Relazione progetto ingegneria del software

AlertCity

Belli giacomo Angelo Cirillo Davide Lupo Stefano Dugo

**Descrizione del problema**

AlertCity nasce con l’idea di creare una maggior comunicazione tra cittadini comuni ed enti pubblici, introducendo la possibilità di automatizzare il processo di segnalazione delle problematiche che si possono trovare all’interno di una qualsiasi città

Il sistema di segnalazione che al giorno d’oggi (almeno per quello che riguarda l’Italia) è presente si svolge tramite lunghe pratiche comunali che non permettono una corretta individuazione delle problematiche e allo stesso tempo una tempestiva risposta ai problemi.

AlertCity gestirà tutto questo processo in modo da rendere questo procedimento molto più flessibile, dinamico e in maniera molto più chiara di quanto lo sia ora.

**Requisiti funzionali**

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrizione** | **MoSCoW** |
| 0 | Il programma dovrà permettere agli utenti di autenticarsi dentro al sistema | Must have |
| 1 | Il programma dovrà riconoscere la tipologia di utente che utilizza l’applicazione | Must have |
| 2 | Il programma dovrà permettere ai cittadini di creare le segnalazioni | Must have |
| 3 | Il programma dovrà mostrare contenuti relativi al proprio comune di appartenenza | Must have |
| 4 | Il programma dovrà permettere la creazione degli interventi in risposta alle segnalazioni da parte dei comuni | Must have |
| 5 | Il programma dovrà riconoscere lo stato attuale dei lavori relativi all’intervento | Must have |
| 6 | Il programma dovrà permettere la creazione di  informative da parte dei comuni | Must have |
| 7 | Il programma dovrà eliminare in maniera automatica i report degli utenti quando questi risultino risolti | Should have |
| 8 | Il programma dovrà permettere il corretto store dei dati degli utenti | Should have |
| 9 | Il programma dovrà permettere di visualizzare i log degli utenti | Should have |
| 10 | Il programma dovrà inviare una notifica quando un intervento o una informativa verrà creata | Should have |

**Requisiti funzionali**

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrizione** | **MoSCow** |
| 0 | Il programma dovrà essere sviluppato in ionic | Must have |
| 1 | Il programma dovrà risultare fluido per gli utilizzatori | Should have |
| 2 | Il programma dovrà incentivare i comuni ad aderire a questa iniziativa | Should have |
| 3 | Il database dovrà prevedere molto spazio per l’archiviazione | Must have |

**Analisi**

**Firebase**

Per gestire il corretto store degli account e dei file che si andranno a creare in seguito a segnalazioni, interventi e informative è necessario adoperare un database.

Il database che è stato scelto per lo svolgimento di questo progetto è Firebase



Firebase permetterà lo storing degli account dei cittadini che si iscrivono all’applicazione, manterrà permettendo così la distinzione degli account in modo tale da far riconoscere all’applicazione quale utente è connesso al sistema e quali permessi possiede.

AlertCity registrerà ogni nuovo utente come cittadino mentre per gli account quali (comune ed ente) si avvarrà della creazione manuale da parte dell’Admin che genererà questi account speciali con permessi speciali e distribuirà le chiavi di accesso a comuni ed enti che vorranno partecipare all’iniziativa.

Firebase si occuperò inoltre di immagazzinare i dati delle segnalazioni, degli interventi e delle informative permettendo così il corretto funzionamento dell’applicazione.

**Linguaggio**

Per la scrittura del codice abbiamo deciso di adoperare Ionic-4 il quale essendo un linguaggio ibrido permetterà di sviluppare una applicazione che potrà facilmente essere adoperata sia come web application sia come mobile application.

