UNIVERSITA’ DEGLI STUDI DI NAPOLI

***FEDERICO II***

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL’INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO INGEGNERIA DEL SOFTWARE ANNO ACCADEMICO 2021/2022

**Progettazione e sviluppo di NaTour21**

**Una moderna piattaforma social per appassionati di escursioni**

**Gruppo INGSW2122\_N\_03**

Autori: Docenti:

Pierluigi Supino: N86002833 Prof. Sergio di Martino

*Pio Francesco Falzarano N86002978 Prof. Francesco Cotugno*

Sommario

1. **Document History**3
2. **Introduzione alla Documentazione**5

**PARTE I: Requisiti Software**

1. **Analisi dei Requisiti**

3.1 Requisiti Funzionali

3.2 Casi d’Uso

3.3 Tabelle di Cockburn

3.4 Prototipazione Visuale

* 1. Prototipazione Funzionale

**4. Analisi del Contesto**

4.1 Analisi del Target degli Utenti

4.2 Valutazione dell’Usabilità a Priori

**5. Modelli di Dominio**

5.1 Oggetti e Relazioni del Dominio

5.2 Class Diagrams di Analisi

5.3 Diagrammi di Sequenza di Analisi

5.4 Diagrammi di Attività

**PARTE II: Design del Sistema**

**6. Architettura del Sistema**

6.1 Analisi dell’Architettura Esterna

6.2 Architettura Backend

6.3 Servizi Cloud utilizzati

6.4 Architettura Frontend

**7. Design del Sistema**

7.1 Class Diagrams di Design

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Autore | Descrizione |
| 27/10/2021 | 0.1 | P. Falzarano  P. Supino | Creazione del format di base del documento |
| 27/10/2021 | 0.1 | P. Supino | Aggiunta: Requisiti Funzionali |
| 28/10/2021 | 0.1 | P. Falzarano | Aggiunta: Introduzione alla Documentazione |
| 29/10/2021 | 0.1 | P. Falzarano  P. Supino | Aggiunta: Use Case e Tabelle di Cockburn |
| 4/11/2021 | 0.1 | P. Supino | Aggiunta: Prototipazione Visuale |
| 12//11/2021 | 0.1 | P. Falzarano  P. Supino | Aggiunta: Prototipazione Funzionale |
| 22/11/2021 | 0.1 | P. Falzarano | Aggiunta: Analisi del Contesto |
| 25/11/2021 | 0.2 | P. Falzarano | Revisione: Format generale Paragrafi |
| 29/11/2021 | 0.2 | P. Falzarano | Aggiunta: Oggetti e Relazioni del Dominio e Class Diagrams di analisi |
| 06/12/2021 | 0.2 | P. Falzarano | Aggiunta: Sequence Diagrams di Analisi |
| 07/12/2021 | 0.2 | P. Falzarano  P. Supino | Aggiunta: Activity Diagrams |
| 13/12/2021 | 0.2 | P. Falzarano | Modifiche: Class e Sequence Diagrams di Analisi |
| 11/01/2022 | 0.2 | P. Falzarano | Aggiunta: Nuova Sezione (II) e Analisi dell’architettura Esterna |
| 17/01/2022 | 0.2 | P. Falzarano | Aggiunta: Architettura Backend e servizi cloud utilizzati |
| 24/01/2022 | 0.2 | P. Falzarano | Aggiunta: Architettura Frontend |

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Autore | Descrizione |
| 31/01/2022 | 0.2 | P. Falzarano  P. Supino | Aggiunta: Class Diagrams di Design |

# Introduzione alla Documentazione

La seguente documentazione segue il processo di sviluppo del Software richiesto dal Committente; Ogni fase dello sviluppo è descritta in una sezione, in totale sono tre:

1. La prima si focalizza sui Requisiti Software, avendo in considerazione le specifiche fornite dal committente e soprattutto dalle esigenze degli utenti finali; vengono descritti tutte le funzionalità offerte dalla piattaforma, dando esempi di prototipazione per mostrare l’interazione col sistema;

infine, è dedicata uno spazio all’ OOA del dominio, attraverso diagrammi di classi, di sequenza e di attività.

1. La seconda espone l’architettura del Sistema, le tecnologie utilizzate per il back-end e l’analisi del design pattern architetturale utilizzato per il front-end, accompagnati da artefatti UML di OOD; una parte è incentrata sulla definizione delle gerarchie funzionali dell’applicativo.
2. L’ultima incentrata sul Testing del Sistema, del codice mediante Unit-Testing effettuato con JUnit e dell’interfaccia utente mediante Beta Testing.

# **PARTE I**: Requisiti Software

# Analisi dei Requisiti

Il Committente richiede lo sviluppo di un’applicazione su client mobile volta alla commercializzazione del social “NaTour” tra gli amanti delle escursioni. Il fulcro principale della piattaforma social è la condivisione di Itinerari: un utente inserisce un itinerario fornendone una descrizione, gli altri possono visualizzare l’itinerario ed i suoi dettagli. L’esperienza social scaturisce dall’interazione tra utenti, attraverso lo scambio di messaggi, condivisione di foto relative ad un percorso e miglioramento della descrizione di un percorso attraverso riscontri. Ecco le funzionalità in dettaglio:

# 3.1 Requisiti Funzionali

* Il sistema deve permettere il login all’utente, in particolare:
  + L’utente deve avere la possibilità di accedere utilizzando indirizzo e-mail e password (in caso di avvenuta registrazione con l’indirizzo e-mail corrispondente);
  + L’utente deve poter resettare la propria password;
  + L’utente deve avere anche la possibilità di effettuare il login utilizzando providers esterni (Google);
* Il sistema deve permettere all’utente di registrarsi utilizzando il proprio indirizzo e-mail; inoltre, durante la registrazione, l’utente deve inserire dei campi obbligatori quali nome e password;
* Il sistema deve permettere all’utente di condividere nuovi itinerari caratterizzati da un nome, una durata, un livello di difficoltà, un punto di inizio, una descrizione (opzionale), un insieme di fotografie e un tracciato geografico che lo rappresenta su una mappa, in particolare l’utente può:
  + Decidere di inserire un nuovo itinerario manualmente avvalendosi di una mappa interattiva;
  + Decidere di inserire un nuovo itinerario utilizzando un file in formato GPX;
* Il sistema deve permettere all’utente di visualizzare una lista di itinerari e una pagina di riepilogo di tutte le informazioni riguardanti ogni itinerario, incluse eventuali fotografie caricate da altri utenti;
* L’utente deve aver la possibilità di visualizzare su una mappa il percorso geografico inerente ad un itinerario;
* Il sistema deve permettere all’utente di dare un feedback alla descrizione di un itinerario fornendo un punteggio di difficoltà e/o tempo di percorrenza diverso da quelli indicati nella pagina di riepilogo dell’itinerario; in questo caso il punteggio di difficoltà e il tempo di percorrenza per il sentiero saranno ricalcolati come media delle difficoltà e dei tempi indicati.
* Il sistema deve permettere all’utente di inviare/ rispondere a messaggi privati verso altri utenti.
* Il sistema deve permettere all’utente di visualizzare la lista dei messaggi ricevuti da altri utenti.
* Il sistema deve permettere all’utente di caricare fotografie di punti di interesse scattati sul sentiero di un itinerario; queste saranno visualizzate nella pagina di dettaglio dell’itinerario. Inoltre, se la fotografia ha una posizione geografica di scatto salvata nei metadati, sarà mostrato all’interno della mappa un marker con la fotografia corrispondente.
* Il sistema deve permettere agli Amministratori di rimuovere o modificare itinerari inseriti dagli utenti, ed in caso di modifica verrà mostrato un warning nella schermata di dettaglio dell’itinerario con relativa data di modifica.
* Il sistema deve permettere all’utente di effettuare il logout da un dispositivo (in questo caso dovrà necessariamente effettuare di nuovo il login per utilizzare le funzionalità dell’applicazione).

