

UNIVERSITÀ DEGLI STUDI DI PERUGIA
Dipartimento di Matematica e Informatica

CORSO DI LAUREA TRIENNALE IN INFORMATICA



Tesi di Laurea

Progettazione e sviluppo dell'applicazione CGIL

Laureando:

Dominici Alex

Relatore:

Prof. Osvaldo Gervasi

Dr. Damiano Perri

Anno Accademico 2018–2019

Ai miei genitori

Indice

Introduzione	vi
1 Descrizione del progetto e delle tecnologie utilizzate	1
1.1 Scopo del progetto	1
1.2 Idea base della CGIL	2
1.3 Finalità del progetto	3
1.3.1 Popolamento dell'applicazione	4
1.4 Tecnologie utilizzate	5
1.4.1 Android	5
1.4.2 Java	10
1.4.3 XAMPP	10
1.4.4 Apache	11
1.4.5 MySQL	11
1.4.6 SQL	12
1.4.7 PHP	12
1.4.8 phpMyAdmin	14
2 Analisi dei requisiti e progettazione	15
2.1 Specifica dei requisiti	15

2.1.1	Requisiti funzionali	15
2.1.2	Requisiti di vincolo	16
2.1.3	Requisiti di interfaccia	17
2.1.4	Requisiti software	18
2.2	Progettazione e sviluppo dell'applicazione	18
2.2.1	Geolocalizzazione del dispositivo	19
2.2.2	La libreria Jsoup	24
2.2.3	Google Places API	34
3	Descrizione e Funzionamento applicazione	48
3.1	AsyncTask	48
3.2	RecyclerView	55
3.3	Spinner	60
3.4	Descrizione Funzionale	62
Conclusioni		74
Bibliografia		76

Elenco delle figure

1.1	Mobile Operating System Market Share Worldwide	8
1.2	Tabella comparativa vendita smartphone prodotta da IDC .	8
1.3	Tabella e grafico riassuntivi delle distribuzioni Android. . .	9
1.4	Scambio di messaggi client/server/DBMS per la generazione di una pagina web dinamica	13
2.1	Logo e nome applicazione	17
2.2	Esempio di struttura ad albero HTML	28
2.3	Risultato della ricerca	39
2.4	Tabella prezzi Places - Text Search.	46
2.5	Tabella prezzi Dati di base.	46
2.6	Tabella prezzi Dati di Contatto	47
2.7	Tabella prezzi Dati di Atmosfera	47
2.8	Tabella prezzi Place Photos	47
3.1	Funzionamento Asynctask	49
3.2	Esempio spinner	60
3.3	Homepage applicazione.	63
3.4	Cambio posizione nella Homepage.	64
3.5	Diagramma caso d'uso Homepage	65

3.6	Seconda pagina applicazione	66
3.7	Diagramma caso d'uso seconda pagina	67
3.8	Eccezione seconda pagina	68
3.9	Terza pagina applicazione.	69
3.10	Segnalazione azienda.	70
3.11	Diagramma caso d'uso terza pagina	71
3.12	Eccezione terza pagina	72

Introduzione

La presente Tesi ha avuto come oggetto la realizzazione di una App Android avente lo scopo di mostrare agli utenti quali attività commerciali sono certificate dal Sindacato dei Lavoratori CGIL, sezione Regionale dell’Umbria, in merito all’applicazione di contratti di lavoro secondo le normative vigenti. L’App certifica pertanto quelle attività commerciali che si distinguono per il rispetto del diritto dei lavoratori, da cui il nome scelto, *rightsapp*.

Il lavoro, abbastanza corposo per la sua estensione e complessità, è stato portato avanti in collaborazione con Matteo Baldassarrini.

Il progetto è stato suddiviso in due parti, cosa che ha permesso uno sviluppo in parallelo: la prima, trattata da me, è stata dedicata allo sviluppo dell’applicazione in ambiente Android, mentre la seconda, sviluppata da Matteo Baldassarrini, ha visto la creazione del Database e dell’ambiente di gestione del backoffice.

Questo documento descrive tutte le motivazioni e le scelte intraprese per portare a termine l’obiettivo prefissato, ovvero quella dello sviluppo dell’applicazione. Per ottenere questo sono state svolte diverse attività, che vanno dall’analisi dei requisiti per definire le richieste del cliente, allo studio di nuove API per la ricerca delle informazioni, fino alla progettazione software, tutte finalizzate allo sviluppo dell’applicazione stessa.

Capitolo 1

Descrizione del progetto e delle tecnologie utilizzate

In questo capitolo si vogliono dare alcune informazioni generali riguardo allo scopo del progetto, l’idea di base che ha mosso la CGIL dell’Umbria a cercare la collaborazione con il Dipartimento di Matematica e Informatica per definire un’app che consentisse il raggiungimento dell’obiettivo, ed infine l’analisi che ha portato allo sviluppo dell’applicazione sul sistema operativo Android ed a scegliere le tecnologie che sono state utilizzate.

1.1 Scopo del progetto

Lo scopo del progetto è stato lo sviluppo di un’applicazione su ambiente Android nell’ambito della convenzione tra la CGIL regionale dell’Umbria e il Dipartimento di Matematica e Informatica dell’Università di Perugia che ha come scopo quello di instaurare un rapporto non episodico di collaborazione fra le parti, nel quale le attività dell’Università e le attività del Sindacato

Descrizione del progetto e delle tecnologie utilizzate

possano integrarsi e coordinarsi reciprocamente. In quest'ottica l'Università e il Sindacato hanno condiviso l'obiettivo di collaborare per sviluppare progettualità strategiche nei territori di riferimento, attivando una rete di relazioni virtuose che condivida risorse, conoscenze e competenze, volte a valorizzare scientificamente e a promuovere la conoscenza, oltre che a consentire la ricerca scientifica e la sperimentazione in campo, rispetto al tema dello sviluppo della tecnologia legato alla creazione di nuova occupazione e al miglioramento delle condizioni di lavoro esistenti.

1.2 Idea base della CGIL

Proprio su questi temi nasce il proposito di realizzare una applicazione con l'obiettivo focalizzato sulla qualità del lavoro in ambito turistico, che si affianchi alle già numerose applicazioni che valutano la qualità delle materie prime o dei servizi, in modo da far sì che anche questo fattore, la qualità del lavoro, diventi un elemento discriminante nella scelta del consumatore. Dunque un'applicazione Android in grado di mostrare all'utente le attività commerciali, strutture ricettive e di ristorazione presenti nelle zone limitrofe e non, mettendo in evidenza quelle che promuovono la qualità del lavoro.