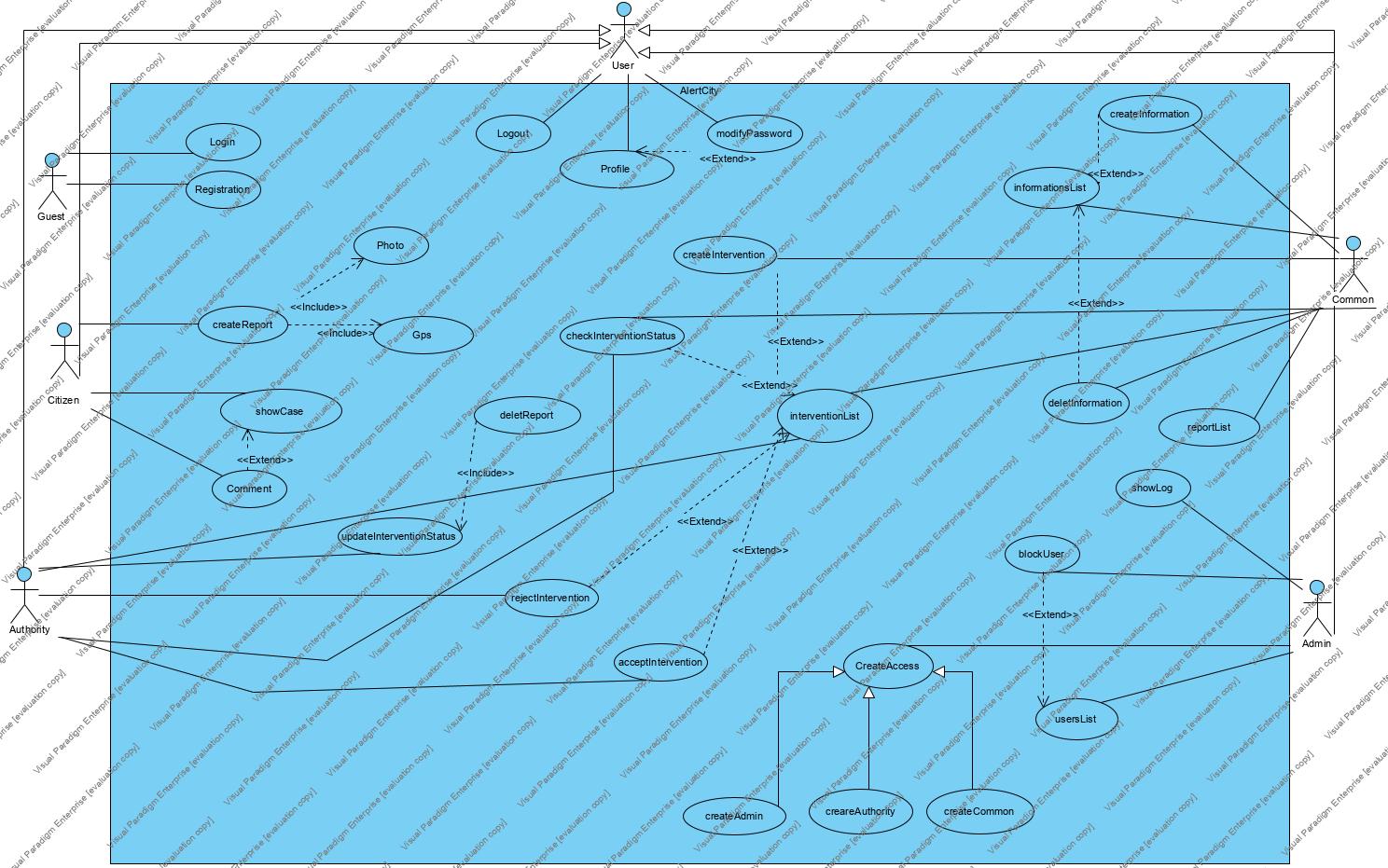
Sarà inoltre facile collegare firebase all’applicazione avvalendoci del relativo modulo firebase.

Per facilità applicative e logiche abbiamo deciso di non consentire agli utenti che non si siano registrati di pubblicare contenuti all’interno dell’applicazione in modo tale da eliminare eventuali segnalazioni che non siamo chiare e che possano compromettere l’intero sistema.

Sono stati utili allo sviluppo di AlertCity anche l’utilizzo di specifiche librerie che consentono di geolocalizzare le segnalazioni e allegare relative foto in modo tale da avere per i comuni una visione più chiara della problematica.

