



UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA  
Dipartimento di Informatica, Sistemistica e  
Comunicazione  
Corso di Laurea in Informatica

## Sviluppo di applicazioni mobile con supporto di ChatGPT: GreenWay per la sensibilizzazione sulla Co2

**Relatore:** Prof. Daniela Micucci

**Correlatore:** Dott. Maria Teresa Rossi

**Tesi di Laurea di:**  
Davide Domenico Montani  
Matricola 890107

Anno Accademico 2023-2024

# Indice

<b>1</b>	<b>Introduzione</b>	<b>4</b>
<b>2</b>	<b>Analisi dei requisiti</b>	<b>6</b>
2.1	Requisiti funzionali . . . . .	6
2.2	Requisiti non funzionali . . . . .	8
2.2.1	Efficienza . . . . .	8
2.2.2	Sicurezza . . . . .	9
2.2.3	Usabilità . . . . .	9
2.3	Stakeholder . . . . .	9
2.4	Strumenti utilizzati . . . . .	10
<b>3</b>	<b>Descrizione dell'implementazione</b>	<b>12</b>
3.1	Design . . . . .	12
3.1.1	Material Design . . . . .	12
3.1.2	Progettazione UI . . . . .	13
3.2	Architettura . . . . .	15
3.2.1	Schema Single activity - multiple fragments . . . . .	15
3.2.2	Flusso di esecuzione . . . . .	18
3.2.3	Standard MVVM . . . . .	19
3.3	Funzionalità . . . . .	22
3.3.1	Accesso: . . . . .	22
3.3.2	Visualizzazione mappa inquinamento . . . . .	23
3.3.3	Ricerca percorso . . . . .	24
3.3.4	Visualizzazione challenge . . . . .	27
3.3.5	Interazione con gli amici . . . . .	28
3.3.6	Visualizzazione e modifica informazioni profilo . . . . .	30
<b>4</b>	<b>Progetto di testing</b>	<b>32</b>
4.1	Test locali . . . . .	32
4.2	Test strumentati . . . . .	33
4.2.1	Test di integrazione . . . . .	33
4.2.2	Test di interfaccia utente . . . . .	33
4.3	Test di performance . . . . .	34
4.3.1	Perfetto . . . . .	34
4.3.2	Android Profiler . . . . .	35
4.4	Test di qualità del software . . . . .	36
4.4.1	SonarQube . . . . .	36

<b>5 Utilizzo di ChatGPT</b>	<b>37</b>
5.1 Analisi dei requisiti . . . . .	37
5.2 Descrizione dell'implementazione . . . . .	37
5.3 Progetto di testing . . . . .	38
5.4 Statistiche sull'utilizzo di ChatGPT . . . . .	39
<b>6 Conclusioni</b>	<b>40</b>

# Elenco delle figure

3.1	Light Mode . . . . .	13
3.2	Dark Mode . . . . .	13
3.3	Logo GreenWay . . . . .	13
3.4	Interfaccia Material Theme Builder . . . . .	14
3.5	Prototipo schermate . . . . .	15
3.6	Gestione esecuzione WelcomeActivity . . . . .	16
3.7	WelcomeActivity con fragment principale in esecuzione . . . . .	17
3.8	Gestione esecuzione MainActivity . . . . .	18
3.9	MainActivity con fragment principale in esecuzione . . . . .	18
3.10	Diagramma architettura MVVM Greenway . . . . .	21
3.11	Login . . . . .	22
3.12	Registrazione . . . . .	22
3.13	Inquinamento aria . . . . .	23
3.14	Legenda inquinamento . . . . .	23
3.15	Autocomplete Google . . . . .	24
3.16	Ricerca indirizzi . . . . .	24
3.17	Lista itinerari . . . . .	26
3.18	Percorso itinerario su mappa . . . . .	27
3.19	Fragment Challenge . . . . .	28
3.20	Tab Friend . . . . .	29
3.21	Fragment Challenge . . . . .	30
3.22	Informazioni utente . . . . .	31
3.23	Cambio informazioni . . . . .	31
4.1	Interfaccia trace Perfetto . . . . .	35
4.2	Interfaccia Android Profiler . . . . .	35
4.3	Interfaccia SonarQube . . . . .	36

# Capitolo 1

## Introduzione

In questa relazione verrà affrontata la tematica dell'efficacia e del supporto che possono fornire gli LLM (*Large Language Models*) per lo sviluppo di un software, in questo caso particolare ci concentreremo su **ChatGPT**.

Negli ultimi anni, gli LLM hanno acquisito una notevole popolarità per la loro capacità di assistere gli utenti in molti ambiti. Quest'ultimi, sono modelli basati su intelligenza artificiale e sono in grado di comprendere e generare testo coerente rispetto a quello che gli chiediamo. Nel contesto dello sviluppo software, l'utilizzo di ChatGPT o altri LLM può accelerare notevolmente il lavoro, aiutando gli sviluppatori, ad esempio, nella creazione di idee, correzione di errori, migliorare la qualità del prodotto o qualsiasi altro compito da svolgere.

Abbiamo sviluppato un'applicazione chiamata **GreenWay**, la quale ha come obiettivo primario di fornire all'utente la consapevolezza della quantità di CO<sub>2</sub> emessa in base agli spostamenti che compie.

La scelta di focalizzare l'applicazione sulla CO<sub>2</sub> è avvenuta dopo aver compresa la sua importanza informandoci attraverso articoli e pubblicazioni. La CO<sub>2</sub> è uno tra i gas ad effetto serra che più contribuisce al riscaldamento globale. Questo gas cattura il calore del sole impedendogli di tornare nello spazio. Attualmente, la concentrazione in atmosfera della CO<sub>2</sub> supera del **40%** il valore registrato agli inizi dell'era industriale ed è responsabile del **63%** del riscaldamento globale causato dall'uomo [Eur23][Anc19].

L'applicazione quindi, fornirà una schermata dove l'utente potrà inserire un determinato punto di partenza e di arrivo e successivamente verranno mostrati tutti i percorsi possibili per ogni tipo di trasporto, ovvero macchina, mezzi pubblici e a piedi, con associato il relativo consumo di CO<sub>2</sub> e il tempo di percorrenza.

Per rendere l'applicazione più interattiva, abbiamo sviluppato un sistema di amicizia e di sfide, cosicché gli utenti si possano confrontare, avendo l'opportunità di visualizzare lo stato di avanzamento degli amici nelle sfide, creando competizione e l'aumento dell'utilizzo di GreenWay.

Come schermata principale, abbiamo implementato una mappa, dove è possibile vedere l'inquinamento dell'aria in tutto il mondo (ove è possibile recuperare i dati) attraverso dei colori presenti sulla mappa e una legenda per confrontare, in modo chiaro, i valori della scala **US AQI**. Con questa funzionalità informiamo l'utente sulla qualità dell'aria, condizionata per una buona parte, dalla presenza di CO2.

L'utente ovviamente, ha a disposizione le funzionalità per la gestione del proprio profilo, come il cambio della password oppure della propria immagine. Può anche impostare il tipo di carburante che utilizza la sua auto di proprietà, andando così a ricevere dall'applicazione calcoli più precisi sul consumo di CO2 che produce quando selezionerà una tratta con l'automobile. Ha l'opportunità di poter vedere quanta CO2 ha risparmiato usando un determinato mezzo di trasporto e quanta ne ha consumata in totale con le tratte che ha percorso.

Durante la realizzazione dell'applicazione, abbiamo usufruito frequentemente di ChatGPT in ogni sua fase. Ci ha fornito spesso codice utile ma non preciso. Il problema, a volte, era dovuto alla scrittura del **prompt** che fornivamo a ChatGPT, spiegandogli male il contesto e il problema che stavamo affrontando. Il prompt, dovrebbe essere scritto specificando in modo dettagliato l'output che si vuole da ChatGPT e come dovrebbe essere generato incluse anche informazioni come il tono, la lunghezza, lo stile e la struttura [Cou24].

Per la fase di ideazione dell'UI, abbiamo utilizzato gli strumenti di Material Design 3 su *Figma*, mentre per il logo abbiamo chiesto a *Microsoft Copilot* (che utilizza bot di ChatGPT) per generare un'immagine adatta al contesto dell'app. Nella fase riguardante l'analisi dei requisiti, abbiamo utilizzato ChatGPT solo per richiedere delle idee sulle funzionalità dell'applicazione come ad esempio il sistema di sfide oppure la stesura del documento in modo opportuno fornendogli tutte le informazioni che avevamo ideato noi.  
Le fasi principali dello sviluppo di GreenWay, dove abbiamo utilizzato maggiormente ChatGPT, sono state quelle dell'implementazione, dove abbiamo richiesto del codice per risolvere problemi oppure delle idee su come gestire in modo più efficiente alcuni aspetti dell'applicazione, e la fase di testing, in cui ci è stato utile per creare una test suite, andando a verificare il funzionamento dei metodi e la comunicazione tra gli strati differenti, che compongono la nostra app.

Attualmente, l'uso esclusivo di ChatGPT senza avere conoscenze specifiche e la verifica dei contenuti generati non è ancora sufficiente. Pertanto, è fondamentale combinare l'uso di ChatGPT con competenze umane per garantire la precisione e l'affidabilità. Questo tema verrà trattato in modo dettagliato in seguito all'interno della relazione.

# Capitolo 2

## Analisi dei requisiti

Una fase importante del ciclo di sviluppo di un software è l'analisi dei requisiti, l'obiettivo è identificare le esigenze degli utenti e tradurle in requisiti specifici, che il team di sviluppo può utilizzare per progettare e sviluppare il sistema. Questo processo è di estrema importanza in quanto contribuisce a garantire che il prodotto finale soddisfi le parti interessate e gli utenti.

L'analisi dei requisiti è un processo di tipo **iterativo**, che continua per tutta la durata del progetto e può subire modifiche in ogni momento. Ciò comporta che deve essere flessibile e facilmente aggiornabile[App23].

In questa fase abbiamo definito: i requisiti, che si dividono in **funzionali** e **non funzionali**, stakeholder, che riguardano gli individui interessati al progetto, infine gli strumenti, che forniscono supporto per la realizzazione dei requisiti.

