Per definire in modo più preciso tale definizione useremo il termine più specifico di “Usabilità”.

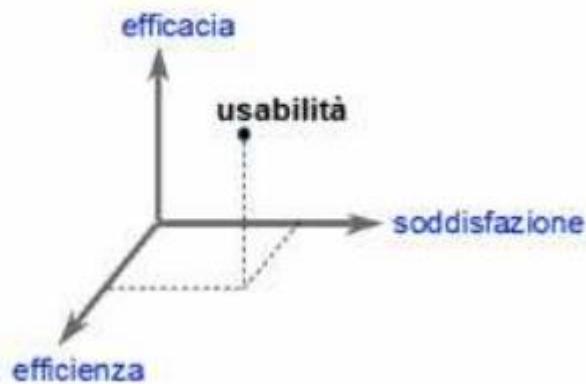
Per usabilità si intende «il grado in cui un prodotto può essere usato da particolari utenti per raggiungere certi obiettivi con efficacia, efficienza, soddisfazione in uno specifico contesto d’uso».

L’usabilità focalizza la dimensione funzionale dell’interazione tra un sistema (ad es. un sito web) e l’utente, in relazione a precisi obiettivi e contesti d’uso.

Per Norman serve “quantificare” l’usabilità, dandone una misura oggettiva. Una definizione che ci tornerà utile è quella proposta nel modello standard ISO 9241:

“L’usabilità di un prodotto è il grado con cui esso può essere usato da specificati utenti per raggiungere specificati obiettivi con efficacia, efficienza e soddisfazione in uno specificato contesto d’uso”

Questa definizione scomponete l’usabilità su 3 assi, cioè 3 variabili che permettono di misurare il valore da noi desiderato.



L'**efficacia** è detta anche accuratezza con cui gli utenti raggiungono un determinato obiettivo

L'**efficienza** è definita come la quantità di risorse spese in relazione all'obiettivo che si vuole raggiungere, come ad esempio il tempo impiegato per un determinato obiettivo, il numero di tasti da premere per una funzione ecc.

La **soddisfazione** dipende da vari fattori e sta a indicare l'indice di gradimento di un'azione; quindi, va a “ricompensare” l'utente attraverso la sua “facilità d'uso” e “velocità nel raggiungere gli obiettivi”.

A tal proposito abbiamo deciso di realizzare un'interfaccia pulita e veloce per accomodare l'utente nelle poche e utili funzioni di cui esso potrà usufruire. Abbiamo dunque deciso di optare per un design minimale durante la fase di **Mock-up** e di conseguenza decidere di tenere lo stesso design anche in beta testing e nel prodotto finale.

Per realizzare il Mock-up è stato utilizzato **Figma** e grazie a quest'ultimo siamo riusciti a simulare delle finestre di interazione responsivi e dinamici.

Ci è stato molto utile la libreria scritta da Google: “[Material 3 Design Kit](#)”, è una serie di componenti grafici uguali a quelli che si usano durante la creazione effettiva delle applicazioni Android, questo ci permetterà di creare la grafica dell’applicazione molto similare ai mockup realizzati su Figma.

I Mock-up sono stati realizzati tenendo presente le 8 regole d’oro di [Shneiderman](#).

Shneiderman’s 8 Golden Rules

1. Consistency (L)
2. Shortcuts (EF)
3. Feedback (V)
4. Dialog closure (V)
5. Simple error handling (ER)
6. Reversible actions (UC)
7. Put user in control (UC)
8. Reduce short-term memory load (ER)

In generale per l’usabilità abbiamo deciso di usare tasti e schermate molto larghe, unidirezionali e con informazioni minimali e giuste così da poter far andare velocemente l’utente avanti nell’applicazione. Ad esempio, è stato optato di far visualizzare subito dei percorsi all’interno di una “home” così da far ambientare subito l’utente nella selezione. Inoltre, la barra sottostante di navigazione permette di passare da un’attività all’altra velocemente, anche grazie all’uso di immagini esplicative delle loro funzioni. Per qualsiasi tipo di interazione che l’utente deve avere con l’applicazione sono stati forniti pop-up che permettono di essere sempre aggiornato con qualsiasi operazione avvenuta su schermo e non.

Le tecniche utilizzate per la valutazione dell’usabilità a priori sono varie. Inizialmente, dopo la realizzazione dei prototipi, abbiamo deciso di effettuare un test di usabilità, attraverso la simulazione di usabilità di Figma, simile alla **tecnica del mago di Oz**, ma virtualmente.

Generalmente si usano due tecniche per valutare l’usabilità dei sistemi: Le valutazioni euristiche e i test di usabilità. Le valutazioni euristiche sono eseguite da esperti di usabilità. I test vengono invece effettuati su un campione di utenti di diversi tipi di conoscenza del settore.

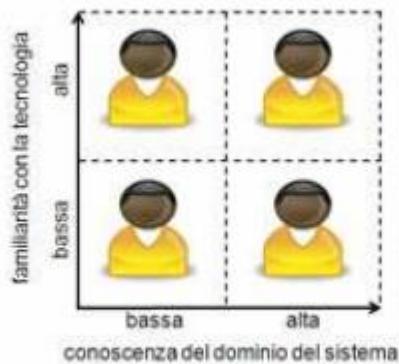


Figura 12. Le due dimensioni del profilo degli utenti

Le nostre valutazioni sono state effettuate tenendo conto delle 10 regole di Nielsen. La pianificazione del test di usabilità inizia con una attenta scelta dei valutatori che devono utilizzare i prototipi e poi esaminarli.

Useremo la conoscenza del dominio (competenza in materia) e la conoscenza tecnologica per suddividere i valutatori in categoria.

I Valutatori:

- Bianca Silvestri: con bassa conoscenza tecnologica e bassa conoscenza di dominio
- Giovanni Orsini: bassa conoscenza tecnologica e alta conoscenza del dominio
- Nedda Toninelli: alta conoscenza della tecnologia e bassa conoscenza del dominio
- Susanna Vespa-Tarchetti: alta conoscenza della tecnologia e bassa conoscenza del dominio

Di seguito saranno elencati in lista i compiti svolti dai tester:

1. Registrazioni
2. Login
3. Creazione Percorso
4. Ricerca Percorso
5. Opinione Percorso

Livello di dettaglio	Compiti Svolti					
		Compito 1	Compito 2	Compito 3	Compito 4	Compito 5
Bianca Silvestri	Passed	Semi-Passed	Failed	Semi-Passed	Passed	
Giovanni Orsini	Failed	Passed	Semi-Passed	Passed	Failed	
Nedda Toninelli	Passed	Passed	Passed	Semi-Passed	Failed	
Susanna Vespa-Tarchetti	Passed	Semi-Passed	Semi-Passed	Semi-Passed	Passed	

I risultati dimostrano come i test superati o semi-superati sono stati 16/20, un buon risultato iniziale, dato che stiamo testando il nostro design su Figma.

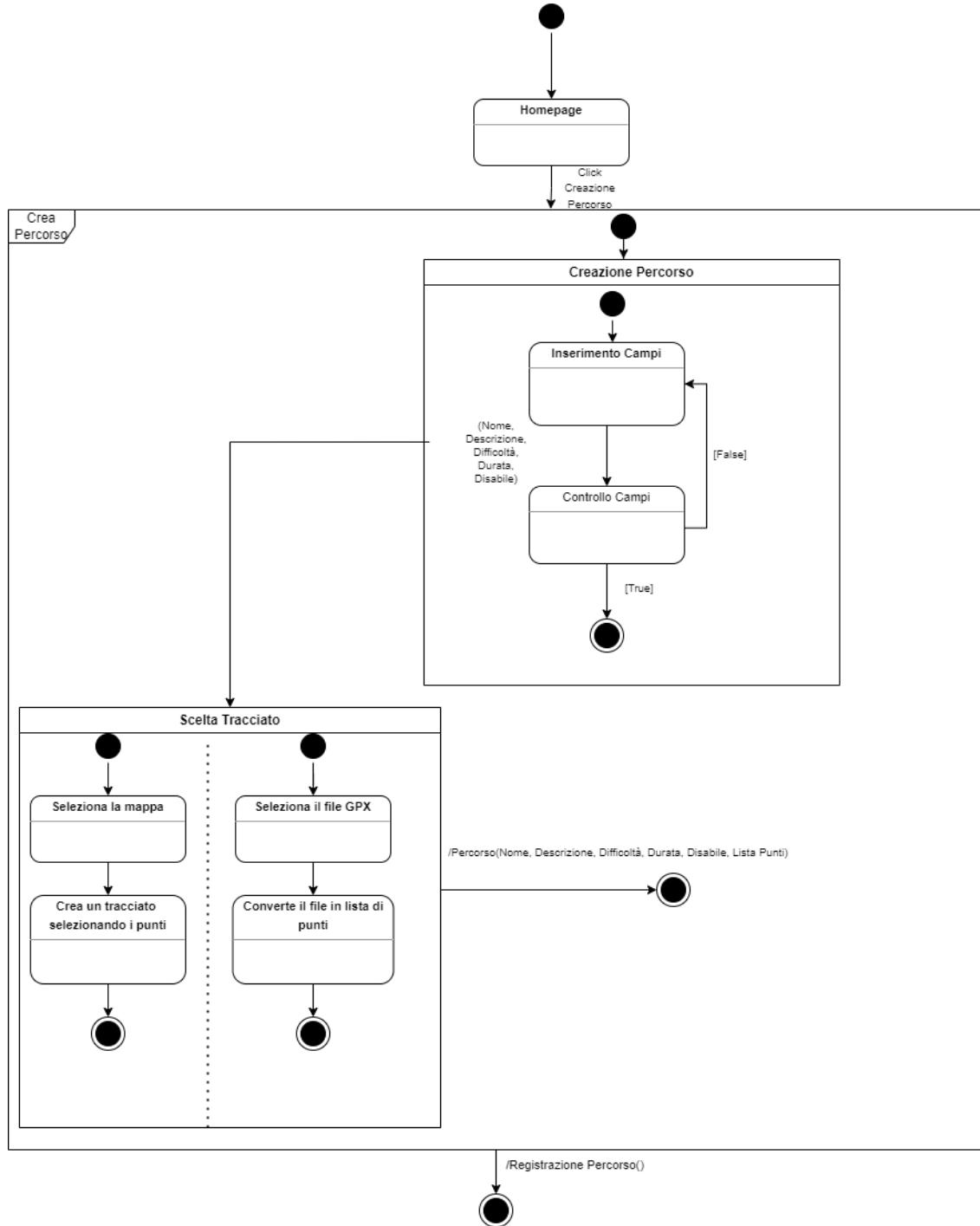
Abbiamo ottenuto delle informazioni che ci hanno permesso di cambiare il layout del programma e semplificare la schermata per incorrere in meno problemi ed errori, come ad esempio:

-L'inserimento di un FAB fisso, al lato della schermata home per l'inserimento di un percorso.

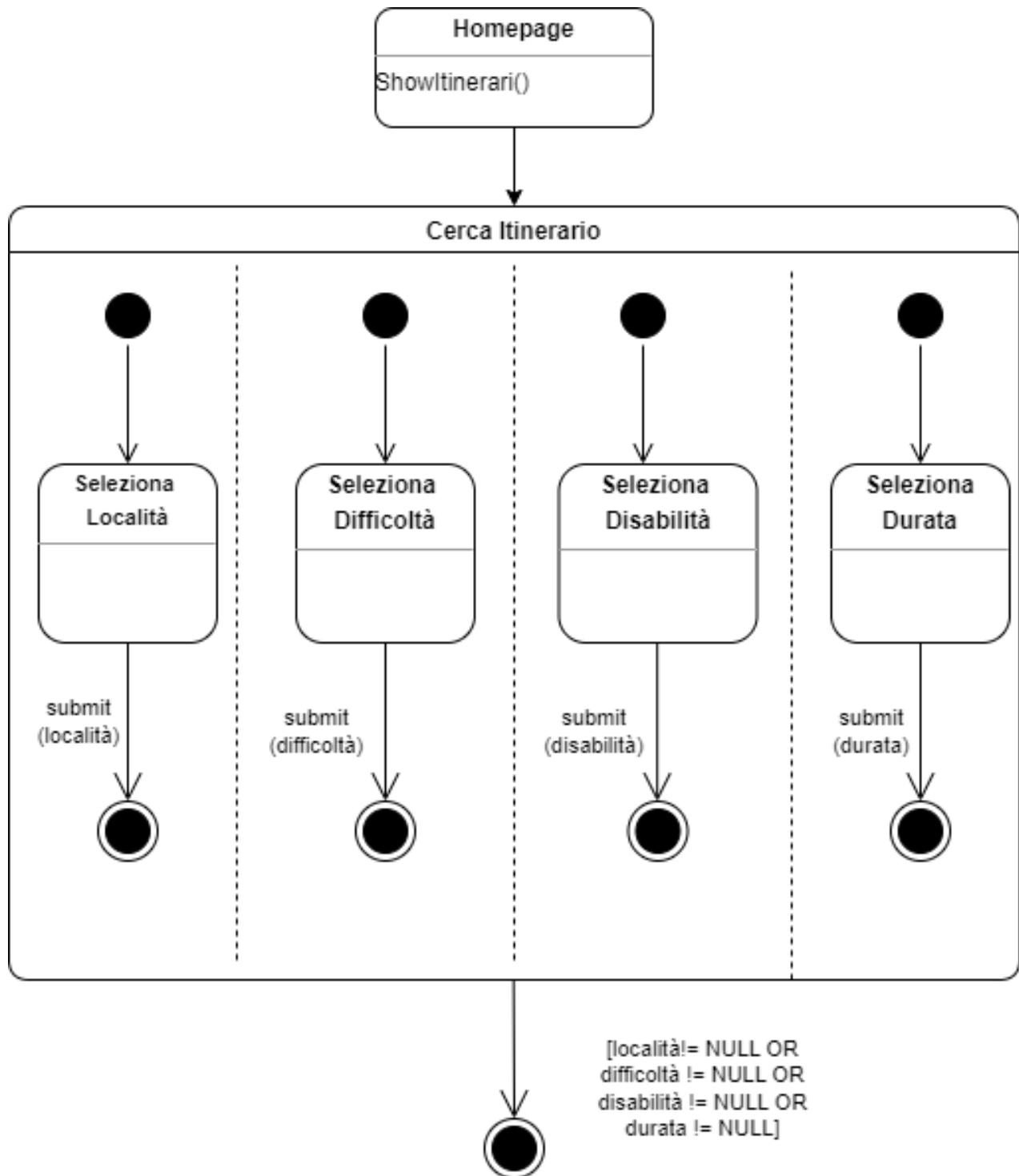
-La search bar è stata ingrandita per essere visibile e sempre presente nella homepage

2.10 Prototipazione funzionale via Statechart dell'interfaccia grafica

2.10.1 Statechart Inserimento percorso



2.10.2 Statechart Ricerca Percorso



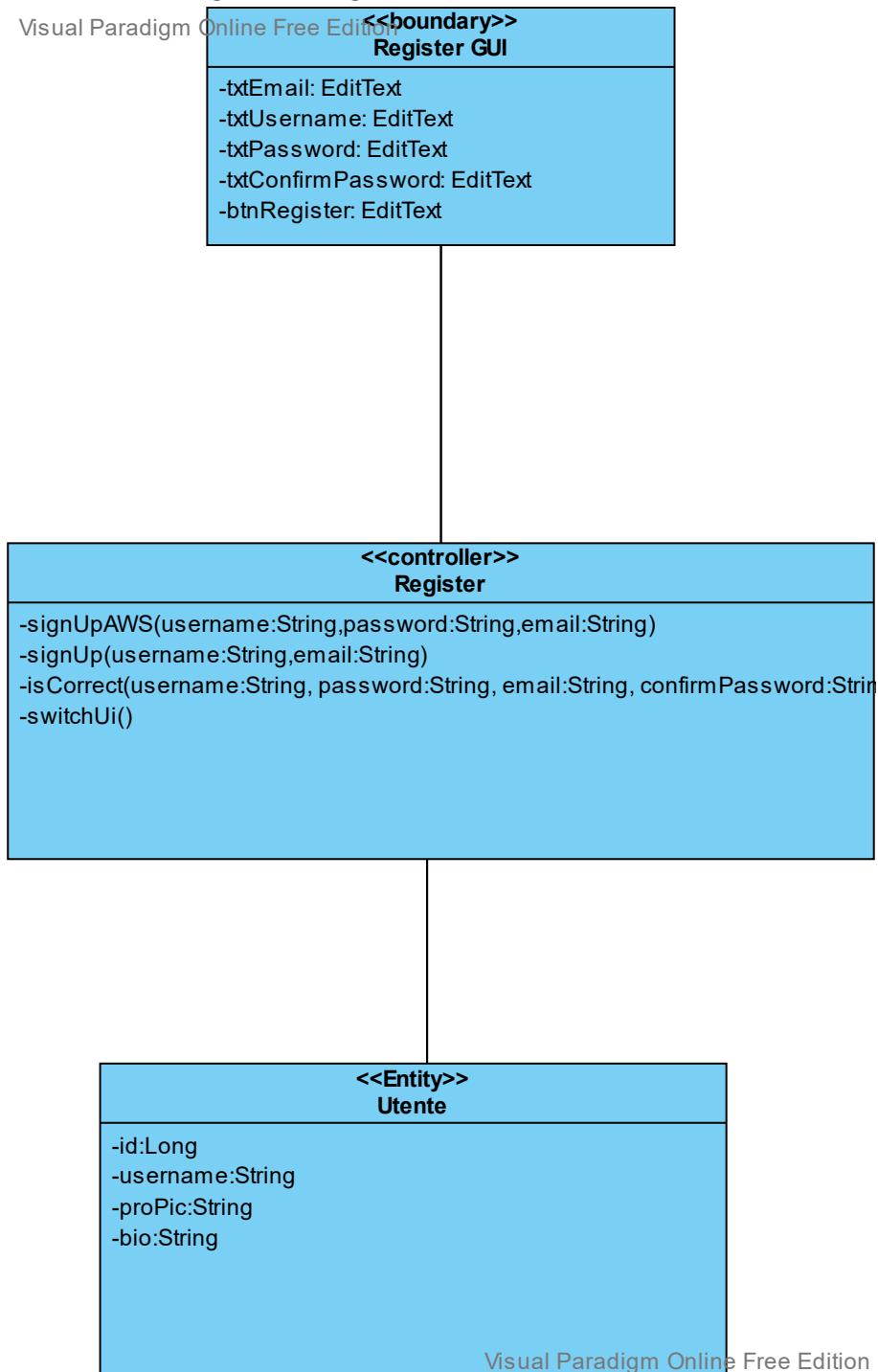
3. Modelli di Dominio

3.1 Classi, oggetti e relazioni d'analisi

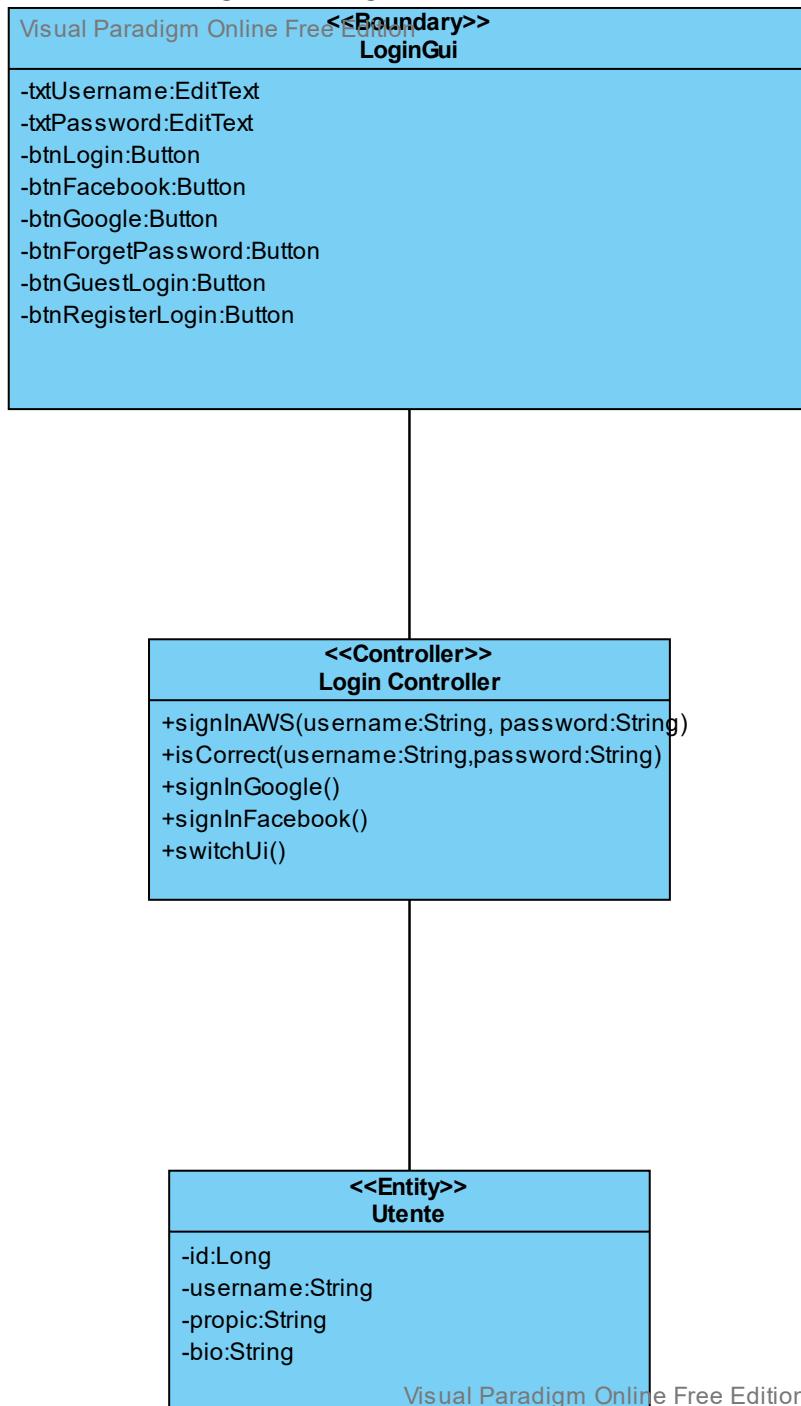
Per rappresentare classi e entità abbiamo deciso di usare l'euristica **ECB** (Entity-Boundary-Control) che ci permette di specificare i nostri requisiti in tre oggetti.