**UML**

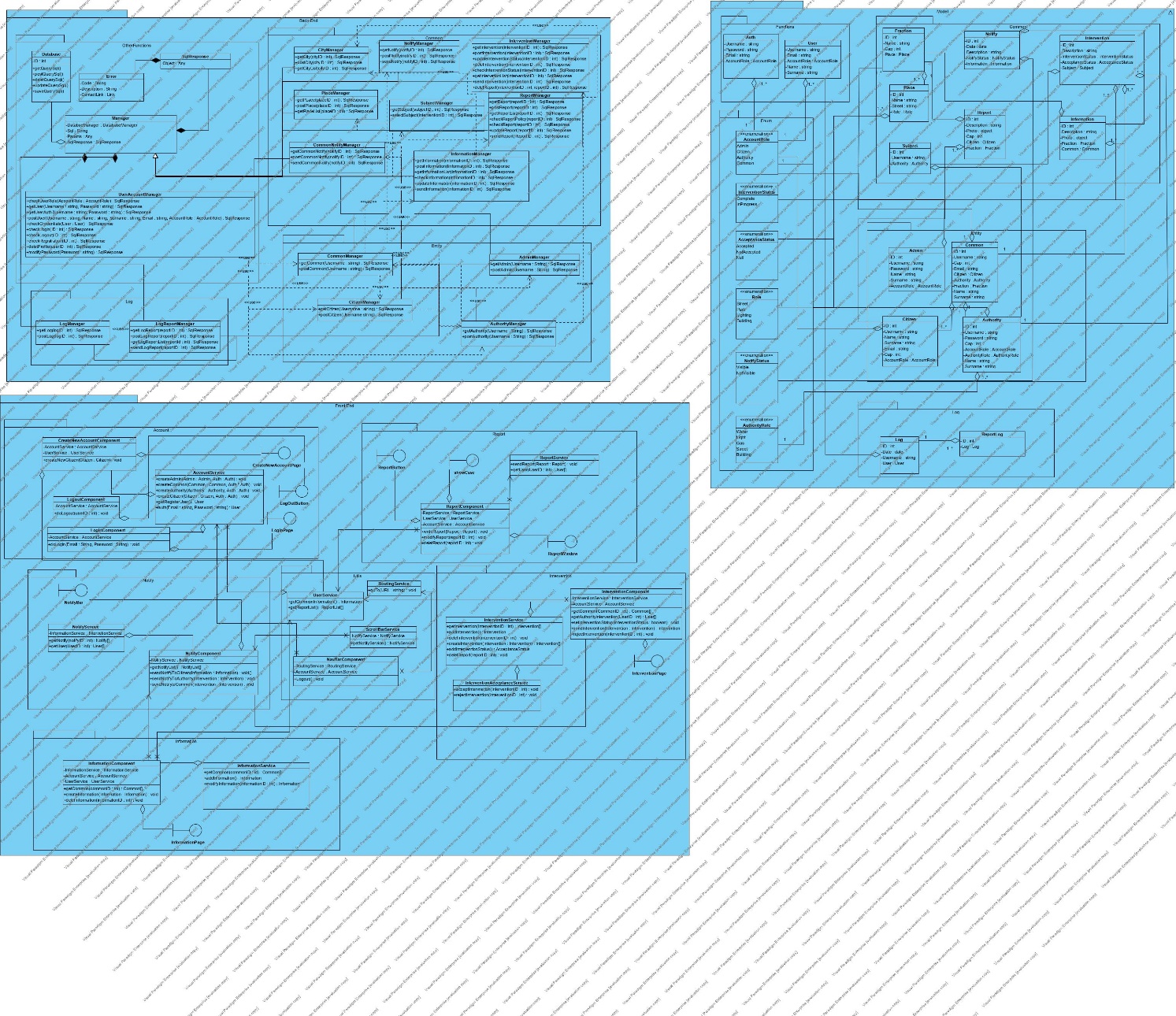
**Use Case Diagram**

****

Tutte le informazioni riguardanti lo Use Case Diagram (specifica dei casi d’uso e flow event) sono visibili all’interno del file AlertCity.vpp

Link GitHub <https://github.com/GiacomoBelli/AlertCity-IDS>

**Class Diagram**

****

Per una maggiore facilità di comprensione e una stesura più chiara e comprensibile è stato suddiviso il class diagram secondo il pattern MVC.

Possiamo trovare tre sub-system i quali individuano il model il back-end e il front-end dell’applicazione.

Abbiamo optato per questa modalità di progettazione in quanto essendo ionic un linguaggio basato su questo pattern risulta essere molto più chiara la lettura da parte di altri utenti e l’effettiva realizzazione del codice stesso

Il model è suddiviso anche esso in sottosistemi quali:

* Common
* Enum
* Entity
* Log

Common racchiude tutte le entità che si occupano della gestione dei report, degli interventi e delle informative più eventuali funzionalità che sono in fase di progettazione che saranno elencate alla fine.