### 2.1 Requisiti funzionali

I requisiti funzionali definiscono: i servizi che devono essere rilasciati dal sistema, come deve reagire il sistema a particolari input, come il sistema si dovrebbe comportare in particolari situazioni. Talvolta, i requisiti funzionali possono anche definire che cosa il sistema non dovrebbe fare. [Som10]

Abbiamo definito i seguenti requisiti funzionali:

- **Visualizzazione della qualità dell'aria:**
  1. L'app deve implementare la mappa fornita da Google.
  2. L'app deve utilizzare le API fornite da Google per ricevere le immagini relative alla qualità dell'aria ed aggiungerle sopra la mappa.
  3. L'app deve permettere di poter geolocalizzare la posizione dell'utente, se quest'ultimo le fornisce le autorizzazioni.
  4. L'app deve mostrare la legenda per poter aiutare l'utente a leggere correttamente i colori visualizzati sulla mappa.

- **Calcolo e confronto produzione di CO2 in base al percorso:**

1. L'utente deve poter inserire un punto di partenza e di destinazione all'interno della schermata navigatore.
2. L'app deve calcolare la produzione di CO2 per diversi tipi di trasporto (auto, trasporto pubblico, a piedi) per raggiungere la destinazione.
3. L'app deve mostrare all'utente tutti i possibili percorsi, ordinati in base al consumo di CO2 con il relativo tempo di percorrenza e la lunghezza in km.
4. L'utente deve poter vedere il percorso sulla mappa dell'applicazione se seleziona il pulsante "*Apri mappa*".

- **Visualizzazione del risparmio e consumo di CO2:**

1. L'app deve tenere traccia del risparmio di CO2 dell'utente per ogni tipo di trasporto diverso.
2. L'app deve tenere traccia della CO2 consumata dall'utente.

- **Sistema di amicizie e sfide:**

1. L'utente deve poter aggiungere o rimuovere eventuali amici.
2. L'utente deve poter vedere lo stato di avanzamento delle sfide dei suoi amici.
3. L'app deve includere un sistema di sfide per gli utenti, incoraggiandoli a ridurre la loro produzione di CO2.
4. L'app deve tenere traccia dei punti dell'utente che vengono aggiornati una volta completate le sfide.

- **Sistema di login e registrazione:**

1. L'utente deve poter creare un profilo attraverso la propria mail.
2. L'utente deve poter creare un profilo utilizzando il suo account Google.
3. L'utente deve poter effettuare l'accesso e recuperare tutte le sue informazioni.
4. Se l'utente ha già effettuato l'accesso sul proprio dispositivo, viene abilitato l'auto-login.
5. L'utente deve poter effettuare il logout dal profilo registrato sul dispositivo.

- **Sistema di cambio informazioni utente:**

1. L'app deve permettere all'utente di poter cambiare la password dell'account.
2. L'app deve permettere all'utente di poter cambiare la foto profilo dell'account.
3. L'app deve permettere all'utente di poter cambiare il tipo di carburante che utilizza la propria macchina, di default sarà impostato a benzina.

- **Accessibilità:**

1. L'app deve supportare oltre che la lingua italiana, anche quella inglese per essere utilizzabile da un maggior numero di utenti e deve essere impostata in modo automatico a seconda della lingua del dispositivo.
2. L'app deve includere sia il tema chiaro che il tema scuro e impostarlo in modo automatico a seconda del tema del dispositivo.

## 2.2 Requisiti non funzionali

I requisiti non funzionali non sono direttamente connessi con le specifiche offerte dal sistema agli utenti. Essi definiscono le proprietà del sistema ad esempio: l'affidabilità, il tempo di risposta, lo spazio occupato dalla memoria, le prestazioni, la sicurezza o la disponibilità su diversi dispositivi [Som10]. Quindi, per migliorare l'esperienza dell'utilizzo all'utente, i requisiti non funzionali sono essenziali. Senza di essi l'utente potrebbe non utilizzare più il nostro software, a causa di attese eccessive, saturazione di memoria o, nei casi peggiori, fughe di dati personali.

### 2.2.1 Efficienza

I requisiti non funzionali sull'efficienza riguardano aspetti quali prestazioni e memoria che l'applicazione deve rispettare.

Noi abbiamo scritto i seguenti requisiti:

- **Prestazioni**

1. L'app deve utilizzare meno di 150 MB di memoria durante il funzionamento normale.
2. L'app deve essere ottimizzata per ridurre al minimo il consumo di batteria.
3. L'app deve scaricare in locale le immagini relative alla mappa termica di Google, che indica l'inquinamento dell'aria, in modo da ridurre il traffico dati e rendere più efficiente il caricamento della mappa.

- **Tempo di risposta:**

1. L'app deve completare il caricamento di apertura entro 3 secondi.
2. Le chiamate alle API devono rispondere entro 2 secondi in condizioni di rete normali.
3. L'aggiornamento delle immagini riguardanti l'inquinamento dell'aria deve essere eseguito ogni 2 ore.

- **Memoria**

1. L'app, una volta installata, deve occupare massimo 100 MB di memoria.

### **2.2.2 Sicurezza**

I requisiti non funzionali sulla sicurezza vanno a descrivere aspetti importanti per l'applicazione, andando a definire la modalità di protezione e di garanzia di integrità dei dati:

1. L'app deve proteggere i dati sensibili degli utenti utilizzando la crittografia.
2. Tutti i dati trasmessi tra il dispositivo dell'utente e il database devono essere criptati utilizzando il protocollo HTTPS.
3. L'accesso in lettura o scrittura al database, deve essere definito da regole specifiche.

### **2.2.3 Usabilità**

I requisiti non funzionali che riguardano l'usabilità vanno a definire aspetti specifici che influiscono sull'esperienza dell'utente:

1. L'interfaccia utente deve essere reattiva.
2. L'app deve avere un'interfaccia intuitiva.
3. L'app deve essere facile da usare e da navigare, utilizzando menu chiari e coerenti.
4. L'app deve essere disponibile per dispositivi Android (da Android 12.0 in su).

## **2.3 Stakeholder**

Gli Stakeholder comprendono tutti gli individui, gruppi o organizzazioni che in qualche modo sono interessate al progetto, dallo sviluppo, all'utilizzo dell'applicazione.

Questi quindi, sono coloro che hanno un'influenza indiretta sui requisiti di sistema, le parti interessate vanno dall'utente finale che interagirà col sistema, a chiunque altro nell'organizzazione che partecipa al progetto.

- **Utenti:**

1. Persone di tutte le età, interessate a ridurre la propria emissione di CO<sub>2</sub>.
2. Persone consapevoli dell'inquinamento atmosferico, che si vogliono informare sulla qualità dell'aria in tempo reale.

- **Sviluppatori:**

1. Team di sviluppatori che creeranno l'app.

## 2.4 Strumenti utilizzati

La sezione degli strumenti utilizzati nell'analisi dei requisiti è una parte del documento molto importante perché vengono elencati e descritti gli strumenti, librerie oppure servizi esterni che vengono utilizzati durante lo sviluppo del progetto.

Questo aiuta a fornire una visuale chiara delle tecnologie coinvolte e a comprendere meglio l'ambiente di lavoro.

- **SDK**

1. GOOGLE MAPS SDK: utilizzata per implementare la mappa e funzione di geolocalizzazione [Goo23b].
2. GOOGLE PLACES SDK: utilizzata per la ricerca dei luoghi di partenza e di arrivo attraverso l'auto-completamento [Goo23c].

- **API**

1. GOOGLE AIRQUALITY API: utilizzata per scaricare la mappa termica che riguarda l'inquinamento dell'aria, che viene sovrapposta alla mappa Google [Goo24a].
2. GOOGLE ROUTES API: utilizzata per calcolare i possibili itinerari tra due luoghi [Goo23a].

- **Database**

1. FIREBASE: utilizzato come database remoto e strumento di autenticazione dell'utente [Goo24c][Goo24b].

- **UI Design**

1. MATERIAL DESIGN 3: strumento open-source, che fornisce componenti, strumenti e linee guida per supportare la creazione di interfacce grafiche utente in modo più professionale secondo le linee guida Google [Goo24e].
2. FIGMA: strumento online, che permette di progettare le schermate di un'applicazione. Utilizzando il plug-in di Material Design 3, offre la possibilità di generare una paletta di colori e dare un'anteprima fedele del risultato finale dell'interfaccia grafica utente [Fig24].

- **Testing**

1. FIREBASE: utilizzato per testare l'applicazione su un pool vasto di modelli di smartphone, in modo da verificare l'alta usabilità [Goo24d].
2. ESPRESSO: utilizzato per testare le interazioni tra componenti grafici e azioni a loro associate[And24a].
3. MONKEY: utilizzato per testare l'affidabilità dell'applicazione, poiché fornisce una sequenza di gesture casuali, che permettono di identificare eventuali bug [And23].
4. ROBOLECTRIC: utilizzato per testare strati e classi dell'applicazione, che non comprendono l'interfaccia grafica [Rob24].

5. SONARQUBE: utilizzato per identificare eventuali code smell e calcolare la copertura dei test eseguiti [Son24].
6. ANDROID PROFILER: utilizzato per verificare la prestazione e la memoria occupata dall'applicazione durante la sua attività [And24b].

# Capitolo 3

# Descrizione dell'implementazione

Nel capitolo che riguarda l'implementazione verrà illustrato tutto il processo di sviluppo che ha caratterizzato la creazione della nostra applicazione. Per avere una panoramica il più completa possibile, il capitolo inizierà parlando del design, successivamente dell'architettura che abbiamo seguito per la costruzione dell'applicazione ed infine, si andrà a descrivere le funzionalità principali spiegando come sono state implementate.

## 3.1 Design

All'interno di questo sotto paragrafo relativo al design, discuteremo il processo di studio e gli strumenti utilizzati, la scelta di seguire le linee guida di Google per avere un'interfaccia efficace e intuitiva attraverso il framework **Material Design**.

### 3.1.1 Material Design

Material design offre una vasta gamma di componenti XML. Questi, appartenendo allo stesso stile, rendono l'applicazione più intuitiva e armoniosa agli occhi dell'utente. Ogni componente ha delle regole ben definite sullo stile di ogni componente, sia dal punto di vista strutturale, ad esempio i margini del testo, la forma di un bottone, la disposizione di componenti all'interno di una card, che dal punto di vista cromatico. In questo modo i colori della paletta, precedentemente generata, vengono utilizzati nella maniera corretta, sia in modalità light-mode[3.1], sia in modalità dark-mode[3.2], in base alle impostazioni del dispositivo.

