Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

URBAN STORIES SHARING: LA APP DI RACCOLTA

Relatore: Prof.ssa Daniela Micucci

Correlatore: Dott. Davide Ginelli

Relazione della prova finale di:
Andrea Carubelli
Matricola 803192

Anno Accademico 2017-2018

Indice

Introduzione

In un mondo, sempre più urbanizzato, dove la percentuale di anziani cresce sempre di più, nasce il pensiero rispetto a quali debbano essere i canoni per una città del futuro. Da qui, nasce il concetto di ***walkability***. Generalmente, il termine walkability, indica il livello di comfort e sicurezza per i pedoni in una certa area urbana.
In concreto, una città ha un buon livello di walkability, se le strade e i marciapiedi sono in buono stato, se è presente una segnaletica stradale chiara e se sono presenti negozi, in una distanza facilmente percorribile a piedi.
Per questo, l'esigenza costante di individui che, vogliono rimanere sempre connessi al mondo che li circonda, condividendo sempre di più le proprie esperienze, ha portato alla creazione di un'applicazione mobile.
Il progetto Urban Stories Sharing, nasce da un'idea del dipartimento di Informatica, Sistemistica e Comunicazione dell'Università degli Studi di Milano-Bicocca, ed ha come obiettivo sia quello di migliorare la walkability di un'area urbana, ma soprattutto, quello di spingere le persone anziane ad essere ancora partecipi nella società, ad uscire nelle strade e quindi permettergli di raccogliere i dati relativi all'area urbana da loro frequentata, descrivendo storie attraverso: note scritte, foto, video e note vocali.
Per permettere tutto ciò, come enunciato qui sopra, le strade devono essere facilmente percorribili, in buono stato e senza pericoli.
Tutti questi dati geolocalizzati verrano salvati su un repository centralizzato, così che potranno essere condivise fra gli utenti.
La realizzazione del sistema è stata suddivisa in due parti: lo sviluppo del ***front-end*** e lo sviluppo del ***back-end***.
In questo rapporto di stage, si illustrerà la parte di sviluppo del ***front-end***.
Nel capitolo 1 verranno enunciate le specifiche dell'applicazione, nel capitolo 2 verranno ampiamente discusse le scelte implementative.
Successivamente nel capitolo 3 verrà proposto un manuale utente, in cui si spiegherà come utilizzare l'applicazione, non dal punto di vista tecnico, ma come poter fare una determinata azione, e, si terminerà, con il capitolo 4, in cui verranno presentate le conclusioni e i possibili sviluppi futuri.

Alla mia famiglia,
e a mio zio Thomas.

Capitolo 1

Specifiche dell'applicazione

In questo capitolo verrà analizzata la parte delle specifiche dell'applicazione, ovvero tutto ciò che è stato fatto prima dello sviluppo del progetto e dell'implementazione dell'app. Quindi, che cos'è la Walkability, il target di utenza a cui si è deciso rivolgersi, gli obiettivi principali dell'app ed il Material Design.

1.1 Walkability

Come già accennato all'interno dell'introduzione, il termine walkability, generalmente, indica il livello di comfort e sicurezza per i pedoni in una certa area urbana.
La walkability ha benefici per la salute, l'ambiente e l'economia.
I fattori che influenzano maggiormente una buona walkability sono: presenza o assenza di marciapiedi, la qualità di essi, condizioni del traffico e delle condizioni stradali, accessibilità e sicurezza degli edifici, sicurezza generale dell'area urbana interessata.
La walkability è un concetto importante nella progettazione urbana sostenibile [5].
Attualmente si parla molto di creare ambienti percorribili e di migliorare la walkabilty.
Tali strategie sono pensate per risolvere numerosi problemi derivanti dalla mancanza di vitalità del centro città come la congestione del traffico, l'ingiustizia ambientale e l'isolamento sociale [6].
Allo stato attuale, esistono due trend in continua crescita: uno è quello dell'invecchiamento della popolazione e l'altro è quello dell'urbanizzazione.
Proprio per questo si è cominciato a pensare come dovrebbe essere strutturata una città del futuro e si è così cominciato a parlare di walkability.
Si può osservare la figura 1.1, che raffigura l'invecchiamento delle persone negli ultimi, e futuri, 30 anni:

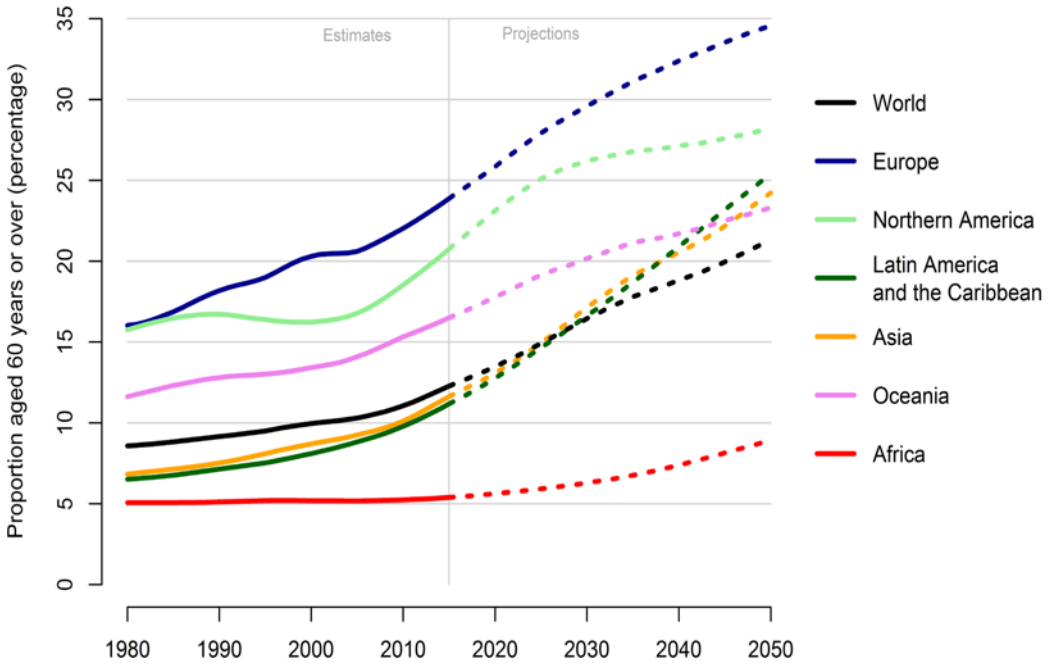

Figura 1.1: *Invecchiamento popolazione [8]*

Invece si può osservare, la figura 1.2, che raffigura l'andamento dell'urbanizzazione globale negli anni passati, ed una previsione nei prossimi:

