UNIVERSITÀ DEGLI STUDI DI MESSINA

Corso di laurea Triennale in Scienze Informatiche
Dipartimento MIFT

Progetto JAVA:
TANGOapp

Studente: Zuccarello Federica

Indice

1. Introduzione

Sei un amante dei viaggi e, se fosse per te, non saresti mai a casa? Oggi vorresti andare a Parigi, domani a New York, il prossimo week-end, invece, a Tokyo? Sei un vero globetrotter e vuoi stare al passo con la tua passione per le diverse culture e tradizioni? Vuoi iniziare a pianificare il tuo viaggio, ma non sai quale città scegliere?
Una volta scelta la tua vacanza, hai una marea di foglietti dove hai appuntato di tutto e sei così disordinato che spesso non li trovi rischiando di dimenticare l'orario del tuo volo o qualcosa di importante da portare con te?
Non disperare, fortunatamente per te è appena uscita sul mercato la nuovissima TANGOapp. In pochissimi passaggi ti permetterà di poter cercare tutte le città del mondo, inserirle in uno dei tuoi progetti e scrivere tutti gli appunti di viaggio che vuoi. E pensa che puoi anche stampare il tutto o salvare il documento in PDF, portarlo con te su smartphone e condividerlo con i tuoi amici.
Non ti resta altro che andare a provarla oppure continuare a leggere questa tesina per visionare la guida di utilizzo e scoprire i dettagli implementativi.

1.1 Scelte di progettazione

E' stato scelto di scrivere questa tesina utilizzando il tipo di progettazione top-down, ovvero uno stile di programmazione in cui la pianificazione inizia descrivendo le parti complesse e suddividendole successivamente in parti più piccole. Inizialmente verranno, quindi, trattati i "problemi" principali per poi approfondire verso i "sottoproblemi" che li caratterizzano.
Il linguaggio di programmazione utilizzato è Java, linguaggio di programmazione ad alto livello, orientato agli oggetti e a tipizzazione statica, che si appoggia sull'omonima piattaforma software e specificamente progettato per essere il più possibile indipendente dalla piattaforma hardware di esecuzione.

2. Descrizione del progetto

Questa descrizione ad alto livello fornisce all'utente una guida per utilizzare l'applicativo nel modo più efficace possibile. Verranno descritte tutte le funzionalità senza entrare nei particolari progettuali.

2.1 Sistema di Login

Il software applicativo comprende per l'utente un sistema semplice di autenticazione [Figura 2.1], che può essere effettuato attraverso l'inserimento manuale delle credenziali oppure utilizzando il sistema di autenticazione tramite Linkedin.

Figura 2.1 Interfaccia Login di TANGOapp

Se l'autenticazione andrà a buon fine si ha l'accesso alla piattaforma TANGOapp [Figura 2.2], dove verrà visualizzato un Tutorial che spiega all'utente ciò che la piattaforma offre.

2.2 Applicazione TANGOapp

Come si può notare dalla Figura 2.2:

- in alto a sinistra, alcune schede sono oscurate ed altre invece no, rendendo possibile il loro utilizzo;
- in basso a sinistra è presente il tasto di Logout;

La prima scheda visualizzabile è quella del Tutorial.

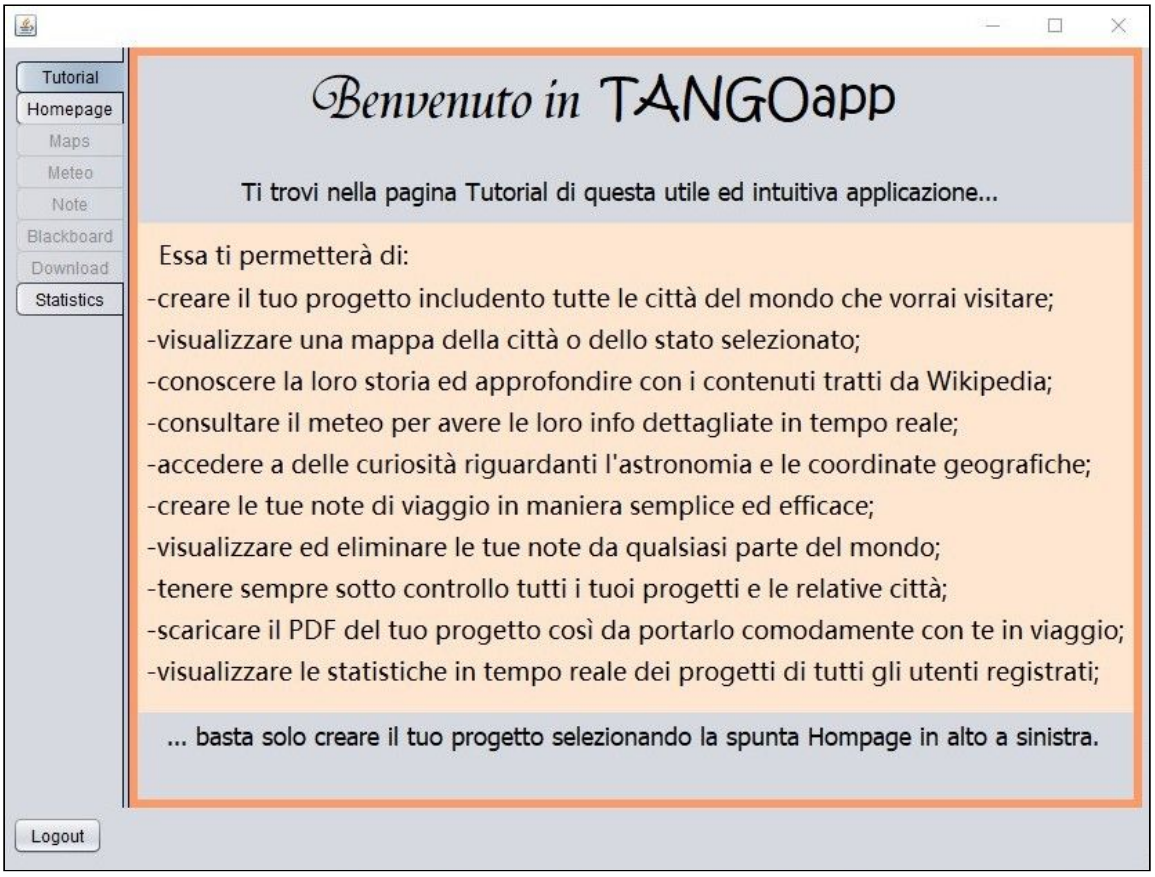

Figura 2.2 Interfaccia Tutorial di TANGOapp

2.2.1 Creazione di un nuovo progetto

La seconda scheda visualizzabile [Figura 2.3], riguarda l'Homepage. E' una delle schede più importanti dove si può creare un nuovo progetto inserendo il suo nome nel campo vuoto e cliccando su "Crea un nuovo progetto" oppure sceglierne uno già esistente nel menù a tendina nella parte sottostante del pannello. Ogni progetto potrà poi contenere da 1 a N città e da 0 ad N note. Una volta creato o scelto il progetto si è pronti per iniziare cliccando sul pulsante "INIZIA". Si sbloccheranno tutte

schede e si aggiornerà il pannello “Project detail” con il nome del progetto scelto.

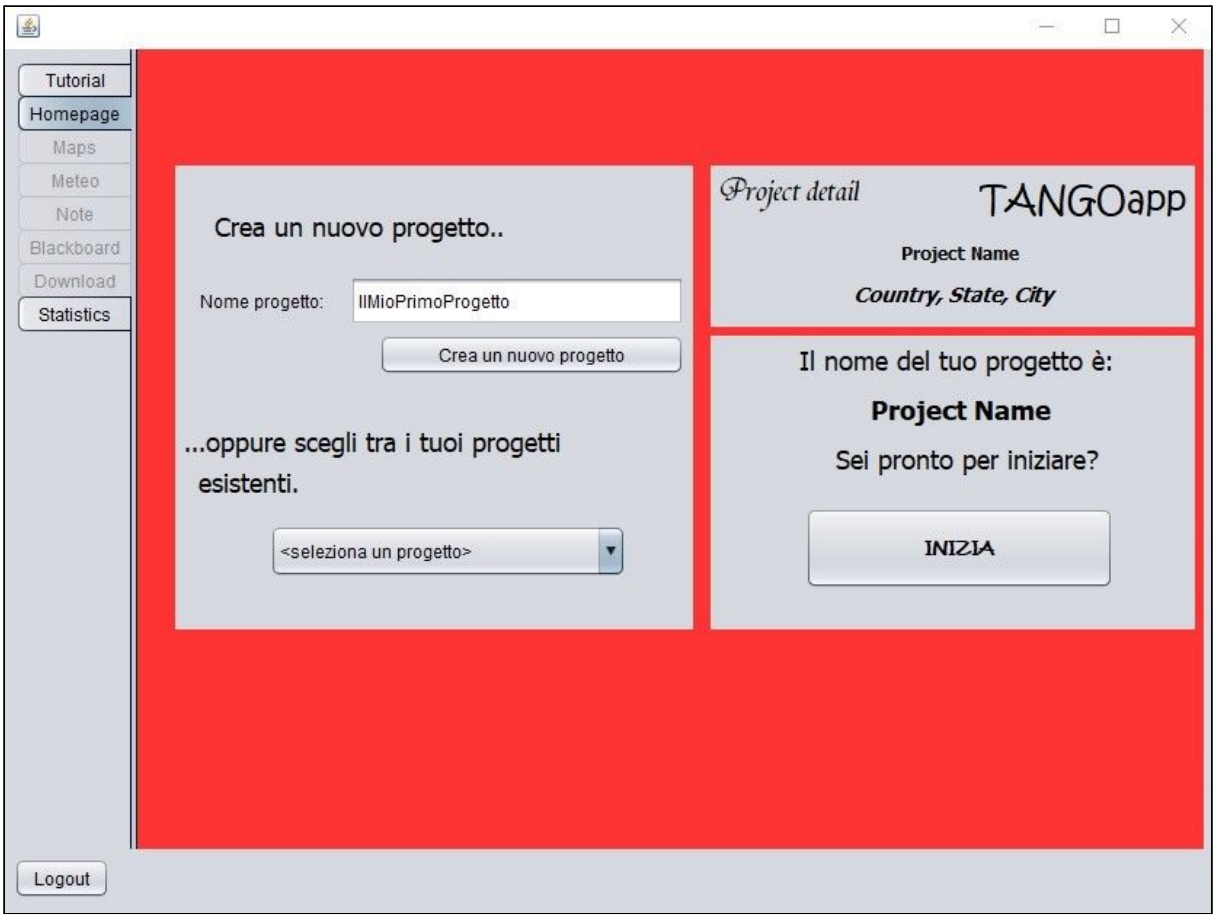

Figura 2.3 Interfaccia Homepage di TANGOapp

2.2.2 Visualizzazione delle statistiche

La terza scheda visualizzabile [Figura 2.4] è quella delle statistiche.
Una breve descrizione spiega quali dati vengono trattati e spiega che tipo di grafico sarà visualizzato. Per visualizzare le statistiche in tempo reale cliccare sul pulsante “Visualizza Statistiche Globali”.
Si aprirà un nuovo pannello [Figura 2.5] contenente il grafico ad Area dei risultati. Sull’asse delle *x* sono presenti tutti gli Stati menzionati dagli utenti, sull’asse delle *y* il numero delle volte che una Città di quello Stato è stata salvata in un progetto. Ogni Stato è differenziato da un colore diverso ed in aiuto si trova la Legenda in basso al grafico.
Questo pannello fornisce la possibilità di scaricare il grafico e modificare alcuni parametri cliccando semplicemente con il tasto destro del mouse su di esso.

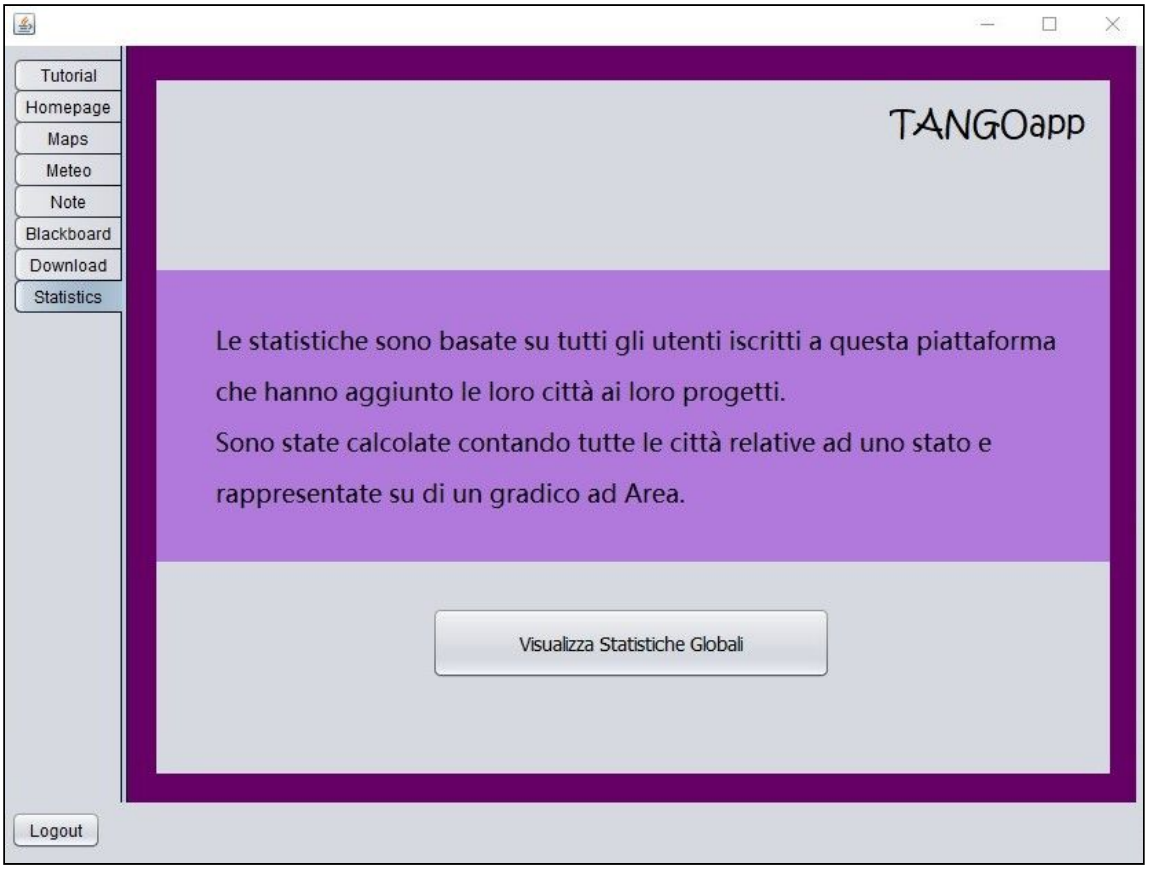

Figura 2.4 Interfaccia descrizione Statistiche di TANGOapp

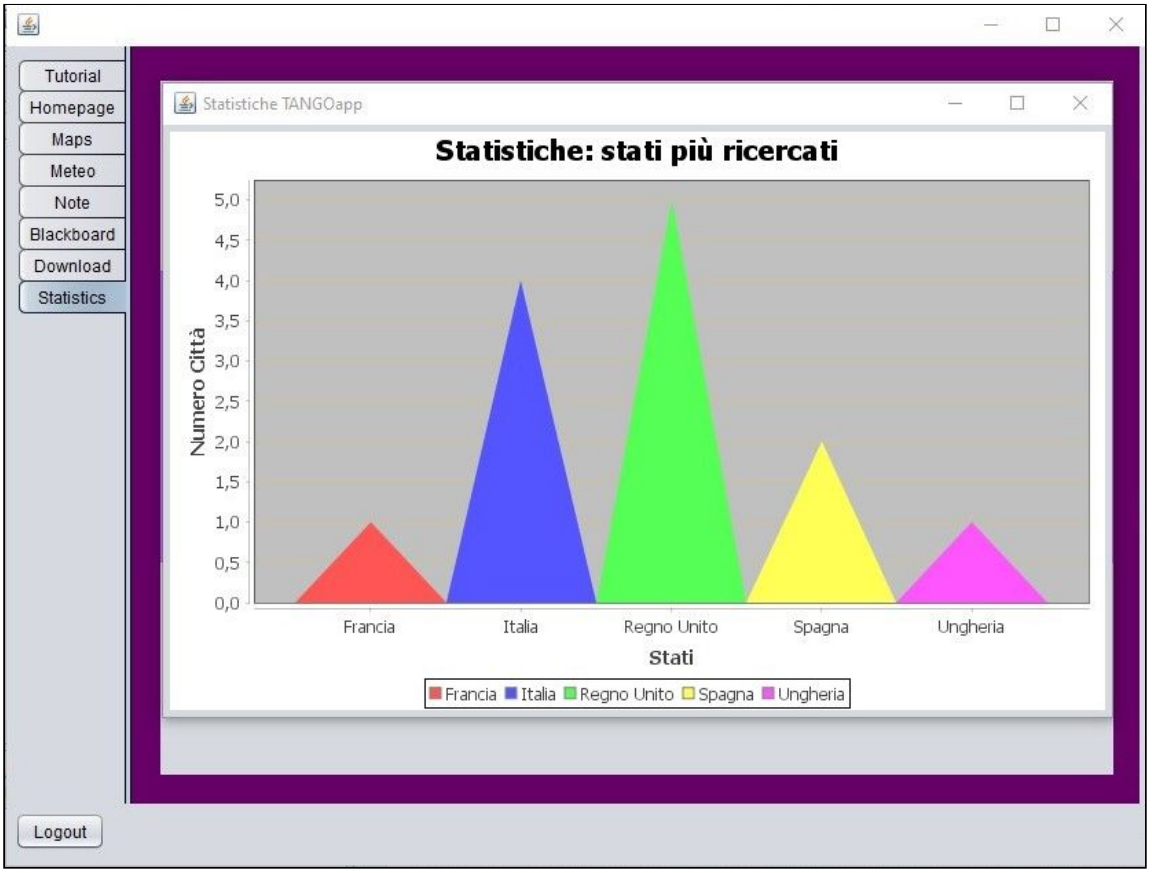

Figura 2.5 Interfaccia Statistiche di TANGOapp

2.2.3 Ricerca delle città

Verrà visualizzata in automatico la scheda Maps [Figura 2.6] una volta scelto o creato il progetto. Inserire nel campo Stato o Città o in entrambi il nome e cliccare sul pulsante “Cerca” per ricercare il luogo desiderato.