# 3.2 Casi d’Uso

Di seguito sono riportati i requisiti funzionali introdotti precedentemente attraverso rappresentazione grafica con l’utilizzo di Use Case Diagrams; ogni diagramma è relativo a macro-funzionalità del Sistema:

* Gestione del Profilo Utente;
* Condivisione di Itinerari;
* Messagistica tra Utenti;
* Funzionalità dell'Amministratore;

Gli attori secondari che riguardano web services sono stati omessi per brevità, visto che nella parte II del documento è dedicato un paragrafo ai servizi cloud utilizzati.

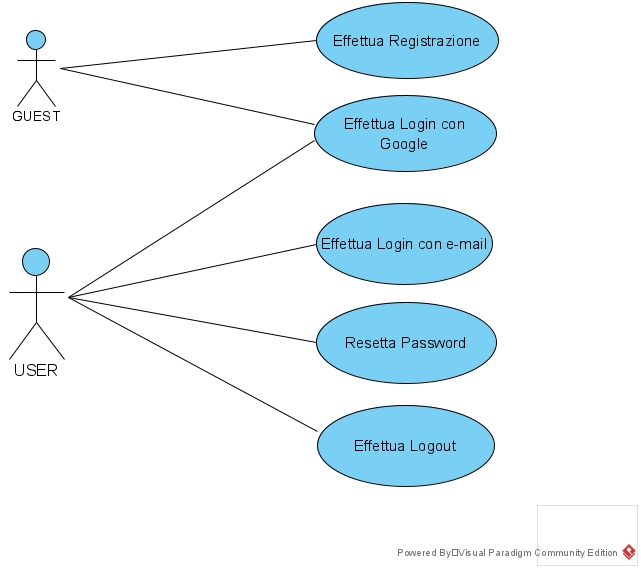


Figura 1: Use Case Diagram relativo alla gestione del Profilo Utente

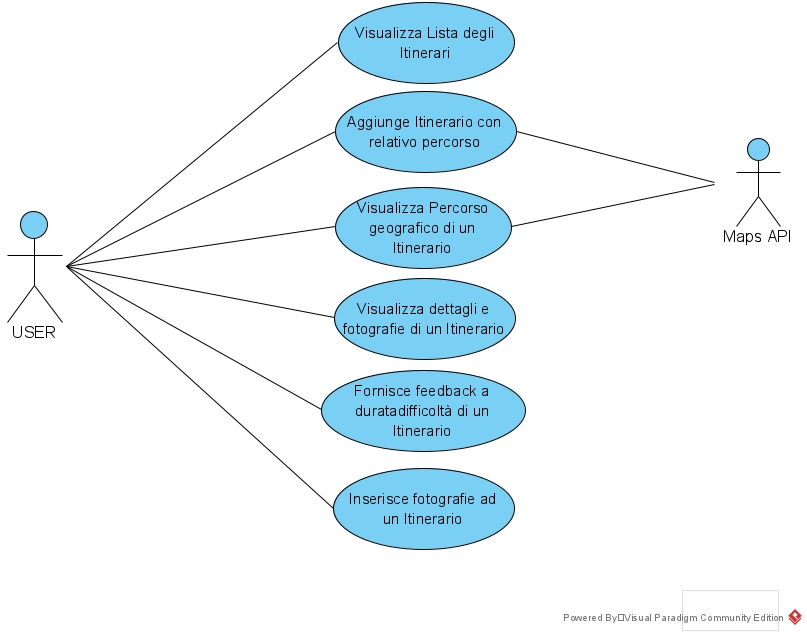


Figura 2: Use Case Diagram relativo alla condivisione di Itinerari

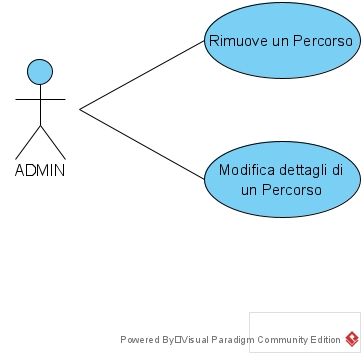


Figura 3: Use Case Diagram relativo all’Amministratore (include anche i casi d’uso del normale utente)

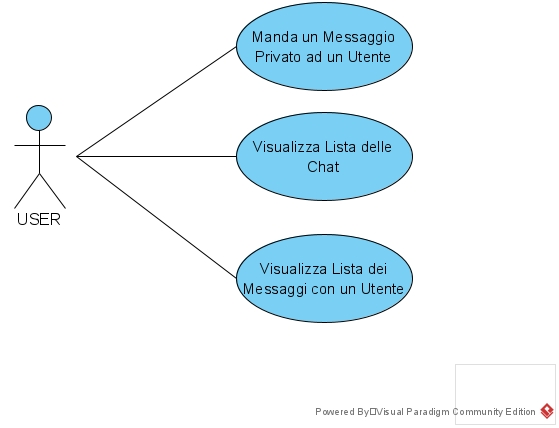


Figura 4: Use Case Diagram relativo alla Messaggistica

## 3.3 Tabelle di Cockburn

In questa sezione vengono analizzati due casi d’uso: il primo “Aggiunge Percorso” (Use Case #1) cardine della piattaforma, e il secondo “Fornisce feedback alla descrizione di un Percorso” (Use Case #2) per migliorare l’accuratezza dei dettagli dei percorsi secondo le esperienze personali degli utenti.

Al fine di garantire una coerente rappresentazione e una facile interpretazione si è fatto uso del formalismo delle Tabelle di Cockburn, riportate in seguito, con allegati mock-up grafici dell’interfaccia utente e statecharts nelle sezioni successive.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USE CASE *#1* | Aggiunge Percorso | | | | |
| Goal in Context | Permettere all’utente di inserire un percorso e condividerlo sulla piattaforma; | | | | |
| Preconditions | L’utente deve aver effettuato il login; | | | | |
| Success End Condition | L’utente è riuscito ad inserire il percorso nella piattaforma; | | | | |
| Failed End Condition | Il Percorso non è stato inserito nella piattaforma; | | | | |
| Primary Actor | Utente | | | | |
| DESCRIPTION | *Step* | **Utente** | | **Sistema** | |
| *1* | L’utente clicca sul pulsante “+”  nella schermata Main-Page | |  | |
| *2* |  | | *Il sistema mostra la schermata*  *Add-Itinerary* | |
| 3 | L’utente inserisce un nome per l’itinerario | |  | |
| 4 (\*) | L’utente inserisce una descrizione per l’itinerario | |  | |
| 5 | L’utente inserisce una difficoltà per l’itinerario | |  | |
| *6* | L’utente inserisce una durata per l’itinerario | |  | |
| *7* | L’utente clicca sul pulsante “*Select Photos*” | |  | |
| *8* |  | | *Il sistema mostra il File Explorer del dispositivo* | |
| *9* | L’utente seleziona le foto da allegare all’itinerario e conferma | |  | |
| *10* |  | | *Il sistema chiude il File Explorer e associa le foto al percorso* | |
| *11* | L’utente clicca sul pulsante “*Select GPX file*” | |  | |
|  | *12* |  | | *Il sistema mostra il File Explorer* | |
| *13* | L’utente seleziona il file GPX da allegare all’itinerario e conferma | |  | |
| *14* |  | | *Il sistema chiude il File Explorer ed effettua il parsing del file a JSON* | |
| *15* | L’utente clicca sul pulsante “*Add*” | |  | |
| *16* |  | | *Il sistema aggiunge l’itinerario nella piattaforma e ne mostra la schermata Itinerary-Detail* | |
| EXTENSION #1  Errore Generico | *Step* | **Utente** | | **Sistema** | |
|  | *1* |  | | *Il sistema mostra un messaggio di errore non riuscendo a caricare l’itinerario* | |
|  | *2* | L’utente riprova a salvare l’itinerario (ritorno al punto 15 del normal flow) | |  | |
| EXTENSION #2  File non supportati | *Step* | **Sistema** | | | |
|  | *1* | *Il sistema mostra un simbolo di warning e un messaggio di errore accanto ai file caricati* | | | |
| EXTENSIO #3  Le fotografie hanno una posizione di scatto distante dal tracciato geografico | *Step* | **Sistema** | | | |
|  | *1* | *Il sistema mostra un messaggio di errore quando l’utente cerca di caricare l’itinerario, invitandolo ad eliminare le fotografie non attinenti al percorso* | | | |
| SUBVARIATIONS | *Step* | **Utente** | **Maps Api** | | **Sistema** |
|  | *11.a* | L’utente clicca su “*Open Maps*” |  | |  |
| *12.a* |  | Recupera Posizione corrente | |  |
| *12.a* |  |  | | *Il sistema apre Maps-Viewer* |
| *13.a* | L’utente crea un percorso sulla mappa e lo conferma |  | |  |
| *14.a* |  | *Salva il percorso su un file JSON* | |  |
| *15.a* |  |  | | *Il sistema associa il file JSON*  *Al percorso e (ritorno al punto 15 del normal flow)* |