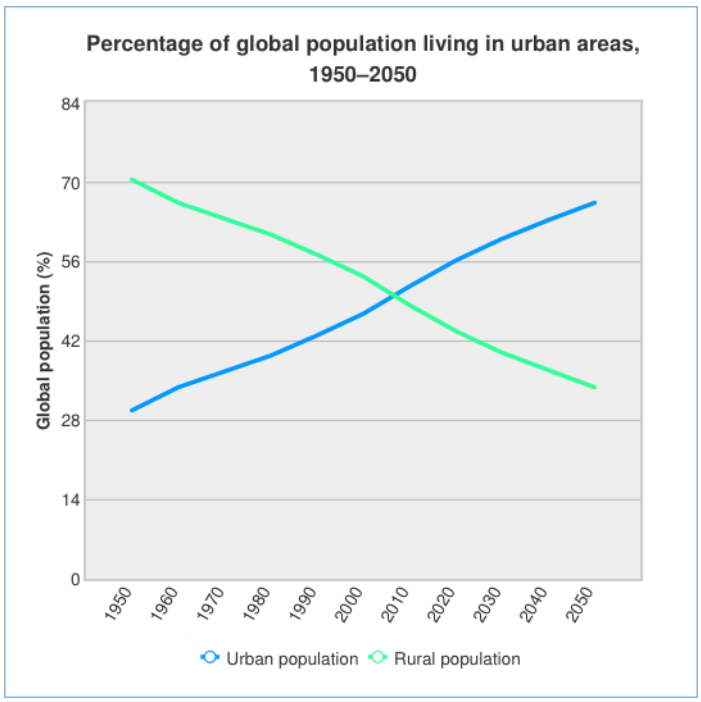

Figura 1.2: *Urbanizzazione [7]*

1.1.1 Punti chiave della Walkability

I punti principali, per cui una città abbia una buona walkability, sono i seguenti:

- **Ambienti attraversabili**: hanno le condizioni fisiche di base per consentire alle persone di spostarsi da un luogo all'altro senza grandi impedimenti, ad esempio percorsi relativamente regolari.
- **Luoghi compatti**: forniscono brevi distanze alle destinazioni per coloro che camminano per utilità.
- **Sicurezza**: è fondamentale in quanto, ad esempio, il crimine percepito e la sicurezza del traffico percepita riguardano un potenziale danno per la persona.
- **Ambienti fisicamente attraenti**: presenza di strutture pedonali come marciapiedi o percorsi, attraversamenti pedonali contrassegnati, illuminazione appropriata e arredo urbano, segnaletica utile e alberi da strada.
  Possono anche includere un'architettura interessante, viste piacevoli e servizi, che permettono alle persone di muoversi liberamente senza l'utilizzo delle automobili.

Di solito, un ***walkable environment***, ovvero un ambiente con una buona walkability, è definito tale, quando è vivace, socievole, piacevole, pulito e pieno di persone interessanti. [6].

1.2 Target utenti

Il target di utenza, a cui si è deciso rivolgersi, sono utenti anziani, o comunque tutti quegli utenti che si muovono in un'area urbana ristretta, vicina alla propria abitazione.
L'applicazione, in ogni caso, potrà essere utilizzata anche da qualsiasi altro utente che avrà piacere di condividere foto, video, note vocali o note scritte inerenti all'area urbana interessata. Sia per quanto riguarda esperienze positive, che negative.
Optare per questo tipo di utenza, ha fatto si, di adottare, alcune scelte implementative, rispetto ad altre, che verranno definite nel capitolo 2.
Inoltre, aver scelto questo tipo di utenza, ha fatto in modo di scegliere, a maggior ragione, il sistema operativo Android, in quanto più diffuso.

1.3 Obiettivi

Gli obiettivi principali del progetto (e quindi dell'applicazione) sono:

- Visualizzare la posizione corrente dell'utente all'interno di una mappa GPS servita da Google Maps
- Permettere all'utente di creare una nota caratterizzata da elementi multimediali

- Permettere all'utente di inserire nella nota: foto, video, note vocali, una nota scritta, nome del luogo ed indirizzo in cui si trova
- Visualizzare e permettere all'utente di scegliere se eliminare alcuni, o tutti, gli elementi acquisiti fino a quel momento
- Visualizzare il numero di elementi multimediali acquisiti fino a quel momento
- Inviare il contenuto della nota al repository centralizzato

1.4 Interfaccia utente

In questa sezione verrà presentato il Material Design.
Un requisito fondamentale dell'applicazione, è stato renderla il più intuitiva e semplice possibile da utilizzare, visto il target utenza scelto.
Dunque l'interfaccia è stata pensata ed ideata secondo le linee guida del Material Design, che verrà presentato ed introdotto nella sezione 1.4.1.
Successivamente nella sezione 2.5 verranno presentati gli elementi utilizzati di Material Design.

1.4.1 Material Design

Il Material Design è un design sviluppato da Google, annunciato il 25 giugno 2014 in occasione del Google I/O.
Il Material Design è supportato nativamente a partire da Android 5.0, ma può essere utilizzato nelle versioni precedenti attraverso la libreria v7 appcompat disponibile agli sviluppatori.
Per questa applicazione, si è fatto fede, ad alcune linee guida del Material Design.

Il Material Design è un linguaggio visuale che sintetizza i principi classici del buon design con l'innovazione della tecnologia e della scienza, ovvero, sviluppa un singolo sistema sottostante che unifica l'esperienza utente su piattaforme, dispositivi e metodi di input.
Sviluppa un singolo sistema sottostante che unifica l'esperienza utente su piattaforme, dispositivi e metodi di input.
Il Material Design è ispirato al mondo fisico, compreso il modo in cui riflettono la luce e si proiettano le ombre.
Il Material Design è guidato da metodi di progettazione della stampa (tipografia, griglie, spazio, scala, colore e immagini) per creare una gerarchia, significato e concentrazione per immergere gli spettatori nell'esperienza.
Il Material Design, focalizza l'attenzione e mantiene la continuità, attraverso sottili feedback e transizioni coerenti. Quando gli elementi appaiono sullo schermo, trasformano e riorganizzano l'ambiente, con interazioni che generano nuove trasformazioni.

Il Material Design mantiene la stessa interfaccia utente su tutte le piattaforme, utilizzando componenti condivisi su Android, iOS, Flutter e sul web [3].

Capitolo 2

Implementazione

Questo capitolo focalizzerà l'attenzione del lettore sull'implementazione ***front-end*** adottata.
Con il termine front-end si intende la parte di un sistema software che gestisce l'interazione con l'utente o con sistemi esterni che producono dati di ingresso. Concerne tutte le parti visibili all'utente, con cui può interagire e generare dati che verrano trasmessi al back- end, dove verrano elaborati.
Il capitolo concernerà il Sistema Operativo scelto, l'ambiente di sviluppo, le API utilizzate e relativo utilizzo, la struttura generale dell'applicazione.
Successivamente si entrerà nel dettaglio di ogni singolo componente di cui è composta l'applicazione, discutendone ampiamente, sia dal punto di vista tecnico, sia per le scelte implememententative adottate.

2.1 Sistema Operativo

Il sistema operativo, su cui eseguire l'applicazione mobile, che si è deciso di utilizzare per questo progetto, è stato **Android**.
Android è un sistema operativo per dispositivi mobili sviluppato da Google e basato sul kernel Linux.
Ad aprile 2017, Android, è il sistema operativo per dispositivi mobili più diffuso al mondo, con una fetta di mercato attestatasi a quota 62,94% sul totale, seguito da iOS con il 33,9% [1].
L'applicazione realizzata, è un'applicazione nativa Android, ovvero creata esclusivamente per questo sistema operativo. La scelta è ricaduta su Android, poichè copre la maggior parte di mercato rispetto ad iOS.
Inoltre, un'app nativa, ha alcuni vantaggi rispetto ad un'app ibrida (ovvero un'applicazione rivolta a diversi sistemi operativi), fra cui: maggiore velocità e accesso più facile a tutte le funzionalità del telefono.
I linguaggi utilizzati per la realizzazione dell'app sono stati due: **Java** e **XML**.

Si è scelto Java poichè è risultato il linguaggio più idoneo, consono e diffuso per lo sviluppo di app di questo tipo.