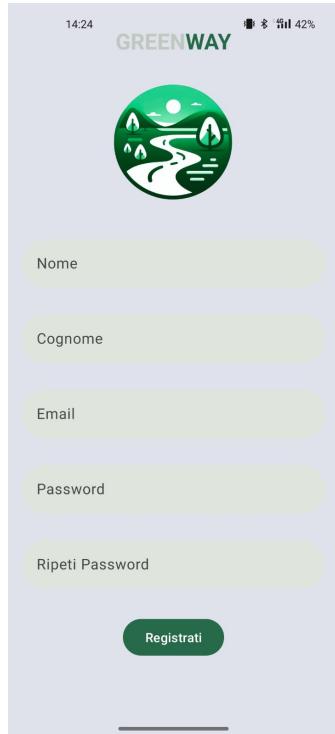


Figura 3.1: Light Mode

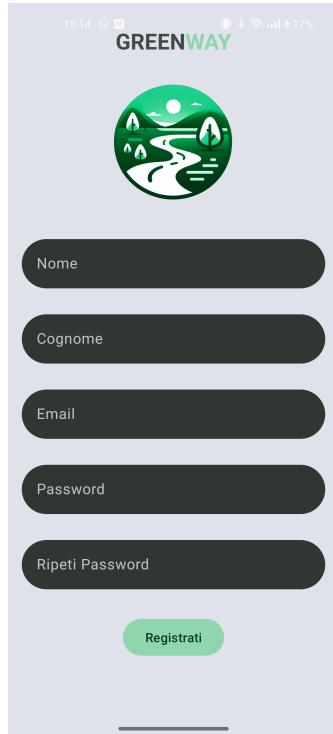


Figura 3.2: Dark Mode

### 3.1.2 Progettazione UI

Riguardo alla progettazione della UI abbiamo utilizzato lo strumento Figma, per la creazione e prototipazione delle schermate delle interfacce utente.

Per iniziare abbiamo cercato di capire quali colori potessero andare bene tra di loro per comporre i vari elementi, che faranno parte dell'interfaccia utente.

Inizialmente abbiamo chiesto aiuto all'intelligenza artificiale di Microsoft, Copilot (utilizza i bot di ChatGPT 4), per creare il logo dell'applicazione, fornendogli il nome GreenWay e il colore principale, ovvero il verde. Tra tutte le proposte che ci ha fornito, abbiamo scelto quella che ci soddisava di più [3.3].



Figura 3.3: Logo GreenWay

In seguito abbiamo usufruito di un plug-in fornito da Figma che si chiama **Material Theme Builder**. Questo plug-in, fornendogli l'immagine del logo o qualsiasi altro elemento composto dai colori primari dell'applicazione, crea la paletta di colori che seguono le linee guida di Google per gli elementi di Material Design come è possibile vedere dall'immagine, in modo da creare il miglior contrasto di colori nell'applicazione. [3.4].

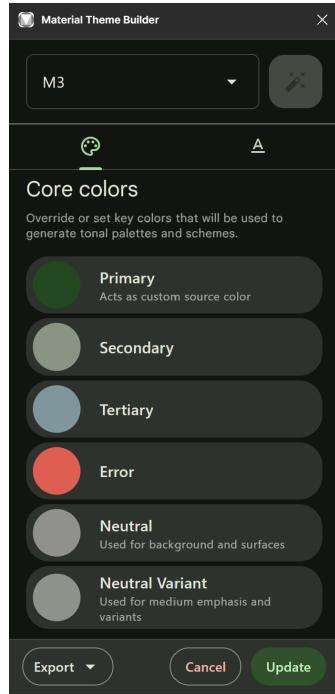


Figura 3.4: Interfaccia Material Theme Builder

In seguito verranno creati i seguenti colori con le relative varianti: [Des24]

- **Primary:** Colori utilizzati per i componenti più importanti all'interno dell'interfaccia utente come pulsanti, text input e altri elementi principali.
- **Secondary:** Colori utilizzati per i componenti che hanno meno influenza nella UI come ad esempio, il colore delle icone quando vengono selezionate nella barra di navigazione.
- **Tertiary:** Colori utilizzati per accettare i contrasti che servono per bilanciare i contrasti tra colore primario e secondario oppure attirano un'attenzione verso un determinato elemento come un campo di input selezionato.
- **Error:** Colori utilizzati per comunicare gli errori, ad esempio una password sbagliata che è stata inserita in un campo di testo.
- **Neutral:** Colori utilizzati per gli sfondi e per le superfici.
- **Neutral Variant:** Colori utilizzati per le enfasi medie e per le varianti.

Dopo aver generato la palette di colori, siamo stati in grado di definire una bozza di quelle che sarebbero state le schermate dell'app[3.5].

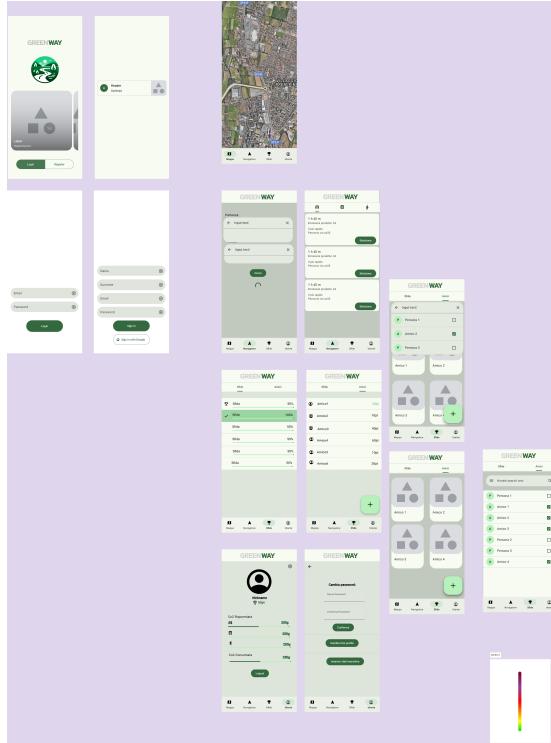


Figura 3.5: Prototipo schermate

## 3.2 Architettura

In questa sezione, verrà descritta l'architettura utilizzata per lo sviluppo dell'applicazione Greenway.

La scelta di un'architettura adeguata è fondamentale per garantire che l'applicazione sia scalabile e facilmente estendibile. La nostra applicazione utilizza un'architettura basata su un'unica attività (Single Activity) con molteplici frammenti (Multiple Fragments) e segue lo standard MVVM (Model-View-ViewModel) definita da Android.

### 3.2.1 Schema Single activity - multiple fragments

Questo schema permette una gestione più modulare delle interfacce utente, è caratterizzato da un contenitore principale (Activity) che, al suo interno, presenta diversi fragmenti.

Utilizzando questa modalità possiamo migliorare la navigazione e l'utilizzo di risorse. Un fragment non può vivere da solo ma deve essere "ospitato" da una activity oppure da un altro fragment, possiede il proprio ciclo di vita, definisce e gestisce il proprio layout ed è in grado di gestire i propri eventi di input.

In seguito verrà spiegata la suddivisione delle due activity principali che compongono l'applicazione e i loro relativi fragment.

### WelcomeActivity

La prima activity che caratterizza l'applicazione è la *WelcomeActivity*.

Quest'ultima comprende tutti i fragment iniziali ovvero, quello di benvenuto con la prima schermata e poi quelli di login e registrazione per eseguire l'accesso al profilo. Una volta completato l'accesso verrà inizializzata la *MainActivity* e verrà chiusa la *WelcomeActivity*, come è possibile vedere nell'immagine sottostante. [3.6]

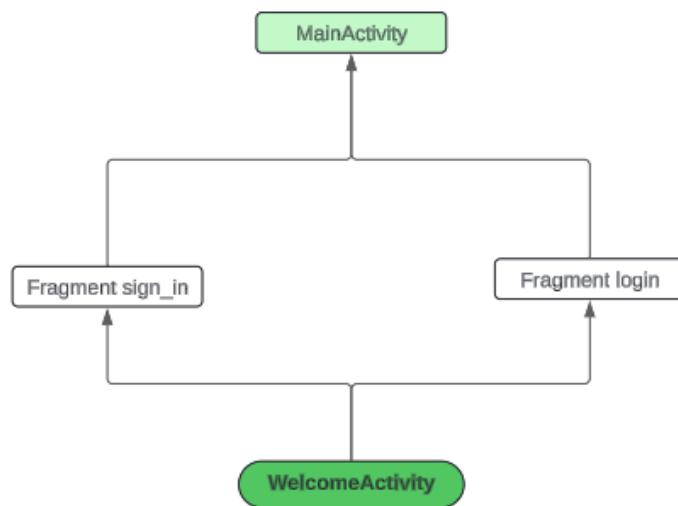


Figura 3.6: Gestione esecuzione WelcomeActivity



Figura 3.7: WelcomeActivity con fragment principale in esecuzione

### MainActivity

La seconda activity principale è la *MainActivity*, contiene tutto il resto dei fragment che costituiscono le altre schermate dell'applicazione per le funzionalità di cui usufruisce l'utente.

La gestione dei fragment principali è affidata alla barra di navigazione che, una volta selezionata una schermata tra le quattro disponibili, inizierà un nuovo ciclo di vita per quello selezionato e distruggerà quello prima. In alcuni di questi quattro fragment sono presenti altre azioni, come la selezione di un bottone, che permettono l'apertura di altri frammenti a partire da un fragment anziché da un'Activity. In questo caso però, il fragment vecchio non verrà distrutto ma messo in attesa cosicché, quando verrà chiuso il fragment nuovo, verrà ripreso quello prima con la medesima istanza in esecuzione[3.8].