(\*) Opzionale

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE *#2* | Fornisce feedback alla descrizione di un Percorso | | |
| Goal in Context | Permettere all’utente di indicare un diverso punteggio di difficoltà e/o tempo di percorrenza di un itinerario; | | |
| Preconditions | L’utente deve aver effettuato il login; | | |
| Success End Condition | L’utente è riuscito a fornire un riscontro sui dettagli di un itinerario; | | |
| Failed End Condition | L’utente non è riuscito a dare un feedback; | | |
| Primary Actor | Utente | | |
| DESCRIPTION | *Step* | **Utente** | **Sistema** |
| *1* | L’utente clicca su un Itinerario nella schermata Main-Page |  |
| *2* |  | *Il sistema mostra la schermata*  *Itinerary-Detail dell’itinerario selezionato* |
| 3 | L’utente clicca su “*Send a feedback*” |  |
| 4 |  | *Il sistema mostra la schermata*  *Edit-Detail* |
| 5 | L’utente inserisce una durata e/o livello di difficoltà diverso per l’itinerario |  |
| *6* | L’utente clicca sul pulsante “*Salva*” |  |
| *7* |  | *Il sistema ricalcola la difficoltà e la durata del percorso* |
| *8* |  | *Il sistema chiude Edit-Detail e mostra la schermata Itinerary-Detail con i dati aggiornati* |
| EXTENSION #1  Errore Generico | *Step* | **Utente** | **Sistema** |
| Ref. Use Case #1 | |  |  |

## 3.4 Prototipazione Visuale

Schermata 1: Main Page

Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene mappa

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamente

Figura 5: Prototipazione Visuale Use Case #1

Schermata 4: Itinerary Detail

Schermata 2: Add Itinerary

Schermata 3: Maps Viewer

## 

Schermata 1: Main Page

Schermata 3: Edit Detail

Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamente

Figura 6: Prototipazione Visuale Use Case #2

Schermata 2: Itinerary Detail

## 3.5 Prototipazione Funzionale



Figura 7: Prototipazione Funzionale Use Case #1



Figura 8: Prototipazione Funzionale Use Case #2

# 4. Analisi del Contesto

# 4.1 Analisi del Target degli Utenti

Secondo un’indagine svolta nel 2019, in Italia gli escursionisti sono in aumento, complice la scoperta di sempre più “Vie del Cammino”; Dai risultati del questionario effettuato dalla casa editrice di **Terre di Mezzo** emerge un ritratto del camminatore tipo in Italia: tra le motivazioni che spingono all’escursionismo troviamo in testa il “Fare Trekking” (52%), “Stare nella Natura” (50%) e “Scoprire il territorio” (46%).



Figura 9: Sondaggio a Risposte Multiple su un campione di escursionisti

In generale uno dei motivi principali che spinge a mettersi in cammino è la voglia di conoscere i borghi ed il territorio; ed è proprio questo uno dei cardini della piattaforma: “Scoperta”, conoscere luoghi grazie al contributo di altre persone e ricambiarle facendo conoscere, ampliando in questo modo l’esperienza social, la condivisione che porta allo sviluppo di una community attiva nella piattaforma. Altro dato interessante riguarda l’età: il 27,8% ha tra 51 e 60 anni, il 22,5% tra 61 e 70 anni, il 19,2% tra 41 e 50 anni.



Figura 10: Rilevazione dati (età) su un campione di escursionisti

Vista l’alta percentuale di camminatori over 40, l’applicazione deve essere intuitiva, facile da usare per favorire la commercializzazione tra i meno avvezzi a nuove piattaforme. Dunque, molta enfasi è stata data all’interfaccia utente, usabile ma allo stesso tempo accattivante, per attrarre tutti ed incuriosire anche chi non appartiene al mondo delle escursioni. Secondo i dati **ISTAT**, la categoria delle escursioni, rilevata esplicitamente per la prima volta nel 2015, è praticata da più di 1 milione 173 mila persone, ai quali si aggiungono chi pratica sport a stretto contatto con la natura (e.g. Orienteering, Ciclismo). In questo gruppo si può identificare l’insieme dei principali utenti finali del Software.

Fonti e sitografia: -[La pratica sportiva in Italia (istat.it)](https://www.istat.it/it/archivio/204663); -[OUT\_TREKKING (sport-press.it)](https://outdoormag.sport-press.it/wp-content/uploads/sites/3/2021/05/05_OUT_TREKKING.pdf); - [Terre di Mezzo](https://www.terre.it/cammini-percorsi/i-dati-dei-cammini/ora-i-camminatori-italiani-preferiscono-gli-itinerari-in-italia/) Editore (terre.it).

Ecco in dettaglio le tipologie principali di *personas* a cui si potrebbe riferire il prodotto e che, con maggiore probabilità, lo utilizzerebbero con più frequenza rispetto ad altri:



**Katia  
Età: 25  
Professione: Studentessa   
Hobby:**

* Sport tra cui Ciclismo e Corsa
* Andare al Cinema

**Segni particolari:**

* Ama provare cose nuove
* Divoratrice di libri

**Rapporti con la Tecnologia:**

* Ottimi

**Ore di utilizzo di Social Network (al giorno):**

* 4

**Maria  
Età: 42  
Professione: Commerciante  
Hobby:**

* Lunghe Passeggiate
* Yoga e Meditazione

**Segni particolari:**

* Introversa
* Mangia salutare
* Precisa in tutto quello che fa

**Rapporti con la Tecnologia:**

* Discreti

**Ore di utilizzo di Social Network (al giorno):**

* 2/3

**Roberto  
Età: 56  
Professione: Architetto  
Hobby:**

* Musica Classica
* Escursioni nella Natura
* Viaggi

**Segni particolari:**

* Gran chiacchierone
* Ama la compagnia

**Rapporti con la Tecnologia:**

* Buoni

**Ore di utilizzo di Social Network (al giorno):**

* 1

|  |
| --- |
|  |

# Valutazione dell’Usabilità a priori

# 5. Modelli di Dominio

# Oggetti e Relazioni del Dominio

L’entità principale del dominio è l’< **Itinerario** >, caratterizzato da un nome, una durata, una difficoltà, una descrizione e una data di creazione; ogni itinerario può avere più < **POI** > punti di interesse con una eventuale posizione geografica e una fotografia del punto. Ad ogni Itinerario inoltre corrisponde biunivocamente un < **Tracciato Geografico** >, composto da un punto iniziale, uno finale e una serie di < **Punti Intermedi** >. Si noti la scelta di distinguere tra Itinerario e Tracciato corrispondente, per manipolare al meglio la parte descrittiva e quella meramente geografica di un Percorso.