L'idea è nata all'interno di un corso di formazione sindacale rivolto ai giovani delegati e organizzato della CGIL regionale dell'Umbria con la collaborazione del dipartimento nazionale della formazione della CGIL nazionale. Tale idea parte dalla considerazione del ruolo sempre più importante che svolgono i consumi e i consumatori nella società attuale, cercando di mettere a loro disposizione uno strumento che faciliti un consumo critico, responsabile e consapevole, da parte di quello che noi definiamo il cittadi-

Descrizione del progetto e delle tecnologie utilizzate

no/lavoratore, che agirà non più solo in base a fattori come prezzo offerto o qualità del servizio, ma anche al legame di solidarietà con i lavoratori che si trovano in quel momento a prestare la propria attività lavorativa per l’azienda interessata, con la consapevolezza che una maggiore qualità del lavoro produca effetti migliorativi anche sui servizi offerti.

1.3 Finalità del progetto

I lavoratori, che al di fuori della propria prestazione svolgono anche il ruolo di consumatori, se adeguatamente informati su aspetti che riguardano i propri colleghi, che operano nelle varie aziende, possono diventare partner attivi nel processo che deve portare verso un consumo consapevole che determinerà l’indirizzamento verso prodotti e/o servizi forniti dalle aziende che fanno del rispetto dei diritti dei lavoratori stesso e della buona pratica sindacale una metodologia di produzione.

La speranza è che così facendo, le aziende, per poter restare e continuare ad affermarsi nel mercato siano portate a mantenere uno standard dei diritti dei lavoratori che consenta loro di essere “scelte” dal potenziale utente dell’applicazione.

La novità che ha stimolato la CGIL dell’Umbria ad investire sull’idea è quella di utilizzare la legge del mercato della domanda, a favore di coloro che solitamente sono quelli maggiormente colpiti, ovvero i lavoratori, facendo risaltare quelle realtà settoriali che fanno della qualificazione delle risorse umane il loro carattere distintivo, cercando di produrre così nel consumatore una scelta eticamente più giusta, andando ad interrompere quel circolo vizioso che purtroppo negli ultimi decenni si sta sempre maggiormente dif-

Descrizione del progetto e delle tecnologie utilizzate

fondendo, dove la concorrenza si realizza soltanto ribassando il costo del lavoro ed erodendo i diritti dei lavoratori.

L'utilizzo di questo strumento ha come scopo anche quello di monitorare il fenomeno dell'abusivismo e della sleale concorrenza, nonché del fenomeno sempre più dilagante del dumping contrattuale oltre quello di contrastare le forme di lavoro non regolare attraverso la possibilità, tramite l'applicazione, di fare segnalazioni anonime per denunciare i comportamenti sleali.

1.3.1 Popolamento dell'applicazione

Il popolamento del database mantenuto dalla CGIL Regionale Umbra, prevede che le aziende si registrino al sito secondo le modalità di seguito elencate:

1. Le aziende richiedono l'iscrizione all'applicazione attraverso l'autocertificazione di alcuni requisiti fondamentali.
2. La richiesta viene valutata dalla CGIL territoriale e dalle categorie sindacali di riferimento dell'azienda, per verificare eventuali difformità tra quanto dichiarato e la realtà dei fatti.

Superata positivamente la valutazione della categoria, l'azienda è inserita all'interno del Database dell'applicazione e diventa visibile a coloro che utilizzano l'Applicazione *rightsapp*.

Il monitoraggio delle caratteristiche richieste alle aziende continua anche dopo l'inserimento della stessa nel Database dell'applicazione utilizzando lo strumento, contenuto nella stessa, relativo alla possibilità di effettuare segnalazioni anonime. Queste non saranno visibili on-line ma inviate direttamente agli amministratori dell'applicazione che, con l'ausilio

Descrizione del progetto e delle tecnologie utilizzate

della categoria sindacale di riferimento, ne verificheranno la veridicità andando a determinare, in caso di riscontro positivo, l'esclusione dell'azienda dall'applicazione.

1.4 Tecnologie utilizzate

1.4.1 Android

Android, il famoso sistema operativo per dispositivi mobili sviluppato da Google, risulta avere diversi aspetti e sfumature. In questa sezione verrà mostrata una piccola introduzione ad Android e ai fattori che hanno portato a sviluppare l'applicazione tramite questa piattaforma.

La storia di Android

Nel 2003 Andy Rubin, Rich Miner, Nick Sears, e Chris White, fondarono la Android Inc. con lo scopo di creare dispositivi più *intelligenti* di quelli dell'epoca. Inizialmente la società operò in segreto, rivelando solo di progettare software per dispositivi mobili.

Nel 2005 Google ha acquisito Android Inc., lasciando a capo del progetto il team di Rubin. Due anni dopo, nel 2007, Google stipula contratti con i più grandi produttori e con le più grandi aziende che lavorano nel settore della telefonia (Samsung, HTC e T-Mobile) e della produzione di microprocessori (Texas Instruments e Qualcomm), in vista del fatto che la società desiderava entrare nel mercato della telefonia mobile. Nel 2008 Google svela il suo progetto presentando HTC Dream, il primo smartphone Android.[1].

Descrizione del progetto e delle tecnologie utilizzate

Android, sotto licenza Apache, è rilasciato in formato open source portando ad una grande diffusione del sistema operativo. Nonostante Android sia nato per dispositivi mobili, potendo essere modificato e distribuito liberamente, la sua diffusione è stata estesa ai più svariati dispositivi, come ad esempio le Smart TV.

Nel 2017 Android diventa il sistema operativo per dispositivi mobili più diffuso al mondo e ad oggi detiene circa il 70% del mercato, grazie anche al sostegno fornito da Google e ai tanti sviluppatori che si mettono alla prova nel creare app; applicazioni che estendono le funzionalità di un dispositivo Android.

Per quale ragione si è scelto Android

Le ragioni che hanno portato alla scelta di questo sistema operativo sono frutto di una ricerca completa, nella quale sono stati toccati diversi aspetti per poi riportarli qui di seguito:

1. La vasta diffusione di Android a livello mondiale: dopo un'accurata ricerca è stato rilevato che il market share controllato da Android è ora al 76,2%, in crescita rispetto al 71,3% di fine 2018. Mentre Apple iOS con il 22,4%, si trova in declino rispetto al 25,6% di fine 2018 [2] come mostrato in Figura 1.1.
2. La licenza Apache è la seconda ragione per cui è stato scelto Android. È una licenza open-source e si trovano con una certa facilità delle API Android e eccellenti tutorial che introducono facilmente e velocemente alla programmazione nel mondo Android.

