

***Youth Club***

***System Design Document***

***Versione 3.0***

****

***DATA: 23/01/2019***

**Coordinatore del progetto:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Francesco Truono | 0512104830 |

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Vincenzo Liguorino | 0512104692 |
| Aniello Petrosino | 0512104680 |
| Cristiana Elena Lazar | 0512104662 |

|  |  |
| --- | --- |
| **Scritto da:** | \* |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 18/01/2019 | 0.1 | Prima Stesura | Truono Francesco |
| 19/01/2019 | 0.2 | Aggiunta controllo degli accessi e sicurezza; boundary condition e condizioni globali del sistema | Liguorino Vincenzo |
| 19/01/2019 | 0.3 | Aggiunta divisione in sottosistemi | Truono Francesco |
| 19/01/2019 | 1.0 | Modifiche Dati persistenti:  diagramma EER, diagramma ER, struttura delle tabelle, schema logico | Petrosino Aniello, Lazar Cristiana Elena |
| 20/01/2019 | 1.1 | Revisioni Boundary Condition: aggiunta fallimenti del sistema | Truono Francesco, Liguorino Vincenzo |
| 20/01/2019 | 2.0 | Inserimento Use Case avvio, fallimento e terminazione | Liguorino Vincenzo |
| 23/01/2019 | 3.0 | Stesura finale con aggiustamento font e struttura | \* |

Sommario

[**1.** **Introduzione** 4](#_Toc536005781)

[1.1 Obiettivi del sistema 4](#_Toc536005782)

[1.2 Obiettivi di design 4](#_Toc536005783)

[1.3. Definizioni,acronimi e abbreviazioni 5](#_Toc536005786)

[1.4. Riferimenti 5](#_Toc536005787)

[**2** **Architettura del sistema proposto** 6](#_Toc536005788)

[2.1. Panoramica 6](#_Toc536005790)

[2.2. Decomposizione in sottosistemi 6](#_Toc536005791)

[2.3. Mapping Hardware/Software 8](#_Toc536005792)

[2.4. Gestione dati persistenti 9](#_Toc536005793)

[2.4.1. Struttura delle tabelle 10](#_Toc536005794)

[2.4.2. Tavola dei volumi 11](#_Toc536005795)

[2.4.3. Tavola delle operazioni 12](#_Toc536005796)

[2.4.4. Tavola degli accessi 12](#_Toc536005797)

[2.4.5. Controllo degli accessi 14](#_Toc536005798)

[2.5. Controllo del flusso globale del sistema 14](#_Toc536005799)

[2.6. Condizioni Boundary 15](#_Toc536005800)

[**3. Servizi dei sottosistemi** 17](#_Toc536005801)

[3.1. Gestione Ricerca 17](#_Toc536005803)

[3.2. Gestione Api 17](#_Toc536005804)

[3.3. Gestione Recensione 18](#_Toc536005805)

[3.4. Gestione Autenticazione 18](#_Toc536005806)

# **Introduzione**

## Obiettivi del sistema

Il sistema è stato progettato al fine di semplificare la ricerca dei locali usando e comparando i risultati di diversi servizi come google, yelp e foursquare che a causa delle stringhe immesse dagli utenti producono risultati diversi e incompleti.  
Per questo motivo l’applicazione si pone di identificare i tag e l’opportuna identificazione geografica per ottimizzare i risultati ed includere più locali possibili.  
Inoltre essendo un servizio principalmente di consultazione che serve quando una persona si trova per strada.  
Il modello delle recensioni invece è interno al sistema.

## Obiettivi di design

**Adaptability**: Il sistema ottimizza le ricerche memorizzando i luoghi, dove ad ognuno di essi è associato una lista di locali. Se una ricerca è già stata effettuata da un utente in un lasso di tempo relativamente breve i tempi di risposta si ridurranno alla sola geolocalizzazione del luogo e alla lettura della lista dei locali.

**Usability**: Essendo un applicativo mobile deve essere semplice da comprendere e veloce da utilizzare quindi non sono presenti molte opzioni o form lunghi da compilare dato che non rispecchiano i principi di usabilità in ambito mobile.