- Entity – sono concetti del dominio che sono persistenti e simili alle entità.
- Boundary – o confini, rappresentato la linea di confine tra utente e sistema (Le GUI)
- Control – I controller, le classi che controlleranno tutta la logica del sistema

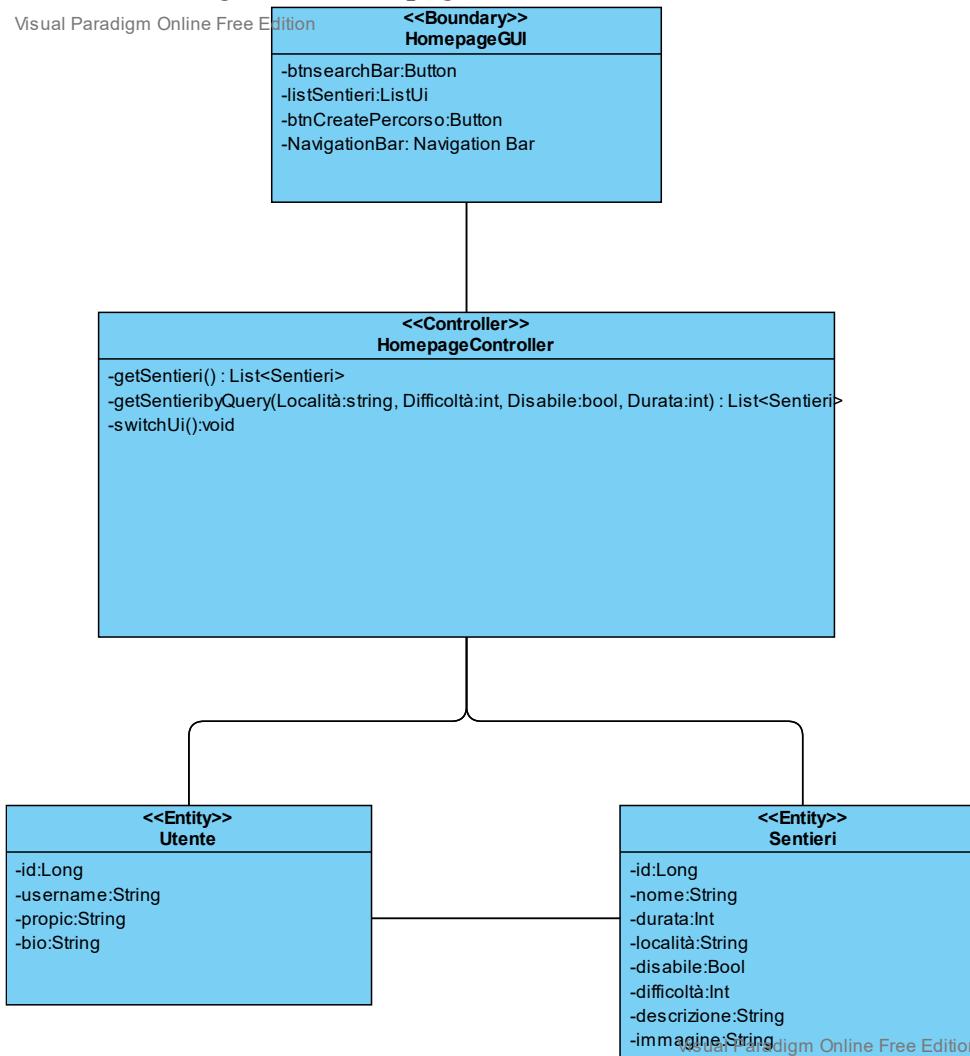
3.1.2 Class Diagram – Registrazione



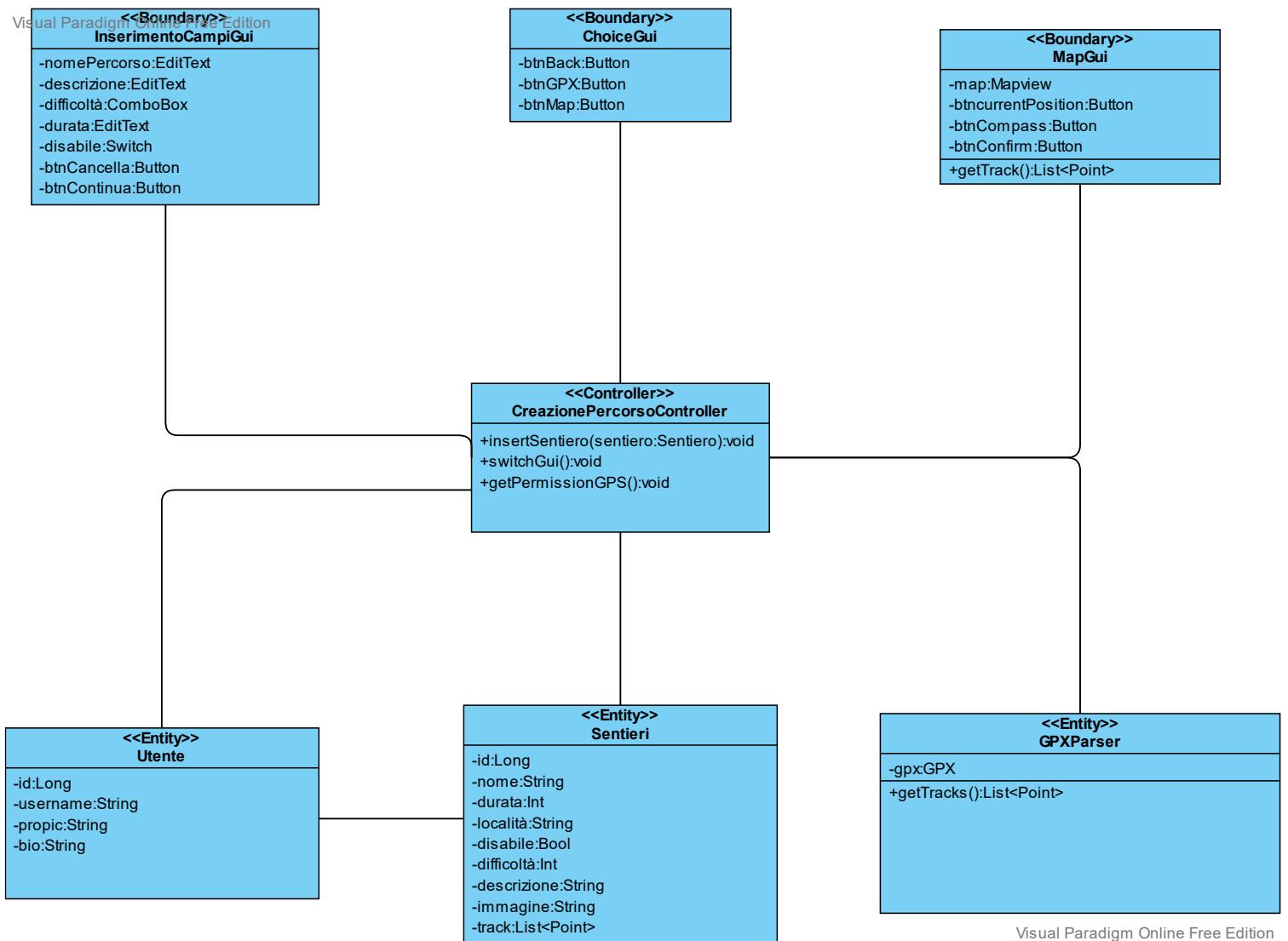
3.1.3 Class Diagram – Login



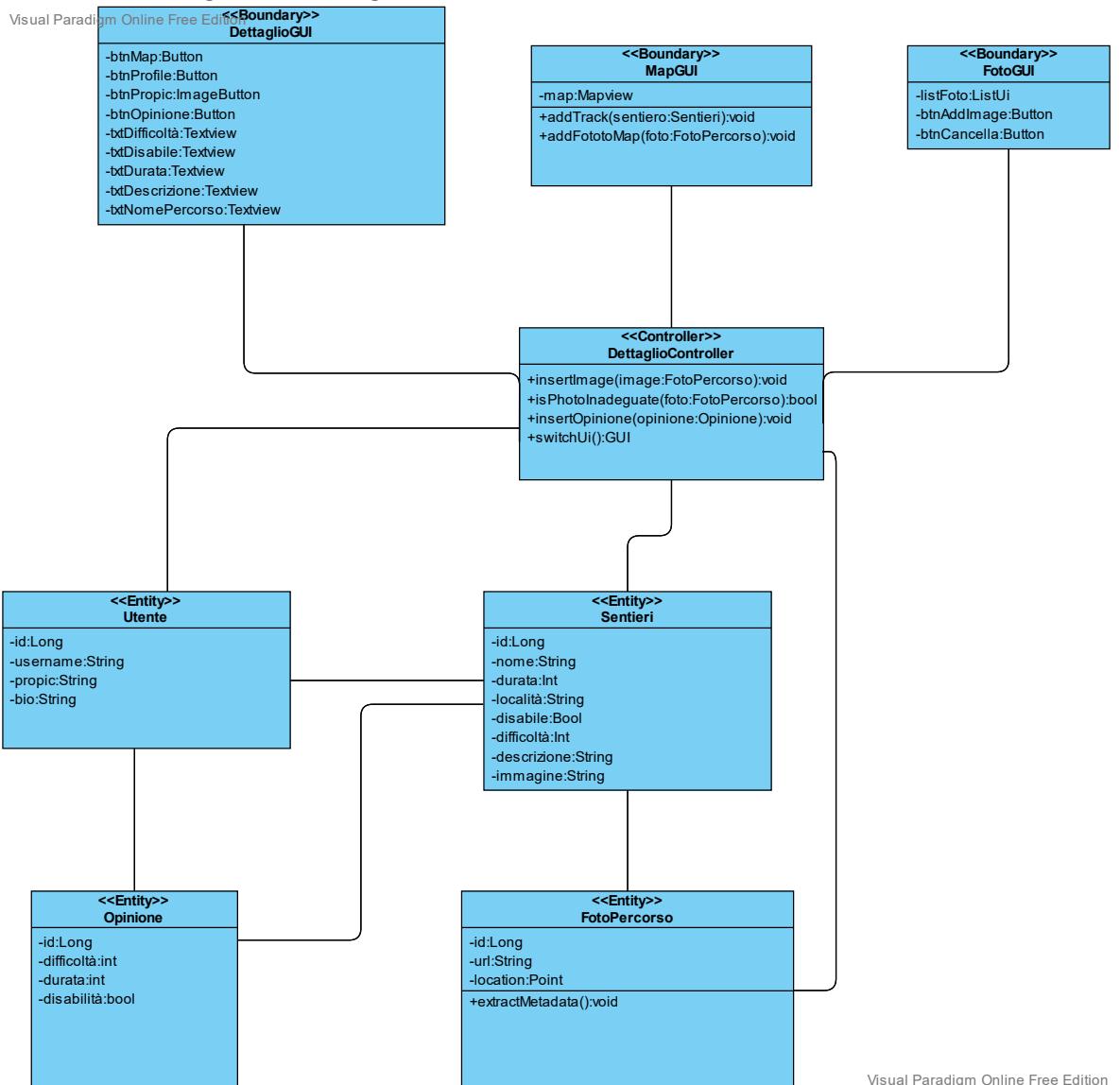
3.1.4 Class Diagram – Homepage



3.1.5 Class Diagram – Inserimento Percorso



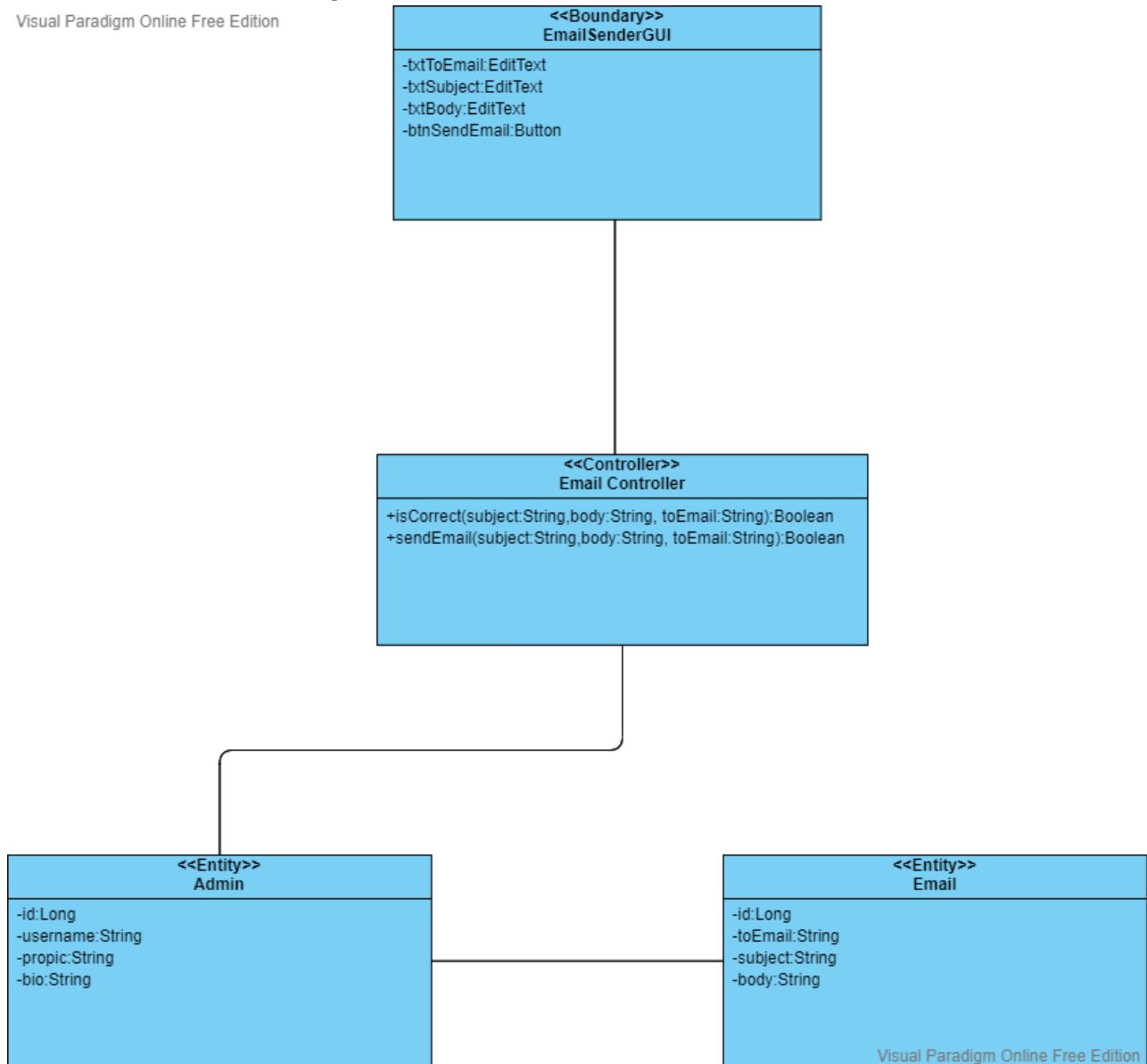
3.1.6 Class Diagram - Dettagli Percorso



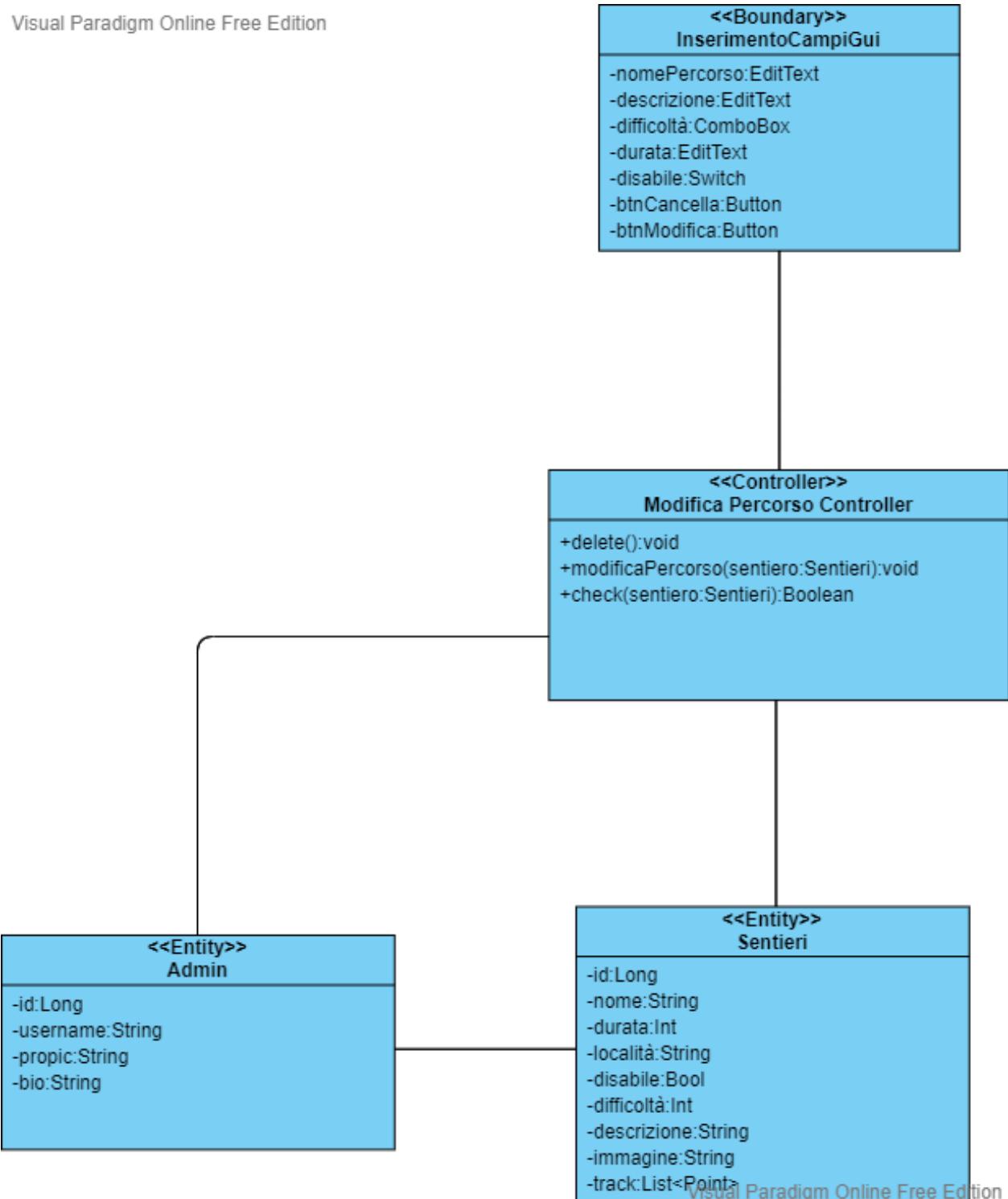
Visual Paradigm Online Free Edition

3.1.7 Class Diagram – E-mail Promozionale

Visual Paradigm Online Free Edition



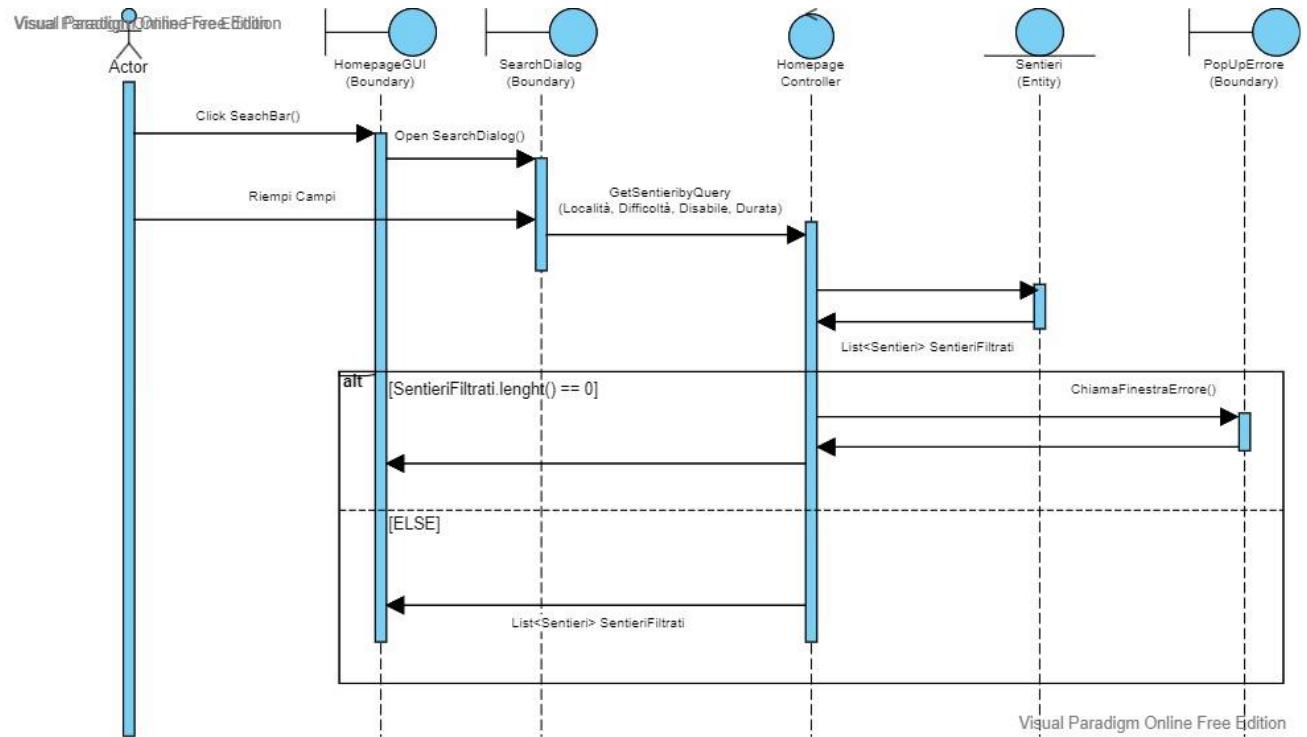
3.1.8 Class Diagram – Modifica/Elimina Percorso



3.2 Sequence Diagram

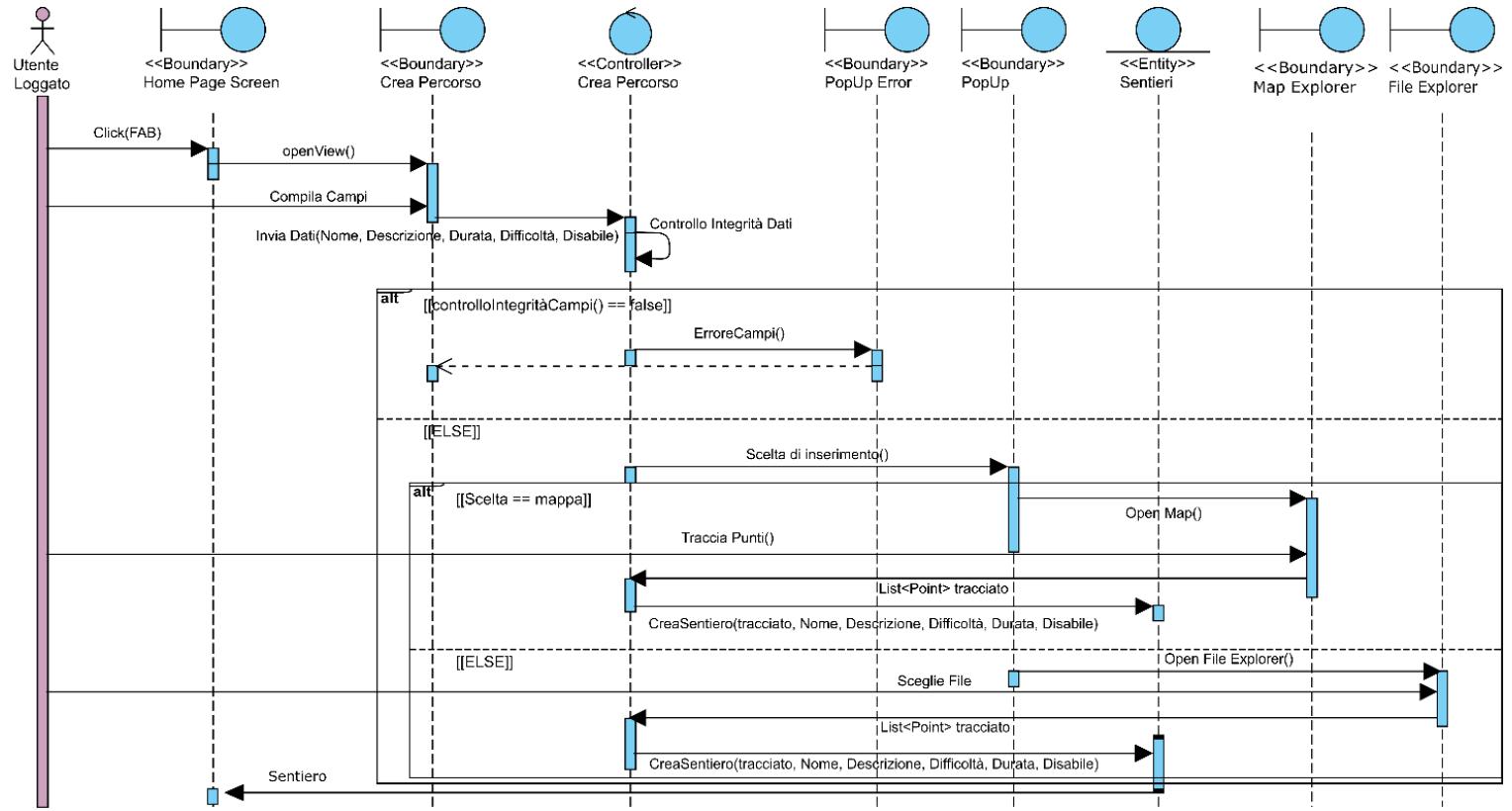
I sequence diagram stanno a rappresentare le interazioni del sistema con sé stesso o altri sistemi per far avvenire determinate azioni.

3.2.1 Sequence Diagram – Cerca Percorso



Visual Paradigm Online Free Edition

3.2.2 Sequence Diagram – Crea Percorso

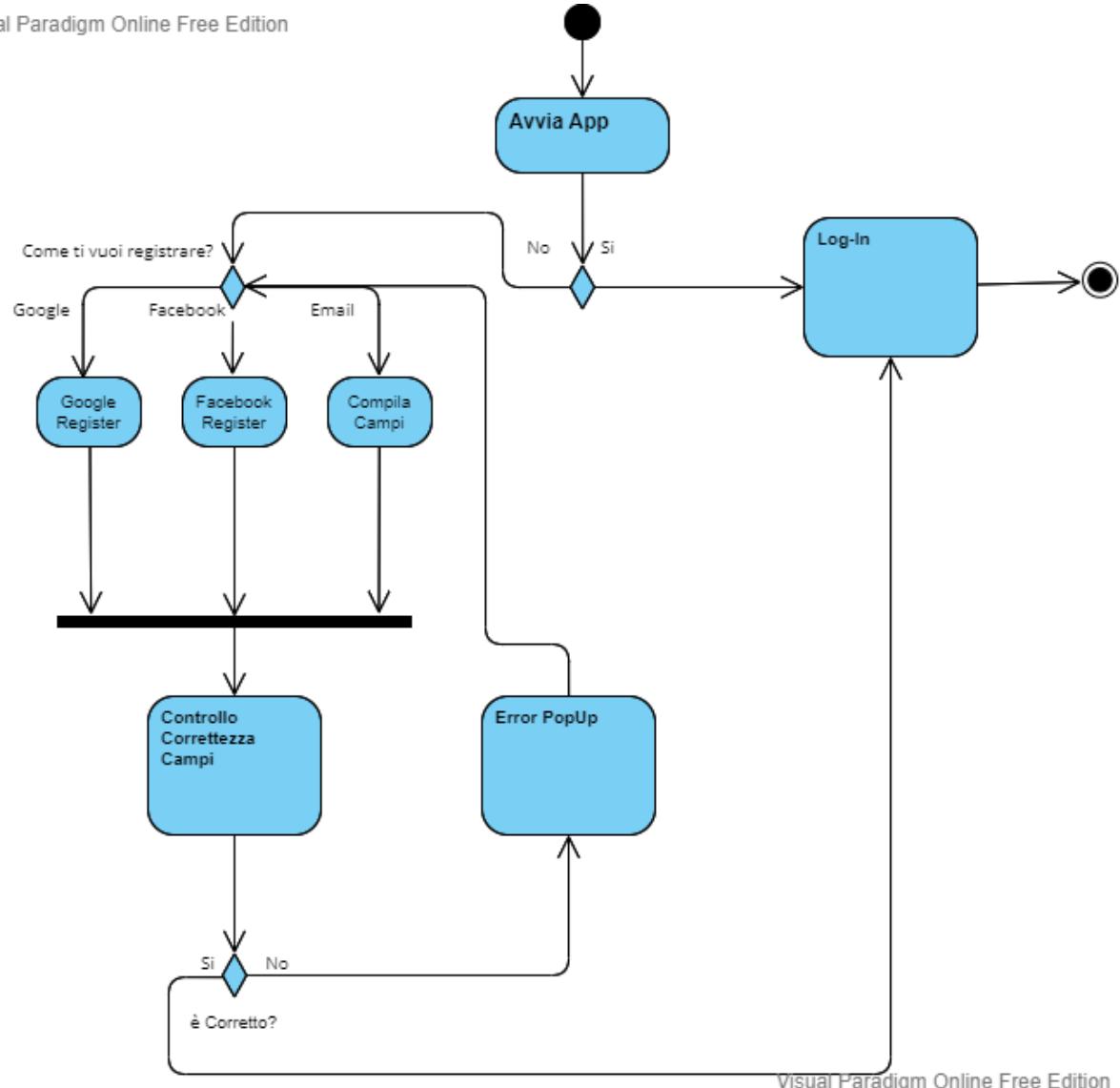


3.3 Activity Diagram

In questa sezione mostreremo gli activity diagram per tutte le funzionalità richieste da implementare.