Enum serve a specificare informazioni aggiuntive riguardo le nostre entità, queste informazioni permetteranno infatti di effettuare tutti i controlli che sono associati ad ogni entità.

Entity racchiude gli attori del nostro progetto (Common Citizen Admin Authority) Questi attori sono direttamente connessi con il Sub-System Common definendo come in figura le proprie composizioni e associazioni.

Log racchiude tutte le operazioni che possono essere sfruttate dall’Admin in modo tale da racchiudere uno storico degli accessi.

Il Back-End è il cuore della nostra applicazione ed è interconnesso con il model e il Front-End, questo ha il compito di gestire tramite i manager tutte le funzionalità che si possono fare all’interno di AlertCity.

Ad ogni entità è associato un manager che sfrutta il database per trasformare tutte le chiamate applicative in chiamate SQL che saranno propriamente lette ed eseguite dal database.

Esso si occupa inoltre di tutti i controlli associati alle chiamate evitando così di incorrere in errori logici.

Ogni chiamata effettuata all’interno di AlertCity viene interpretata dal suo manager specifico permettendo così il monitoraggio delle operazioni che avvengono.

Il Front-End si occupa della vista dell’utente e permette la corretta visualizzazione della applicazione.

Il Front-End racchiude tutte le pagine che possono essere visualizzate dagli utenti e i relativi format per creare segnalazioni interventi e informative, le istanze comunicano tra loro mediate l’uso dei component e dei service. Ad ogni component sono associati uno o più service i quali si occupano della trasmissione delle informazioni verso il Back-End per ogni funzionalità è previsto un send ovvero un’operazione di passaggio dei dati da Front a Back.

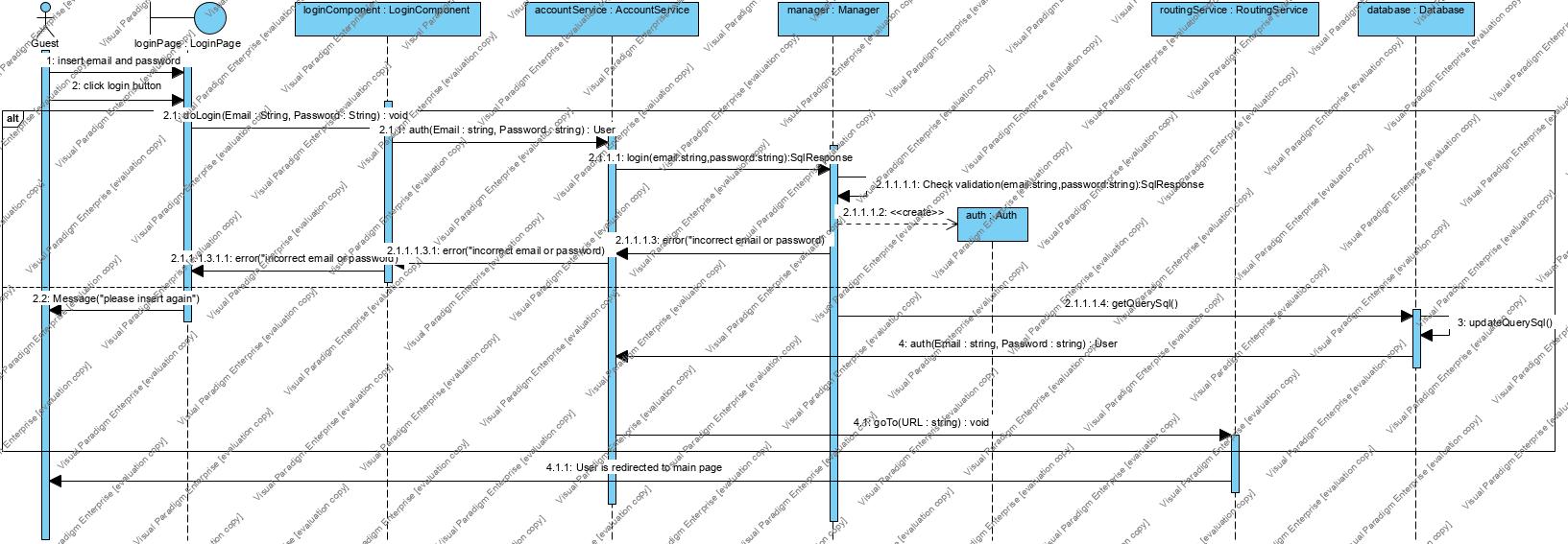
Per visualizzare in maniera più specifica i metodi e le associazioni tra le varie entità aprire il file AlertCity.vpp dove saranno disponibili maggiori informazioni.