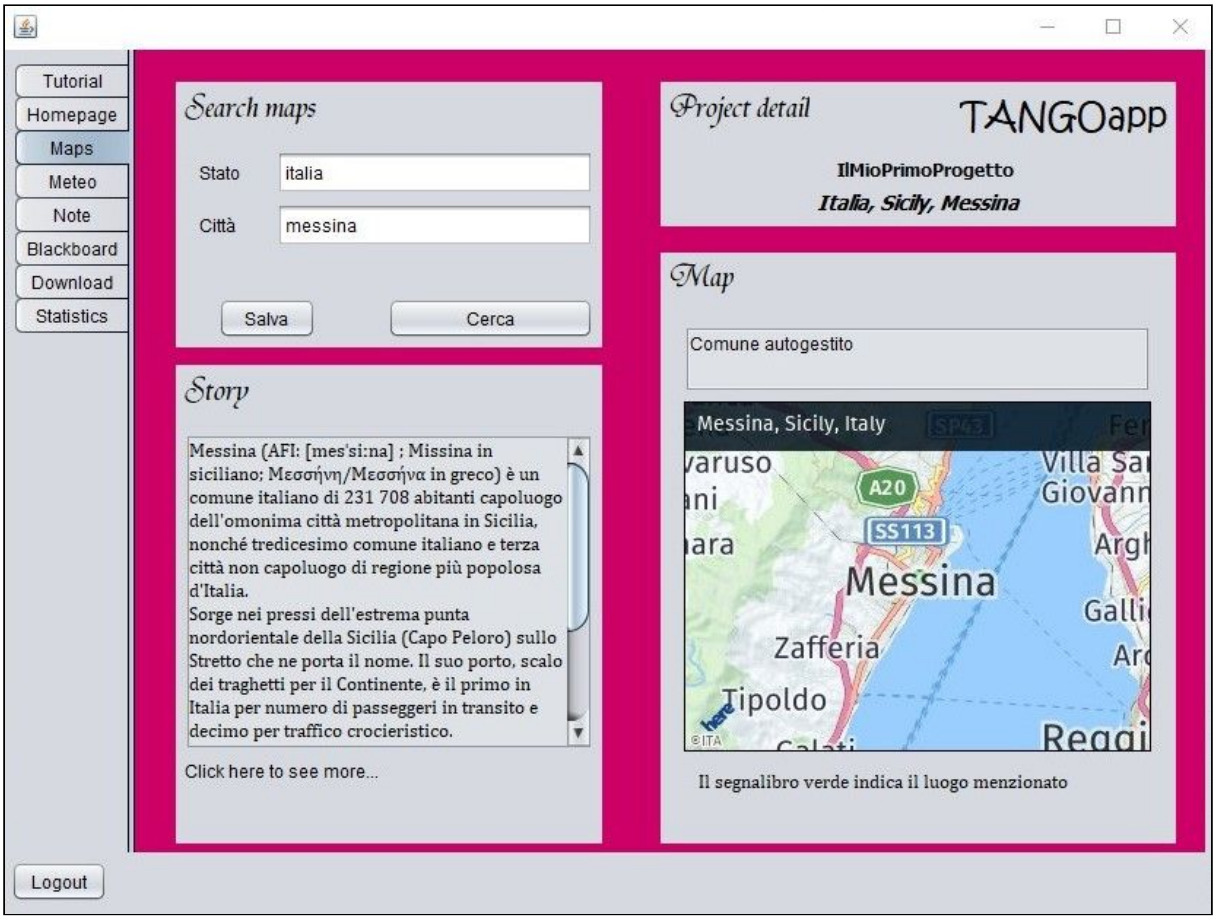

Figura 2.6 Interfaccia Maps di TANGOapp

Verranno visualizzate:

- la storia riguardante il luogo cercato con rispettivo link a Wikipedia per eventuali approfondimenti;
- descrizione breve del luogo cercato, nel blocco “Map” a destra;
- mappa del luogo cercato;

L’utente ha la possibilità di ricercare tutte le Città e gli Stati del Mondo e una volta scelto il suo o i suoi luoghi, può decidere di aggiungerli al suo progetto corrente cliccando sul pulsante “Salva”.
Nel blocco “Project detail” verrà aggiornato il label con l’ultima città salvata.

2.2.4 Il meteo e l'astronomia

La scheda successiva riguarda l'astronomia, a sinistra, e il meteo, a destra [Figura 2.7].

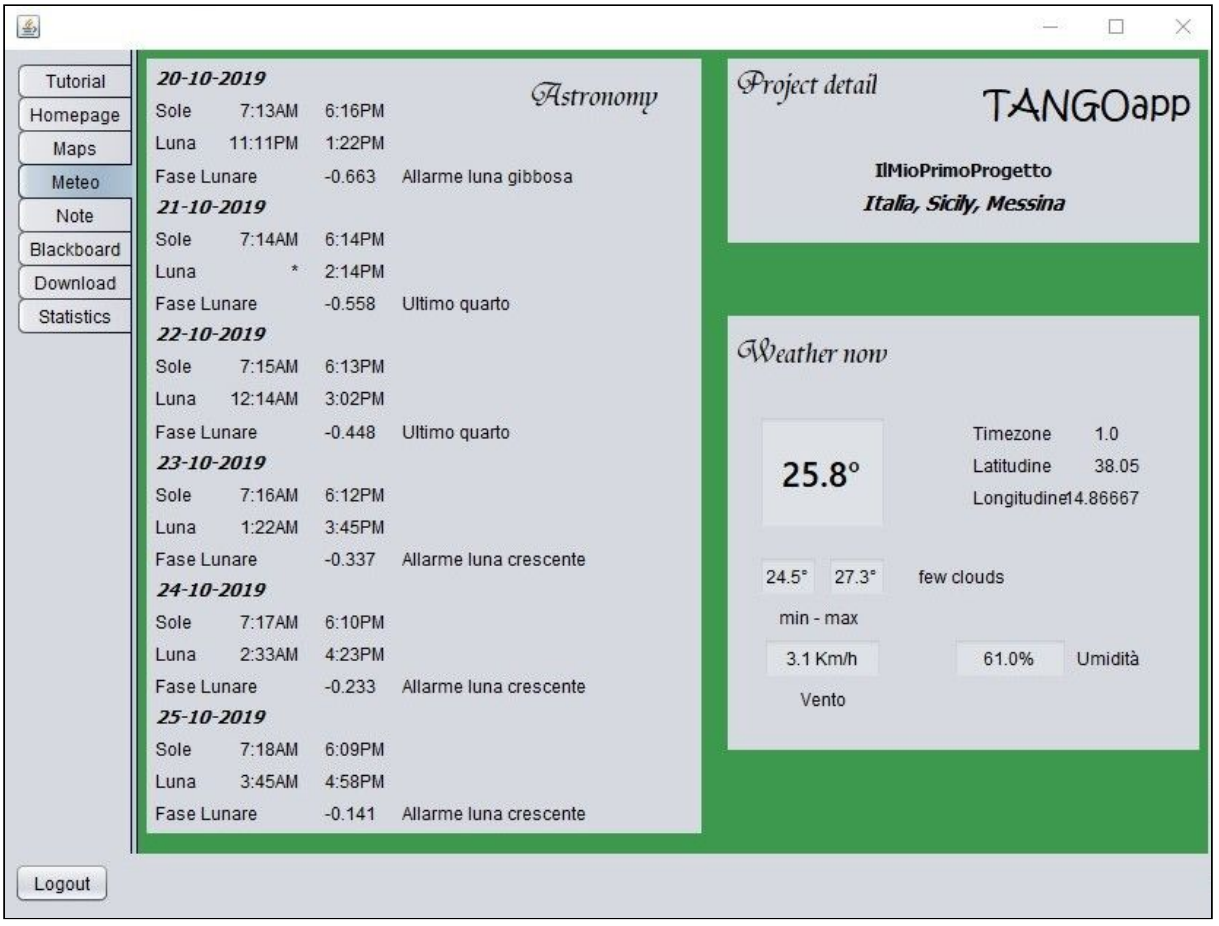

Figura 2.7 Interfaccia Meteo di TANGOapp

Il pannella "Astronomy" è composto dalla descrizione di 6 giorni a partire dalla data di ricerca:

- data di riferimento;
- alba e tramonto solare;
- alba e tramonto lunare;
- fase lunare rappresentata dai valori tra:
    - -1, luna piena;
    - 1, luna nuova
- descrizione fase lunare;

Il pannello “Weather now” è composto dai valori relativi al momento della ricerca:

- temperatura;
- temperatura massima;
- temperatura minima;
- vento;
- umidità;
- descrizione meteo;
- timezone del luogo;
- latitudine e longitudine;

2.2.5 Creazione e visualizzazione delle note

La scheda relativa alle note permette di creare delle note associandole al progetto selezionato [Figura 2.8].

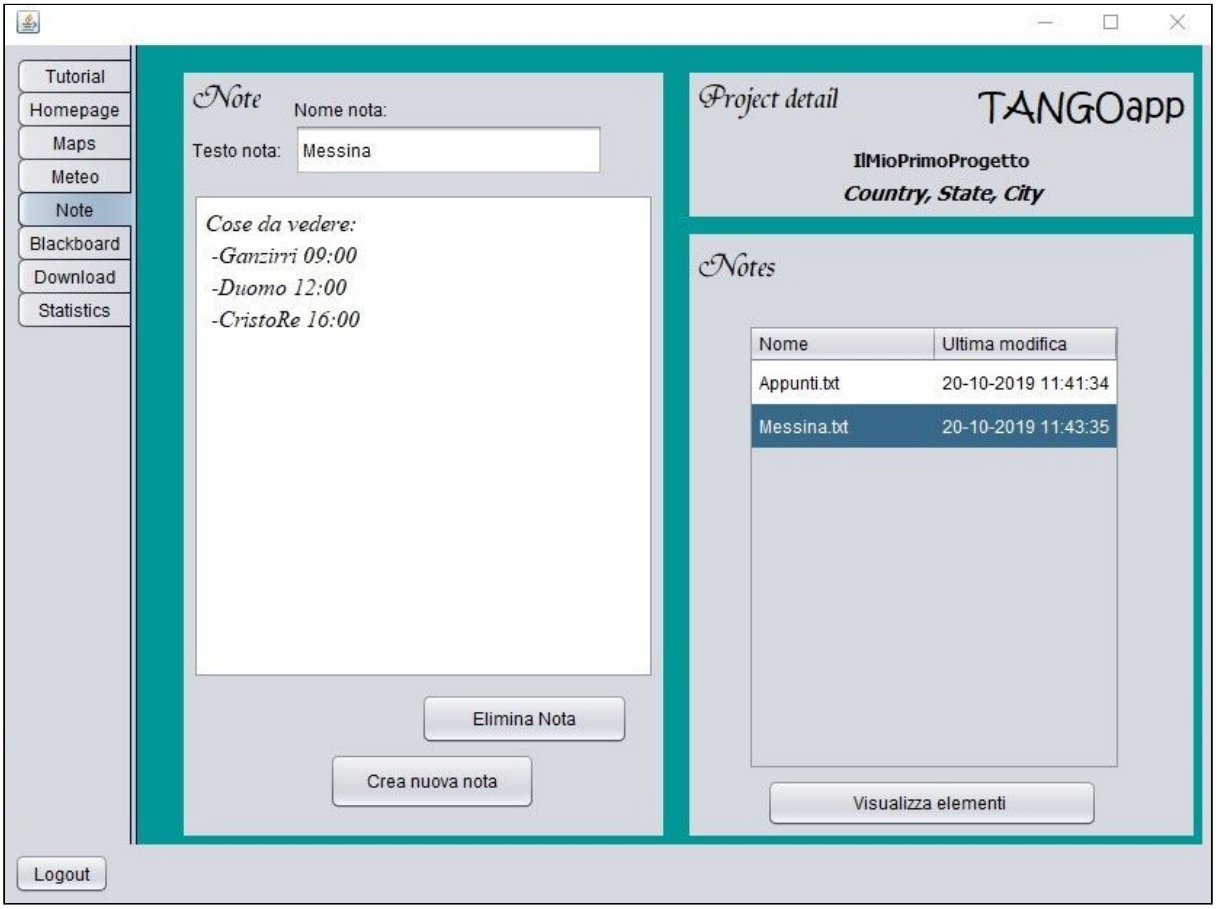

Figura 2.8 Interfaccia Note di TANGOapp

Basta inserire il nome della nota nel campo “Nome nota” ed il testo nel campo “Testo nota” e cliccare il pulsante “Salva”.

Si possono visualizzare tutte le note già salvate dal pannello “Notes” a destra cliccando su “Visualizza elementi” e il testo di una nota selezionando la riga della colonna e “Visualizza” nel messaggio [Figura 2.9]. Una volta visualizzata la nota si può scegliere di Eliminarla cliccando sul pulsante “Elimina Nota”.

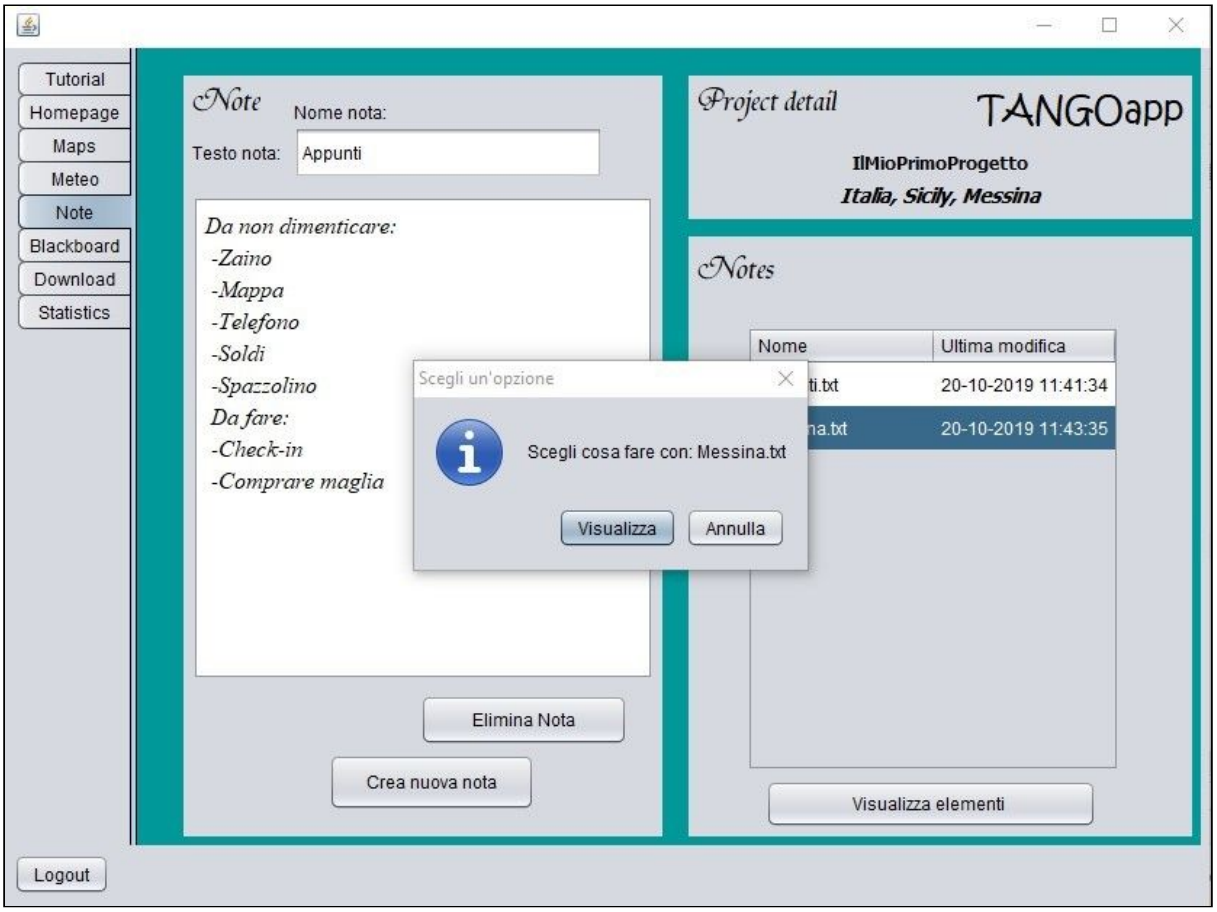

Figura 2.9 Interfaccia Visualizza Maps di TANGOapp

2.2.6 Visualizzazione progetti dell’utente

La scheda “Blackboard” è una lavagna dove, cliccando sul pulsante “Push”, si visualizzano tutti i progetti creati dall’utente con le relative Città, Stato e data di creazione [Figura 2.10].

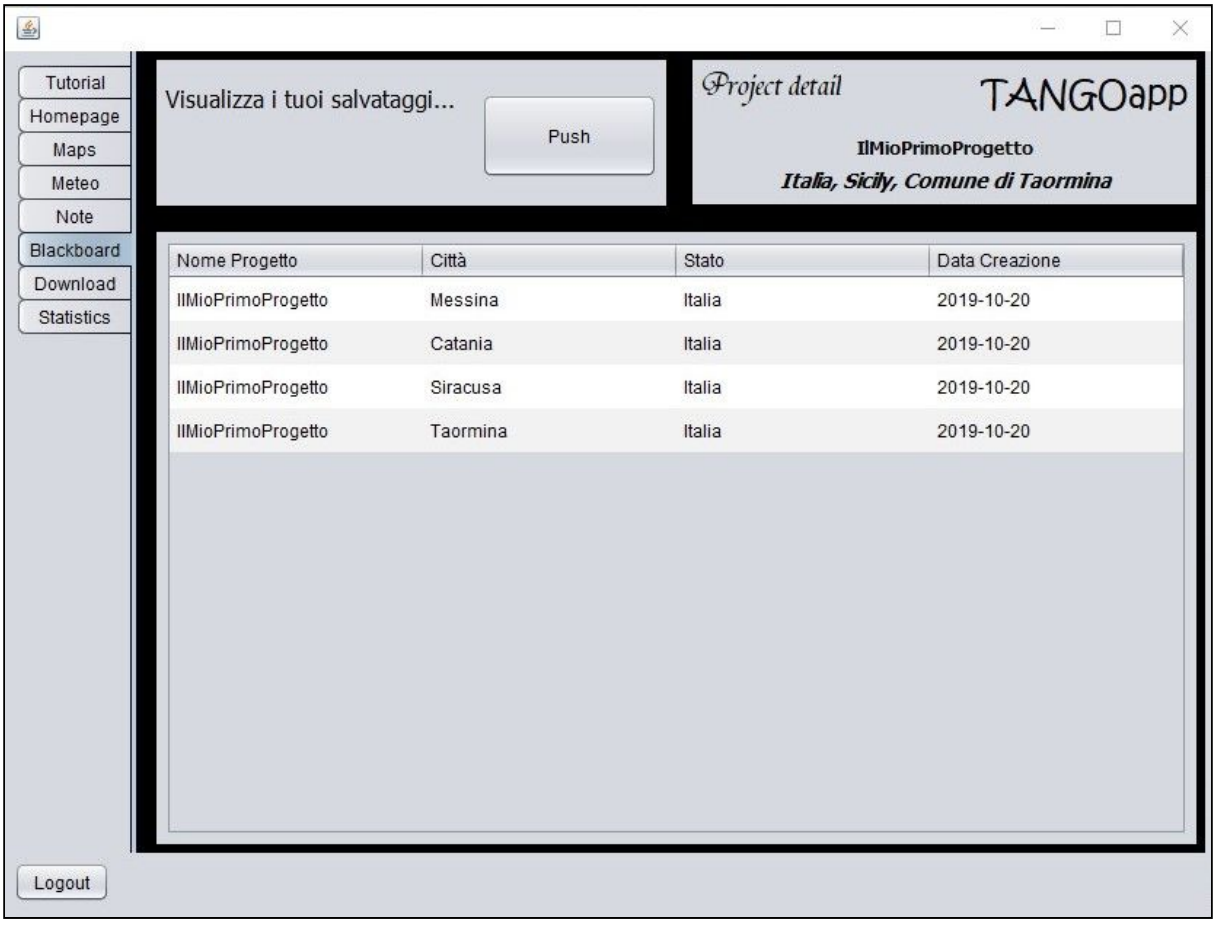

Figura 2.10 Interfaccia Blackboard di TANGOapp

2.2.7 Download PDF del progetto

Questa scheda permette all'utente di poter visualizzare e scaricare tutti i dettagli relativi al progetto selezionato, in formato PDF, cliccando su "Visualizza e scarica" [Figura 2.11].
Si può inoltre risalire ad un PDF creato in precedenza utilizzando il codice univoco che contraddistingue ogni PDF creato nel blocco a destra della Figura 2.11.
Saranno presenti nel PDF:

- descrizione applicazione TANGOapp;
- descrizione progetto:
    - nome;
    - codice identificativo utente;
    - codice univoco PDF;
    - data creazione PDF;
- lista delle città relative al progetto, compreso per ognuna:

- descrizione dettagliata e storia con link di approfondimento;
- mappa formato jpeg;
- meteo attuale con relativa temperatura massima, minima, umidità, vento e descrizione;
- coordinate geografiche quali latitudine longitudine e timezone;
- astronomia per 8 giorni con: fasi lunari e descrizione, orari dell'alba e tramonto solare e lunare;

- tutte le note relative al progetto;

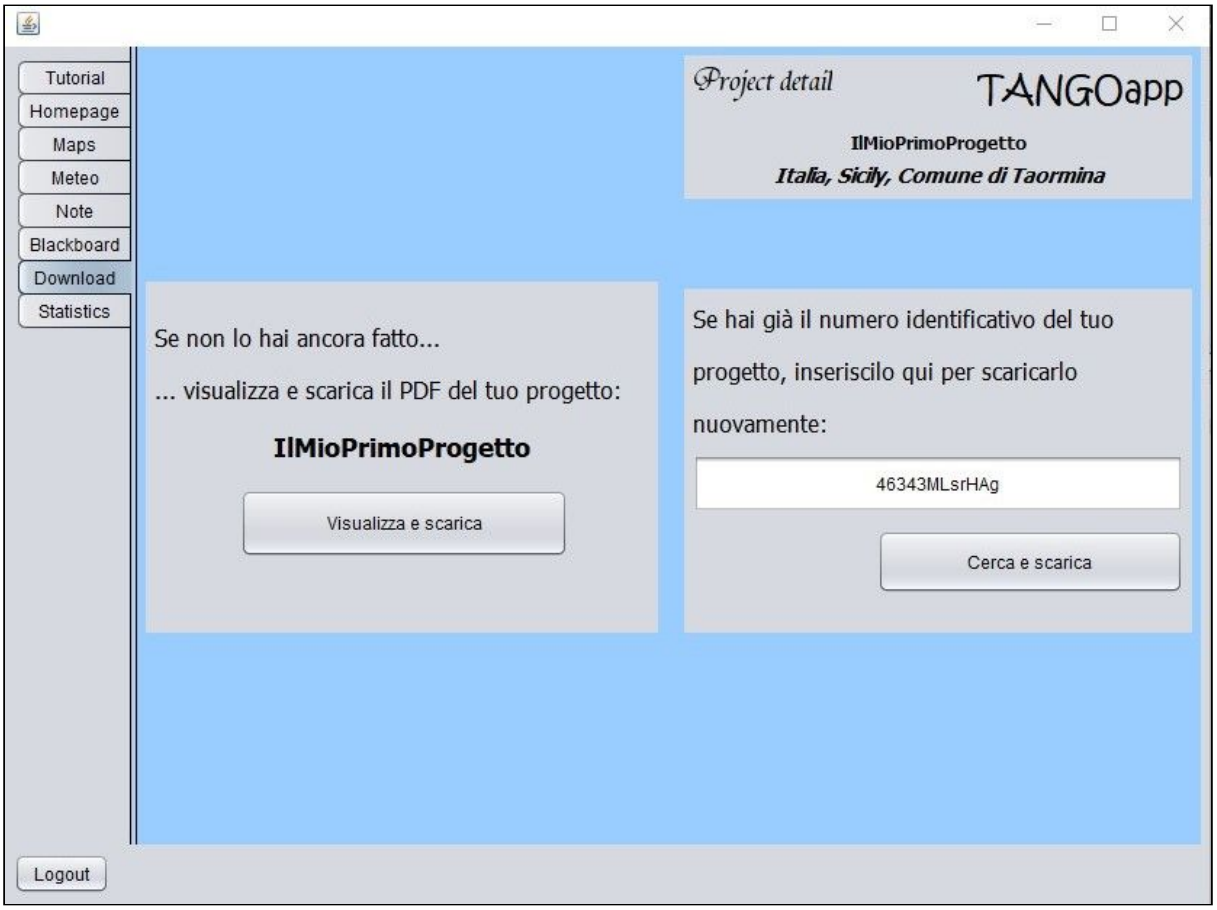

Figura 2.11 Interfaccia Download di TANGOapp

3. Guida per la configurazione

Questa guida è stata scritta per spiegare come configurare il software ai fini di eseguire l'applicativo. Di seguito verranno descritti i software utilizzati e le procedure di esecuzione.