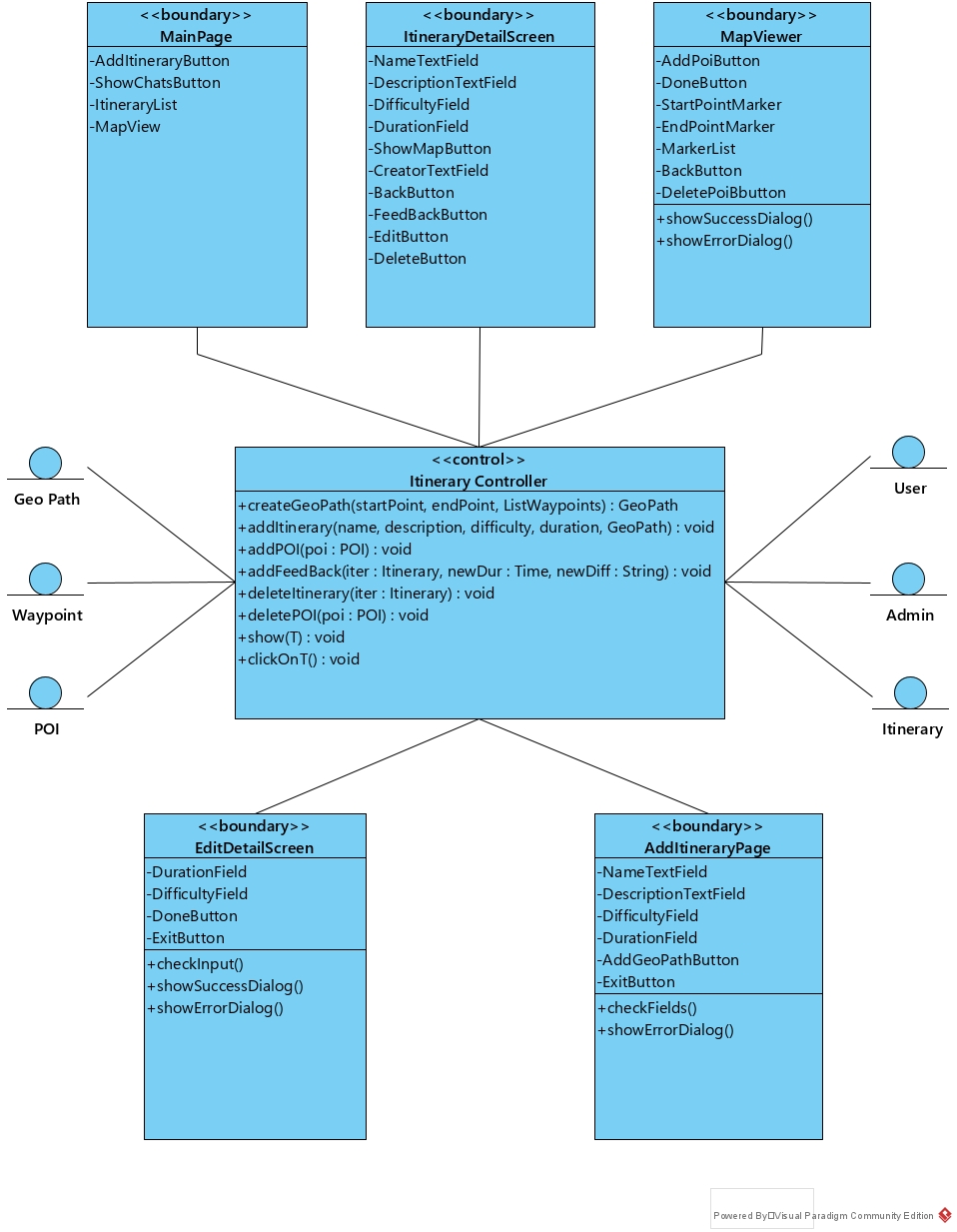
L’< **Utente** > si occupa dell’inserimento di Itinerari e può eventualmente fornire feedback per un Itinerario (per il calcolo della media di durata e/o difficoltà). Gli Utenti possono scambiarsi < **Messaggi** > in formato testuale, con data e ora di invio.

Infine, < **Admin** > è una specializzazione di Utente, che possiede dei privilegi rispetto ad un normale utilizzatore del software: rimozione e modifica di itinerari nella piattaforma.

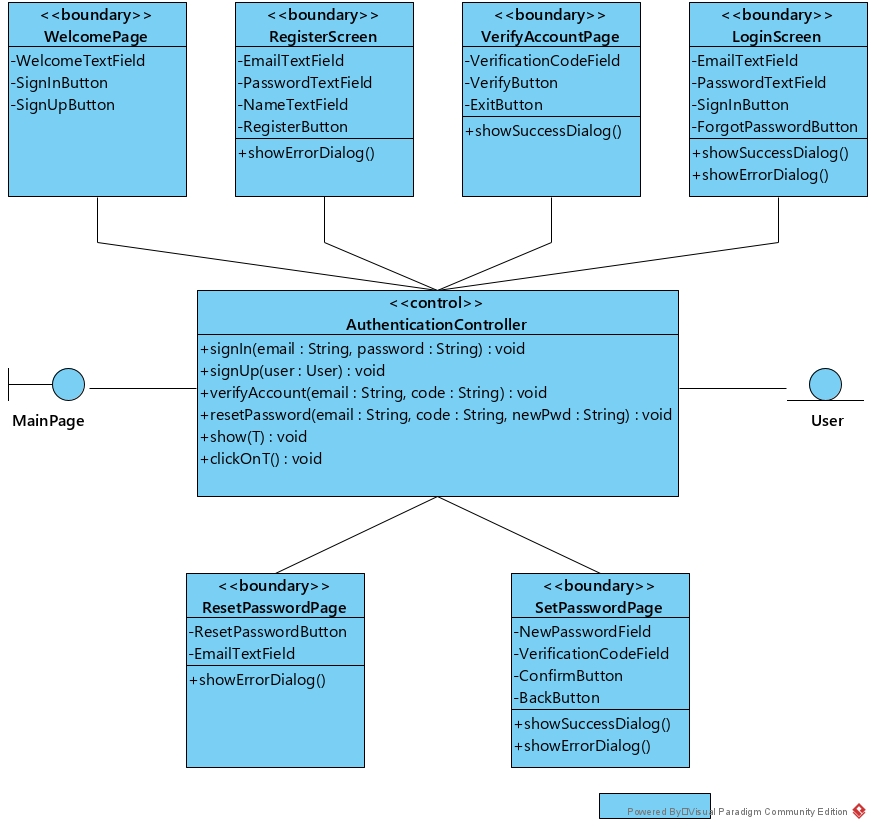
Nelle pagine successive si troveranno in formato Class Diagram UML gli oggetti di analisi, divisi secondo l’euristica three-object-type che prevede un raggruppamento in tre layer: *Entity* (i concetti del Dominio), *Boundary* (l’interfaccia al Sistema) e *Control* (la logica del Sistema); sono inoltre descritti i comportamenti delle classi di analisi attraverso Sequence Diagrams (avvalendosi anche dei Data Access Objects) e Activity Diagrams.

# Class Diagrams di Analisi

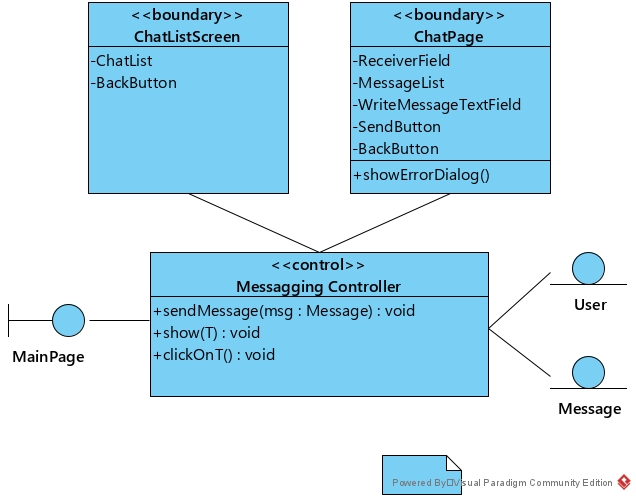
Class Diagram 1: Entity Layer (implicit getters e setters)