In particolar modo c’è da tener conto del principio di iterazione con una solo mano, in cui si immagine che l’utente utilizza l’app con il pollice , ciò significa che la posizione dei widget deve essere strategica.

Un altro aspetto importante riguarda l’input, dove appunto il numero di tocchi deve essere ridotto al minimo, nel nostro specifico caso si riduce alla selezione delle categorie e al input della ricerca.

Eventualmente fossero necessari caricamenti, l’utente deve essere necessariamente informato dello status di avanzamento.

**Reliability :** Risulta fondamentale la capacità di evitare errori , per questo motivo l’esecuzione della ricerca con le informazioni relative ai locali deve essere vista come un’unità **atomica** che quindi premette solo la possibilità di riuscita o fallimento. Inoltre il servizio essendo un servizio web deve essere in grado di esaudire tutte le richieste dei client.

**Performance:** Il sistema ha come obiettivi di fornire risposte rapide e corrette agli utenti, tenendo presente della capacità di adattamento precedentemente descritta, le performance variano anche dalla “conoscenza” dell’app per questo motivo le performance hanno una dipendenza dai servizi REST usati.

L’obiettivo medio è di fornire una risposta in meno di 3 secondi.

**Supportability:** Il concetto di supportability varia tra client e server, nel primo caso si riferisce esclusivamente al supporto tra le varie versioni di android , in particolar modo si cerca di ricoprire almeno il 90% dei dispositivi android invece per quanto riguarda il lato server il concetto di portabilità è intrinseco in java anche se rimane legato a tomcat

**Maintainability:** (Sottopunto di supportability) Per semplificare il riadattamento a modifiche dovute alla tecnologia o al cambiamento dei servizi REST usati il software sarà suddiviso in componenti e moduli assestanti che interagiscono tra di loro al fine del corretto funzionamento generale.



## Definizioni,acronimi e abbreviazioni

|  |  |
| --- | --- |
| **Sigla/Termine** | **Definizione** |
| Servizi REST | Representational State Transfer è un tipo di architettura software per i sistemi distribuiti |
| App | Applicativo mobile |

## Riferimenti

Il progetto essendo un **Greenfield Project** significa chelo sviluppo comincia da zero, non esiste nessun sistema a priori e i requisiti sono ottenuti dall’utente finale e dal cliente.

Nonostante non esista un’applicazione del genere, esistono servizi integrati in altre app o siti web che permettono di effettuare ricerche dei locali.

Come ad esempio **maps** o **tripadvisor** che permettono di cercare un locale o di ottenere una lista di punti d’interesse in un determinato luogo, per questo motivo questi due sistemi sono stati presi come punto di riferimento.

Per la stesura del seguente documento sono stati presi come punti di riferimento i seguenti testi:

- B. Bruegge, A.H. Dutoit, Object Oriented Software Engineering, Using UML, Patterns and Java Prentice Hall

-http://elearning.informatica.unisa.it/el-platform/mod/folder/view.php?id=9164, System Design Documentation

