POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Progetto e sviluppo di un'app mobile ad impatto sociale: il caso di Wher

Relatore
Prof. Giovanni Malnati

Correlatore
Dott. Andrea Valenzano - Wher

Candidato
Fausto Savarino
231916

Ottobre 2019

Indice

Abstract

La percezione di sicurezza in strada delle donne è diversa rispetto alla percezione maschile (ISTAT, 2018) [1]. Questo pone la necessità di una priorità di intervento rispetto ad un problema sentito che limita la libertà personale delle donne.

L'oggetto di studio di questa tesi è la progettazione e lo sviluppo di un'applicazione per smartphone, denominata "Wher", che ha l'obiettivo di conferire maggiore senso di sicurezza e migliorare la qualità della vita di tutte quelle donne che vogliono intraprendere un viaggio da sole in città che non conoscono.

Nell'universo tecnologico esistono diverse applicazioni che permettono di sentirsi più sicure quando si cammina da sole per strada.

A differenza delle soluzioni più diffuse sul mercato, Wher utilizza un approccio preventivo al problema piuttosto che reattivo. Invece di intervenire al momento dell'emergenza, offre informazioni per conoscere prima la condizione di una strada. L'applicazione permette infatti di visualizzare le città turistiche più importanti mostrando i livelli di sicurezza percepiti, da chi conosce quei luoghi, nei vari punti della città.

La progettazione è avvenuta adottando la metodologia dello user-centered design che ha permesso di seguire un processo iterativo con delle fasi di testing dopo lo sviluppo di ogni prototipo. Partendo dall'analisi dei bisogni delle utenti, si definiscono i requisiti e si scelgono le tecnologie adatte ad implementare il servizio. L'app ha diverse funzionalità:

- Ricerca delle informazioni di sicurezza sulla mappa, disponibili anche in più lingue.
- Navigatore con i percorsi consigliati dalla Community di utenti.
- Inserimento dei dati sulla sicurezza (illuminazione, affollamento, consiglio personale).

È stato anche scelto di introdurre un meccanismo di gamification per motivare le utenti a utilizzare l'app.

L'architettura della soluzione prevede l'utilizzo di un server remoto, due database, uno relazionale e uno non relazionale e l'utilizzo dello smartphone dell'utente dotato di una connessione Internet per contattare il server tramite RESTful API.

Il lavoro di tesi si è concentrato principalmente sullo sviluppo della versione Android nativa dell'applicazione utilizzando il linguaggio Java. Sono state utilizzate le mappe di OpenStreetMap distribuite in versione open source da Mapbox.

Il prodotto sviluppato, come tutti i prodotti appena lanciati sul mercato presenta alcuni limiti sia di tipo tecnico, come la scelta di un linguaggio nativo che seppure più performante risulta d'altra parte più dispendioso in termini di tempo rispetto ad un linguaggio cross-platform. Sia di tipo progettuale, ad esempio la raccolta dei dati da

parte delle utenti è sicuramente molto ingaggiante ma necessita un investimento in termini di user experience e marketing operation.

Tali limiti possono essere superati da un continuo e reiterato sviluppo di funzionalità che rendano l'applicazione sempre più completa e user friendly.

Capitolo 1
Analisi del contesto

Uno dei principali problemi per le donne che viaggiano sole, specialmente in una città che non conoscono, è quello degli spostamenti notturni a piedi.

Ancor di più se, consciamente o inconsciamente, ci si trova in zone con tassi di criminalità non trascurabili, spesso ignorati dalla maggior parte dei visitatori che si recano per la prima volta in certi posti.

Il rapporto Istat sulla sicurezza del 2018 non aiuta a rassicurare la popolazione femminile in quanto rivela che seppur continui la diminuzione degli omicidi di uomini, rimangono invece stabili quelli di donne.

Sul fronte delle percezioni della popolazione emerge una situazione complessivamente positiva. Sebbene diminuisca la preoccupazione per sé o per altri della propria famiglia di subire una violenza sessuale e si notino con meno frequenza segni di degrado sociale nella zona in cui si vive, rimane però stabile il divario nella percezione della sicurezza tra i due sessi. Le donne percepiscono in misura maggiore rispetto agli uomini i rischi di subire reati.

Globalmente infatti nel 2017 gli uomini hanno riportato una maggiore percezione di sicurezza personale (75%) rispetto alle donne (61%) [2].

Secondo una statistica aggregata dal 2006 al 2017, solo il 49% delle donne italiane si sente al sicuro camminando da sola la notte [3].

1.1 Percezione della sicurezza

Una indagine sulla domanda di sicurezza delle donne nelle città di Piacenza e Ravenna, svolta a cura della Regione Emilia Romagna nel 2000, distingueva la sicurezza oggettiva che fa riferimento al rischio di essere vittime di reati, da quella soggettiva su cui influiscono in modo determinante valutazioni, sensazioni, emozioni, pregiudizi, norme e modelli culturali individuali [4].

Alcuni elementi come la presenza di attività e di vita, la sorveglianza spontanea da parte della comunità, unitamente alla presenza costante delle forze dell'ordine e della polizia municipale agiscono sia come deterrenti contro la criminalità che come trasmettitori di sensazioni positive di sicurezza.

Secondo questa indagine la percezione di sicurezza migliora al crescere dello spirito di comunità, e promuovendo un clima di maggiore fiducia e solidarietà.

1.1.1 Colpevolizzazione della vittima

I dati che abbiamo a disposizione sulle denunce per stupro o tentato stupro purtroppo sono insufficienti in quanto una buona parte di queste violenze non vengono denunciate dalle donne a causa del fenomeno di colpevolizzazione.

La colpevolizzazione della vittima consiste nel ritenere la vittima di un crimine parzialmente o interamente responsabile di ciò che le è accaduto e nell'indurre spesso la vittima stessa ad autocolpevolizzarsi.

Questo atteggiamento è anche connesso all'ipotesi che si debba accettare la "natura umana" e di conseguenza adeguarvisi anche a scapito dei propri desideri e della propria libertà.

1.1.2 Colpevolizzazione nei crimini sessuali e d'odio

Nell'ambito della violenza domestica e nel mobbing il processo di colpevolizzazione della vittima è stato descritto come parte integrante della violenza fisica e verbale e i comportamenti della vittima vengono sistematicamente interpretati come provocazione alla violenza [5][6][7].

Secondo certe vittime, azioni come lo stupro e la violenza di genere sarebbero provocate in diversi modi: dal flirtare, al tipo di abbigliamento, all'essersi trovata nel posto sbagliato al momento sbagliato [8][9].

Generalmente si parla di vittimizzazione secondaria quando le vittime di crimini subiscono una seconda vittimizzazione da parte delle istituzioni, dagli operatori sociali o dall'esposizione mediatica non voluta.

1.1.3 Teoria delle finestre rotte

La teoria delle finestre rotte è una teoria criminologica sulla capacità del disordine urbano e del vandalismo di generare criminalità aggiuntiva e comportamenti antisociali.

Secondo questa teoria, introdotta dai criminologi Wilson e Kelling nel 1982 nell'articolo "Broken Window Theory", non punire piccole trasgressioni può generare fenomeni di emulazione che portano fenomeni più gravi [10][11].

L'idea nasce dall'esempio della finestra rotta: se qualcuno rompe una finestra di un edificio e questa non viene aggiustata, chi la vede è portato a pensare che l'edificio sia abbandonato o lasciato senza cura, questo attrae altri teppisti nel rompere altre finestre e generando fenomeni di violenza contro la proprietà.

Questa formulazione, applicabile anche ad altri fenomeni di teppismo urbano, è stata adottata da diverse municipalità americane come modalità di gestione del territorio e prevenzione del crimine contro le proprietà, come ad esempio la riverniciatura dei muri su cui sono presenti graffiti.

La percezione di uno spazio pubblico come disordinato e insicuro può rafforzare negli abitanti il senso di abbandono da parte della comunità locale e delle autorità. Questo introdurrà comportamenti diversi nella popolazione, gli elementi più civili saranno impauriti, eviteranno o abbandoneranno queste aree, dando spazio all'occupazione del territorio da parte delle classi più disagiate.

1.2 Le soluzioni smart per sentirsi sicure

Esistono numerose app con un approccio reattivo che consentono alle donne di sentirsi più sicure e chiamare automaticamente i soccorsi in caso di emergenza o permettendogli di essere geolocalizzate e inviare la posizione ai propri amici, tra queste le più famose sono Kitestring, SecurWoman2.0, Siamo Sicure, S.H.A.W., BSafe, Where Are U e Stalking Buster [12][13].

Altre app invece permettono alle donne di incontrarsi in viaggio e condividere del tempo insieme o danno suggerimenti "al femminile" sui posti da visitare, ad esempio Pink Trotters o Tourlina [14].

1.2.1 Kitestring

Questo servizio monitora l'utente quando è fuori casa e avvisa gli amici quando questo non risponde [15].

Il punto di forza di questo prodotto è la sua estrema semplicità, non è necessario installare nessuna applicazione, l'interazione tra utente e servizio avviene esclusivamente tramite sms. L'utente imposta un intervallo di tempo mandando un messaggio, una volta scaduto il tempo, Kitestring contatta l'utente, sempre tramite sms, per assicurarsi che stia bene, se questo non risponde, viene automaticamente avvisata una lista di contatti tramite dei messaggi d'emergenza personalizzati.

Si può usufruire del servizio in modo gratuito ma con una limitazione nel numero dei viaggi mensili e impostando un solo contatto di emergenza oppure a pagamento alla modica cifra di 3$ al mese senza nessuna limitazione.

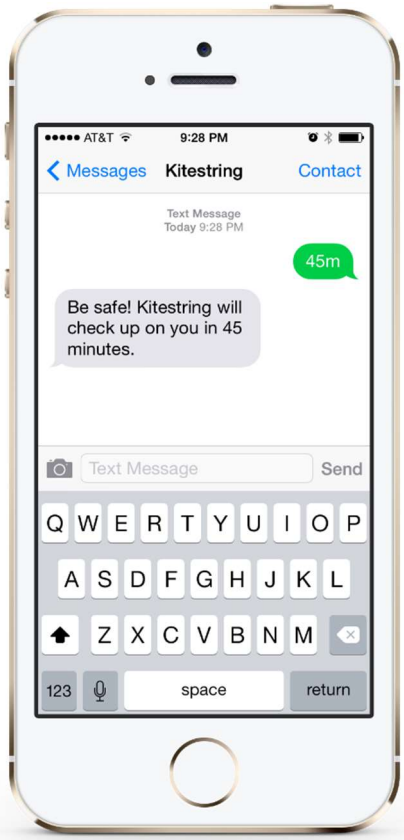

Figura 1.1: Kitestring

1.2.2 SecurWoman2.0

Questa applicazione accompagna l'utente al telefono in tutte quelle circostanze che possono generare ansia o preoccupazione o protegge in situazioni potenzialmente a rischio, collegando lo smartphone ad un Centrale Operativa, attiva H24, 7/7 in due lingue (italiano ed inglese) in grado di interfacciarsi se necessario a Forze dell'Ordine e di Primo Soccorso [16].

Il servizio offre anche la copertura assicurativa che garantisce le seguenti coperture:

- Assistenza sanitaria.
- Intervento di manodopera specializzata per l'apertura della serratura in caso di scippo/furto e ripristino.
- Rimborso per rifacimento chiavi e documenti.

Ogni utente, tramite un apposito modulo, può segnalare agli altri componenti del network di SecurWoman la presenza di un presunto pericolo o di una persona molesta in un punto preciso.

Anche qui il prezzo dell'abbonamento è molto contenuto, ovvero 2,99€ al mese.

1.2.3 Siamo Sicure!

Questa applicazione presenta 4 funzioni [17]:

- Allarme & suoni: permette di attivare effetti sonori e la funzione torcia per richiamare l'attenzione.
- Chiamata di emergenza: effettua una chiamata diretta immediata a un numero scelto dall'utente.
- Sms "Dove sono": invia un sms, con una richiesta d'aiuto contenente l'ultima posizione nota, a uno o più destinatari.
- Il Decalogo: questa è più una funzione di informazione che pone l'attenzione sulla prevenzione e sugli atteggiamenti consigliati per evitare aggressioni.

1.2.4 S.H.A.W.

Tutte le informazioni utili alla sicurezza, in dodici lingue [18].

Oltre ai tasti di chiamata rapida al 112 o al 1522 (snodo operativo delle attività di contrasto alla violenza di genere e stalking), permette alle utenti di individuare i centri antiviolenza o di ascolto più vicini alla propria posizione, tramite liste, mappe o ricerche avanzate, regione per regione. Nella sezione "Leggi" sono sintetizzati i principali aspetti legislativi relativi ai reati di violenza sessuale, con un focus aggiuntivo sullo stalking. È una soluzione gratuita.

1.2.5 Bsafe

Una delle più note app di tracciamento, è adatta a chi è pratico di social network e non ha problemi con l'inglese. L'app è disponibile gratuitamente. Permette di creare una rete di amici o familiari all'interno della quale indicare il contatto principale da avvisare in caso di pericolo [19].

Il pulsante "Alert Friends" consente di avvisare i contatti della rete in caso di bisogno, di lanciare un allarme sonoro, di registrare video e audio e di inviare immediatamente la propria posizione.

È possibile anche invitare gli amici a seguire l'utilizzatore nei suoi spostamenti, tramite tracciamento GPS e in caso di percorsi poco sicuri.

1.2.6 Where Are U

L'app ufficiale del NUE 112. Permette di chiamare rapidamente il 112, inviando contestualmente la propria posizione geografica (ottenuta tramite geolocalizzazione), i propri dati anagrafici e i contatti di riferimento scelti al momento della registrazione. In

questo modo, l'operatore del 112 che risponderà alla chiamata, visualizzerà subito tutti i riferimenti utili per la gestione dell'emergenza. Inoltre, dal software è possibile anche effettuare una "chiamata silenziosa" inviando una segnalazione per emergenza sanitaria, incendio o pubblica sicurezza nel caso in cui non possiate parlare direttamente al telefono [20][21].

1.2.7 Stalking Buster

È un'app gratuita per la gestione delle emergenze in situazioni di stalking o violenza sulle donne [22][23].

Stalking Buster permette a tutti di effettuare chiamate e di inviare SMS di emergenza fornendo automaticamente informazioni circa la posizione esatta dell'utente.

Mette a disposizione una mappa dettagliata con l'indirizzo e le coordinate della propria posizione in città, il pulsante centrale SOS 112, per effettuare in maniera immediata una chiamata al numero unico europeo per le emergenze (112) e il pulsante SOS STK utile per chiamare un numero dell'unità antistalking impostato dall'utente nella pagina delle impostazioni. Infine, con il tasto SOS SMS si invia un SMS automatico al numero dell'unità antistalking con link a Google Maps della posizione rilevata.

1.3 Il problema e la soluzione proposta: Wher

Tutte le soluzioni descritte precedentemente si basano su un approccio reattivo, non impediscono quindi che l'evento si manifesti ma cercano di rimediare in qualche modo al momento in cui il problema si manifesta.

La soluzione realizzata per questa tesi adotta, al contrario di quelle precedentemente descritte, un approccio preventivo. Non fornisce infatti strumenti da utilizzare in situazioni di emergenza ma si concentra sul fornire informazioni utili alla sicurezza personale prima o durante l'esplorazione della città.

Il prodotto che descriverò in questa tesi, il cui nome commerciale è "Wher", consiste in un'applicazione per smartphone sviluppata a Torino presso l'azienda Freeda S.R.L. insieme a un team multidisciplinare in quasi un anno di lavoro.

Di quest'app è stata sviluppata una versione per Android (ovvero il mio lavoro, di cui parlerò nel dettaglio) e una versione iOS (sviluppata da un altro componente del team). Il prodotto è destinato a un target femminile, principalmente composto da donne viaggiatrici, in particolare quelle che viaggiano sole o si apprestano a visitare una città che non conoscono.

L'applicazione svolge il compito di un'amica che ti consiglia dove andare e dove non andare in una città che lei conosce.

In questo periodo in cui ho partecipato alla realizzazione dell'app il team ha visto al suo interno varie figure professionali tra cui addetti al marketing e alle campagne pubblicitarie e di fidelizzazione, psicologi, UI e UX designer, ingegneri informatici.

Oltre a occuparmi della versione Android, mi sono occupato, in misura minore, del lato backend, in particolare della traduzione dei commenti e dell'implementazione della gamification.

1.3.1 Obiettivo

L'applicazione si pone come obiettivo principale quello di rendere le città più sicure per le donne che intendono visitarle.

Per far questo gli strumenti che abbiamo voluto fornire alle utenti sono stati delle valutazioni delle zone di certe particolari città (principalmente mete turistiche gettonate in Italia e in generale in Europa).

Queste valutazioni vengono effettuate da altre donne che conoscono già queste città e successivamente vengono mostrate alla turista su una mappa tramite diverse rappresentazioni.

Un altro obiettivo che ci siamo posti è stato quello di fornire all'utente una funzionalità di navigazione, al contrario però della maggior parte dei sistemi attuali di navigazione che forniscono il percorso più breve da una partenza a una destinazione, il nostro scopo è stato quello di fornre il percorso più sicuro tra i due punti.

Le valutazioni delle utenti utilizzano anche alcuni metri di giudizio oggettivi come l'illuminazione stradale e la presenza di persone in zona. Questi ovviamente possono variare in base alle fasce orarie quindi abbiamo anche avuto bisogno di dare all'utente la possibilità di scegliere la fascia oraria o le fasce orarie di riferimento per ogni valutazione.

1.3.2 L'utente finale

Le utenti quindi sono di due tipologie (che possono sovrapporsi):

- "Wherrior":

  Ovvero l'elemento della community che mappa le città che conosce e fornisce la materia prima per il funzionamento dell'applicazione, cioè le valutazioni.

  Queste utenti sono l'elemento più importante e per questo sono state fatte diverse campagne di fidelizzazione, sono stati creati dei contest a premi in diverse città e si è cercato costantemente di creare tra queste un forte senso di comunità.

  Tutto ciò anche grazie ad attività sui social e alla presenza di "Wherrior ambassador" in ogni città in cui è possibile inserire valutazioni.

  Queste ultime hanno il compito di gestire le altre Wherrior, motivarle e includerle nelle varie iniziative ideate dal team di Wher.

- Turista:

  Questa è invece in grado di usufruire delle valutazioni inserite dalle Wherrior tramite diverse forme di visualizzazione.

  È stato scelto in particolare per questa categoria un target di giovani donne e ragazze che hanno la possibilità di viaggiare da sole e hanno particolare dimestichezza con smartphone e i social network.

1.3.3 Contesti e scenari d'uso

Le Wherrior devono essere in grado di valutare le varie parti della città indipendentemente dalla loro posizione corrente, l'attività di "mapping" quindi presumibilmente viene svolta direttamente da casa o in alternativa per le strade (ad esempio durante i contest organizzati per la community), oppure subito dopo aver percorso delle strade (es. turista) per lasciare un feedback relativo al percorso.

Anche le turiste devono essere in grado di utilizzare l'app indipendentemente dalla loro posizione, ad esempio potrebbero essere interessate a visualizzare la situazione di un quartiere o di una città prima ancora di recarvisi. Possono visualizzare le valutazioni delle zone che hanno intenzione di visitare o nelle quali si trovano già. Possono cercare percorsi più sicuri per raggiungere una certa posizione a partire dalla posizione attuale o da una posizione arbitraria.

Il modo più immediato per visualizzare queste informazioni è tramite una mappa, su questa deve quindi essere possibile vedere qualitativamente la sicurezza delle varie zone e deve essere possibile visualizzare un percorso tra due punti con annesse informazioni relative alla sicurezza del percorso stesso.

Per le Wherrior deve anche essere possibile inserire le valutazioni sempre attraverso la mappa.

1.3.4 Le tecnologie necessarie

L'applicazione, essendo distribuita, necessita almeno dello sviluppo di un client e di un server.

Lato front-end l'interazione è attualmente prevista esclusivamente tramite uno smartphone o tablet dotato di connessione a internet o connesso a una rete wifi e opzionalmente dotato di gps. Lo scenario d'uso più comune prevede la fruizione tramite uno smartphone.

Lato back-end bisogna fornire un servizio in grado di essere interrogato dai client in qualsiasi momento. Questo servizio non ha la necessità di interrogare a sua volta i client o di contattarli autonomamente ma le sue funzionalità possono essere attivate sempre su richiesta. Inoltre non è necessario che le richieste dei client producano modifiche tempestive sul server e che quindi siano visualizzate immediatamente da tutte le altre utenti.

Il client è stato sviluppato per dispositivi Android interamente tramite l'IDE Android Studio mentre, per quanto riguarda il server, questo è stato sviluppato su Visual Studio Code grazie al runtime system Node.js. Descriverò in seguito questi prodotti nel dettaglio.

Capitolo 2
La progettazione

La progettazione è avvenuta facendo molta attenzione alla user experience (UX), questa grazie alla sua suddivisione in diversi livelli ha permesso anche una più facile suddivisione dei compiti.

Più in generale è stata adottata una strategia conforme allo schema iterativo dello user-centered design (UCD), permettendoci di realizzare alla fine di ogni fase di sviluppo dei prototipi intermedi da far testare alle beta tester ed eventualmente mandare in release sullo store.