Descrizione del progetto e delle tecnologie utilizzate

3. Questa tipologia di programmazione risulta essere gratuita, non obbligando l'utente all'acquisto di un kit di sviluppo né di una licenza; inoltre la distribuzione sul Play Store è meno restrittiva e più rapida rispetto alla concorrenza.
4. In aggiunta Android ha permesso, dato il costo relativamente basso rispetto ai concorrenti, l'accesso agli smartphone per un ampio numero di utenti e allo stesso tempo ha indotto chi lavora nell'ambiente della programmazione di applicazioni a doversi confrontare con esso, come mostrato in Figura 1.2.
5. Le applicazioni a livello utente sono scritte principalmente in Java, nonostante l'intero sistema operativo sia basato su Linux.
6. È possibile scrivere applicazioni utilizzando Android Studio, un eccellente strumento di sviluppo libero e molto ben documentato.
7. Infine l'affidabilità e la consistenza di Google rende Android un sistema resistente, in continuo sviluppo e che difficilmente smetterà di essere usato in poco tempo.

Versione Android

L'applicazione è stata sviluppata con la versione 9.0 Pie con API 28 ed è stata scelta come versione minima la 5.0 Lollipop con API 21. Le motivazioni che hanno spinto a scegliere queste versioni risultano essere puramente grafiche e funzionali, nonchè nell'avere la possibilità di installare l'app su un parco di dispositivi molto ampio. Inoltre Android 9 Pie è passato dallo 0,1% dell'intera base installata al 10,4% in poco più di sei mesi. Questa

Descrizione del progetto e delle tecnologie utilizzate

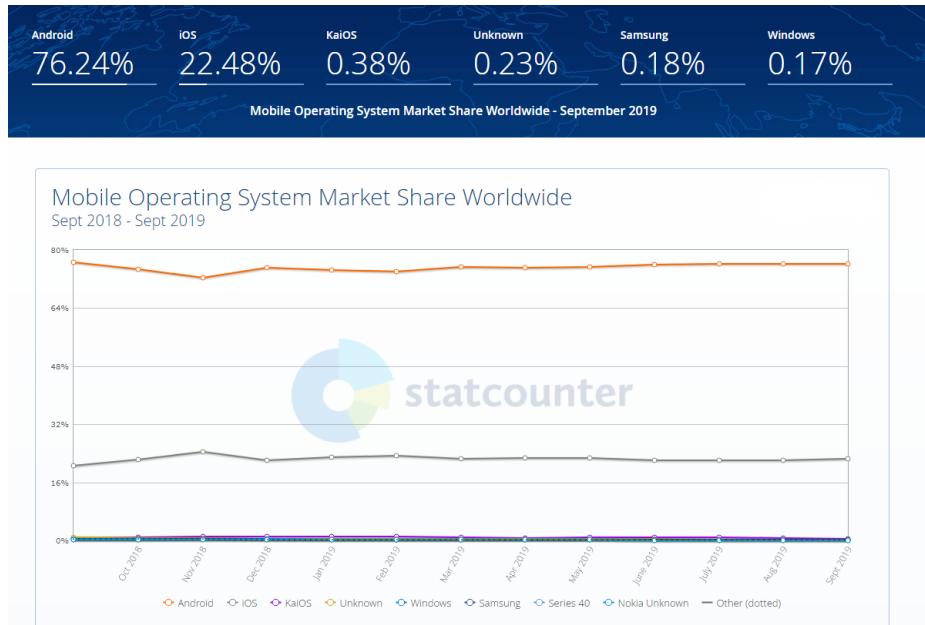
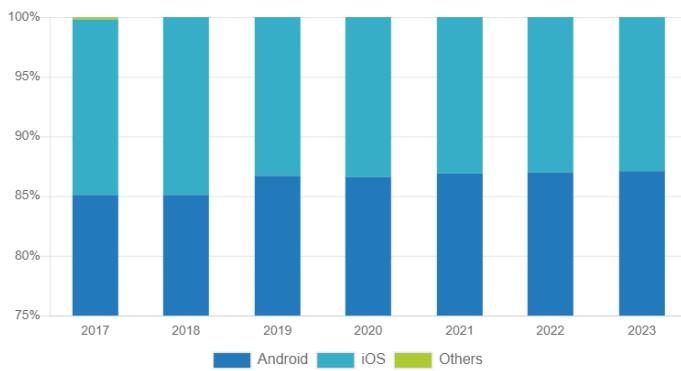


Figura 1.1: Mobile Operating System Market Share Worldwide

Worldwide Smartphone Shipment OS Market Share Forecast



Year	2017	2018	2019	2020	2021	2022	2023
Android	85.1%	85.1%	86.7%	86.6%	86.9%	87.0%	87.1%
iOS	14.7%	14.9%	13.3%	13.4%	13.1%	13.0%	12.9%
Others	0.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
TOTAL	100.0%						

Figura 1.2: Tabella comparativa vendita smartphone prodotta da IDC

Descrizione del progetto e delle tecnologie utilizzate

è una percentuale che non può essere presa come maggioritaria , dato che le due distribuzioni di Oreo realizzano complessivamente un 28,3%, seguite dalle due di Nougat che riscuotono un 19,2% in totale. C'è però da notare che il tasso di adozione è stato tuttavia maggiore rispetto alle due precedenti release che ci hanno messo più di dieci mesi per superare il 10%, mentre a Pie ne sono bastati solo 8 [3] come mostrato in Figura 1.3.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

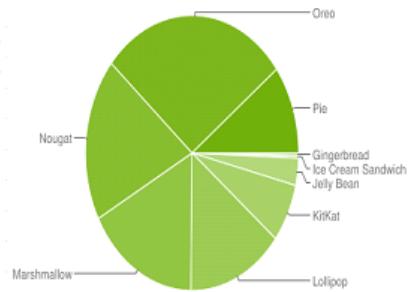


Figura 1.3: Tabella e grafico riassuntivi delle distribuzioni Android.

Android Studio

Si tratta di un ambiente di sviluppo integrato (IDE, Integrated Development Environment) totalmente open-source e ufficialmente supportato da Google mediante plugin appositamente sviluppati, che rende semplice lo studio e lo sviluppo di applicazioni Android.[4]

Android Studio permette di scrivere applicazioni tramite tre diversi linguaggi di programmazione: Java, Kotlin e C++. Nel nostro caso in partico-

Descrizione del progetto e delle tecnologie utilizzate

lare è stato utilizzato esclusivamente Java. Uno dei punti di forza di questo software è la capacità di generare diversi APK dell'applicazione con caratteristiche anche differenti descritte nel file `build.gradle` del progetto. Nello specifico Gradle è un software per l'automazione dello sviluppo basato sulle idee di Apache che include un DSL, Domain-Specific Language regolato su Groovy, al posto della comune modalità XML utilizzata per la dichiarazione della configurazione del progetto.

1.4.2 Java

Si tratta di un linguaggio di programmazione ad alto livello, tipizzato staticamente e orientato agli oggetti, il quale è progettato in modo specifico per non essere dipendente dalla sistema hardware di esecuzione.