3.1 Software utilizzati

I software utilizzati per lo sviluppo del progetto sono i seguenti:

- Netbeans 8.0.2,
  ambiente di sviluppo integrato (IDE) multi-linguaggio open source. Integra quindi editor, compilatore ed interprete. Esso favorisce una velocità di sviluppo maggiore rispetto all'utilizzo di un semplice editor, permette quindi di scrivere il codice Java, per installarlo si ha bisogno del JDK;

- JDK 1.8,
  il Java Development Kit è un ambiente di sviluppo Java scaricabile gratuitamente dal sito Oracle, che è lo sviluppatore. Implementa una suite di applicazioni come un compilatore, una JVM(Java Virtual Machine, è il software che simula un hardware capace di interpretare ed eseguire il bytecode contenuto in un file Java compilato), il JRE(Java Runtime Machine, è un software che mette a disposizione l'ambiente per utilizzare la JVM, infatti la contiene), un formattatore di documentazione, un'altra JVM per interpretare applet, un generatore di file JAR e così via.

- Apache Maven 4.0,
  è uno strumento di gestione di progetti software basati su Java e build automation. Maven usa un costrutto conosciuto come Project Object Model (POM); un file XML che descrive le dipendenze fra il progetto e le varie versioni di librerie necessarie nonché le dipendenze fra di esse. In questo modo si separano le librerie dalla directory di progetto utilizzando questo file descrittivo per definirne le relazioni.

Maven effettua automaticamente il download di librerie Java e plugin Maven dai vari repository definiti scaricandoli in locale o in un repository centralizzato lato sviluppo. Questo permette di recuperare in modo uniforme i vari file JAR e di poter spostare il progetto indipendentemente da un ambiente all'altro avendo la sicurezza di utilizzare sempre le stesse versioni delle librerie.

3.2 Struttura directory files di configurazione

La struttura che si presenta è la seguente:

- ➔ *"Progetto"*:
  - ◆ *"Relazione"*,
    cartella che contiene tutti i file relativi alla relazione, tra cui anche questo documento;
  - ➢ *"ProgettoJavaEsame.zip"*,
    file .zip che contiene il progetto stesso;
  - ◆ *"dumpDB"*,
    cartella contenente il dump del database;

3.3 Procedure di esecuzione

Il progetto si trova dentro la cartella "Progetto" sotto forma di file .zip che verrà importato direttamente grazie a Netbeans dopo aver eseguito quest'ultimo, attraverso i passaggi:

File > Import Project > From ZIP > *selezionare il file "ProgettoJavaEsame.zip"* > Import.

Per dare profondità al progetto si è deciso di utilizzare un database a scopo statistico, gestionale degli utenti registrati e di elaborazione di alcuni dati. Si importi quindi il dump del database su PHPMyAdmin attraverso i seguenti passaggi:

- attivare la connessione al DB utilizzando ad esempio XAMPP;
- accedere a phpMyAdmin;
- seguire i passaggi:
  Importa > Sfoglia > *selezionare il file .sql dentro la cartella "dumpDB*" > Esegui.

Eseguire tramite Netbeans i due files .java contenuti in

- ➔ *“ProgettoJavaEsame”*,
  - ◆ *“src”* > “main” > “java” > “client” > Client.java;
  - ◆ *“src”* > “main” > “java” > “server”> Server.java;

Il file Client.java permetterà di accedere all’interfaccia di Figura 2.1 ed utilizzare l’applicativo.

4. Progettazione

Seguendo il modello di processo di progettazione descritto in Figura 4.1, la parte fino ad ora trattata fa parte del primo blocco “Design inputs”, in particolare l’Introduzione ed il Capitolo 2, si identificano rispettivamente con i sottoblocchi “Data description” e “Platform information”.

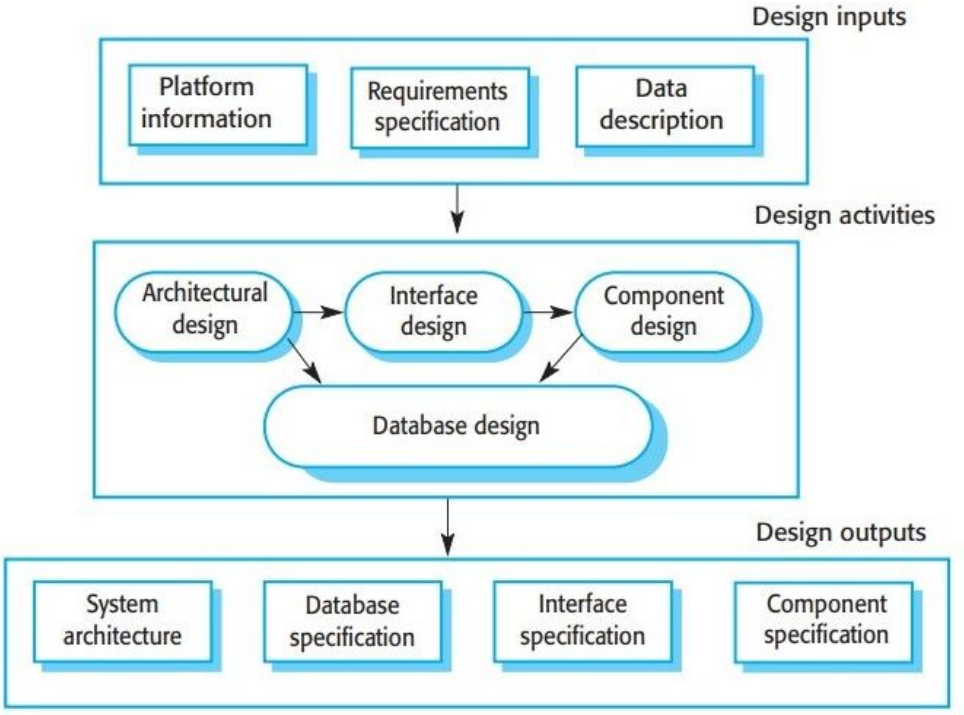

Figura 4.1 Modello generale del processo di progettazione

Il sottoblocco “Requirements specification” verrà trattato nel prossimo paragrafo [Paragrafo 4.1]. I restanti paragrafi riguardano il blocco “Design activities”.

4.1 Specifica e definizione dei requisiti

I requisiti di base che sono stati prestabiliti al fine del superamento dell’esame, sono i seguenti:

1. applicazione client-server REST;
2. comunicazione tramite file xml\json;
3. utilizzo di framework e librerie;
4. non integrare application server;
5. file di configurazioni esterni;
6. meccanismi di cifratura;

7. utilizzo di database;
8. integrazione con almeno tre servizi di terze parti;

4.1.1 Definizione dei requisiti

Verranno spiegati ed approfonditi i requisiti prefissati:

1. Applicazione client-server REST
   Un'applicazione client-server, è un'evoluzione dei sistemi basati sulla condivisione semplice delle risorse, cui gode della presenza di un server che permette ad un certo numero di client di condividerne le risorse, lasciando che sia il server a gestire gli accessi alle risorse appoggiandosi alla sottostante architettura protocollare e per evitare conflitti di utilizzazione tipici dei primi sistemi informatici.

Figura 4.1.1.1 Schematizzazione della comunicazione tra client e server.

Il software *client* in genere è di limitata complessità, limitandosi normalmente ad operare come interfaccia verso il server. In generale nel campo informatico il termine client indica una componente che accede ai servizi o alle risorse di un'altra componente, detta *server*.
Il software *server*, oltre alla gestione logica del sistema, deve implementare tutte le tecniche di gestione degli accessi, allocazione e rilascio delle risorse, condivisione e sicurezza dei dati o delle risorse.
I *client* ed il *server* sono in collegamento tramite un protocollo di comunicazione attraverso una rete di comunicazione [Figura 4.1.1.1]. Il protocollo può essere in chiaro o in certi casi crittografato.
L'applicazione client-server realizzata è di tipo REST o più esplicitamente Representational State Transfer, uno stile architetturale per i sistemi distribuiti che rappresenta un sistema di trasmissione di dati che utilizza il protocollo HTTP senza ulteriori livelli e non prevede il concetto di sessione.

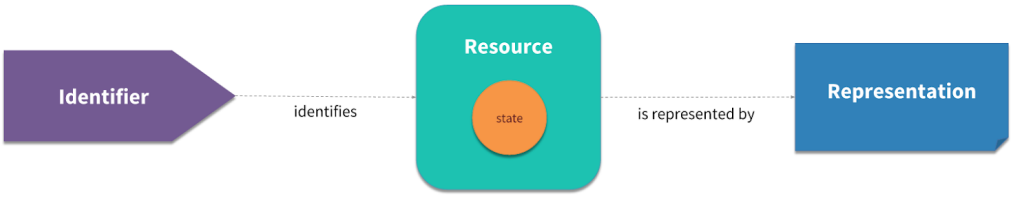

Figura 4.1.1.2 Rappresentazione delle risorse condivisibili contenenti sia metadata, come dimensioni, tipo di supporto o caratteri, che il proprio contenuto come immagini o testo.

Il funzionamento prevede una struttura degli URL ben definita, atta a identificare univocamente una risorsa o un insieme di risorse[Figura 4.1.1.2], e l'utilizzo dei verbi HTTP specifici per il recupero di informazioni (GET), per la modifica (POST, PUT, PATCH, DELETE) e per altri scopi (OPTIONS, ecc.)[Figura 4.1.1.3].

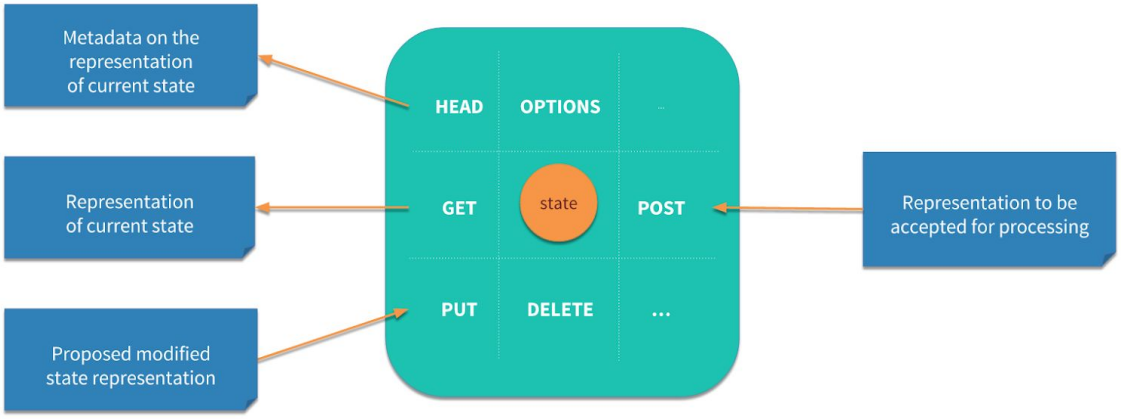

Figura 4.1.1.3 La risorsa è composta da uno stato raffigurato al centro e da metodi standard che insieme definiscono la sua interfaccia uniforme.

Fornire una REST API assegna allo stesso tempo funzionalità utili ai propri consumatori, è un insieme di risorse con collegamenti ipertestuali, che forma un sottoinsieme del Web, esposto da un servizio Web, siti Web, microservizi e così via[Figura 4.1.1.4].

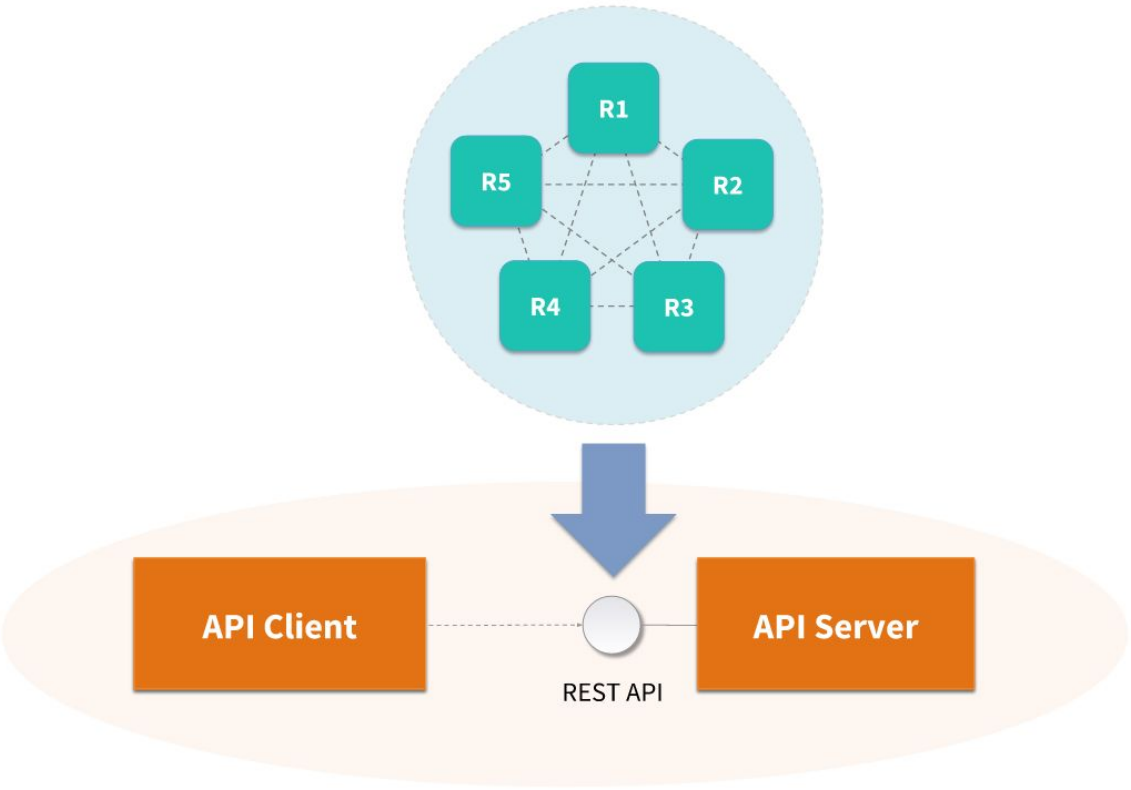

Figura 4.1.1.4 Rappresentazione di una REST API

L'architettura REST utilizzata è RESTlet, framework di API Web RESTful completo e leggero per Java che consente di abbracciare lo stile di architettura del Web (REST) e di beneficiare della sua semplicità e scalabilità. Permette di fondere i servizi web, siti web e client web in applicazioni web uniformi.
Restlet ha un core leggero, ma grazie alla sua estensione collegabile, è anche un framework REST completo per Java. Supporta tutti i concetti REST (risorse, rappresentazioni, connettori, componenti, ecc.) ed è adatto per applicazioni Web client e server.
Supporta i principali standard Web come HTTP, SMTP, XML, JSON, OData, OAuth, RDF, RSS, WADL e Atom.
l'API Restlet è un piccolo insieme di pacchetti, con classi, interfacce ed enumerazioni che insieme forniscono un framework [Figura 4.1.1.5].

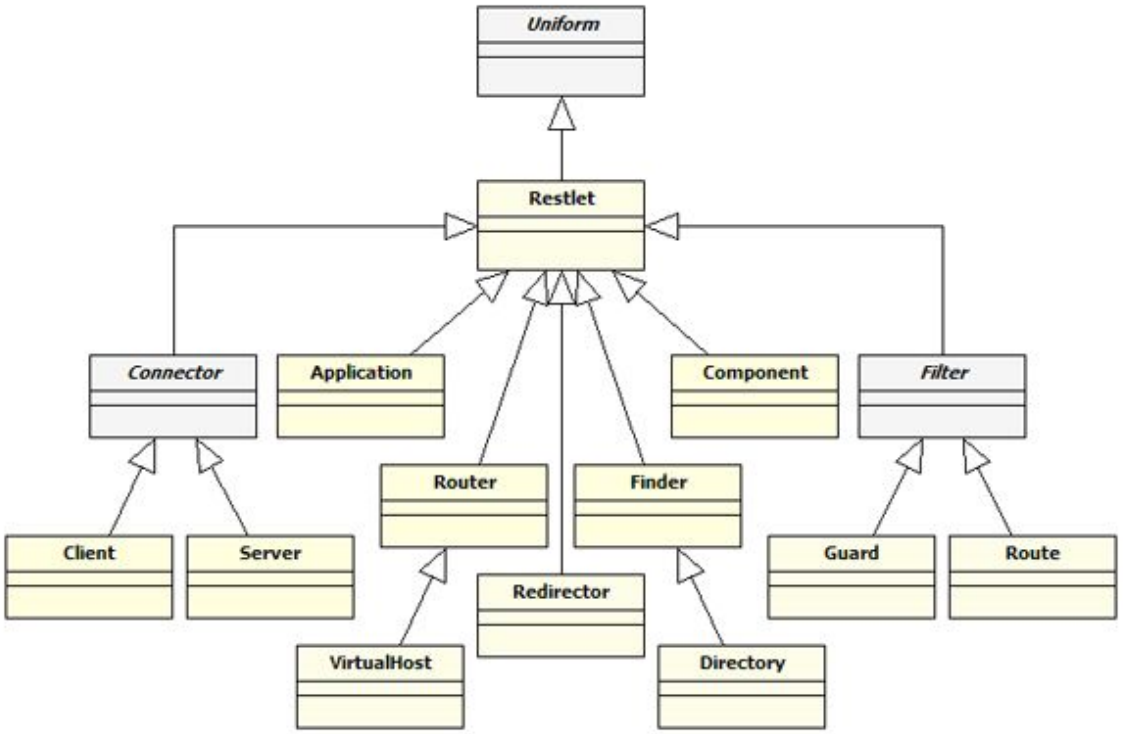

Figura 4.1.1.5 diagramma della gerarchia che mostra i concetti principali dell'API e le loro relazioni

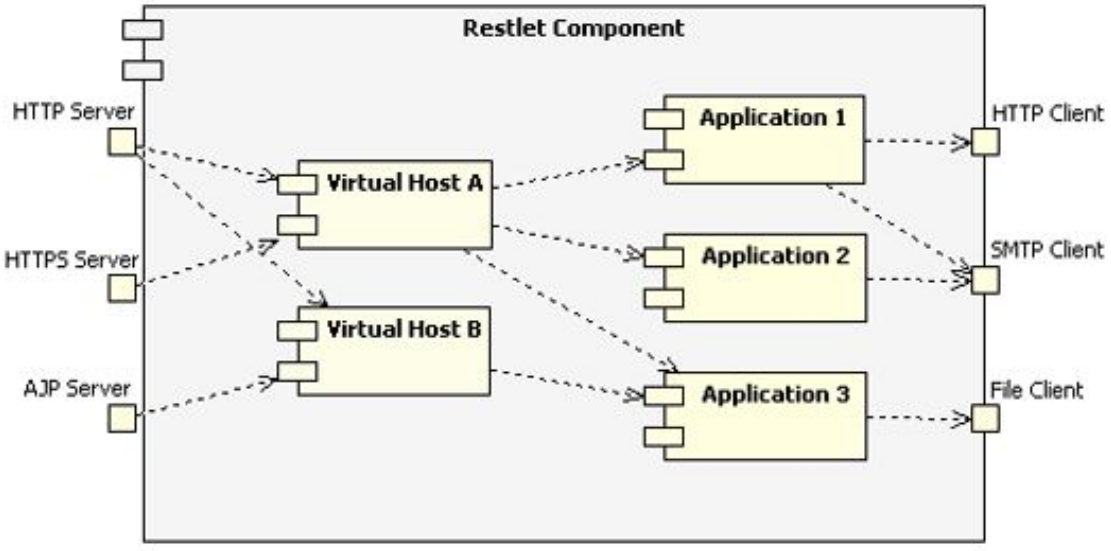

Figura 4.1.1.6 Mappatura delle intestazioni HTTP.
Diagramma che illustra come l'API compone componenti, connettori, host virtuali e applicazioni. Le applicazioni sono a loro volta composte da risorse.