2.1 User experience (UX)

La user experience ha a che fare con la specifica esperienza che l'utente ha con i prodotti che usa. Si riferisce a come l'utente vive e interagisce con un prodotto o un servizio, è più un concetto che un processo.

Secondo la International Organization for Standardization la user experience consiste nelle percezioni e risposte di una persona risultanti dall'uso o dall'uso anticipato di un prodotto, sistema o servizio.

Secondo Jesse James Garrett, il design della UX di un software può essere suddiviso in cinque step corrispondenti a cinque diversi piani. Questi piani vengono percorsi dal basso verso l'alto e vanno da un livello più astratto (strategy) a uno più concreto (surface) [24].

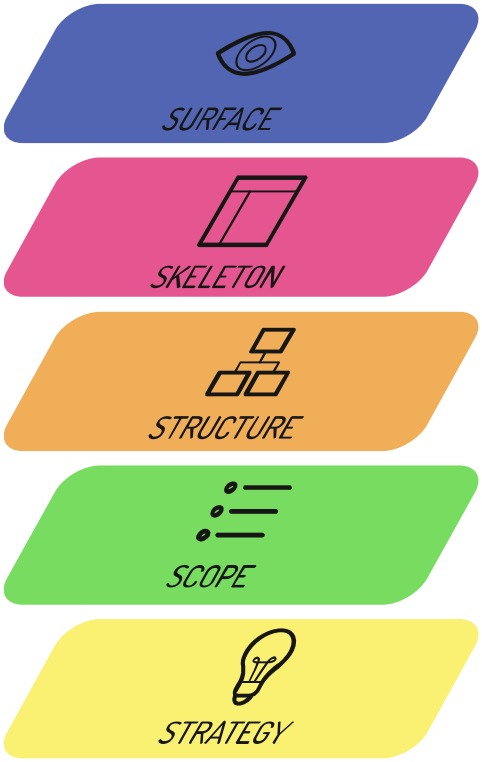

Figura 2.1: I cinque piani della user experience

- Strategy:

  Il livello più basso (strategia) analizza i bisogni dell'utente e gli obbiettivi che questo può raggiungere attraverso l'applicazione. Quindi ci si interroga sul perché creare l'applicazione e sul target verso cui questa è indirizzata.

  Spesso questa fase è accompagnata da interviste ai potenziali utenti e agli stakeholder.

- Scope:

  Il livello successivo (scope) ovvero lo scopo, serve a rispondere alla domanda "cosa farà l'applicazione?".

  Vengono definiti i requisiti, quindi le funzionalità e i contenuti. I requisiti devono soddisfare e devono essere allineati agli obiettivi strategici. I contenuti sono le informazioni necessarie per fornire un valore all'applicazione, possono essere testi, immagini, audio, video, etc. Le funzionalità permettono la fruizione di questi contenuti.

- Structure:

  La struttura definisce i modi in cui l'utente interagisce col prodotto e come quindi il sistema reagisce agli input dell'utente.

  Questo livello può essere suddiviso in Interaction Design e Information Architecture. Il primo, dati i requisiti funzionali, definisce come l'utente può interagire col prodotto e come il sistema deve reagire. Il secondo, dati i contenuti,

definisce la loro organizzazione per facilitarne la comprensione da parte dell'utente.

- Skeleton:

  Lo scheletro definisce la forma visuale delle schermate, la presentazione e la disposizione degli elementi con cui l'utente dovrà interagire.

  Questo livello può essere suddiviso in Interface Design, Navigation Design e Information Design. Il primo riguarda la disposizione degli elementi sull'interfaccia per permettere all'utente di interagire con le funzionalità del sistema. Il secondo stabilisce come navigare tra le diverse informazioni usando l'interfaccia. Il terzo stabilisce la presentazione delle informazioni in modo da facilitarne la comprensione.

- Surface:

  L'ultimo livello, ovvero la superficie, si occupa di come apparirà il prodotto, quindi di scegliere i giusti layout, i colori, etc.

  In questa fase, anche definita Sensory Design, si cerca di progettare l'aspetto visivo del contenuto in modo da dare all'utente dei suggerimenti su cosa può fare e come interagire con i contenuti, rendendo tutto semplice e immediato da capire.

Una strategia spesso affiancata alla UX, per la progettazione del prodotto, è l'UCD.

2.2 User-centered design (UCD)

Lo user-centered design è una struttura di supporto che posiziona i bisogni dell'utente al centro del processo di design [25]. Questo non è necessariamente limitato al design di prodotti digitali e fa parte dello standard ISO 9241-210 [26] (che ha rimpiazzato l'ISO 13407) [27].

Il processo UCD non stabilisce a priori metodi o tool da utilizzare. Una delle sue caratteristiche fondamentali è quella di coinvolgere attivamente l'utente ad ogni fase del processo di design per sviluppare un prodotto più efficiente, sicuro ed efficace.

Queste fasi includono:

- Ricerca e analisi
- Concept e strategia
- Design
- Sviluppo
- Implementazione e testing

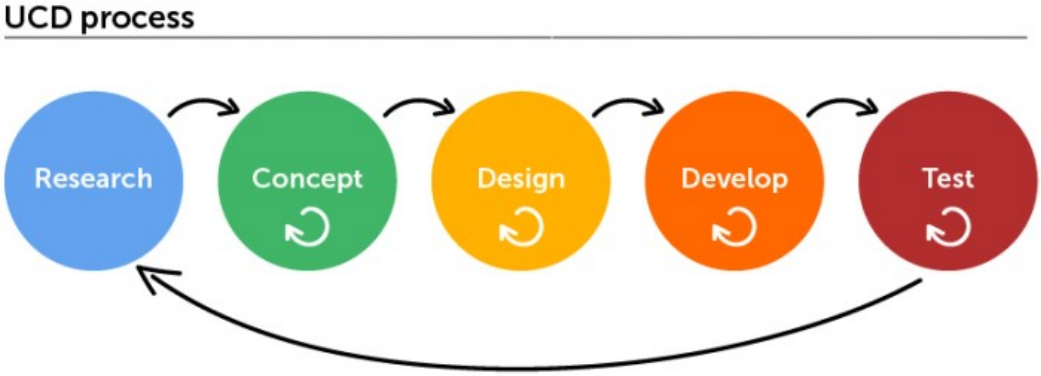

Figura 2.2: Le fasi del processo UCD

Le sopracitate fasi vengono ripetute più volte per rifinire ulteriormente il prodotto [28].

All'interno dell'UCD, specialmente in ambienti web e mobile, viene spesso inclusa la user experience (UX), questa aggiunge un importante valore al processo ed è necessaria per creare esperienze che attraggano gli utenti e li tengano interessati.

In relazione alla UX si può dire che l'UCD fa riferimento al processo o alla strategia applicata per progettare le esperienze.

2.3 Iniziare la progettazione

L'idea iniziale era quella di fornire all'utente finale un'applicazione semplice e pratica da utilizzare anche in movimento (direttamente sul luogo interessato) per potersi muovere in libertà, sentendosi più sicura.

A tale scopo è stato scelto di utilizzare un'interfaccia simile alla maggior parte delle applicazioni il cui funzionamento è basato sulla visualizzazione di mappe stradali.

Doveva anche essere data la possibilità di individuare il percorso migliore da un punto A a un punto B dove la qualità è data appunto dalla percezione di sicurezza da parte delle utenti.

Inoltre l'utente doveva essere in grado di consigliare alle altre utenti le strade che ella ritiene più o meno sicure, quindi fornirle uno strumento per la mappatura, dandole la possibilità di commentare le proprie scelte e valutare certe precise misure di sicurezza.

È quindi stata fatta una differenziazione tra le due differenti tipologie di utenti (che possono ovviamente anche sovrapporsi) ovvero chi visualizza le mappe e chi le valuta, nella prima categoria rientrano principalmente le viaggiatrici mentre nella seconda la community di supporto ovvero chi conosce bene un luogo perché ci vive ed è quindi in grado di dare consigli.

È stato anche necessario trovare dei meccanismi che incentivassero la mappatura delle strade da parte della community.

Sono quindi stati individuati dei requisiti funzionali di base.

2.4 Requisiti funzionali

I requisiti funzionali, ovvero quei requisiti che specificano cosa deve essere fatto (non come), sono indipendenti dalla tecnologia, dall'architettura, dalla piattaforma e dal linguaggio di programmazione [29].

Quelli riportati qui sotto sono i requisiti più importanti che sono stati identificati fin dalla prima fase di progettazione:

- L'utente deve avere la possibilità di registrarsi e fare il login direttamente dall'applicazione.
- Deve poter visualizzare un luogo qualsiasi spostandosi all'interno di una mappa o tramite una ricerca testuale.
- La Wherrior deve essere in grado di valutare le strade in base a certi criteri standard ed eventualmente commentare in modo libero la sua valutazione.
- La Wherrior deve essere in grado di visualizzare le sue valutazioni ed eventualmente modificarle o eliminarle.
- L'utente deve poter accedere al suo profilo dove poter modificare i suoi dati e le sue preferenze.

2.5 Vincoli

Devono anche essere presenti alcune restrizioni e vincoli che permettano ad esempio all'utente di interagire col sistema solo se autenticato e autorizzato. I principali individuati fin da subito sono stati:

- Il login deve essere permesso solo tramite account Facebook o Google.
- Possono registrarsi soltanto utenti di sesso femminile.

Altri vincoli sono stati aggiunti in seguito principalmente perché andavano a favore di una maggiore comprensibilità dell'applicazione da parte dell'utente o per facilitare l'implementazione o ancora perché imposti dagli strumenti utilizzati.

Ad esempio, sia per semplicità implementativa che per fornire uno strumento più semplice e intuitivo da utilizzare che abbia un comportamento sempre prevedibile e coerente, è stato scelto un ulteriore vincolo:

- Possono essere valutate sulla mappa soltanto le strade, quindi non edifici, parchi, etc.

2.6 Casi d'uso

L'utente, come è già stato detto precedentemente può interagire da attore col il sistema in diversi modi ovvero come turista, come Wherrior o come entrambi.

Per poter utilizzare il sistema innanzitutto l'utente deve autenticarsi.

In seguito il rapporto tra Wherrior e turista sarà simile a un rapporto produttore/consumatore dove i dati vengono inseriti in un certo modo dalla Wherrior e visualizzati in un altro modo ancora dalla turista.

L'utente in generale quindi potrà:

- Effettuare il login (o registrarsi se sta accedendo per la prima volta).
- Effettuare il logout.
- Visualizzare il suo profilo.
- Modificare il suo profilo.

La turista potrà:

- Visualizzare la sicurezza nelle strade sotto forma di valutazioni grafiche o testuali (possibilmente tramite l'ausilio di una mappa).
- Visualizzare un percorso a piedi sicuro tra due punti di una città.

La Wherrior invece dovrà essere in grado di:

- Inserire una valutazione.
- Modificare una valutazione precedentemente inserita.
- Eliminare una valutazione precedentemente inserita.

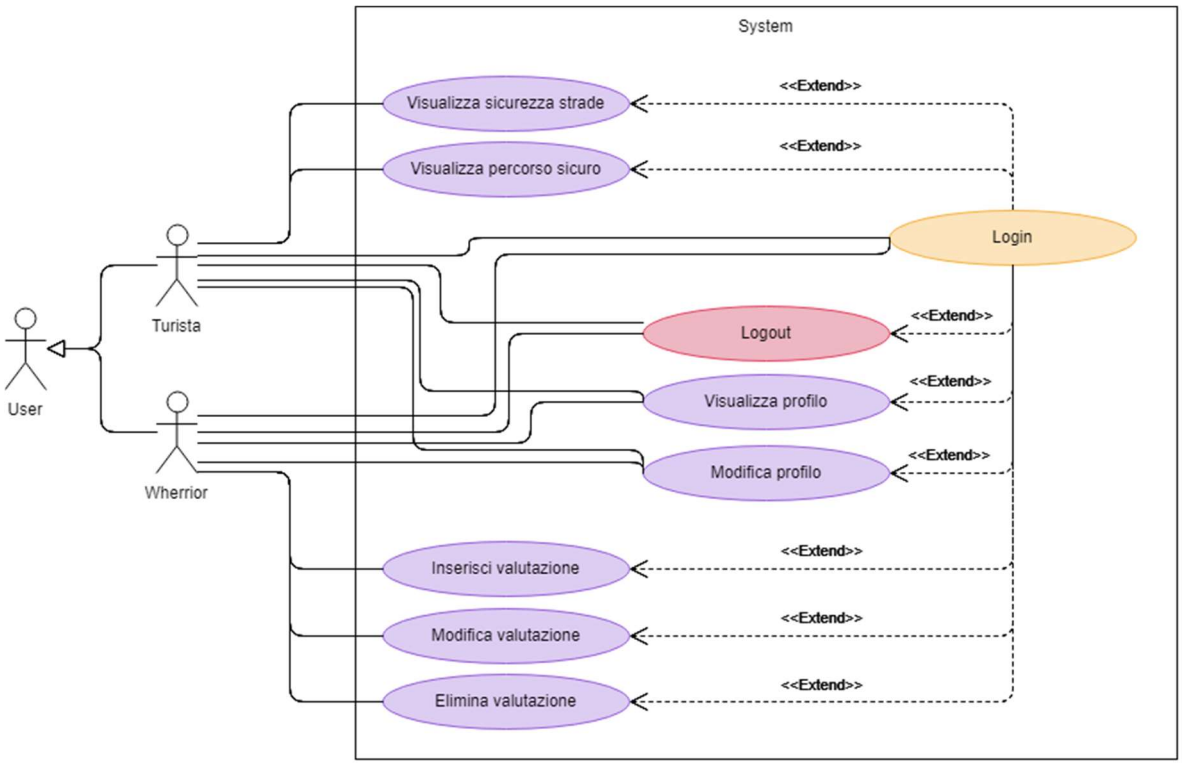

Figura 2.3: Casi d'uso

Date le precedenti considerazioni, è emersa la necessità di sviluppare l'applicazione per un sopporto mobile come uno smartphone, di inserire una mappa per facilitare la ricerca e la navigazione da parte dell'utente e di una interazione del client con uno o più server per richiedere informazioni fondamentali per la fruizione del prodotto.

A questo punto è stato introdotto un ulteriore elemento per motivare le Wherrior ad arricchire la banca dati ovvero la gamification.

2.7 Gamification

La gamification consiste nell'utilizzo di elementi presi in prestito dai giochi e di tecniche di game design in contesti esterni ai giochi. Lo scopo principale di questa tecnica è quello di coinvolgere le persone e divertirle nello svolgimento di certe azioni attraverso il gioco stimolando gli istinti umani, creando e appagando desideri e bisogni umani [30].

Un prodotto gamificato fornisce obiettivi da raggiungere, livelli in cui progredire, competizione con altri utenti, condivisione dei successi (possibilmente tramite i social) e guadagni di ricompense.

Tra le meccaniche dei giochi utilizzate nella gamification vi è quindi l'utilizzo di punti, livelli, premi, beni virtuali e classifiche. I punti offrono una ricompensa immediata che viene ottenuta dall'utente in seguito all'adempimento di una certa azione, questi aumentano la partecipazione dell'utente, tramite i punti è possibile vincere dei premi.

I livelli invece identificano lo stato gerarchico dell'utente classificandolo in base al punteggio ottenuto, l'accesso ad un livello più alto può fornire dei privilegi esclusivi. I beni virtuali incentivano l'utente ad impegnarsi per crearsi una propria identità all'interno della community. Le classifiche servono invece a creare la competizione.

La tecnica può essere applicata in tutte le occasioni dove l'obiettivo è quello di comunicare e diffondere un messaggio, tra i possibili campi di applicazione vi sono ad esempio salute e benessere, e-commerce, educazione, e-learning, programmi di fidelizzazione ed engagement, enterprise, addestramento, social learning, ambiente ed ecologia, ricerca scientifica.

La figura che si occupa della progettazione della gamification viene chiamata Game Designer e, anche se questo processo viene in genere applicato in contesti non ludici, spesso ci si riferisce al fruitore del servizio non come utente ma come giocatore.

2.7.1 La piramide degli elementi

Pensando al prodotto da realizzare come a un gioco, in gamification, gli elementi vengono solitamente inquadrati all'interno di una struttura piramidale chiamata appunto piramide degli elementi [31].

Questa piramide è composta da tre categorie:

- Dinamiche
- Meccaniche
- Componenti

La somma di queste categorie non costituisce l'intero gioco ma attorno agli elementi vi è l'esperienza, quest'ultima comprende ad esempio anche l'estetica.

L'esperienza ha la stessa importanza degli elementi.

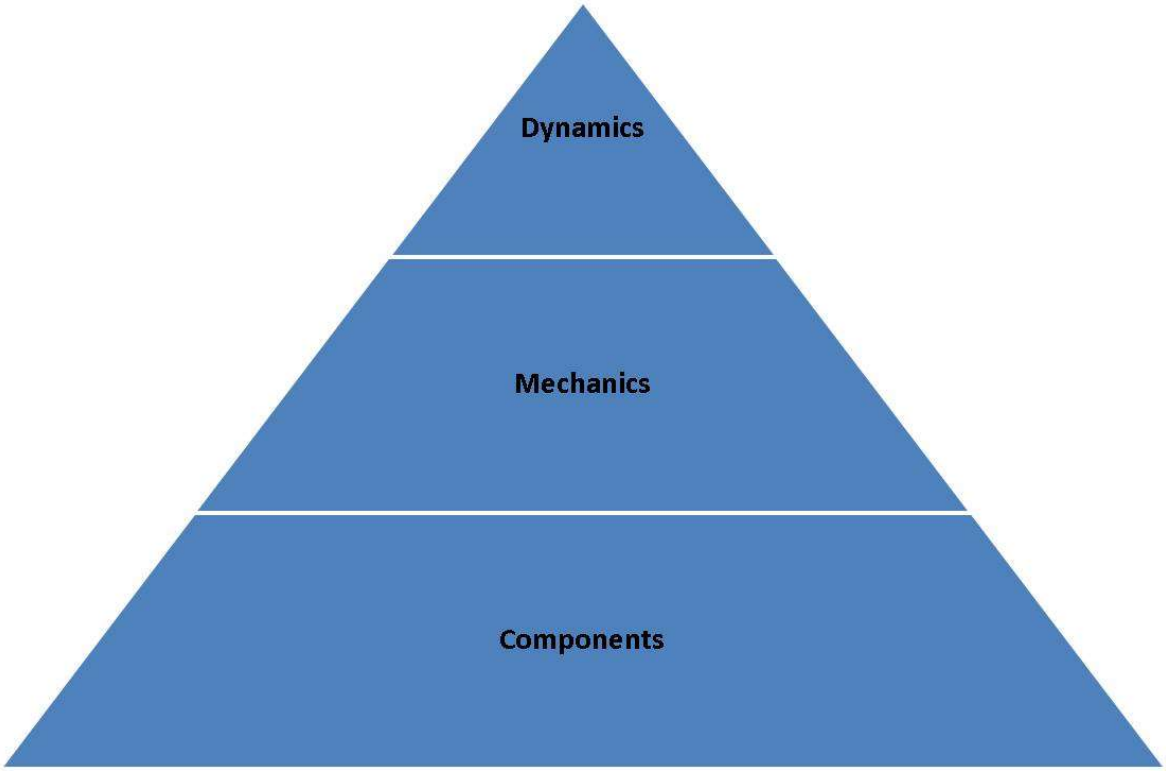

Figura 2.4: La piramide degli elementi

In cima alla piramide ci sono le dinamiche. Questi sono gli elementi di più alto livello e più concettuali, costituiscono la grammatica e rendono l'esperienza coerente.

A metà della piramide, le meccaniche, costituiscono i verbi della gamification quindi i motori dell'azione.

Il livello più basso, quello dei componenti, contiene i modi specifici di applicare i livelli sovrastanti.

Nell'immagine successiva sono rappresentati i principali elementi di ogni livello.

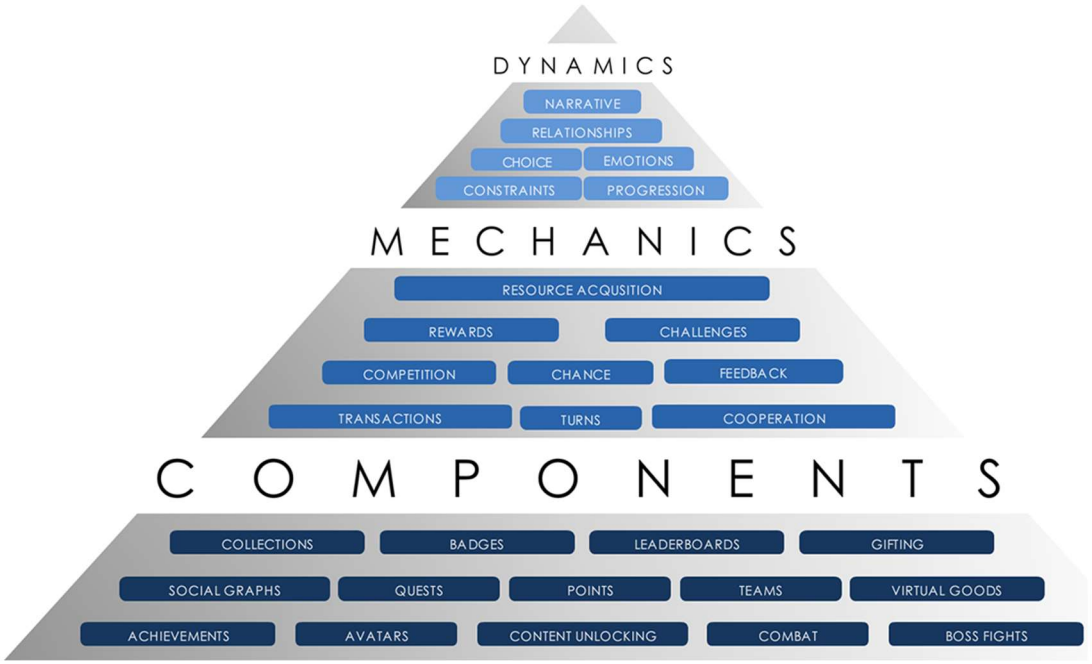

Figura 2.5: La piramide degli elementi con i componenti interni

I diversi livelli sono collegati tra loro, in quanto per sviluppare alcuni concetti di alto livello è necessario sfruttarne alcuni di livello più basso.