# **Architettura del sistema proposto**



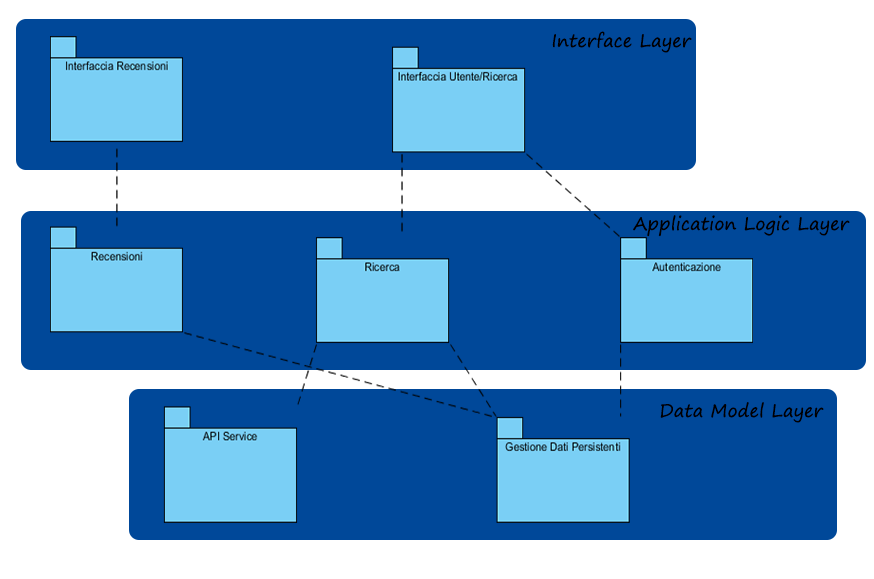
## Panoramica

L’architettura scelta per il sistema è ibrida in quando sfrutta un modello client-server per quando riguarda le richieste inviate ed elaborate dal server, ma il server viene suddiviso in 3 layer Model-View-Control.

Il lato client si occupa di eseguire le richieste e decodificare e mostrare opportunamente le risposte nell’applicazione.

Il lato server presenta invece la logica e la conoscenza del dominio applicativo, il concetto di view diventa più generico e si occupa solo della formattazione dei dati in risposta alle richieste del client.

## Decomposizione in sottosistemi

****

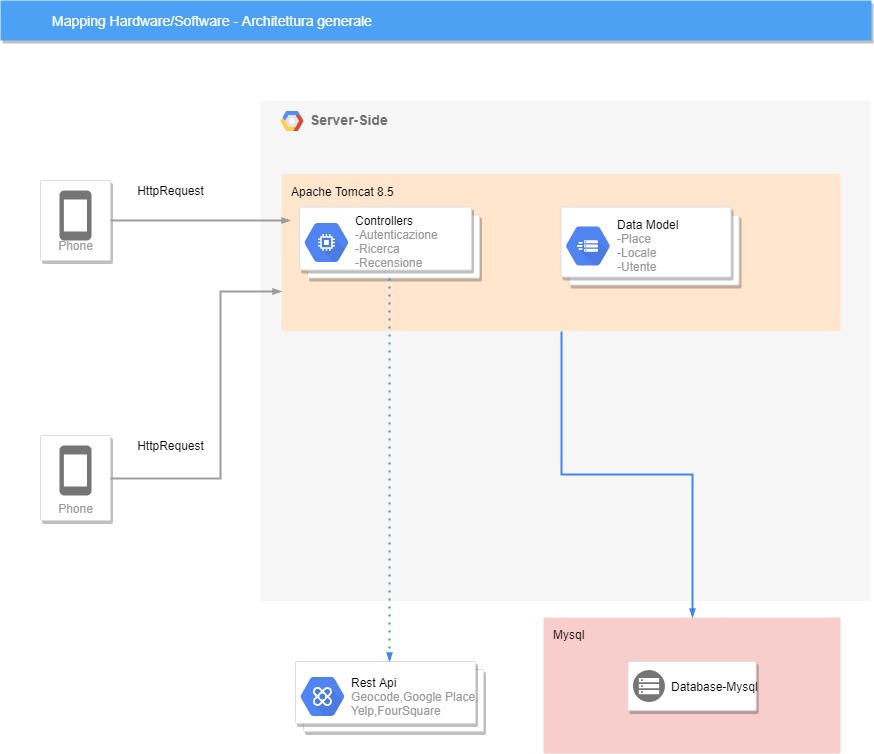
I Sottosistemi individuati sono: gestione ricerca, gestione delle API, gestione recensioni, gestione autenticazione.

|  |
| --- |
| **Interface Layer** |
| Interfaccia recensioni | Raggruppa tutte le interfacce utente per la stesura e lettura di una recensione |
| Interfaccia Utente/Ricerca | Raggruppa tutte le interfacce che raggruppano i tipi di ricerca e il profilo utente |

|  |  |
| --- | --- |
| **Application Logic Layer** |  |
| Recensioni | * Permette la modifica di una recensione * Permette di aggiungere una recensione |
| Ricerca | * Permette di effettuare una ricerca dato in input un nome di un locale * Permette di effettuare una ricerca dato un luogo e delle categorie |
| Autenticazione | * Permette la registrazione al servizio (non richiede iterazione con l’utente) * Permette l’autenticazione al servizio (non richiede iterazione con l’utente) |

|  |
| --- |
| **Data Model Layer** |
| Api Service | * Si occupa del recupero dati dai webservice * Si occupa della geolocalizzazione del luogo |
| Gestione data persistenti | * Gestisce il luogo * Gestisce il locale e le recensioni * Gestisce l’utente |

## Mapping Hardware/Software

****

Il sistema software è un ibrido tra Client-Server e MVC.