1.4.3 XAMPP

Acronimo di Cross, Apache, MySQL, PHP e Perl, è un software che facilita l'installazione e la gestione degli strumenti più comuni per lo sviluppo di applicazioni web, raggruppandoli in un unico luogo. Infatti, attraverso un'unica installazione è possibile avere in una sola cartella Apache, cioè il web server che gestisce le richieste che arrivano da un qualsiasi client attraverso il protocollo HTTP, MySQL cioè il DBMS, PHP e Perl, cioè i due linguaggi utili per lo sviluppo di applicazioni web.

Una volta installato, all'interno della cartella `xampp` si troverà un'applicazione che ha la funzione di pannello di controllo, ovvero permette di far partire e spegnere i vari server. Per esigenze di sviluppo sarà necessario avviare Apache e MySQL e per verificarne il corretto funzionamento è suf-

Descrizione del progetto e delle tecnologie utilizzate

ficiente aprire un qualsiasi browser internet e digitare `http://localhost` per il primo e `http://localhost/phpmyadmin` per il secondo.[5]

1.4.4 Apache

Si tratta di un software libero e open-source sviluppato dalla Apache Software Foundation a partire dal 1994; implementa un server Web secondo le caratteristiche del protocollo HTTP. Apache è quindi un software che ha l'incarico di accettare richieste HTTP da un host e ritrovare i dati necessari per fornirle al richiedente, dunque svolge funzioni di trasporto delle informazioni, di internetwork e di collegamento, ed ha il vantaggio di offrire funzioni di controllo per la sicurezza come quelle effettuate da un proxy.

Dati i numerosi vantaggi si è deciso di scegliere questo software: è open-source, portabile (OS Linux, Windows, ecc..), ha un'architettura modulare, è flessibile e affidabile.

1.4.5 MySQL

MySQL è un DBMS, cioè un DataBase Management System, sviluppato nel 1996 da una società di consulenza svedese, la TcX, che aveva bisogno di un database veloce e che richiedeva poche risorse, pur dovendo gestire notevoli quantità di dati.

Ad oggi è un software open source ed è sviluppato per essere il più possibile conforme agli standard ANSI SQL e ODBC SQL. I linguaggi di programmazione e i sistemi che supportano MySQL sono numerosi, tra questi abbiamo: ODBC, Java, PHP, Python e molti altri.

Descrizione del progetto e delle tecnologie utilizzate

1.4.6 SQL

SQL è un linguaggio di programmazione nato per gestire e interrogare database. Per fare ciò vengono usati particolari strumenti di programmazione detti **query**.

In principio è stato progettato come linguaggio di tipo dichiarativo, poi però si è evoluto con l'inserzione di procedure, istruzioni di controllo, definizione dei tipi di dati dall'utente e estensioni differenti del linguaggio.

I comandi SQL usati sono divisi in:

- DDL (Data Definition Language): consente di creare e cancellare basi di dati o di modificarne la loro struttura.
- DML (Data Manipulation Language): consente di inserire, cancellare, leggere e modificare i dati.
- DCL (Data Control Language): consente di gestire i permessi e gli utenti.
- DMCL (Device Media Control Language): consente di controllare i supporti dove vengono registrati i dati.

Nello sviluppo di pagine web, per esempio, si utilizza spesso un DBMS, ovvero un software o un'interfaccia grafica progettata per permettere la generazione, la gestione e l'interrogazione efficiente del database.

1.4.7 PHP

PHP (acronimo di “PHP: Hypertext Preprocessor”, originariamente acronimo di “Personal Home Page”) è un linguaggio di scripting che risiede in

Descrizione del progetto e delle tecnologie utilizzate

un server in remoto e che in fase di esecuzione interpreta le informazioni ricevute da un client grazie al Web server, le elabora e restituisce un risultato al client che ha formulato la richiesta.

Nato nel 1994, la versione presente è un' estensione di Rasmus Lerdorf insieme a Zeev Suraski e Andi Gutmans ed mette a disposizione molte possibilità tra cui l'interazione con diversi DBMS e con codici HTML, tuttavia viene principalmente utilizzato per sviluppare applicazioni web lato server. In Figura 1.4 viene mostrato il comportamento del PHP nell'interazione Client-Server tipica delle sessioni HTTP.

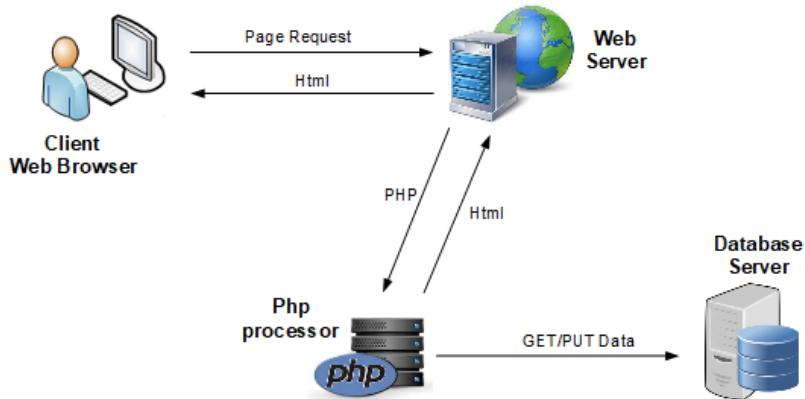


Figura 1.4: Scambio di messaggi client/server/DBMS per la generazione di una pagina web dinamica

Il client inoltra la richiesta di una risorsa al server il quale, mediante PHP, costruisce dinamicamente la pagina da restituire, seguendo le istruzioni presenti in un file con estensione .php. Il risultato dell'esecuzione viene restituito al client, che visualizzerà una pagina web statica nel proprio browser.

Descrizione del progetto e delle tecnologie utilizzate

1.4.8 phpMyAdmin

Si tratta di un'applicazione web scritta in PHP che permette di gestire un database MySQL attraverso un qualsiasi browser. L'applicazione è rivolta sia agli amministratori del database, sia agli utenti e amministra i permessi prendendoli dal DB stesso. PhpMyAdmin permette di generare un database da zero, creare le tabelle ed eseguire operazioni di miglioramento sulle stesse. Inoltre sono previste delle funzioni per l'inserimento dei dati, per le query, per il backup dei dati, ecc. L'amministratore infine ha anche un'interfaccia grafica per la gestione degli utenti: essa permette di aggiungere un nuovo utente, modificare la relativa password e gestire i permessi che l'utente ha sul database.

Capitolo 2

Analisi dei requisiti e progettazione

In questo capitolo verranno trattati e descritti i requisiti imposti e/o concordati dallo studente, dalla CGIL e dall'ambiente in cui si sarebbe definita e sviluppata l'applicazione.

2.1 Specifica dei requisiti

Nella presente sezione viene svolta l'analisi generale delle funzionalità e dei moduli necessari allo sviluppo. Tale analisi ha fatto emergere i requisiti obbligatori che l'applicazione dovrà soddisfare.