L'API Restlet offre una vista di livello superiore del protocollo HTTP. Cerca di astrarre e presentare in un modello di oggetto pulito, la semantica a livello di applicazione di HTTP.

2. comunicazione tramite file xml\json;

    a. JSON, acronimo di JavaScript Object Notation, è un formato adatto all'interscambio di dati fra applicazioni client/server.
    b. XML, acronimo di eXtensible Markup Language, è un metalinguaggio per la definizione di linguaggi di markup, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.

3. utilizzo di framework e librerie;
   Un framework è un'architettura logica di supporto su cui un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore. Un framework è definito da un insieme di classi astratte e dalle relazioni tra esse.
   Una libreria è un insieme di funzioni o strutture dati predefinite e predisposte per essere collegate ad un programma software attraverso un opportuno collegamento. Lo scopo delle librerie software è fornire una collezione di entità di base pronte per l'uso, ovvero riuso di codice, evitando al programmatore di dover riscrivere ogni volta le stesse funzioni o strutture dati e facilitando così le operazioni di sviluppo e manutenzione. Le librerie e framework utilizzati per lo sviluppo del progetto sono i seguenti:

    ➢ JDBC 8.0.16,
    Java DataBase Connectivity è una libreria per accedere ai database.
    Documentazione: clicca qui.
    Questa libreria permette di:
        1. stabilire una connessione con la sorgente dei dati;
        2. inviare comandi di interrogazione e modifica dei dati;
        3. elaborare i risultati.

Per ognuna delle operazioni precedenti è presente una classe JDBC incaricata della gestione delle operazioni relative alle funzionalità della stessa:

1. `Connection` per collegarsi al DB;
2. `Statement` per eseguire query SQL al "volo" o `PreparedStatement` per SQL dinamico;
3. `ResultSet` racchiude il risultato di una query. Esso è sostanzialmente un array bidimensionale: di righe e colonne.

➢ Swing,
è un framework per Java, appartenente alle Java Foundation Classes (JFC) e orientato allo sviluppo di interfacce grafiche. Parte delle classi del framework Swing sono implementazioni di widget (oggetti grafici) come caselle di testo, pulsanti, pannelli e tabelle.
La libreria Swing viene utilizzata come libreria ufficiale per la realizzazione di interfacce grafiche in Java. È un'estensione del precedente Abstract Window Toolkit. La differenza principale tra i due è che i componenti Swing sono scritti completamente in codice Java.
Documentazione: clicca qui ;

➢ RESTlet 2.3.1,
è un framework di API Web RESTful completo e leggero per Java che consente di abbracciare lo stile di architettura del Web (REST) e di beneficiare della sua semplicità e scalabilità. Restlet ha un core leggero, ma grazie alla sua estensione collegabile, è anche un framework REST completo per Java. Supporta tutti i concetti REST (risorse, rappresentazioni, connettori, componenti, ecc.) Ed è adatto per applicazioni Web client e server. Supporta i principali standard Web come HTTP, SMTP, XML, JSON, OData, OAuth, RDF, RSS, WADL e Atom.
Il framework Restlet è composto da due parti principali [Figura 2.1]. Innanzitutto, esiste la "Restlet API", un'API neutrale che supporta i concetti di REST e facilita la gestione delle chiamate sia per le applicazioni lato client che lato server. Questa API è supportata dal "Restlet Engine" ed entrambi sono forniti in un unico JAR.

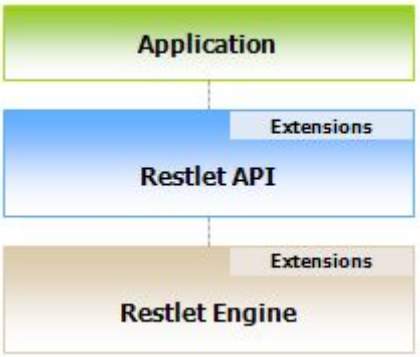

Figura 2.1 Parti del framework Restlet.

Questo framework è al contempo un client e un framework server. Ad esempio, Restlet può facilmente lavorare con risorse remote usando il suo connettore client HTTP. Un connettore in REST è un elemento software che consente la comunicazione tra i componenti, in genere implementando un lato di un protocollo di rete. Restlet offre diverse implementazioni di connettori client basate su progetti open source esistenti.
Documentazione: clicca qui ;

➢ Jackson 2.9.9,
è una libreria JSON standard per Java, inoltre Jackson è una suite di strumenti di elaborazione dati per Java, incluso il parser JSON, la libreria di binding dei dati (POJO da e verso JSON) e moduli di formattazione dati aggiuntivi per elaborare i dati codificati in Avro, BSON, CBOR, CSV, Smile, (Java) Properties, Protobuf, XML o YAML. Documentazione: clicca qui ;

➢ Commons HTTP Client 3.1,
L'HyperText Transfer Protocol (HTTP) è forse il protocollo più significativo utilizzato oggi su Internet. I servizi Web, i dispositivi abilitati alla rete e la crescita dell'informatica di rete continuano ad espandere il ruolo del protocollo HTTP oltre i browser Web guidati dall'utente, aumentando il numero di applicazioni che richiedono il supporto HTTP.
Sebbene il pacchetto java.net offra funzionalità di base per l'accesso alle risorse tramite HTTP, non offre la piena flessibilità o funzionalità necessarie a molte applicazioni. Il componente

HttpClient di Jakarta Commons cerca di colmare questo vuoto fornendo un pacchetto efficiente, aggiornato e ricco di funzionalità che implementa il lato client degli standard e delle raccomandazioni HTTP più recenti. Progettato per l'estensione, fornendo al contempo un solido supporto per il protocollo HTTP di base, il componente HttpClient potrebbe essere di interesse per chiunque costruisca applicazioni client compatibili con HTTP come browser Web, client di servizi Web o sistemi che sfruttano o estendono il protocollo HTTP per la comunicazione distribuita.
Documentazione : clicca qui ;

➢ JFreeChart 1.0.19,
è una libreria di grafici Java che consente agli sviluppatori di visualizzare grafici di qualità professionale nelle loro applicazioni. L'ampia gamma di funzionalità di JFreeChart include:

1. un'API coerente e ben documentata, che supporta un'ampia gamma di tipi di grafici;
2. un design flessibile che è facile da estendere e si rivolge ad applicazioni lato server e lato client;
3. supporto per molti tipi di output, inclusi componenti Swing e JavaFX, file di immagini (compresi PNG e JPEG) e formati di file di grafica vettoriale (inclusi PDF, EPS e SVG);

E' un software open source e distribuito sotto i termini della GNU Lesser General Public License (LGPL), che ne consente l'uso in applicazioni proprietarie.
Documentazione: clicca qui ;

➢ ITextPDF 5.5.13.1,
iText è un SDK sviluppato per consentire agli sviluppatori di:

1. Generare documenti e report basati su dati da un file XML o da un database;
2. Creare mappe e libri, sfruttando numerose funzionalità interattive disponibili in PDF;
3. Aggiungere segnalibri, numeri di pagina, filigrane e altre funzionalità ai documenti PDF esistenti;
4. Dividere o concatenare pagine da file PDF esistenti;

5. Compilare i moduli interattivi;
6. Firmare digitalmente documenti PDF;
7. Offrire documenti PDF generati o manipolati dinamicamente a un browser web;

E' facile usarlo nelle proprie applicazioni in modo da poter automatizzare il processo di creazione e manipolazione dei PDF. In genere, iText viene utilizzato in progetti con uno dei seguenti requisiti:

1. Il contenuto non è disponibile in anticipo: viene calcolato in base all'input dell'utente o alle informazioni del database in tempo reale.
2. I file PDF non possono essere prodotti manualmente a causa dell'enorme volume di contenuti: un gran numero di pagine o documenti.
3. I documenti devono essere creati in modalità automatica, in un processo batch.

Documentazione: clicca qui ;

4. non integrare application server, esempi di application server che non devono essere utilizzati sono: Glassfish e Tomcat;
5. file di configurazioni esterni, ovvero bisogna che non sia hard coded;
6. meccanismi di cifratura dei dati, quindi usare algoritmi come AES256, RSA, etc;
7. utilizzo di database, ma solo ed esclusivamente per dare profondità al progetto stesso, ad esempio per gestire meccanismi di premialità di utenti registrati, realizzare meccanismi di logging utili alla realizzazione di statistiche, etc;
8. integrazione con almeno tre servizi di terze parti da cui estrarre attraverso chiamate HTTP dei dati da elaborare.
   Le API di Terze parti Integrate nell'applicativo sono:

   ❖ Linkedin,
   utilizzato per gestire la fase di Autenticazione;
   Documentazione: clicca qui ;

- ❖ Dropbox,
  utilizzato come storage per la memorizzazione delle note;
  Documentazione: clicca qui ;

- ❖ Open Weather maps,
  utilizzato per avere le informazioni del meteo sulla città ricercata;
  Documentazione: clicca qui ;

- ❖ HERE maps,
  - utilizzato per visualizzare la mappa relativa alla città;
    Documentazione: clicca qui ;
  - utilizzato per avere informazioni sull'astronomia;
    Documentazione: clicca qui ;

- ❖ Wikipedia,
  utilizzato per avere la storia delle città ricercate;
  Documentazione: clicca qui ;

4.1.2 Use Case Diagram Applicativo

In base ai requisiti prefissati, si sceglie di realizzare un'applicazione con le seguenti caratteristiche, rappresentate tramite uno Use Case Diagram.
Uno Use Case è un insieme di scenari, ovvero sequenze di passi che descrivono l'interazione tra un sistema e l'utente. Descrivono cosa il sistema offre, ma non come è realizzato, dal punto di vista dell'utente, quindi ad alto livello [Figura 4.1.2.1].

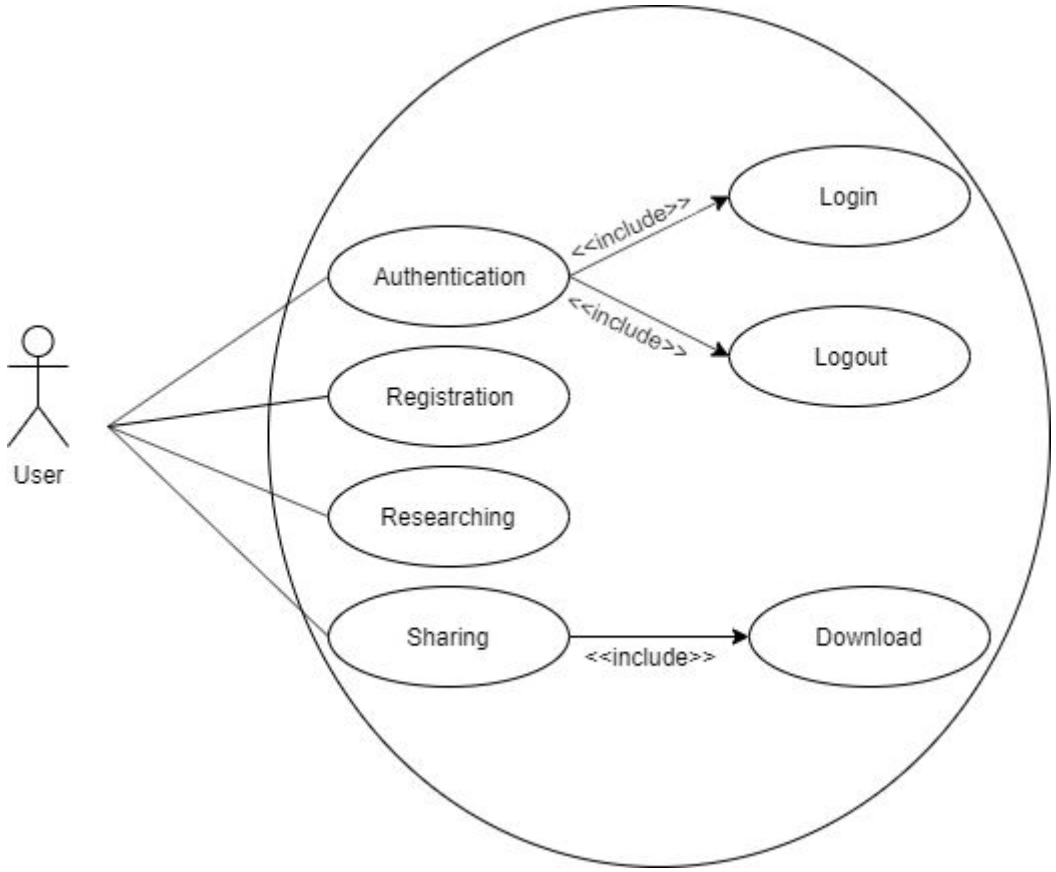

Figura 4.1.2.1 Use Case Diagram del progetto

Le tre componenti fondamentali di questo UML diagram sono:

1. requisiti funzionali, vengono rappresentati come “use cases” e sono come il verbo che descrive un’azione, in Figura 4.1.1.1 sono gli ovali;
2. gli “actor”, interagiscono con il sistema, in questo caso è l’utente identificato come “User”;
3. le “relationship” tra gli “actors” e i “use cases” sono rappresentati da linee o frecce;

L’ovale più grande contiene tutte le azioni che può commettere l’utente, quali Autenticazione, Registrazione, Ricerca e Condivisione delle risorse.

4.2 Architectural design

L'architettura generale dell'applicativo è costituita da due parti, il client ed il server, che come definito nel paragrafo 4.1.1, hanno due compiti diversi. Inoltre ci sarà la figura del servizio di Terze parti che verrà interrogato dal server nel caso dell'Autenticazione o della richiesta di risorse.

4.2.1 Autenticazione

Nella figura 4.2.1 viene rappresentato il Sequence Diagram della fase di autenticazione dell'utente. Vi sono quattro entità: l'utente o User, il Client, Linkedin API che è il Servizio di Terze Parti utilizzato per il login ed il Server alla porta 4000.

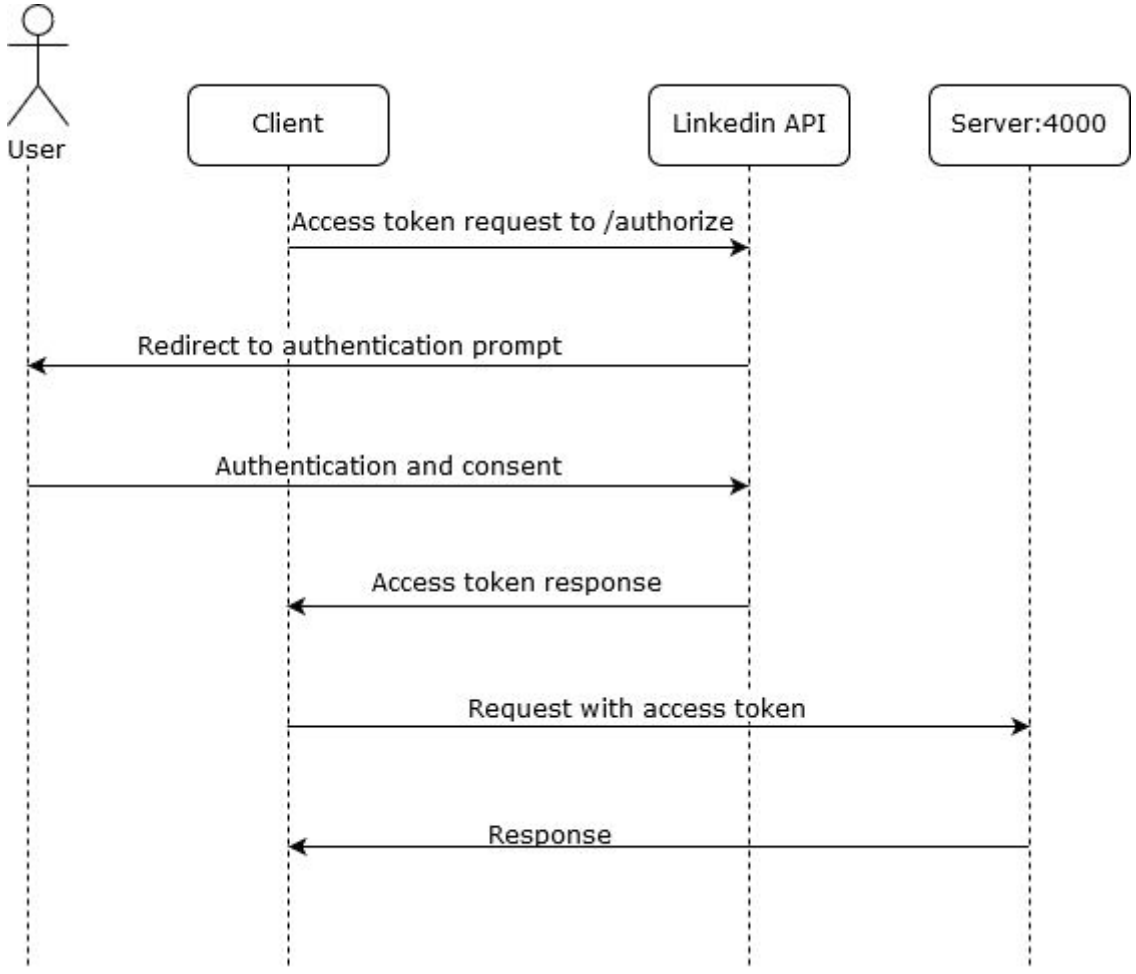

Figura 4.2.1 Sequence Diagram Login

La prima azione che viene compiuta è quella di richiedere l'access token a Linkedin inviando la richiesta di tipo GET:

```
https://www.linkedin.com/oauth/v2/authorization?
response_type=code&
client_id=CLIENT_ID&
redirect_uri=REDIRECT_URI
&state=STATO
&scope=r_liteprofile%20r_emailaddress%20w_member_social
```

I caratteri in maiuscolo sono le costanti identificative proprie del progetto, non vengono inserite direttamente nel codice per non renderlo hard coded, ma si trovano in un file separato. Il CLIENT_ID viene reperito effettuando l'accesso su Linkedin e registrando la propria applicazione, lo STATO serve a verificare che non ci siano stati problemi di alterazione delle informazioni o attacchi CSRF(Cross-site request forgery, è una vulnerabilità a cui sono esposti i siti web dinamici quando sono progettati per ricevere richieste da un client senza meccanismi per controllare se la richiesta sia stata inviata intenzionalmente oppure no).
Nella seconda e terza azione, Linkedin restituisce una pagina web di autenticazione all'utente, dove quest'ultimo inserirà ed invierà le credenziali di accesso e il consenso a Linkedin. Se le scorse azioni hanno avuto successo, Linkedin restituirà un codice temporaneo relativo al client registrato ed un access token tramite la seguenta richiesta POST:

```
https://www.linkedin.com/oauth/v2/accessToken?
Content-Type=application/x-www-form-urlencoded&
grant_type=authorization_code&
code=code&
redirect_uri=REDIRECT_URI&
client_id=CLIENT_ID&
client_secret=CLIENT_SECRET
```

Adesso per avere i dati relativi al Client, viene inviata una richiesta GET con il token ricevuto grazie alla precedente richiesta:

```
https://api.linkedin.com/v2/me?
Connection=Keep-Alive&
Authorization=Bearer ACCESS-TOKEN
```

Ciò restituirà un json contenente i dati dell'utente loggato, come questo mostrato:


```
{
"localizedLastName":"COGNOME",
"lastName":{
	"localized":{
		"it_IT":"COGNOME",
		"en_US":"COGNOME"
	},
	"preferredLocale":{
		"country":"IT",
		"language":"it"
	}
},
"firstName":{
	"localized":{
		"it_IT":"NOME",
		"en_US":"NOME"
	},
	"preferredLocale":{
		"country":"IT",
		"language":"it"
	}
},
"profilePicture":{
	"displayImage":"urn:li:digitalmediaAsset:C5603AQGnr1-xnZp2dw
"
},
"id":"akyzSbv32Y",
"localizedFirstName":"NOME"
}
```

4.2.2 Richiesta risorse da terze parti

Nella figura 4.2.2 viene rappresentato il Sequence Diagram della fase di richiesta delle risorse dai servizi di terze parti. Vi sono quattro entità: l'utente o User, il Client, il Server alla porta 4000 e il servizio di terze parti: Servizio API.