È possibile notare che la parte di **Presentation Layer** è spostata sul lato client che si occupa esclusivamente di inviare richieste ed elaborare graficamente ed opportunamente i dati ricevuti.

Il modello di client utilizzato è Fat-Client dato che il client è responsabile di gestire i dati e le connessioni con il server.

La comunicazione tra Client e Server avviene tramite protocollo HTTP (Hypertext Transfer Protocol), un protocollo di trasferimento di ipertesti utilizzato per trasmettere l’interazione tra Client e Server attraverso un meccanismo di request/response.

Il web-server scelto è Tomcat dove appunto è presente la parte logica dell’applicazione e la gestione dei dati, sulla stessa macchina risiede anche il database MySQL con la quale interagisce mediante JDBC.

Al di fuori del sistema sono presenti servizi necessari per l’identificazione del luogo e la ricerca dei locali, si tratta di End-Point messi a disposizione da varie società che si collocano nel ambito dei webservice che permettano **l’interoperabilità** tra diversi elaboratori in un contesto distribuito.

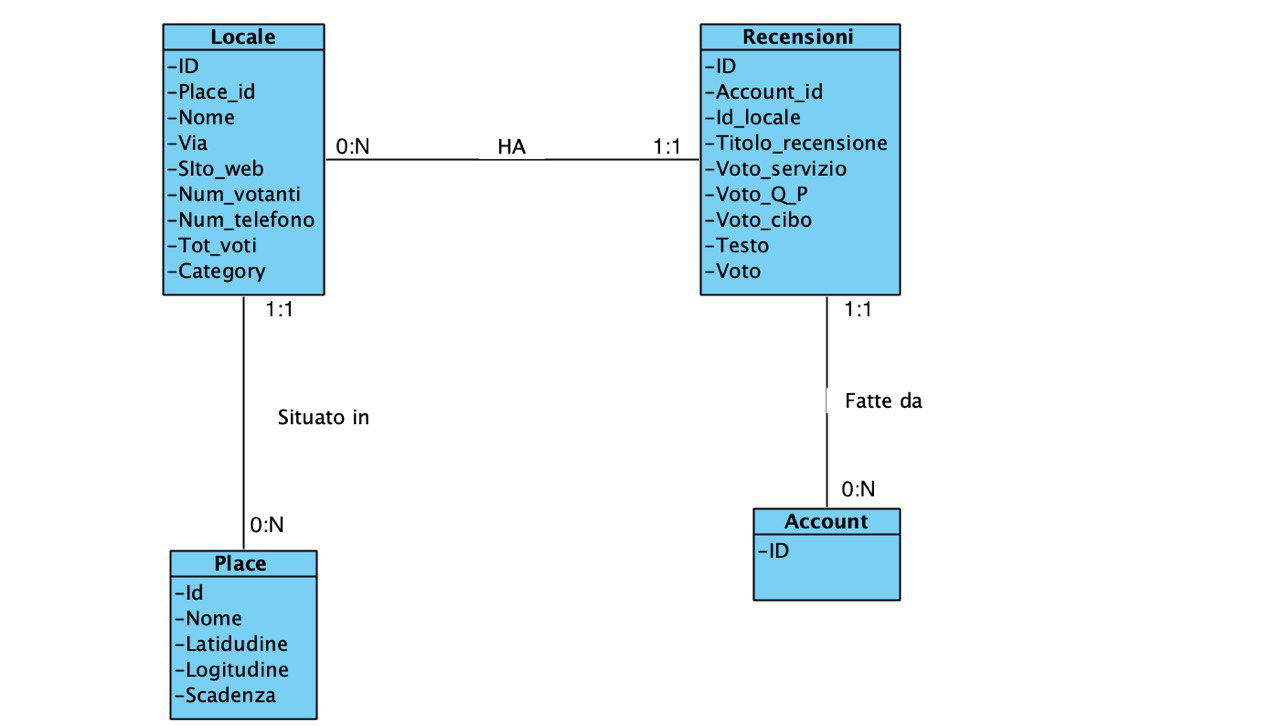
Il modello Model-View-Control è presente nell’organizzazione del progetto server-side dove appunto ogni sottosistema implementa determinate operazioni in maniera indipendente, il layer di View si occupa solo di organizzare la risposta da inviare. Il sottosistema del model opera con i dati sfruttando la comunicazione con mysql e i controller valutano le richieste proveniente dall’esterno ed elaborano i dati da notificare alla view sfruttando l’iterazione con le api esterne e il sottosistema di data-model.