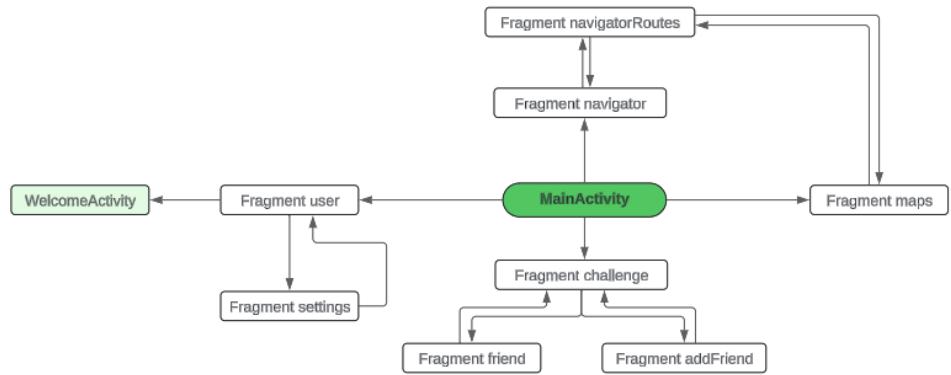


Figura 3.8: Gestione esecuzione MainActivity

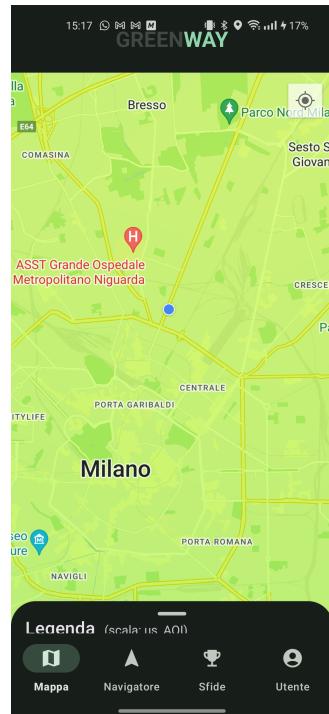


Figura 3.9: MainActivity con fragment principale in esecuzione

### 3.2.2 Flusso di esecuzione

L'applicazione, per rendere l'esperienza utente fluida e coerente, ha un flusso di esecuzione che segue un percorso logico e sequenziale. Questi sono i passaggi principali del flusso:

1. **Avvio dell'applicazione:** All'avvio di Greenway, viene caricata la *WelcomeActivity*, che introduce l'utente all'applicazione attraverso delle immagini e gestisce l'autenticazione dello user.
2. **Autenticazione dell'utente:** L'utente, al primo avvio dell'applicazione, deve completare l'autenticazione attraverso il login o la registrazione o tramite account Google. Se l'utente si è già autenticato la prima volta, nei successivi avvi è presente l'auto-login, quindi questa fase viene saltata. Dopo quest'ultima, viene visualizzata la *MainActivity*.
3. **Inizializzazione della *MainActivity*:** Dopo che la *WelcomeActivity* è stata chiusa, viene inizializzata la *MainActivity*. La *MainActivity* carica la barra di navigazione e il fragment principale, che include una mappa termica interattiva dell'inquinamento dell'aria con una legenda dedicata.
4. **Navigazione e utilizzo dell'app:** Grazie alla barra di navigazione, l'utente può navigare tra i diversi fragment e quindi accedere alle altre funzionalità offerte da GreenWay. Ognuna ha un fragment dedicato in modo da rendere più intuitiva l'esperienza utente.
5. **Gestione dello stato:** Mentre l'utente naviga tra i diversi fragments, lo stato dell'applicazione viene gestito dal ViewModel, secondo le regole del pattern MVVM, in questo modo i dati vengono mantenuti e sincronizzati tra il Model e la View.
6. **Chiusura dell'applicazione:** Quando avviene la chiusura dell'applicazione, l'activity e i fragments vengono terminati in modo ordinato, mentre vengono salvati gli stati necessari per la riapertura successiva, come lo stato dell'autenticazione o l'orario dell'ultimo aggiornamento della mappa termica.

### 3.2.3 Standard MVVM

Lo standard MVVM (*Model View ViewModel*) è un tipo di architettura consigliata da Android, per ottenere un'applicazione reattiva a più livelli e un flusso di dati unidirezionale in tutti i livelli dell'app [3.10]. [Dev24]

Per seguire questo standard, bisogna organizzare gli elementi che compongono l'applicazione in due livelli principali:

- **Livello UI:** La sua funzione principale è quella di visualizzare i dati dell'applicazione, ogni volta che i dati cambiano a causa dell'utente, l'interfaccia deve essere in grado di aggiornarsi in tempo reale.

Il livello UI è costituito da due elementi:

- **Elementi UI:** Mostrano i dati sullo schermo, questi possono essere creati attraverso gli elementi XML.
- **Contenitori di stato:** chiamati anche **ViewModel**, contengono i dati, li espongono all'interfaccia utente e gestiscono la logica di aggiornamento.

- **Livello dati:** Questo livello contiene la *Logica di business*, ovvero le regole con cui l'applicazione crea, salva e modifica i dati.  
È composta da due componenti principali:

- **Repository:** Lo strato di repository è uno strato intermedio tra i ViewModel e i Data Source ed è incaricato di gestire i dati richiesti dal ViewModel. Le sue funzioni principali sono la raccolta di dati dalle sorgenti (Data Source) che possono essere remote o locali, la sincronizzazione dei dati, la gestione degli errori attraverso l'uso delle callback e la fornitura dei dati al ViewModel.
- **Data Source:** Lo strato dei data source rappresenta le varie sorgenti di dati da cui l'applicazione può recuperare le informazioni. Queste si possono dividere in locali e remote.

Come sorgente locale abbiamo utilizzato il database *Room*, fornito da Android Studio che ci permette il salvataggio di alcuni dati come le immagini sull'inquinamento atmosferico, che vengono applicate alla mappa.

Per quanto riguarda le sorgenti remote, abbiamo *Firebase Realtime Database* e *Firebase Authentication*, che utilizziamo rispettivamente come database principale per il salvataggio di tutte le informazioni degli utenti e l'altro come servizio per eseguire l'accesso all'applicazione.

Come ultima fonte remota di dati, ci sono tutti i servizi di API che permettono di avere le informazioni necessarie, affinché la nostra app funzioni correttamente.

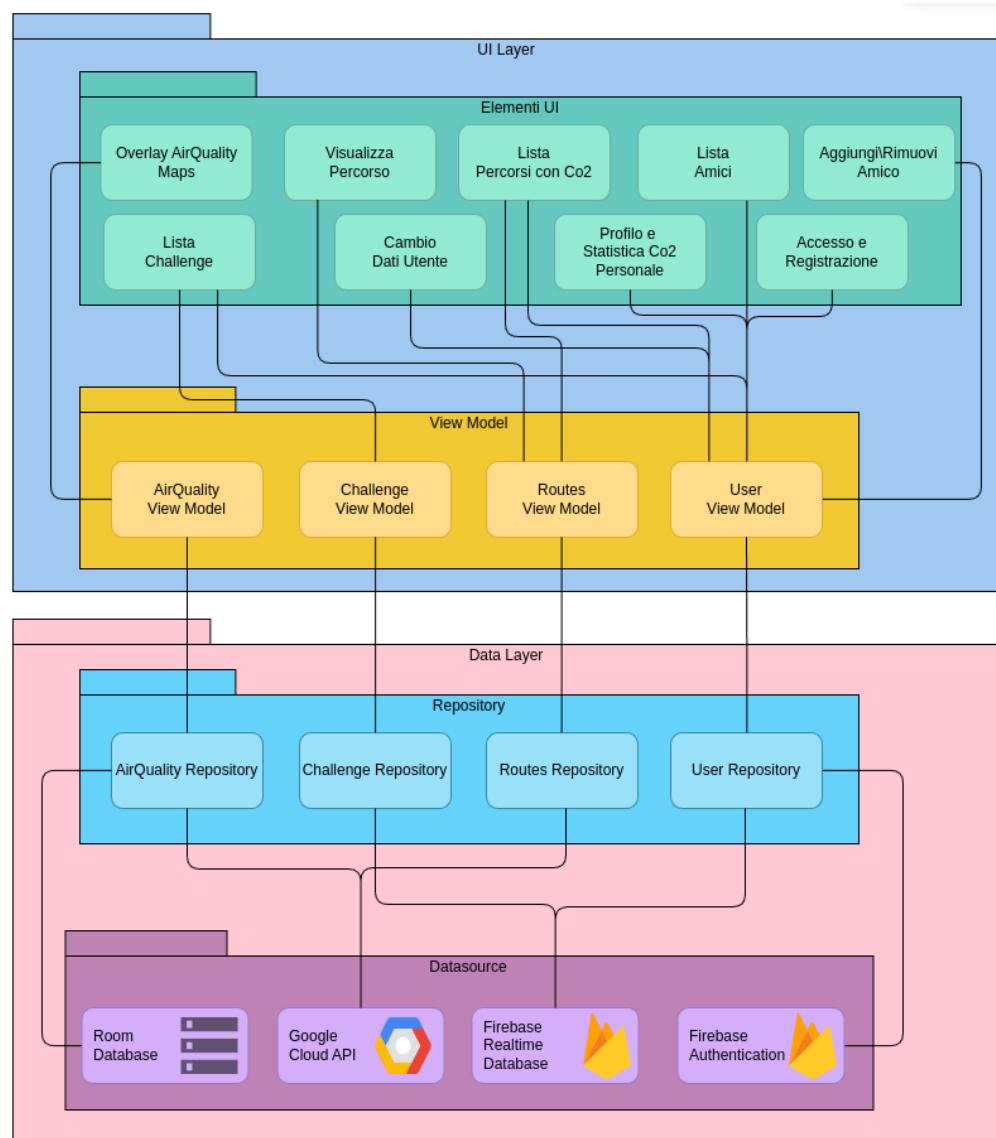


Figura 3.10: Diagramma architettura MVVM Greenway

### 3.3 Funzionalità

In quest'ultima parte della descrizione dell'implementazione, verranno spiegate e illustrate le funzionalità disponibili all'interno dell'applicazione GreenWay. Si dividono principalmente in:

#### 3.3.1 Accesso:

Per descrivere questa funzionalità, dobbiamo prendere in considerazione ciò che avviene in locale sul dispositivo e successivamente, come si comporta Firebase Authentication. Nel caso in cui un utente si voglia registrare [3.12], il dispositivo controlla che mail e password siano accettabili e li trasmette a Firebase Authentication, che successivamente si occupa di creare un identificativo, in modo da avere un nuovo nodo utente all'interno del Realtime Database. Nel caso in cui l'utente fosse già registrato e volesse effettuare il login [3.11], Firebase Authentication verifica se le credenziali inserite sono corrette.

Utilizzando invece, il login con un account Google, Firebase Authentication utilizza le credenziali associate, sia come registrazione, che come accesso, in questo modo rende più semplice e veloce il processo di autenticazione dell'utente.