2.2 Ambiente di sviluppo

L'ambiente che si è deciso di utilizzare per lo sviluppo e l'implementazione dell'applicazione è stato **Android Studio**.
Android Studio è un ambiente di sviluppo integrato (IDE) per lo sviluppo per la piattaforma Android. Si è scelta questa piattaforma poichè risulta la più stabile e compatibile, praticamente, con tutti i dispositivi Android e permette anche di emulare, in tempo reale, un dispositivo virtuale per provare e testare la propria applicazione.
Inoltre, Android Studio è un software molto diffuso quindi è stato relativamente facile trovare la documentazione e i tutorial per imparare ad utilizzarlo. In più, tra i vantaggi ci sono: il poter firmare le applicazioni, uno strumento per il disegno della UI, la possibilità di fare test per le performance e il controllo di versioni; tutto questo senza dover installare nessun plug-in.
Per testare e provare l'applicazione, è stato utilizzato un dispositivo fisico, ovvero un Samsung Galaxy S7 con Android 8.0, poichè è risultato più semplice, in fase di implementazione, provare le varie funzionalità dell'applicazione.

2.3 API

Un elemento fondamentale delle applicazioni sono le API.
Con il termine application programming interface (API) si indica un insieme di procedure atte al completamento di un dato compito [2]. Spesso tale termine designa le librerie software di un linguaggio di programmazione. Una buona API fornisce una "scatola nera", cioè un livello di astrazione che evita al programmatore di sapere come funzionano le API ad un livello più basso. Questo permette di riprogettare o migliorare le funzioni all'interno dell'API senza cambiare il codice che si affida ad essa.
Le API permettono di espandere le funzionalità di un programma. Per uno sviluppatore mettere a disposizione un set di API di un suo software significa dare la possibilità ad altri di interagire con la sua piattaforma e, soprattutto, estendere le funzioni e le caratteristiche della struttura base della piattaforma. In altri termini, le API sono un ottimo strumento per promuovere un programma offrendo ad altri un modo per interagirci.
Sono quindi degli strumenti di programmazione che vengono messi a disposizione degli sviluppatori dalle maggiori software house e industrie del mondo informatico come Microsoft, Google e Facebook. Servono principalmente per facilitare la realizzazione e per espandere le funzionalita' di applicazioni di vario genere.
Le API possono assumere diverse "forme": possono essere delle librerie di funzioni che permettono al programmatore di interagire con un programma o una piattaforma software

o semplicemente una serie di "chiamate" a parti di un programma che uno sviluppatore può utilizzare per abbreviare il suo lavoro.

2.3.1 Utilizzo delle API

Utilizzando un'API, un programmatore può far interagire due programmi (o due piattaforme, o un programma e una piattaforma) altrimenti tra loro incompatibili. Si possono quindi estendere le funzionalita' di un programma ben oltre le reali intenzioni dello sviluppatore o della software house che l'ha realizzato. Un esempio pratico sono le API di Google Maps che sono a disposizione di tutti gli sviluppatori che le volessero utilizzare per un loro programma o piattaforma web.
Sfruttando le API, ad esempio, è possibile utilizzare il servizio di cartografia digitale di Google per realizzare delle mappe personalizzate; oppure integrarle in siti web per servizi di ricerca georeferenziati; o ancora utilizzarle all'interno di applicazioni per smartphone e tablet.

2.4 Struttura generale

In questa sezione verrà presentata, la struttura generale dell'applicazione. La struttura dell'app è stata, di volta in volta, aggiornata e modificata durante l'implementazione del progetto, fino a presentarsi in questo modo.
L'applicazione si suddivide principalmente in 3 macro-aree:

- Activity e classi *.java*
- Risorse (Layout e values)
- AndroidManifest.xml

Le Activity e i rispettivi layout sono definiti nel seguente modo:

ACTIVITY	LAYOUT
MapsActivity.java	activity_maps.xml
MicrophoneActivity.java	activity_microphone.xml
ModifyPhotoActivity.java	activity_modify_photo.xml
ModifyVideoActivity.java	activity_modify_video.xml
ModifyVoiceNoteActivity.java	activity_modify_voice_note.xml
PostActivity.java	activity_post.xml
TextNoteActivity.java	activity_text_note.xml

Tabella 2.1: *Activity e rispettivi layout*

Inoltre, è stata utilizzata un'interfaccia ***FileInformation.java*** e altre tre classi di appoggio ***SavingOfFile.java***, ***MySingleton.java*** e ***GenericFileProvider.java***.

Nella sezione 2.4.1 verrà ampiamente discusso lo scopo di ogni Activity e classe Java implementata.
Nella sezione 2.4.2 verranno presentati i layout ma la loro presentazione verrà rimandata nel capitolo 3.
Nella sezione 2.4.3 verrà lasciato uno spazio per presentare le risorse utilizzate durante l'implementazione.
Nella sezione 2.4.4 verrà presentato l'Android Manifest, la sua utilità e come è stato implementato.

2.4.1 Activity e Classi Java

L'Activity è l'elemento principale di un'applicazione Android ed è essenzialmente una finestra che contiene l'interfaccia utente e può essere quindi vista come una schermata.
Il suo scopo è quello di consentire un'iterazione con l'utente. Un'applicazione può avere una o più Activity, ma solo una schermata alla volta può essere in primo piano. Vi è quindi un'Activity stack dove sono registrate tutte le Activity e si può passare da un'Activity all'altra grazie all'Activity Manager.
L'applicazione presenta **7 Activity** ognuna delle quali verrà ripresa poco più avanti dettagliatamente, spiegandone l'utilità.
Le Activity implementate sono le seguenti:

- MapsActivity.java
- MicrophoneActivity.java
- ModifyPhotoActivity.java
- ModifyVideoActivity.java
- ModifyVoiceNoteActivity.java
- PostActivity.java
- TextNoteActivity.java

MapsActivity.java

Questa è l'Activity che si incontra appena si apre l'applicazione, ovvero quella che rappresenta la mappa di Google Maps.
Si è deciso di iniziare da questa Activity, in questo modo, poichè si è pensato che era un ottimo modo di contestualizzare l'obiettivo dell'applicazione dando all'utente un feedback su dove si potesse trovare in un determinato ambiente ed orientarsi al suo interno.
La posizione del dispositivo viene continuamente aggiornata ogni qualvolta l'utente si sposti.

PostActivity.java

Questa è l'Activity principale dell'applicazione, ed è stata implementata inizialmente ispirandosi al *"Aggiungi luogo"* di Google Maps, per poi riprogettarla unendo una vista basata su CardView e su due TextInputLayout poichè si è rivelata la miglior soluzione per quanto riguarda lo sviluppo di questo tipo di applicazione, più semplice ed intuitiva. All'interno delle CardView sono stati implementati diverse componenti, come: le rispettive icone dei file multimediali, il numero dei file aggiunti fino a quel momento ed infine un bottone per la modifica dei file salvati in locale fino a quel punto. Foto e video verranno creati direttamente all'interno di questa Activity quando si apriranno rispettivamente fotocamera e videocamera.
L'invio dei file al database è stato eseguito sfruttando la libreria Volley, che si riprenderà in seguito nella sezione 2.6, convertendo tutti i file acquisiti fino a quel momento in *Base64*, in quanto si è visto che è il formato più diffuso per l'invio di file multimediali attraverso richieste *POST* di tipo HTTP.
Inoltre è stata implementata la creazione di un file di tipo *CSV* che includesse le informazioni recuperate all'interno di questa Activity, come: latitudine, longitudine, nome del luogo ed indirizzo; così che si potesse, tramite espressioni regolari, recuperare un certo tipo di informazioni.
Al momento della stesura di questa relazione, non è stato ancora implementato l'invio al database di questo file, poichè, lato back-end, non è stato predisposta la ricezione di questo file.
E' stato implementato anche un Alert Dialog all'interno di un ***AsyncTask*** durante la richiesta HTTP, in modo tale che l'utente avesse un feedback sul processo di caricamento dei file sul repository centralizzato.
Nel capitolo 3 verrà mostrato questo comportamento.