Le richieste inviate dai client, sul lato server sono di tipo stateful, ciò significa che viene mantenuta una sessione per ogni utente correttamente autenticato.

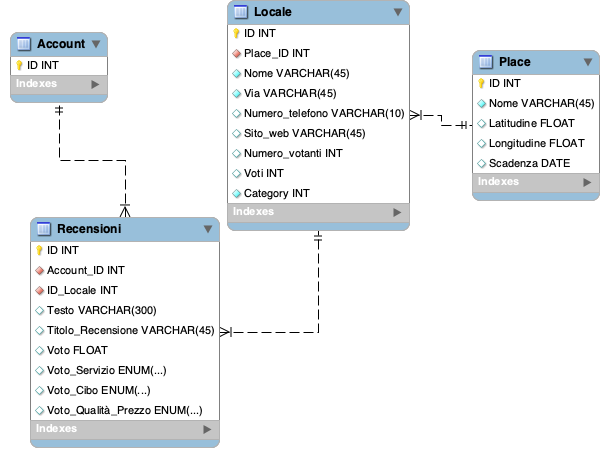
## Gestione dati persistenti

La struttura del nostro database non è molto complicata perché essendo che i dati che noi prendiamo dalle api si basano su latitudine e longitudine abbiamo potuto risparmiare molte tabelle che ci sarebbero servite per tenere traccia dei luoghi che ora stanno tutti nella entità place poi abbiamo l’entità locale che come dice parole tiene traccia di tutti i locali ed è collegata a place da una relazione 1:N dove però un locale deve avere un place però non il contrario essendo che potrebbe capitare che in quella zona non siano trovati locali. Per ogni locale potremmo avere una recensione quindi un'altra relazione 1:N perché un locale può non avere recensioni se per esempio è aperto da poco o non sia mai stato cercato ma una recensione deve avere per forza un riscontro e infine abbiamo Account dove salviamo ogni nostro utente

**Class Diagram**



**Schema logico**

****

### Struttura delle tabelle

|  |  |  |
| --- | --- | --- |
| **ACCOUNT (contiene informazioni sull’account)** | | |
| **Campo** | **Vincoli** | **Tipo** |
| ID | Lunghezza massima: 30 caratteri;  Primary key;  Not null | Varchar |

|  |  |  |
| --- | --- | --- |
| **PLACE (contiene informazioni sul luogo)** | | |
| **Campo** | **Vincoli** | **Tipo** |
| ID | Primary key;  Not null | Int |
| Nome | Lunghezza massima: 45 caratteri;  Not null | Varchar |
| Latitudine | Not null | Float |
| Longitudine | Not null | Float |
| Scadenza | Not null | Datetime |

|  |  |  |
| --- | --- | --- |
| **RECENSIONI (contiene informazioni sulle recensioni)** | | |
| **Campo** | **Vincoli** | **Tipo** |
| ID | Primary key;  Not null | Int |
| Account\_ID | Lunghezza massima: 30 caratteri;  Foreign key Account;  Not null | Varchar |
| ID\_Locale | Foreign key Locale;  Not null | Int |
| Testo | Lunghezza massima: 300 caratteri; | Varchar |
| Titolo\_Recensione | Lunghezza massima: 45 caratteri;  Not null | Varchar |
| Voto | Not null | Float |
| Voto\_Servizio | Not null | Int |
| Voto\_Qualità\_Prezzo | Not null | Int |
| Voto\_Cibo | Not null | Int |