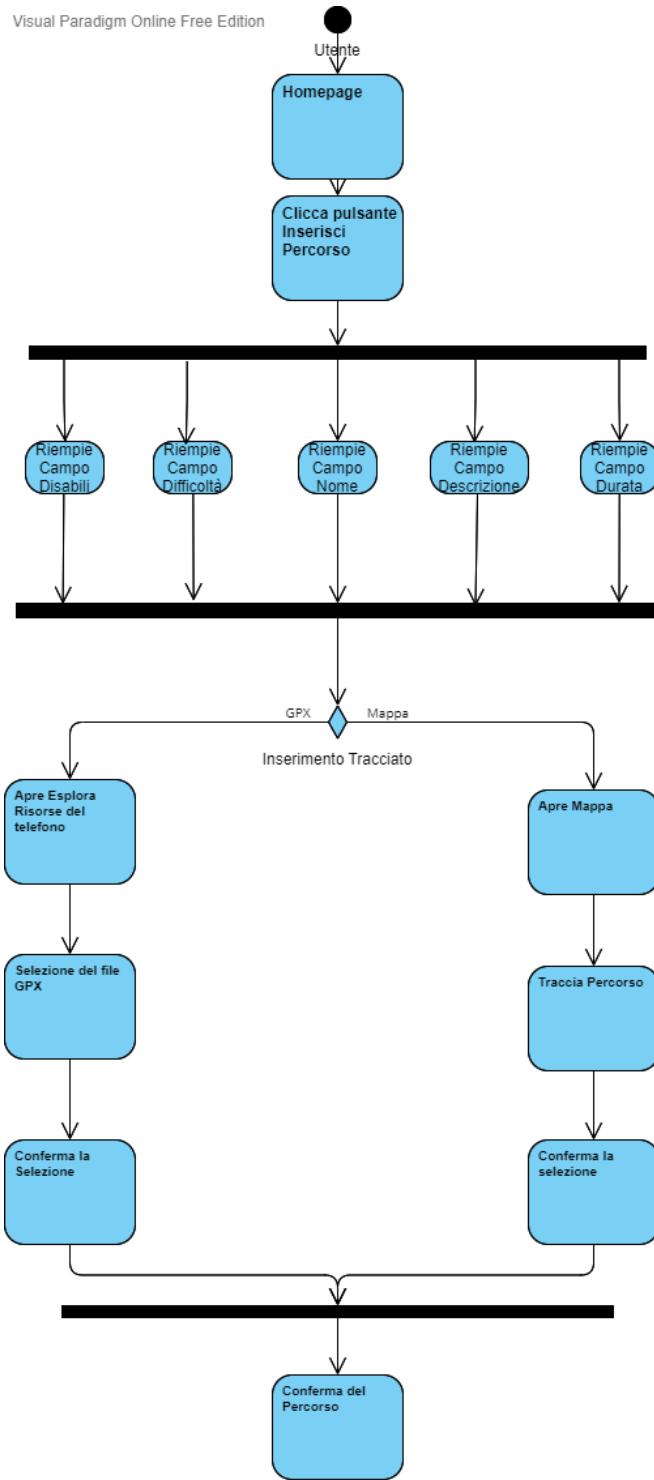
3.3.1 Registrazione Utente e Log-In

Visual Paradigm Online Free Edition



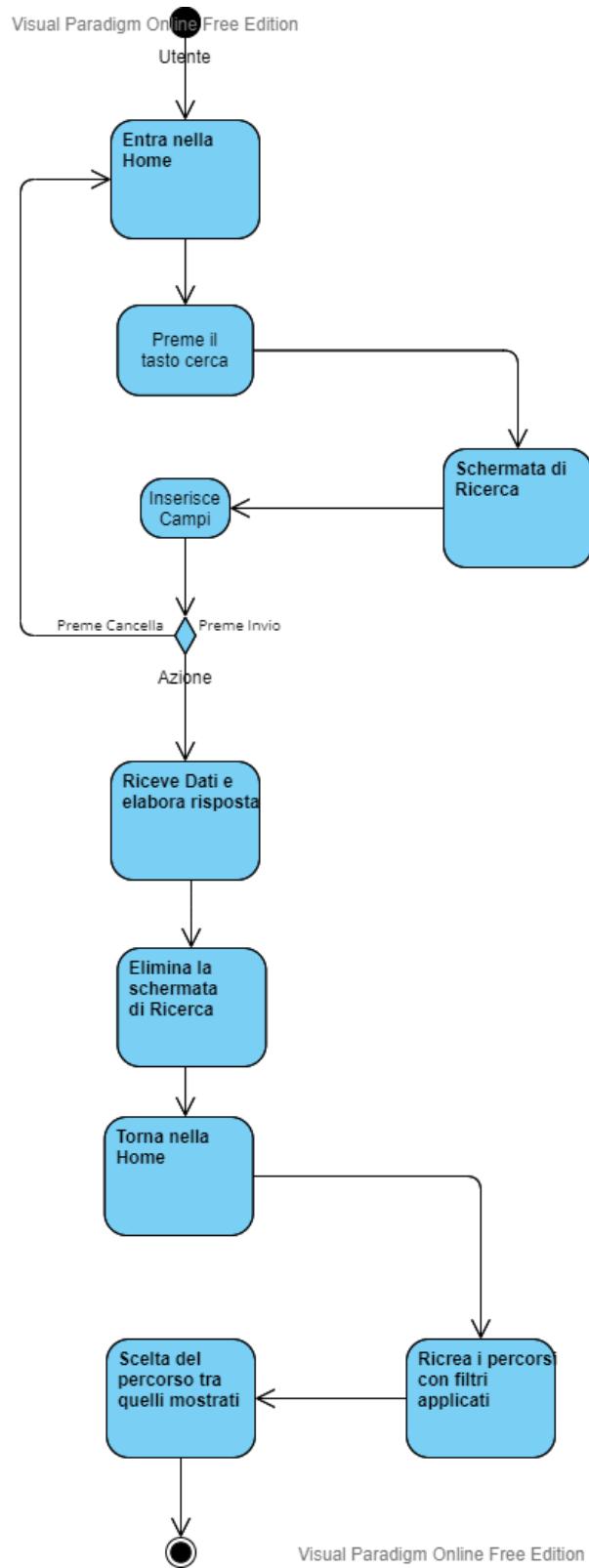
Visual Paradigm Online Free Edition

3.3.2 Crea Itinerario

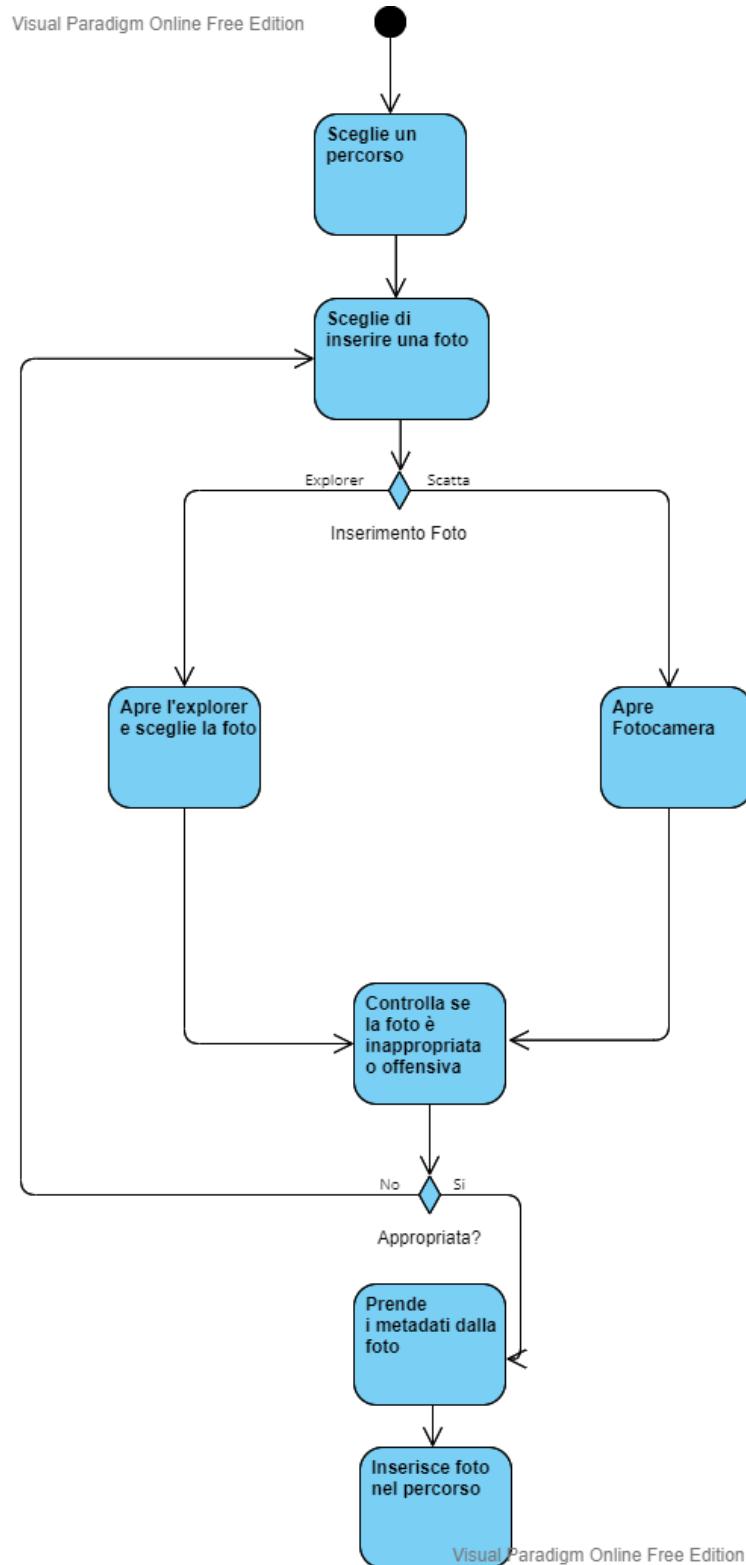


Visual Paradigm Online Free Edition

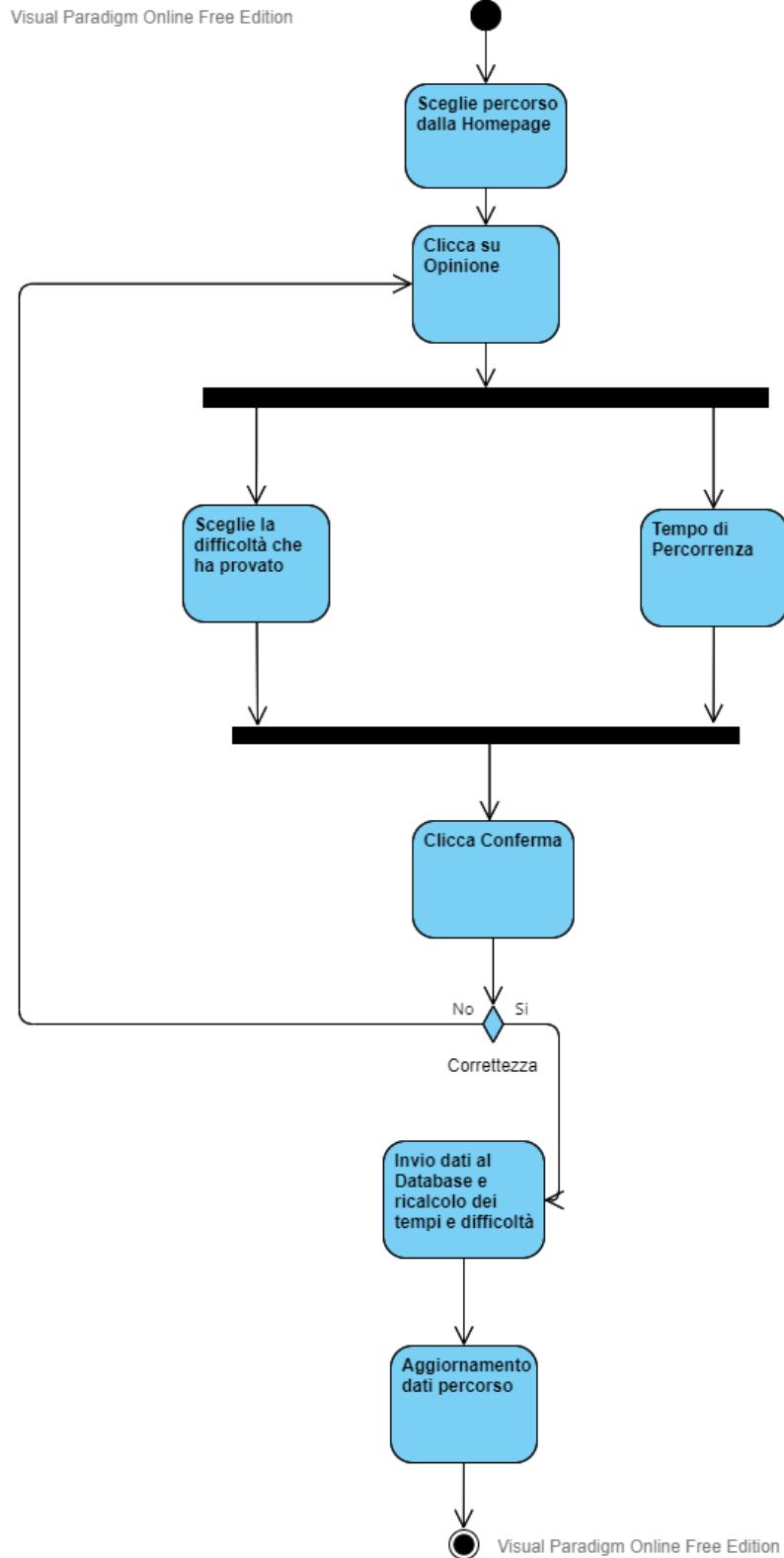
3.3.3 Cerca Itinerario



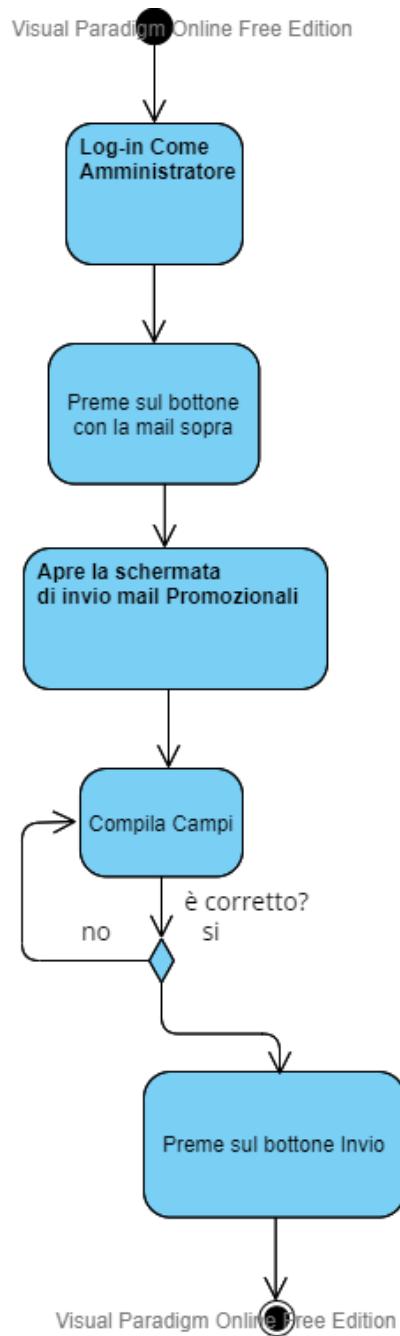
3.3.4 Caricamento Foto



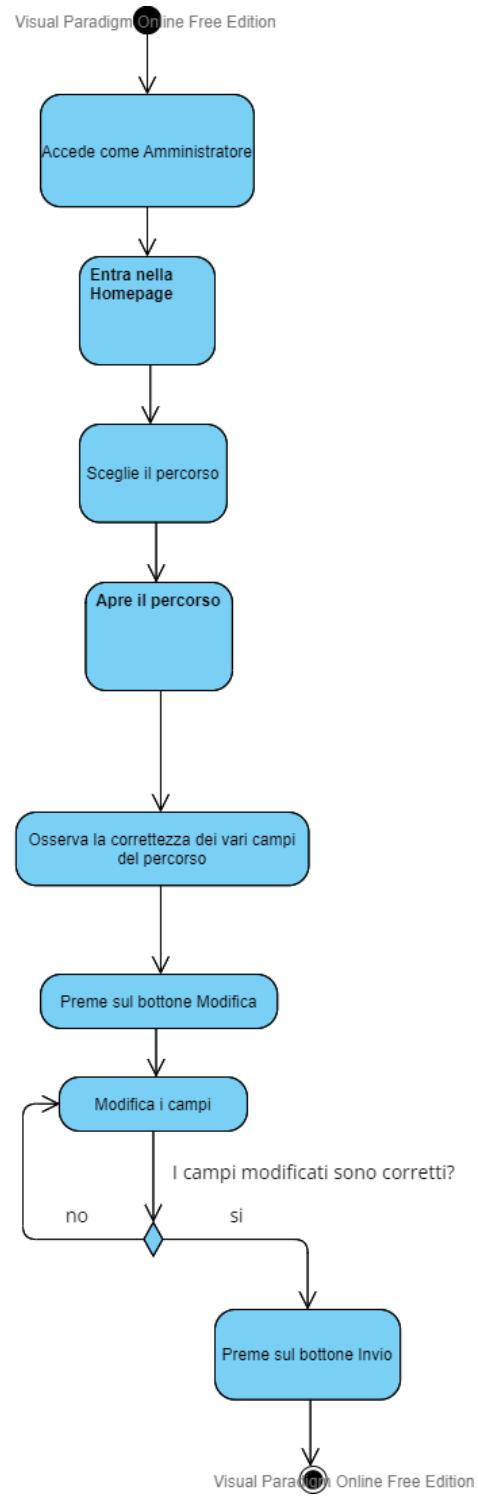
3.3.5 Opinioni Percorso



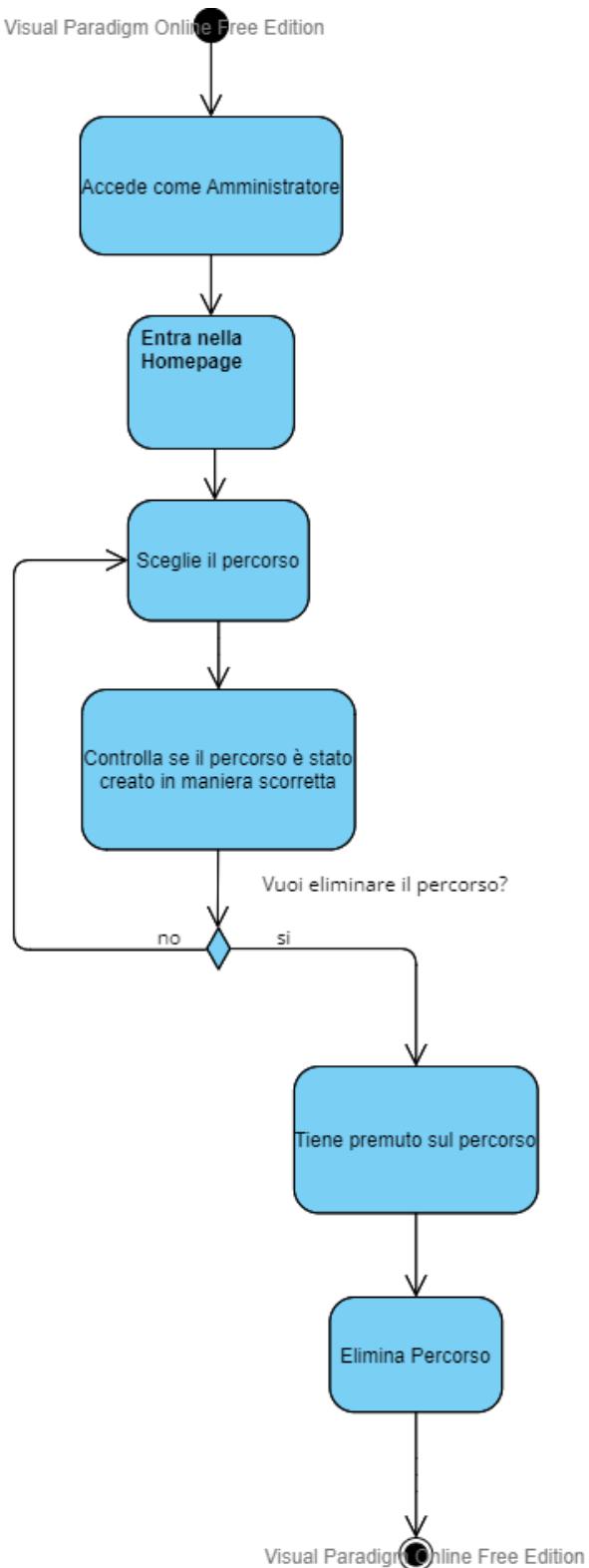
3.3.6 Mail Promozionali



3.3.7 Modifica Percorso



3.3.8 Elimina Percorso



4. Documento Di Design Del sistema

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

L'architettura usata per il progetto Na-Tour2021 prevede l'utilizzo di Java, linguaggio adoperato sia per l'applicazione mobile dell'applicazione, sia per il server che gestisce le richieste. L'utilizzo di Java ci è utile perché è un linguaggio semplice e flessibile ma soprattutto orientato agli oggetti, che ci ha facilitato la compatibilità su tutti i tipi di piattaforme da noi richiesti.

Design del progetto

La scelta per lo sviluppo del programma applicativo è stata affidata a tre prodotti che ci facilitano la creazione, revisione, versionamento e testing.

- **Android Studio:** per la parte di sviluppo dell'applicativo Android
- **IntelliJ IDEA:** per il lato server
- **Postman:** per il testare il funzionamento delle varie chiamate http verso gli endpoint del server

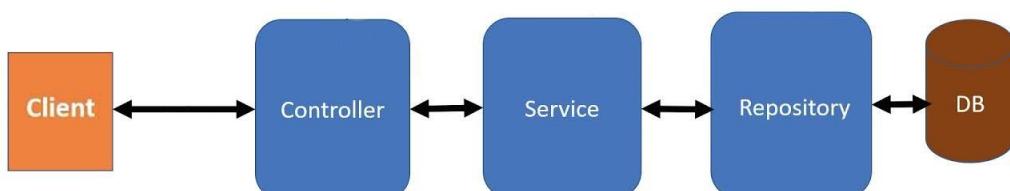
Backend

Per la gestione del server che sarà essenziale per i nostri servizi ci siamo affidati a uno dei framework più famosi in ambito Java, che dividerà la nostra applicazione in Elaboratore di dati e gestore delle richieste (Server) e Cliente (Android) il cui compito è semplicemente quello di effettuare richieste al server e formalizzare la risposta di ritorno.

- Spring Boot

Questo framework ci permette di creare diversi endpoint per le chiamate REST API (o representational state transfer style). Ci permette di effettuare scambi di informazioni in modo “lightweight” per il server, poiché si basa essenzialmente sulla comunicazione di dati via richieste HTTP, che riescono a performare funzioni di database standard (CRUD) per le risorse all’interno.

Abbiamo usato il classico layout che consiglia Springboot:



Possiamo notare che è suddiviso in tre principali layer:

- **Controller:** Gestisce gli endpoint per le chiamate http.
- **Service:** Funge da “controller” principale, è lui che si occupa della logica di tutto il backend.
- **Repository:** Serve a mettere in comunicazione il database relazionale con il Service, ci permette di astrarre tutte le chiamate a quest’ultimo.

Sono stati utilizzati inoltre:

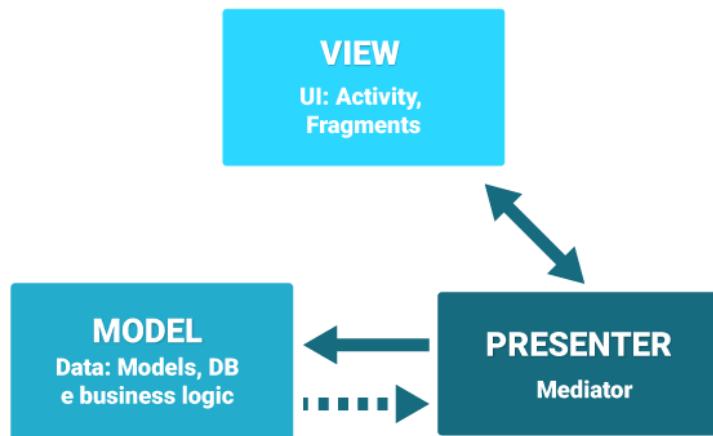
- **DTO:** Una semplificazione delle classi Model, per facilitare lo scambio di informazioni tra Client e Server.
- **Model Mapper:** Facilita la conversione da DTO a Model e viceversa.

Frontend

Per la mobile application abbiamo deciso di usare il design pattern MVP:

- Model View Presenter, comunemente molto più utilizzato per il mobile development del MVC per le seguenti caratteristiche:
 - La maggior parte della logica viene affidata al Controller. Durante una lifetime di un'applicazione, il Controller potrebbe diventare più grande e quindi più complicato da gestire
 - Controller e View dovranno sicuramente risiedere nello stesso fragment/activity per la loro interconnessione, rendendo difficili le modifiche in successione.
 - Per il testing diventa difficile testare i vari layer.

Model View Presenter



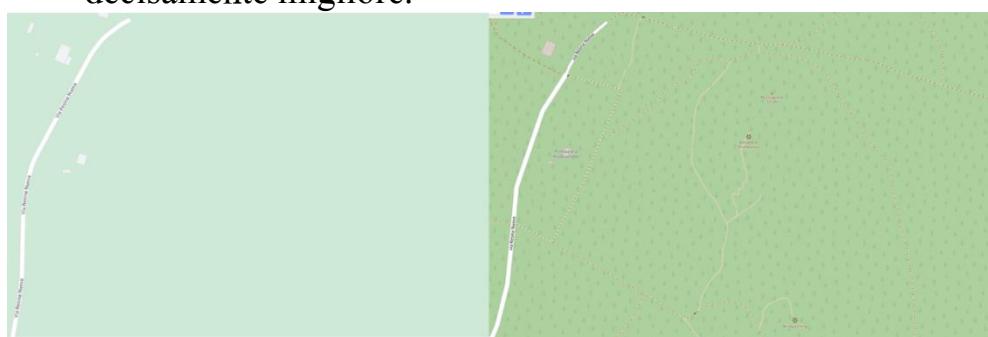
- **Model:** Un layer per l'immagazzinamento dei dati. È responsabile della comunicazione tra domino logico e comunicazione tra database.
- **View:** Layer di interfaccia grafica. Permette di visualizzare i dati e tenere traccia delle azioni.
- **Presenter:** Prende i dati dalla view e decide la logica di disposizione dell'UI, scegliendo come e cosa far vedere a schermo.

Ci sono stati utili queste librerie per aiutarci in alcune parti della programmazione:

Open Street Map: è una libreria Open Source utilizzata per le azioni e rappresentazioni sulla mappa dell'applicazione.