2.1.1 Requisiti funzionali

I requisiti funzionali si presentano come elenchi di funzionalità o servizi che il sistema deve fornire. Essi descrivono anche il comportamento del sistema

a fronte di particolari input e come esso dovrebbe reagire in determinate situazioni.

F1. Ricerca diverse tipologie di aziende.

F2. Ricerca aziende in prossimità.

F3. Ricerca aziende in luoghi diversi.

F4. Possibilità di visualizzare le aziende presenti nel DB.

F5. Possibilità di effettuare segnalazioni anonime.

F6. Avviso all'utente in caso di mancanza di connessione del dispositivo alla rete internet.

F7. Avviso all'utente in caso di mancanza di connessione del dispositivo alla rete internet quando si sta per eseguire un'operazione che ne richiede.

2.1.2 Requisiti di vincolo

V1. Nella ricerca, le aziende certificate vengono segnalate con una stella illuminata.

V2. La ricerca di default avviene in prossimità della propria posizione.

V3. Il sistema deve essere veloce e preciso , permettendo all'utente di ricevere le informazioni desiderate in base alla posizione e al tipo di richiesta.

V4. Il sistema deve permettere la definizione della posizione dell'utente mediante il GPS.

- V5.** L'applicazione deve sempre essere connessa alla rete per poter visualizzare i dati aggiornati.
- V6.** Il logo e il nome dell'applicazione vengono forniti dalla CGIL, mostrati in Figura 2.1.



Figura 2.1: Logo e nome applicazione

2.1.3 Requisiti di interfaccia

- I1.** I button di ricerca nella homepage devono contenere un'immagine coerente rispetto al tipo di ricerca che verrà effettuata.
- I2.** I colori utilizzati devono essere coerenti con quelli della CGIL.
- I3.** L'applicazione deve contenere al suo interno il simbolo della CGIL.
- I4.** Il sistema deve avere un'interfaccia semplice e intuitiva, in modo da facilitare anche gli utenti con meno esperienza con dispositivi mobili.
- I5.** Action bar per la navigazione tra sezioni.
- I6.** Barre di caricamento durante l'acquisizione di dati.
- I7.** Scheda azienda con le seguenti informazioni se presenti: nome azienda, immagine, valutazione, certificazione, indirizzo e orario apertura.

2.1.4 Requisiti software

S1. Android Studio. 1.4.1

S2. Java. 1.4.2

S3. Apache. 1.4.4

S4. MySQL. 1.4.5

S5. SQL. 1.4.6

S6. PHP. 1.4.7

S7. phpMyAdmin. 1.4.8

Infine si vuole esplicitare lo studio affrontato prima di poter iniziare lo sviluppo su piattaforma Android. Fondamentale è stato capire come reperire le informazioni delle diverse aziende per poterle riutilizzare. Il passo è risultato basilare per sviluppare un'applicazione di questo genere in cui i dati risiedono all'esterno dell'applicazione stessa e vengono caricati dinamicamente dietro input dell'utente.

Successivamente è stato molto importante capire come attingere ai dati presenti nel database, per poterli confrontare con i risultati ottenuti.

2.2 Progettazione e sviluppo dell'applicazione

In questa sezione viene mostrata la progettazione che ha portato all'evoluzione dell'applicazione, nonché al suo sviluppo. È stato condotto uno studio estensivo dei metodi di Android, al fine di poter sviluppare al meglio *rigtsapp*.

2.2.1 Geolocalizzazione del dispositivo

Il primo punto affrontato per lo sviluppo dell'applicazione è stato quello di riuscire a localizzare la posizione del device, in modo tale da poter effettuare la ricerca delle aziende nella zona circostante.

Android, fortunatamente, mette a disposizione diverse metodologie e servizi per ottenere tale posizione. Quello utilizzato nella nostra applicazione è stato `FusedLocationProviderClient` che è la principale Application Programming Interface (API) dei servizi di localizzazione di Google.[6]

Quale API scegliere?

L'API `LocationServices` di Google è quella effettivamente utilizzata per accedere alla posizione del dispositivo. Tuttavia per accedere a questi servizi la nostra applicazione deve connettersi ai Google Play Services, creando un client e specificando quali API saranno poi utilizzate.

In precedenza con `FusedLocationProviderApi` era nostra responsabilità avviare e gestire la connessione ai Google Play Services e solo dopo che la connessione era riuscita potevamo recuperare la posizione.

Nel caso avessimo provato a recuperare la posizione prima che la connessione fosse completa, si sarebbe ottenuto in risposta un errore del tipo: `IllegalStateException`. Dunque questo tipo di approccio risulta avere molti problemi come:

- La circostanza in cui la connessione ai Google Play Services non riuscisse o venisse annullata non viene gestita.
- Sarebbe difficile condividere la posizione tra più attività senza ripetere nuovamente la connessione.

- Per un'app che vuole solo recuperare la posizione, la gestione ogni volta della connessione risulta essere uno sforzo non necessario ed extra.

Mentre con `FusedLocationProviderClient` si hanno diversi vantaggi:

- L'utente non deve più inizializzare `GoogleApiClient`, cioè creare un client, né gestire la connessione.
- Restituisce il risultato come un oggetto Task che è facile da gestire e condividere.
- L'utente non deve attendere fino a quando non viene stabilita la connessione per richiedere una posizione. Quando si richiede la posizione corrente, la chiamata all'API attende automaticamente fino a quando non viene stabilita la connessione, riducendo così al minimo le possibilità di un errore di tipo `IllegalStateException`.

Recupero della posizione corrente

Vediamo adesso come recuperare la posizione corrente dell'utente utilizzando l'API `FusedLocationProviderClient`.

Il procedimento utilizzato si divide in tre parti distinte, nella prima aggiungiamo le autorizzazioni e le dipendenze che servono all'applicazione per poter ottenere la localizzazione, mentre nella seconda ci assicuriamo che l'utente abbia dato il consenso all'utilizzo della posizione; infine nella terza vengono implementate le funzioni necessarie al recupero della posizione.

1. Le autorizzazioni dell'app devono essere definite nel file *manifest*.

Quindi il primo passo sarebbe quello di aggiungere le autorizzazioni di posizione in `AndroidManifest.xml`:

```
"android.permission.ACCESS\_COARSE\_LOCATION"  
"android.permission.ACCESS\_FINE\_LOCATION"
```

Inoltre `FusedLocationProvider` fa parte dei Google Play Services, pertanto è necessario includere i servizi di Google Play come dipendenze nel *build.gradle*:

```
implementation 'com.google.android.gms:play-services:12.0.1'
```

2. A partire da *Android Marshmallow*, le autorizzazioni di posizione devono essere esplicitamente approvate a runtime dall'utente prima che l'applicazione inizi a localizzare la posizione del dispositivo.