|  |  |  |  |
| --- | --- | --- | --- |
| **LOCALE (contiene informazioni sul locale)** | | | |
| **Campo** | **Vincoli** | **Tipo** | |
| ID | Primary key;  Not null | | Int |
| ID\_Place | Foreign key Place;  Not null | | Int |
| Category | Valore Massimo:15  Not null | | TinyInt |
| Nome | Lunghezza massima: 30 caratteri;  Not null | | Varchar |
| Via | Lunghezza massima: 45 caratteri;  Not null | | Varchar |
| Numero\_telefono | Lunghezza massima: 10 caratteri; | | Varchar |
| Sito\_web | Lunghezza massima: 45 caratteri; | | Varchar |
| Numero\_votanti |  | | Int |
| Tot\_voti |  | | Int |

### Tavola dei volumi

L’applicativo Youth Club essendo un sistema ancora non presente sul mercato è difficile dare dei valori veritieri alle entità presenti nel nostro DB però essendo che sarà presente su tutto il territorio nazionale abbiamo ipotizzato di avere 1000 utenti e sempre sulla base di ipotesi abbiamo pensato che ogni utente cerchi 3 locali diversi però che non in ogni caso scriva una recensione. Un altro fattore principale dell’applicativo sono i luoghi noi abbiamo stimato che tutti i locali cercati stiano in 800 zone diverse essendo che in molti casi i locali si trovano sempre tutti vicini e solo raramente sono fuori mano.

|  |  |  |
| --- | --- | --- |
| **Concetto** | **Costrutto** | **Volume** |
| Account | Entità | 1000 |
| Locale | Entità | 3000 |
| Recensione | Entità | 2500 |
| Place | Entità | 800 |
| Situato in | Relazione | 20 |
| Ha | Relazione | 500 |
| Fatte da | Relazione | 1700 |

### Tavola delle operazioni

|  |  |  |
| --- | --- | --- |
| **Operazione** | **Tipo** | **Frequenza** |
| Login | Interattiva | 45000/anno |
| Scrittura Recensione | Interattiva | 20000/anno |
| Modifica Recensione | Interattiva | 100/anno |
| Lettura voto | Batch | 30000/anno |
| Ritrovamento locale  (in base all’id del luogo) | Batch | 40000/anno |
| Aggiunta locale | Batch | 25000/anno |

### Tavola degli accessi

**Login**

|  |  |  |  |
| --- | --- | --- | --- |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo di accessi** |
| Account | E | 1 | L |

**Totale Accessi:** 1 x 45000=45000 accessi/anno

**Scrittura Recensione**

|  |  |  |  |
| --- | --- | --- | --- |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo di accessi** |
| Account | E | 1 | L |
| Fatte da | R | 1 | L |
| Recensioni | E | 1 | S |

**Totale Accessi:** 1 x 45000 + 1 x 20000 + 20000 x 2= 105000 accessi/anno

**Modifica Recensione**

|  |  |  |  |
| --- | --- | --- | --- |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo di accessi** |
| Account | E | 1 | L |
| Fatte da | R | 1 | L |
| Recensioni | E | 1 | L |
| Recensioni | E | 1 | S |

**Totale Accessi:** 1 x 45000 + 1 x 100 + 1 x 100 + 100 x 2= 45400 accessi/anno

**Lettura Voto**

|  |  |  |  |
| --- | --- | --- | --- |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo di accessi** |
| Account | E | 1 | L |
| Fatte da | R | 1 | L |
| Recensioni | E | 1 | L |

**Totale Accessi:** 1 x 45000 + 1 x 30000 + 1 x 30000= 105000 accessi/anno

**Ritrovamento Locale (in base all’id del luogo)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo di accessi** |
| Place | E | 1 | L |
| Locale | E | 1 | L |

**Totale Accessi:** 1 x 40000= 40000 accessi/anno

**Aggiunta locale**

|  |  |  |  |
| --- | --- | --- | --- |
| **Concetto** | **Costrutto** | **Accessi** | **Tipo di accessi** |
| Place | E | 1 | S |
| Locale | E | 1 | S |
| Locale | E | 1 | L |

**Totale Accessi:** 5\*25000=125.000

### Controllo degli accessi

L’applicativo mobile YouthClub permette l’accesso a un solo tipo di persona cioè l’utente.