Vantaggi:

1. Abbiamo effettuato questa scelta rispetto a un Google Maps poiché essendo un'alternativa Open Source, possiede più dati riguardo percorsi più secondari creati da altri utenti così da facilitare anche noi con la creazione dei percorsi all'interno del Database.
2. Rispetto a Google Maps la visione dei percorsi di montagna o su zone non ben coperte da strade è decisamente migliore.



Google Maps

Open Street Maps

Retrofit: modulo di comunicazione tramite http per le chiamate REST.

Material Design 3: Nuovissima libreria grafica di Android, ci ha permesso di avere una app con un design molto moderno e minimale, inoltre avendo usato i kit di design nei mockup di Figma (vedi par [2.9](#)), ci ha permesso una veloce e semplice trasposizione da Figma ad app.

Amazon AWS Amplify: insieme di strumenti per developer di applicazioni per la comunicazione rapida con AWS e i suoi servizi. Ci è servito particolarmente per Cognito e S3.

Servizi Cloud

Amazon AWS Cognito: che ci aiuta nell'autenticazione degli utenti. Grazie all'uso di una sua interna pool di utenti che viene riempita ad ogni accesso registrato dell'applicazione, esso ci permette di tenere traccia di qualsiasi utente e admin all'interno del nostro software.

Ad ogni profilo utente viene generato un codice di unicità direttamente da Cognito, che permetterà allo stesso software, in modo interno, di effettuare accesso, generando in risposta un segnale positivo o negativo.

Ovviamente qualsiasi modalità o gestione degli spazi e delle connessioni è affidata allo stesso Cognito, che non ci permette tantomeno di poter metter mano nel proprio algoritmo.

Nonostante questa forma di black-box utilizzata da Amazon, abbiamo deciso comunque di immagazzinare, nei nostri server, anche per ottenere una forte ridondanza di dati, i codici identificativi dei nostri utenti, così da non dover essere costretti a fare affidamento solo e soltanto sul loro sistema.

Amazon AWS S3: ci permette di immagazzinare le foto che gli utenti caricano dal frontend, è comodamente diviso in bucket per poter separare i diversi tipi di file.

Amazon AWS Rekognition: non implementato, ci permette di filtrare le foto inopportune direttamente da server, prima di decidere di essere immagazzinate nel bucket di S3.

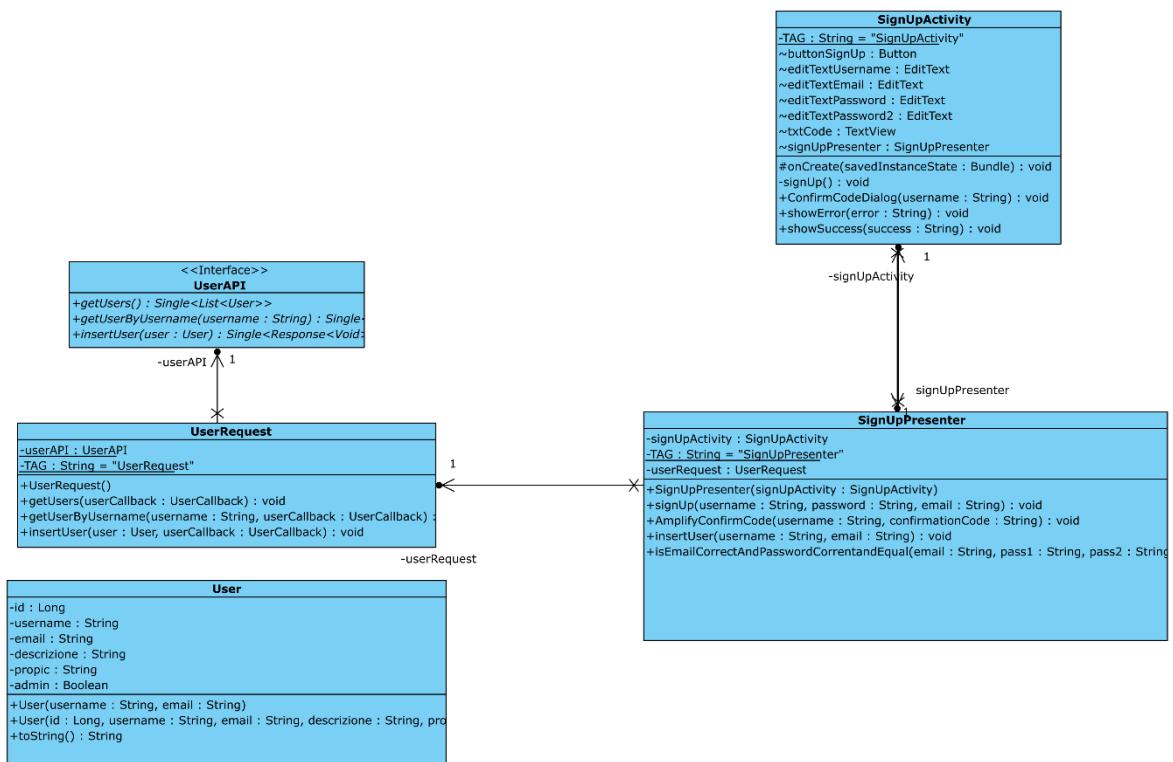
Amazon AWS EC2: ci permette di hostare una macchina virtuale su cui girare il server (SpringBoot) tramite una Java Virtual Machine. Il servizio offre la possibilità di scalare in base alle esigenze (ad esempio un boom di utenza) su una macchina più performante per gestire al meglio tutte le richieste.

Amazon AWS RDS: ci permette di hostare il database relazionale fondamentale per il funzionale del backend SpringBoot.

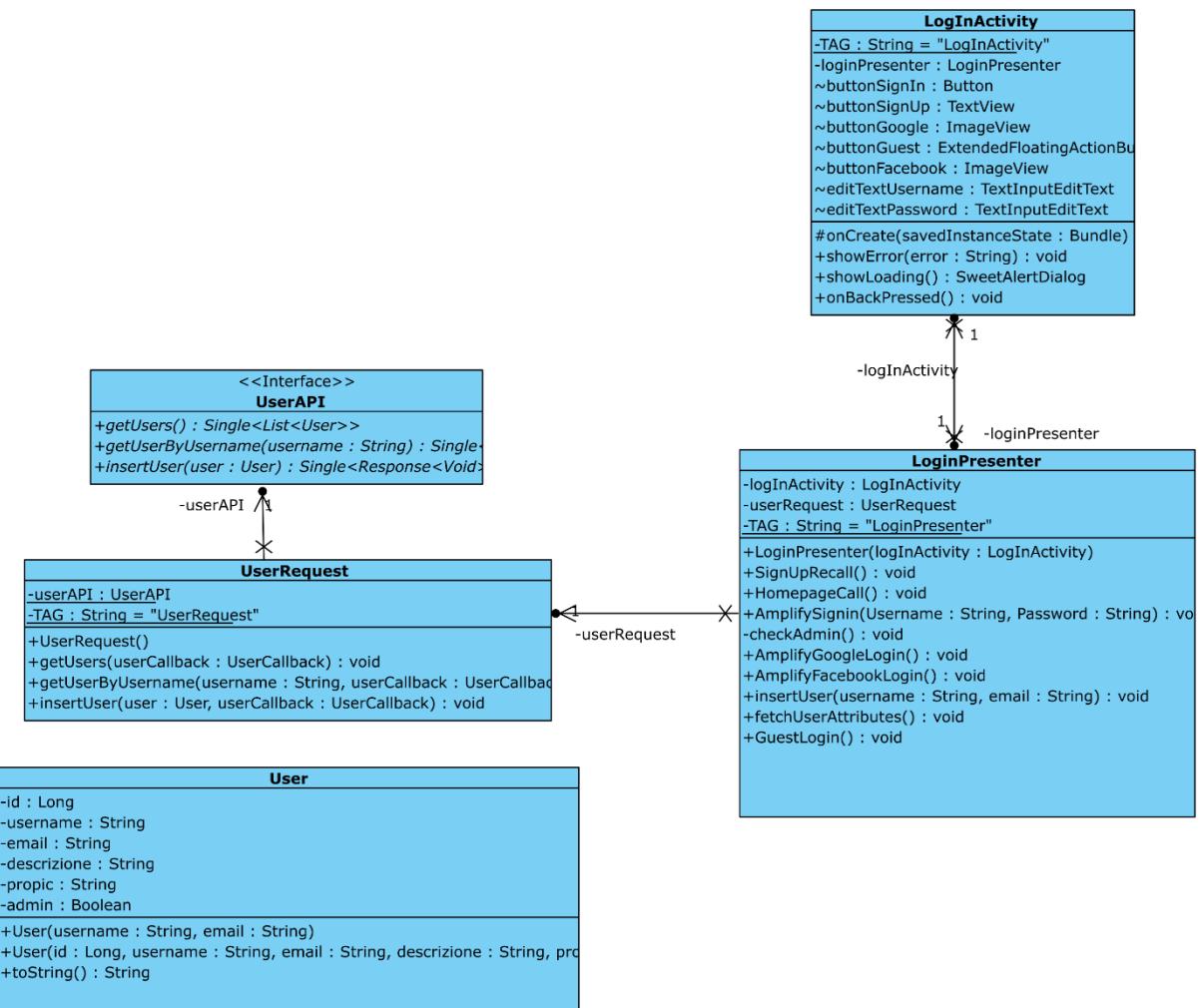
4.2 Diagramma delle Classi di Design

I diagrammi sono la versione implementata dei diagrammi presenti a [3.1](#)