MicrophoneActivity.java

Questa Activity concerne l'implementazione della creazione dei file audio in formato *.3gp*. Si è scelto questo formato data la sua semplicità di utilizzo.
Il modo in cui avviene il loro salvataggio verrà rimandato nella sottosezione 2.4.1, nella spiegazione della classe *SavingOfFile.java*, poichè sarà la classe Java principale adibita al salvataggio dei file in locale, una delle classi principali per quanto riguarda l'applicazione.
Sostanzialmente, in questa Activity, sono stati creati due bottoni che hanno rispettivamente il compito di avviare e terminare una registrazione vocale, interpellando, ovviamente, l'utilizzo del microfono. Si è scelto questo tipo di approccio, di creare due bottoni grandi ed evidenti, in vista del target di utenza.
Verrano allocate le risorse necessarie per far si che si registri effettivamente la nota vocale per poi poter essere salvate in locale.
Sono anche state implementate delle Snackbar e un Alert Dialog per dare dei feedback

all'utente su ciò che sta accadendo.
Nel capitolo 3, verranno mostrati questi componenti.

TextNoteActivity.java

Questa Activity concerne la creazione della nota testuale. Si è semplicemente pensato come una riga in cui l'utente potesse scriverci, per poi salvarla in locale.
Anche in questo caso, si è pensato di implementare diverse Snackbar e Alert Dialog che verranno ripresi nel capitolo 3.
Si è deciso di implementare anche la possibilità per l'utente di modificare la nota scritta, facendo in modo che il campo di testo si riempisse con il testo scritto in precedenza una volta che si entra nuovamente nell'Activity.

ModifyPhotoActivity.java

Questa Activity concerne la modifica delle foto, ovvero la loro cancellazione.
Si è pensato ad adottare una soluzione a griglia, stile galleria, creata in maniera dinamica durante l'esecuzione, in base al numero dei file presenti, dove vengono presentate le anteprime delle immagini, con sottostante un bottone che permettesse la cancellazione della foto.
E' stato anche implementato un Alert Dialog per chiedere all'utente se è effettivamente sicuro di cancellare una determinata foto, in quanto c'è la possibilità di ripensarci oppure che si sia schiacciato il bottone per sbaglio.
Nel capitolo 3, verrà mostrato questo componente.

ModifyVideoActivity.java

Si rimanda il lettore alla sottosezione soprastante, inerenti alla classe *ModifyPhotoActivity.java* poichè il funzionamento e le scelte implementative sono identiche.

ModifyVoiceNoteActivity.java

Si rimanda il lettore alla sottosezione soprastante, inerenti alla classe *ModifyPhotoActivity.java* poichè il funzionamento e le scelte implementative sono identiche.

FileInformation.java

Si tratta di un'interfaccia che contiene una serie costanti che sono state utilizzate all'interno dell'applicazione per tenere in maniera più organizzata ed ordinata la struttura delle variabili.

SavingOfFile.java

Questa classe ha il compito di salvare in locale i vari file multimediali e si è implementata in questo modo: creare una cartella principale, in cui all'interno ci fossero una serie di cartelle, che come nome avessero un *timestamp*, creato quando venisse iniziata una nuova nota, in modo tale che si potessero distinguere univocamente una nota rispetto ad un'altra. Successivamente, all'interno di questa cartella creare le varie subdirectory che contenessero i vari tipi di file multimediale.

MySingleton.java

Questa classe è stata creata per dare un appoggio per poter utilizzare la libreria Volley per permettere il caricamento sul repository centralizzato.

GenericFileProvider.java

Questa classe è stata creata estenendo la classe Provider, già implementata in Android, per poter salvare i file in locale.

2.4.2 Layout

In informatica **XML** (sigla di eXtensible Markup Language) è un metalinguaggio per la definizione di linguaggi di markup, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.
I layout dell'applicazione e le risorse utilizzate sono state implementate secondo questo metalinguaggio.
L'applicazione presenta **7 layout**:

- activity_maps.xml
- activity_microphone.xml
- activity_modify_photo.xml
- activity_modify_video.xml
- activity_modify_voice_note.xml
- activity_post.xml
- activity_text_note.xml

Nel capitolo 3 verranno mostrati nel dettaglio gli aspetti di questi layout.
In linea generale sono stati usati componenti di grandi dimensioni, molto evidenti ed i layout sono stati pensati per renderli più intuitivi e facili possibili, in vista del target di utenza scelto.

2.4.3 Values

Nella cartella values sono presenti le "risorse" a cui si affideranno i vari metodi e classi durante l'implementazione del progetto. All'interno sono presenti:

- **colors.xml**
  In questo file possono essere definiti i vari colori che andranno poi ad essere utilizzati durante l'implementazione

- **dimens.xml**
  All'interno possono essere definite alcune dimensioni che si andranno ad utilizzare durante l'implementazione

- **strings.xml**
  All'interno saranno definite tutte le stringhe che verrano utilizzate durante l'implementazione dell'applicazione.

- **styles.xml**
  In questo file, saranno definiti tutti gli stili che verranno utilizzati durante l'implementazione.

I values, sono molto utili, in quanto evitano l'hardcoding (soprattutto per quanto riguarda le stringhe).

2.4.4 AndroidManifest.xml

Ogni applicazione deve avere il file ***AndroidManifest.xml*** (con questo preciso nome) alla radice del proprio progetto.
Il manifest, descrive le informazioni essenziali di ogni applicazione come: strumenti di Android, sistema operativo Android e Google Play.
Tra le altre cose, è richiesto il file manifest per dichiarare quanto segue:

- **Nome del pacchetto dell'app**
  Quando si compila il codice, gli strumenti di compilazione sostituiscono questo valore con l'ID dell'applicazione dai file di build di Gradle, che viene utilizzato come identificatore di app univoco sul sistema e su Google Play.

- **I componenti dell'app**
  Comprendono tutte le Activity, i servizi, i ricevitori di trasmissione e i fornitori di contenuti.
  Ogni componente può anche dichiarare funzionalità quali le configurazioni dei dispositivi che può gestire e filtri di intent che descrivono come il componente può essere avviato.

- **Autorizzazioni**
  Le autorizzazioni di cui l'app ha bisogno per accedere a parti protette del sistema o altre app.

Per poter utilizzare l'API di Google Maps bisogna però registrarsi sul sito web specifico ed ottenere una chiave univoca, senza la quale non sarebbe possibile visualizzare neanche la mappa.
Si può osservare l'implementazione all'interno del file **AndroidManifest.xml**:

```
<meta-data
           android:name="com.google.android.geo.API_KEY"
           android:value="@string/google_maps_key" />
```

Dove ***"@string/google_maps_key"*** è una stringa costante definita nella cartella ***Values/strings.xml*** sopracitata e permetterà di accedere ai servizi delle mappe di Google.

Per quanto riguarda l'implementazione per il salvataggio dei file, attraverso il *FileProvider.java* descritto nella sottosezione 2.4.1, all'interno dell'Android Manifest, è stato così implementato:

```
<provider
           android:name=".GenericFileProvider"
           android:authorities="com.example.android.fileprovider"
           android:exported="false"
           android:grantUriPermissions="true">
           <meta-data
               android:name="android.support.FILE_PROVIDER_PATHS"
               android:resource="@xml/provider_paths" />
       </provider>
```

2.5 Design

In questa sezione verranno presentati gli elementi principali di design di cui è composta l'applicazione.