Per tanto per ottenere ciò vengono utilizzate diverse funzioni:

- `CheckSelfPermission()`: viene utilizzata per verificare se si dispone di un'autorizzazione. Se l'app dispone dell'autorizzazione, il metodo ritorna `PERMISSION_GRANTED` e l'app può procedere con l'operazione, altrimenti il metodo ritorna `PERMISSION_DENIED` e l'app deve chiedere esplicitamente all'utente l'autorizzazione.
- `RequestPermissions`: fa apparire una finestra di dialogo standard con cui l'utente esprime la propria volontà, il risultato viene passato alla funzione `OnRequestPermissionsResult()`.
- `ShouldShowRequestPermissionsRationale()`: funzione che restituisce `true` se l'utente ha precedentemente negato la richiesta

e restituisce `false` se un utente ha negato un'autorizzazione e selezionato l'opzione “Non chiedere più” nella finestra di dialogo della richiesta di autorizzazione.

Viene fornita dunque una spiegazione o indicazione solo se l'utente ha già negato tale richiesta di autorizzazione.

- `OnRequestPermissionsResult()`: Funzione che controlla se il permesso è stato negato o concesso.

In caso sia stato negato viene visualizzato un Toast message che informa l'utente della propria scelta, mentre nel caso in cui sia stato concesso sarà possibile ottenere gli aggiornamenti sulla posizione.

3. Una volta verificato e ottenuto il permesso dell'utente vengono implementate le funzioni per l'ottenimento degli aggiornamenti sulla posizione:

- `getFusedLocationProviderClient()`: Permette di creare un'istanza della classe `FusedLocationProviderClient`.
- `BuildLocationRequest()`: Costruisce la richiesta della posizione da utilizzare per gli aggiornamenti.

Tale richiesta viene definita mediante metodi come: `setPriority()` che imposta la priorità della richiesta, `setInterval()` che imposta l'intervallo per gli aggiornamenti e `setSmallestDisplacement()` che imposta lo spostamento minimo per gli aggiornamenti.

Nel nostro caso abbiamo utilizzato una priorità di tipo: “*Priority_high_accuracy*”, con un intervallo di *5000ms* e uno sposta-

mento minimo di *10m*.

- **BuildLocationCallback()**: Costruisce il callback per gli aggiornamenti della posizione, al suo interno viene poi richiamato `onLocationResult()` quando le informazioni sulla posizione del dispositivo sono disponibili.
- **RequestLocationUpdates()**: Metodo che esegue la richiesta vera e propria degli aggiornamenti sulla posizione, passando come parametri la richiesta della posizione, il callback per gli aggiornamenti e un oggetto Looper. Questo oggetto mantiene una coda di messaggi che verrà utilizzata per implementare il meccanismo di callback.

E' importante sapere che la chiamata di questo metodo manterrà attiva la connessione dei servizi di Google play, quindi una volta ottenuto l'aggiornamento della posizione è necessario chiamare `removeLocationUpdates()`, in modo da evitare di perdere i vantaggi della gestione automatica della connessione.

In particolare, i metodi che contribuiscono all'ottenimento della posizione in Rightsapp sono stati raggruppati all'interno di un'unica funzione, la quale viene richiamata solo quando il permesso di posizione viene concesso dall'utente.

Inoltre il metodo `removeLocationUpdates()` è chiamato non appena la posizione del device viene catturata, in modo tale da non avere eccessivi sprechi di batteria dato che l'utilizzo prolungato del GPS ne comporta un elevato consumo.

2.2.2 La libreria Jsoup

Il secondo punto affrontato nello sviluppo dell'applicazione è stato capire come reperire le informazioni delle aziende per poi inserirle all'interno di essa.

Per fare ciò, il primo approccio è stato utilizzare la ricerca di Google, la quale ha permesso di restituire diverse informazioni con una procedura standard.

Essa consiste nel passare come chiave di ricerca la tipologia di aziende e il luogo in cui ricercarle (la posizione del device o un luogo scelto dall'utente).

Le pagine web contenenti tali informazioni sono state poi sottoposte ad un parsing dei dati mediante la libreria JSoup.[7]

JSoup, distribuito sotto licenza MIT, è un progetto open source sviluppato da Jonathan Hedley. È una libreria Java di parsing HTML/XML molto potente, le cui API permettono di recuperare dati e gestire in modo estremamente facile documenti usando le potenzialità di DOM (Document Object Model), CSS e metodi di accesso simili a quelli forniti da JQuery.

La libreria ad oggi è in grado di offrire tali servizi:

- Permette di caricare pagine HTML da un URL, un file o una stringa;
- Consente di trovare ed estrarre i dati utilizzando un attraversamento DOM o dei selettori CSS;
- Riesce a manipolare gli elementi HTML come attributi e testo;
- Consente di organizzare i dati in una white-list sicura, per prevenire attacchi XSS;
- Ricevere l'output con una struttura ben ordinata.

Jsoup è stato sviluppato per esaminare tutte le varianti di HTML presenti, poiché l'analizzatore genera per ogni pagina un albero di analisi.

Caricamento dei documenti da analizzare

Come già detto Jsoup consente di realizzare semplicemente il parsing di una pagina HTML fornita come :

1. Stringa.
2. Url.
3. File.

In Jsoup un documento XML è raffigurato dalla classe `Document`, i nodi del documento da istanze della classe `Node` e i tag da istanze della classe `Element`.

Volendo realizzare il parsing a partire da una *stringa*, viene utilizzato il seguente metodo statico mostrato nel codice 2.1.

```
1 String HTML = "<HTML><head><title>Primo parsing</title></head>
  +"<body><p>parsing html dentro un doc.</p></body></HTML>
  ";
2 Document doc = Jsoup.parse(HTML);
```

Listing 2.1: Esempio parsing partendo da una stringa.

Il metodo base `parse(String HTML)` viene utilizzato quando si conosce il codice HTML, mentre il metodo `parse(String HTML, String baseUri)` genera un nuovo `Document` con il codice HTML di entrata, dove l'argomento `baseUri` si usa per definire URL relativi in URL assoluti e deve essere impostato con l'URL in cui il documento è stato ritirato.

Quando abbiamo ottenuto il Document, possiamo recuperare i dati usando i metodi opportuni della classe `Document` e delle super classi `Element` e `Node`.

Se invece si vuole recuperare il documento HTML direttamente dall'*URL* si può utilizzare il metodo `connect` della classe Jsoup, come mostrato nel codice 2.2.

```
1 Document doc = Jsoup.connect('http://esempio.com/').get();
```

Listing 2.2: Esempio ottenimento HTML tramite URL.

Il metodo `connect(String url)` genera una nuova connessione e `get()` preleva ed esamina la pagina HTML. In caso si verificasse un errore durante il prelievo, viene spedita un'eccezione della classe `IOException` che lo gestisce.