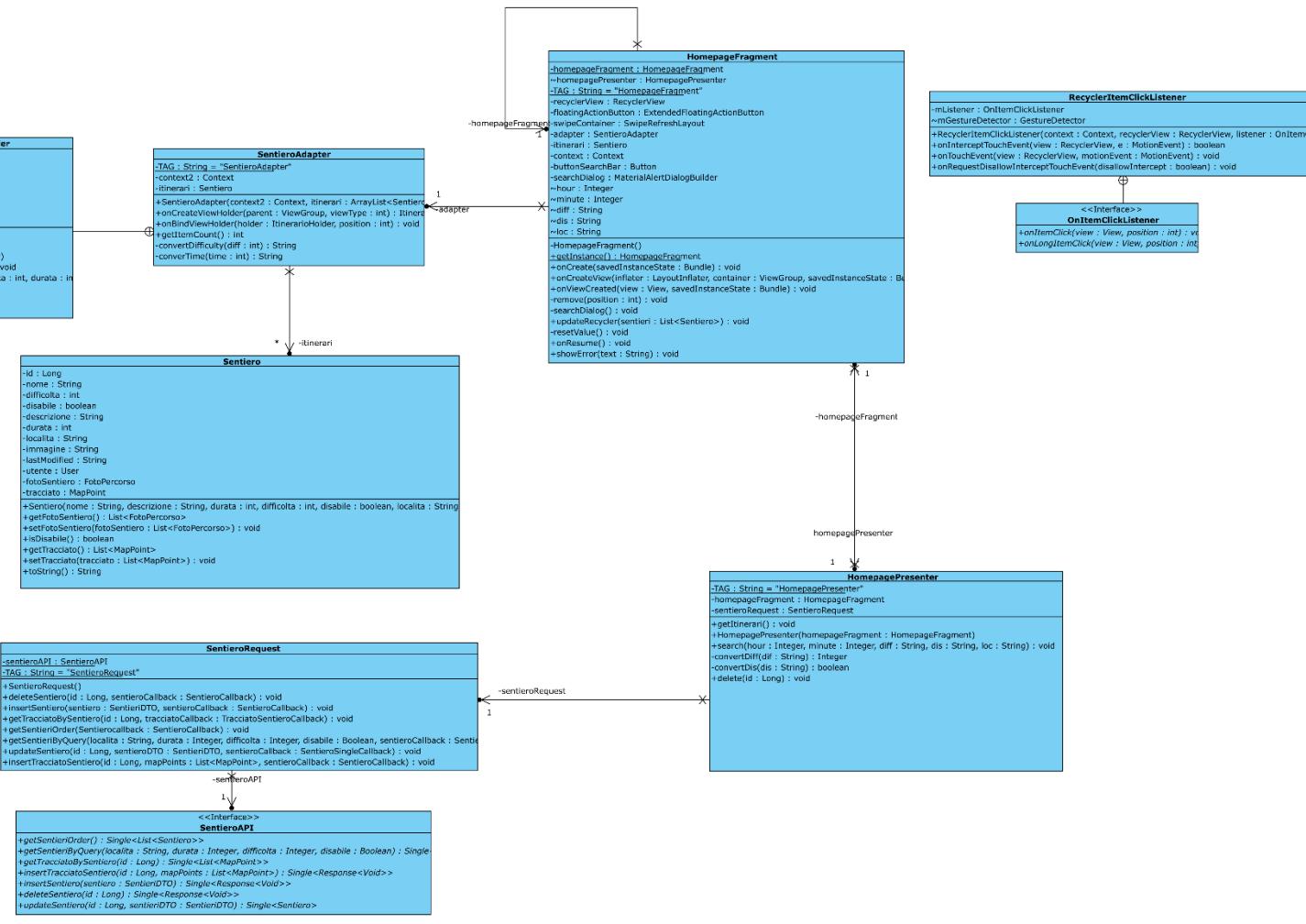
4.2.1 Class Diagram – Registrazione



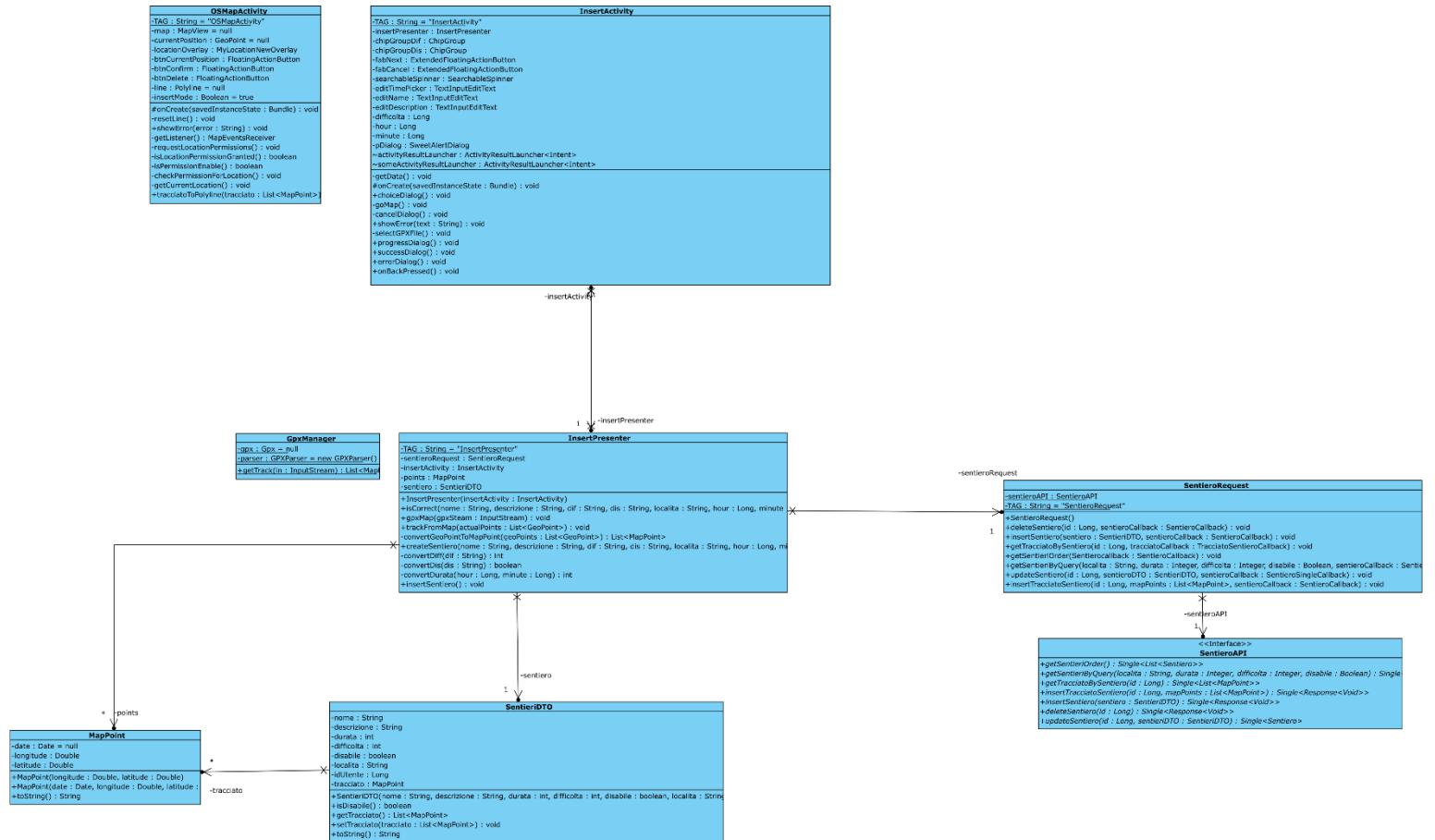
4.2.2 Class Diagram – Login



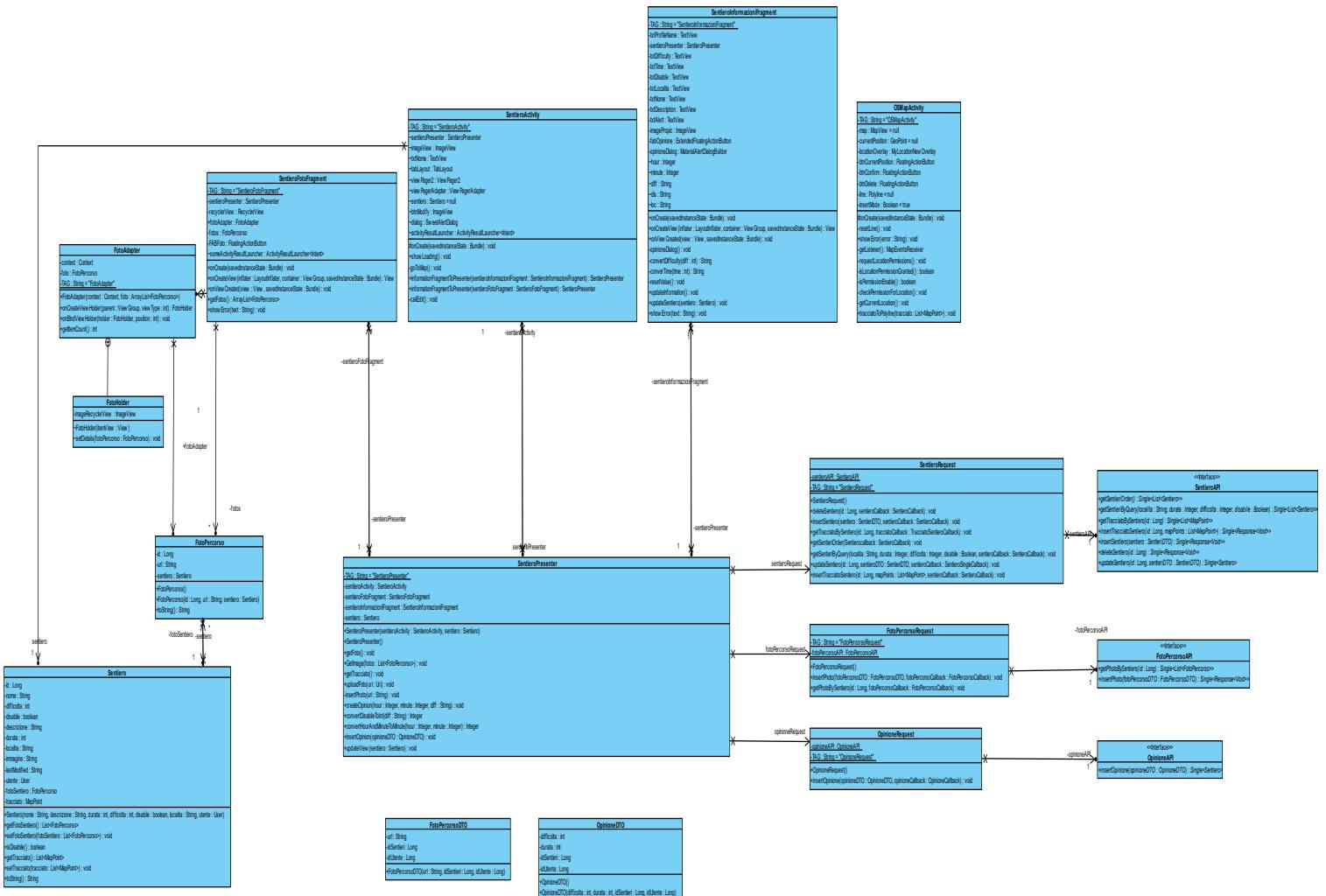
4.2.3 Class Diagram – Homepage



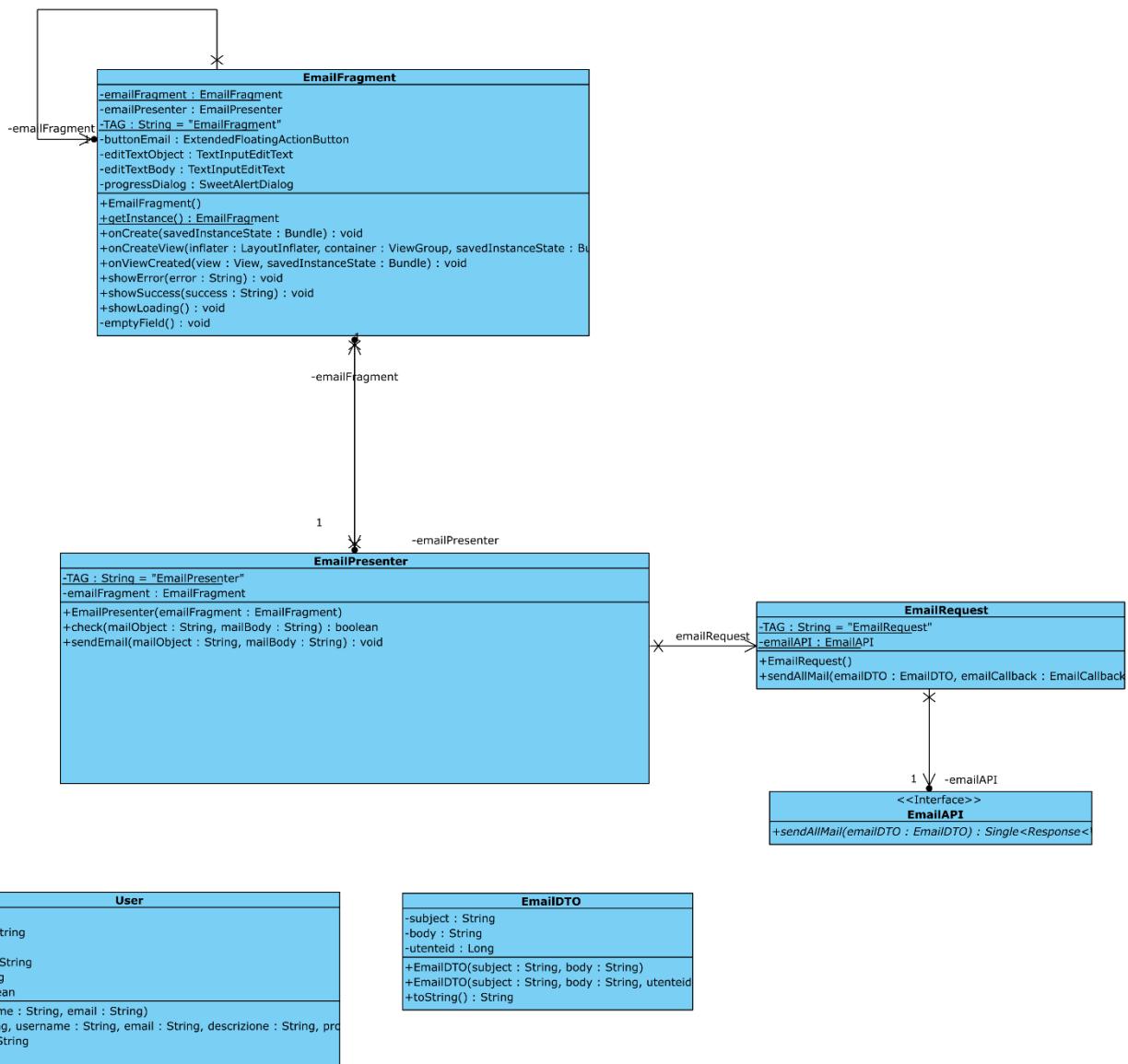
4.2.4 Class Diagram – Inserimento Percorso



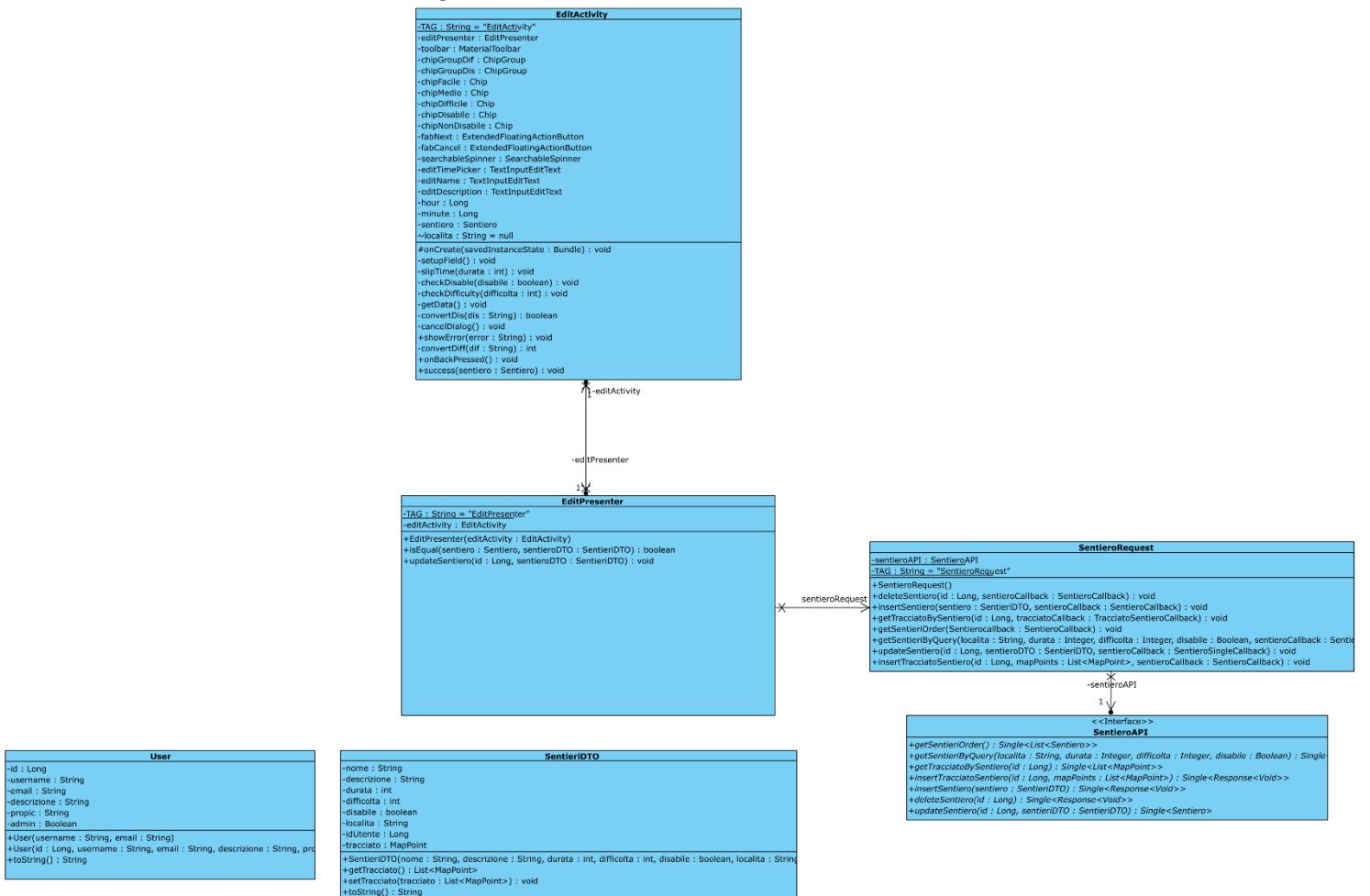
4.2.5 Class Diagram – Dettagli Sentiero



4.2.6 Class Diagram – E-mail Promozionale



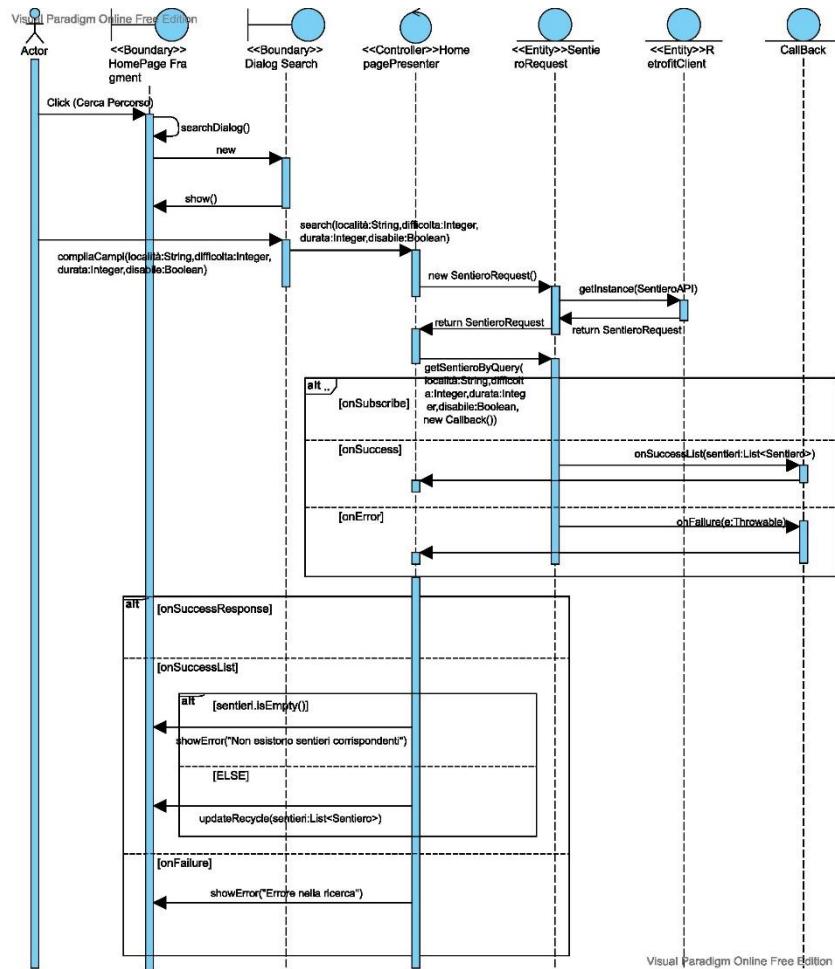
4.2.7 Class Diagram – Modifica/Elimina Percorso



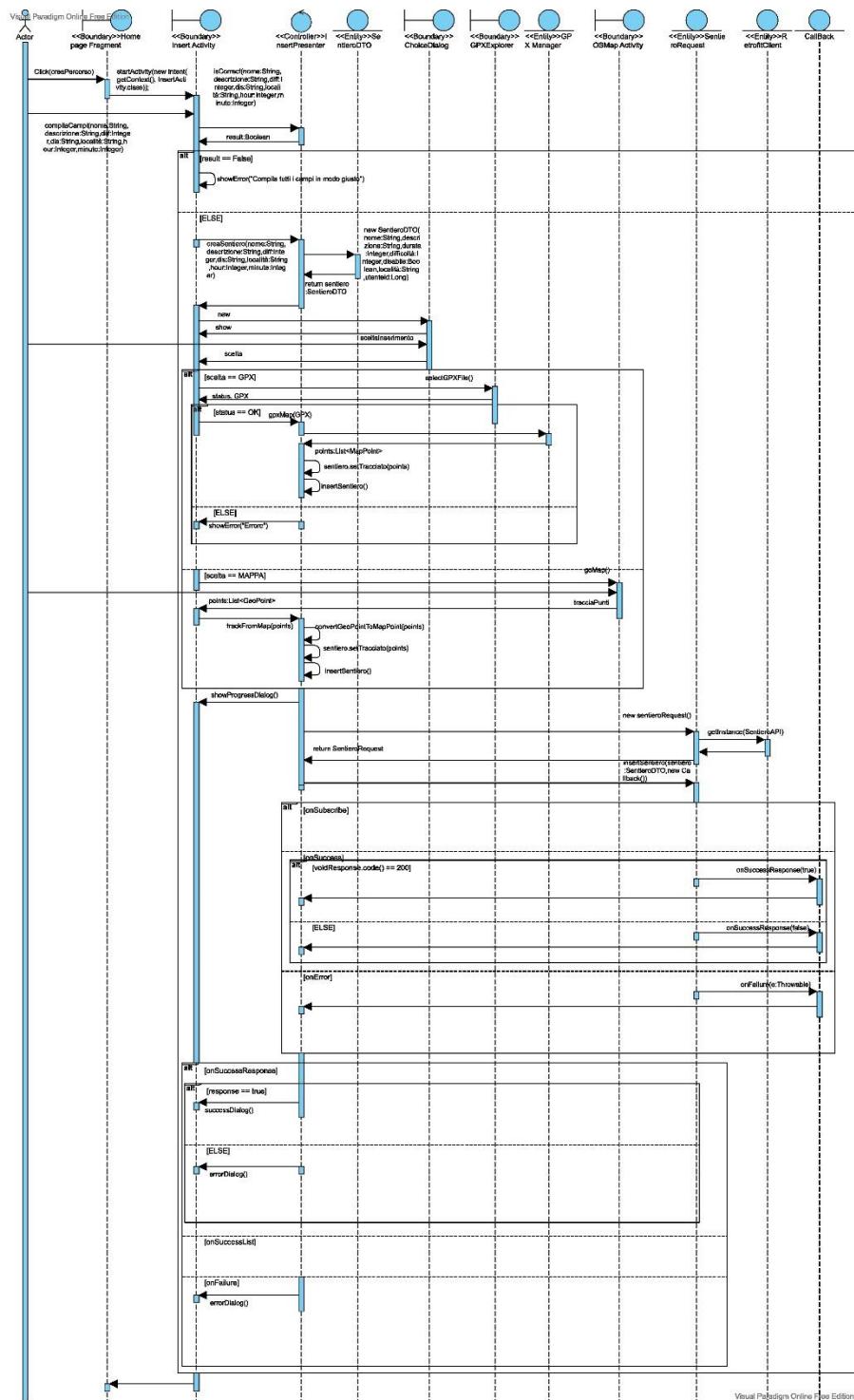
4.3 Diagrammi di Sequenza di Design per 2 casi d'uso significativi

I diagrammi sono la versione implementata dei diagrammi presenti a [3.2](#)

4.3.1 Cerca Sentiero



4.3.2 Crea Sentiero