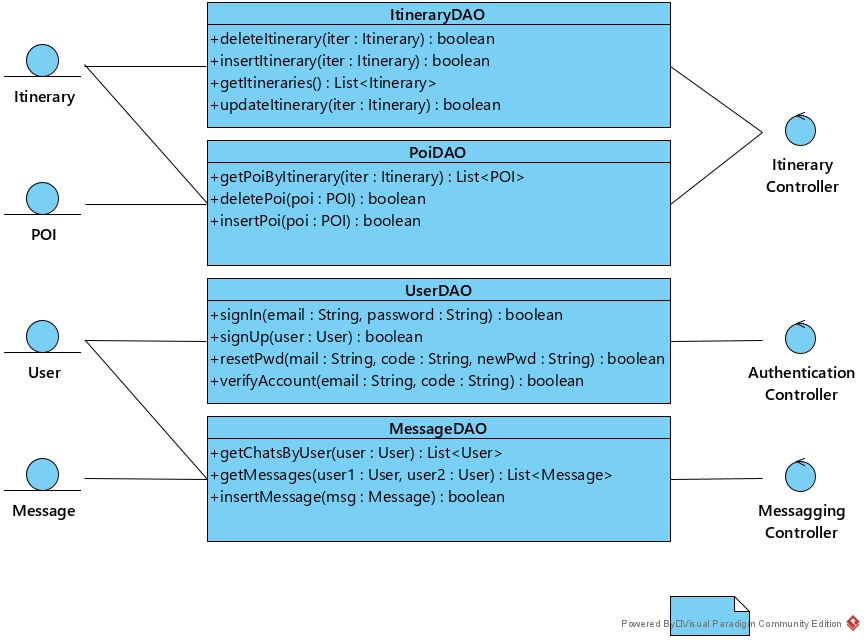
Class Diagram 2: Itinerary Management



Class Diagram 3: Authentication

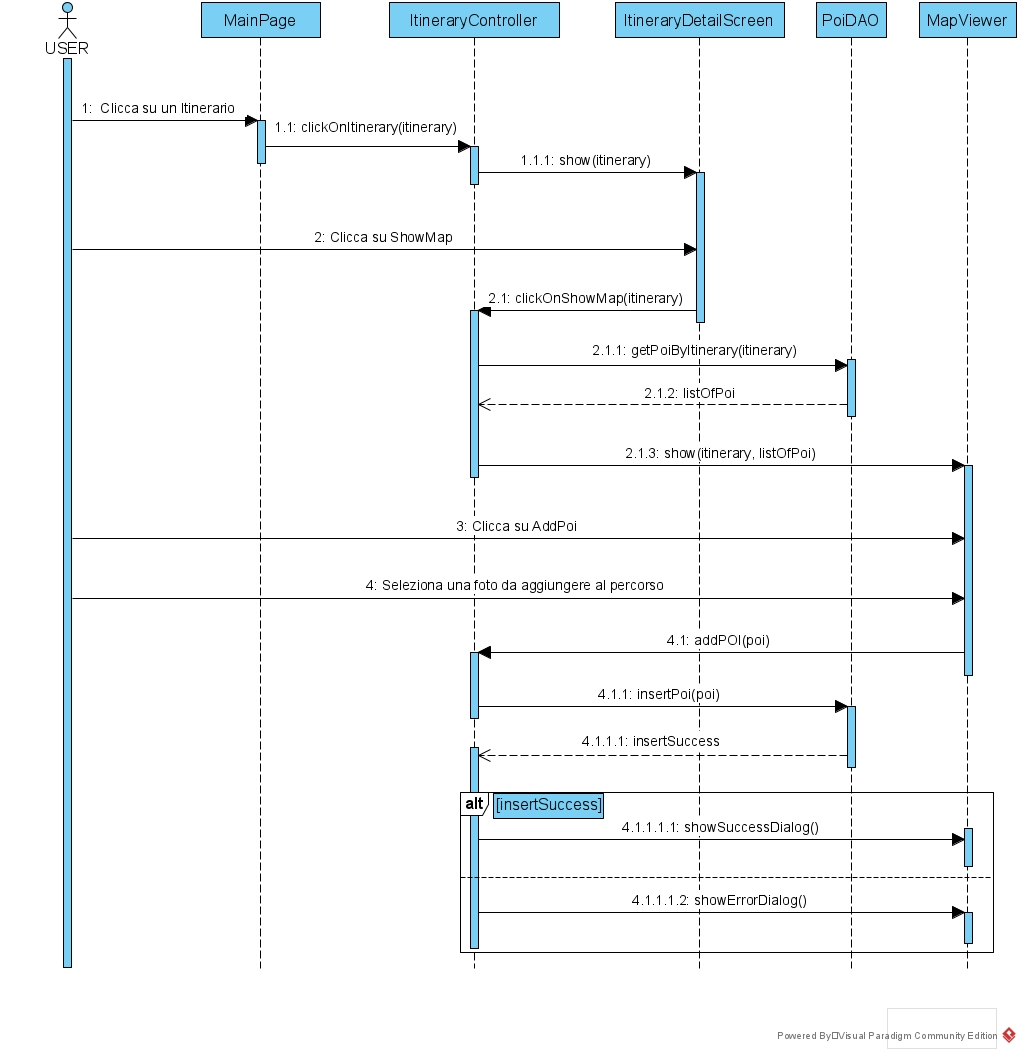


Class Diagram 4: Messaging

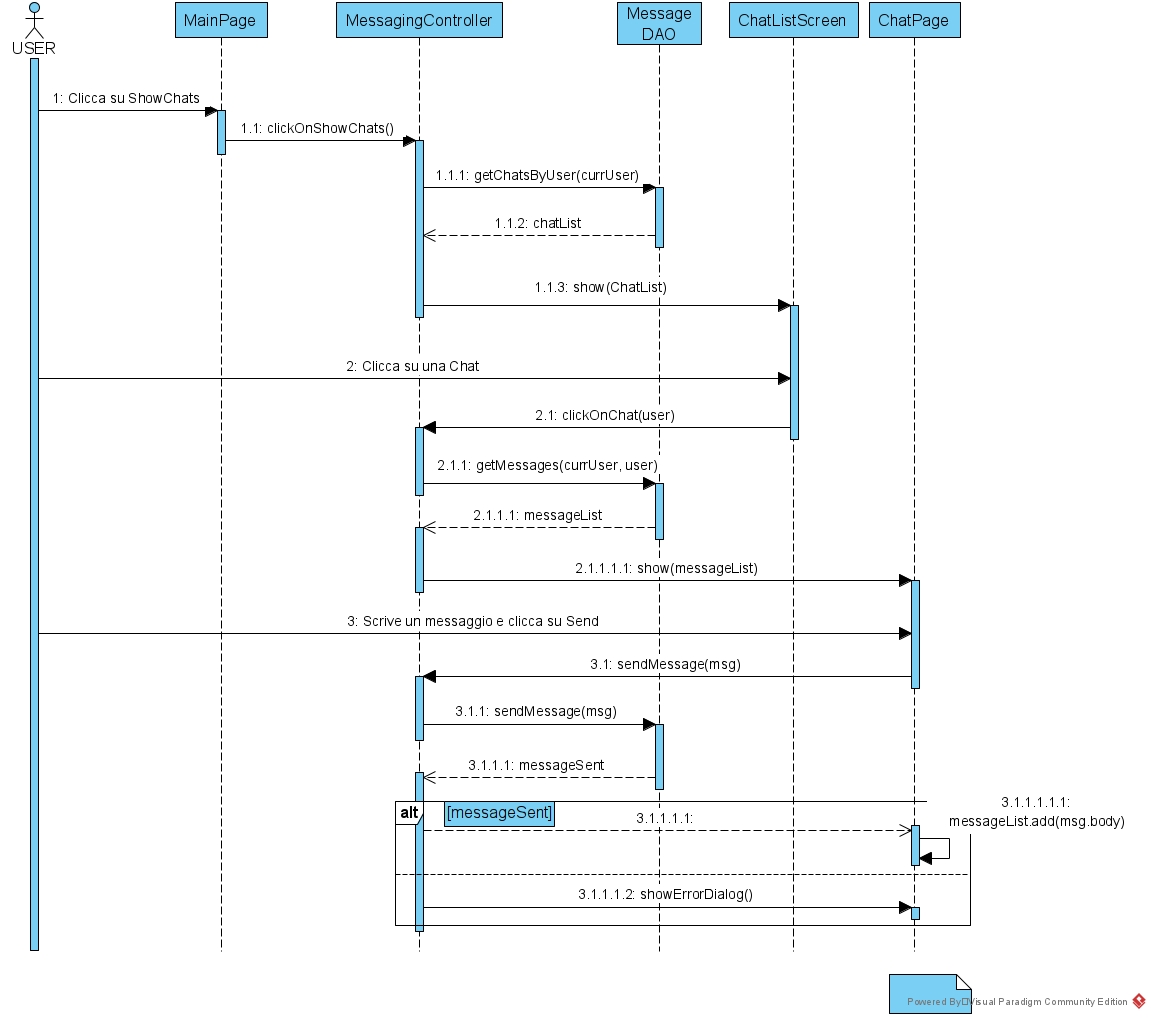


Class Diagram 5: DAO Layer

# Diagrammi di Sequenza di Analisi



Sequence Diagram 1: Add Point of Interest



Sequence Diagram 2: Reply to a message

# Diagrammi di Attività



Activity Diagram 1: Login



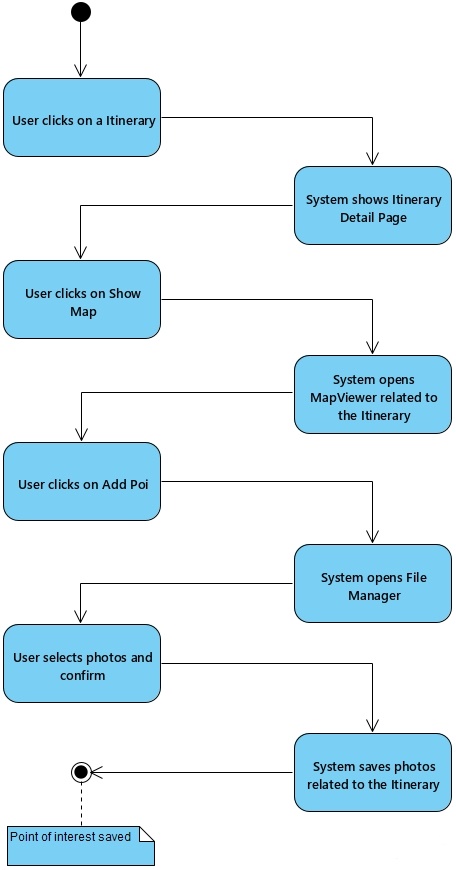
Activity Diagram 2: Registration



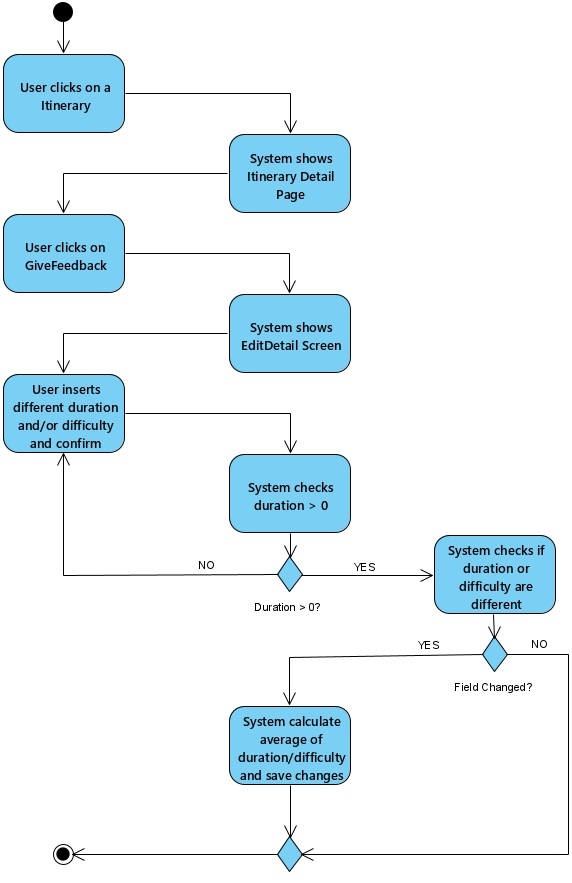
Activity Diagram 3: Reset Password



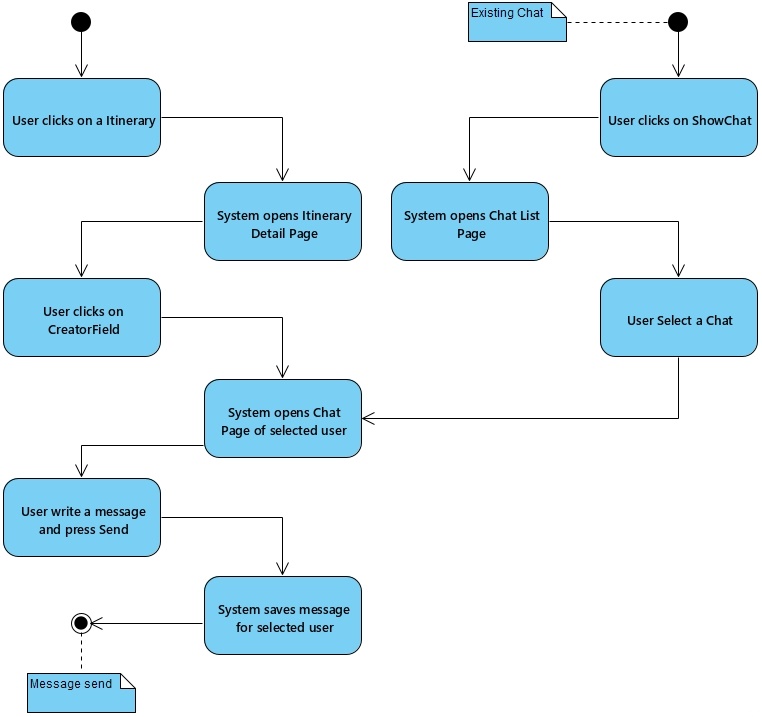
Activity Diagram 4: Add Itinerary



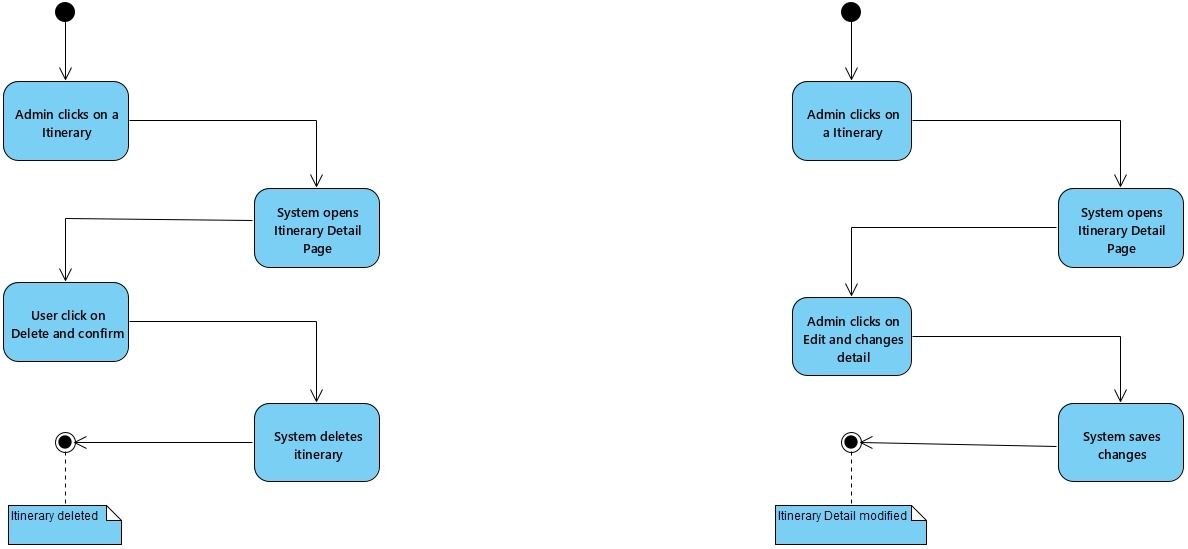
Activity Diagram 5: Add POI



Activity Diagram 6: Feedback



Activity Diagram 7: Messaging



Activity Diagram 8: Delete Itinerary

**PARTE II**: Design del Sistema

# Architettura del Sistema

# Analisi dell’Architettura Esterna

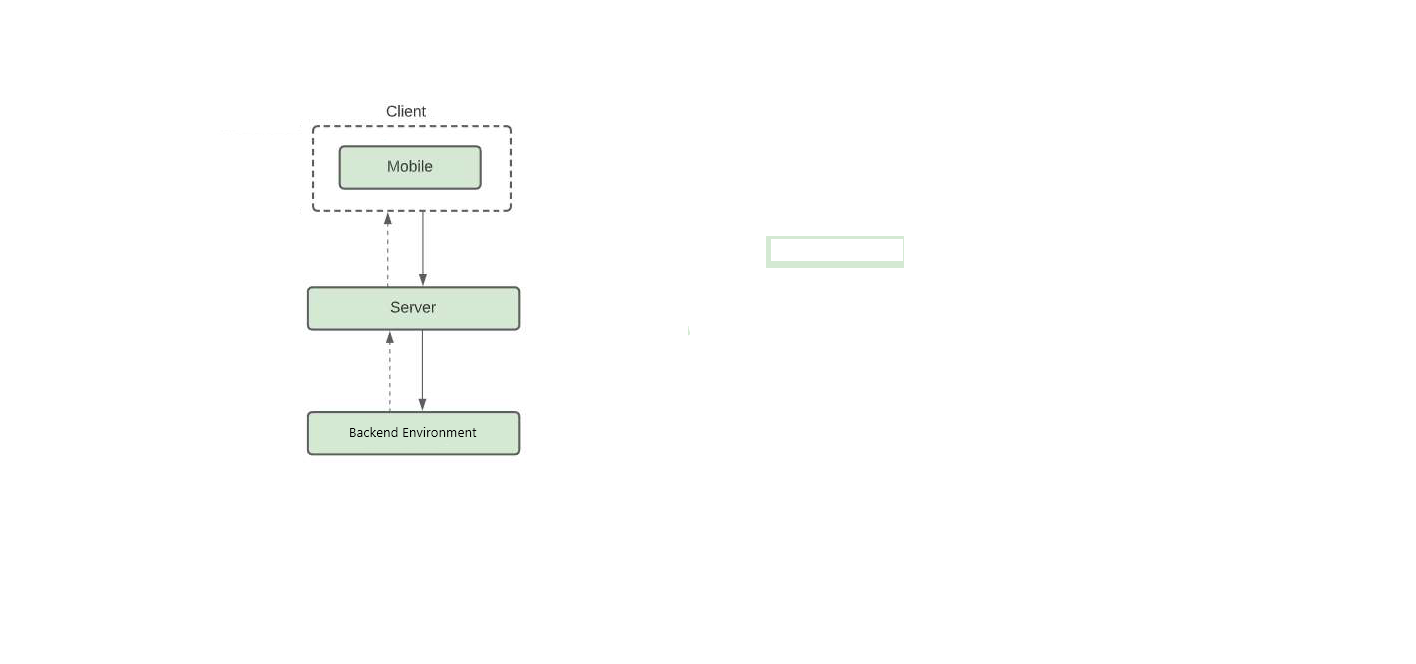
Per quando concerne l’architettura esterna del Sistema, attraverso una suddivisione in layer, in cui ogni livello comunica direttamente solo con quello sottostante, è stato possibile costruire un’architettura chiusa e sicura.

Figura 1: Three-Tier Architecture