**Sequence Diagram**

Questi diagrammi spiegano in maniera molto più dettagliata tutte le chiamate e le risposte che vengono effettuate all’interno dell’applicazione.

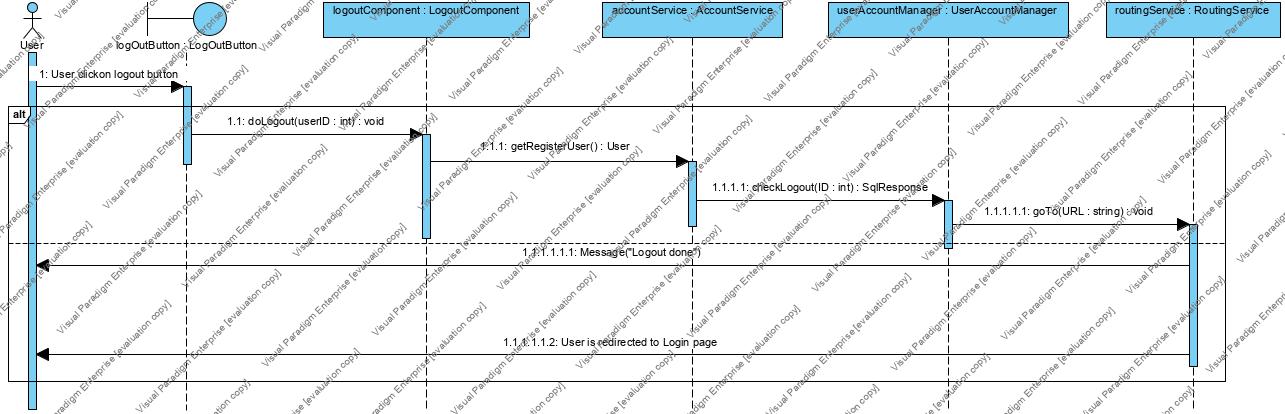
Tutti i file si possono trovare dentro il file AlertCity.vpp e per descrivere queste funzionalità inseriremo qui sotto le funzioni più importanti che si possono effettuare dentro all’applicazione.

**Login**



Il nostro attore principale è il Guest, ovvero un utente non ancora identificato dal sistema, inserendo email e password e cliccando sul pulsante submit nella pagina di Login viene effettuata una chiamata al login component contenendo email e password digitate, viene effettuato un controllo auth() per poi passare al manager, il quale converte la chiamata in una SQL e interroga il database per verificare la correttezza e l’integrità dei dati. Viene effettuato un controllo ortografico e una corrispondenza nel database, se questo controllo va a buon fine viene creata un auth ovvero una autenticazione, il database aggiorna la lista degli utenti attivi e tramite il report service(che permette di muoversi tra le varie page) il guest che ora è diventato un utente viene indirizzato alla main page. Nel caso una corrispondenza non vada a buon fine il sistema restituirà diversi tipi di errori in base al tipo di incombenza riscontrata.