4.4 Definizione delle Gerarchie Funzionali

In questo paragrafo presentiamo le gerarchie funzionali del sistema, che vanno a riassumere in un diagramma ad albero tutte le funzionalità di ogni tipo di utente.

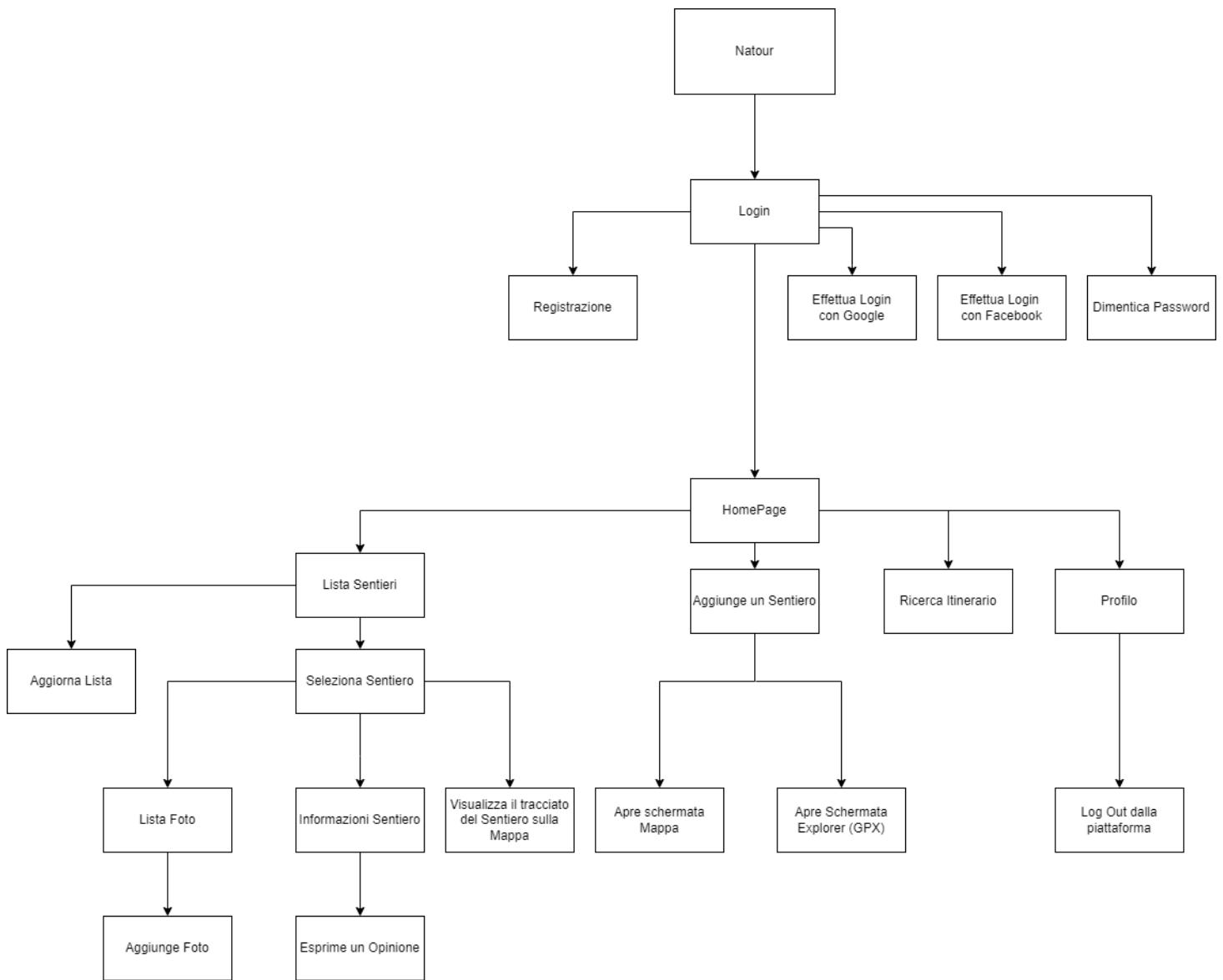
Nell'applicazione potranno entrare 3 diversi tipi di utente:

- **Utente:** Ha accesso all'applicazione standard. Ogni persona che si registra acquisisce questo grado.
- **Amministratore:** Ha gli stessi privilegi di un utente normale, ma ha poteri di modifica e di rimozione dei percorsi. Inoltre, può inviare e-mail promozionali. Agli admin viene fornito un account con i privilegi d'amministratore.
- **Guest:** è l'utente non loggato, può solo entrare nella piattaforma, cercare i percorsi e visualizzarli, ma non potrà creare i propri.