Dopo che lo user ha eseguito il primo accesso sulla propria app, lo stato del suo account viene salvato in locale, in questo modo, ad una nuova apertura dell'applicazione, il dispositivo trasmette immediatamente a Firebase Authentication le credenziali associate all'account, effettuando in modo automatico l'accesso.

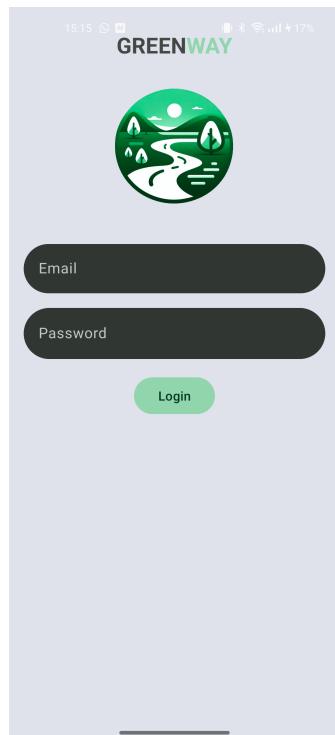


Figura 3.11: Login

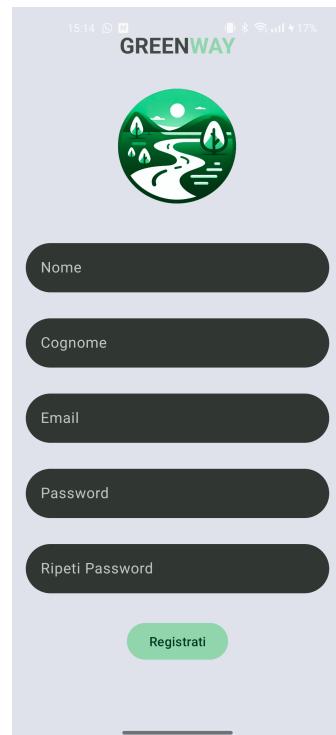


Figura 3.12: Registrazione

### 3.3.2 Visualizzazione mappa inquinamento

Per la visualizzazione della mappa con l'inquinamento [3.13], bisogna aver effettuato l'accesso e l'applicazione porterà l'utente direttamente al *MapFragment*, essendo questo il fragment di default della *MainActivity*.

Se lo user ha effettuato per la prima volta l'accesso, quando verrà aperto il *MapFragment*, verrà effettuata una chiamata alla API di **Google Air Quality** per ricevere indietro le immagini, che verranno, una volta scaricate tutte, posizionate sopra alla mappa. Queste immagini inoltre, vengono salvate sul database locale **Room**, per evitare ripetute chiamate con medesimo risultato e velocizzare il caricamento sulla mappa di quest'ultime.

Google Air Quality aggiorna le immagini riguardanti l'inquinamento ogni 2 ore [Goo24a], quindi noi abbiamo fatto in modo tale che, ogni volta che viene effettuato un download, venga salvato l'orario all'interno di un file criptato nella memoria locale del dispositivo e se, l'intervallo di tempo dell'accesso successivo supera le due ore viene eseguito l'aggiornamento.

All'interno del fragment è possibile anche visualizzare la legenda dei valori su scala US\_AQI [3.14], così da permettere all'utente di comprendere in modo più rapido i colori presenti sulla mappa.

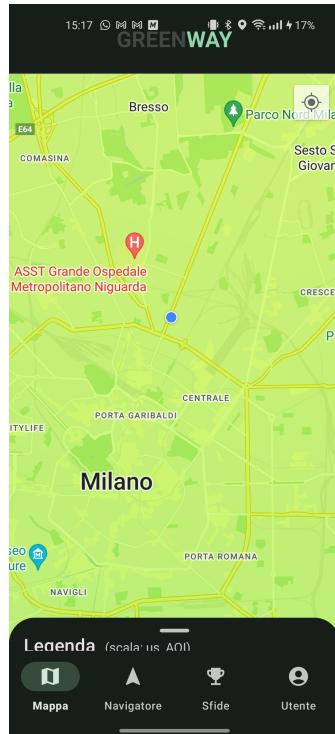


Figura 3.13: Inquinamento aria

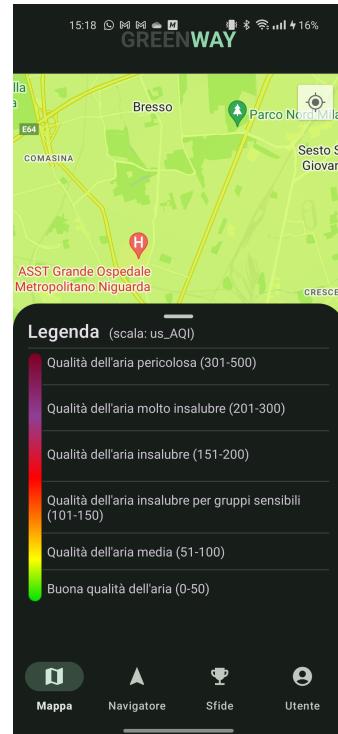


Figura 3.14: Legenda inquinamento

### 3.3.3 Ricerca percorso

La funzionalità che riguarda la ricerca del percorso, è quella che caratterizza maggiormente l'applicazione, ma è anche una delle più complesse. Per fornire una descrizione il più completa possibile, verrà divisa in due parti seguendo le azioni che deve compiere l'utente.

#### Ricerca indirizzo

Il primo step che bisogna effettuare è la ricerca dell'indirizzo di **partenza** e quello di **destinazione**.

Lo user si ritroverà con due text input [3.16], presenti nella schermata che, una volta aperti, verrà avviato un **fragment Autocomplete** che utilizza le SDK di Google per la ricerca degli indirizzi [3.15].

Quando viene selezionato l'indirizzo, il fragment Autocomplete viene chiuso e vengono salvati in locale le coordinate di latitudine e longitudine del luogo che l'SDK ritorna come risultato. Una volta che vengono compilati entrambi i campi, lo user eseguirà la ricerca attraverso il bottone "Cerca" e verrà aperto il nuovo fragment con i relativi risultati.



Figura 3.15: Autocomplete Google

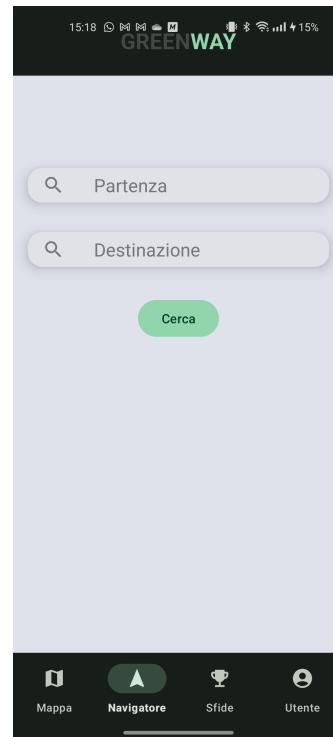


Figura 3.16: Ricerca indirizzi

#### Selezione itinerario

Il fragment riguardante la selezione degli indirizzi, una volta avviata la ricerca, manderà al fragment, per la scelta dell'itinerario, le coordinate *X* e *Y* di entram-

bi i luoghi. A questo punto viene eseguita la chiamata all'API **Routes** di Google per la ricerca dei percorsi. Questa chiamata ritorna un oggetto di tipo *Route*, in formato json, che è composto a sua volta da diversi oggetti, ognuno contenenti informazioni diverse. Google ha reso possibile personalizzare la chiamata, per far ritornare come risultato solo le informazioni a noi necessarie, quindi noi avremo l'oggetto *Route* con dati come la lunghezza del tragitto, il tempo di percorrenza, il tipo di mezzo utilizzato (Auto, mezzi pubblici e a piedi) e la *polyline*.

La polyline è una stringa di caratteri che viene codificata da Google per avere il tracciato del percorso sulla mappa, questa è utile, perché avendo il bottone "Apri Mappa", siamo in grado di mostrare il percorso sulla mappa con l'inquinamento atmosferico.

Una volta convertiti da formato json in oggetti java tutti i percorsi, li riordiniamo in tre liste, per dividerli in base al mezzo di trasporto utilizzato. In seguito viene calcolato, per ogni itinerario, l'emissione di CO2 in base ai metri percorsi.

Per l'automobile viene utilizzato il valore di emissione che è stato impostato, all'interno della pagina utente, in base al carburante che utilizza la macchina di proprietà, oppure il valore personalizzato che ha inserito lui nel caso fosse a conoscenza delle emissioni.

Nel caso l'utente non avesse impostato nulla, di default, ci sarà il valore delle emissioni di un motore a benzina.

Per i mezzi pubblici invece, l'API Routes ritorna anche il tipo di mezzo pubblico specifico che bisogna utilizzare in quella tratta, che compone il percorso, quindi siamo in grado di dare stime di produzione di CO2 più precise, andando a sommare i valori medi di emissione per ogni mezzo di trasporto utilizzato, ovviamente prima vengono moltiplicati per la distanza in metri.

Nel caso degli itinerari a piedi, abbiamo scelto di mettere come standard l'emissione a zero, visto che la produzione è talmente minima che tende ad un valore nullo.

Riguardo alla ricerca dei valori medi di consumo della CO2, ci siamo affidati a ChatGPT, confrontando anche i risultati che ci ha fornito con delle altre ricerche (Questo argomento verrà trattato in modo più dettagliato nel capitolo 5 "Utilizzo di ChatGPT").

Una volta eseguito il calcolo delle emissioni, viene effettuato un ordinamento all'interno delle tre liste in base alla produzione di CO2 in ordine crescente andando in seguito, a stamparle all'interno dei tre tab divisi per mezzo di trasporto [3.17].