**Logout**

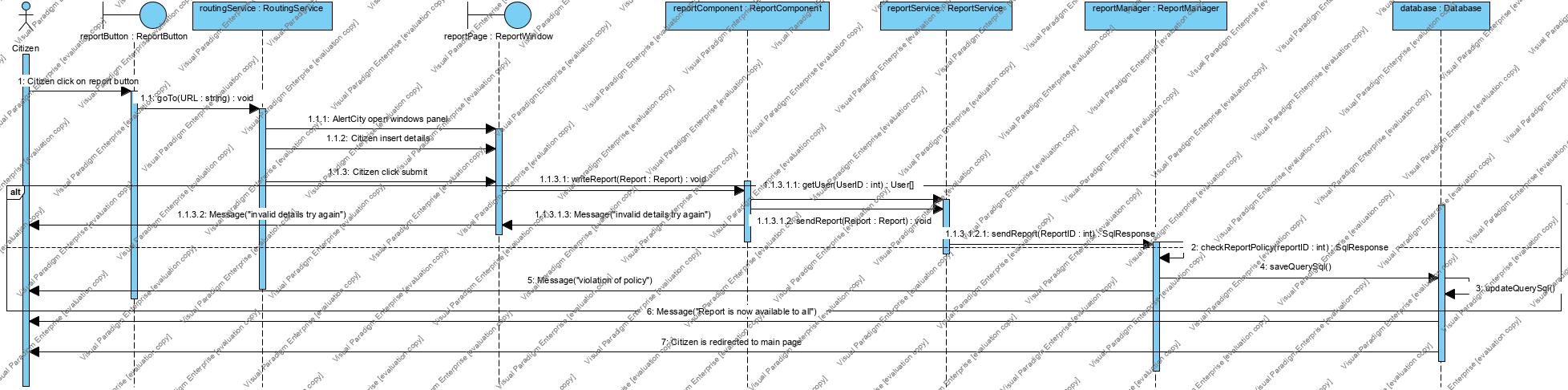


Il logout descrive la procedura per uscire dall’applicazione, l’attore principale di questo procedimento è l’User.

Ogni User può effettuare il logout tramite il bottone “logout” il quale invia al component il metodo doLogout() con i campi email e password. Il service ottiene l’utente registrato e trasforma, tramite il manager la chiamata in una chiamata SQL al database.

Una volta effettuato il logout l’utente sarà riportato ala schermata di login con un messaggio da parte del sistema.

**CreateReport**



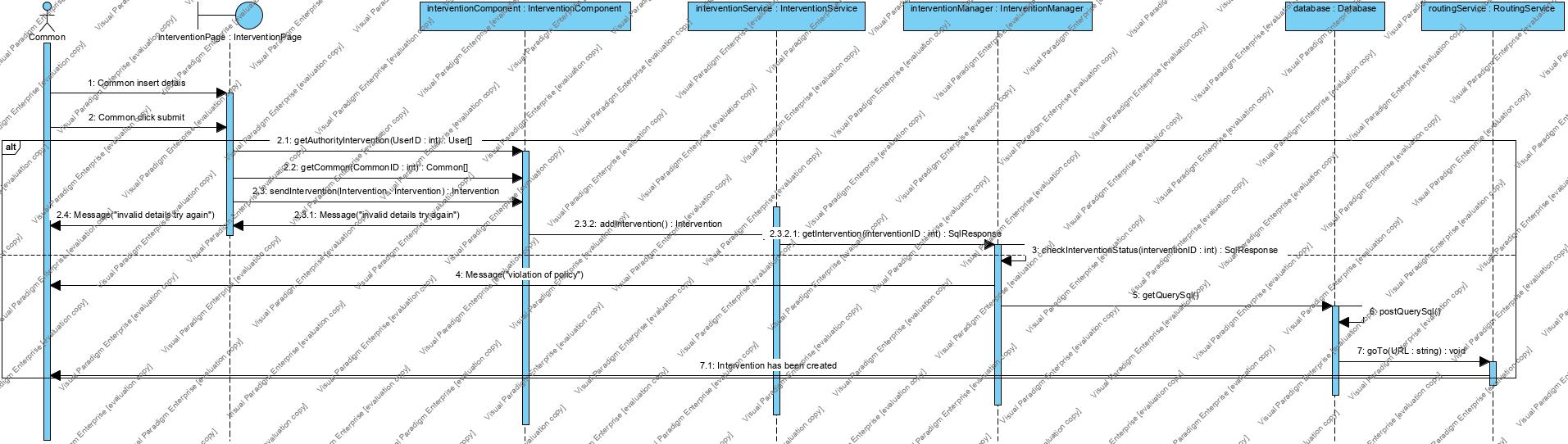
Questa sequenza descrive in maniera dettagliata come avviene la creazione delle segnalazioni da parte dei cittadini.

Una volta che il cittadino ha cliccato sul bottone “crea segnalazione” verrà aperto da parte dell’applicazione un layout da compilare.

Una volta inserita la descrizione specificato il tipo di problema, allegata una foto e inserite le coordinate(manuali o GPS) si potrà inviare la segnalazione. Il component provvederà a inviare il contenuto del report avvalendosi del service il quale tramite la comunicazione con il manager trasformerà la chiamata in SQL permettendo così il controllo della validità ortografica e politica della segnalazione. Tramite il metodo checkReportPolicy() AlertCity farà in modo che nessun contenuto che non sia consono alla segnalazione potrà essere pubblicato(foto non inerenti, o di natura pornografica ecc ecc).