Ad ogni utente sono associati un id, utilizzato come credenziale per essere riconosciuto dal sistema, e una lista di recensioni dove possiamo risalire alle recensioni già effettuate.

Ogniqualvolta un utente abbia intenzione di utilizzare una funzionalità contenuta all’interno dell’applicazione, e quindi di avviare una nuova sessione, dovrà effettuare l’accesso all’app che in automatico risalirà al suo id tramite il codice IMEE del cellulare.

La sessione di utilizzo verrà terminata alla chiusura dell’app.

Il sistema, di conseguenza, può essere utilizzato da qualsiasi utente, le cui funzionalità disponibili sono descritte all’interno della matrice degli accessi.

**Oggetto**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attori** | Gestione | | | |
| **Ricerca** | **Mappa** | **Lista Locali** | **Recensioni** |
| Utente | Visualizzazione;  Inserimento valori chiave;  Aggiornamento;  Possibilità di scelta;  Annullamento | Visualizzazione;  Annullamento; | Visualizzazione;  Interazione; | Visualizzazione;  Modifica;  Completamento;  Feedback utente;  Annullamento |

## Controllo del flusso globale del sistema

Il controllo del flusso del software è garantito attraverso la realizzazione di classi Java che fanno da ricevitore per gli eventi rispondendo alle attivazioni dei client.  
A generare le richieste è il client, tale richiesta genera un evento e classe preposta a gestirlo si occupa di inizializzare le richieste e di inoltrarle per lo svolgimento dell’operazione che il client desidera portare a compimento. Una volta terminata l’operazione verrà prodotto un risultato.  
Il sistema software YouthClub è ibrido in quando sfrutta un modello Client-Server per quando riguarda le richieste inviate ed elaborate dal server , ma il server viene suddiviso in 3 layer Model-View-Control.

## Condizioni Boundary

**Avvio del sistema**Distinguiamo l’avvio dell’app “YouthClub” in lato server e in lato client:  
−Server: l’avvio avviene nel momento in cui viene avviato il server Tomcat dalla macchina su cui  
risiede;  
−Client: l’avvio avviene ogni volta che l’Utente accede all’app. Quando ciò avviene viene presentata un’interfaccia che permette all’utente di accedere alle funzionalità offerte dal sistema.

**Terminazione del sistema**−Client: la terminazione avviene al momento della chiusura dell’app da parte dello stesso. Quando ciò avviene vengono interrotte tutte le operazioni non concluse o pendenti che il Client stava tentando di eseguire , per mantenere la consistenza dei dati si concluderanno inevitabilmente con un fallimento.

**Fallimento del sistema**  
−Server: Il fallimento del sistema può avvenire per molteplici ragioni elencate di seguito:

1. Il sistema non conosce il luogo ma in quel momento nessuna delle 3 api esterne può fornire un risultato per down temporanei del servizio , ovviamente le probabilità che nessuno dei 3 servizi funzionano sono estremamente esigue.
2. Il sistema conosce il luogo ma MySQL smette di funzionare.
3. Altre problematiche potrebbero essere causate dalla rottura di qualche componente hardware e/o a sbalzi di corrente che causerebbero a una mancanza del servizio in maniera temporanea.