Scegliendo quindi quali elementi dei livelli inferiori implementare è possibile sviluppare uno o più elementi dei livelli superiori.

2.7.2 La triade PBL

Una delle strutture di base più utilizzate nella gamification consiste nella triade PBL (points, badges, leaderboards) ovvero punti, distintivi e classifiche.

Questa struttura si basa sull'acquisizione di punti, sullo sblocco di badges come riconoscimenti e su una classifica dei giocatori.

2.7.3 I punti

Il punto è l'entità più piccola, può essere vista come unità di scambio, come una valuta.

Le funzioni dei punti sono quelle di:

- Tenere un punteggio, utilizzabile ad esempio in una classifica.
- Determinare uno stato di vittoria, identificato da un certo numero di punti.
- Collegarsi a dei premi, sbloccabili grazie all'acquisizione di punti.
- Fornire feedback immediati al giocatore.

- Mostrare il progresso all'interno del sistema.
- Fornire informazioni ai game designer sull'attività dei giocatori.
- Essere interscambiabili, ovvero ogni punto è equivalente ad un qualsiasi altro punto. Questo permette al giocatore di scegliere le azioni da effettuare per ottenere i punti.

2.7.4 I badge

I badge sono degli elementi grafici, spesso hanno forme riconducibili a quelle di un trofeo. Non vengono utilizzati in tutti i sistemi di gamification in quanto, in certi casi, rischiano di essere controproducenti.

Le funzioni dei badge sono quelle di:

- Rappresentare visivamente un risultato ottenuto.
- Essere flessibili. Un badge può rappresentare qualsiasi cosa il game designer voglia e può essere ottenuto in diversi modi, anche randomicamente senza che l'utente se l'aspetti.
- Avere uno stile grafico che possibilmente rispecchi quello dell'ambiente al quale viene applicata la gamification.
- Segnalare l'importanza di qualcosa.
- Fornire credenziali. Un giocatore, tramite i suoi badge, se questi sono visibili agli altri giocatori, fornisce loro delle informazioni sui risultati raggiunti.
- Costituiscono una collezione, motivando certi giocatori ad accumularne il più possibile.
- Fanno da indicatore sociale.

2.7.5 Le classifiche

Le classifiche costituiscono elementi molto utili ma anche molto pericolosi in quanto in certi casi possono demotivare il giocatore.

La funzione principale di questi elementi sono:

- Mostrare un posizionamento, quindi fornire un feedback nelle competizioni.
- Essere personalizzabili. Non è detto che un giocatore debba vedere la classifica globale ma può anche solo vedere la classifica relativa ai suoi amici.

2.7.6 Limitazioni di questi elementi

Focalizzandosi troppo sugli elementi della gamification e mettendo in secondo piano il contesto si rischia di rendere tutto il processo inefficace.

Una cosa importante da tenere sempre a mente è che gli elementi non sono il gioco. Non tutte le ricompense sono divertenti e non tutto ciò che è divertente dà delle ricompense. Se il progettista si concentra troppo sul raggiungimento delle ricompense, dando a queste più importanza del processo stesso necessario a raggiungerle, rischia di far diminuire il coinvolgimento degli utenti [32].

Infine, se si basa tutto il sistema attorno al meccanismo PBL, il giocatore potrebbe identificare l'intero prodotto come uguale a tanti altri prodotti sviluppati attorno allo stesso meccanismo, indipendentemente dalle funzionalità fornite. Questo può demotivarlo se non ha intenzione di ripetere lo stesso meccanismo già ripetuto tante altre volte.

2.7.7 Comportamentismo

L'approccio scientifico alla psicologia che sta dietro la maggior parte degli elementi trattati finora e in particolare dietro lo schema PBL viene definito comportamentismo o behaviorismo.

Questo approccio, sviluppato agli inizi del Novecento da John Watson, è basato sull'assunto che il comportamento esplicito dell'individuo è l'unica unità di analisi scientificamente studiabile della psicologia. Le uniche due entità osservabili dallo studioso sono dunque lo stimolo ambientale e la risposta comportamentale.

Quindi l'individuo viene considerato come una black box di cui possiamo misurare solo input e output.

Questo approccio ha però dei difetti e delle limitazioni, una di queste è dovuta al concetto di hedonic treadmill del comportamentismo economico, questo termine fu coniato da Brickman e Campbell e nel loro articolo "Hedonic Relativism and Planning the Good Society" del 1971 [33].

L'hedonic threadmill o hedonic adaptation è la tendenza osservata nell'essere umano a ritornare velocemente ad un livello di felicità relativamente stabile indipendentemente da un cambio di fortuna o dalla realizzazione degli suoi maggiori obiettivi [34].

Bisogna stare attenti a non rendere i premi sempre prevedibili, in quanto a un certo punto il giocatore può stancarsi di accumularli.

Un'altra limitazione di questo approccio è l'eccessiva enfasi sullo status del giocatore. Non tutti i giocatori possono essere interessati a prevalere sugli altri o magari il raggiungimento dello status di altri individui può sembrare troppo difficile e demotivante. Inoltre molti non controllano costantemente il loro status. Ci sono altre ragioni per le quali una persona può essere portata a condurre un certo comportamento, come ad esempio per ragioni tangibili, per altruismo o per ragioni sociali che hanno a che fare con i suoi amici.

2.7.8 Cognitivismo

Il cognitivismo, o psicologia cognitiva, è una branca della psicologia applicata allo studio dei processi cognitivi, teorizzata intorno al 1967 dallo psicologo statunitense Ulric Neisser.

Il cognitivismo ha come obiettivo lo studio dei processi mentali mediante i quali le informazioni vengono acquisite dal sistema cognitivo, elaborate, memorizzate e recuperate.

L'essere umano non è quindi più considerato una black box e si fanno delle distinzioni tra diversi tipi di motivazione e diversi tipi di ricompense.

In particolare differenzia le ricompense in intrinseche ed estrinseche.

Le ricompense intrinseche si hanno quando le azioni del giocatore sono già di per sé appaganti, queste purtroppo sono diverse per ogni persona e per ogni momento.

Le ricompense estrinseche invece si hanno quando il giocatore fa qualcosa per qualche altro motivo diverso dalla cosa in sé.

Questa seconda tipologia in certi casi può demotivare il giocatore in quanto, se ci si focalizza troppo sulla motivazione estrinseca, anche se l'azione di per sé risulta piacevole, si rischia di perdere la motivazione intrinseca che vi era precedentemente. Dunque alla fine ci si ritrova a essere meno motivati rispetto all'inizio.

2.7.9 Tipi di ricompense

Il tipo di ricompensa fa la differenza nel motivare/demotivare il giocatore.

È stato riscontrato che i premi tangibili rischiano di demotivare molto più di quelli intangibili in quanto si sostituiscono molto più facilmente alla ricompensa intrinseca.

Le ricompense inaspettate invece sono più motivanti rispetto a quelle prevedibili in quanto il giocatore non compie il lavoro solo per ottenere la ricompensa.

Inoltre è stato notato come i riconoscimenti inerenti alla qualità della performance siano più efficaci e motivanti rispetto a quelli dati all'inizio dell'attività o al raggiungimento dell'obiettivo.

Certi studi, basati sulla SDT (Self-Determination Theory), tentano di utilizzare la motivazione estrinseca per arrivare a quella intrinseca, in particolare cercano di far leva sull'individuo facendolo sentire competente in quello che sta facendo, rendendolo autonomo in scelte significative e mettendolo in relazione con altri individui, ad esempio facendolo sentire parte di una comunità.

Capitolo 3
Le tecnologie utilizzate

Per lo sviluppo della parte front-end ho utilizzato Android Studio come IDE di sviluppo con l'ausilio di un emulatore per smartphone con sistema operativo Android 8.1.0 (API 27).

Per il client è stato scelto come linguaggio di programmazione Java versione 8 e sono state utilizzate diverse librerie, che discuterò in seguito, per la gestione della mappa, delle immagini, della comunicazione col server e altro ancora.

Per la parte di back-end invece ho utilizzato l'IDE Visual Studio Code su sistema operativo Linux e la piattaforma Node.js.

Sono stati utilizzati anche due database:

Il primo relazionale, PostgreSQL, con l'estensione PostGIS, utilizzato principalmente per la gestione delle valutazioni, il mio lavoro non ha previsto la creazione o modifica di questo database seppure sia stato indispensabile per il funzionamento del server. Ne parlerò dunque brevemente dato che costituisce un elemento importante per il progetto.

Il secondo invece non relazionale, MongoDB, utilizzato principalmente per la gestione degli utenti e delle valutazioni. Parlerò in particolare di MongoDB in quanto è stato quello al quale ho lavorato.

Per lo scambio di dati è stato utilizzato il paradigma REST.

La fase di prototipazione delle interfacce utente è stata resa possibile ai designer grazie al software Sketch per macOS.

3.1 L'architettura della soluzione

In particolare la soluzione da me realizzata ha visto l'impiego di una macchina virtuale Linux di Azure per ospitare il server. La logica del server è stata scritta interamente in Node.js e per la manipolazione dei dati si poggia sui due DB sopra citati.

Lato client invece l'interazione avviene al momento esclusivamente tramite smartphone, io in particolare ho lavorato sul sistema operativo Android.

Client e server comunicano utilizzando il paradigma REST.

3.1.1 Scalabilità della soluzione

Il tipo di server realizzato permette di creare eventualmente più macchine virtuali da utilizzare in parallelo e, tramite l'utilizzo ipotetico di un load balancer, si potrebbero evitare eventuali sovraccarichi.

L'utilizzo del paradigma REST permette, anche grazie alla sua caratteristica di essere stateless, di poter essere applicato praticamente a qualsiasi tecnologia, sarebbe infatti possibile implementare una versione desktop o web del client senza dover modificare il server e i messaggi scambiati.

3.2 Android Studio

Android Studio è l'IDE ufficiale per lo sviluppo di app Android, basato su IntelliJ IDEA. Fornisce agli sviluppatori numerose feature tra cui un Layout Editor che permette di costruire layout posizionando elementi dell'interfaccia utente all'interno di un editor visuale, un APK Analyzer, un emulatore che permette di testare l'applicazione su diversi dispositivi virtuali, un build toolkit Gradle per automatizzare e gestire il processo di build e un profiler che fornisce dati real time sull'utilizzo di CPU, memoria, rete e batteria [35].

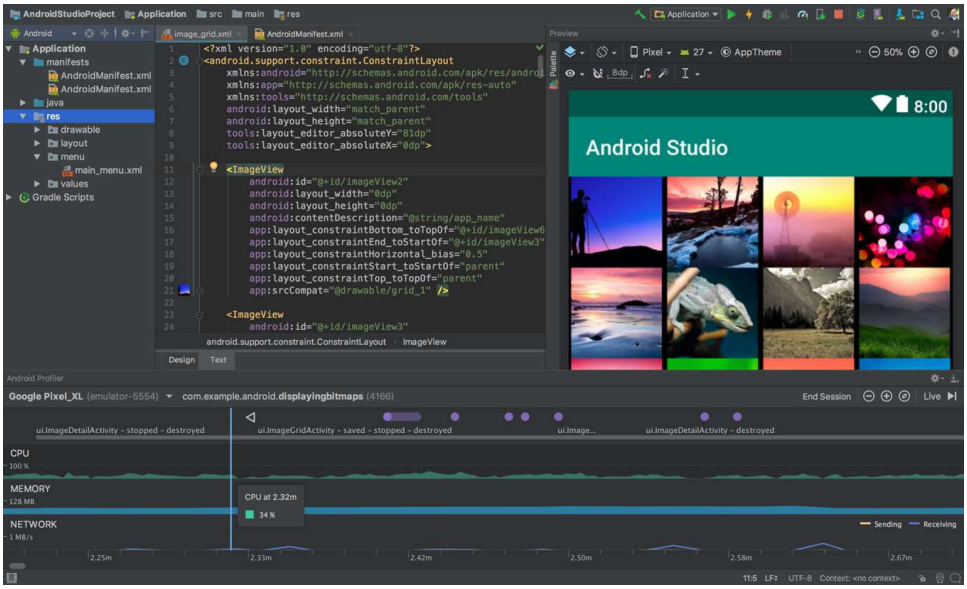

Figura 3.1: Android Studio

3.3 Android

Android è un sistema operativo per dispositivi mobili sviluppato da Google Inc. e basato sul kernel Linux. È un sistema embedded progettato principalmente per smartphone e tablet ma anche per televisori, automobili, smartwatch, occhiali, etc [36].

È il sistema operativo per smartphone più diffuso al mondo.

I linguaggi più utilizzati per sviluppare su questo sistema sono Java e Kotlin, è anche possibile sviluppare in C++ grazie alla relativa API.

Ogni applicazione gira in un suo processo con la propria istanza della macchina virtuale ART (Android RunTime) basata su tecnologia AOT (Ahead Of Time) [37][38].

I componenti principali di un'applicazione sono:

- Activity
- Service
- Content Provider
- Broadcast Receiver

Questi non devono necessariamente essere tutti inclusi nell'app [39].

3.4 Java

Come linguaggio di programmazione per il client è stato utilizzato Java.

È un linguaggio ad alto livello orientato agli oggetti a tipizzazione statica. È uno dei linguaggi più usati al mondo, specialmente per applicazioni client-server. Questo linguaggio viene in una prima fase compilato in bytecode e successivamente interpretato da una Java Virtual Machine (JVM).

Il motto principale di questo linguaggio è "WORA" (Write Once, Run Anywhere) in quanto il bytecode può essere eseguito da una qualunque implementazione della JVM.

Java venne creato per soddisfare i seguenti obiettivi [40]:

- Essere semplice, orientato agli oggetti e familiare.
- Essere robusto e sicuro.
- Essere indipendente dalla piattaforma.
- Contenere strumenti e librerie per il networking.
- Essere progettato per eseguire codice da sorgenti remote in modo sicuro.

Per eseguire il programma compilato è necessaria un'implementazione del Java Runtime Environment (JRE) mentre per sviluppare il programma è necessario Il Java Development Kit (JDK).

3.5 Le mappe

Per la creazione della mappa sono stati utilizzati i servizi offerti da Mapbox.

Data la loro diffusione specialmente in ambiente mobile, la scelta sul servizio da utilizzare per la visualizzazione della mappa ha visto come possibili candidati Google Maps e Mapbox.

Nella sezione successiva descriverò pro e contro di entrambe le soluzioni e i motivi che hanno portato alla scelta di una soluzione piuttosto che l'altra.

3.5.1 Google Maps vs Mapbox

Entrambi forniscono mappe e servizi di localizzazione. Entrambi vantano di mappe interattive, dettagliate e altamente customizzabili. Molte delle funzionalità offerte da Google Maps sono presenti (o per lo meno in modo simile) anche in Mapbox [41].

Entrambi ad esempio permettono l'inserimento di marker standard o customizzati, percorsi, forme e differenti stili.

Una delle differenze più evidenti tra i due servizi e conseguentemente uno dei punti di forza di Mapbox che ha portato alla scelta di questa soluzione consiste nella differenza di prezzo.

Le mappe dinamiche di Mapbox sono infatti molto meno costose di quelle di Google Maps.

Nelle tabelle sottostanti sono elencate le tariffe di entrambi i fornitori del servizio per quanto riguarda il numero di richieste di caricamento di mappe dinamiche.

Google Maps Platform Dynamic Maps (A partire da Luglio 2018)	
Fino a 28.000 caricamenti mensili	Gratis
0 – 100.000	7.00 USD ogni 1000 caricamenti
100.001 – 500.000	5.60 USD ogni 1000 caricamenti
Più di 500.000	Prezzo variabile

Tabella 3.1: Prezzi di Google Maps

Mapbox Dynamic Maps	
Fino a 50.000 caricamenti mensili	Gratis
Più di 50.000	0.50 USD ogni 1000 caricamenti

Tabella 3.2: Prezzi di Mapbox

La somiglianza tra le sorgenti dati supportate e le funzionalità dei due prodotti rendono facile il passaggio da Google Maps a Mapbox ma soprattutto Mapbox offre funzionalità simili, è facile da impostare ed è più economico di Google Maps.

3.5.2 Mapbox

Mapbox è una piattaforma per applicazioni mobile e web che permette di inserire feature di localizzazione come mappe, ricerca e navigazione all'interno di altre applicazioni [42]. Offre ad esempio una libreria (Mapbox GL JS) che usa WebGL per renderizzare mappe interattive, offre delle maps SDK per iOS, Android, Unity e Qt oltre che navigation SDK per iOS e Android e altri servizi come Maps API, Navigation API e Geocoding API.

Le mappe interattive sono altamente personalizzabili, offrono feature simili a quelle di Google Maps, offrono markers customizzabili, clustering di marker, mappe di calore, percorsi, forme e stili customizzabili.

3.6 L'architettura REST

REST, acronimo di Representational State Transfer, è uno stile architetturale per sistemi distribuiti [43][44]. Solitamente viene visto associato al protocolo HTTP ma in realtà questo legame non è obbligatorio. In generale un servizio RESTful (cioè un servizio che segue la logica REST) deve rispettare certi principi e vincoli:

- Vi deve essere una separazione dei ruoli di client e server, questi comunicano tramite una serie di interfacce uniformi e, fintanto che queste restano invariate, client e server possono essere sostituiti indistintamente.
- Ogni richiesta da parte di un client contiene tutte le informazioni necessarie per richiedere un servizio, la comunicazione è quindi stateless.
- Lo stato della sessione è memorizzato nel client oppure può essere delegato a un altro servizio presente nel server, ad esempio un database.
- Se è consentito dal tipo di risorse richieste, i client devono poter salvare queste risorse in una cache. Il client non può dire se è connesso direttamente a un server di livello più basso o intermedio.
- Client e server comunicano tramite un'interfaccia di comunicazione omogenea permettendo di disaccoppiare l'architettura.
- Ogni risorsa che può essere richiesta dal client deve essere identificata da un identificativo globale (URI).

L'interfaccia offerta da HTTP fornisce le operazioni CRUD (Create, Read, Update, Delete) definite da REST, dove all'operazione create è associato il metodo POST, alla read il GET, alla update il PUT e alla delete il DELETE. Sono disponibili anche operazioni aggiuntive tramite altri metodi HTTP.

Queste operazioni permettono di operare sulle risorse e queste vengono scambiate tra client e server tramite delle rappresentazioni in formati noti come XML o JSON.

3.7 JSON: JavaScript Object Notation

JSON è un formato leggero, basato su un subset del linguaggio di programmazione JavaScript, adatto all'interscambio di dati utilizzato molto in applicazioni client-server [45].

Ha una sintassi facilmente leggibile e scrivibile da esseri umani ed è semplice per le macchine generarlo e farne il parsing.

JSON è costruito su due strutture:

- Una collezione di coppie nome-valore
- Una lista ordinata di valori

Queste, essendo strutture dati universali sono supportate da praticamente tutti i moderni lignuaggi di programmazione, ciò rende facile lo scambio di informazioni in questo formato anche tra diversi linguaggi basati su questo tipo di strutture.

I tipi di dati supportati sono:

- Booleani
- Numeri interi
- Numeri in virgola mobile
- Stringhe
- Array
- Array associativi
- Null

Solitamente uno stream JSON ha in HTTP un'intestazione "Content-Type: application/json".

3.8 GeoJSON

Il GeoJSON è un formato di scambio dati geospaziali basato su JSON [46]. È stato spesso utilizzato principalmente nelle richieste di geocoding di cui parlerò a breve.

Questo formato definisce diversi tipi di oggetti JSON e il modo in cui questi sono combinati per rappresentare dati riguardanti feature geografiche con le lori proprietà e estensioni spaziali.

Il GeoJSON usa un sistema di coordinate geografiche chiamato World Geodetic System 1984 (WGS84), basato su un ellissoide di riferimento elaborato nel 1984. Usa unità di gradi decimali.

3.9 Forward geocoding e reverse geocoding

Con geocoding (o forward geocoding) si intende il processo computazionale di trasformazione di un indirizzo fisico in una posizione sulla superficie terrestre presentata tramite coordinate numeriche.

Con reverse geocoding si intende al contrario il processo di conversione di coordinate geografiche in descrizione di un posto (solitamente tramite un nome o un indirizzo).