Il database salverà il report e lo aggiungerà alla report list ovvero la lista di tutte le segnalazioni dei cittadini.

**CreateIntervention**



Questo diagramma descrive la creazione degli interventi da parte dei comuni, il comune inserirà all’ interno della pagina di intervento i dettagli del lavoro da svolgere, specificando così l’autorità associata e la descrizione dell’intervento.

Una volta confermato il component otterrà il comune e l’autorità di riferimento e aggiungerà uno stato all’intervento che sarà settato di default come (“Null”) questo passerà così al service e al manager che trasformerà il tuto in una chiamata al database controllando lo stato dell’intervento e la validità di compilazione dei campi. Se tutto andrà a buon fine verrà salvato all’interno del database e aggiunto alla lista degli interventi, verrà inoltre inoltrata una notifica all’autorità che è stata selezionata, infine il comune sarà riportato tramite il routingService alla pagina degli interventi inziale.

Nel caso si presentino degli errori, ortografici o logici, il sistema restituirà un messaggio di errore.

**ER Diagram**

****

Questo tipo di diagramma serve a specificare come è strutturato il database al suo interno e le sue relazioni.

Ogni entità possiede un ID identificativo le relazioni possono essere di natura uno a molti come ad esempio tra comune e cittadino o molti a molti come nel nostro esempio tra comune e fraction.

Questo tipo di diagramma ci è stato molto utile per capire al meglio la tipologia di relazione che si ha tra e varie entità.

Firebase da la possibilità di gestire il database in real-time ovvero apportando modifiche senza pregiudicare il funzionamento dell’applicazione.

L’ER diagram ha inoltre il compito di individuare quali campi delle entità dovranno essere classificati come univoci (unici).

**Test**

La fase di testing è stata insieme alla progettazione del class diagram la parte più corposa e complicata da eseguire. il testing di una applicazione infatti presenta una moltitudini di possibilità e di errori di tipo logico, di coding o di performance.

Prima di tutti abbiamo effettuato dei test per verificare il corretto funzionamento dell’autenticazione, questa infatti rappresenta un elemento chiave dell’applicazione e deve essere chiaro il ruolo dei nostri attori all’interno di AlertCity.

La seconda fase di testing è stata effettuata sul corretto funzionamento del database e di tutto quello che riguarda l’immagazzinamento dei dati, degli account e delle chiamate SQL.

Abbiamo verificato che gli identificativi dei nuovi report siano correttamente archiviati nella report list senza sovrapposizioni o perdita di dati.

In secondo luogo abbiamo verificato il corretto funzionamento dello storing di firebase per archiviare immagini e file.

Un’altra fase sicuramente saliente è stata quella del test coding ovvero abbiamo verificato la corretta stesura del codice in modo tale che fosse chiaro, che le chiamate fossero il più semplice possibile e che i controlli fosse precisi.

**Conclusioni**

Questa tipologia di progetto è stata senza ombra di dubbio una delle cose più complete che abbiamo potuto osservare pianificare e sviluppare in questi 3 anni, è stato incredibile osservare come la fase di progettazione di un qualsiasi software o applicazione è qualcosa di molto complesso che richiede la partecipazione collettiva del team. E’ stato molto costruttivo soprattutto capire la metodologia procedurale con cui si sviluppa davvero un software, partire da una idea fino a concretizzarla in qualcosa di realizzabile, ci siamo trovati molte volte a dover cambiare intere entità, relazioni e logica dell’applicazione e sembrava come se ad ogni problema risolto si generassero automaticamente nuove problematiche che richiedevano la medesima attenzione per essere risolte.

**Futures**

Durante lo sviluppo di AlertCity abbiamo identificato delle possibili applicazioni future che abbiamo provveduto ad allegare nello sviluppo UML ma che non sono ancora state introdotte in maniera definitiva nell’applicazione, queste due implementazioni sono rispettivamente il LogReport e il Notify. La prima darà in futuro la possibilità all’ admin di visualizzare i log degli utenti direttamente dall’applicazione senza dover entrare in maniera manuale dentro al database, mentre la seconda darà agli enti la possibilità di ricevere delle notifiche ogni qual volta un intervento verrà creato.