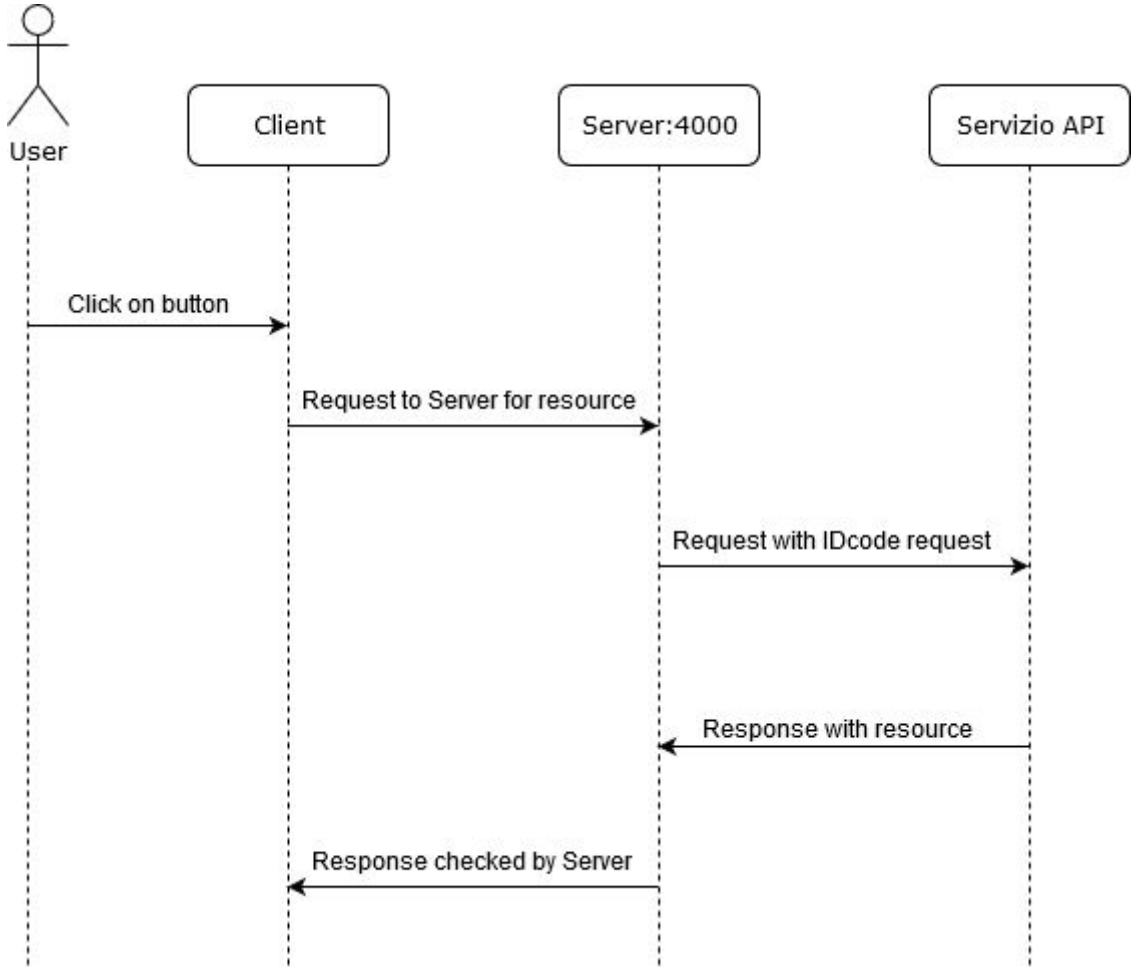

Figura 4.2.2 Sequence Diagram dell'utilizzo di un servizio di terze parti

Quando un utente clicca su un componente button specifico, viene inoltrata una richiesta al Server che a sua volta riformula la richiesta verso il servizio di terze parti con degli specifici parametri, se il servizio accetta la richiesta, ritorna i dati ricercati sotto forma di json al server che dopo averli elaborati adeguatamente per il Client li restituisce. Questi procedimenti verranno spiegati nel dettaglio utilizzando dei diagrammi specifici per ogni componente [Paragrafo 4.4].

4.3 Interface Design

L'interfaccia utente è la vista dell'applicazione front-end a cui l'utente interagisce per utilizzare il software. Il software diventa più popolare se la sua interfaccia utente è:

- Attraente;
- Semplice da usare
- Reattiva in breve tempo
- Chiara da capire
- Coerente su tutte le schermate dell'interfaccia

L'Interfaccia utente grafica fornisce una semplice interfaccia interattiva per interagire con il sistema. La GUI può essere una combinazione di hardware e software. Utilizzando la GUI, l'utente interpreta il software. Il framework grafico utilizzato è Swing, descritto nel Sottoparagrafo 4.1.1.
Per avere tutte le caratteristiche sopra elencate, le interfacce hanno subito un processo di validazione che segue le fasi rappresentate dalla Figura 4.3.1.

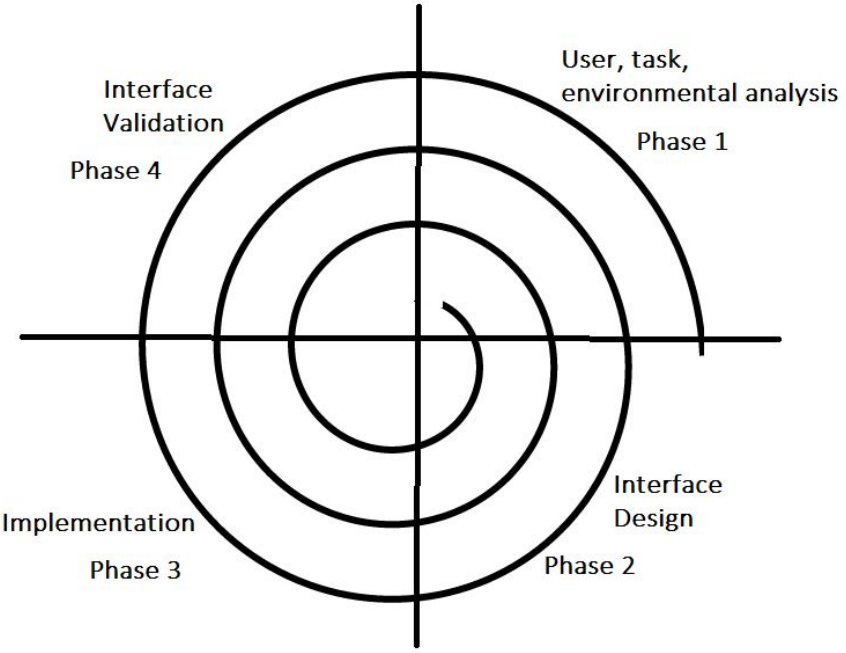

Figura 4.3.1 Processo di progettazione dell'interfaccia utente

Fase 1 **Utente, compito, analisi ambientale e modellistica:** inizialmente, l'attenzione si basa sul profilo degli utenti che interagiranno con il sistema, ovvero comprensione, abilità e

conoscenza, tipo di utente, ecc., In base al profilo dell'utente si possono definire diverse categorie. Da ogni categoria vengono raccolti i requisiti. La categoria che si vuole coinvolgere è quella che va dagli 8 ai 99+ anni, ovvero tutti gli utenti capaci di leggere e scrivere, infatti l'interfaccia è stata resa molto semplice, intuitiva e con il minimo di interazioni disponibili in modo da facilitarne l'utilizzo. Da altre analisi si prevede che l'applicativo venga utilizzato da laptop o fisso, quindi l'utente avrà una postura seduta e con buona illuminazione;

Fase 2 **Progettazione dell'interfaccia:**

l'obiettivo di questa fase è definire l'insieme di oggetti e azioni dell'interfaccia, ad esempio meccanismi di controllo che consentono all'utente di eseguire le attività desiderate. Ciò che si evince dall'analisi è che il modo più semplice per guidare l'utente, è di nascondere o visualizzare alcuni bottoni in base al loro utilizzo:

a. Nell'interfaccia "Maps" [Figura 2.6], il bottone "Salva" compare solo dopo aver effettuato una ricerca;
b. Nell'interfaccia "Note" [Figura], il bottone "Elimina Nota" comparirà solo dopo aver visualizzato una nota, non mentre la si crea, al contrario "Svuota Testo", solo quando si sta scrivendo la nota e il pulsante "Salva" solo quando si sta creando la nota e non mentre la si visualizza;

Fase 3 **Costruzione e implementazione dell'interfaccia:**

l'attività di implementazione inizia con la creazione del prototipo (modello) che consente di valutare gli scenari di utilizzo. Questo prototipo è stato realizzato su carta per ogni scenario e poi ricostruito utilizzando il framework Swing;

Fase 4 **Convalida dell'interfaccia:**

questa fase si concentra sul test dell'interfaccia. L'interfaccia è fatta in modo tale da essere in grado di eseguire correttamente le attività e raggiungere tutti i requisiti dell'utente. E' facile da usare e da imparare;

4.4 Component Design

L’applicativo è costituito da molte componenti, esse interagiscono tra di loro grazie alle classi che li definiscono. Nella figura 4.4.1 e 4.4.2 sono rappresentati rispettivamente i diagrammi delle classi dell’applicativo sia lato Client che Server. Mostrano la struttura statica dei classificatori in un sistema. E’ composto da un insieme di classi e relazioni tra di esse. Una classe è una descrizione di un gruppo di oggetti tutti con ruoli simili nel sistema, che consiste in:

- **Caratteristiche strutturali** (attributi):
    - definiscono cosa gli oggetti della classe "sanno";
    - rappresentano lo stato di un oggetto della classe;
    - sono descrizioni delle caratteristiche strutturali o statiche di una classe;
- **Caratteristiche comportamentali** (operazioni) :
    - definiscono cosa gli oggetti della classe "possono fare";
    - definiscono il modo in cui gli oggetti possono interagire;
    - sono descrizioni di caratteristiche comportamentali o dinamiche di una classe;

La rappresentazione grafica di una classe è composta da tre parti:

1. **Nome della classe**
    - Il nome della classe appare nella prima partizione.
2. **Attributi di classe**
    - Gli attributi sono mostrati nella seconda partizione.
    - Il tipo di attributo viene visualizzato dopo i due punti.
    - Gli attributi vengono mappati su variabili membro (membri dati) nel codice.
3. **Operazioni di classe** (metodi)
    - Le operazioni sono mostrate nella terza partizione. Sono servizi forniti dalla classe.
    - Il tipo restituito di un metodo viene visualizzato dopo i due punti alla fine della firma del metodo.
    - Il tipo dei parametri del metodo viene visualizzato tra parentesi dopo il nome del metodo.

I simboli +, -, # e ~ prima di un attributo e il nome di un'operazione in una classe, indicano la visibilità dell'attributo e dell'operazione:

- + indica attributi o operazioni pubblici
- - indica attributi o operazioni private
- # indica attributi o operazioni protetti
- ~ indica gli attributi o le operazioni del pacchetto

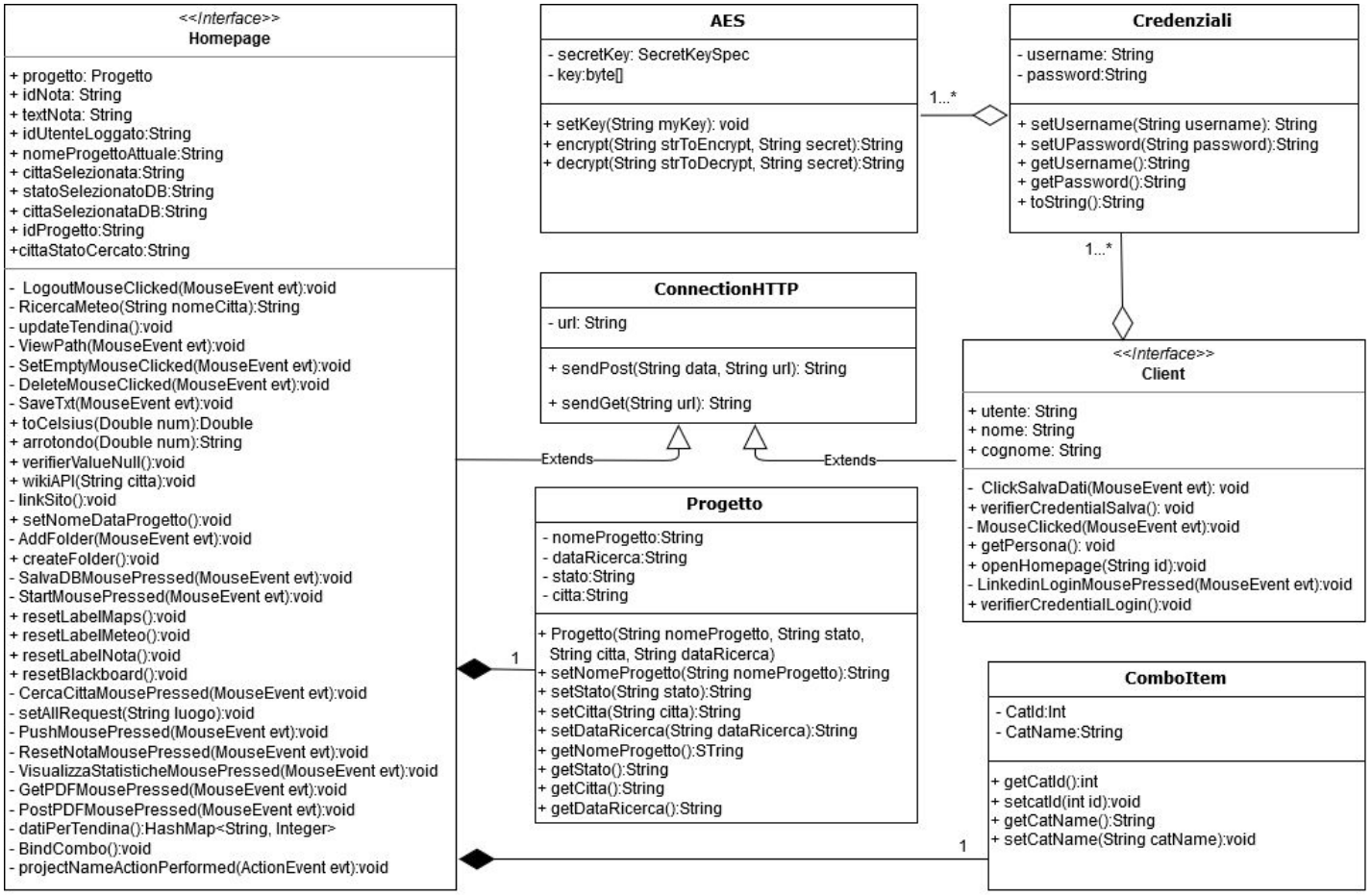

Figura 4.1.1 Class Diagram lato Client.

Lato Client si hanno due interfacce principali, l'interfaccia Client e quella Homepage. Sono le classi principali ed estendono, entrambe, la classe ConnectionHTTP che le permette di inviare richieste HTTP al Server per elaborare le risorse. In particolare la classe Homepage è la più complessa poichè gestisce tutta l'interfaccia grafica dell'applicativo, invece la Classe Client, solo il servizio di Autenticazione e Registrazione per gli utenti.

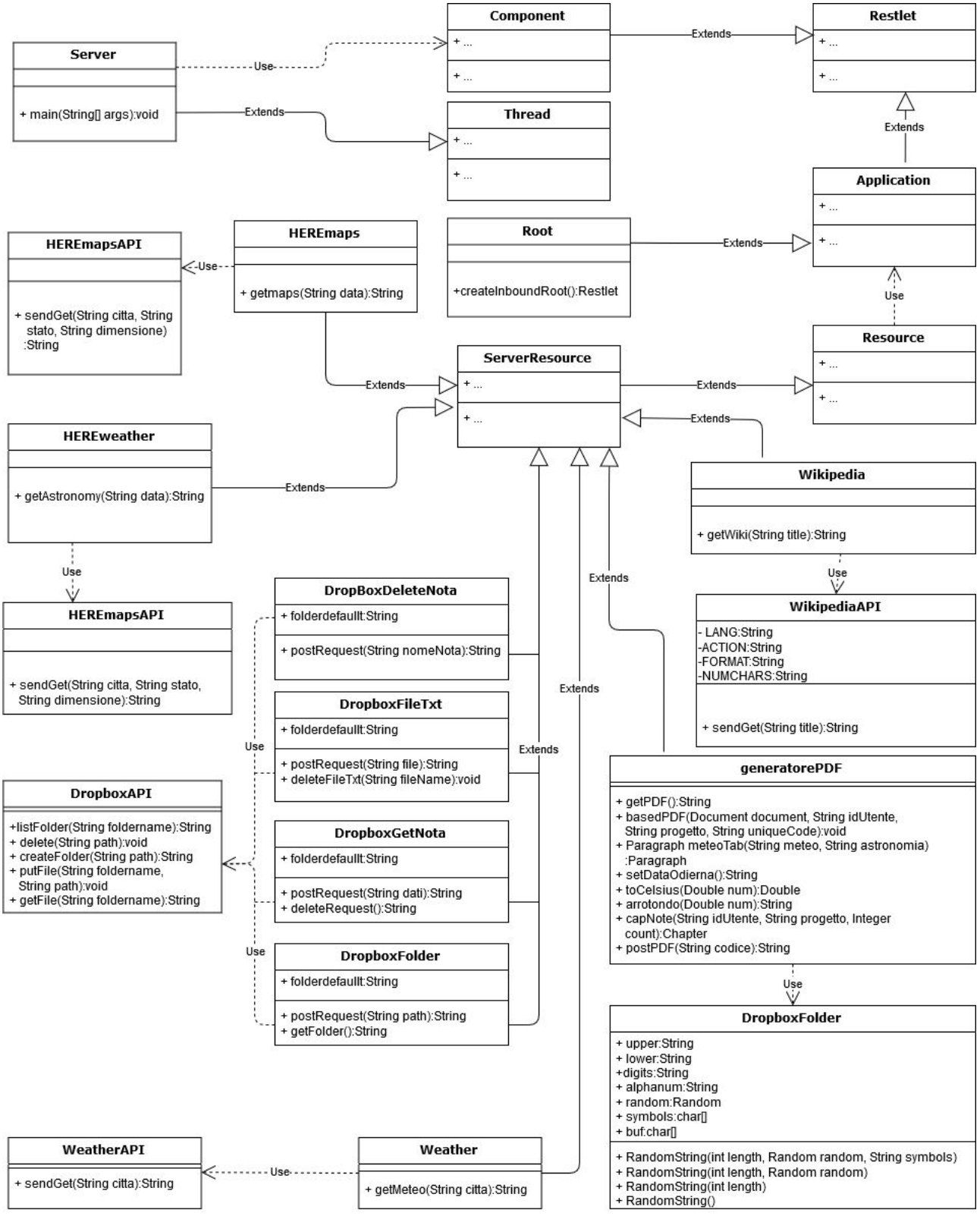

Figura 4.1.2 Class Diagram lato Server

Lato Server si ricevono le richieste HTTP dal Client e si smistano grazie al meccanismo di routing e al framework Restlet.