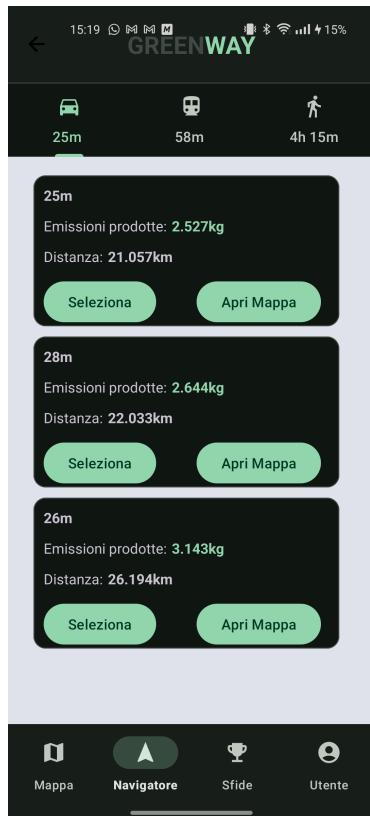


Figura 3.17: Lista itinerari

### Bottone seleziona

Se viene cliccato il bottone "seleziona", vuol dire che l'utente ha effettuato uno specifico percorso. In seguito, vengono aggiornate le statistiche del profilo come la CO<sub>2</sub> risparmiata (andando a fare la differenza con l'itinerario con consumo più alto), la CO<sub>2</sub> prodotta ed, eventualmente, le challenge che rientrano tra quelle in cui si può aumentare il progresso.

### Bottone apri mappa

Se invece, viene cliccato il bottone "Apri Mappa", verrà aperto il fragment della mappa con l'inquinamento atmosferico e mostrato il percorso al suo interno attraverso la polyline decodificata associata all'itinerario [3.18].



Figura 3.18: Percorso itinerario su mappa

#### 3.3.4 Visualizzazione challenge

Per quanto riguarda la visualizzazione delle challenge, il dispositivo esegue due chiamate al database remoto, una per scaricare l'elenco delle challenge, con la descrizione e il corrispettivo valore per essere completata, dal nodo dedicato, mentre la seconda per scaricare i progressi delle challenge presenti nel nodo utente corrispondente. Dopo aver ricevuto questi dati, il fragment carica la recycle view con l'elenco delle challenge e i progressi associati ad ognuna di esse [3.19]. Nel caso in cui la challenge è stata completata, la card associata cambia colore e icona. In questo modo l'utente potrà riconoscere in modo immediato quali challenge ha già completato.



Figura 3.19: Fragment Challenge

### 3.3.5 Interazione con gli amici

L'applicazione mette a disposizione un sistema di amicizie, a cui si può avere accesso dal fragment delle challenge, attraverso il tab presente nella schermata, il quale aggiorna la recycler view e mostra l'elenco degli utenti amici in ordine alfabetico.

Durante il caricamento del challenge fragment, il dispositivo esegue una chiamata al database remoto, chiedendo l'elenco degli amici presente nel nodo dell'account dell'utente. In questo modo, il contenuto di ogni card, che rappresenta un amico, contiene la foto profilo, il nome e il punteggio maturato completando le challenge. Dopo che l'utente avrà selezionato una card, verrà caricato un fragment, simile al *ChallengeFragment*, il quale mostra il progresso delle challenge dell'amico [3.20].

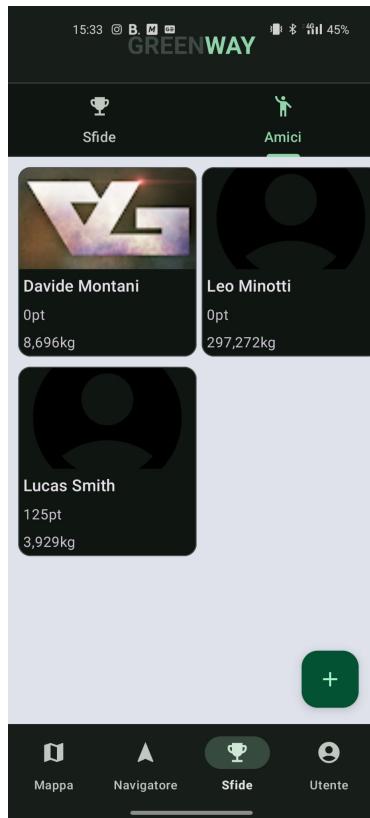


Figura 3.20: Tab Friend

Tornando alla pagina degli amici è inoltre possibile aggiungere un amico, utilizzando il bottone FAB in basso a destra, il quale carica il fragment *AddFriendFragment*. In questo fragment il dispositivo scarica la lista di tutti gli utenti presenti sul database remoto e li riporta in una reclycle view, se un account è già seguito riporterà una spunta verde. Cliccando sull'account sarà possibile aggiungere o rimuovere l'istanza dalla propria lista di amici in tempo reale [3.21].

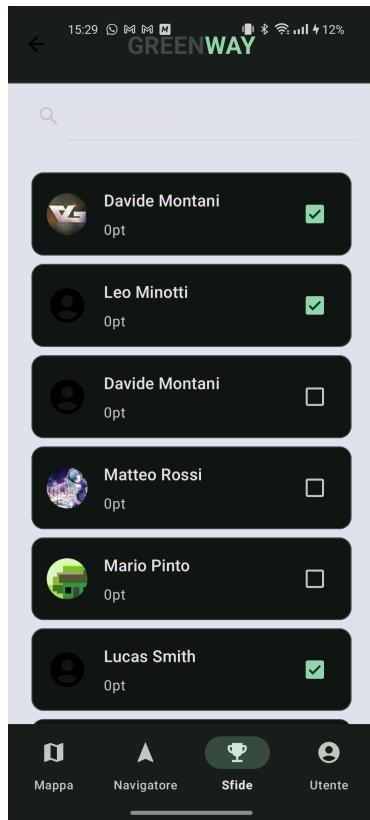


Figura 3.21: Fragment Challenge

### 3.3.6 Visualizzazione e modifica informazioni profilo

Per quanto riguarda le informazioni dell'utente si possono vedere, aprendo il *UserFragment* dalla barra di navigazione.

All'interno è possibile vedere tutte le informazioni, come la CO<sub>2</sub> risparmiata divisa per ogni tipo di trasporto, la CO<sub>2</sub> prodotta, il nome utente, la foto profilo, i punti dati dalle challenge completate e la possibilità di eseguire il logout [3.22]. Tutte queste informazioni vengono prese da Realtime Firebase attraverso l'identificativo dello user loggato nella sessione corrente.



Figura 3.22: Informazioni utente

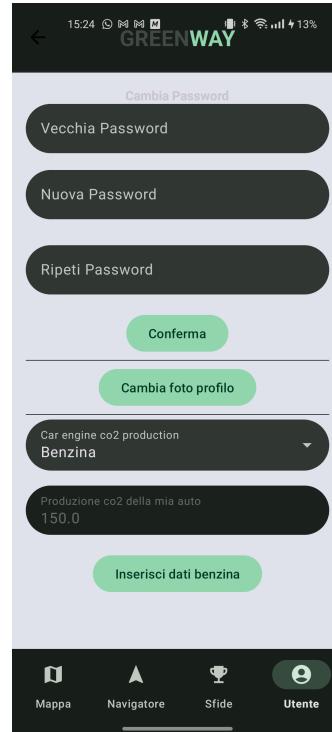


Figura 3.23: Cambio informazioni

Se invece, lo user volesse cambiare alcune informazioni, gli basterà cliccare l'icona delle impostazioni e verrà indirizzato al fragment apposito. In questa schermata, sarà in grado di cambiare la password, andando a scriverne una nuova, cambiare la foto profilo, dove l'applicazione chiederà i permessi per accedere alla fotocamera, se non sono stati già concessi, e potrà scattare la nuova foto.

Inoltre, è possibile cambiare il tipo di carburante che la propria macchina utilizza tra quelli preimpostati, oppure un valore specifico delle emissioni nel caso sappia questa informazione, per andare ad avere i calcoli della produzione di CO<sub>2</sub> più precisi[3.23].

Tutti questi cambiamenti verranno sovrascritti su Realtime Firebase e Firebase Authentication (Quest'ultimo solo nel caso del cambio password).

# Capitolo 4

# Progetto di testing

In questo capitolo verrà discusso tutto il processo di testing che è stato effettuato. Questa è una delle ultime parti di un processo di sviluppo, ma anche una delle più importanti. Una fase di testing accurata è fondamentale per avere la garanzia che l'applicazione funzioni correttamente e soddisfi tutti i requisiti.

Il testing, oltre ad aiutare ad identificare bug e malfunzionamenti, assicura che l'app sia affidabile e performante. In questo capitolo verranno descritti i diversi tipi di test eseguiti, le metodologie adottate e gli strumenti utilizzati con, in alcuni casi specifici, i relativi risultati.

## 4.1 Test locali

I test locali, chiamati anche *host-side test*, sono i test che vengono eseguiti sulla macchina di sviluppo o su un simulatore.

Solitamente, sono di piccole dimensioni, veloci e isolano l'oggetto del test dal resto dell'app.

I test locali si dividono in diverse categorie, noi abbiamo implementato una specifica tipologia chiamata **test unitari**.

Un test unitario è un sistema con cui singole unità di codice sono sottoposte a test per determinare se sono idonee per l'uso. Nella programmazione ad oggetti un'unità è spesso un'intera classe oppure un singolo metodo.

Questi test dovrebbero isolare il componente da testare e rendere possibile la ripetizione del test un numero indefinito di volte.[Val14]

Per lo sviluppo di questi test, ci siamo affidati al framework **Robolectric** e alla libreria **jUnit**, questi ci hanno permesso di creare classi di test per verificare il funzionamento di tutte le singole classi.

Abbiamo fatto test sulle classi, che non avevano relazioni con altre, come ad esempio quelle all'interno del package *util*, andando a verificare tutti i singoli metodi che erano contenuti all'interno di esse creando dei casi di test con valori limite ed intermedi.

Inoltre, abbiamo testato anche gli strati dell'applicazione come repository e data source andando ad utilizzare i *mock*.

I *mock* sono oggetti che simulano il comportamento di componenti reali. Vengono utilizzati per isolare il componente che stiamo testando, così da permetterci di verificare il suo funzionamento in maniera singolare. Utilizzando i *mock* generati dalla libreria **Mockito**, abbiamo potuto testare ogni metodo all'interno delle classi degli strati, assicurandoci il corretto funzionamento.

## 4.2 Test strumentati

Dopo aver eseguito i test di unità ed aver verificato l'affidabilità delle singole unità di codice, possiamo passare ai test strumentati, i quali sono una categoria di test eseguiti su un dispositivo o emulatore Android. In questo modo si può verificare il corretto funzionamento delle applicazioni, interagendo direttamente con il sistema operativo e l'hardware del dispositivo.

I test strumentati includono i test di integrazione e i test di interfaccia utente.