Il client fa affidamento ai servizi offerti dal Server, come ad esempio rispondere alle richieste di una risorsa, caricare una risorsa da parte del client e così via; il server a sua volta propaga le richieste ai servizi che gestiscono i dati per poi mandare un responso al client. Dunque, nessuno può inserire, modificare o eliminare risorse a parte il server che funge da “amministratore” del Backend Environment.

# Architettura Backend

L’architettura del Backend è stata realizzata usufruendo dei servizi offerti da Amazon Web Services. Il server è stato realizzato con un approccio serverless, tale da eliminare completamente le attività di gestione di una macchina in rete, quali distribuzione di carico, manutenzione o aggiornamenti: semplicemente è solo necessario creare pacchetti di codice per il deployment in cloud, dato che in genere i servizi di cloud computing sono scalati automaticamente in base all’uso e vengono aggiornati continuamente dal service provider.

Come già affermato, il client non ha la possibilità di accedere direttamente ai servizi se non dopo l’autenticazione ed inoltre tali servizi non sono richiesti direttamente dal client, ma devono necessariamente passare per la logica serverless; ad esempio, per l’upload di una foto, il client invia la richiesta di upload al server, il quale risponde con un URI (fornito dal bucket) adempito per l’operazione. I servizi sono offerti attraverso delle API proprietarie, che offrono un’interfaccia sicura e adattabile per l’architettura.

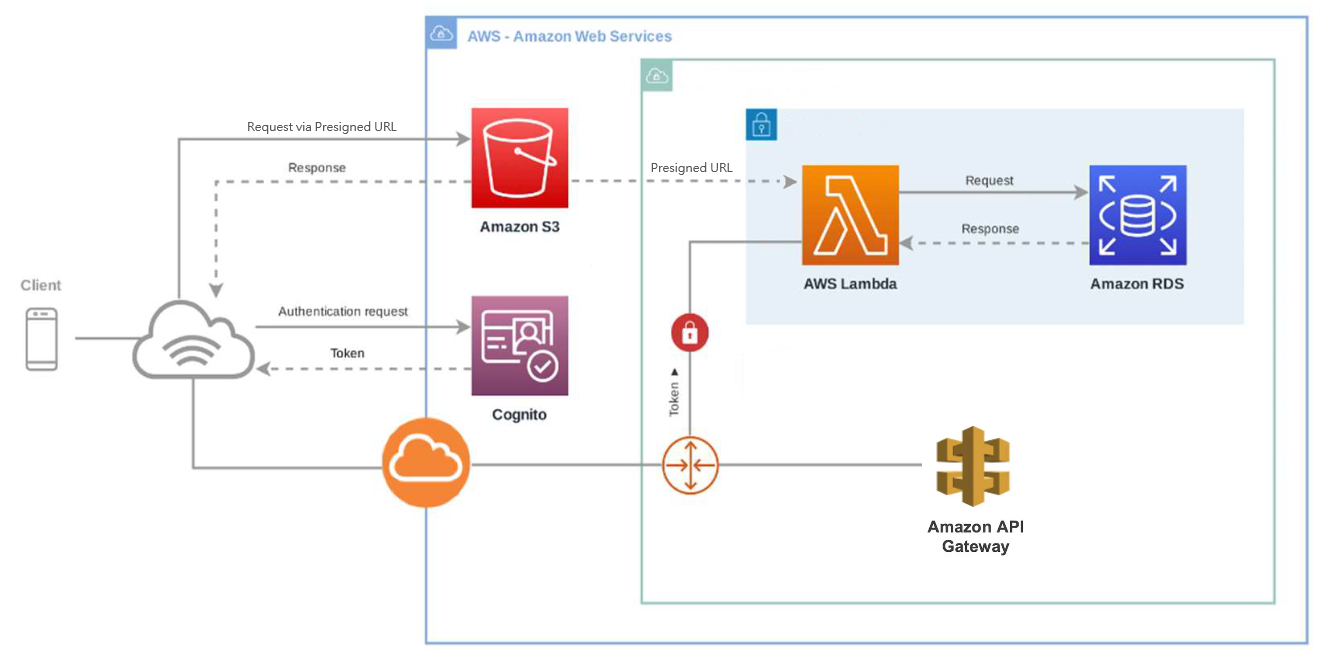