Le componenti principali che interagiscono tra di loro sono spiegate nei prossimi sottoparagrafi utilizzando degli Activity Diagrams.

4.4.1 Autenticazione Linkedin

All'avvio dell'applicativo, la prima interfaccia visibile è quella chiamata Client, dove si nota il pulsante di Autenticazione rapida tramite Linkedin [Figura 2.1]. Il suo funzionamento dopo il click è rappresentato dalla Figura 4.4.1.1.

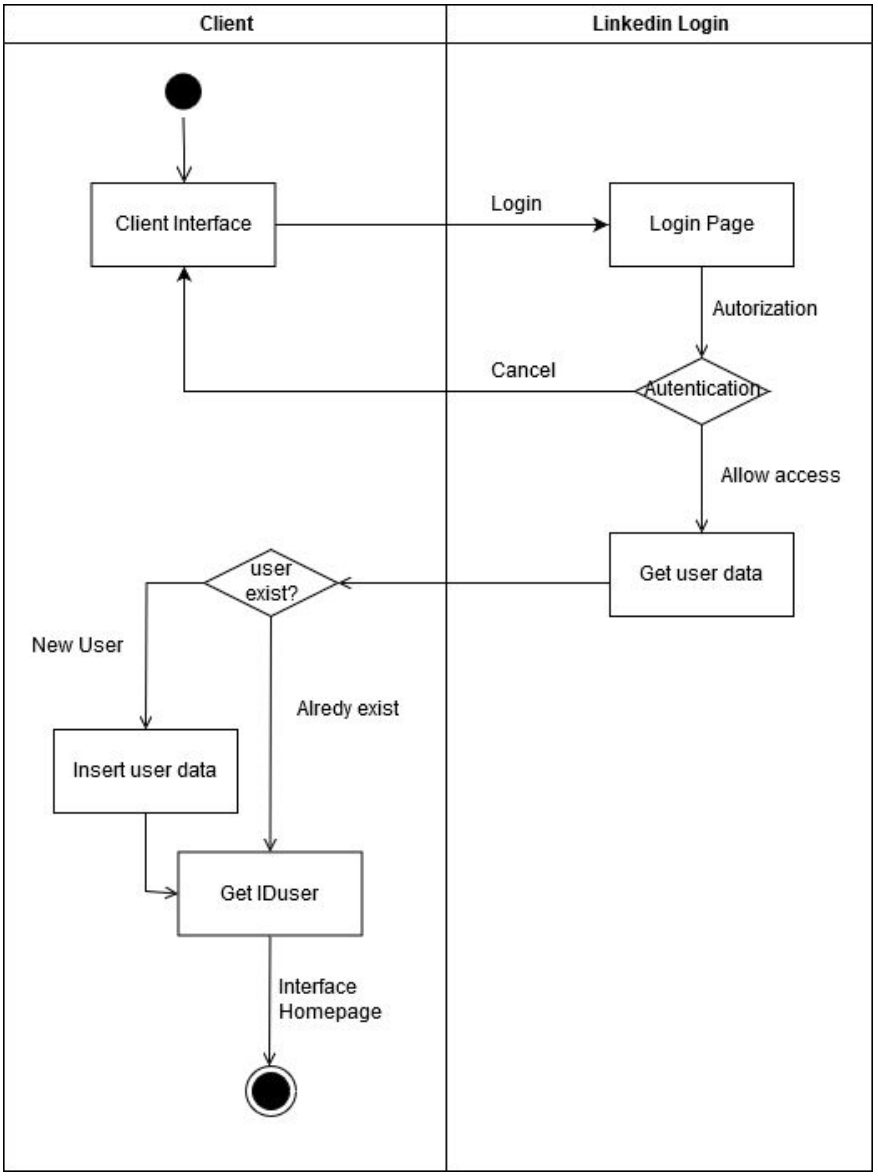

Figura 4.4.1.1 Activity Diagram Login Likedin.

L'utente intende autenticarsi e, dopo aver pigiato sul pulsante "Sign in with Linkedin", avrà di fronte la pagina di login Linkedin.

Se non si autentica, viene reinviato all'interfaccia di login Client, altrimenti viene consentito l'accesso e inviati i dati dell'utente al Client che, dopo aver controllato se l'utente è registrato, tramite la seguente query:

```
SELECT id FROM utenti WHERE id_linkedin = "ID_LINKEDIN"
```

se restituisce 0 righe, lo inserisce nel DB:

```
INSERT INTO utenti(username, id_linkedin)
VALUES ("USERNAME", "ID_LINKEDIN")
```

altrimenti va direttamente all'interfaccia Homepage.

Le richieste HTTP inviate per effettuare la comunicazione con l'API di Linkedin sono descritte in dettaglio nel sottoparagrafo 4.2.1.

4.4.2 Registrazione manuale utente

Un modo per accedere all'applicativo senza usare i social, è quello di registrarsi manualmente dall'interfaccia Client [Figura 2.1].

Nella figura 4.4.2.1 sono analizzate le attività riguardanti la registrazione manuale dell'utente. Dopo aver inserito username e password e cliccato sul pulsante "Registrati", i dati mostrati in formato json:

```
{
        username: "USERNAME",
        password: "PASSWORD_ENCRIPTED"
}
```

verranno usati per verificare se l'utente è già stato registrato o meno inviandoli prima al Server tramite una richiesta HTTP, che elabora i dati in modo da essere comprensibili per il database e fa una query di verifica se è presente l'utente che intende registrarsi:

```
SELECT * FROM utenti WHERE username = "USERNAME" AND
password = "PASSWORD_ENCRIPTED"
```

- se è presente invia al Client un messaggio facendo presente che si è già registrato;

- se non è presente, lo inserisce e prende il suo ID tramite un'altra query. Tutto viene inviato al server, che elabora ed invia tutto al client. Il client estrapola dalla risposta HTTP l'id dell'utente e avvia l'interfaccia Homepage;

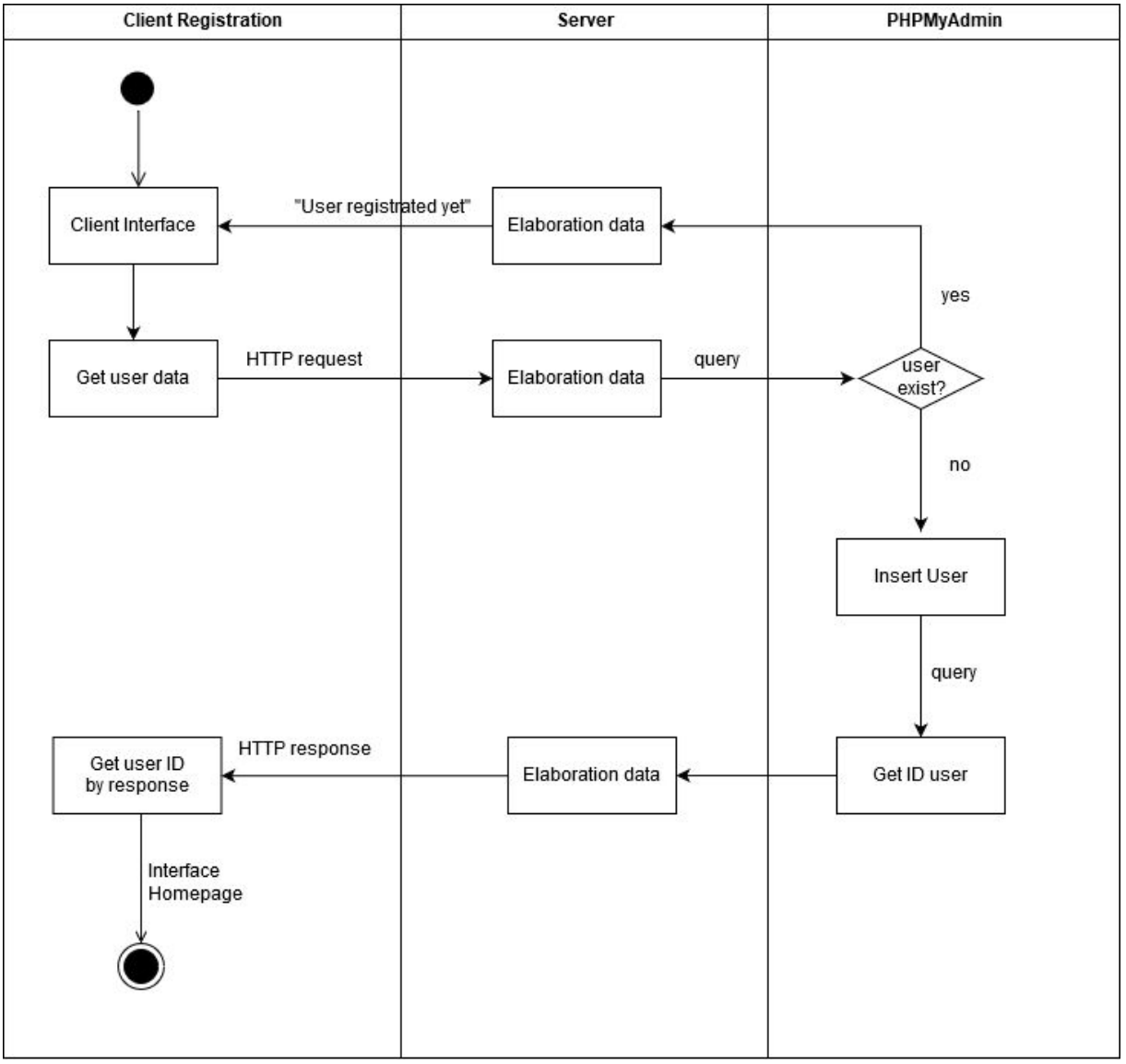

Figura 4.4.2.1 Activity Diagram registrazione manuale utente

4.4.3 Autenticazione manuale utente

Come la registrazione, anche l'autenticazione può essere manuale.
Dalla figura 4.4.3.1, si evince che una volta presi i dati inseriti dall'utente:

```
{
    username: "USERNAME",
```

```
        password: “PASSWORD_ENCRIPTED”
}
```

cliccando il pulsante “Login”, essi vengono inviati al Server che li elabora per renderli comprensibili al database e esegue una query di verifica se l’utente è registrato.

```
SELECT * FROM utenti WHERE username = “USERNAME” AND
password = “PASSWORD_ENCRIPTED”
```

- se non è registrato, ritorna al client un messaggio dicendo che non può autenticarsi;
- se è registrato, prende il suo ID e lo invia tramite HTTP response al client che lo estrapola e avvia l’interfaccia Homepage;

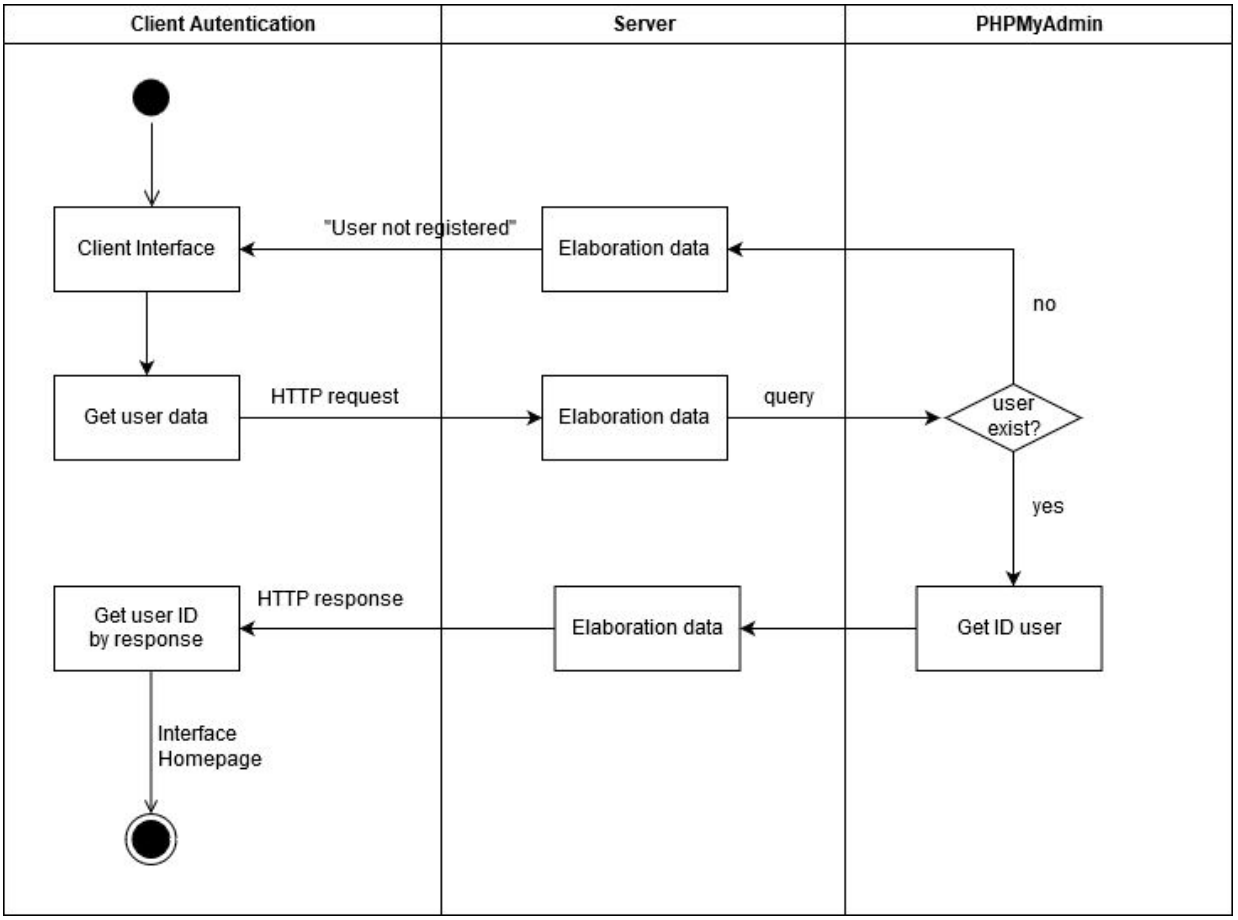

Figura 4.4.3.1 Activity Diagram autenticazione manuale utente

4.4.4 Crea progetto

Per creare un nuovo progetto [Figura 2.3], bisogna cliccare su “Crea un nuovo progetto” e, come si evince dalla Figura 4.4.4.1, verrà inviato un json al server contenente i dettagli relativi al progetto che si vuole creare:

```
{
        nomeProgetto: “NOME”,
        dataRicerca: “DATA_ODIERNA”,
        id_utente: “IDUTENTE”
}
```

Il server elaborerà i dati e invierà una query al database di INSERT:

```
INSERT INTO progetti(nome, data, id_utente)
VALUES (“NOME”, “DATA_ODIERNA”, “IDUTENTE”)
```

seguita da una SELECT ID del progetto appena inserito:

```
SELECT id FROM progetti ORDER BY id DESC LIMIT 1
```

L’iD sarà poi elaborato dal server e reinviato al Client che lo utilizzerà.

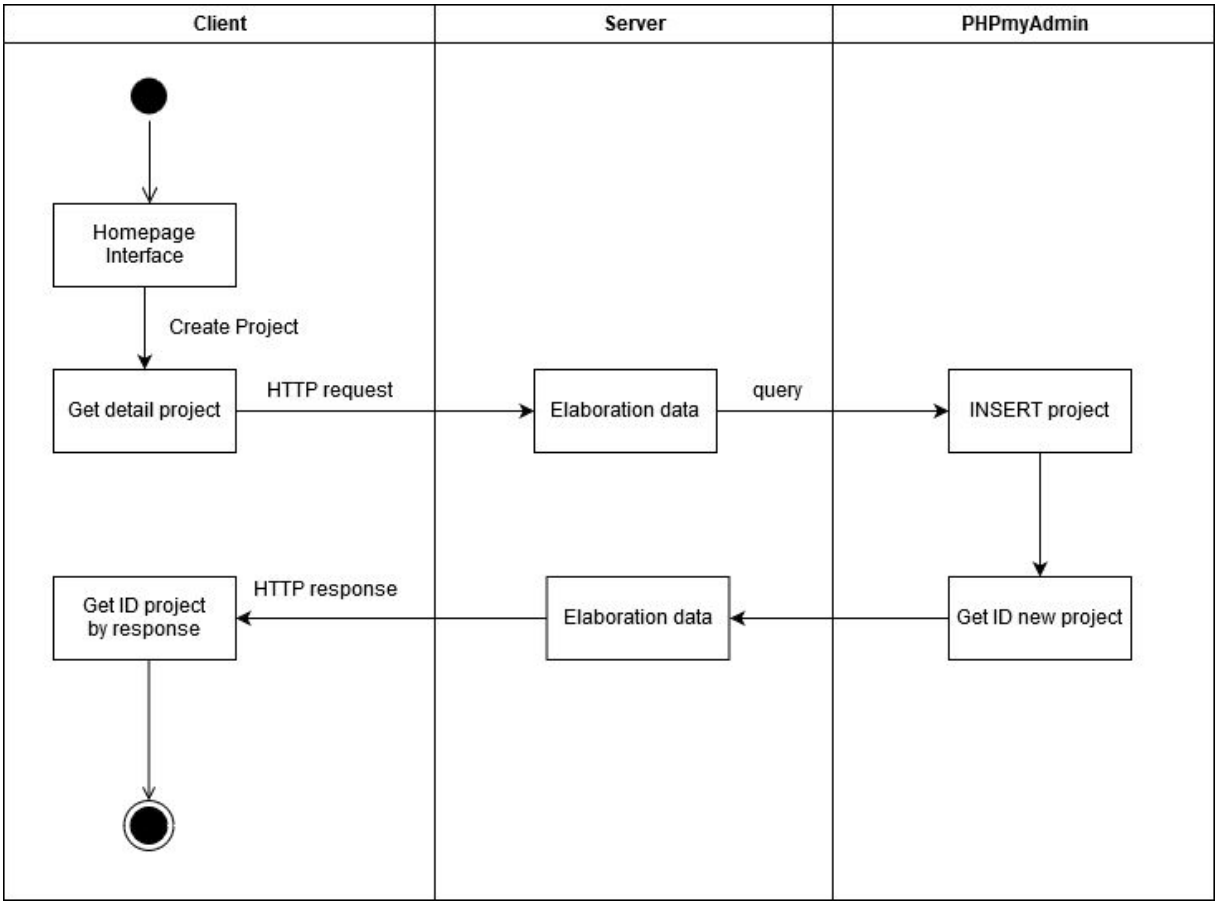

Figura 4.4.4.1 Activity Diagram creazione progetto.