### 4.2.1 Test di integrazione

I test di integrazione si occupano della corretta interazione e integrazione tra i diversi componenti o strati di un'applicazione. Più nello specifico, si testa l'interazione tra *viewModel*, *casi d'uso* e *Repository*.

Grazie all'aiuto di *jUnit*, *Mockito* e *Robolectric*, siamo riusciti a verificare il corretto funzionamento del sistema, una volta che le componenti interagiscono tra di loro, in modo da assicurare che le integrazioni tra i moduli siano corrette e le dipendenze soddisfatte.

### 4.2.2 Test di interfaccia utente

Per quanto riguarda i test di interfaccia utente (UI test), andremo a testare lo strato che include *fragment* e *viewModel*. È importante eseguire questo tipo di test per assicurarsi che l'interfaccia utente si comporti come previsto e fornisca un'esperienza utente coerente e senza errori.

L'obiettivo di questi test quindi è di:

1. controllare che gli elementi xml(bottoni, barre di ricerca, mappe, ecc.) siano presenti, visibili e correttamente posizionati nelle schermate dell'applicazione.
2. garantire il corretto funzionamento delle interazioni dell'utente con i componenti xml.
3. verificare che le azioni dell'utente producano i risultati attesi, quindi l'aggiornamento in tempo reale dei dati, il passaggio alle schermate corrette o la visualizzazione di messaggi di errore.

Per assicurarsi che tutti questi punti siano soddisfatti, ci siamo affidati a i seguenti tool:

#### **Monkey:**

tool che permette la generazione di azioni casuali da parte dell'utente. Questa pratica ritorna utile per identificare eventuali bug o crash improvvisi nel sistema, poiché utilizza un flusso inusuale di azioni utente.

#### **Firebase TestLab:**

grazie a *Firebase TestLab* si ha la possibilità di far eseguire virtualmente la propria applicazione su un ampio pool di dispositivi e versioni di Android. Quindi permette di ad esempio di verificare se i componenti xml siano posizionati nel posto giusto nella schermata, nonostante la differenza di grandezza dello schermo. Oppure, aspetto più importante, di testare la compatibilità con diverse versioni di Android, poiché alcune librerie o funzionalità potrebbero non essere supportate al meglio.

#### **Espresso:**

questo tool facilita la creazione di test per il front-end in modo completo, in modo da creare flussi di azioni automatizzati, ognuno dedicato per ogni funzione che offre l'applicazione.

### **4.3 Test di performance**

I Test di performance permettono di verificare come si comporta l'applicazione in termini di utilizzo delle risorse del dispositivo su cui è in esecuzione l'applicazione.

In questa sezione parleremo di come abbiamo utilizzato due software appositi per questo tipo di valutazioni.

Grazie a questi, siamo riusciti a verificare che i requisiti non funzionali di efficienza, definiti durante il progetto, vengano rispettati .

#### **4.3.1 Perfetto**

Perfetto è uno strumento che permette di monitorare le prestazioni di tutte le applicazioni che sono in esecuzione all'interno del dispositivo. Questo è utile soprattutto per avere un confronto dell'utilizzo delle risorse di GreenWay rispetto agli altri processi.

Dai risultati che abbiamo ricevuto, siamo stati in grado di vedere le azioni che abbiamo compiuto (nella parte inferiore dell'immagine [4.1]) e l'utilizzo della memoria dell'applicazione in quel momento (nella parte superiore dell'immagine [4.1]).

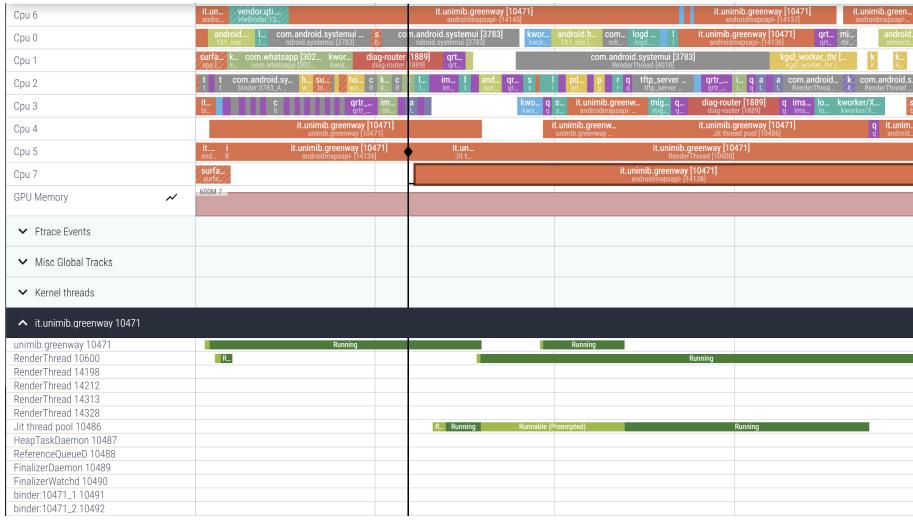


Figura 4.1: Interfaccia trace Perfetto

### 4.3.2 Android Profiler

Android Profiler invece, è un tool simile a Perfetto, ma integrato su Android Studio.

La differenza sostanziale è che si concentra solo sull’analisi e il benchmark dell’applicazione e non su tutti i processi in esecuzione sul dispositivo.

È caratterizzato da tre tipi di parametri che analizza, chiamati **profiler**[4.2][And24b]:

- **CPU profiler:** Consente di tenere traccia dei problemi di prestazione runtime.
- **Memory profiler:** Consente di tenere traccia delle allocazioni di memoria.
- **Energy profiler:** Consente di monitorare il consumo energetico.



Figura 4.2: Interfaccia Android Profiler

## 4.4 Test di qualità del software

I Test di qualità del software, consistono nel verificare la leggibilità del codice, l'assenza di bug, la conformità agli standard di sicurezza e verificare se ci sono implementazioni errate sostituendole con alcune più efficienti. Per condurre questo tipo di valutazione abbiamo usato un software noto come *SonarQube*.

### 4.4.1 SonarQube

SonarQube è un software open source per l'ispezione della qualità del codice, offrendo una vasta gamma di metriche e strumenti per verificare problemi a riguardo.

Ci ha fornito dei warning riguardanti soprattutto duplicazioni di variabili, complessità ciclomatiche e potenziali bug.

Nello specifico, per la qualità del software, ci sono tre categorie principali in cui vengono raggruppati gli error:

- **Security:** Questo parametro valuta la presenza di vulnerabilità nel codice. L'obiettivo è quello di avere il software protetto da possibili attacchi e minacce.
- **Reliability:** Questo parametro si riferisce alla presenza di bug che potrebbero causare malfunzionamenti al software. L'obiettivo fondamentale è appunto, prevenire problemi che potrebbero compromettere l'esperienza dell'utente.
- **Maintainability:** Questo parametro misura quanto sia facile per gli sviluppatori, capire, modificare e migliorare il codice. Viene analizzata la lunghezza dei metodi, la duplicazione e altri fattori che influenzano la manutenibilità.

The screenshot shows the SonarQube web interface. At the top, there's a navigation bar with tabs for Overview, Issues (which is selected), Security Hotspots, Measures, Code, and Activity. On the right side of the header, there are Project Settings and Project Information dropdowns. Below the header, the main content area has a sidebar on the left containing filters for Software Quality (Security: 1, Reliability: 10, Maintainability: 3), Severity, Type, Scope, Status, and Security Category. The main panel displays three issues from the file build.gradle.kts:

- Group dependencies by their destination.** (Maintainability) - Open, Not assigned. L63 - 5min effort - 1 month ago - Code Smell - Major
- Do not hardcode version numbers.** (Maintainability) - Open, Not assigned. L72 - 5min effort - 2 months ago - Code Smell - Major
- Do not hardcode version numbers.** (Maintainability) - Open, Not assigned. L75 - 5min effort - 1 month ago - Code Smell - Major

At the top of the main panel, there are buttons for Bulk Change, Select issues, Navigate to issue, and a summary of 3 issues and 1h effort. There are also buttons for Consistency and gradle.

Figura 4.3: Interfaccia SonarQube

# Capitolo 5

## Utilizzo di ChatGPT

In questo capitolo, verrà illustrato in modo dettagliato la motivazione principale di questa relazione, ovvero l'utilità che abbiamo riscontrato, di ChatGPT, durante la creazione dell'applicazione.  
Il capitolo è diviso per ogni fase dello sviluppo (*Analisi dei requisiti, implementazione e testing*) in quanto, ci sono pareri diversi per ognuna.

Nei paragrafi seguenti, verrà esplorato ciascuna di queste fasi in dettaglio. Questo permetterà di comprendere l'impatto effettivo di ChatGPT sul nostro progetto e di valutare i possibili benefici complessivi derivanti dal suo utilizzo.

### 5.1 Analisi dei requisiti

Durante la fase di analisi dei requisiti, abbiamo utilizzato gli LLM per avere una spiegazione sulla forma di scrittura del documento.

Quando è stato inviato il prompt, ChatGPT ci ha generato dei requisiti con le relative categorie, quest'ultime le abbiamo tenute interamente per avere un'idea su come suddividere i relativi requisiti.

Abbiamo fatto un confronto con progetti passati svolti all'interno di altri corsi, e abbiamo notato che molte di queste categorie coincidevano.

Per quanto riguarda gli elementi all'interno di ogni sezione, come i requisiti funzionali e alcune sotto categorie dei requisiti non funzionali, la maggior parte sono stati cambiati tenendo quelli ideati da noi.

In seguito, abbiamo eseguito un prompt per richiedere alcune idee da aggiungere all'applicazione e tra tutte quelle fornite, quella che si adattava di più alla nostra idea di come volevamo implementare l'applicazione, è stata il sistema di amicizie.

### 5.2 Descrizione dell'implementazione

Nel periodo di implementazione invece, avendo già acquisito le competenze nel corso di *Programmazione di dispositivi mobili*, ho usufruito maggiormente di ChatGPT per la correzione di errori all'interno del codice e per avere idee su

come migliorare l'efficienza di alcuni metodi delle classi.

Per l'architettura dell'applicazione invece, personalmente non ho eseguito dei prompt su come rispettarla visto che avevo già idea di come comportarmi e seguire in modo corretto le regole che impone **Android**.

Un altro prompt che ho inserito, è stato quello per la generazione delle regole di sicurezza per l'accesso al Realtime Firebase.