3.9.1 Reverse geocoding in Mapbox

Per questo progetto è stato utilizzato il reverse geocoding offerto da Mapbox in particolare per ricavare la città in cui si trova l'utente a partire dalle sue coordinate spaziali.

Bisogna sottomettere una query per ogni (reverse) geocoding request. In seguito alla query si ottiene un geocoding response ovvero un documento formattato in JSON contenente i risultati più rilevanti per quella query.

La Mapbox Geocoding API contiene sorgenti dati provenienti dalle amministrazioni, da progetti open data e da compagnie private.

I source data contengono i seguenti tipi di informazioni geografiche:

- Points of interest (POI): Includono business commerciali, edifici pubblici, monumenti, parchi, etc.
- Address: Un indirizzo di posta specifico.
- Neighborhood: Un nome per un'area più piccola all'interno del Place.
- Locality: Una unità amministrativa che è più piccola di un Place. Presente solo in alcuni Country.
- Postcode: Codice postale.
- Place: Città, cittadine e villaggi. Alcune città molto grandi sono invece categorizzate come Region.
- District: Una unità amministrativa più grande di un Place ma più piccolo di una Region. Presente solo in alcuni Country.
- Region: Stati, province e prefetture. Questo è tipicamente la più grande unità amministrativa subnazionale di un Country.
- Country: Paesi generalmente riconosciuti o in alcuni casi, come Hong Kong, aree con status amministrativo quasi-nazionale che hanno un country code secondo la specifica ISO 3166-1.

Una query del tipo reverse geocoding permette di cercare una singola coppia di coordinate e ritorna gli elementi geografici che esistono a quella posizione.

È possibile raffinare i risultati della query con dei parametri opzionali, ad esempio limitando i risultati a dei Country specifici.

La risposta è una FeatureCollection (un tipo di GeoJSON della specifica del formato), un oggetto contenente una collezione di Feature ovvero di elementi conformi ai parametri specificati nella richiesta [47].

3.9.2 Forward geocoding in Google

Per quanto riguarda invece il forward geocoding, è stato utilizzato quello di Google per trovare la posizione sulla mappa a partire da un indirizzo o nome di un luogo inserito tramite la barra di ricerca.

Le API di geocoding forniscono un modo diretto per accedere a questi servizi tramite una richiesta HTTP.

È possibile specificare il formato dell'output, in particolare si può scegliere tra XML e JSON. È anche possibile utilizzare la crittografia con HTTPS.

Anche qui è alcuni parametri sono obbligatori mentre altri sono facoltativi.

L'accesso al servizio di geocoding è asincrono dato che le API di Google Maps necessitano di effettuare una chiamata a un server esterno. Per questo motivo è necessario passare una callback da eseguire una volta terminata la richiesta.

La callback riceve due parametri: la risposta e lo status code [48].

3.10 Place Autocomplete

Per una migliore fruizione della modalità di ricerca all'interno dell'app, è stato scelto di utilizzare i servizi di completamento automatico offerti da Google.

Il servizio Place Autocomplete è un web service che ritorna predizioni di posti in risposta a una richiesta HTTP. La richiesta specifica una stringa testuale di ricerca e opzionalmente dei confini geografici. Quindi il servizio può essere usato per fornire funzionalità di autocompletamento per ricerche geografiche testuali, ritornando posti come aziende, indirizzi e punti di interesse, mentre l'utente scrive.

È possibile richiedere l'output in formato XML o JSON.

Bisogna abbinare alla richiesta dei parametri obbligatori ovvero la stringa di input e l'API key mentre altri parametri, così come per il geocoding, rimangono opzionali [49].

3.11 Visual Studio Code e Node.js

Il codice che gira sul server è stato scritto grazie a Visual Studio Code e Node.js.

Visual Studio Code è un IDE sviluppato da Microsoft per Windows, Linux e macOS [50]. È possibile sviluppare in diversi linguaggi tra cui C, C++, C#, F#, HTML, PHP, Ruby, Java e JavaScript. Inoltre è basato sul framework Electron il quale permette di sviluppare applicazioni Node.js. È dotato della feature IntelliSense che offre funzionalità di code completion per agevolare la scrittura del codice al programmatore [51], offre un debugger con break point, call stack ed una console interattiva, ha i comandi git built-in permettendo le operazioni di push, pull, commit, etc. direttamente dall'editor, ed è estensibile e customizzabile (possono essere aggiunti linguaggi supportati, temi, debugger, connessioni a servizi aggiuntivi, etc.).

Node.js è invece una piattaforma open source per l'esecuzione di codice JavaScript server side. Si basa su un modello di networking IO event driven, in pratica Node rimane in sleep fino a quando non riceve una notifica dal sistema operativo relativa ad un determinato evento, a quel punto torna attivo per eseguire le istruzioni della corrispondente callback. Questo modello è particolarmente efficiente nelle situazioni in cui si verifica un elevato traffico di rete. Node.js è disponibile su Windows, Linux, macOS, come immagine Docker e per altri sistemi ancora [52].

Per facilitare l'interazione coi client, è stato utilizzato Swagger, un framework software open-source che facilita agli sviluppatori la progettazione, la costruzione, la documentazione e l'utilizzo dei servizi RESTful [53]. Il suo set di strumenti include il supporto per la documentazione automatizzata, la generazione di codice e la generazione di casi di test. In particolare è stato installato il modulo swagger tramite npm permettendo un'integrazione semplice con il codice in Node.js [54].

3.12 Database

Per la realizzazione di questa applicazione sono stati sfruttati PostgreSQL e MongoDB. Mentre il primo è stato utilizzato fin dall'inizio per memorizzare e recuperare le informazioni sulla sicurezza dei vari segmenti di strada, il secondo, del quale mi sono occupato per quanto riguarda la modifica e la realizzazione di diverse collezioni, ha dovuto permettere la memorizzazione di grandi quantità di dati prevalentemente testuali, come profili utente, commenti relativi alle valutazioni, etc.

Alcune di queste strutture dati sono in continua evoluzione e possono vedere aumentare o diminuire il loro numero di campi in base alle esigenze riscontrate in seguito alle varie fasi di testing. Per questo motivo, per queste strutture è stato scelto di utilizzare un DBMS non-relazionale come MongoDB.

3.12.1 SQL e NoSQL

Vediamo brevemente le principali differenze tra due tipi di DBMS (Database Management System), ovvero quelli relazionali (anche detti SQL) e quelli non relazionali (anche detti NoSQL), soffermandoci principalmente su quelli a documenti.

3.12.2 SQL

I database relazionali si basano su relazioni univoche fra i dati. Questi sono memorizzati per riga in strutture tabellari con un numero fissato di colonne, dove sono posti l'uno di seguito all'altro a formare singole tuple o record.

Ogni tabella rappresenta una entità e ogni riga contiene un oggetto ovvero una istanza di quella entità.

Questa struttura obbliga la frammentazione delle informazioni fra differenti tabelle, anche quando i dati descrivono un medesimo oggetto e per ottenere una visione dell'oggetto nella sua totalità è spesso necessario combinare tra loro differenti tabelle.

Ciò che rende un database relazionale è la presenza di legami fra queste tabelle cioè di connessioni logiche chiamate relazioni.

A causa della frammentazione degli oggetti in entry di più tabelle, in fase di progettazione è necessario effettuare un processo di normalizzazione ovvero di eliminazione della ridondanza informativa e del rischio di incoerenza dal database.

Questo spesso porta a decomporre grandi relazioni in relazioni più piccole se la relazione di partenza presenta più concetti tra loro indipendenti [55].

3.12.3 NoSQL

Nei database non relazionali, in particolare in quelli a documenti, i dati sono conservati invece in documenti e non in tabelle.

È possibile immagazzinare nel db degli oggetti usando la semantica JSON e associando quindi a delle chiavi i relativi valori, array di valori o altri oggetti. In oltre l'assenza di uno schema predefinito rende questo tipo di database estremamente flessibile in quanto, ad esempio, per aggiungere un campo a un oggetto già presente, è sufficiente aggiungere una nuova coppia chiave valore all'oggetto e non modificare la struttura di un'intera tabella contenente tutte le entità di quel determinato tipo.

Poiché in un DBMS NoSQL a documenti non sono presenti le relazioni, le informazioni vengono collegate tra loro tramite due metodi, in base alla situazione è preferibile usare l'uno piuttosto che l'altro [56]:

- Embedding:

  Ovvero inserendo un oggetto dentro un altro. Questo è sconsigliabile se ciò porta l'oggetto contenitore a crescere eccessivamente in dimensione oppure se i due oggetti hanno frequenze di accesso molto diverse. Utile principalmente nelle relazioni uno a uno o uno a molti.

- Referencing:

  Sostituisce efficacemente le relazioni. Consiste nell'inserire, all'interno di un documento, un campo il cui valore corrisponde all'identificativo di un altro documento. Utile principalmente nelle relazioni molti a molti .

3.12.4 Vantaggi e svantaggi dei database NoSQL

Le principali caratteristiche differenti dei database NoSQL rispetto a quelli SQL sono [57]:

- Leggerezza computazionale:

Non prevedono operazioni di aggregazione dei dati, spesso tutte le informazioni relative a un documento sono contenute nel documento stesso.

- Informazioni duplicate:

  Di contro questo porta a tenere nel db informazioni duplicate anche se questo, per quanto riguarda l'occupazione di spazio su disco, non è un grosso problema grazie ai costi relativamente bassi dei sistemi di storage.

- Assenza di schema:

  È possibile arricchire i campi di un documento definendoli all'interno dello stesso senza rischi per l'integrità dei dati.

- Scalabilità orizzontale:

  Grazie alla possibilità di non dover fare aggregazione sui dati e all'assenza di uno schema definito a priori, è possibile scalare orizzontalmente il database senza rischi operativi.

3.12.5 PostgreSQL

Nonostante tutte le buone qualità dei DBMS NoSQL, per la gestione dei segmenti di strada (vector tiles) è stato scelto di utilizzarne uno relazionale come PostgreSQL dato che con l'estensione PostGIS fornisce molti più metodi per l'elaborazione di elementi geografici rispetto ad altre soluzioni che pure trattano questo tipo di dato come ad esempio MongoDB.

PostGIS è un geodatabase ovvero un'estensione spaziale del DBMS a oggetti PostgreSQL, distribuito con licenza GPL. Questo aggiunge il supporto di oggetti geografici [58].

Tra i punti di forza di PostgreSQL vi è infatti la possibilità degli utenti di definire nuovi tipi basati sui normali tipi di dato SQL, permettendo al database di comprendere dati complessi, e il supporto per l'ereditarietà dei tipi. PostgreSQL permette anche di implementare funzioni in uno dei molti linguaggi supportati [59].

3.12.6 MongoDB

MongoDB è un DBMS (Database Management System) non relazionale quindi NoSQL, orientato ai documenti, in particolare vengono utilizzati documenti JSON-like che possono essere facilmente mappati in oggetti all'interno del codice dell'applicazione che vi si interfaccia. I documenti sono costituiti da alberi che possono contenere svariati dati, anche annidati [60]. I documenti sono raggruppati all'interno di collezioni che possono essere eterogenee. È possibile quindi gestire sia dati eterogenei e senza uno schema che dati omogenei e con uno schema.

Tra le caratteristiche principali di MongoDB troviamo:

- Ricerche per campi, intervalli e regular expression.

- Indicizzazione disponibile per qualsiasi campo, sono presenti anche indici secondari, indici unici, indici sparsi, indici geospaziali e indici full text.
- Alta affidabilità grazie ai replica set che consistono in copie dei dati.
- Scalamento orizzontale tramite shard e relativo bilanciamento dei dati per evitare shard troppo o troppo poco carichi.
- Capacità di file storage con associazione di metadati.
- Aggregazione dei dati tramite MapReduce o Aggregation Framework.
- Presenza di collezioni di dimensione fissa (capped collection) che si comportano come liste circolari.

Possiamo quindi identificare i vantaggi nell'usare MongoDB [61]:

- È schemaless. Se a un certo punto ho bisogno di aggiungere campi a un documento posso senza dover modificare la struttra della collezione relativa e tutti i documenti in essa presenti.
- È performante anche in presenza di grandi quantità di dati.
- È scalabile in orizzontale (sharding e repliche).

Come svantaggi invece abbiamo:

- Non sono presenti le foreign keys (tipiche dei database relazionali) ma è possibile ottenerle con dei prodotti ODM (Object-Document Mapper) come Mongoose, Mongoid o MongoMapper [62].
- Le informazioni sono salvate come coppie chiave-valore, occupano quindi più spazio rispetto ad un RDBMS.

Alcuni altri piccoli "problemi" sono stati risolti in passato, ad esempio dalla versione 3.2 è stata inserita la possibilltà di effettuare operazioni di JOIN (tipiche dei RDBMS) grazie agli Aggregation Pipeline States (in particolare il "$lookup") [63] e dalla versione 4.0 è stato aggiunto il supporto alle operazioni ACID (Atomicity, Consistency, Isolation, Durability) dando la possibilità di generare delle transazioni garantendo quindi l'atomicità di operazioni su più documenti e più collezioni [64][65].

Per la comunicazione tra server e db è stata utilizzata l'API mongoose per node.js.

Mongoose fornisce una soluzione schema-based per modellare i dati dell'applicazione. Include un type casting integrato, validazione, costruzione di query e altro ancora [66].

3.13 Altro software

Per la realizzazione delle grafiche dell'app ho fatto riferimento ai mock realizzati dai designer tramite il software Sketch [67].

Sketch è un editor grafico vettoriale sviluppato per piattaforma macOS utilizzato principalmente per la fase di design di UI e UX per applicazioni mobile e web.

I file creati con questo software vengono salvati in file .sketch e possono essere condivisi tra differenti applicazioni, anche in ambiente Windows, ad esempio un altro software in grado di utilizzare questo formato è Adobe XD [68].

I design possono essere anche salvati come semplici immagini PNG, JPG, etc.

Capitolo 4
Lo sviluppo

Per lo sviluppo della parte client è stato adottato il pattern Model-View-Controller, questo ha permesso una più facile suddivisione dei compiti e una più rapida modifica del codice grazie ai livelli di astrazione introdotti.

4.1 Pattern Model-View-Controller

In informatica il Model-View-Controller (MVC) è un pattern architetturale che suddivide l'applicazione in tre grandi parti interconnesse [69].

Tradizionalmente è utilizzato in ambiente desktop per lo sviluppo di interfacce utente grafiche (GUI) ma è diventato popolare anche in ambiente mobile e web.

La componente model è quella che gestisce direttamente i dati, la logica e le regole dell'applicazione, questa è indipendente dalla interfaccia utente.

La view comprende invece la visualizzazione delle informazioni, è possibile visualizzare gli stessi dati in diversi modi quindi tramite diverse view.

Il controller è lo strato intermedio che serve a interconnettere il model e la view, questo solitamente riceve input dall'utente che arrivano dalla view o gli input dal server e li attua modificando lo stato del model ed eventualmente la view.

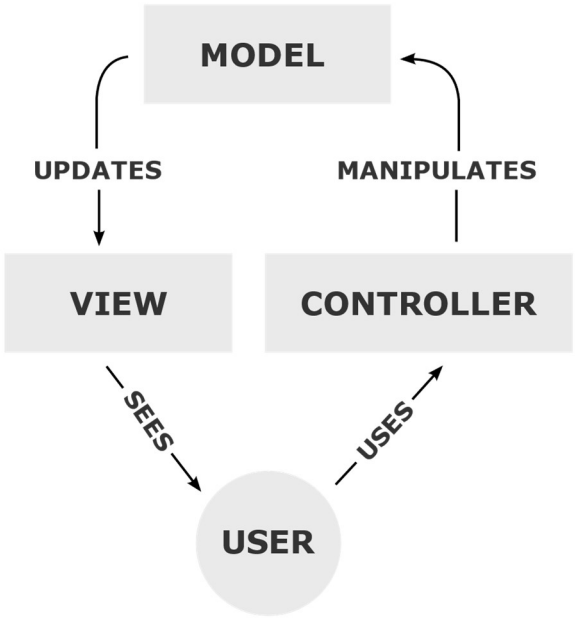

Figura 4.1: Pattern Model-View-Controller

Principali vantaggi:

- Un maggiore livello di parallelismo durante la progettazione, in quanto ogni sviluppatore può occuparsi di una delle tre parti senza conoscere le altre.
- Disaccoppiamento view/model: è possibile ad esempio collegare diverse view a un unico model per fornire rappresentazioni diverse dello stesso.
- Disaccoppiamento view/controller: è possibile modificare il modo in cui la view reagisce agli input dell'utente senza cambiare la sua rappresentazione visuale.

4.1.1 Model

La maggior parte dei dati utili per il funzionamento dell'applicazione risiedono sul server, questi comprendo i segmenti di strade valutabili, informazioni relative agli utenti registrati, le valutazioni già effettuate, i commenti degli utenti, le informazioni sulle città valutabili ed eventualmente già valutate e i relativi quartieri consigliati, gli obiettivi sbloccabili dagli utenti che mappano le strade.

Lato client, i dati che vengono memorizzati riguardano principalmente i dati di accesso dell'utente e una relativa struttura dati comprendente le informazioni di base, delle preferenze come ad esempio la visualizzazione dei commenti tradotti nella lingua del telefono o visualizzati in lingua originale, dati temporanei utili ad una migliore fruizione come ad esempio l'ultima posizione della camera all'interno della mappa, in modo tale da poter riprendere la navigazione dall'ultima posizione visualizzata sullo schermo quando si cambia vista per poi ritornare sulla mappa oppure quando si chiude l'app e poi si riapre.

4.1.2 View

A livello visivo l'applicazione Android, durante il suo utilizzo, oltre ad una ***Activity*** per il Login e una per il Tutorial, visualizza principalmente una sola ***Activity*** denominata ***TabbedHomeActivity***. Questa ospita al suo interno un solo Frame alla volta. I Frame sono inseriti all'interno di tre grandi sezioni:

- Mappa
- Esplora
- Profilo

Queste sezioni vengono selezionate da una ***BottomBar*** [70], il cui funzionamento è simile a quello della ***BottomNavigationView*** di Google introdotta nella API 26.1.0 [71].

Tramite la sezione Mappa si accede alle funzioni di visualizzazione delle informazioni sulla mappa, al dettaglio dei commenti e delle valutazioni effettuate dagli utenti, alla modalità di ricerca dei luoghi, di navigazione e di mappatura delle strade.

Tramite la sezione Esplora si accede invece a una selezione di città, già mappate dagli utenti, con relative informazioni sui quartiere e statistiche sulle valutazioni.

Tramite la sezione Profilo si accede ai dati personali dell'utente loggato.

Uno degli elementi grafici più utilizzati all'interno dell'app e presente in tutte e tre le sezioni è il ***`RecyclerView`***. Questo componente permette la creazione di liste per la visualizzazione di dati [72].

Per funzionare solitamente si appoggia a ben quattro componenti:

- ***`Datasource`***:

  L'insieme dei dati utilizzato per popolare la lista.

- ***`Adapter`***:

  L'elemento che estrae e prepara gli elementi dal datasource per essere successivamente visualizzati nel ***`RecyclerView`***.

- ***`ViewHolder`***:

  Fornisce il layout per gli elementi gestiti dall'***`Adapter`***. Mette a disposizione un numero limitato di view che vengono però continuamente riciclate per contenere un numero generico di elementi.

- ***`LayoutManager`***:

  È il responsabile della creazione e del posizionamento delle view all'interno del ***`RecyclerView`***.

4.1.3 Controller

La maggior parte della logica che regola la manipolazione dei dati è delegata al server, tuttavia il client si trova a dover tradurre tutta una serie di semplici input dell'utente in comandi più complessi che vengono successivamente interpretati dal server.

La comunicazione tra questi avviene tramite messaggi HTTP grazie all'architettura REST, questi messaggi vengono inviati dal client utilizzando la librerie Retrofit, OkHttp e Gson per Android.

In particolare Retrofit ha permesso la creazione delle richieste HTTP e ha fornito interfacce utili per la realizzazione delle callback da eseguire in seguito alla ricezione delle risposte dal server [73].

OkHttp ha fornito classi per l'invio delle richieste, per il parsing delle risposte e relativo body e un interceptor per gestire facilmente l'autenticazione dei client a ogni richiesta [74].

Gson invece ha reso semplice il lavoro di marshalling/unmarshalling degli oggetti Java in oggetti JSON e viceversa [75].

I pochi dati invece presenti nel dispositivo vengono memorizzati tramite shared preferences, un meccanismo nativo che permette di salvare in locale (in modo persistente o almeno finché non viene disinstallata l'app) coppie chiave-valore [76].

Viene anche mantenuta una cache con le immagini precedentemente richieste al server grazie alla libreria fresco di Facebook [77].

4.2 Build variants

Su Android Studio è possibile configurare delle build variants, queste permettono di creare diverse versioni di un'app a partire da un singolo progetto. Ad esempio è possibile creare due versioni dell'app, una gratuita e una a pagamento, bloccando nella prima certe funzionalità, oppure ancora versioni differenti in base al dispositivo su cui si vuole eseguire l'app [78].

È possibile creare differenti flavor ognuno col suo file AndroidManifest.xml, quindi aggiungere o eliminare elementi, modificare variabili di configurazione, etc.