4.4.5 Cerca città

Una volta creato o scelto un progetto, l'utente può cercare tutte le città che vuole prima di salvarle all'interno del progetto. Inserendo nel campo stato o città o entrambi i valori scelti, e cliccando su "Cerca", verrà inviato al server il json:

```
{
        citta: "CITTA'",
        stato: "STATO",
        dimensioneMappa: "DIMENSIONE_MAPPA"
}
```

dopo aver elaborato i dati, il server invia la richiesta HTTP seguente ad Open Weather maps API:

```
http://api.openweathermap.org/data/2.5/weather?
q= “CITTA’”&
APPID= “APP_ID”
```

un esempio di risposta HTTP è la seguente:


```
{
  "coord": {
    "lon": -122.08,
    "lat": 37.39
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 296.71,
    "pressure": 1013,
    "humidity": 53,
    "temp_min": 294.82,
    "temp_max": 298.71
  },
  "visibility": 16093,
  "wind": {
    "speed": 1.5,
    "deg": 350
  },
  "clouds": {
    "all": 1
  },
  "dt": 1560350645,
  "sys": {
    "type": 1,
    "id": 5122,
    "message": 0.0139,
    "country": "US",
    "sunrise": 1560343627,
```

```
    "sunset": 1560396563
  },
  "timezone": -25200,
  "id": 420006353,
  "name": "Mountain View",
  "cod": 200
}
```

- se nel campo “cod” è presente il numero 404 significa che la città non è stata trovata, quindi il server gira la risposta al client che visualizzerà il messaggio: “Città inesistente”;
- se nel campo “cod” è presente il numero 200, la città è stata trovata e si passa allo step successivo;

Lo step successivo prevede l’invio di una richiesta HTTP a HERE maps per ricavare le informazioni sull’astronomia e sulla mappa. Le richieste sono rispettivamente le seguenti:

```
https://weather.cit.api.here.com/weather/1.0/report.json?
product=forecast_astronomy&
name= “CITTA’”
&app_id= “APP_ID”&
app_code= “APP_CODE”
&language=it
```

con un esempio di risposta HTTP:

```
{
  "astronomy": {
  "astronomy": [
    {
    "sunrise": "7:12AM",
    "sunset": "4:14PM",
    "moonrise": "12:25PM",
    "moonset": "12:20AM",
    "moonPhase": 0.54,
    "moonPhaseDesc": "First Quarter",
    "iconName": "cw_first_qtr",
    "city": "Chicago",
    "latitude": 41.83,
    "longitude": -87.68,
```

```
    "utcTime": "2013-12-10T00:00:00.00-06:00"
    },
    ...per 8 giorni...
    {
    "sunrise": "7:17AM",
    "sunset": "4:15PM",
    "moonrise": "5:02PM",
    "moonset": "7:07AM",
    "moonPhase": 0.998,
    "moonPhaseDesc": "Full moon",
    "iconName": "cw_full_moon",
    "city": "Chicago",
    "latitude": 41.83,
    "longitude": -87.68,
    "utcTime": "2013-12-17T00:00:00.00-06:00"
    }
  ],
  "country": "United States",
  "state": "Illinois",
  "city": "Chicago",
  "latitude": 41.85003,
  "longitude": -87.65005,
  "timezone": -6
  },
  "feedCreation": "2013-12-10T15:44:37.120",
  "metric": true
}
```


la richiesta HTTP per avere la mappa è:

```
https://image.maps.api.here.com/mia/1.6/mapview?
co= “STATO”&
i=1&
app_id= “APP_ID”&
app_code= “APP_CODE”&
ci= “CITTA’”&
w= “DIMENSIONE_MAPPA”
```

un esempio di risposta è la seguente:

Adesso il server invia a Wikipedia API la richiesta per avere la storia della città cercata, attraverso la seguente richiesta HTTP:

```
https://it.wikipedia.org/w/api.php?
action= query&
format=json&
prop=extracts%7Cdescription&
titles= “CITTA’”&
exchars=800&
explaintext=1&
exsectionformat=plain
```

un esempio di risposta sarà:

```
{
  "batchcomplete": "",
  "warnings": {
     "extracts": {
        "*": "\"exlimit\" was too large for a whole article extracts request,
lowered to 1."
     }
  },
```

```
  "query": {
     "pages": {
        "2866894": {
           "pageid": 2866894,
           "ns": 0,
           "title": "Messina",
           "extract": "Messina (AFI: [mesˈsiːna] ; Missina in siciliano;
Μεσσήνη/Μεσσήνα in greco) è un comune italiano di 231 708 abitanti
capoluogo dell'omonima città metropolitana in Sicilia, nonché
tredicesimo comune italiano e terza città non capoluogo di regione più
popolosa d'Italia.\nSorge nei pressi dell'estrema punta nordorientale
della Sicilia (Capo Peloro) sullo Stretto che ne porta il nome. Il suo
porto, scalo...",
           "description": "Comune autogestito",
           "descriptionsource": "central"
        }
     }
  }
}
```


Questa insieme alle altre già elaborate dal Server verranno restituite al Client che si premurerà di rappresentarle usando l'interfaccia Homepage [Figura 2.6].

Queste fasi sono ben rappresentate nella figura 4.4.5.1.

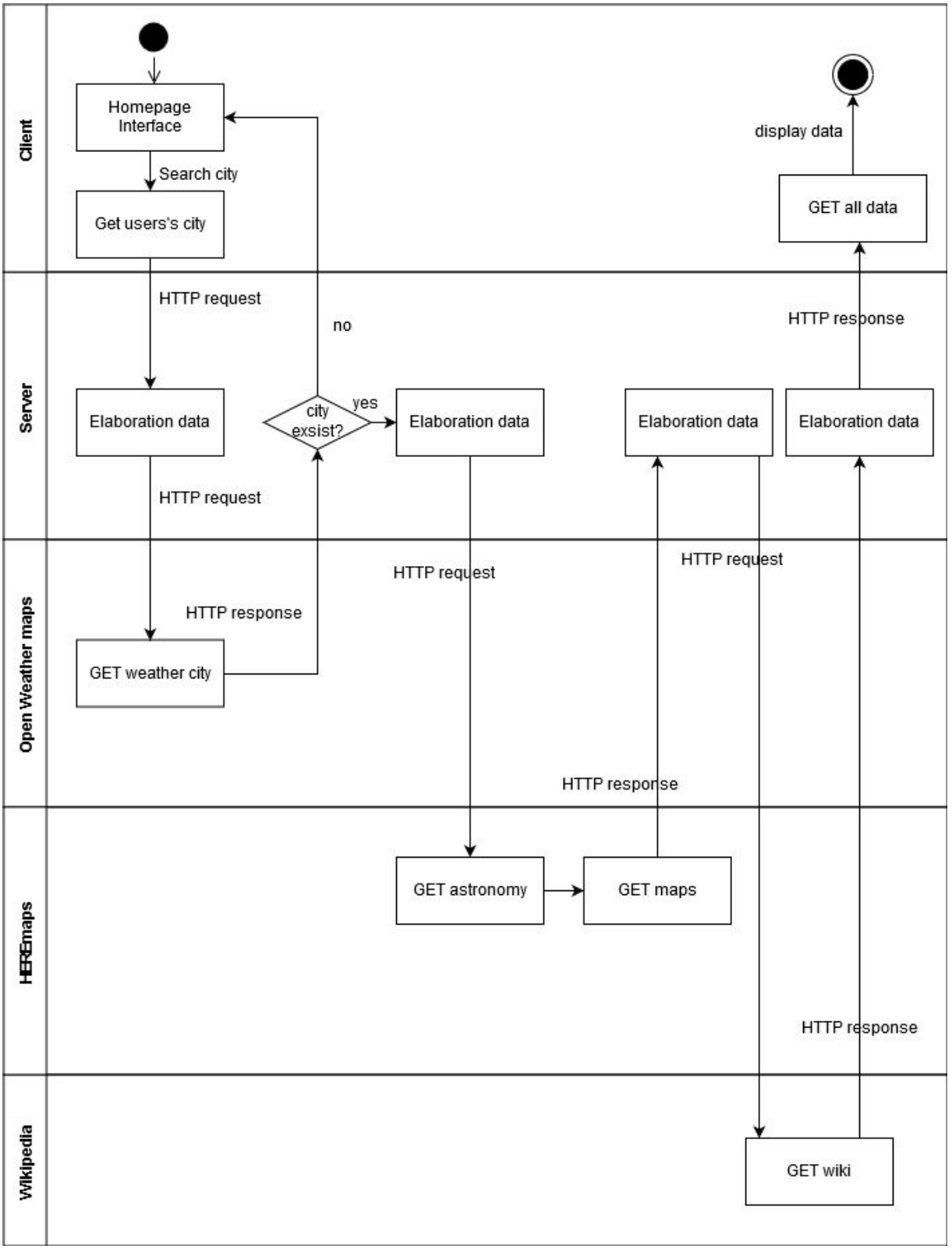

Figura 4.4.5.1 Activity Diagram della ricerca di una città o stato tramite le API di Wikipedia, HERE maps e Open Weather maps

4.4.6 Salva città

Una volta scelta la città utilizzando il pulsante “Cerca”, l’utente può decidere di scegliere e salvare quella città cliccando su “Salva”. Verrà inviato al server un json di questa forma:

```
{
        citta: “CITTA_SCELTA’”,
        id_progetto: “ID_PROGETTO_SELEZIONATO”,
        stato: “STATO_SCELTO”
}
```

dove ID_PROGETTO_SELEZIONATO è l’ID restituito quando viene creato un nuovo progetto o quello selezionato dal menù a tendina.
I dati vengono inviati al server che invia la seguente query al database:

```
INSERT INTO luoghi(citta, stato, id_progetto)
VALUES (“CITTA_SCELTA”, “STATO_SCELTO”,
“ID_PROGETTO_SELEZIONATO”)
```

Il database restituirà un messaggio che sarà rigirato al client. Nella figura 4.4.6.1 sono rappresentate in maniera chiara queste fasi tramite un activity diagram.

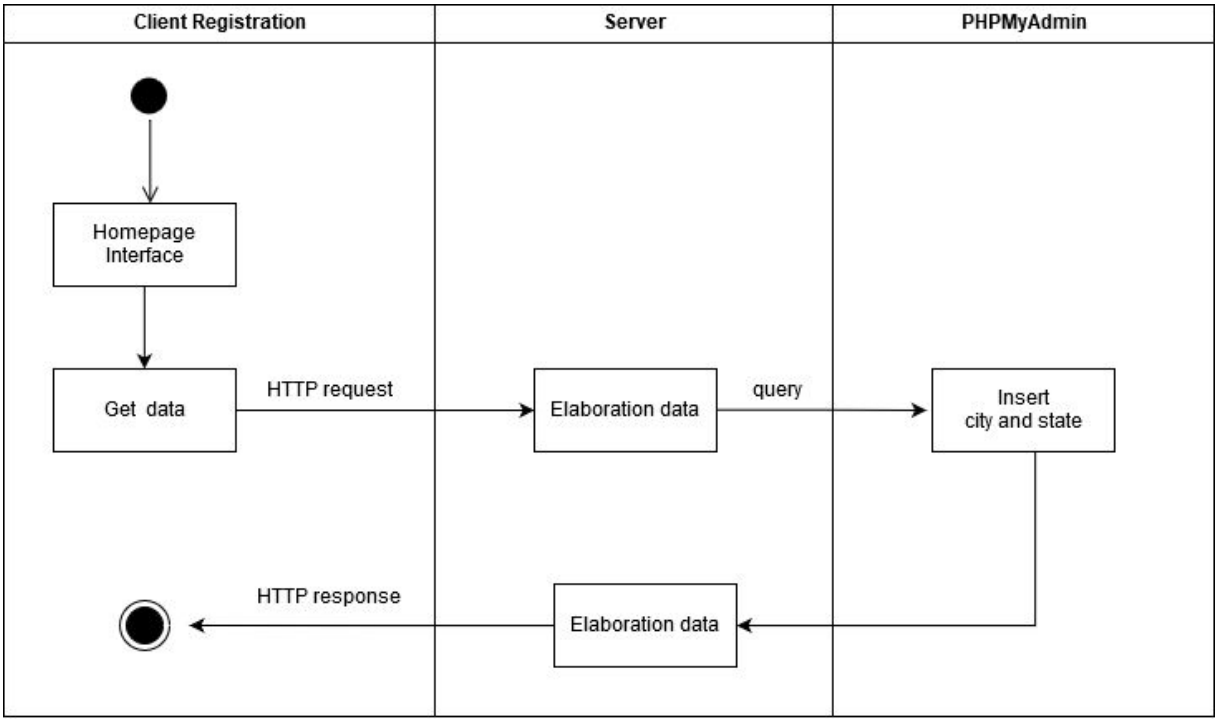

Figura 4.4.6.1 Activity diagram salvataggio citta e stato

4.4.7 Salva e visualizza note

Si possono associare al progetto corrente delle note salvandole e visualizzandole tramite l'interfaccia raffigurata in figura 2.8 e 2.9. Una volta inserito il nome della nota e il suo testo nei rispettivi campi, cliccando sul pulsante "Salva", si invia al server una richiesta HTTP con json:

```
{
        name: "NOME_NOTA",
        text: "TESTO_NOTA",
        directory: "ID_UTENTE + NOME_PROGETTO_ATTUALE"
}
```

dopo aver elaborato la richiesta, il server, invia una richiesta POST a Dropbox per capire se la nota è già stata creata:

```
https://api.dropboxapi.com/2/files/list_folder?
"path\": \"" + FOLDER_NAME+ "\",\"recursive\":
false,\"include_deleted\":
false,\"include_has_explicit_shared_members\": false&
Authorization= "TOKEN_DROPBOX"&
Content-Type=application/json
```

se esiste già, ritorna al client un messaggio notificando ciò, altrimenti inserisci la nota nel database tramite questa query:

```
INSERT INTO note (nome, id_progetto)
VALUES ("NOME_NOTA", "ID_PROGETTO")
```

e su dropbox tramite questa richiesta POST:

```
https://content.dropboxapi.com/2/files/upload?
\"path\": \"" + FOLDER_NAME+ "\&
Authorization="TOKEN_DROPBOX"&
Content-Type=application/octet-stream
```

Le risposte saranno elaborate dal server e ritornate sotto forma di messaggio al client che invierà a sua volta una richiesta HTTP al server per visualizzare tutte le note nel box. Il server invia una richiesta GET a dropbox:

```
https://api.dropboxapi.com/2/files/list_folder?
"path\": \"" + FOLDER_NAME+ "\",\"recursive\":
false,\"include_deleted\":
false,\"include_has_explicit_shared_members\": false&
Authorization= “TOKEN_DROPBOX”&
Content-Type=application/json
```

che restituirà tutte le note create in un file json che il client farà comparire nella box a destra [Figura 2.8].
L’activity diagram seguente mostrerà le fasi sopra descritte [Figura 4.4.7.1].

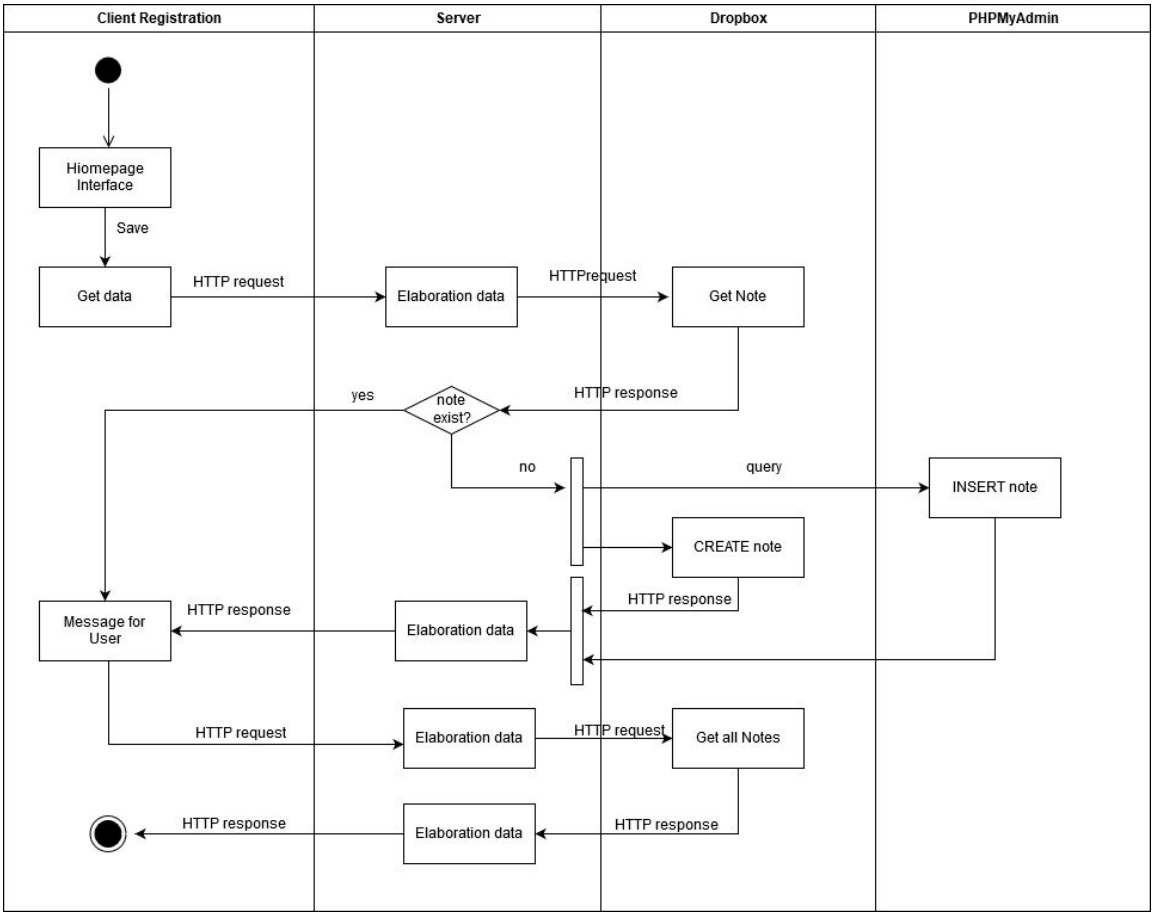

Figura 4.4.7.1 Activity Diagram per la visualizzazione e il salvataggio delle note.

4.4.8 Elimina note

Una volta visualizzata, una nota può essere eliminata utilizzando il pulsante “Elimina nota” [Figura 2.8]. Il client invierà il seguente file json:

```
{
   name: “ID_UTENTE + NOME_PROGETTO_ATTUALE + NOME_NOTA”
}
```

al server, che dopo aver elaborato i dati invia una richiesta POST a Dropbox:

```
https://api.dropboxapi.com/2/files/delete_v2?
paramiters={path: “NAME_PATH”}
Authorization= “DROPBOX_TOKEN”
Content-Type= “application/json”
```

dove NAME_PATH corrisponde al nome della nota inviato compreso di path: ID_UTENTE/NOME_PROGETTO_ATTUALE/NOME_NOTA.

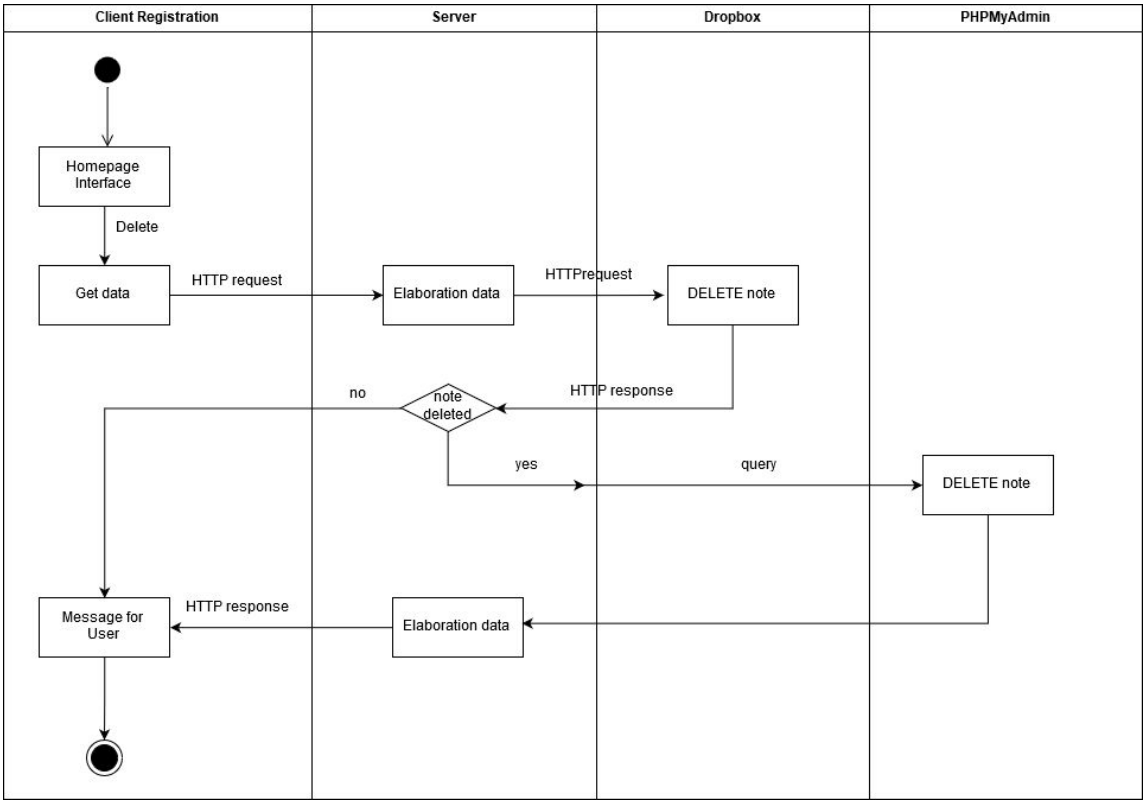

Figura 4.4.8.1 Activity Diagram eliminazione nota.