Figura 2: Backend in cloud

Le API realizzate sono di due tipi:

1. **RESTful** (REpresentational State Transfer) **API**
2. **Web Socket API**
3. Le prime (***RESTful API***) definiscono il protocollo di comunicazione tra il client, che richiede una risorsa o un servizio, e il server che la fornisce. Le risorse vengono scambiate tra le due entità tramite una rappresentazione in formato JSON (JavaScript Object Notation), uno tra i più diffusi nonché consigliabile se si lavora con un linguaggio di programmazione server-side basato su JavaScript. Infatti, per il server si è adottato il linguaggio Node.JS con l’ausilio del framework Express basato sul protocollo di comunicazione HTTP per gestire le richieste. La comunicazione tra client e server è stateless, in quanto nessuna delle due entità mantiene informazioni sulle precedenti interazioni, rendendo ogni richiesta distinta con le altre.
4. Le seconde (***Web Socket API***) basate su Socket TCP full-duplex (che permettono lo scambio di dati in entrata e in uscita contemporaneamente) sono risultate indispensabili al fine di fornire un servizio di chatting real-time; Quando un client apre la Socket, viene registrata la sua sessione in modo tale che altri client possano mandare direttamente un messaggio sulla Socket. Anche in questo caso, il server che gestisce le sessioni degli utenti online è di tipo serverless.