In particolare sono state realizzate due varianti, una destinata all'utente finale e una destinata ai developer. Questa seconda versione è stata utilizzata per testare feature non ancora del tutto pronte o per abilitare delle opzioni sviluppatore.

La più utilizzata tra queste è stata quella che dà allo sviluppatore la possibilità di scegliere il server al quale collegarsi. Dato che, ogni volta che è stato necessario modificare il codice del server la nuova versione è sempre stata provata prima in locale, sono stati aggiunti (nella versione per gli sviluppatori) dei bottoni per selezionare il server da contattare senza bisogno di modificare il codice e ricompilare l'app. Questi bottoni appaiono sulla schermata di login toccando il centro dello schermo.

Figura 4.2: Developer options

4.3 Traduzione dei testi

L'applicazione prevede la possibilità di essere utilizzata in due lingue ovvero l'inglese (lingua di default) e l'italiano. La lingua viene impostata automaticamente in italiano se anche il sistema lo utilizza.

Sono quindi stati creati due file strings.xml (uno per lingua) all'interno della cartella delle risorse, contenenti tutti i valori delle stringhe di testo presenti nell'app. Il sistema utilizza automaticamente il file corretto in base alla lingua impostata.

Per facilitare il compito di traduzione evitando incongruenze tra le diverse versioni dello stesso testo e/o i nomi delle variabili utilizzate, è stato utile il tool Translations Editor di Android Studio.

Altro discorso è invece quello delle traduzioni dei commenti degli utenti, di queste parleremo nel dettaglio più avanti.

4.4 La gestione dell'utente

Come abbiamo visto precedentemente nella sezione sui casi d'uso, tutte le utenti, sia se agiscono da turiste che da Wherrior, devono poter essere in grado di effettuare il login/logout e interagire con il loro profilo online, creandolo, modificandolo o eliminandolo.

4.4.1 Login

Per poter utilizzare le funzionalità dell'applicazione è necessario aver effettuato il login. Questo avviene esclusivamente tramite i social Facebook o Google (nessun username o password è quindi richiesto dall'applicazione).

Questi servizi rilasciano un token all'utente, l'utente quindi invia il token appena ottenuto al server di Wher, questo effettua un controllo per vedere se l'utente è già presente nel database del sistema o se è la sua prima autenticazione.

Quando il server riceve il token esterno genera un suo JWT (JSON Web Token), della durata di 3 ore, che il client dovrà inviare a ogni richiesta REST per garantire l'autenticazione, e un refresh token da inviare nel caso in cui il precedente token sia scaduto (saltando la fase di login tramite il servizio esterno), in questo modo vengono aggiornati entrambi e reinviati al client.

Se l'utente sta accedendo per la prima volta o non aveva ancora completato la fase di registrazione, si passa a una fase preliminare dove bisogna inserire dei dati personali per completare la creazione del profilo nel database.

In seguito, l'access token, grazie a un interceptor che cattura tutti i pacchetti in uscita dal client, viene inviato insieme alle richieste fintanto che l'utente rimane loggato all'interno del sistema. A questo scopo è stata creato un oggetto che implementa l'interfaccia Interceptor di okhttp3, questo oggetto viene collegato al client e, ogni volta che viene inviata una richiesta che non sia di accesso, viene aggiunto il token all'header del messaggio.

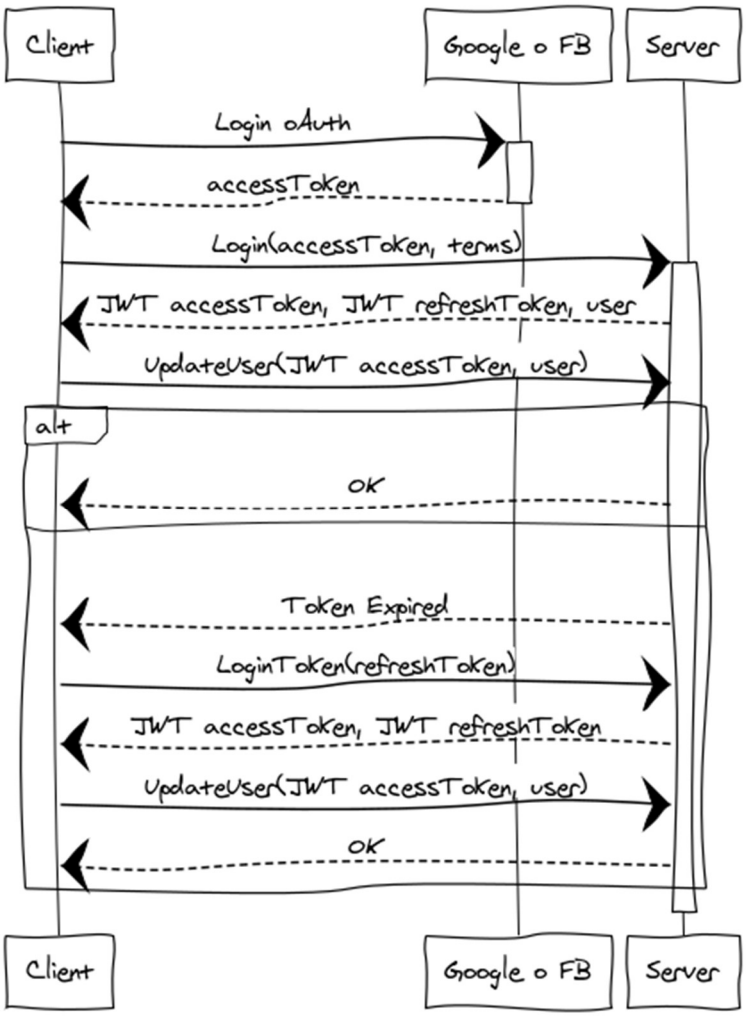

Figura 4.3: Accesso al servizio

4.4.2 Il modello dell'utente

Una volta effettuata la registrazione, viene creato sul server il nuovo utente, questo è costituito da più oggetti memorizzati su MongoDB:

- User:

  Contiene le informazioni di base, queste non vengono visualizzate all'interno dell'app, sono necessarie alla fruizione dell'applicazione o utili per fini statistici. Comprendono i seguenti campi, non sono modificabili dopo la creazione:

  - Provider id
  - Provider name
  - Nome

        - Cognome
        - Email
        - Refresh token
        - FCM tokens (token di Firebase per le notifiche push)
        - Data di creazione
    - Profile:

      Contenente le informazioni inserite dall'utente stesso o ottenute dal social con cui è stato fatto l'accesso al momento dell'iscrizione e da cui è stato importato il profilo base. Queste informazioni, modificabili in qualunque momento dall'utente tramite l'app, sono:

        - Avatar
        - Nickname
        - Gender
        - Data di nascita
        - Città primaria (la città in cui vive)
        - Città secondaria (eventuale altra città che conosce bene)
        - Data di modifica
    - GameProfile:

      Contenuto all'interno dell'oggetto Profile. Ha al suo interno informazioni ridondanti calcolate in base all'attività dell'utente, queste vengono aggiornate ogni volta che si raggiunge un certo obiettivo e sono:

        - Livello
        - Punti richiesti per salire di livello
        - Punti ottenuti nel livello corrente

4.4.3 Controlli sul nuovo utente

È necessario un controllo preliminare sugli utenti che tentano di accedere all'app.

Oltre a essere previsto l'accesso esclusivamente tramite social network (Facebook o Google), viene infatti effettuato anche un controllo sul sesso dichiarato dall'utente nel profilo del social scelto o dichiarato al momento della registrazione se non specificato sul social.

Se il sesso non è definito viene richiesto di esplicitarlo.

Se il sesso è maschile, viene impedito l'accesso.

Se invece il sesso risulta femminile viene effettuato un controllo sul nome: se il nome risulta femminile viene garantito l'accesso, se il nome risulta maschile o non

categorizzabile viene consentito l'accesso ma inserito l'utente in una lista per un successivo controllo.

4.4.4 Schermata di login e registrazione

La schermata di login dunque non prevede nessun campo per l'inserimento manuale del testo ma esclusivamente due pulsanti per l'accesso e due link per visualizzare termini e condizioni e le politiche sulla privacy.

Alla pressione dei pulsanti di login viene chiamato il relativo metodo della libreria di login del social associato. Viene quindi mostrata una nuova activity che consente all'utente di inserire le sue credenziali e fornire l'accesso ai dati personali a Wher. Dopo aver effettuato l'accesso sul social e aver acconsentito alla condivisione dei dati, viene richiamata una callback asincrona che riceve un token d'accesso, questo viene quindi inviato al server di Wher per completare il login e ottenere il JWT personale utilizzato per autenticare tutte le successive richieste.

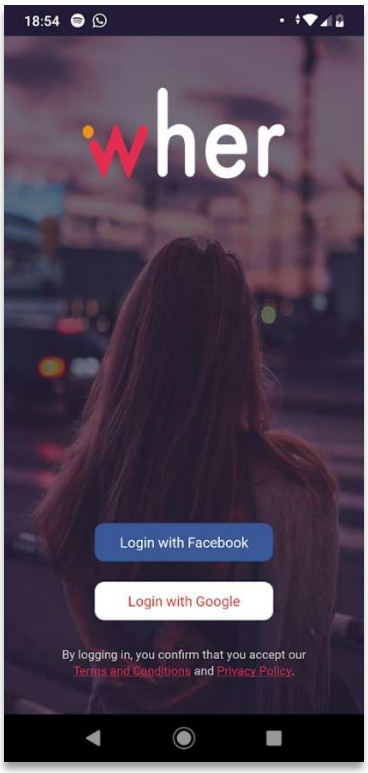

Figura 4.4: Schermata di login

Se l'utente accede per la prima volta, viene reindirizzata a una schermata dove poter completare il suo profilo con alcuni informazioni, alcune di queste sono rese pubbliche e quindi visibili dagli altri utenti, altre sono private altre ancora a discrezione dell'utente (es. immagine del profilo).

I campi che l'utente deve compilare durante la fase di registrazione sono:

- Nickname (il valore di default viene ricavato dal nome utente del social col quale è stato effettuato l'accesso)
- Data di nascita
- Città nella quale vive
- Altra città con cui ha familiarità (questo campo è facoltativo)
- Mezzi di trasporto più utilizzati in città
- Frequenza di spostamenti serali in solitaria
- Altri campi utili per fini statistici

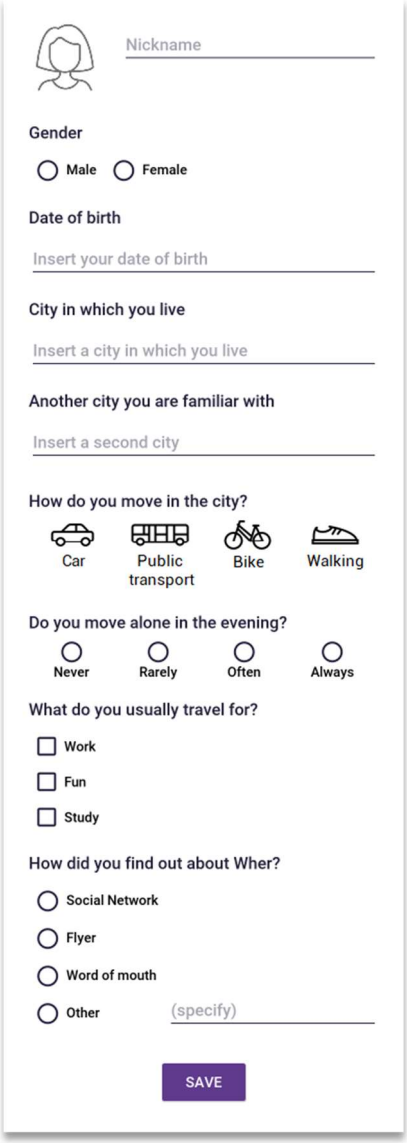

Figura 4.5: Form di registrazione

Associato al profilo vi è anche un avatar, questo viene ricavato dal social dal quale si effettua l'accesso, selezionando l'icona dell'avatar è possibile mostrarlo o nasconderlo agli altri utenti.

4.4.5 Profilo

Dall'apposito tab della barra in basso, è possibile entrare nella sezione relativa alle impostazioni e alla gestione del proprio profilo.

In particolare tra le funzioni più importanti vi è la possibilità di visualizzare le valutazioni effettuate, con la possibilità di modificarle o eliminarle singolarmente ed effettuare il logout.

Una sottosezione importante è quella dell'area personale che mostra gli obiettivi raggiunti, si accede a quest'area selezionando la prima voce dell'elenco (dove viene mostrato un resoconto del livello attuale dell'utente). Da questa sottosezione sarà anche possibile modificare i dati personali inseriti in fase di registrazione.

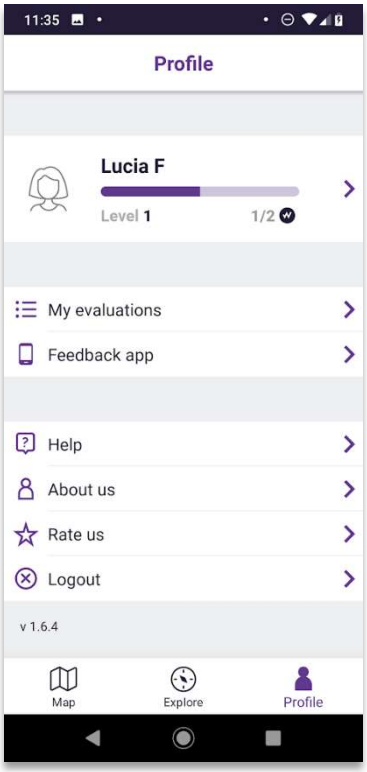

Figura 4.6: Sezione profilo

Figura 4.7: Modifica dei dati personali

È possibile anche modificare il proprio nickname. Sia nella fase di registrazione che di modifica dei dati, per evitare un lento approccio trial and error è stato inserito un

piccolo badge colorato tramite una ***ImageView*** (un oggetto contenente una immagine) di fianco all'***EditText*** (un oggetto dotato di input testuale) dove viene inserito il nickname che indica se il testo inserito è valido o meno grazie all'utilizzo di due differenti simboli.

Per validare il testo inserito, a ogni modifica, il client invia una richiesta al server per chiedere se il valore scelto è già utilizzato da qualcun altro o se è ancora libero. Alla ricezione della risposta viene aggiornato il badge.

Figura 4.8: Nickname badge

Se il nickname non è disponibile, non è valido o se alcuni campi non sono stati compilati (fatta eccezione per la città secondaria), al momento della richiesta di salvataggio, viene impedita la modifica e viene richiesto all'utente di riempire i campi mancanti tramite la classe ***Snackbar*** della Android support library.

È stato utilizzato spesso questo componente, in generale per dare brevi messaggi di errore da visualizzare per un breve tempo predefinito, per la sua semplicità e velocità di implementazione. Per utilizzarlo basta infatti richiamare il metodo statico ***Snackbar make(@NonNull View view, @NonNull CharSequence text, @Duration int duration)*** della classe ***Snackbar*** per creare una snackbar e, su questa istanza, il metodo ***void show()*** per visualizzarla [79].

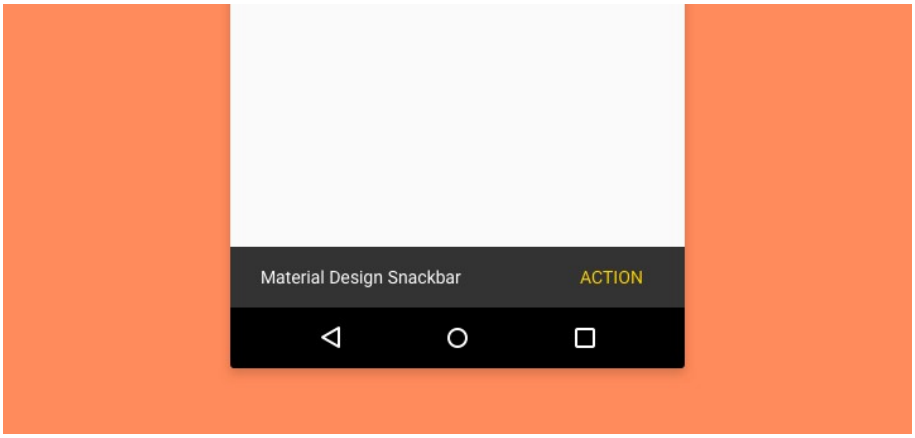

Figura 4.9: Snackbar

4.4.6 Obiettivi

Allo scopo di motivare le Wherrior a mappare le strade delle loro città sono stati introdotti una serie di obiettivi. Questi sono visualizzabili nell'area personale dell'utente e necessitano di un certo numero di azione da compiere per essere raggiunti.

Ogni utente ha un livello. Ogni obiettivo raggiunto fa incrementare il numero di punti dell'utente e, ottenuto un certo numero di punti si ottiene una promozione al livello successivo. Il numero di punti necessari per salire di livello segue una curva quadratica cosicché risulti relativamente facile scalare i primi livelli e via via più difficile raggiungere quelli successivi.

La logica che sta dietro l'assegnazione dei punti è stata realizzata interamente sul server e non necessita di richieste esplicite da parte del client ma l'algoritmo relativo all'assegnazione dei punti viene interpellato direttamente dai metodi del server che svolgono le azioni richieste dai client.

In questo modo può essere cambiato facilmente l'algoritmo della gamification senza modificare il client e senza modificare l'interfaccia REST del server.

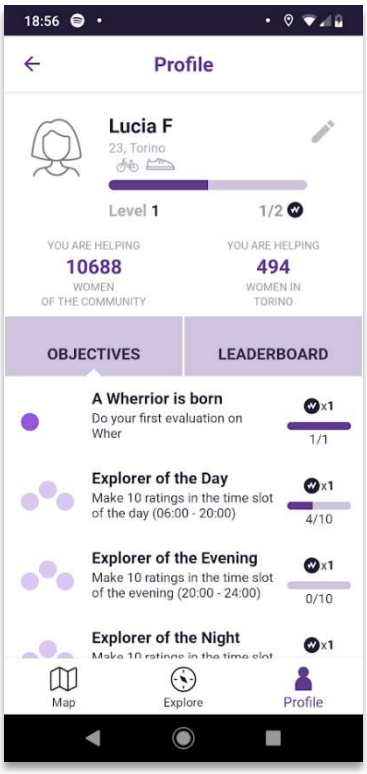

Figura 4.10: Elenco degli obiettivi

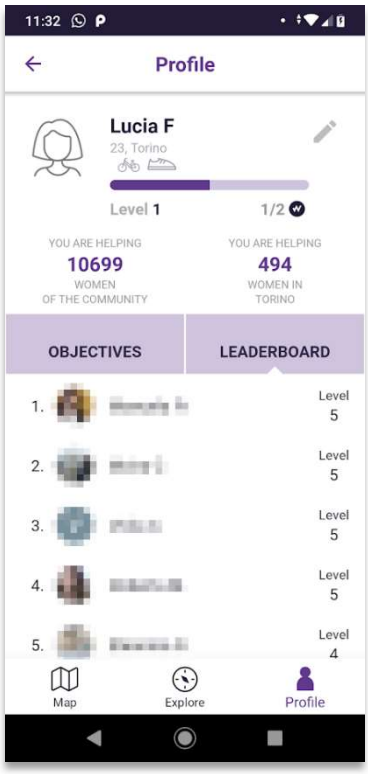

Figura 4.11: Classifica

Gli obiettivi sono memorizzati in una collezione di oggetti Trophy su MongoDB.

Ogni obiettivo contiene un identificativo e dei sotto-obiettivi, questi hanno un nome, una descrizione, un numero di task di un certo tipo da effettuare per l'adempimento e un numero di punti che vengono assegnati all'utente al momento del completamento.

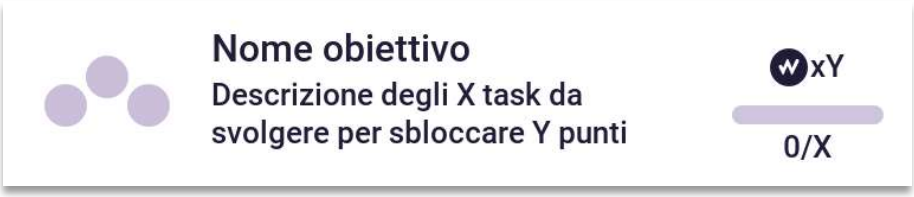

Figura 4.12: Esempio di obiettivo

Sebbene gli obiettivi siano rivolti principalmente alle Wherrior in quanto premiano la collaborazione e la popolazione del database con valutazioni e altro, alcuni possono anche essere ottenuti da una turista che non è intenzionata a mappare aree di citta, ad esempio uno di questi prevede che l'utente dia dei feedback ai commenti delle altre utenti selezionando le icone "pollice in giù" o "pollice in su".

Il raggiungimento degli obiettivi (e lo scalamento dei livelli) permette all'utente di posizionarsi all'interno di una classifica della quale è possibile visualizzare i primi posti contenenti i profili che più hanno contribuito a fornire materia prima per l'app grazie alle loro azioni.

Ogni volta che l'utente chiede di effettuare una particolare azione, richiesta per il completamento di uno o più obiettivi, questa azione viene eseguita dal server e subito dopo viene eseguito il metodo ***incrementTasksCounter()*** del controller ***trophies***.