Se la nota non è stata trovata, viene inviato un messaggio all'utente comunicandoglielo, altrimenti viene inviata una query al database di eliminazione della nota memorizzata li. Alla fine viene inviato all'utente un messaggio di avvenuta cancellazione nota. L'activity diagram che rappresenta ciò è rappresentato in figura 4.4.8.1.

4.4.9 Visualizza salvataggi

Per visualizzare tutti i salvataggi dei progetti con le relative città [Figura 2.10] basta cliccare sul pulsante "Push" per inviare l'ID dell'utente loggato al server che invierà una query al database per avere tutti i dati.

```
SELECT DISTINCT progetti.id as id, progetti.nome as nomeProgetto, luoghi.citta as citta, luoghi.stato as stato, progetti.data as dataCreazione FROM progetti, luoghi, utenti WHERE progetti.id = luoghi.id_progetto AND progetti.id_utente = "ID_UTENTE_LOGGATO"
```

L'activity Diagram è rappresentato in figura 4.4.9.1.

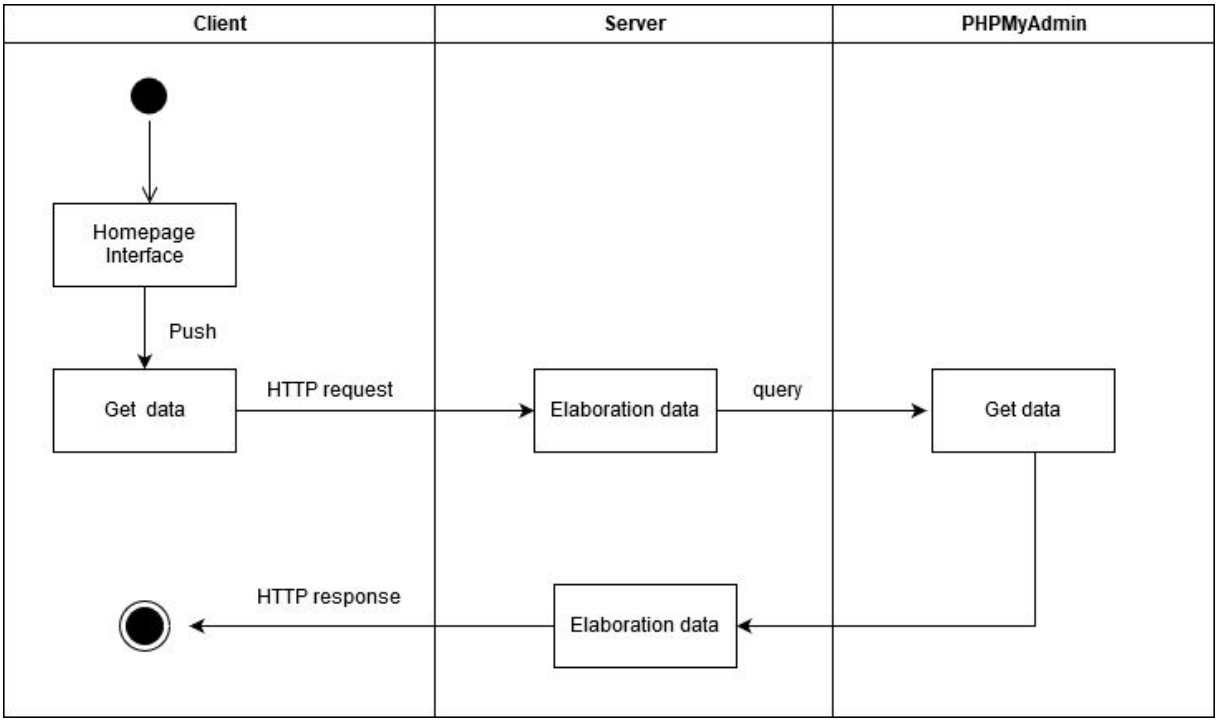

Figura 4.4.9.1 Activity Diagram ricerca dei progetti salvati dall'utente

4.4.10 Scarica PDF

Per scarica PDF si intende che l'utente ha la possibilità di scaricare il suo progetto con tutte le città salvate e le relative note. Tutto in formato PDF, quindi comodo da condividere su tutti i dispositivi o da stampare per portarlo in viaggio. Basta cliccare su "Visualizza e scarica" [Figura 2.11]. Il client invierà una richiesta al server contenente l'id dell'utente che contatterà il database inviando la query:

```
SELECT DISTINCT progetti.id as id, progetti.nome as nomeProgetto,
luoghi.citta as citta, luoghi.stato as stato, progetti.data as
dataCreazione FROM progetti, luoghi, utenti WHERE progetti.id =
luoghi.id_progetto AND progetti.id_utente = "ID_UTENTE"
```

dopo aver elaborato i dati, il server invia delle richieste ai servizi terze parti, per avere tutto ciò che gli serve per comporre il PDF ad ogni richiesta:

1. Invia una richiesta POST a Open Weather maps:

```
http://api.openweathermap.org/data/2.5/weather?
q= "CITTA'"&
APPID= "APP_ID"
```

2. Invia una richiesta POST a HEREmaps per l'astronomia e la mappa:

```
https://weather.cit.api.here.com/weather/1.0/report.json?
product=forecast_astronomy&
name= "CITTA'"
&app_id= "APP_ID"&
app_code= "APP_CODE"
&language=it
```

```
https://image.maps.api.here.com/mia/1.6/mapview?
co= "STATO"&
i=1&
app_id= "APP_ID"&
app_code= "APP_CODE"&
ci= "CITTA'"&
```

```
w= “DIMENSIONE_MAPPA”
```

3. Invia una richiesta POST a Wikipedia per la storia relativa alle città:

```
https://it.wikipedia.org/w/api.php?
action= query&
format=json&
prop=extracts%7Cdescription&
titles= “CITTA’”&
exchars=800&
explaintext=1&
exsectionformat=plain
```

Dopo aver creato il PDF, il server lo salverà sul database con la seguente query:

```
INSERT INTO pdf(idUnivoco, contenuto, id_utente)
VALUES (“CODICE_UNIVOCO_PDF”, “DOCUMENTO”,
“ID_UTENTE”)
```

E lo invierà al client.

Quest’ultimo può avere i PDF scaricati in precedenza con il possesso del codice identificativo univoco con cui è contrassegnato il PDF. Il server recupererà dal database il BLOB attraverso la query:

```
SELECT contenuto FROM pdf
WHERE idUnivoco= “CODICE_IDENTIFICATIVO”
```

Tutti gli step sono rappresentati dall’Activity Diagram in figura 4.4.10.1.

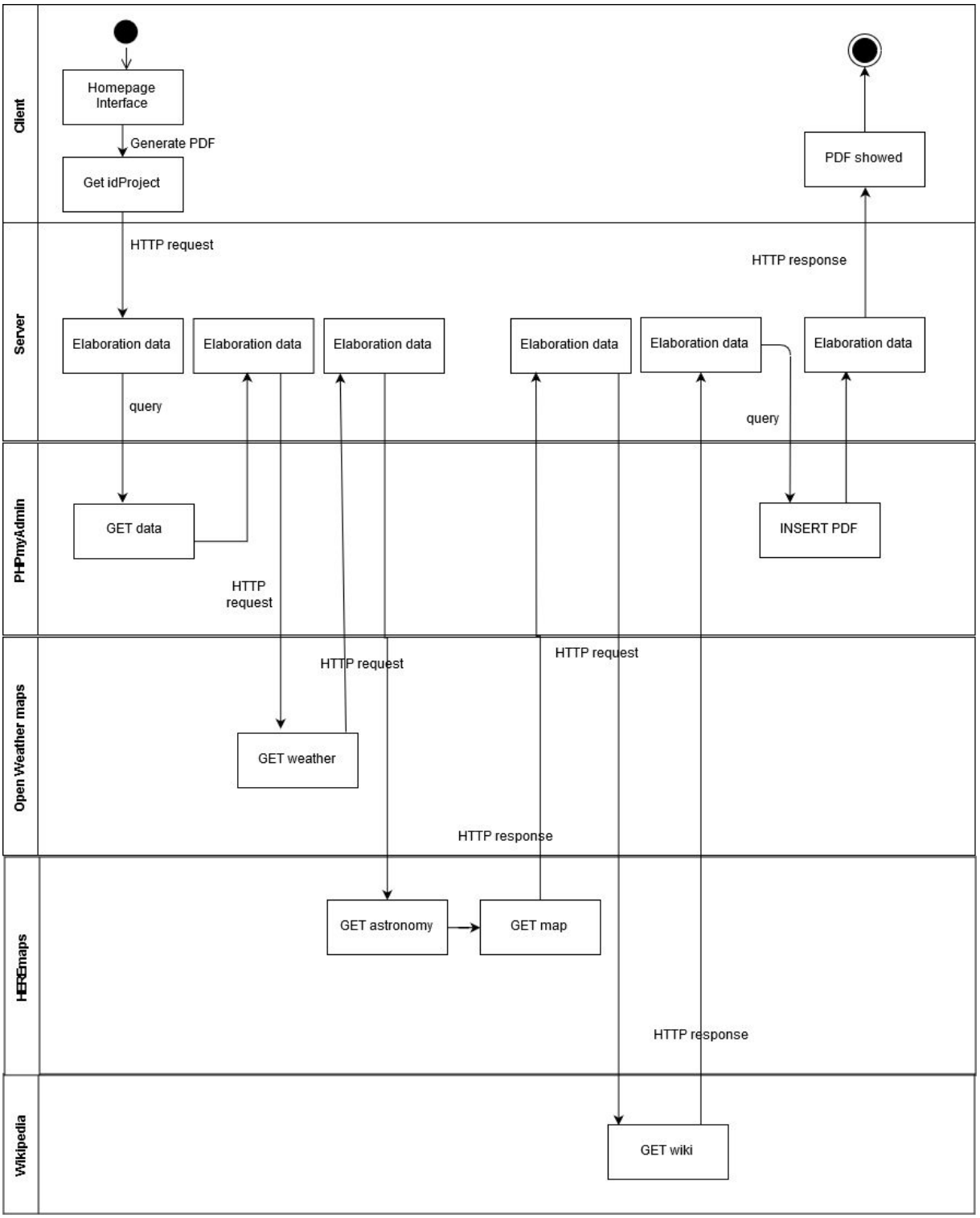

Figura 4.4.10.1 Activity diagram della creazione del PDF

4.4.11 Visualizza statistiche

Per vedere le statistiche globali, l'utente cliccherà su "Visualizza Statistiche Globali" [Figura 2.4], il client invierà al server una richiesta GET che verrà inoltrata al database tramite la query:

```
SELECT luoghi.stato as stato, count(luoghi.stato) as tot from luoghi
group by luoghi.stato
```

che restituirà il numero di città per stato salvate dagli utenti nei loro progetti. Ciò è descritto dall'Activity Diagram in figura 4.4.11.1

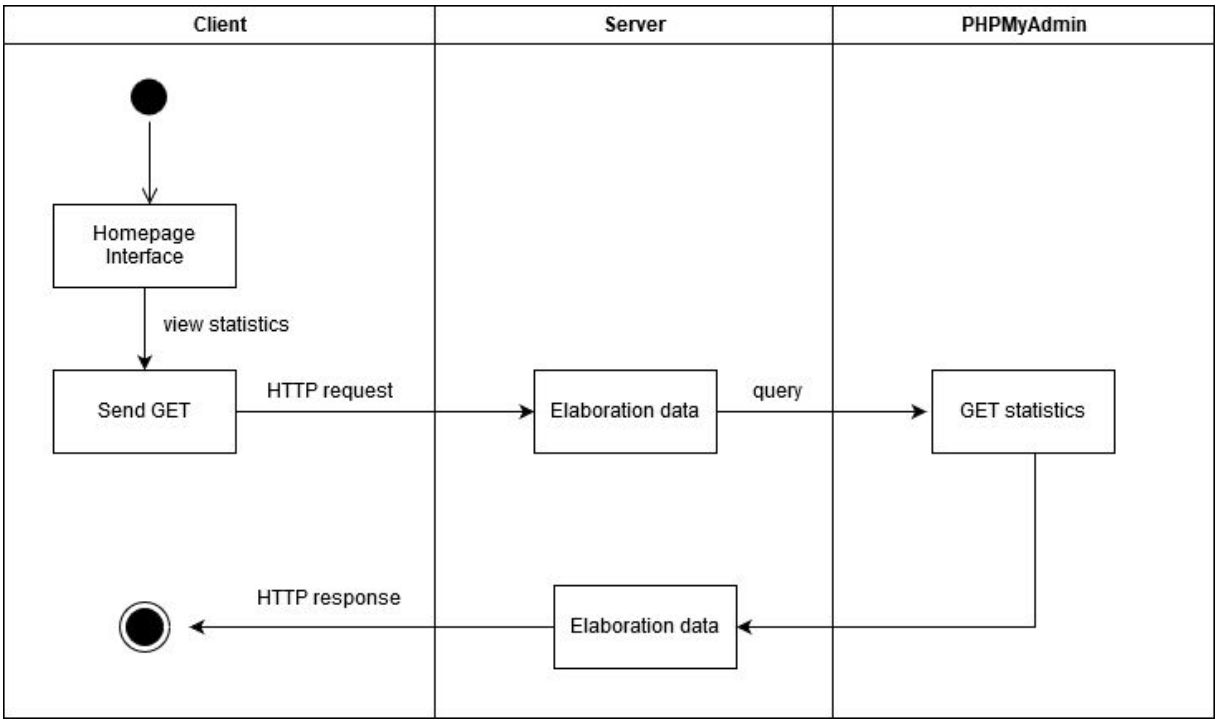

Figura 4.4.11.1 Activity Diagram raffigurazione statistiche globali.

4.5 Database Design

La progettazione del database è l'organizzazione dei dati secondo un modello di database. Il designer determina quali dati devono essere archiviati e in che modo gli elementi di dati sono correlati. E' stato scelto di utilizzare un database per fare statistiche, gestire gli utenti e i progetti registrati. Di seguito è rappresentato un modello Entità-Relazione che si distacca dal database utilizzato fisicamente. Rappresenta le relazioni tra i dati [Figura 4.5.1] grazie ai rombi e i dati stessi tramite i rettangoli, i pallini vuoti rappresentano gli attributi, i pallini pieni gli attributi con chiave primaria.

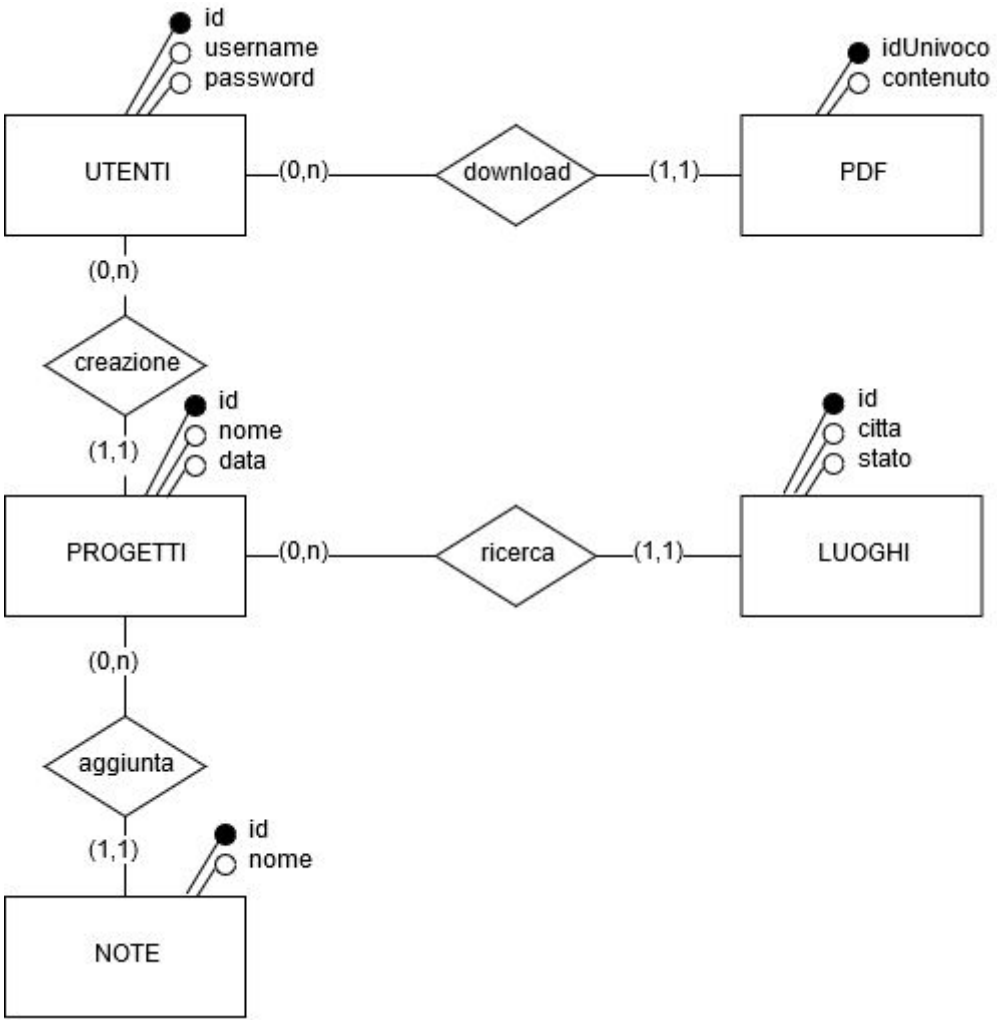

Figura 4.5.1 Modello E-R del Database

Una volta progettato il modello entità relazione, si passa a rappresentare i dati direttamente su database. Quello utilizzato è PHPMyAdmin. Il modello fisico mette in evidenza i vincoli tra gli attributi delle tabelle e il tipo di dato memorizzato in esse [Figura 4.5.2].

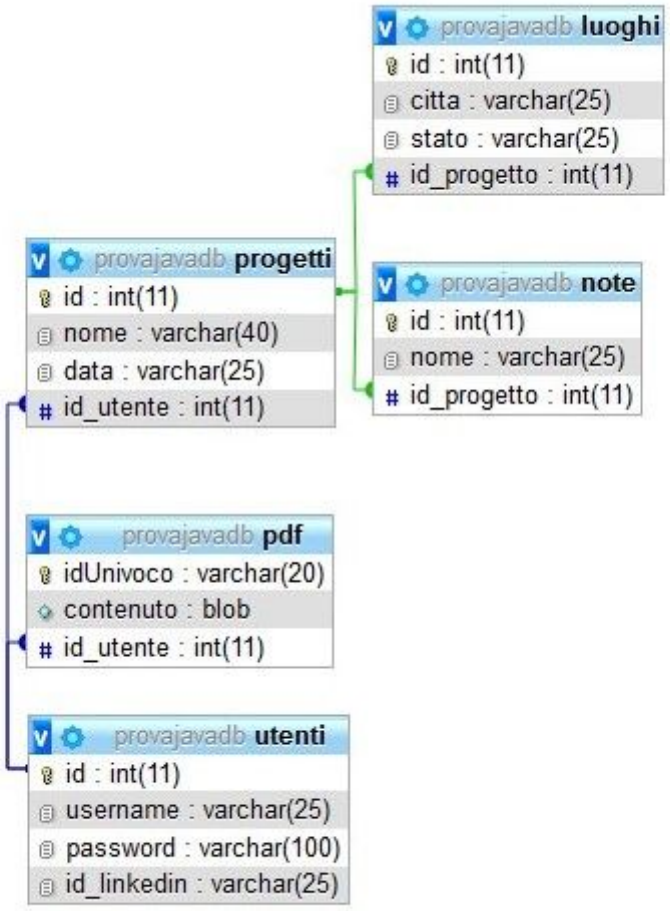

Figura 4.5.2 Modello fisico del Database