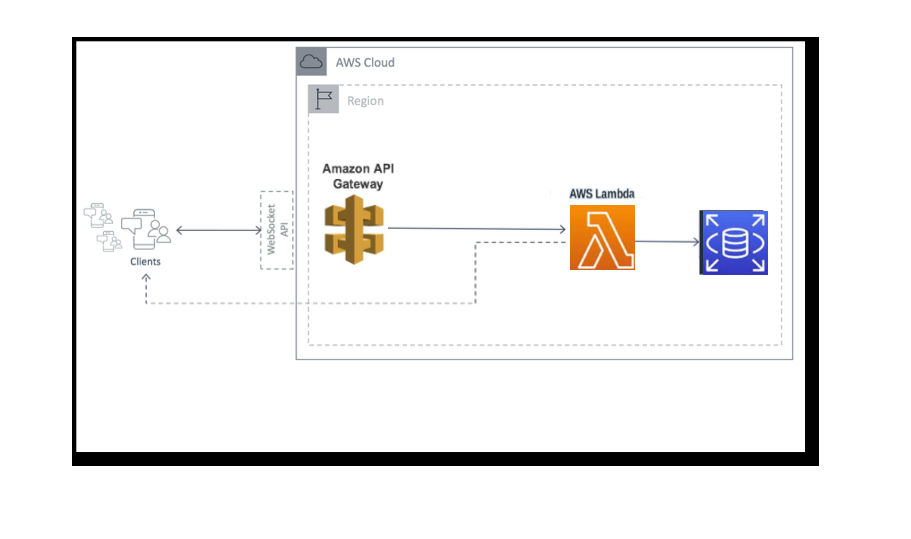


Figura 3: Web Socket API

Di seguito è riportato l’elenco delle funzionalità offerte dalla RESTful API:

|  |  |  |
| --- | --- | --- |
| METODO HTTP | PERCORSO | DESCRIZIONE |
|  | /itineraries | Permette il recupero di itinerari dal database |
| GET | /photos | Permette il recupero di presigned-url per il download di foto dal bucket S3 |
|  | /chats | Permette il recupero delle chats di un particolare utente |

|  |  |  |
| --- | --- | --- |
| METODO HTTP | PERCORSO | DESCRIZIONE |
|  | /itineraries | Permette di sostituire un itinerario aggiornato modificato dall’Admin |
| PUT | /feedback | Permette il sostituire un itinerario attraverso i feedback degli utenti |
|  | /photos | Permette il recupero di presigned-url per l’upload di foto nel bucket S3 |

|  |  |  |
| --- | --- | --- |
| METODO HTTP | PERCORSO | DESCRIZIONE |
| POST | /itineraries | Permette di aggiungere un itinerario creato da un utente nel database |

|  |  |  |
| --- | --- | --- |
| METODO HTTP | PERCORSO | DESCRIZIONE |
| DELETE | /itineraries | Permette di rimuovere un itinerario nel Database da parte dell’Admin |

# Servizi cloud utilizzati

* API Gateway

è un servizio completamente gestito che semplifica la creazione, la pubblicazione, la manutenzione, il monitoraggio e la protezione delle API su qualsiasi scala. Le API fungono da “porta di entrata” per consentire l’accesso delle applicazioni alle funzionalità dai servizi back-end.

* Lambda

è una piattaforma di calcolo serverless ed event-driven.

Lo scopo di Lambda, comparato ad AWS EC2 e altri server tradizionali, è di

semplificare la costruzione di applicazioni on-demand:

infatti, esegue automaticamente il codice senza dover effettuare il provisioning né gestire server. Inoltre, ridimensiona automaticamente le risorse dell'applicazione eseguendo il codice in risposta a ogni trigger.

Il codice viene eseguito in parallelo ed elabora ciascun trigger separatamente, ricalibrando le risorse in base al carico di lavoro.

* Cognito

fornisce autenticazione, autorizzazione e gestione degli utenti per le applicazioni Web e mobili. Gli utenti possono accedere (dopo la registrazione)

direttamente con un’e-mail e una password, oppure tramite Providers di terze parti, come Google di cui abbiamo fornito l’implementazione. Grazie all’integrazione con API Gateway, solo gli utenti registrati (memorizzati nel pool d’utenza) possono accedere ai servizi dell’applicativo.

* Simple Storage Service (S3)

è un servizio di storage di oggetti che offre scalabilità, disponibilità dei dati e sicurezza. Amazon S3 offre caratteristiche di gestione semplici da utilizzare che consentono di organizzare i dati in modo ottimale.

* Relational Database Service (RDS)

Semplifica l'impostazione, il funzionamento e il dimensionamento di database relazionali nel cloud. Questo servizio fornisce una capacità ridimensionabile, automatizzando al tempo stesso le attività di amministrazione del database più dispendiose in termini di tempo, quali il provisioning di hardware, l'impostazione di database, gli aggiornamenti e i backup.

# Architettura Frontend

L’applicazione su client Android si basa sul linguaggio Object-Oriented (come richiesto dal committente) Java. Le classi sono state raggruppate in tre macro-package, rispettivamente *application* (per la logica di controllo), *model* (per la gestione della persistenza dei dati) e *presentation* (per le classi di interfaccia utente), sviluppando un’architettura di tipo **Three-Layer**, in cui ogni strato comunica con lo strato immediatamente superiore e inferiore.

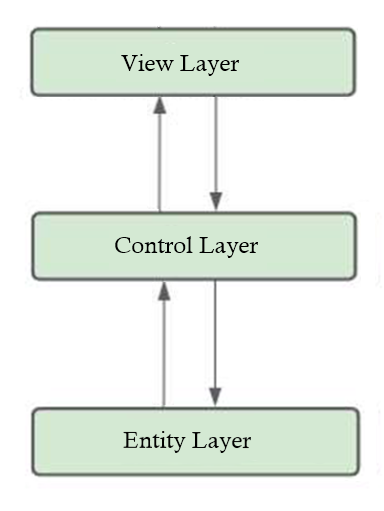


Figura 4: Frontend Three Layer Architecture

Per fare un esempio, quando l’utente interagisce con un *affordance*, dallo strato d’interfaccia grafica si propaga un’azione nel Controller che lo propaga ad una classe di livello inferiore; quando la computazione è terminata (ad esempio si è riusciti ad ottenere una response dal Backend), si ripete il procedimento in modo inverso.

La scelta di utilizzare il pattern Three-Layer deriva dalla sua semplicità, flessibilità nelle modifiche al variare dei requisiti e soprattutto dalla modularità, che ci ha consentito di lavorare contemporaneamente su moduli differenti senza intaccare sul lavoro complessivo, riuscendo ad accelerare il processo di sviluppo software. Ecco una descrizione in dettaglio dei livelli architetturali:

□ View Layer

Rappresenta l’interfaccia utente, è responsabile della rappresentazione dei dati che riceve dal livello dal Control Layer e cattura ogni richiesta propagandola al livello inferiore; per facilitare la costruzione della GUI, oltre alle classi di interfaccia utente sono state create classi custom per rendere accattivante e facilitare la disposizione dei dati (chiamate *Adapter).*

□ Control Layer

È il livello che permette la comunicazione tra le views e i model, nonché il livello che permette di interpretare le richieste degli utenti coordinando i cambiamenti di views e gli update dell’interfaccia grafica in base agli eventi.

□ Entity Layer

Si occupa della definizione delle classi di dominio e gestisce la persistenza dei dati: infatti a questo livello si è utilizzato il pattern *Data-Access-Object* (DAO), in modo tale da offrire al livello di controllo un’interfaccia ad una serie di funzionalità per la gestione di dati e la manipolazione degli oggetti aumentando la manutenibilità con la divisione del livello in due sub-layer.

# Design del Sistema

# Class Diagrams di Design