Questa classe si occupa della gestione della gamification e il metodo appena menzionato ha il compito di incrementare (o decrementare) i contatori del numero di task di un certo tipo effettuati. A tal proposito esistono due contatori distinti, associati ad ogni coppia utente – obiettivo, questi contatori sono inseriti all'interno di documento ***TrophyUser***, gli attributi di questo oggetto sono:

- ***userId*** :

  L'identificativo dell'utente.

- ***trophyId*** :

  L'identificativo dell'obiettivo.

- ***tempTasksExecuted*** :

  Il numero esatto di task effettuati. Questo può essere incrementato o decrementato. Ad esempio, per l'obiettivo che richiede di valutare i commenti delle altre utenti, questo mi viene incrementato ogni volta che seleziono su un commento la reazione "pollice in su" o "pollice in giù". Viene invece decrementato deselezionando una delle reazioni precedentemente inviate.

- ***maxTasksExecuted*** :

  Questo contatore invece può solo essere incrementato e corrisponde al valore massimo mai raggiunto da ***tempTasksExecuted***.

Entrambi i contatori non possono mai assumere valori minori di zero.

Per il calcolo dei task eseguiti per il raggiungimento di un obiettivo viene considerato esclusivamente il valore di ***maxTasksExecuted***. In questo modo, anche annullando dei task precedentemente effettuati, l'utente non perde mai il suo livello o i premi ricevuti ma viene comunque penalizzato in quanto per proseguire ulteriormente nel medesimo obiettivo dovrà prima compiere un numero di task extra pari a quelli che ha precedentemente annullato.

Sul client sia la classifica che la lista degli obiettivi sono state implementate tramite l'elemento ***RecyclerView***.

4.4.7 Logout

Dalla schermata del profilo è anche possibile selezionare l'opzione logout, selezionandola si torna alla schermata iniziale di login.

In questo modo vengono eliminate dalla memoria del telefono tutte le preferenze, i token e in generale tutto cioè che è legato alla persona precedentemente loggata.

Al successivo login, se effettuato col medesimo social network, al quale è sempre associato lo stesso utente, questo verrà riconosciuto dal server e indirizzato alla schermata principale dell'app senza ovviamente dover effettuare una nuova registrazione.

4.5 Valutazioni

Come è stato già detto, la materia prima attorno alla quale è stato costruito il sistema è data dalle valutazioni delle utenti.

Queste sono memorizzate lato server su MongoDB in una collezione di oggetti Rating.

Ogni Rating può contenere più segmenti di strada, un owner, la fascia oraria alla quale la valutazione fa riferimento, il voto della valutazione (positiva, negativa, intermedia), altre informazioni quali l'illuminazione, l'affollamento, un eventuale commento associato, date di inserimento e modifica, due contatori con numero di like e dislike ottenuti per il commento scritto.

Il set dei segmenti invece, sempre lato server, è memorizzato su PostGIS. Ogni segmento ha un identificativo univoco. Su questo db stanno anche i dati aggregati relativi alla media delle valutazioni su ogni segmento.

Sul client è possibile visualizzare le valutazioni effettuate dall'utente tramite l'apposita voce nella sezione profilo.

Queste sono disposte in ordine cronologico all'interno di una lista, mostrata tramite un ***RecyclerView***, ogni elemento della lista mostra il nome della strada valutata, il commento, la data e la fascia oraria di riferimento.

Selezionando una valutazione viene mostrato il dettaglio di questa con una vista della strada sulla mappa, il numero di feedback positivi e negativi ottenuti e due PushButton, uno per la modifica e uno per l'eliminazione della valutazione.

La schermata di modifica è identica quella che ne permette l'inserimento, per questo motivo la descriverò direttamente poco più avanti nel paragrafo dedicato all'inserimento delle valutazioni.

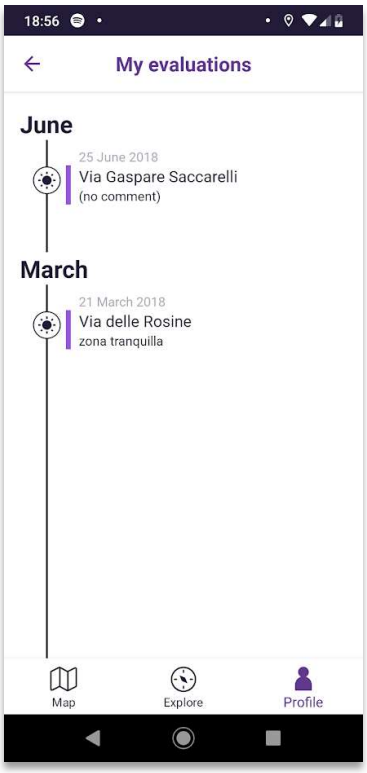

Figura 4.13: Elenco delle valutazioni effettuate

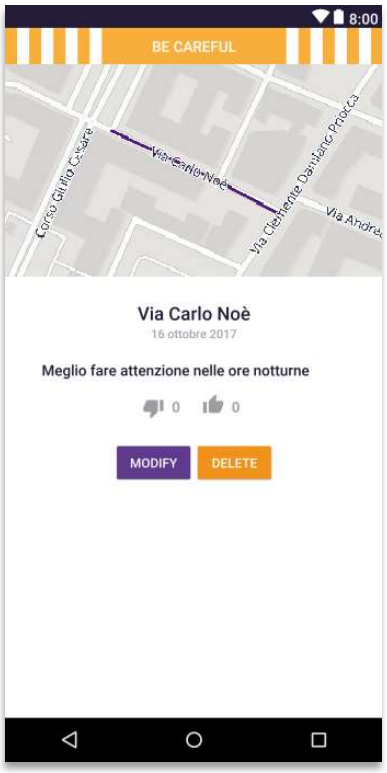

Figura 4.14: Dettaglio della valutazione

Altri importanti attributi delle valutazioni sono le traduzioni del commento, queste vengono generate dal server in modo lazy quanto un utente richiede di visualizzare il commento nella lingua del suo sistema.

È possibile tradurre qualsiasi lingua in un set di lingue prestabilite, ovvero italiano, tedesco, francese, spagnolo e inglese. Quest'ultima è la lingua di default, ovvero se viene richiesta la traduzione in una lingua non appartenente al set, viene ritornata al client la traduzione in inglese.

Per la traduzione e il rilevamento automatico della lingua del testo è stata utilizzata la Microsoft Translator Text API di Microsoft Azure. Questa API offre un servizio cloud-based e supporta numerose lingue [80].

4.6 Esplorazione delle città mappate

Tramite l'apposito tab sulla bottom bar della schermata principale è possibile accedere alla visualizzazione delle città sulle quali è possibile effettuare il mapping da parte dalla community.

Queste città sono visualizzare su una lista ordinata, sempre tramite ***`RecyclerView`***. Vengono visualizzate per prime le città dove sono state raccolte più valutazioni.

Gli elementi della lista sono realizzati con una sottoclasse di ***`CardView`***, la loro dimensione mantiene un aspect ratio di 1:1 e viene adattata in base alla dimensione dello schermo, in modo da riempire ogni riga della lista con due elementi.

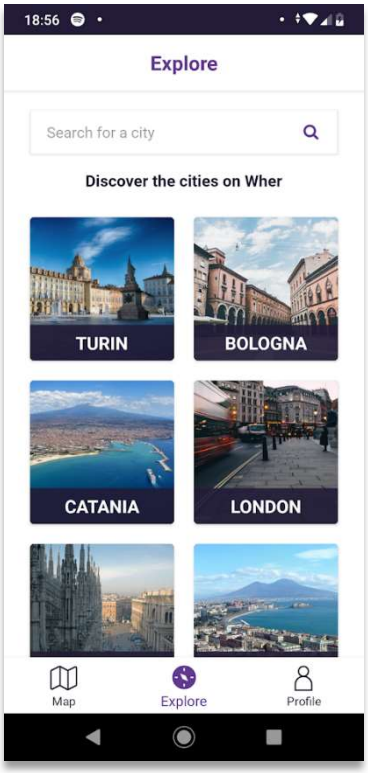

Figura 4.15: Elenco delle città

È possibile selezionare una città per passare alla vista contenente i quartieri suggeriti alle turiste che intendono visitare la città scelta.

Questa è composta da due parti ed entrambe permettono di selezionare un quartiere per visualizzarlo nel dettaglio.

La parte alta della schermata è composta da un fragment con all'interno una mappa della città, su questa mappa sono evidenziati i quartieri consigliati, è possibile selezionare un quartiere toccandolo direttamente sulla mappa.

Nella parte bassa invece vi è una lista popolata da questi quartieri. Questa è stata realizzata sempre tramite un ***RecyclerView*** al quale però è stato attaccato un ***PagerSnapHelper***. Questo componente della libreria support di Android, permette di scorrere la lista mantenendo sempre un elemento centrale ed eventualmente riposizionando al centro l'elemento più vicino nel caso in cui l'utente non l'abbia centrato manualmente, in questo modo risulta evidente che il quartiere attualmente selezionato è quello che si trova visualizzato al centro sullo schermo. Gli altri quartieri vengono lasciati fuori dalla visuale dell'utente, il quale può solo intravederne i bordi laterali ed eventualmente trascinarli verso il centro per attivare l'evidenziazione sulla mappa sovrastante del quartiere appena trascinato.

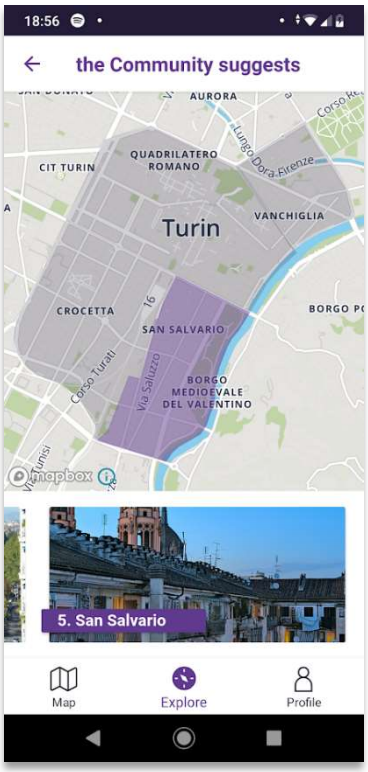

Figura 4.16: Elenco dei quartieri

I quartieri sono disegnati sulla mappa grazie ad oggetti di tipo ***PolygonOptions*** appartenenti al package ***com.mapbox.mapboxsdk.annotations*** di Mapbox. Questo tipo di oggetto contiene al suo interno un campo ***Polygon*** privato, un elemento geometrico composto da una forma chiusa i cui vertici sono delle coordinate spaziali, e mette a disposizione dei metodi per disegnarlo.

È possibile creare un nuovo oggetto ***PolygonOptions*** chiamando il suo costruttore privo di parametri e settando le caratteristiche desiderate tramite i metodi che la classe mette a disposizione, quelli utilizzati sono stati:

- ***PolygonOptions addAll(Iterable<LatLng> points)***:

Aggiunge le coordinate che andranno a definire la forma del poligono.

- ***PolygonOptions strokeColor(int color)***:

  Definisce il colore dei contorni. Il parametro ***color*** è un colore ARGB mappato su 32 bit.

- ***PolygonOptions fillColor(int color)***:

  Definisce il colore di riempimento. Il parametro ***color*** è un colore ARGB mappato su 32 bit.

- ***PolygonOptions alpha(float alpha)***:

  Definisce l'opacità del poligono. Il valore di ***alpha*** è compreso tra 0 e 1.

Tutti questi metodi ritornano l'oggetto stesso, è quindi possibile aggiungerli in cascata alla chiamata del costruttore per avere una sintassi più compatta.

È possibile aggiungere un poligono così creato alla mappa col metodo ***Polygon addPolygon(@NonNull PolygonOptions polygonOptions)*** della classe ***MapboxMap*** e associare alla mappa un listener della selezione dei poligoni col suo metodo pubblico ***void setOnPolygonClickListener(@Nullable OnPolygonClickListener listener)***.

Toccando l'elemento centrale della lista si passa alla visualizzazione del dettaglio del quartiere.

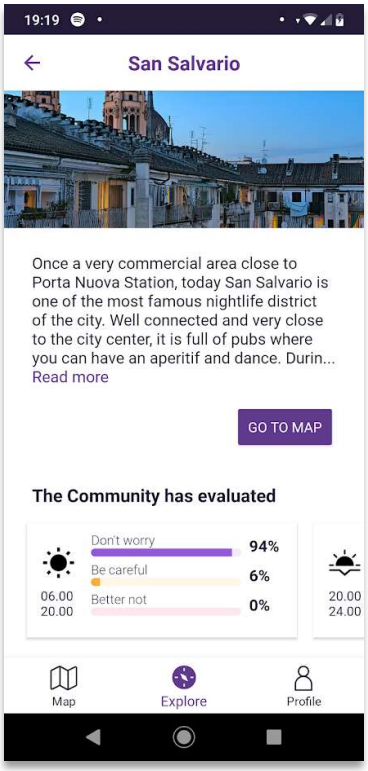

Figura 4.17: Dettaglio del quartiere

Qui viene visualizzata una breve descrizione tramite una ***ReadMoreTextView*** [81] e una media delle valutazioni in quella zona per ogni fascia oraria.

La descrizione è disponibile in italiano e in inglese. La lingua, così come quella di tutti gli altri testi dell'app viene scelta in base alla lingua del sistema.

La visualizzazione delle valutazioni relative alle diverse fasce orarie è simile alla visualizzazione della lista dei quartieri, anche questa è stata costruita con un ***RecyclerView*** e ***PagerSnapHelper***.

È anche presente un pulsante "Vai alla mappa", selezionandolo si passa alla visualizzazione del quartiere sulla mappa di Wher. Parlerò nel dettaglio della mappa nel prossimo paragrafo.

Tutte queste informazioni sono memorizzate su MongoDB in due tipi di documento:

- ***City***:

  Contenente, tra le varie informazioni, un nome, un url con un'immagine e delle coordinate spaziali.

- ***Neighborhood***:

  Ovvero i quartieri, appartenenti alle relative città. Questo oggetto contiene, tra le varie informazioni, un nome, un url con un'immagine, delle coordinate spaziali, le medie di valutazioni positive, neutre e negative per ogni fascia oraria e i commenti più apprezzati.

4.7 La mappa

Passiamo ora al componente più importante del software, quello tramite il quale le utenti interagiranno più spesso, ovvero la Mappa, realizzata grazie alle API di Mapbox.

Queste API hanno permesso l'inserimento di marker customizzati sui punti di interesse, sui segmenti di strada ai quali è stato associato un commento, sui luoghi cercati tramite la barra di ricerca e sui punti di partenza e di destinazione per la navigazione.

È stato inoltre possibile disegnare sulla mappa dei segmenti colorati sulle strade valutabili, modificare la posizione della camera e lo zoom in base alle ricerche effettuate o per mostrare la posizione dell'utente, modificare lo stile di visualizzazione dell'intera mappa in base alla fascia oraria selezionata.

Tramite la mappa l'utente sarà in grado di eseguire in modo semplice e intuitivo una serie di azioni, di cui le principali:

- Visualizzare la sicurezza delle strade.
- Inserire nuove valutazioni.
- Trovare un percorso a piedi sicuro tra due punti all'interno di una città.

Innanzitutto deve essere possibile visualizzare sulla mappa le informazioni sulla sicurezza in relazione alla fascia oraria richiesta dall'utente.

Per questo scopo è stato deciso di fornire tre diverse visualizzazioni della mappa, una diurna, una serale e una notturna, selezionabili da una barra posta in alto, immediatamente sotto alla barra di ricerca.

In base alla fascia oraria scelta cambiano le informazioni visualizzate e lo stile della mappa per dare un feedback più chiaro all'utente che, grazie ai colori capisce facilmente l'orario di riferimento senza bisogno di andare a cercare l'informazione altrove.

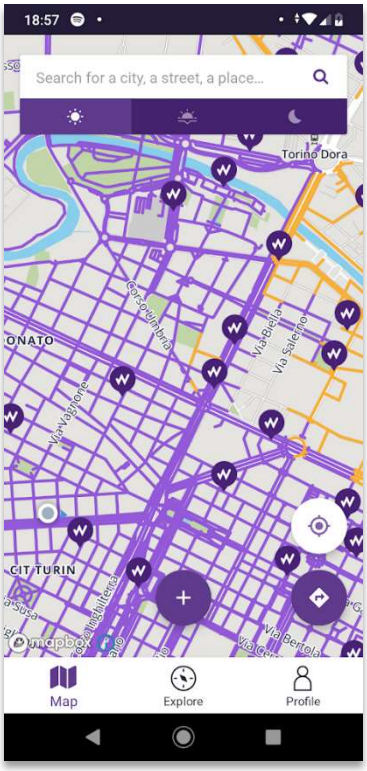

Figura 4.18: Mappa di giorno

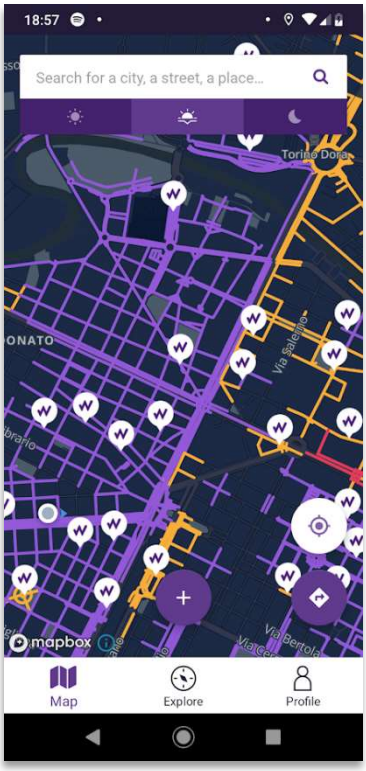

Figura 4.19: Mappa di sera

Figura 4.20: Mappa di notte

Normalmente all'interno della mappa sono visibili tre differenti tipi di informazione:

- Segmenti di strada valutati
- Commenti
- POI

A ognuna di queste tipologie è associato un ***Layer***, nel caso dei segmenti si usa un ***LineLayer*** e negli altri due casi un ***SymbolLayer***.

Il LineLayer permette di disegnare dei segmenti stabilendone a priori le proprietà come spessore e opacità. Il ***SymbolLayer*** permette invece di inserire icone o etichette testuali. Ogni layer ha bisogno di un ***Source*** ovvero di un dataset di elementi da

disegnare sulla mappa. Una volta associata la sorgente è possibile quindi rappresentarne il contenuto secondo le proprietà stabilite nel Layer.

Una feature utile dei ***Layer*** è quella di essere posizionabili secondo l'ordine stabilito dal programmatore ovvero è possibile stabilire quale ***Layer*** debba stare sopra o sotto un altro.

La mappa, un oggetto di tipo ***MapboxMap***, appartenente al package ***com.mapbox.mapboxsdk.maps***, è contenuta all'interno di un fragment denominato ***CustomMapFragment***.

Alla creazione del fragment, nel metodo ***onCreate()***, viene settata una variabile enumerativa contenente il tipo di mappa che si vuole mostrare, questa variabile viene estratta dall'oggetto di tipo ***Bundle*** ricevuto come parametro.

Questo enumerativo viene poi esaminato, subito prima della visualizzazione, nel metodo ***onCreateView()***, per decidere quali elementi (bottoni, layer, listener, etc.) attivare e quali no, in base al tipo di informazioni che l'utente ha richiesto di visualizzare.

I possibili valori di questa variabile sono:

- ***evaluate***:

  Visualizzazione di classica con i segmenti colorati. Il valore di default.

- ***pickComment***:

  Visualizzazione nella quale sono colorate esclusivamente le strade appena selezionate dall'utente durante una valutazione.

- ***evaluateRoute***:

  Vengono nascoste tutte le valutazioni presenti sulla mappa permettendo all'utente di disegnare il suo percorso da valutare.

- ***viewRating***:

  Viene mostrata una visuale comprendente una singola valutazione effettuata precedentemente, tutte le altre non vengono visualizzate. La telecamera è bloccata su quella posizione.

- ***cityDetail***:

  Viene visualizzata una singola citta e bloccata la telecamera in modo tale da mostrarla tutta. Vengono evidenziati i quartieri suggeriti dalla community.

4.7.1 Il significato dei colori

I valori della sicurezza di un segmento di strada vengono forniti attraverso tre diversi colori. Per il calcolo del colore viene effettuata una media di tutte le valutazioni presenti in un determinato punto.

Se consideriamo i valori "non sicuro", "mediamente sicuro" e "sicuro" come tre numeri interi a distanza fissa l'uno dal successivo, ad esempio 1, 2 e 3, è possibile

calcolare la media di questi valori per ogni valutazione presente in un particolare segmento di strada e in una particolare fascia oraria. Questa media viene quindi discretizzata all'interno di un intervallo e gli viene associato un colore.

I colori volevano risultare intuitivi e al tempo stesso poter essere immediatamente associabili ai colori dell'app, e stato scelto quindi di partire dal modello del semaforo, ovvero rosso, giallo e verde, e sostituire i colori con quelli dell'app cercando di non stravolgerne il significato, quindi le associazioni sono state:

- Rosso (***#E63C5C***): Meglio non percorrere la strada da sola.
- Giallo (***#F8AF3B***): Percorri la strada con attenzione.
- Viola (***#9359D9***): Procedi tranquillamente.