2.5.1 Material Button

E' stata utilizzata una libreria che implementasse i Material Buttons, ovvero bottoni con cui l'utente potesse interagire, che seguissero le linee guida del Material Design.
La libreria in questione, è la seguente: ***android.support.design.button.Material But-***

ton

In Figura 2.1 si può osservare un esempio di Material Button *"Crea Nota"*:

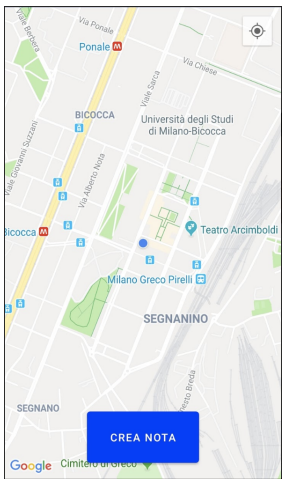

Figura 2.1: *Esempio di MaterialButton "Crea nota"*

2.5.2 Support Library Packages

Una libreria, è un insieme di funzioni o strutture dati predefinite e predisposte per essere collegate ad un programma software attraverso un opportuno collegamento.
Lo scopo delle librerie software è fornire una collezione di entità di base pronte per l'uso, ovvero, riuso di codice, evitando al programmatore di dover riscrivere ogni volta le stesse funzioni o strutture dati e facilitando così le operazioni di sviluppo e manutenzione. Questa caratteristica si inserisce quindi nel più vasto contesto del "richiamo di codice" all'interno di programmi e applicazioni ed è presente in quasi tutti i linguaggi.
I vantaggi principali derivanti dall'uso di un simile approccio sono i seguenti:

- Si può separare la logica di programmazione di una certa applicazione da quella necessaria per la risoluzione di problemi specifici, quali il calcolo di funzioni matematiche o la gestione di collezioni
- Le entità definite in una certa libreria possono essere riutilizzate da più applicazioni
- Si può modificare la libreria separatamente dal programma, senza limiti alla potenziale vastità di funzioni e strutture dati man mano disponibili nel tempo

La libreria di supporto Android contiene diversi pacchetti di librerie che possono essere inclusi nell'applicazione. Ciascuna di queste librerie supporta una gamma specifica di versioni della piattaforma Android e un insieme di funzionalità.

TextInput Layout

Un'altra libreria utilizzata è stata la seguente: ***android.support.design.widget.TextInputLayout*** Ovvero è una libreria che potesse permettere l'utilizzo di Text Input, ovvero campi compilabili dall'utente, che permettessero di seguire i canoni estetici e funzionali del Material Design.
In figura 2.2 se ne può osservare un esempio:

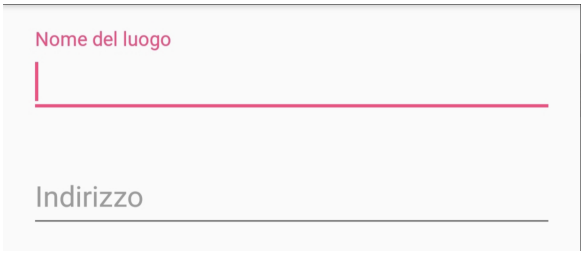

Figura 2.2: *Esempio di TextInput*

Toolbar

Ulteriore libreria implementata, necessaria al fine dello sviluppo, è stata la seguente: ***android.support.v7.widget.Toolbar*** Questa libreria ha fatto si che si potesse implementare una toolbar che permettesse all'utente di eseguire delle operazioni. In figura 2.3 si può osservare un esempio di toolbar:

Figura 2.3: *Esempio di Toolbar*

Card View

Questa libreria implementata: ***android.support.v7.widget.CardView*** ha permesso di aggiungere delle Card View, ovvero degli elementi che consentono di mostrare informazioni all'interno di schede che hanno un aspetto coerente su qualsiasi app.
Queste schede sono utili per le implementazioni di Material Design e sono ampiamente utilizzate nei layout per le app TV. In figura 2.4 si può nortare un esempio:

Figura 2.4: *Esempio di CardView*

Icone

Tutte le icone utilizzate seguono le linee guida del Material Design e sono state implementate tramite Android Studio attraverso il Vector Asset, ovvero immagini vettoriali modificabili in fase implementativa.
Si può vedere un esempio di icone di Material Design in figura 2.5:

Figura 2.5: *Esempio icone Material Design*

Tutti questi aspetti design verrano ripresi nel capitolo 2, più precisamente nella sezione 2.4.2, per poter descrivere le ragioni per cui sono state adottate alcune scelte rispetto ad altre.

2.6 Volley Library

Volley è una libreria HTTP che semplifica la connessione in rete per le app Android e, cosa più importante, più veloce. Volley è disponibile su GitHub.
Volley offre i seguenti vantaggi:

- Pianificazione automatica delle richieste di rete.
- Più connessioni di rete simultanee.
- Supporto per la prioritizzazione delle richieste.
- API di richiesta di cancellazione. È possibile annullare una singola richiesta oppure è possibile impostare blocchi di richieste da annullare.
- Facilità di personalizzazione
- Strumenti di debug e tracciamento.

All'interno di questo progetto di stage, è stata utilizzata questa libreria per eseguire la richiesta **POST**, ovvero, la richiesta HTTP per inviare i file multimediali al repository centralizzato [4].

Capitolo 3

Manuale Utente

In questo capitolo verranno presentate tutte le schermate che concernono l'applicazione e le varie funzioni di ognuna.
Più precisamente, nella sezione 3.1 verrà presentata la prima schermata che l'utente incontra quando apre l'applicazione per la prima volta, nella sezione 3.2 verrà presentata la schermata principale dell'applicazione, nella sezione 3.3 verrà presentata la schermata che riguarda la registrazione della nota vocale, nella sezione 3.4 verrà presentata la schermata che riguarda l'inserimento della nota scritta.
Successivamente, nelle sezioni 3.5, 3.6, 3.7, verranno presentate le schermate addette al compito di modificare i file multimediali registrati fino a quel momento.

3.1 Mappa

Quando l'utente apre l'applicazione per la prima volta, gli verrà richiesto di accettare i permessi, come si può vedere in figura 3.1:

Figura 3.1: *Permessi posizione*

Accettando i permessi, sarà possibile caricare e visualizzare la mappa e geolocalizzare il dispositivo, come si può vedere in figura 3.2, che si riferisce al layout *activity_ maps.xml*:

Figura 3.2: *Mappa*

All'interno di questa schermata, l'utente, potrà muoversi all'interno della mappa, schiacciare sul bottone "Crea Nota" oppure geolocalizzarsi sulla posizione corrente, cliccando il bottone in alto a destra.
Se, nel caso il dispositivo non riesca a recuperare la posizione, comparirà un messaggio che avviserà l'utente che si sta provando a caricare la posizione. Questo comportamento si può notare nella figura 3.3.

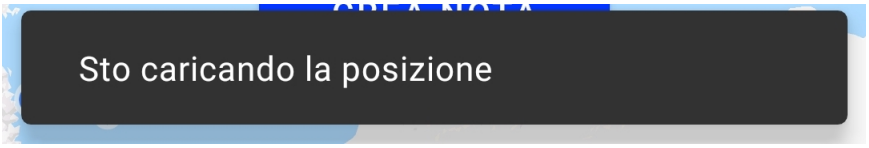

Figura 3.3: *Snackbar posizione non recuperata*

Se per caso l'utente avesse già aperto l'applicazione in precedenza, e avesse già accettato i permessi, la prima schermata dell'applicazione si presenterebbe direttamente con la mappa come in figura 3.2.