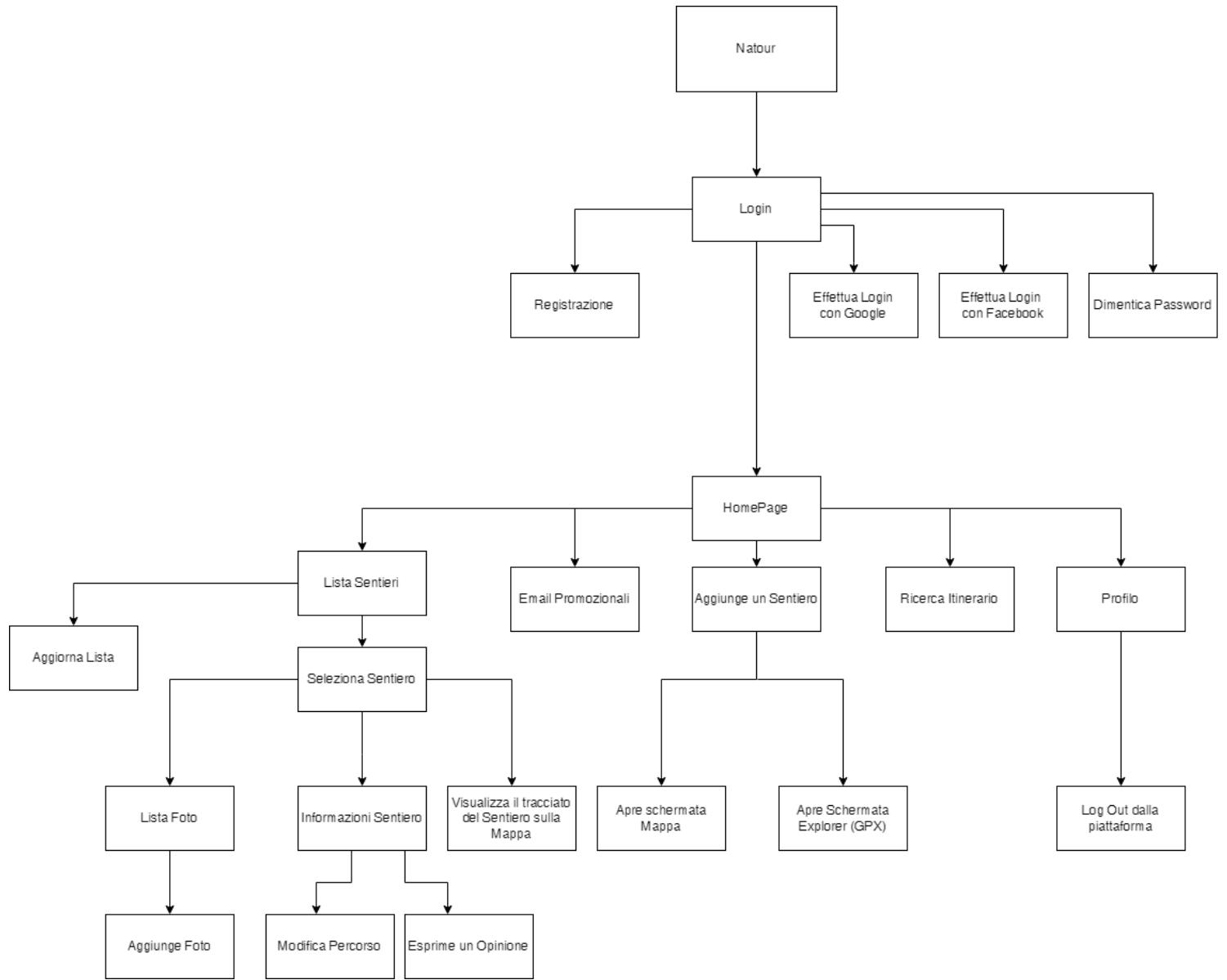
Il sistema si adatta graficamente a ognuno di questi 3 utenti, aggiungendo o rimuovendo funzionalità.

Di conseguenza allegiamo le 3 gerarchie funzionali:

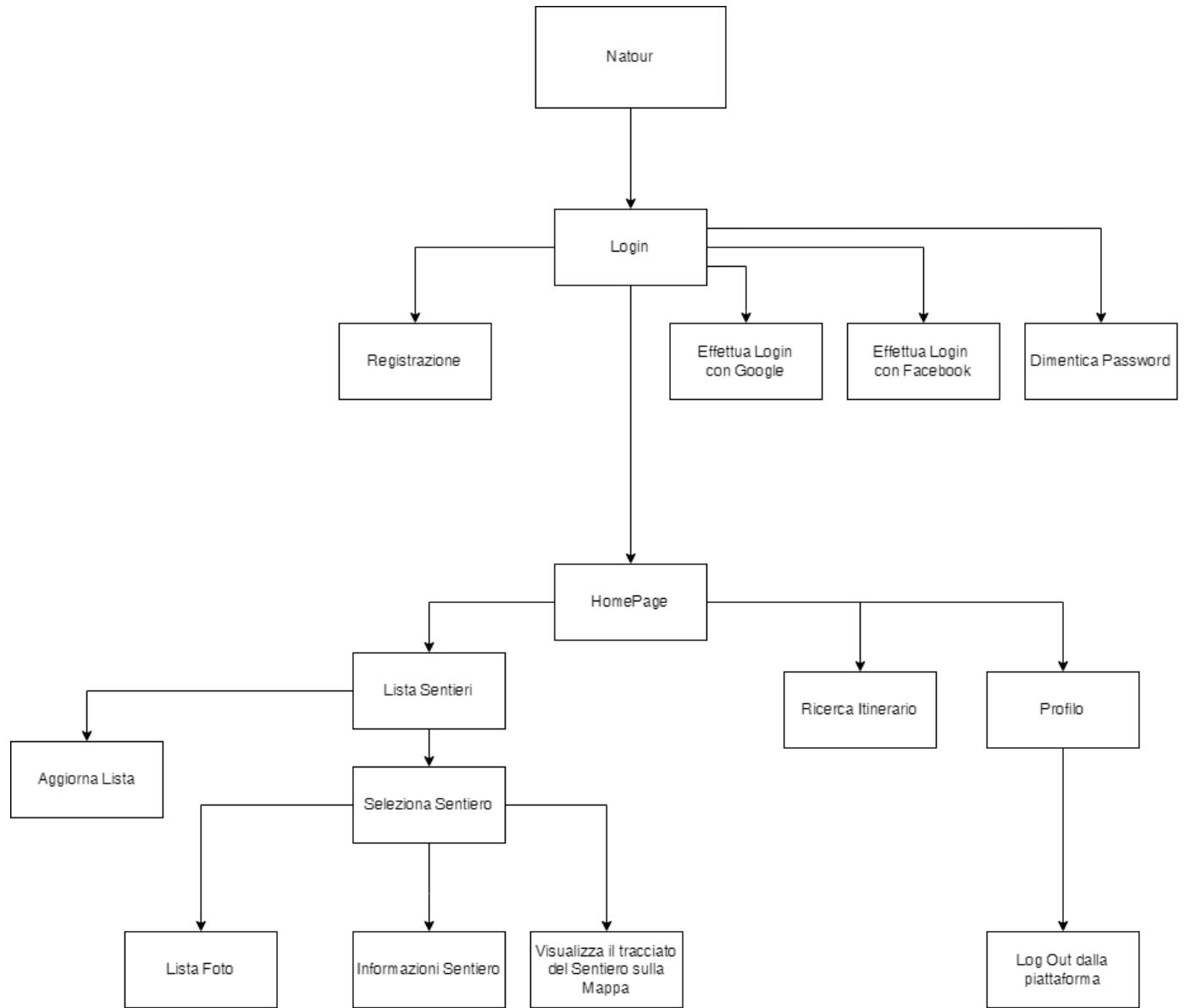
4.4.1 Gerarchia Funzionale Utente



4.4.2 Gerarchia funzionale Amministratore



4.4.3 Gerarchia Funzionale Guest



5. Codice Sorgente Sviluppato

Di seguito riportiamo il link al codice sorgente sviluppato: link che comprende sia parte Android che backend del software sviluppato: [Github](#)

6. Definizione di un piano di testing e valutazione sul campo dell’usabilità

Per un buon piano testing della nostra applicazione android abbiamo deciso di utilizzare il framework JUnit. Un framework molto famoso e facilmente usato per scrivere ed eseguire test esclusivamente su Java.

I metodi Scelti per il testing sono i seguenti:

- convertHourandMinutesToMinutes - funzione per convertire ora e minuti in un intero.
- isEmailCorrectAndPasswordCorrectandEqual – funzione per creare un nuovo utente all’interno della piattaforma.
- ConvertDataToUi – funzione per convertire dei valori numerici a stringhe formattate per U.I.

6.1 convertHourandMinuteToMinute

Per il testing di questo metodo abbiamo deciso di applicare un testing di tipo **White Box**. È un tipo di testing che ci permette di leggere il codice dei metodi, quindi avere dei criteri di copertura più precisi rispetto a un Black Box.

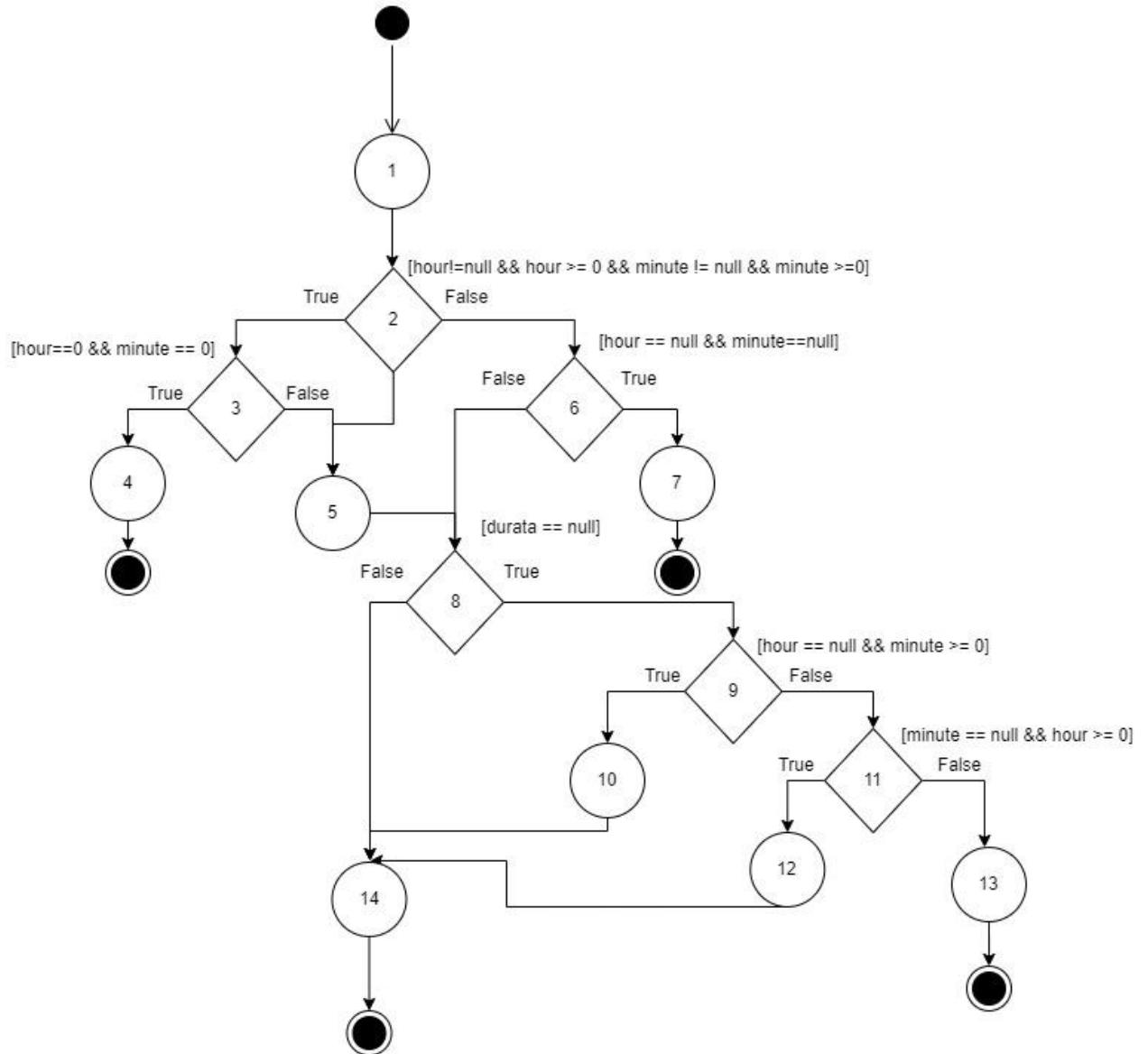
La chiave per un testing efficace per un metodo White Box è quello di tenere presente tre principi nell’esecuzione dei test:

- **Statement Coverage:** Dobbiamo assicurarci che ogni singola linea di codice è testata
- **Branch Coverage:** Dobbiamo assicurarci che ogni “ramo” del codice è testato
- **Path Coverage:** Dobbiamo assicurarci che tutti i possibili path sono testati

Questa è la funzione da testare. Per prima cosa numeriamo ogni operazione che viene eseguita.

```
public Integer convertHourAndMinuteToMinute(Integer hour, Integer minute) {  
    Integer durata = null; //1  
  
    if(hour!=null && hour >= 0 && minute != null && minute >=0) { //2  
        if(hour==0 && minute == 0) //3  
            throw new IllegalArgumentException(); //4  
        durata = (hour*60)+minute; //5  
    } else if(hour == null && minute==null) //6  
        throw new IllegalArgumentException(); //7  
  
    if(durata==null) { //8  
        if(hour == null && minute >= 0) { //9  
            durata = minute; //10  
        } else if(minute == null && hour >= 0) { //11  
            durata = hour*60; //12  
        } else throw new IllegalArgumentException(); //13  
    }  
  
    return durata; //14  
}
```

Utilizzando il grafo del flusso di controllo (CFG) andremo a rappresentare graficamente il metodo illustrato pocanzi.



Con il grafico possiamo scegliere una strategia di testing che ci permetta di testare tutti gli statement. Utilizzeremo quindi la strategia “White Box” – All Statement Coverage.

La strategia prevede il testing di ogni ramo del grafico CFG; quindi, testeremo tutte le possibili strade che porteranno alla fine del metodo.

```
package com.INGSW.NaTour.TestConvertHourandMinuteToMinute;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertThrows;

import com.INGSW.NaTour.Presenter.SentieroPresenter;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class TestConvertHourAndMinuteToMinute {

    /*
     * Strategia White Box - CFG All Branch Coverage
     * Hour:
     *   1) Giusto ( $\geq 0$ )
     *   2) Sbagliato ( $< 0$ )
     *   3) Sbagliato null
     * Minute:
     *   1) Giusto ( $\geq 0$ )
     *   2) Sbagliato ( $< 0$ )
     *   3) Sbagliato null
     */
    SentieroPresenter sentieroPresenter;

    @Before
    public void setUp(){
        sentieroPresenter = new SentieroPresenter();
    }

    @After
    public void tearDown(){
        sentieroPresenter = null;
    }
}
```

```

@Test
public void testCorretto() { //1,2,3,5,8,14
    Integer expected = new Integer(70);
    assertEquals(expected, sentieroPresenter.convertHourAndMinuteToMinute(1,10));
}

@Test
public void testZeroZero() { //1,2,3,4
    assertThrows(IllegalArgumentException.class, () -> sentieroPresenter.convertHourAndMinuteToMinute(0,0));
}

@Test
public void testNULLNULL() { //1,2,6,7
    assertThrows(IllegalArgumentException.class, () -> sentieroPresenter.convertHourAndMinuteToMinute(null,null));
}

@Test
public void testNULLMinore0() { //1,2,6,8,9,11,13
    assertThrows(IllegalArgumentException.class, () -> sentieroPresenter.convertHourAndMinuteToMinute(null,-10));
}

@Test
public void testNULLGiusto() { //1,2,6,8,9,10,14
    Integer expected = new Integer(10);
    assertEquals(expected,sentieroPresenter.convertHourAndMinuteToMinute(null,10));
}

@Test
public void testGiustoNULL() { //1,2,6,8,9,11,12,14
    Integer expected = new Integer(120);
    assertEquals(expected,sentieroPresenter.convertHourAndMinuteToMinute(2,null));
}

}

```

- testCorretto() copre 1,2,3,5,8,14
- testZeroZero() copre 1,2,3,4
- testNULLNULL() copre 1,2,6,7
- testNULLMinore0() copre 1,2,6,8,9,11,13
- testNULLGiusto() copre 1,2,6,8,9,10,14
- testGiustoNULL() copre 1,2,6,8,9,11,12,14

6.2 isEmailCorrectAndPasswordCorrectandEqual

Per questo metodo abbiamo deciso di applicare il testing Black Box, che consiste nel testare la funzione senza sapere come funziona all'interno. Dunque, dobbiamo definire il dominio di tutti i possibili campi per poterli testare.

Analizziamo la firma della funzione:

```
public boolean isEmailCorrectAndPasswordCorrectandEqual  
(String email, String pass1, String pass2)
```

Possiamo notare che prende tre input di tipo String, quindi per ogni attributo andiamo a trovare le classi di equivalenza:

```
email:  
    CEE1) email valida      Esempio: "alex@gmail.com"  
    CEE2) email non valida Esempio: "@gmail.com"  
    CEE3) email nulla       Esempio: null  
    CEE4) email vuota       Esempio: ""  
  
password 1:  
    CEP1) pass valida      Esempio: "Pass@1234"  
    CEP2) pass non valida  Esempio: "pass"  
    CEP3) pass vuota       Esempio: ""  
    CEP4) pass null         Esempio: null  
  
password 2:  
    CEPP1) pass valida     Esempio: "Pass@1234"  
    CEPP2) pass non valida Esempio: "pass"  
    CEPP3) pass vuota      Esempio: ""  
    CEPP4) pass null        Esempio: null
```

Abbiamo preferito usare la strategia “WECT” e non “SECT” poiché altrimenti avremmo avuto un numero di test pari al prodotto cartesiano di tutte le classi di equivalenza ($4^3 = 64$) Invece con WECT eseguiamo un test con tutti i valori validi e poi un test per ogni classe d’equivalenza non valida.