L'interfaccia di collegamento è stata sviluppata per riuscire a concatenare e a comporre specifiche richieste, come mostrato nel codice 2.3.

```
1 Document doc = Jsoup.connect('http://esempio.com/')
2     .data('query', 'Java')
3     .userAgent('Mozilla')
4     .cookie('auth', 'token')
5     .timeout(5000)
6     .post();
```

Listing 2.3: Esempio concatenamento

Per sviluppare la prima versione dell'applicazione è stato proprio utilizzato il metodo `connect`, passandogli come URL la ricerca attraverso Google di una determinata azienda in un determinato luogo, come mostrato nel codice 2.4.

```
1 Document doc = Jsoup.connect("https://www.google.com/search?  
q=" + azienda +" "+ posizione).get();
```

Listing 2.4: Esempio ricerca.

In base al documento ottenuto, sono stati utilizzati metodi differenti per estrarre le informazioni necessarie.

L'ultima variante possibile per l'ottenimento del documento è la lettura del codice HTML presente in un *file* tramite il metodo statico della classe Jsoup, come mostrato nel codice 2.5.

```
1 File input = new File ("/tmp/input.HTML");  
2 Document doc = Jsoup.parse (input,"UTF-8","http://esempio.  
com/");
```

Listing 2.5: Esempio ottenimento HTML tramite File.

Il metodo `parse(file in, String charsetName, String baseUri)` carica ed esamina un file che contiene il codice HTML. Il verificarsi di un errore durante il caricamento del file, provocherà un `IOException`, il quale dovrebbe essere gestito in modo appropriato.

Estrazione e manipolazione dei dati

Per poter passare a descrivere i metodi della classe `Document` è necessario far presente alcuni concetti base.

L'oggetto `Document` include tutti gli elementi della pagina Web, collocati in un albero conservandone la struttura gerarchica originaria. L'argomento delle pagine Web consiste in un documento HTML il quale ha una struttura ad albero con gli elementi annidati e definiti da marcatori chiamati tag, come mostrato in Figura 2.2.

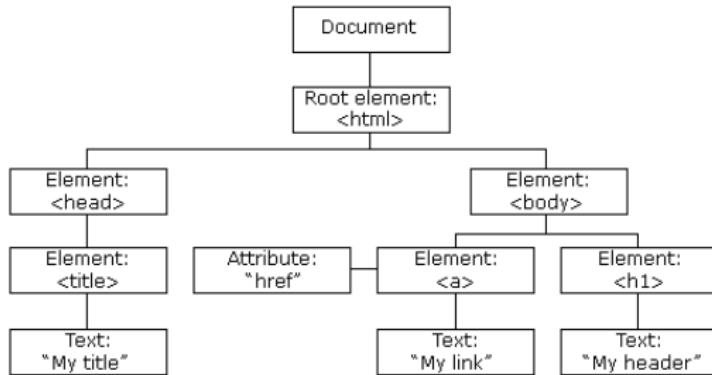


Figura 2.2: Esempio di struttura ad albero HTML

Il documento HTML racchiude un tag di apertura, diversi attributi di tale elemento con i rispettivi valori (i quali possono indicare stile, posizione, colore, etc), il contenuto esplicativo (che può essere del testo o una struttura di altri elementi) e infine un tag di chiusura.

I documenti scritti nel linguaggio HTML con una versione precedente alla 5 iniziano con una definizione del tipo di documento (DTD), la quale è necessaria al browser per identificare le regole di interpretazione e visualizzazione appropriate per lo specifico documento.

Mentre dalla versione 5 non è più necessaria alcuna DTD, pertanto in questi casi la dichiarazione iniziale indica la mera dicitura “HTML”.

Dopo la dichiarazione segue la struttura ad albero degli elementi che compongono la pagina, tale struttura è compresa tra i tag `<HTML>` e `</HTML>`, la quale è suddivisa in due parti principali:

- **Header**, è la sezione di intestazione che contiene informazioni di controllo normalmente non visualizzate dal browser, con l’eccezione di alcuni elementi.

- **Body**, sezione del corpo che contiene la parte informativa vera e propria, ossia il testo, le immagini e i collegamenti che costituiscono la parte visualizzata dal browser.

Dopo aver caricato il Document si può allocare l'oggetto Element, il quale risulta essere il punto centrale della libreria, dato che grazie ad esso è possibile rappresentare un elemento, ossia un tag, in particolare della pagina HTML con la corrispondente sottostruttura.

Quindi è possibile recuperare i dati partendo da esso ed attraversando il grafo dei nodi, come mostrato nel codice 2.6.

Document è sottoclasse di Element con la presenza di alcuni metodi specifici per la radice di una pagina HTML, tra cui uno per recuperare il titolo e due per ottenere l'Element di header o del body della pagina. Dunque esso non è altro che l'Element radice dell'albero completo.

```
1 File input = new File ("/tmp/input.HTML");
2 Document doc = Jsoup.parse (input, "UTF-8", "http://esempio.
    com/");
3 Element content = doc.getElementById("content");
4 Elements links = content.getElementsByTag ("a");
5 for( Element link : links ){
6     String linkHref = link.attr("href");
7     String linkText = link.text(); }
```

Listing 2.6: Esempio del metodo di attraversamento DOM

Elements fornisce una gamma di metodi simili a DOM per trovare elementi, estrarre e manipolare i loro dati. I metodi get sono contestuali, chiamati su un nodo padre Document trova elementi corrispondenti sotto quel

documento; chiamato su un nodo figlio trova elementi iniziando da quel nodo.

Pertanto così vengono ricercati i dati da utilizzare, tramite metodi specifici applicabili su un oggetto Element :

- `getElementById(String id)`: trova l'elemento con l'`id` indicato.

L'`id` è un attributo che assume per ogni elemento un valore univoco in tutto il documento HTML.

- `getElementsByTag(String tag)`: trova gli elementi di un determinato tipo, cioè quelli con il `tag` di apertura corrispondente a quello passato come stringa.

- `getElementsByClass(String className)`: trova gli elementi con il valore dell'attributo `class` corrispondente al valore passato come stringa. Proprio questo, all'interno del codice, è stato il metodo più utilizzato per recuperare gli elementi contenenti i nomi, le descrizioni e altre informazioni delle aziende.

- `getElementsByAttribute(String key)`: trova gli elementi che contengono l'`attributo` passato come stringa.

Esistono anche altri metodi associati che consentono di trovare elementi con attributi che iniziano, terminano o contengono la stringa desiderata, oppure metodi che consentono di trovare elementi con attributi che assumono un valore specifico. Nel nostro caso in particolare ci riferiamo a `getElementsByAttributeValues()`.