Non essendo a conoscenza della sintassi, come prompt ho fornito l'idea di come volevo che funzionasse ovvero, solo gli utenti autenticati potevano eseguire azioni di scrittura e lettura su determinati tipi di dati all'interno del database. Una volta generata la regola, l'ho inserita all'interno della piattaforma. Successivamente ho eseguito dei test per vedere se effettivamente funzionassero notando, in seguito, la corretta implementazione che mi ha fornito ChatGPT.

Infine, un altro aiuto che mi ha fornito è stato quello di trovare dei valori medi di consumo di CO<sub>2</sub>, da inserire poi all'interno dell'app per il calcolo di quest'ultimo in base al tipo di carburante che imposta l'utente.

ChatGPT ci ha fornito i seguenti valori:

1. **Benzina:** 150g di CO<sub>2</sub>.
2. **Diesel:** 120g di CO<sub>2</sub>.
3. **GPL:** 100g di CO<sub>2</sub>.
4. **Metano:** 80g di CO<sub>2</sub>.

Dopo aver letto i seguenti valori, li abbiamo confrontati con altre fonti presenti su internet e, una volta visto che quelli forniti da ChatGPT rientravano nel range di quelli trovati altrove, li abbiamo inseriti all'interno di GreenWay.

### 5.3 Progetto di testing

Per il testing dell'applicazione, gli LLM mi sono stati di notevole aiuto.

Non avendo competenze specifiche nella scrittura di test utilizzando ad esempio, la libreria Junit, mi è venuto in aiuto ChatGPT per la scrittura di quest'ulti-mi. Fornendogli le classi dell'applicazione su cui volevamo eseguire i test, come quelle relative ai singoli strati di comunicazione, ho potuto chiedergli di scrivermi degli esempi di test che andassero a verificare il corretto funzionamento dei metodi. Conoscendo il funzionamento specifico di ogni metodo, con altri prompt sono riuscito ad arricchire questi test andando ad aggiungere anche dei casi limite ad ognuno.

Quindi per questa fase abbiamo utilizzato gli LLM per andare a scrivere quei test che abbiamo citato nel capitolo precedente che rientrano all'interno dei test di integrazione e alcuni di interfaccia utente.

Per quanto riguarda i software utilizzati, come monkey, firebase testLab, espresso oppure sonarqube, non c'è stato bisogno del supporto di ChatGPT essendo questi, molto intuitivi e facili nell'utilizzo.

## 5.4 Statistiche sull'utilizzo di ChatGPT

Durante lo sviluppo dell'applicazione, ho utilizzato ChatGPT nelle seguenti percentuali per ciascuna fase del progetto:

- **Analisi dei requisiti:** 30% con grado di soddisfazione del 60%
- **Implementazione:** 40% con grado di soddisfazione del 50%
- **Testing:** 70% con grado di soddisfazione del 80%

Il contributo di ChatGPT si è rivelato particolarmente utile durante la fase di implementazione e in quella di testing, dove ha assistito nella correzione degli errori e nell'ottimizzazione del codice. Anche nella fase di analisi dei requisiti, ChatGPT ha giocato un ruolo significativo, aiutando a definire le categorie di requisiti e suggerendo nuove funzionalità.

# Capitolo 6

## Conclusioni

In conclusione, abbiamo sviluppato GreenWay per capire l'utilità effettiva degli LLM nel campo dello sviluppo delle applicazioni.

GreenWay può essere d'aiuto a molte persone andando a fornire informazioni utili sulla CO<sub>2</sub> che viene prodotta durante gli spostamenti che una persona compie. Tutto questo viene fatto attraverso l'insieme di diversi servizi che offrono Google e altre piattaforme.

Alla fine di questo processo di sviluppo ho capito che, gli LLM, in questo caso particolare ChatGPT, sono uno strumento di notevole importanza e che sicuramente, dopo la loro uscita, hanno cambiato il modo di pensare e di agire. Inoltre, attraverso delle ricerche e molti tentativi, ho migliorato la mia capacità di scrivere i prompt per ottenere sempre più spesso risultati che facevano al caso mio, senza correggere ChatGPT numerose volte. Questo è dovuto all'aggiunta di informazioni sempre più specifiche e alla capacità di dargli dettagli sul contesto in cui mi trovavo per fargli capire il più possibile cosa volessi.

In particolare sono giunto alla conclusione che, strumenti di questo tipo, possono essere molto di aiuto ma non sono ancora in grado di sostituire completamente la figura di chi lo usa, in questo caso, lo sviluppatore. Per quanto possano essere utili, una persona che non possiede competenze specifiche per la mansione che deve svolgere, non sarà ancora in grado di compierla con il solo utilizzo di intelligenze artificiali come ChatGPT.

Questo è dovuto al fatto che, molte volte purtroppo, vengono fornite risposte errate o comunque, non completamente giuste. Ciò comporta la necessità, da parte di chi la sta utilizzando, di saper analizzare gli output ottenuti e saper valutare i risultati.

Quindi, questo tipo di tecnologia, pur essendo estremamente all'avanguardia, in questo periodo non è ancora completamente pronta.

Sono dell'idea che questo sarà il futuro ma, bisognerà regolamentarne molto bene l'utilizzo attraverso delle linee guida da rispettare altrimenti, in futuro, potrà essere pericoloso. Tutta questa realtà è in continuo sviluppo ed evoluzione e sono sicuro che nel corso degli anni sarà in grado di sostituire completamente molti lavori ma, allo stesso tempo, ne creerà di nuovi.

# Bibliografia

- [Som10] Ian Sommerville. “Software Engineering, 9”‘ed”. In: *England: Education Limited* (2010), pp. 82–85.
- [Val14] Gaetano Valenti. “Strumenti per il Testing di Applicazioni Android”. In: (2013-2014), p. 22.

# Sitografia

- [Anc19] Ancler. *CO<sub>2</sub>, il principale gas serra.* Accessed: 2024-03-28. 2019. URL: [https://ancler.org/co2/#:~:text=L'anidride%20carb%20oni%20ca%20\(CO2\),impedendo gli%20di%20ritornare%20nel%20tempo%20sp%20azio..](https://ancler.org/co2/#:~:text=L'anidride%20carb%20oni%20ca%20(CO2),impedendo gli%20di%20ritornare%20nel%20tempo%20sp%20azio..)
- [And23] Android. *UI/Application Exerciser Monkey.* Accessed: 2024-05-17. 2023. URL: <https://developer.android.com/studio/test/other-testing-tools/monkey?hl=it>.
- [App23] AppMaster. *Analisi dei requisiti software.* Accessed: 2024-04-13. 2023. URL: <https://appmaster.io/it/blog/analisi-dei-requisiti-del-software>.
- [Eur23] Parlamento Europeo. *Emissioni di CO<sub>2</sub> delle auto: i numeri e i dati. Infografica.* Accessed: 2024-03-28. 2023. URL: <https://www.europarl.europa.eu/topics/it/article/20190313ST031218/emissioni-di-co2-delle-auto-i-numeri-e-i-dati-infografica>.
- [Goo23a] Google. *API Routes.* Accessed: 2024-05-15. 2023. URL: <https://developers.google.com/maps/documentation/routes?hl=it>.
- [Goo23b] Google. *Maps SDK for Android.* Accessed: 2024-04-10. 2023. URL: <https://developers.google.com/maps/documentation/android-sdk?hl=it>.
- [Goo23c] Google. *Places SDK for Android.* Accessed: 2024-04-15. 2023. URL: [https://developers.google.com/maps/documentation/places%20/android-sdk?hl=it](https://developers.google.com/maps/documentation/places/android-sdk?hl=it).
- [And24a] Android. *Espresso.* Accessed: 2024-05-17. 2024. URL: <https://developer.android.com/training/testing/espresso?hl=it>.
- [And24b] Android. *Profile your app performance.* Accessed: 2024-05-20. 2024. URL: <https://developer.android.com/studio/profile?hl=it>.
- [Cou24] Coursera. *How write ChatGPT prompt.* Accessed: 2024-04-3. 2024. URL: <https://www.coursera.org/articles/how-to-write-chatgpt-prompts>.
- [Des24] Material Design. *Color roles.* Accessed: 2024-03-28. 2024. URL: <https://m3.material.io/styles/color/roles>.
- [Dev24] Android Developer. *Guida all'architettura delle app.* Accessed: 2024-02-15. 2024. URL: <https://developer.android.com/topic/architecture?hl=it>.

- [Fig24] Figma. *Figma*. Accessed: 2024-03-30. 2024. URL: <https://www.figma.com/>.
- [Goo24a] Google. *API Air Quality*. Accessed: 2024-04-10. 2024. URL: <https://developers.google.com/maps/documentation/air-quality?hl=it>.
- [Goo24b] Google. *Firebase Authentication*. Accessed: 2024-04-20. 2024. URL: <https://firebase.google.com/docs/auth?hl=it>.
- [Goo24c] Google. *Firebase Realtime Database*. Accessed: 2024-04-20. 2024. URL: <https://firebase.google.com/docs/database?hl=it>.
- [Goo24d] Google. *Firebase Testlab*. Accessed: 2024-05-15. 2024. URL: <https://firebase.google.com/docs/test-lab?hl=it>.
- [Goo24e] Google. *Materian Design*. Accessed: 2024-03-30. 2024. URL: <https://m3.material.io/>.
- [Rob24] Robolectric. *test-drive your Android code*. Accessed: 2024-05-17. 2024. URL: <https://robolectric.org/#run-tests-outside-of-the-emulator>.
- [Son24] SonarQube. *SonarQube 10.5 Documentation*. Accessed: 2024-05-20. 2024. URL: [https://docs.sonarsource.com/sonarqube/latest/?\\_gl=1\\*2h78t\\*\\_up\\*MQ..&gclid=CjwKCAjwmYCzBhA6EiwAxFwfgOo2%20jh3VVsgoBOFfPvc7TpN6WItu8scTODzcq%20ilofXoi-sRikvdyVBo%20CspoQA%20vD\\_BwE](https://docs.sonarsource.com/sonarqube/latest/?_gl=1*2h78t*_up*MQ..&gclid=CjwKCAjwmYCzBhA6EiwAxFwfgOo2%20jh3VVsgoBOFfPvc7TpN6WItu8scTODzcq%20ilofXoi-sRikvdyVBo%20CspoQA%20vD_BwE).