3.2 Homepage

La schermata principale dell'applicazione è questa che si può vedere in figura 3.4, che si riferisce al layout *activity_post.xml*:

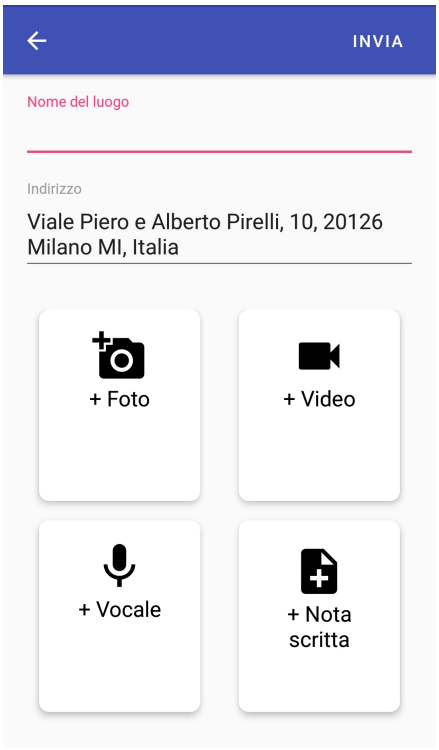

Figura 3.4: *Homepage*

Se è il primo avvio dell'applicazione, verrà chiesto all'utente di accettare i permessi relativi alla scrittura e alla lettura in memoria del dispositivo, come si può vedere in figura 3.5:

Figura 3.5: *Permessi memoria*

Come si può osservare dalla figura 3.4, l'indirizzo verrà popolato in maniera automatica se il dispositivo riuscirà a recuperare la posizione corrente del dispositivo.
In questa schermata l'utente avrà la libertà di scattare foto, registrare video, registrare note audio e scrivere una nota scritta. Inoltre potrà inserire il nome del luogo in cui si trova, che sceglierà a sua discrezione.
Come si può vedere dalla figura 3.6, una volta che è stato inserito almeno un elemento, appartenente ad una determinata categoria, verrà anche visualizzato un bottone "Modifica" ed un numero che corrisponde al numero di elementi presenti fino a quel momento:

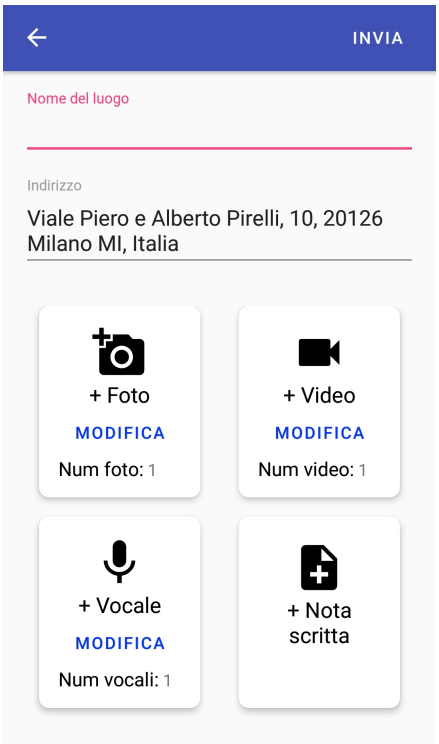

Figura 3.6: *Homepage con tasto modifica*

E' presente una toolbar in alto alla schermata, dove a sinistra l'utente potrà tornare indietro, nella schermata della mappa, schiacciando sulla freccia direzionata verso sinistra, ed a destra, è presente il tasto "invia", che permetterà all'utente di inviare ciò che ha scattato/registrato, al repository centralizzato.
Inoltre, se sono presenti degli elementi all'interno della nota, e l'utente decidesse di tornare indietro, senza aver inviato i dati, verrà avvisato tramite un messaggio che gli chiederà se è sicuro di tornare indietro, poichè, in caso positivo, perderebbe i dati acquisiti fino a quel momento.
Questo comportamento si può osservare in figura 3.7:

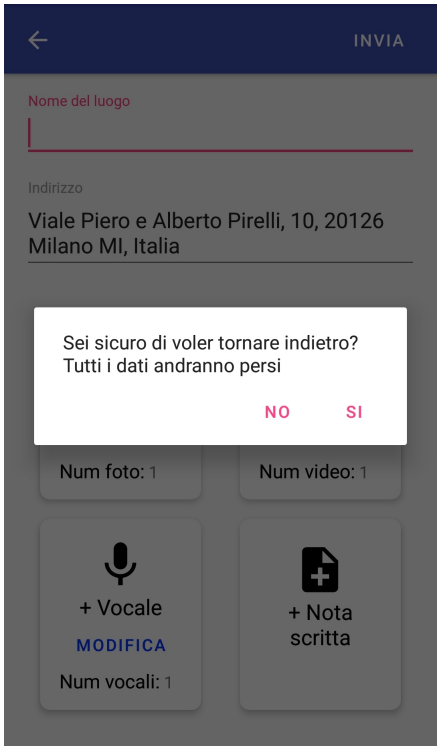

Figura 3.7: *Alert Homepage*

Dopodichè, quando l'utente schiaccerà sul tasto "Invia" verrà mostrato un messaggio che darà all'utente un feedback sul progresso del caricamento dei dati sul repository centralizzato.
Questo comportamento si può vedere in figura 3.8:

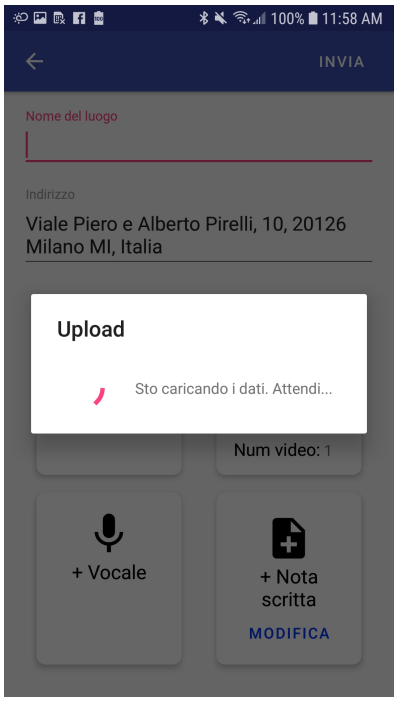

Figura 3.8: *Progresso durante il caricamento*

Una volta che sarà finito il caricamento, l'applicazione tornerà alla schermata principale, ovvero quella della mappa in figura 3.2.
Una volta che l'applicazione sarà tornata nella schermata della mappa, verrà visualizzato un messaggio che darà un feedback all'utente che il caricamento sarà avvenuto con successo.
Questo comportamento si può osservare nella figura 3.9:

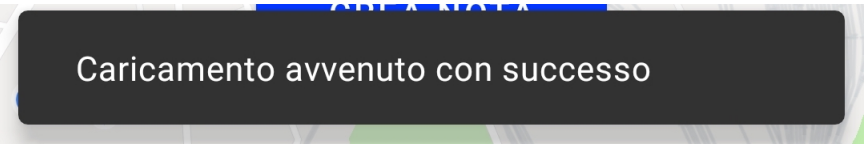

Figura 3.9: *Snackbar caricamento*

3.3 Nota vocale

All'interno di questa schermata, si possono osservare due bottoni che permettono di avviare e terminare una registrazione vocale.
Si può osservare l'aspetto di questa schermata in figura 3.10, che si riferisce al layout *activity_microphone.xml*:

Figura 3.10: *Schermata nota vocale*

Con il tasto a sinistra si avvia la registrazione, e verrà mostrata una barra di progressione sopra questi due bottoni, che darà un feedback all'utente che la registrazione sarà in corso. Con il tasto di destra si interromperà la registrazione e l'applicazione tornerà

all'Activity precedente, ovvero quella che si può notare in figura 3.4.
Anche per questa schermata, sono stati implementati dei controlli, ad esempio, se l'utente decidesse di tornare indietro durante la registrazione, viene mostrato un messaggio.
Questo comportamento si può notare in figura 3.11:

Figura 3.11: *Alert nota vocale e barra di progressione*

Sono presenti anche dei messaggi che avvisano l'utente nel caso in cui schiacci erroneamente sul bottone "Avvia" dopo che è già stata iniziata una registrazione, oppure sul bottone "Ferma" se non è stata ancora iniziata una registrazione.
Queste snackbar si possono osservare rispettivamente nelle figure 3.12 e 3.13:

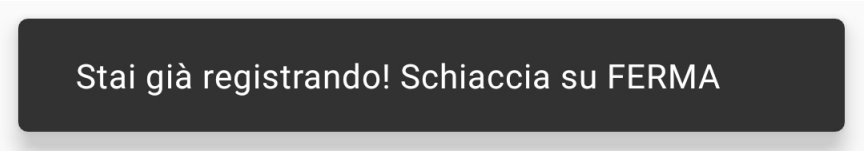

Figura 3.12: *Snackbar tasto avvia*

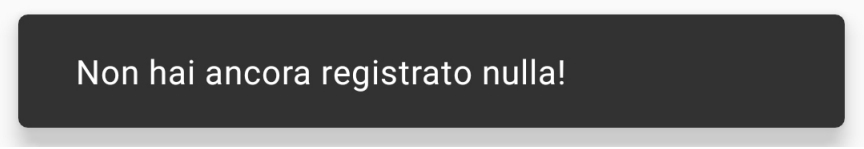

Figura 3.13: *Snackbar tasto ferma*

3.4 Nota Scritta

La schermata per inserire la nota scritta si presenta in questo modo, visibile in figura 3.14, che si riferisce al layout *activity_text_note.xml*:

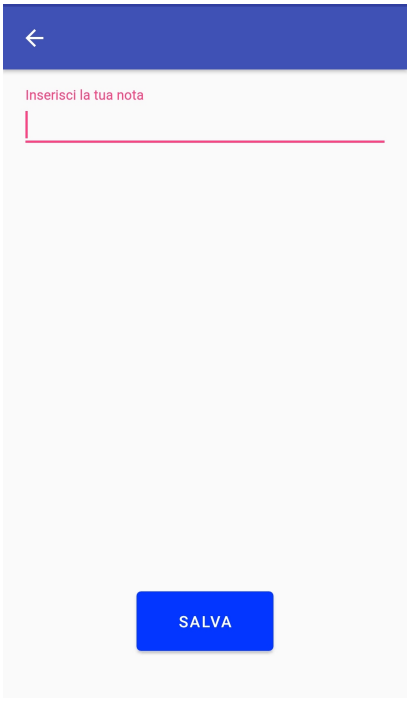

Figura 3.14: *Schermata per inserimento nota scritta*

L'utente potrà inserire un testo, nel campo apposito, per poi salvare la nota, schiacciando sul bottone "Salva".
Una volta salvata la nota scritta, l'applicazione tornerà alla schermata precedente, visibile in figura 3.4.
Se l'utente desiderasse modificare la nota, basterà schiacciare nuovamente sul bottone, nella schermata principale 3.4, dedicato alla nota scritta, e troverà all'interno del campo di testo, ciò che aveva scritto in precedenza.
Se l'utente, invece, decidesse di tornare indietro senza salvare la nota scritta, verrà avvisato tramite un messaggio, che si può osservare in figura 3.15:

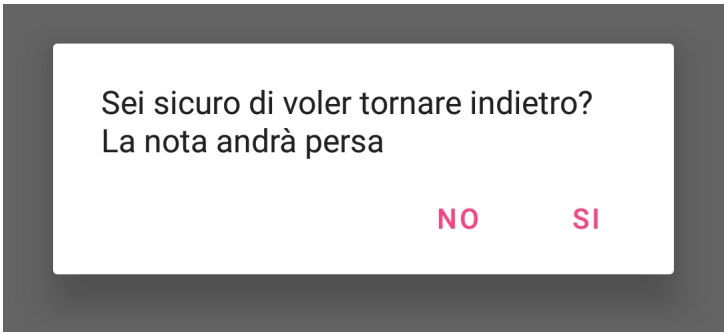

Figura 3.15: *Alert mancato salvataggio nota scritta*

3.5 Modifica foto

In questa schermata, accessibile schiacciando il bottone modifica visibile in figura 3.6 sotto l'icona delle foto, permette all'utente di cancellare una foto alla volta.
La schermata si presenta in questo modo in figura 3.16 e si riferisce al layout *activity_modify_photo.xml*:

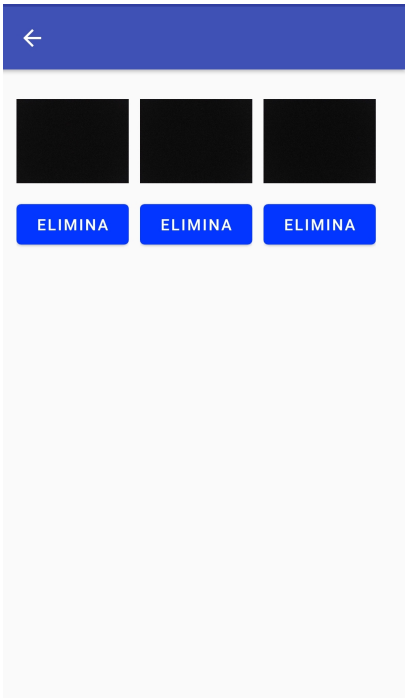

Figura 3.16: *Schermata modifica foto*

Schiacciando sul bottone elimina, comparirà un messaggio, che avviserà l'utente se è effettivamente convinto di cancellare quella foto. Si presenta come in figura 3.21:

Figura 3.17: *Alert eliminazione foto*

A questo punto l'utente potrà decidere se mantenerla oppure cancellarla effettivamente. Se l'utente premerà il bottone "Mantieni" l'applicazione rimarrà nella schermata corrente. Se schiaccerà sul bottone "Cancella", verrà cancellata la foto ed a questo punto l'applicazione tornerà alla schermata precedente.

3.6 Modifica video

In questa schermata, accessibile schiacciando il bottone modifica visibile in figura 3.6 sotto l'icona dei video, permette all'utente di cancellare un video alla volta. Verrà illustrata l'anteprima del rispettivo video.
La schermata si presenta in questo modo in figura 3.18 e si riferisce al layout *activity_modify_video.xml*:

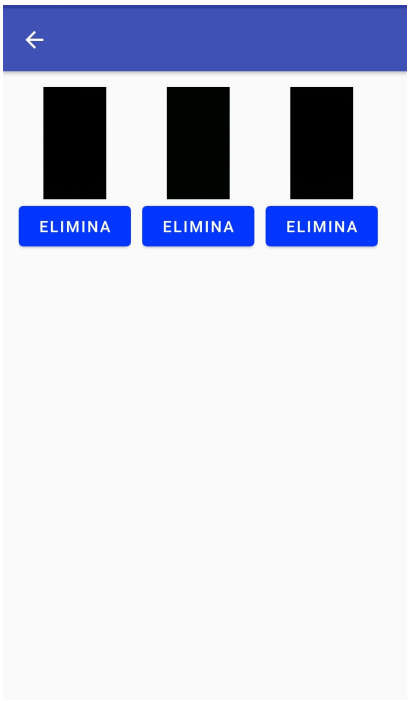

Figura 3.18: *Schermata modifica video*

Schiacciando sul bottone "Elimina", comparirà un messaggio, che avviserà l'utente se è effettivamente convinto di cancellare quel video. Si presenta come in figura 3.21:

Figura 3.19: *Alert eliminazione video*

A questo punto l'utente potrà decidere se mantenerlo oppure cancellarlo effettivamente. Se l'utente premerà il bottone "Mantieni" l'applicazione rimarrà nella schermata corrente, se schiaccerà sul bottone "Cancella", verrà cancellato il video ed a questo punto l'applicazione tornerà alla schermata precedente.