Qui di sotto lasciamo tutti i test trovati:

- Caso 1) Email vuota, pass1 vuota, pass2 vuota
- Caso 2) Email vuota, pass1 vuota, pass2 piena(valida)
- Caso 3) Email vuota, pass1 piena (valida), pass2 vuota
- Caso 4) Email vuota, pass1 piena (valida), pass2 piena (valida)
- Caso 5) Email vuota, pass1 piena (non valida), pass2 piena (non valida)
- Caso 6) Email vuota, pass1 piena (valida), pass2 (non valida)
- Caso 7) Email vuota, pass1 piena (non valida), pass2 (valida)
- Caso 8) Email piena (valida), pass1 vuota, pass2 vuota
- Caso 9) Email piena (valida), pass1 piena (valida), pass2 piena (valida)
- Caso 10) Email piena (non valida), pass1 piena (valida), pass2 piena (valida)

E di seguito lasceremo il codice del testing applicato alla funzione:

```
package com.INGSW.NaTour.TestEmailPassword;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class TestEmailPassword {

    /*
        Strategia Black Box - WECT
        variabili : email, pass1, pass2
        Per password giusta si intende una password che segui questi standard di sicurezza:
            Deve avere almeno 8 caratteri, 1 maiuscola, 1 carattere speciale e 1 numero
        email:
            CEE1) email valida          Esempio: "alex@gmail.com"
            CEE2) email non valida      Esempio: "@gmail.com"
            CEE3) email nulla           Esempio: null
            CEE4) email vuota           Esempio: ""
        password 1:
            CEP1) pass valida          Esempio: "Pass@1234"
            CEP2) pass non valida      Esempio: "pass"
            CEP3) pass vuota           Esempio: ""
            CEP4) pass null             Esempio: null
        password 2:
            CEPP1) pass valida          Esempio: "Pass@1234"
            CEPP2) pass non valida      Esempio: "pass"
            CEPP3) pass vuota           Esempio: ""
            CEPP4) pass null             Esempio: null
        Caso 1) Email vuota, pass1 vuota, pass2 vuota //false
        Caso 2) Email vuota, pass1 vuota, pass2 piena(valida) // false
        Caso 3) Email vuota, pass1 piena (valida), pass2 vuota //false
        Caso 4) Email vuota, pass1 piena (valida), pass2 piena (valida) //false
        Caso 5) Email vuota, pass1 piena (non valida), pass2 piena (non valida)
    // false
        Caso 6) Email vuota, pass1 piena (valida), pass2 (non valida) //false
        Caso 7) Email vuota, pass1 piena (non valida), pass2 (valida) //false
        Caso 8) Email piena (valida), pass1 vuota, pass2 vuota //false
        Caso 9) Email piena (valida), pass1 piena (valida), pass2 piena (valida)
    //true
        Caso 10) Email piena (non valida), pass1 piena (valida), pass2 piena (valida) //false
}
```

```

*/
SignUpPresenterStub signUpPresenterStub;

@Before
public void init() {
    signUpPresenterStub = new SignUpPresenterStub();
}

@After
public void clear() {
    signUpPresenterStub = null;
}

@Test
public void testCorretto() {
    assertTrue(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual("alex@gmail.com", "Pass@1234", "Pass@1234"));
}

@Test
public void testInvalidEmail() {
    assertFalse(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual("gmail.com", "Pass@1234", "Pass@1234"));
}

@Test
public void testEmailNull() {
    assertFalse(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual(null, "Pass@1234", "Pass@1234"));
}

@Test
public void testEmailBlank() {
    assertFalse(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual("", "Pass@1234", "Pass@1234"));
}

@Test
public void testInvalidPass1() {
    assertFalse(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual("alex@gmail.com", "pass", "Pass@1234"));
}

@Test
public void testBlankPass1() {
    assertFalse(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual("alex@gmail.com", "", "Pass@1234"));
}

```

```

    @Test
    public void testNullPass1() {
        assertFalse(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual("alex@gmail.com", null, "Pass@1234"));
    }

    @Test
    public void testInvalidPass2() {
        assertFalse(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual("alex@gmail.com", "Pass@1234", "pass"));
    }

    @Test
    public void testBlankPass2() {
        assertFalse(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual("alex@gmail.com", "Pass@1234", ""));
    }

    @Test
    public void testNullPass2() {
        assertFalse(signUpPresenterStub.isEmailCorrectAndPasswordCorrentan-
dEqual("alex@gmail.com", "Pass@1234", "pass"));
    }
}

```

La classe utilizzata nel test è una classe Stub, perché nella funzione originale erano presenti diverse chiamate alla UI, impossibili da replicare in un test.

6.3 Data Sentiero To UI

Per questo metodo abbiamo deciso di usare di nuovo la strategia White Box con metodo CFG.

Alleghiamo il codice del metodo da testare per poi creare il grafico:

```
package com.INGSW.NaTour.TestDataSentieroToUI;

import org.junit.Test;

import java.util.ArrayList;
import java.util.Arrays;

public class SentieroAdapterStub {

    public ArrayList<String> convertDataSentieroToUI(int difficolta,
int durata) {
        String format, diff;//1

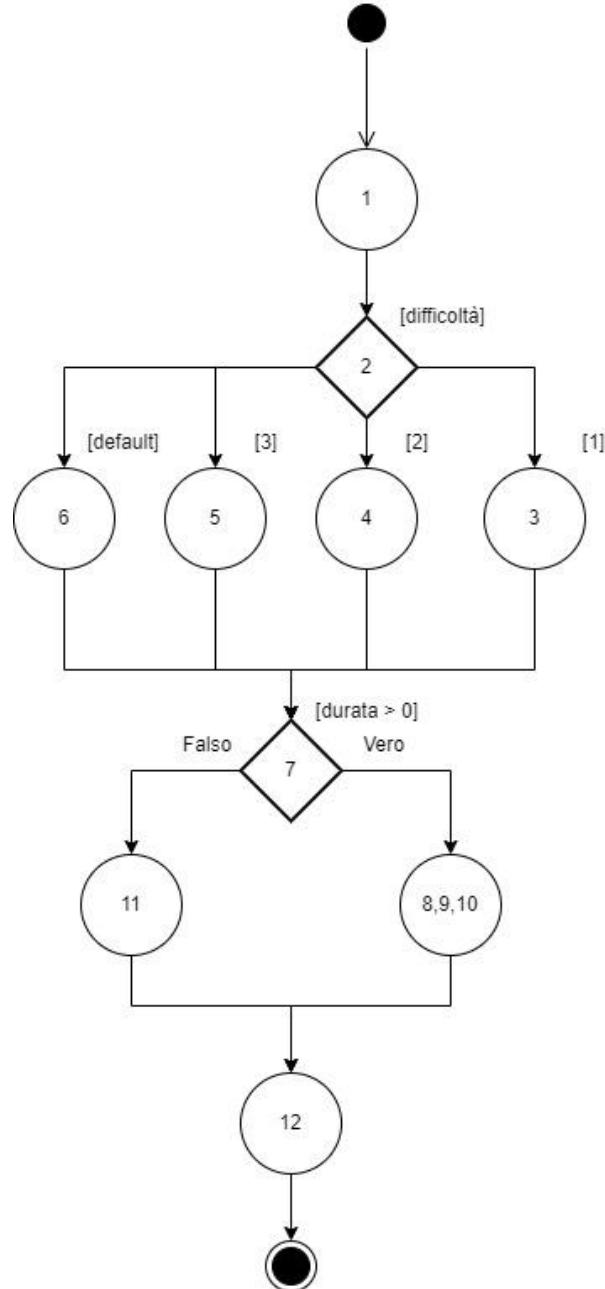
        switch (difficolta){//2
            case 1:
                diff = "Facile";//3
                break;
            case 2:
                diff = "Medio";//4
                break;
            case 3:
                diff = "Difficile";//5
                break;
            default: diff = null;//6
        }

        if(durata > 0){//7
            int hour = durata/60;//8
            int minute = durata - (hour*60);//9
            format = String.format("%02d:%02d",hour,minute);//10
        }else format = null;//11

        return new ArrayList<String>(Arrays.asList(diff,format));//12
    }
}
```

Abbiamo creato una classe Stub poiché il metodo originale andava ad applicare valori sulle UI del sistema. Non potendo testare queste ultime, andiamo a salvarci i dati in un ArrayList in modo da poterli confrontare manualmente.

In seguito, mostriamo il grafico CFG:



Dal seguente grafico sono emersi questi test:

```
package com.INGSW.NaTour.TestDataSentieroToUI;

import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import java.util.ArrayList;
import java.util.Arrays;

public class TestDataSentieroToUI {
/*
    Strategia White Box - CGF All Branch Coverage
    Difficoltà:
        1) Giusto (1...3)
        2) Sbagliato (<= 0 and >=4)
    Durata:
        1) Giusto (>0)
        2) Sbagliato (<=0)
*/
    SentieroAdapterStub sentieroAdapterStub;

    @Before
    public void setUp() {
        sentieroAdapterStub = new SentieroAdapterStub();
    }

    @After
    public void tearDown() {
        sentieroAdapterStub = null;
    }

    @Test
    public void testGiustoFacileGiusto() { //1,2,3,7,8,9,10,12
        ArrayList<String> array = new ArrayList<String>(Arrays.asList("Facile","02:20"));
        assertEquals(array, sentieroAdapterStub.convertDataSentieroToUI(1,140));
    }

    @Test
    public void testGiustoMedioGiusto() { //1,2,4,7,8,9,10,12
        ArrayList<String> array = new ArrayList<String>(Arrays.asList("Medio","02:20"));
        assertEquals(array, sentieroAdapterStub.convertDataSentieroToUI(2,140));
    }
}
```

```

@Test
public void testGiustoFacileGiusto() { //1,2,5,7,8,9,10,12
    ArrayList<String> array = new ArrayList<String>(Arrays.asList("Difficile", "02:20"));
    assertEquals(array, sentieroAdapterStub.convertDataSentieroToUI(3,140));
}

@Test
public void testGiustoSbagliato() { //1,2,3,7,11,12
    ArrayList<String> array = new ArrayList<String>(Arrays.asList("Facile", null));
    assertEquals(array, sentieroAdapterStub.convertDataSentieroToUI(1,-10));
}

@Test
public void testSbagliatoGiusto() { //1,2,6,7,8,9,10,12
    ArrayList<String> array = new ArrayList<String>(Arrays.asList(null, "02:20"));
    assertEquals(array, sentieroAdapterStub.convertDataSentieroToUI(-1,140));
}

```

Copertura dei test:

- testGiustoFacileGiusto() copre 1,2,3,7,8,9,10,12
- testGiustoMedioGiusto() copre 1,2,4,7,8,9,10,12
- testGiustoDifficileGiusto() copre 1,2,5,7,8,9,10,12
- testSbagliatoGiusto() copre 1,2,6,7,8,9,10,12

6.4 Valutazione usabilità sul campo

6.4.1 Valutazione Euristica

Per l'usabilità su campo, cioè test di usabilità a prodotto semi-finito, sono state utilizzate tecniche simili a quelle citate nel punto [2.9](#). In tal senso abbiamo quindi deciso, per le valutazioni euristiche dell'applicativo, di far testare la demo del prodotto a un piccolissimo campione di persone che spaziano per conoscenza del dominio.

La scelta di valutatori, fatta in precedenza ci è venuta di nuovo d'aiuto per questo test finale.

Alta conoscenza tecnologica	Bianca Silvestri	Giovanni Orsini
Bassa conoscenza tecnologica	Nedda Toninelli	Susanna Vespa-Tarchetti
Scelta valutatori	Bassa conoscenza di dominio	Alta conoscenza di dominio

I compiti assegnati a questi valutatori sono i seguenti:

- Registrarsi alla piattaforma con e-mail e password
- Registrarsi alla piattaforma con Google
- Inserire un percorso tramite Mappa
- Ricerca di percorsi filtrati

TESTS	Registrarsi con E-mail	Registrarsi con Google	Inserire percorso con Mappa	Ricerca e filtraggio di percorsi
Bianca Silvestri	Passed	Passed	Passed	Passed
Giovanna Orsini	Passed	Passed	Semi-Passed	Passed
Nedda Toninelli	Failed	Passed	Failed	Passed
Susanna Vespa-Tarchetti	Passed	Passed	Semi-Passed	Passed

Facendo le somme di questi test, andiamo ad analizzare che la criticità, nonché la funzionalità più manuale del sistema, è l'inserimento del percorso tramite Mappa.

Questa difficoltà è causata da un servizio non molto intuitivo, cioè Open Street Maps. Inoltre, nel mockup per ovvie ragioni la mappa era simulata e quindi era una semplificazione rispetto a una vera mappa, questa ha comportato una valutazione dell'usabilità diversa rispetto al prodotto finito.

Per ovviare a questo problema abbiamo inserito dei bottoni di eliminazione di tutti i punti del percorso e degli alert a schermo per comunicare gli errori commessi dagli utenti.

Il tasso di successo è alto per queste funzionalità e in generale, essendo l'app molto accessibile a tutti date le schermate semplici, intuitive e leggere, il campione di persone è riuscito a navigare con facilità e scioltezza tra i menù.

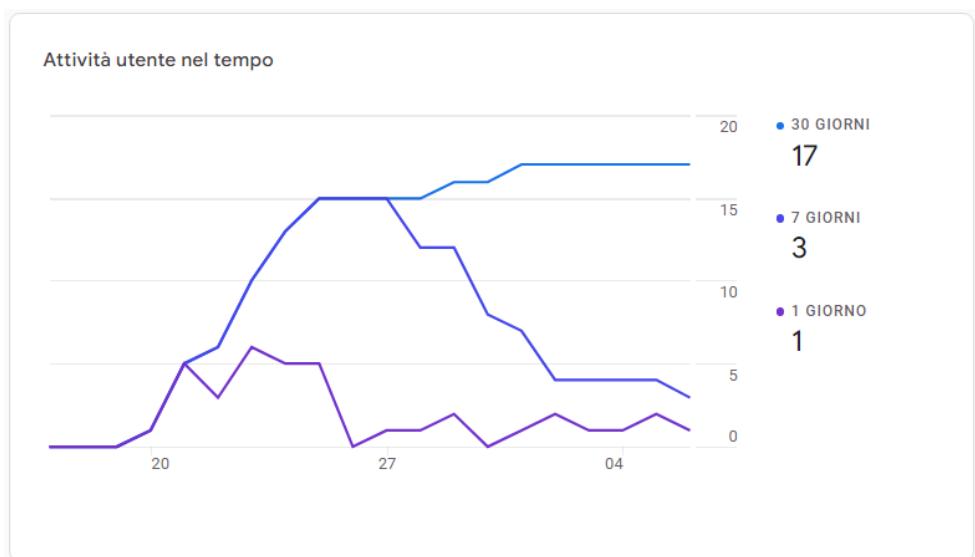
6.4.2 Analisi di Logging

Per l'analisi e monitoraggio dell'applicazione appena finita abbiamo optato sull'uso di [FirebaseAnalytics](#), una piattaforma serverless, creata da Google, poiché fornisce una suite di strumenti per l'analisi e il logging della nostra applicazione.

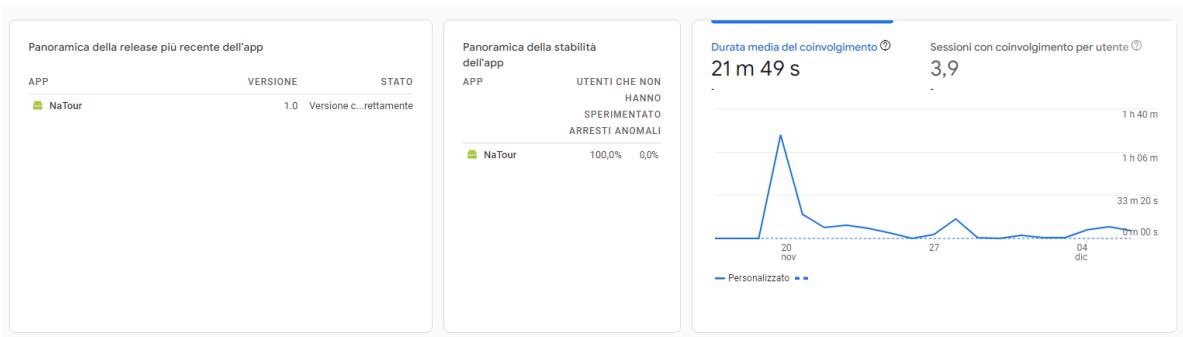
Per l'analisi e debugging di codice all'interno dell'applicazione Android abbiamo utilizzato la **libreria Log**, che ci ha permesso di facilitarci il rintracciamento di errori o arresti anomali.

Inoltre, ovviamente abbiamo utilizzato il **debugging mode** presente all'interno dell'IDE JetBrains.

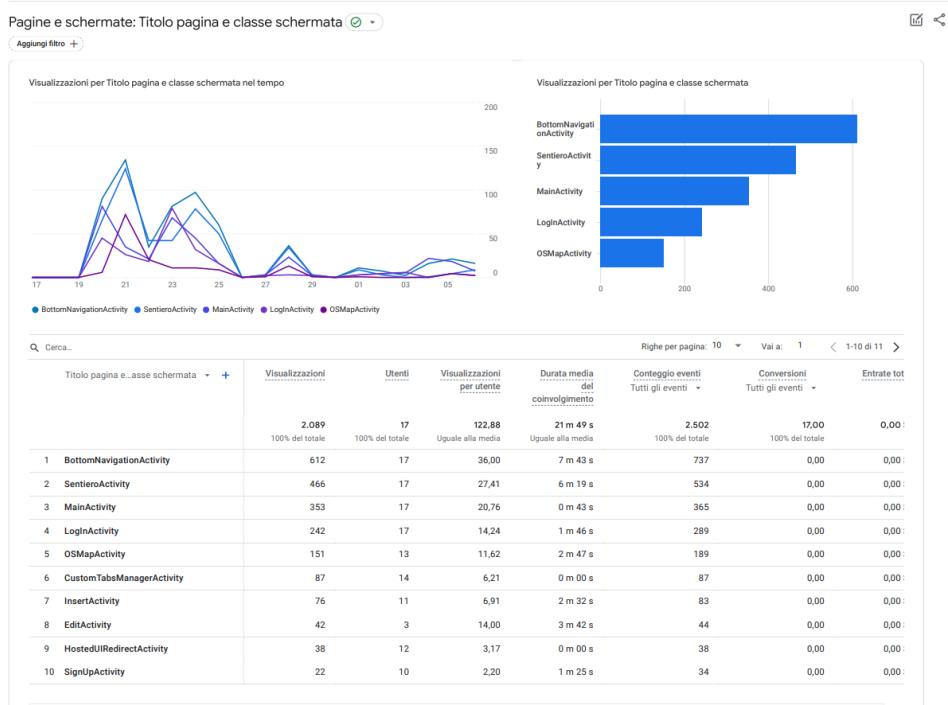
Firebase ci offre una gamma di statistiche standard per l'applicazione. Iniziando dal grafico dell'attività dell'app, che ci calcola quanti utenti hanno usato l'app dall'inizio del periodo di testing.



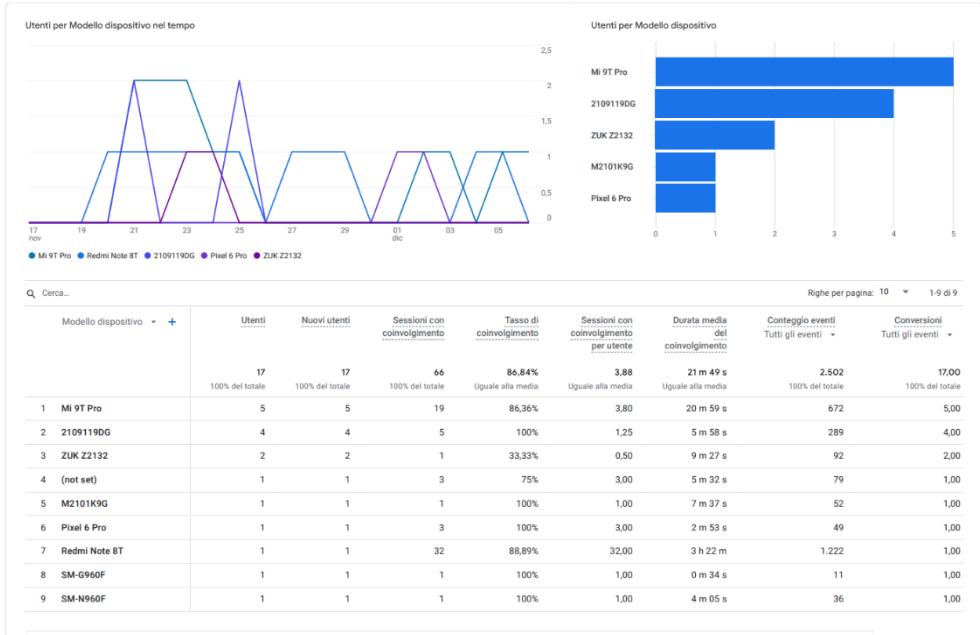
Di seguito, ancora, abbiamo statistiche d'utilizzo dell'app, come quante versioni sono state create, quanto coinvolgimento e quanti minuti sono stati utilizzati all'interno dell'applicazione in media.



Di seguito riportiamo quante e quali sono state le pagine più visitate:



E infine il numero e il tipo di modelli che hanno scaricato e testato l'applicazione.



Conclusione

L'applicazione è stata creata da:

Francesco Ciccarelli N86003285

Alex Ciacciarella N86003179

L'applicazione vuole promuoversi come semplice e innovativa nel suo genere, cercando di far colloquiare una parte social con una parte puramente informativa.



NATOUR