Figura 4.21: I colori delle valutazioni

4.7.2 Visualizzazione nel dettaglio delle valutazioni

Selezionando un segmento colorato o una icona commento (i pallini con il simbolo di Wher) si passa al dettaglio di quel segmento (o del segmento nel quale è posto il commento).

Figura 4.22: Dettaglio di giorno

Figura 4.23: Dettaglio di sera

Figura 4.24: Dettaglio di notte

In questa nuova schermata viene visualizzato su un ***RecyclerView*** in alto, un elemento con la percentuale di valutazioni positive, medie e negative per il segmento di riferimento e per una determinata fascia oraria, identificata da un disegno sulla sinistra dell'elemento.

Anche a questo ***RecyclerView*** è associato un ***PagerSnapHelper*** per mantenere uno degli elementi della lista al centro e, spostando gli elementi della lista, è possibile cambiare la fascia oraria di riferimento.

Se sono presenti dei commenti, questi vengono visualizzati nella lista sottostante, anche questi cambiano in base alla fascia oraria.

È possibile visualizzare i commenti tradotti, indipendentemente dalla lingua in cui sono stati scritti, in alcune lingue prestabilite ovvero italiano, inglese, tedesco, francese e spagnolo.

Per attivare la traduzione basta selezionare la label in basso a sinistra sotto un qualsiasi commento, in questo modo verranno automaticamente tradotti, nella lingua impostata nel telefono (o in inglese se la lingua del sistema non fa parte del subset descritto precedentemente), tutti i commenti visualizzati. Per ripristinare i commenti in lingua originale basta riselezionare la label.

Questa scelta viene memorizzata nelle ***SharedPrefrences*** e rimane valida finché l'utente rimane loggato.

La traduzione dei commenti avviene sul server grazie all'utilizzo dell'API di Microsoft mstranslator. Per utilizzarla basta dichiarare la classe MsTranslator con una sintassi del tipo

```
var MsTranslator = require('mstranslator')
```

e in seguito istanziarne un oggetto con la sintassi

```
var translator = new MsTranslator({api_key: "..."}, true)
```

passando come argomenti la chiave dell'API e un valore booleano per indicare se si vuole utilizzare la funzionalità di auto refresh del token [82].

Una volta istanziato il traduttore, i due metodi utilizzati sono stati:

- ***detect()***:

  Serve a determinare la lingua in cui è stato scritto il testo. A questo metodo viene passato come parametro un oggetto JSON con un campo ***text*** contenente il commento o un suo estratto.

- ***translate()***:

  Traduce effettivamente il testo nel linguaggio richiesto. Come parametro accetta un oggetto JSON contenente un campo ***text*** col testo da tradurre, un campo ***from*** e uno ***to***, contenenti rispettivamente la lingua originale e la lingua in cui si desidera tradurre.

I commenti possono essere valutati in modo anonimo dalla community tramite pollice in giù e pollice in su, questo serve a fornire un feedback a tutti gli altri utenti che visualizzeranno quel commento in quanto vedranno questi due indicatori sul gradimento sempre aggiornati.

Sopra ogni commento è presente anche una breve descrizione dell'utente che l'ha scritto e la data di inserimento. Dell'utente viene visualizzata la foto del profilo (se abilitata), il nickname, l'età, la città in cui vive, i mezzi che usa solitamente per gli spostamenti in città (rappresentati da delle piccole icone) e il livello dell'utente.

4.7.3 Gli elementi sopra la mappa

Su un layer al di sopra della mappa sono posti ulteriori elementi con cui l'utente può interagire.

Abbiamo già visto che è presente una barra di selezione della fascia oraria e immediatamente sopra questa una barra di ricerca, quest'ultima permette di cercare un luogo preciso ed in seguito eventualmente avviare una navigazione.

Sono anche presenti dei floating action button (FAB), ovvero i bottoni rotondi che qui permettono di inserire una nuova valutazione partendo dalla selezione delle strade sulla mappa, centrare la mappa sulla posizione attuale, mostrare un percorso sicuro tra due punti (modalità navigazione).

Quando l'utente seleziona il fab per trovare la sua posizione, la visuale rimane ancorata a quest'ultima e la segue durante gli spostamenti. La visuale smette di seguire

la posizione dell'utente quando viene spostata manualmente trascinando il dito sullo schermo o effettuando una ricerca. Questa funzione di tracking risulta utile principalmente in fase di navigazione.

I FAB sono stati inseriti grazie alla libreria android-fab di Mark Ormesher [83].

4.7.4 Modalità ricerca

Tramite la barra di ricerca l'utente ha la possibilità di cercare un indirizzo o punto di interesse preciso.

Per supportare questa funzione è stata prevista l'integrazione con una funzione di autocompletamento dei luoghi fornita da Google.

Viene quindi mostrato un fragment con un ***EditText*** per l'inserimento testuale a cui è associato un oggetto ***TextWatcher***. Questo ***TextWatcher***, ogni volta che l'utente modifica il contenuto dell'***EditText***, contatta i server di Google mandando una richiesta di place autocomplete.

La richiesta viene effettuata con la relativa query RESTful grazie alla libreria retrofit2. In particolare viene inviata una richiesta GET all'URL

https://maps.googleapis.com/maps/api/place/autocomplete/json

fornendo come query params la chiave dell'API, la stringa di input, il tipo di dato richiesto, la lingua nella quale si desidera ricevere il risultato e la posizione dell'utente.

La risposta viene ricevuta in modo asincrono e contiene un oggetto JSON, da questo vengono estratti i campi relativi all'id e alla descrizione (nome della via, del monumento, etc.). Questi elementi vengono quindi utilizzati per riempire una lista dei suggerimenti.

Per visualizzare sulla mappa il posto cercato si può quindi selezionare uno degli elementi della lista oppure premere invio dalla tastiera virtuale per selezionare automaticamente il primo (o l'unico) elemento della lista.

Se l'***EditText*** è vuoto non viene invece visualizzato alcun suggerimento ma viene visualizzata una lista contenente gli ultimi cinque luoghi cercati, questi vengono ogni volta aggiornati e memorizzati nelle ***SharedPreferences*** serializzandoli in una stringa come un unico oggetto JSON. Entrambe le liste vengono visualizzate sullo schermo, in mutua esclusione, grazie ad un ***RecyclerView***.

Una volta selezionato un posto, questo viene visualizzato sulla mappa tramite un marker colorato e posizionato al centro dell'inquadratura.

Parallelamente si apre una tendina in basso con il nome del luogo cercato e un pulsante, tramite il quale è possibile attivare la navigazione per raggiungere quella destinazione.

Tra il campo ***EditText*** e il ***RecyclerView*** con la lista dei suggerimenti (o delle ultime ricerche) vi è una ***TextView*** con scritto il nome della posizione attuale, è anche possibile spostarsi su questa posizione selezionandola dalla schermata.

La posizione attuale viene ottenuta tramite il location engine di Mapbox, questa viene richiesta automaticamente quando viene aperta la schermata di ricerca.

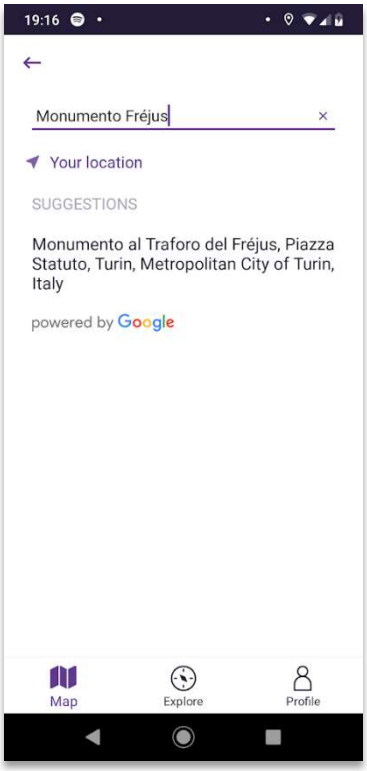

Figura 4.25: Schermata ricerca

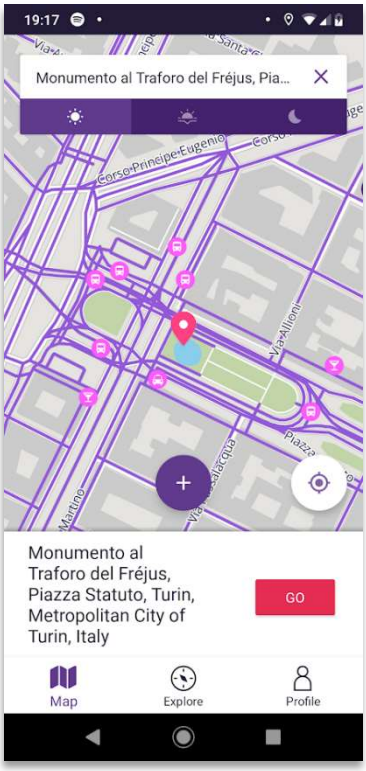

Figura 4.26: Risultato ricerca

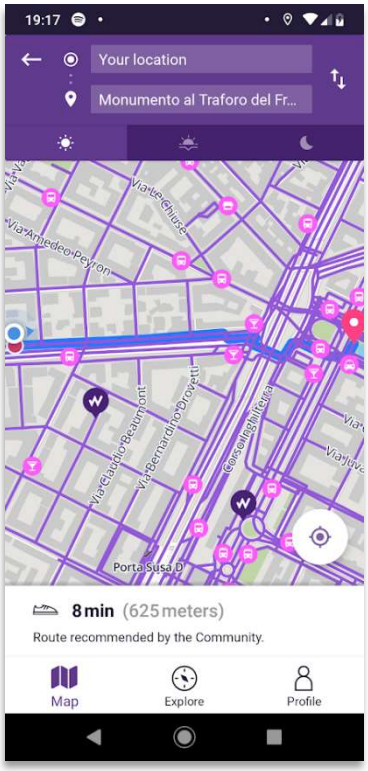

Figura 4.27: Navigazione

4.7.5 I punti di interesse

È possibile selezionare i punti di interesse (POI) sulla mappa per vederne comparire una breve descrizione nel popup in basso.

I POI evidenziati sulla mappa sono:

- Fermate dell'autobus
- Locali
- Parcheggi dei taxi
- Parcheggi rosa
- Associazioni femminili
- Negozi aperti 24 h su 24

Figura 4.28: I POI

Selezionando un POI relativo a una fermata dell'autobus, vengono visualizzate nel popup anche le linee che passano da quella fermata.

Queste sono disposte su un ***FlexboxLayout*** [84], un componente di Android che imita le caratteristiche del modulo CSS Flexible Box Layout [85]. In questo modo, poiché non è possibile conoscere a priori quante saranno le linee di bus, queste vengono posizionate in modo flessibile occupando solo lo spazio necessario.

Una volta selezionato un POI, tramite il popup in fondo allo schermo, è possibile avviare la modalità di navigazione per scoprire come raggiungerlo.

4.7.6 Modalità navigazione

Innanzitutto è possibile selezionare il punto di partenza e di destinazione dalle due ***TextView*** posizionate in alto sullo schermo, di default viene impostata come partenza la posizione attuale e come destinazione quella del posto cercato (se la navigazione viene avviata a partire da una ricerca).

È anche possibile scambiare le due posizioni tramite l'icona con le frecce a destra.

In questa schermata viene quindi visualizzato il percorso consigliato (ossia il percorso ritenuto più sicuro) dalla posizione di partenza alla destinazione.

Il percorso cambia in base alla fascia oraria selezionata, in quanto le strade sicure cambiano in base a questa, è possibile modificare la fascia oraria direttamente dalla modalità di navigazione tramite gli appositi pulsanti, in questo modo il percorso viene aggiornato sullo schermo.

In basso viene anche mostrata su una finestra a comparsa ridimensionabile la distanza dalla destinazione, il tempo stimato per raggiungerla e se sul percorso sono presenti dei tratti poco sicuri.

Il percorso viene mostrato con un tratto blu su uno specifico ***LineLayer*** posto al di sotto del layer delle valutazioni.

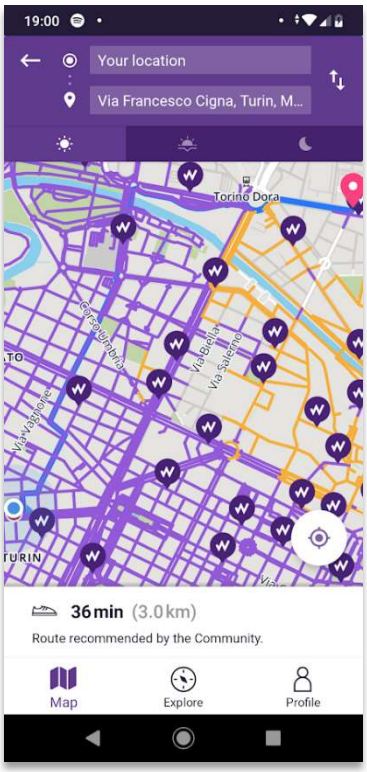

Figura 4.29: Percorso di giorno

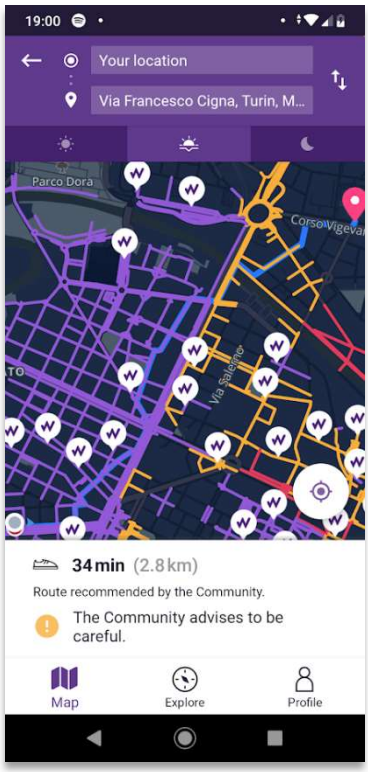

Figura 4.30: Percorso di sera

4.7.7 Inserimento delle valutazioni

Lo strumento più importante delle Wherrior è proprio l'inserimento delle valutazioni. Queste vengono inserite direttamente dalla mappa, una volta che questa viene settata in modalità inserimento tramite la pressione dell'apposito FAB da parte dell'utente.

In questo modo è possibile generare una nuova valutazione. Si possono selezionare le strade da valutare toccandole sullo schermo e, dopo aver completato la selezione, si può compilare un form dove viene richiesto di valutare la strada selezionata secondo certi criteri prestabiliti.

Figura 4.31: Inserimento valutazione

Figura 4.32: Strade selezionate

Viene richiesto di descrivere la sicurezza percepita dall'utente. Alcuni criteri di valutazione sono più soggettivi mentre altri più oggettivi, questi criteri possono cambiare in base alla fascia oraria selezionata, ad esempio la valutazione sull'illuminazione dell'ambiente viene richiesta solo per la fascia oraria serale e per quella notturna.

All'utente viene chiesto di riempire i seguenti campi relativi alle strade selezionate:

- Scegli l'orario a cui ti riferisci
- Quanto è illuminato?
- Quanto è affollato?
- Consiglieresti di passare da questo luogo?
- Aggiungi un commento

Figura 4.33: Valutazione diurna

Figura 4.34: Valutazione notturna

L'unico inserimento testuale viene richiesto per il commento. Tutti gli altri campi richiedono come risposta una opzione (nel caso della fascia oraria) oppure un valore all'interno di un intervallo, per questi ultimi è stata implementata una classe che estende il funzionamento della ***AppCompatSeekBar*** della libreria support di Android.

4.8 Altre funzioni

Un'altra funzione interessante da la possibilità di compilare dei questionari per fornire aiutare gli sviluppatori a migliorare l'app.

Nella sottosezione aiuto della sezione profilo invece sono presenti, oltre alle voci Tutorial e Contact us, due voci che permettono di richiedere i dati personali ed eliminare il proprio account e quindi tutti i dati relativi ad esso che si trovano sul server di Wher.

Queste ultime due voci sono state inserite per rispettare il regolamento generale sulla protezione dei dati (GDPR), diventato operativo a partire dal 25 maggio 2018 [86].

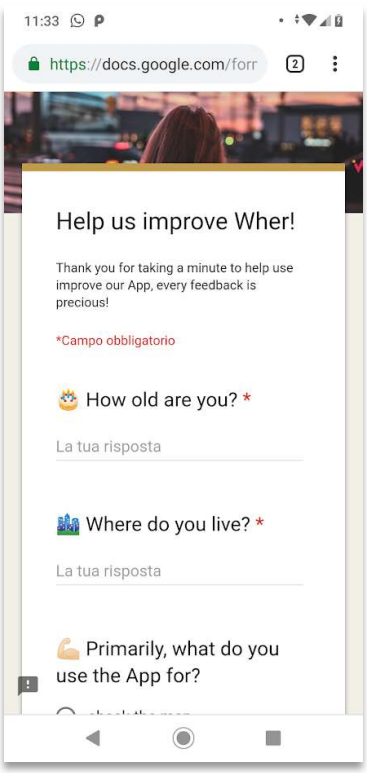

Figura 4.35: Questionario

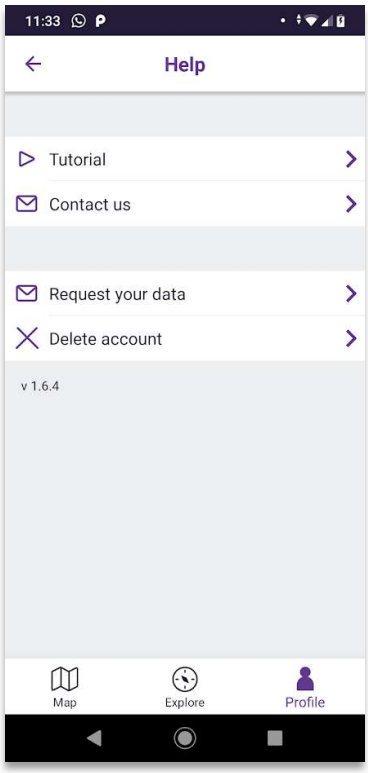

Figura 4.36: Sottosezione aiuto

4.9 Raccolta dei feedback

Per verificare il corretto funzionamento dell'app e sondare le modalità di interazione dell'utente, sono stati sfruttate due soluzioni messe a disposizione da Firebase, ovvero Firebase Crashlytics e Google Analytics.

La prima, Crashlytics, è un reporter leggero e real-time di crash. In particolare permette di vedere quando l'applicazione crasha sullo smartphone di un utente, mostrando anche la riga di codice e lo stack trace nel punto in cui si è verificato il bug. Rende possibile sapere se un particolare bug sta venendo riscontrato da molti utenti, genera avvisi quando un problema aumenta improvvisamente di gravità e altro ancora [87].

La seconda invece, Analytics, fornisce informazioni sull'utilizzo dell'app [88]. Una volta ottenuta l'interfaccia del singleton col metodo statico ***`FirebaseAnalytics getInstance(Context context)`***, è possibile loggare un evento col metodo ***`void logEvent(String name, Bundle params)`*** e ancora tracciare le azioni di un utente impostando un suo id o delle user properties rispettivamente con i metodi ***`void setUserId(String id)`*** e ***`void setUserProperty(String name, String value)`*** [89].

Questi servizi si sono rivelati utilissimi ad esempio per vedere se, dopo un aggiornamento, sono state utilizzate le nuove feature introdotte oppure per ricevere

immediatamente dei report in caso di presenza di bug e poterli risolvere tempestivamente.

Capitolo 5

Limiti del prototipo sviluppato e soluzioni

5.1 Limitazioni degli strumenti

Negli ultimi anni, tra gli strumenti di sviluppo per Android sono apparse interessanti novità, una tra queste è Android Jetpack ovvero una suite di librerie, strumenti e regole per aiutare gli sviluppatori a scrivere codice di alta qualità in modo più semplice [90].

Tra gli elementi più importanti di questa suite vi è il ***`ViewModel`***, una classe progettata per immagazzinare e gestire dati relativi alla UI in modo consapevole del lifecycle, permettendo ai dati di sopravvivere ai cambiamenti di configurazione come ad esempio le rotazioni dello schermo [91].

Un altro elemento interessante è la Room Persistence Library che crea un livello di astrazione sopra SQLite per permettere un accesso al database più robusto [92].

Una migrazione a questa suite e quindi l'utilizzo di queste librerie, come di diverse altre, avrebbe potuto alleggerire lo sviluppo rendendo il codice più semplice e sicuro.

Un'altra possibile migrazione che si sarebbe potuta attuare è quella del linguaggio di programmazione da Java a Kotlin.

Kotlin è un linguaggio sviluppato da JetBrains, capace di interoperare anche con classi Java, che prende molto spunto da questo linguaggio, superando però alcune limitazioni e criticità. Questo fatto ha favorito la conversione di molte librerie Android al linguaggio Kotlin in quanto evita alcune classi di errore come ad esempio gli accessi ai puntatori nulli [93].

5.2 Limitazioni nello sviluppo

La scelta di scrivere l'app utilizzando codice nativo porta sicuramente dei benefici, ad esempio spesso può rende il software più performante e garantire un maggior controllo del sistema ma in certi contesti ha anche degli svantaggi in quanto, sviluppare lo stesso prodotto per due sistemi operativi diversi come Android e iOS richiede il doppio del lavoro rispetto allo sviluppo su un singolo sistema.