3.7 Modifica nota vocale

In questa schermata, accessibile schiacciando il bottone modifica visibile in figura 3.6 sotto l'icona della nota vocale, permette all'utente di cancellare una nota vocale alla volta. La schermata si presenta in questo modo in figura 3.20 e si riferisce al layout *activity_modify_voice_note.xml*:

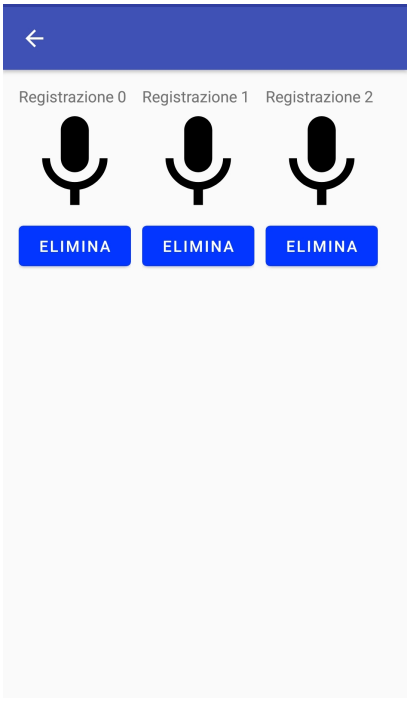

Figura 3.20: *Schermata modifica nota vocale*

Schiacciando sul bottone "Elimina", comparirà un messaggio, che avviserà l'utente se è effettivamente convinto di cancellare quella nota vocale. Si presenta come in figura 3.21:

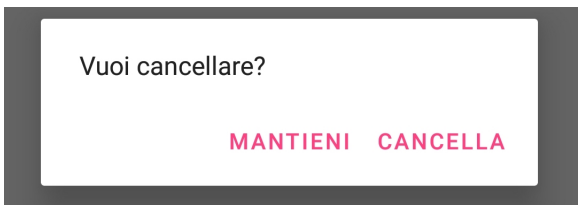

Figura 3.21: *Alert eliminazione video*

A questo punto l'utente potrà decidere se mantenerla oppure cancellarla effettivamente. Se l'utente premerà il bottone "Mantieni" l'applicazione rimarrà nella schermata corrente, se schiaccerà sul bottone "Cancella", verrà cancellata la nota vocale ed a questo punto l'applicazione tornerà alla schermata precedente.

Capitolo 4

Conclusioni e sviluppi futuri

In questo capitolo verrano esposti i possibili sviluppi futuri del progetto e le conclusioni di questo rapporto di stage.

4.1 Il futuro

L'applicazione Urban Stories Sharing, attualmente, soddisfa tutti i requisiti di sistema prefissati all'inizio del percorso. Tuttavia ci sono alcuni aspetti che potrebbero essere migliorati ed implementati in una versione futura.
Principalemente sono:

- Miglioramento del design e User Experience
- Implementare cancellazione multipla degli elementi multimediali
- Miglioramento delle performance
- Splash Screen e logo app
- Tutorial iniziale
- Implementare sistema di autenticazione

Per il primo punto potrebbe essere necessario contattare e prendere in considerazione un esperto di usabilità che permetta di implementare un'interfaccia mirata e ridisegnata, migliorarla in base al target scelto.

Per il secondo punto si potrebbe pensare ad un'interfaccia stile "galleria" in cui l'utente possa selezionare simultaneamente diversi file multimediali per poterli cancellarli contemporaneamente.

Per il terzo punto si potrebbero migliorare le prestazioni intervenendo sulla qualità del codice e suddividere in più classi i vari compiti definiti all'interno dell'applicazione.

Per il quarto punto si potrebbe contattare e prendere in cosiderazione un grafico o designer che sia disposto a disegnare un logo per l'applicazione per poi creare uno Splash Screen, ovvero una schermata iniziale di "caricamento", che dia un feedback di attesa piacevole all'utente intanto che l'applicazione carichi tutte le sue funzionalità, per non lasciarlo in "attesa" inutilmente.

Per il quinto punto si potrebbe implementare un tutorial iniziale, al primo avvio dell'applicazione, che permetta all'utente di capire che cosa può fare e come farlo.

Per l'ultimo punto si potrebbe implementare un sistema di autenticazione in modo tale che ogni utente possa accedere al sistema, vedere uno storico delle note inserite, per poi visionarle all'interno della mappa, così che gli utenti possano interagire fra di loro. In questo caso ovviamente ci sarebbero da fare delle modifiche al ***back-end***.

4.2 Conclusioni

Il lavoro svolto durante questo progetto di stage ha permesso di realizzare un'applicazione Android funzionante, in tutti i suoi aspetti, ovvero la visualizzazione della propria posizione geolocalizzata all'interno di una mappa fornita da Google Maps, l'utilizzo delle risorse hardware del dispositivo, come fotocamera, videocamera e microfono, la richiesta dei vari permessi per utilizzare le varie funzionalità, il salvataggio in locale dei vari file multimediali, l'invio di quest'ultimi (foto, video, note vocali e scritte) convertiti in ***Base64*** ad un repository centralizzato attraverso una chiamata ***HTTP*** di tipo ***POST***, senza tralasciare la possibilità di cancellarli dalla memoria fisica del dispositivo, ed infine, gestire i vari messaggi di avviso per l'utente come Snackbar, Alert Dialog e un ***AsyncTask*** durante la chiamata POST.

Questo percorso mi ha permesso di ampliare e migliorare le mie conoscenze in merito ai linguaggi di programmazione come Java ed XML, conoscere il sistema IDE Android Studio, che mi era sconosciuto all'inizio, imparare a gestire un progetto di questa portata in autonomia, organizzando i vari step decisionali e di implementazione, risolvendo problemi ricorrenti trovando soluzioni flessibili e coerenti con quello che era il progetto.
Tutto questo sotto la supervisione del mio tutor Davide Ginelli.
Approfittandone, colgo l'occasione per ringraziarlo per l'aiuto ed il supporto puntuale e professionale ricevuto durante tutto il percorso.

Bibliografia

[1] Android. https://it.wikipedia.org/wiki/Android.

[2] Definizione api. https://simonecarletti.it/blog/2006/09/che-cosa-sono-le-api-application-programming-interface/.

[3] Material design. https://material.io/design/introduction/.

[4] Volley library. https://developer.android.com/training/volley/.

[5] Walkability. https://en.wikipedia.org/wiki/Walkability.

[6] Ann Forsyth. What is a walkable place? the walkability debate in urban design. *Urban design international*, 20(4):274–292, 2015.

[7] United Nations. World urbanization prospects: the 2014 revision. 2014.

[8] United Nations. World population prospects: the 2017 revision. 2017.