**Use Case Avvio**

|  |  |
| --- | --- |
| **Nome Use Case:** | Avvio |
| **Condizioni di entrata:** | Il ServerAdministrator accede al server |
| **Flusso di eventi:** | 1. Dopo aver effettuato correttamente l’accesso viene eseguito dal ServerAdministrator lo start-Up del server 2. Il sistema riceve la richiesta e attiva tutti i servizi necessari al corretto funzionamento del sistema. Fatto ciò sarà possibile usare il sistema. |
| **Condizioni di uscita:** | * Il Server è attivo e riceve una richiesta da parte del Client |

**Use Case Terminazione**

|  |  |
| --- | --- |
| **Nome Use Case:** | Terminazione |
| **Condizioni di entrata:** | Il ServerAdministrator esegue l’ operazione per arrestare il server |
| **Flusso di eventi:** | 1. Viene eseguito dal ServerAdministrator lo ShutDown del server 2. Il server riceve la richiesta e disattiva tutti i servizi usati dal sistema, inoltre esegue tutte le operazioni necessarie per una corretta terminazione |
| **Condizioni di uscita:** | * Il Server si arresta |

**Use Case Fallimento**

|  |  |
| --- | --- |
| **Nome Use Case:** | Failure luogo |
| **Condizioni di entrata:** | L’utente fa la richiesta |
| **Flusso di eventi:** | 1. Il sistema controlla se il luogo esiste già 2. Se non esiste effettua richiesta esterna al sistema |
| **Condizioni di uscita:** | * Risposta negativa da parte delle Api |

|  |  |
| --- | --- |
| **Nome Use Case:** | Failure database |
| **Condizioni di entrata:** | L’utente fa la richiesta |
| **Flusso di eventi:** | 1. L’utente ricerca il luogo |
| **Condizioni di uscita:** | * MySQL smette di funzionare ed emette un eccezione |

# **3. Servizi dei sottosistemi**



## Gestione Ricerca

|  |  |
| --- | --- |
| **Sottosistema** | **Gestione Ricerca** |
| Descrizione | Il sottosistema permette di effettuare le operazioni di ricerca mediante nome e una serie di categorie oppure mediante un luogo |
| **Servizi offerti** | |
| **Servizio** | **Descrizione** |
| Ricerca per nome | Questo servizio permette di ricerca in base a un nome di un locale. |
| Ricerca per luogo | Questo servizio permette di ottenere una lista di locali in base al luogo e alle categorie selezionate |

## Gestione Api

Si tratta di un sottosistema che si occupa dell’iterazione con i servizi esterni.

|  |  |
| --- | --- |
| **Sottosistema** | **Gestione API** |
| Descrizione | Il sottosistema permette di effettuare le operazioni di ricerca dei locali non noti e di identificare un luogo |
| **Servizi offerti** | |
| **Servizio** | **Descrizione** |
| Geocode | Questo servizio offre la possibilità di identificare latitudine , longitudine , stato , regione e comune. |
| Reverse Geocode | Questo servizio è usato per ottenere informazioni riguardo al luogo partendo da latitudine e longitudine. Risulta necessario nel caso venisse usato il gps lato client |
| Google Place | Si occupa di autenticarsi con google e trovare i locali dato un luogo. |
| Yelp | Si occupa dell’autenticazione su determinati endpoint ed ottenere la lista dei locali |
| FourSquare | Si occupa di autenticarsi con foursquare e trovare i locali dato un luogo |

## Gestione Recensione

|  |  |
| --- | --- |
| **Sottosistema** | **Gestione Recensioni** |
| Descrizione | Il sottosistema si occupa di recuperare le recensioni di un locale , di un utente , di modificare una recensione o aggiungerne una. |
| **Servizi offerti** | |
| **Servizio** | **Descrizione** |
| Lista recensione | Recupera la lista delle recensione in base all’utente o al id di un locale. |
| Modifica recensione | Modifica una recensione in base al utente e al locale |
| Aggiungi recensione | Aggiunge una recensione a un locale. |

## Gestione Autenticazione

|  |  |
| --- | --- |
| **Sottosistema** | **Gestione Autenticazione** |
| Descrizione | Il sottosistema si occupa di autenticare l’utente e mantenere la sessione |
| **Servizi offerti** | |
| **Servizio** | **Descrizione** |
| login | Si occupa di eseguire il login con il sistema , specificando un id univoco. |