Esistono numerose alternative per scrivere codice multipiattaforma una sola volta in grado di funzionare in modo efficiente su ambienti diversi. Una tra queste è Flutter.

Flutter è un framework open-source sviluppato da Google per creare applicazioni per Android e iOS. Il componente Flutter engine, scritto principalmente in C++, oltre a fornire il supporto per il rendering a basso livello utilizzando la libreria grafica di Google, Skia Graphics, si interfaccia con gli SDK della piattaforma specifica [94].

La scelta di un framework di questo tipo può rendere meno dispendioso, a livello di risorse economiche e di tempo utilizzato, lo sviluppo di applicazioni specialmente in piccole realtà come quella di una startup.

È infatti la scelta fatta dal team di Wher negli ultimi mesi, che ha permesso al team di ottimizzare i tempi e dedicarsi parallelamente a nuove feature.

5.3 Limitazioni della gamification

L'approccio adottato per la progettazione della gamification è molto legato alle teorie sul comportamentismo e alla triade PBL piuttosto che alle teorie sul cognitivismo. Questo meccanismo come abbiamo visto è molto basilare e fa molta leva sul concetto di ricompensa senza però studiare il giocatore e i suoi interessi.

La presenza di una classifica dove vengono visualizzate le Wherrior più attive potrebbe demotivare le utenti che usano l'app da poco e sono ancora lontane dal raggiungere un certo livello. Una alternativa sarebbe potuta essere quella di mostrare soltanto una classifica relativa agli amici.

Un altro miglioramento potrebbe essere fatto sul fronte del raggiungimento degli obiettivi, di questi è sempre possibile visualizzarne l'elenco con la percentuale di completamento. L'utente in questo modo sa già quando raggiungerà l'obiettivo. Rendendo invece inaspettata la conquista di una ricompensa (non necessariamente con l'attuale meccanismo che conta il numero esatto di task eseguiti), questo traguardo può risultare più appagante.

In un possibile design successivo si potrebbe ad esempio passare gradualmente dall'approccio comportamentista a uno più cognitivista, cercando di trasformare le ricompense in intrinseche invece di estrinseche e facendo più leva sul fattore social, la condivisione con i propri amici dei traguardi raggiunti, etc.

5.4 Le informazioni sulla mappa

Come è già stato spiegato prima, sulla mappa, le valutazioni vengono viste come segmenti colorati. Questo significa che in ogni strada colorata è presente almeno una valutazione, ma cosa succede quando in una strada non sono presenti valutazioni?

Al momento questi segmenti rimangono invisibili, non forniscono nessuna informazione, quindi all'interno di una città possono essere presenti dei "buchi". Questi vengono parzialmente "tappati" durante la navigazione in quanto i segmenti non valutati vengono considerati dall'algoritmo come segmenti gialli per il calcolo del percorso.

Questa ovviamente è una soluzione molto approssimativa. Una possibile soluzione più efficace potrebbe essere quella di far identificare i segmenti vuoti dal sistema come segmenti colorati, calcolando un ipotetico livello di sicurezza a partire dai segmenti colorati adiacenti o più vicini, magari utilizzando un algoritmo di machine learning.

Un altro limite è quello del numero delle valutazioni. Spesso le utenti non sono portate a valutare una strada già valutata più volte, specialmente se sono d'accordo con la valutazione complessiva, questo porta ad accumulare a volte valutazioni un po' datate che non vengono sostituite con valutazioni nuove, dunque la media che viene effettuata su ogni segmento di strada coinvolge tutte le valutazioni indipendentemente dalla data in cui sono state effettuate.

Una possibilità potrebbe essere quella di rimuovere, non considerare o considerare come meno importanti le valutazioni più vecchie, eventualmente anche rimuovendole dalla mappa, per spingere le utenti a rimappare le stesse strade e ricalcolare la media aggiornata più influenzata quindi dalle valutazioni più recenti. Questa potrebbe anche essere influenzata dai feedback ricevuti tramite "pollice in giù" e "pollice in su".

Ovviamente dei cambiamenti visibili del genere, comporterebbero delle fasi di transizione, in cui l'utente deve adattarsi alla novità, e di testing per vedere se effettivamente le modifiche introdotte migliorano o meno l'usabilità del sistema.

Capitolo 6
Conclusioni

L'applicazione sviluppata in questo lavoro di tesi non costituisce la vera soluzione al problema, ovvero le aggressioni e gli atti criminali rivolti verso le donne da estranei.

Funge dunque in qualche modo da pezza in quanto il problema è molto complesso e ha radici profonde nella società in cui oggi viviamo dove la criminalità e il sessismo sono elementi difficili da sradicare e il fatto stesso che diverse donne sentano la necessità di un servizio del genere ne è la conferma.

Abbiamo visto che le alternative attualmente sul mercato lavorano in modo reattivo, non impedendo la comparsa delle minacce ma cercando di allontanarle utilizzando in modo tempestivo gli strumenti che mettono a disposizione.

Wher al contrario sceglie un approccio preventivo evitando, per quanto possibile, le minacce.

L'utilizzo di un pattern di progettazione iterativo ha permesso di aggiustare il tiro più volte durante le fasi di testing dopo lo sviluppo di ogni prototipo, mostrando ogni volta gli elementi da migliorare per ottimizzare l'esperienza dell'utente.

L'inserimento della gamification si è rivelato spesso utile nelle fasi iniziali in cui l'utente si approccia all'app ma presenta certe limitazioni anche a causa della natura estrinseca delle ricompense e del sistema dei punti abbastanza complesso.

Il sistema utilizza un server Linux remoto e come dispositivo terminale lo smartphone dell'utente. Si è inoltre appoggiato a servizi esterni come Mapbox e Google Maps.

L'architettura realizzata, anche grazie al paradigma REST è abbastanza flessibile ai cambiamenti e si presta bene ipoteticamente anche per una versione web.

Sono state create delle modalità di esplorazione libera della mappa, di ricerca di un luogo e di navigazione. Attraverso la mappa è stato possibile visualizzare la percezione di sicurezza dei vari luoghi e anche inserire le proprie valutazioni. Queste informazioni variano in base alla fascia oraria di riferimento selezionabile dall'utente.

Sono state anche inserite funzionalità più turistiche che mostrano per ogni città mappabile i quartieri consigliati dalla community e la sicurezza percepita in questi quartieri.

L'applicazione, nel periodo nel quale vi ho lavorato, ha visto tanti cambiamenti e ne sta vedendo ancora adesso. È un prodotto in continua evoluzione che si spera possa continuare a migliorare la qualità della vita delle donne in questa società finché sarà necessario.

Elenco delle tabelle

Elenco delle figure

Bibliografia

[1] “bes 2018: Il benessere equo e sostenibile in Italia” – Istat: Istituto Nazionale di Statistica - ISBN 978-88-458-1967-4 - https://istat.it/it/files/2018/12/Bes_2018.pdf

[2] “Race, Income, Gender and Safety in America” – Articolo di Sofia Kluch - Gallup Blog - 13 Luglio 2018 - https://news.gallup.com/opinion/gallup/236972/race-income-gender-safety-america.aspx

[3] “G7 Women Need Safety to Make Gender Inequality History” – Articolo di Sofia Kluch e Jessi Gordon – Gallup Blog - 19 Giugno 2018 - https://news.gallup.com/opinion/gallup/235475/women-need-safety-gender-inequality-history.aspx

[4] “La città si*cura. L'approccio di genere alla sicurezza urbana: manuale di interventi sulla città per la sicurezza delle donne e delle persone più vulnerabili” - Progetto realizzato dalla Consulta delle Elette del Piemonte - A cura di: Marita Peroglio, Luisella Dughera, Giulia Melis – 2012

[5] "Molestie morali. La violenza perversa nella famiglia e nel lavoro" - Marie-France Hirigoyen - Einaudi 2005, ISBN 88-06-17505-X, ISBN 978-88-06-17505-4

[6] "Sottomesse. La violenza sulle donne nella coppia" - Marie-France Hirigoyen - Einaudi 2006, ISBN 88-06-17974-8, ISBN 978-88-06-17974-8

[7] “Violenza psichica endo-familiare, plagio della vittima e rimedi terapeutici” Articolo di Domenico Chindemi e Valeria Cardile - Estratto dalla rivista Responsabilità Civile e Previdenza, Fascicolo n. 3/2007, Giuffrè Editore

[8] "Londra, donne che odiano le donne" - Articolo di Francesca Marretta per "Liberazione"

[9] "Indagine shock sulla violenza sessuale: per il 55% degli italiani è colpa delle vittime troppo attraenti" - Dichiarazioni del presidente dell'Airs, Franco Avenia

[10] “La teoria della broken window” - Articolo di Giovanni Marco Celia del 05/02/2009 - https://sociologia.tesionline.it/sociologia/articolo.jsp?id=3094

[11] “Prevenzione della criminalità attraverso il disegno urbano. Valutazione del rischio criminalità e proposte di intervento per un quartiere di Trento.” - Tesi di Giovanni Marco Celia – Università degli studi di Trento – 2007-08 - https://sociologia.tesionline.it/sociologia/tesi.jsp?idt=24709

[12] “APP Sicurezza, cinque proposte per le donne” – Articolo di Flaminia Scarcia su The Walkman – 2015 - http://www.thewalkman.it/app-sicurezza-cinque-proposte-le-donne/

[13] “Le migliori app sicurezza donne del 2019” – Articolo di Elena Gallina- https://www.qualescegliere.it/app-sicurezza-donne/

[14] “8 marzo: 15 app pensate per la libertà (e la sicurezza) delle donne” – Articolo di Rossana Caviglioli su iO Donna – 6 Marzo 2017 -

https://www.iodonna.it/attualita/famiglia-e-lavoro/2017/03/06/8-marzo-2017-15-app-pensate-per-la-liberta-delle-donne/

[15] Kitestring - https://www.kitestring.io/

[16] securWoman 2.0 - http://www.securwoman.com/

[17] "Siamo sicure: la App ideata per la sicurezza delle donne!" – Articolo di Carlotta su Viaggiare al femminile - http://www.viaggiarealfemminile.com/siamo-sicure-la-app-ideata-per-la-sicurezza-delle-donne/

[18] SHAW - http://www.appshaw.it/

[19] bsafe - https://getbsafe.com/

[20] "Il 112 diventa un'app: cos'è e come funziona Where Are U?" – Articolo di Giuseppe Tripodi su mobileworld - 13/02/2019 - https://www.mobileworld.it/2019/02/13/112-app-where-are-u-207732/

[21] Where Are U - https://where.areu.lombardia.it/index.html

[22] "Stalking Buster, app anti aggressioni: basta un clic per inviare la propria posizione ai Carabinieri" – Articolo di Luisiana Gaita su il Fatto Quotidiano – 14 Giugno 2016 - https://www.ilfattoquotidiano.it/2016/06/14/stalking-buster-app-anti-aggressioni-basta-un-clic-per-inviare-la-propria-posizione-ai-carabinieri/2828935/

[23] Stalking Buster App - http://www.fdmonlus.it/progetti-2/stalking-buster-app/

[24] "UX — A quick glance about The 5 Elements of User Experience" – Articolo di Omar Elgabry su Medium – 15 Settembre 2016 - https://medium.com/omarelgabrys-blog/ux-a-quick-glance-about-the-5-elements-of-user-experience-part-2-a0da8798cd52

[25] "The Elements of User Experience: User-Centered Design for the Web and Beyond (2nd Edition)" – Jesse James Garrett.

[26] ISO 9241-210:2019 - https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-2:v1:en

[27] "Untangling UX, part 1: Design Thinking vs UCD" – Articolo di Marine Barbaroux su Cambridge Consultants – 15 Gennaio 2016 - https://www.cambridgeconsultants.com/insights/untangling-ux-part-1-design-thinking-vs-ucd

[28] "UCD vs UX: What's the difference?" – Articolo di Emily Grace Adiseshiah su JustInMing – 8 Giugno 2018 - https://www.justinmind.com/blog/ucd-vs-ux-whats-the-difference/

[29] "FACILE DA USARE: Una moderna introduzione alla ingegneria dell'usabilità" - Roberto Polillo - Edizioni Apogeo - Giugno 2010

[30] "Gamification e obiettivi principali" - Articolo pubblicato da Gamification Staff il 22 Dicembre 2010 su Gamification.it - http://www.gamification.it/gamification/gamification-e-obiettivi-principali/

[31] "For The Win: How Game Thinking Can Revolutionize Your Business" – Kevin Werbach, Dan Hunter – Wharton Digital Press – 2012

[32] “A Meta-Analytic Review of Experiments Examining the Effects of Extrinsic Rewards on Intrinsic Motivation" - Deci, Koestner and Ryan - 125 Psychological Bulletin 627 (1999)

[33] “Hedonic relativism and planning the good society" – Brickman, Campbell (1971) - New York: Academic Press. pp. 287–302. in M. H. Apley, ed., Adaptation Level Theory: A Symposium, New York: Academic Press

[34] “Hedonic Treadmill" – James Chen – 17 Luglio 2019 - https://www.investopedia.com/terms/h/hedonic-treadmill.asp

[35] android studio - https://developer.android.com/studio

[36] Android - https://developer.android.com/

[37] “Android 5.0: addio Dalvik, ART sarà la runtime di default nel prossimo major update" – Articolo di Candido Romano su International Business Times - 19/06/2014 - https://web.archive.org/web/20150108094108/http://it.ibtimes.com/articles/67539/20140619/android-5-0-dalvik-art-aosp-commit-fine-novita.htm#

[38] “ART sostituisce Dalvik in Android: cosa succede alle apps?" – Articolo di Gaetano Abatemarco su ChimeraRevo – 17 Ottobre 2014 - http://www.chimerarevo.com/android/art-sostituisce-dalvik-in-android-succede-alle-apps-178458/

[39] “Application Fundamentals" – Android Developers documentation - https://developer.android.com/guide/components/fundamentals

[40] “The Java Language Environment" - https://www.oracle.com/technetwork/java/intro-141325.html

[41] “Switching to Mapbox: the Less Expensive Google Maps Alternative" – Articolo di Catherine Kleimeier su Masuga – 22 Agosto 2018 - https://gomasuga.com/blog/mapbox-google-maps-alternative

[42] mapbox – https://www.mapbox.com/

[43] “I principi dell'architettura REST" – Articolo di Andrea Chiarelli su html.it – 23 Novembre 2011 - https://www.html.it/pag/19596/i-principi-dellarchitettura-restful/

[44] “RESTful Web Services" – Articolo di Nicolas Zozol su Developpez.com – 29 Maggio 2008 - https://nicolas-zozol.developpez.com/tutorial/java/rest-jsp-english/

[45] “Introducing JSON" - https://json.org/

[46] The GeoJSON Format – RFC7946 - https://tools.ietf.org/html/rfc7946

[47]" Geocoding" - mapbox Docs - https://docs.mapbox.com/help/how-mapbox-works/geocoding/

[48] “Developer Guide" – Google Maps Platform - https://developers.google.com/maps/documentation/geocoding/intro

[49] “Place Autocomplete" – Google Maps Platform – Places API - https://developers.google.com/places/web-service/autocomplete

[50] Visual Studio Code - https://code.visualstudio.com/

[51] IntelliSense – Visual Studio Code - https://code.visualstudio.com/docs/editor/intellisense

[52] Node.js - https://nodejs.org/it/

[53] Swagger - https://swagger.io/

[54] swagger-node di Swagger su GitHub - https://github.com/swagger-api/swagger-node

[55] "Cos'è un database relazionale" - Articolo di Giuseppe Pupita su FileMaker Guru – 09/04/2014 - https://www.filemakerguru.it/cose-un-database-relazionale/

[56] "SQL e NoSQL a documenti: il confronto" – Articolo di Giuseppe Maggi su html.it – 6 Marzo 2015 - https://www.html.it/articoli/sql-e-nosql-a-documenti-il-confronto/

[57] "SQL e NoSQL: cosa sapere sui database non relazionali" – Articolo di Giovanni Barbieri su blog.artera.net – 24 Giugno 2014 - https://blog.artera.net/programmazione/sql-nosql-database-non-relazionali

[58] PostGIS - https://postgis.net/

[59] PostgreSQL - https://www.postgresql.org/

[60] MongoDB - https://www.mongodb.com/

[61] "Introduzione a MongoDB" – Articolo di Paolo Gerini su Alla fine del palo – 18 Aprile 2017 - http://www.allafinedelpalo.it/introduzione-a-mongodb/

[62] "Class: Mongoid::Fields::ForeignKey" – Mongoid API - https://docs.mongodb.com/mongoid/5.2/api/Mongoid/Fields/ForeignKey.html

[63] "$lookup (aggregation)" – mongoDB Documentation - https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/

[64] "ACID Transactions in MongoDB" - https://www.mongodb.com/transactions

[65] "MongoDB Drops ACID" – Articolo di Eliot Horowitz (CTO and Co-Founder) su mongoDB Blog – 15 Febbraio 2018 - https://www.mongodb.com/blog/post/multi-document-transactions-in-mongodb

[66] mongoose - https://mongoosejs.com/

[67] Sketch - https://www.sketch.com/

[68] Adobe XD - https://www.adobe.com/it/products/xd.html

[69] "What Are The Benefits of MVC?" – 9 Dicembre 2008 - http://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/

[70] BottomBar di Iiro Krankka (roughike) su GitHub - https://github.com/roughike/BottomBar

[71] BottomNavigationView – Android Developers - https://developer.android.com/reference/android/support/design/widget/BottomNavigationView

[72] “RecyclerView: dietro le quinte” – Articolo di Antonio Tedeschi su html.it – 27 Dicembre 2018 - https://www.html.it/articoli/recyclerview-dietro-le-quinte/

[73] Retrofit - https://square.github.io/retrofit/

[74] OkHttp - http://square.github.io/okhttp/

[75] gson di Google su GitHub - https://github.com/google/gson

[76] “Save key-value data” – Android Developers Guides - https://developer.android.com/training/data-storage/shared-preferences

[77] Fresco - https://frescolib.org/

[78] Configure build variants – Android Developers User guide - https://developer.android.com/studio/build/build-variants

[79] “Material Design Snackbar in Android – How and When to Use” – Articolo di Suleiman su Grafix Artist Blog – 3 Aprile 2017 - https://blog.iamsuleiman.com/material-design-snackbar/

[80] Translator Text Documentation - https://docs.microsoft.com/en-us/azure/cognitive-services/translator/

[81] ReadMoreTextView di Borja Bravo (bravoborja) su GitHub - https://github.com/bravoborja/ReadMoreTextView

[82] mstranslator di Kenan Shifflett (nanek) su GitHub - https://github.com/nanek/mstranslator

[83] android-fab di Mark Ormesher (markormesher) su GitHub - https://github.com/markormesher/android-fab

[84] flexbox-layout di Google su GitHub - https://github.com/google/flexbox-layout

[85] CSS Flexible Box Layout Module – W3C Candidate Recommendation, 19 Novembre 2018 - https://www.w3.org/TR/css-flexbox-1/

[86] “GDPR, tutto ciò che c'è da sapere per essere in regola” – Articolo di Alessandro Longo e Raffaella Natale su Agenda Digitale – 26 Maggio 2018 - https://www.agendadigitale.eu/cittadinanza-digitale/gdpr-tutto-cio-che-ce-da-sapere-per-essere-preparati/

[87] Firebase Crashlytics - https://firebase.google.com/docs/crashlytics

[88] Google Analytics - https://firebase.google.com/docs/analytics

[89] FirebaseAnalytics Reference - https://firebase.google.com/docs/reference/android/com/google/firebase/analytics/FirebaseAnalytics

[90] Android Jetpack – Android Developers - https://developer.android.com/jetpack

[91] ViewModel Overview – Android Developers Guides - https://developer.android.com/topic/libraries/architecture/viewmodel

[92] Room Persistence Library – Android Developers Guides - https://developer.android.com/topic/libraries/architecture/room

[93] Using Kotlin for Android Development - https://kotlinlang.org/docs/reference/android-overview.html

[94] Flutter Documentation - https://flutter.dev/docs

Ringraziamenti

Ringrazio tutte le persone che mi hanno supportato durante la carriera universitaria e la stesura di questa tesi.

In particolare ringrazio i miei genitori, zii e cugini che mi sono stati sempre vicini, il mio relatore e il team di Wher.

Ringrazio gli amici della Sicilia e quelli che hanno arricchito la mia vita qui a Torino.

Le persone che sono partite con me per intraprendere lo stesso percorso e quelle che ho conosciuto qui.

Ringrazio il teatro, il Lindy Hop e le mie psicologhe. Senza questi oggi la mia vita sarebbe più povera, io meno.

Ringrazio i miei coinquilini, attuali e precedenti, i vicini di casa, attuali e precedenti, i membri della Krew Kru e le nostre corse in bici.

Ringrazio piazza Santa Giulia e tutti gli altri fantastici posti di Torino per le iniziative che hanno ospitato e per le magnifiche serate trascorse da sobrio e da ciucco.

A tal proposito ringrazio in particolare il Circolo Tabacchi per tutto il whisky abbondante versato da Rocco e ringrazio Da Emilia per il bourbon, quello buono.

Ringrazio infine tutte le persone che ora sono in giro per il mondo ma che continuano a occupare un posto speciale nella mia rubrica telefonica.