Ulteriori metodi forniti dalla libreria per attraversare l'albero sono:

- **siblingElements()**: Ottiene tutti i nodi dell'elemento allo stesso livello dell'albero .
- **firstElementSibling()**: Ottiene il primo elemento figlio del nodo.
- **lastElementSibling()**: Ottiene l'ultimo elemento figlio del nodo.
- **nextElementSibling()**: Ottiene il prossimo elemento figlio del nodo.
- **previousElementSibling()**: Ottiene il precedente elemento figlio del nodo.
- **parent()**: Ottiene il nodo dell'elemento genitore.
- **children()**: Ottiene tutti i nodi degli elementi figli.
- **child(int index)**: Ottiene il nodo dell'elemento figlio con indice indicato dal parametro index.

Esistono poi metodi alternativi più potenti per trovare gli elementi che consistono nell'uso dei **selettori**, i quali permettono di selezionare e manipolare gli elementi HTML, come mostrato nel codice 2.7.

Tali elementi possono essere individuati mediante l'**id**, la **classe**, gli **attributi**, i valori degli attributi e altro ancora.

```
1 File input = new File("/tmp/input.HTML");
2 Document doc = Jsoup.parse(input, "UTF-8", "http://esempio.
    com/");
3 Element masthead = doc.select("div.masthead").first();
4 Elements resultLinks = doc.select("h3.r > a"); }
```

Listing 2.7: Codice di esempio per la ricerca con i selettori.

I selettori si possono richiamare su oggetti di tipo: `Document`, `Element` o `Elements`, perciò sono contestuali, e dunque si può filtrare scegliendo un elemento in particolare oppure scegliendo una catena di selettori.

Infine la selezione restituisce una lista contenuta nell'oggetto chiamante `Elements`, esso produce una serie di metodi per l'estrazione e la manipolazione dei risultati.

I principali selettori sono:

- `tagname`: trova elementi per tag, ad esempio '`a`'.
- `ns | tag`: trova elementi per tag in un name space.
- `#id`: trova elementi per ID.
- `.class`: trova elementi per nome classe.
- `[attribute]`: trova elementi con nome dell'attributo uguale a quello passato.
- `[@attr]`: trova elementi con nome dell'attributo con tale prefisso.
- `[attr=value]`: trova gli elementi con attributo che assume il valore dato.
- `[attr^ = value]`, `[attr$ = value]`, `[attr*= value]`: trova gli elementi con valore dell'attributo che inizia con, termina con o che contiene il valore dato.

I selettori inoltre si possono anche combinare tra loro, ad esempio:

- `el.class`: trova un tipo di elementi con class, ad es.'`div.masthead`'

- `el#id`: trova un certo tipo di elementi con ID, ad es. `'div#logo'`.
- `el[attr]`: trova un certo tipo di elementi con attributo specifico.
- `ancestor child`: trova gli elementi figlio che discendono dall'antenato, ad esempio `'.body p'` trova tutti i `'p'` elementi sotto il blocco della classe `'body'`.
- `parent>child`: trova gli elementi figlio che discendono direttamente dal padre.
- `siblingA+SiblingB`: trova l'elemento siblingB immediatamente preceduto dal siblingA.
- `siblingA ~ siblingX`: trova l'elemento siblingX preceduto da siblingA.
- `el, el, el`: raggruppa più selettori, trova elementi unici che corrispondono a uno qualsiasi dei selettori.

Tutti i selettori si possono combinare tra loro, è proprio per questo che si distingue la loro potenza rispetto ai singoli metodi visti in precedenza.

I selettori vengono attuati usando il metodo `select(String cssQuery)` della classe `Element`. Tale metodo, come quelli visti inizialmente, restituisce la lista di elementi come un oggetto di tipo `Elements`, che altro non è un'implementazione dell'interfaccia `List` di Java.

Questi sono i metodi principali per estrarre i dati da un oggetto `Element`:

- `text()`: ottiene il testo puro contenuto tra i tag di apertura e di chiusura.

- `attr(String key)`: per ottenere il valore dell'attributo key.
- `attributes()`: per ottenere tutti gli attributi.
- `HTML()`: restituisce il codice HTML contenuto tra i tag di apertura e chiusura.
- `outerHTML()`: per ottenere il valore HTML esterno.

Aspetti negativi

Tuttavia questo primo approccio è stato abbandonato dato che quando si accede a dati contenuti su pagine web esiste il rischio che, se per qualsiasi motivo la struttura della pagina cambia, anche solo dal punto della grafica, il meccanismo di parsing potrebbe non funzionare più correttamente.

Infatti sarebbe meglio accedere a dati in XML o in JSON, che non hanno questo problema poiché non sono pensati per essere visualizzati.

Nella sezione successiva vederemo come è stato implementato questo secondo approccio.

2.2.3 Google Places API

L'API **Places** è un servizio che restituisce informazioni sui luoghi utilizzando le richieste HTTP. I luoghi sono definiti all'interno di questa API come stabilimenti, posizioni geografiche o punti di interesse importanti.[8]

Le richieste di luogo disponibili sono le seguenti:

- **Place Search**: restituisce un elenco di luoghi in base alla posizione dell'utente o alla stringa di ricerca.

- **Place Details:** restituisce informazioni più dettagliate su un luogo specifico, comprese le recensioni degli utenti.
- **Place Photos:** consente di accedere a milioni di foto relative ai luoghi archiviate nel database di Google Place.
- **Place Autocomplete:** inserisce automaticamente il nome e/o l'indirizzo di un luogo durante la digitazione degli utenti.
- **Query Autocomplete:** fornisce un servizio di previsione delle query per ricerche geografiche basate su testo, restituendo le query suggerite durante la digitazione degli utenti.

Ciascuno di questi servizi è accessibile come richiesta HTTP e restituisce una risposta JSON o XML.

Tutte le richieste a un servizio Places devono utilizzare il protocollo **https://** e includere una **chiave API**, la quale è un identificatore univoco utilizzato per autenticare le richieste associate al progetto ai fini dell'utilizzo e della fatturazione.

Place Search

Essa restituisce un elenco di luoghi insieme a informazioni di riepilogo su ciascun luogo; ulteriori informazioni sono disponibili tramite una query **Place Details**.

Esistono diverse tipologie di richieste al riguardo di Place Search, tra queste abbiamo:

- **Find Place requests:** la quale accetta un input di testo e restituisce un luogo. L'input di testo può essere qualsiasi tipo di dati di Places, ad esempio un nome, un indirizzo o un numero di telefono.

Tale richiesta è un URL HTTP nel formato mostrato nel codice 2.8.

```
1 https://maps.googleapis.com/maps/api/place/  
    findplacefromtext/output?parameters
```

Listing 2.8: Esempio richiesta Find Place

Dove **output** può essere JSON o XML mentre i **parameters** richiesti sono: l'**API Key** dell'applicazione, un **input** che specifica il luogo da cercare e un **inputtype** il quale può essere textquery o phonenumbers.

Si hanno inoltre anche una serie di parametri opzionali come: **language**, **locationbias** (indica la preferenza dei risultati in un area specifica) e **fields** (indica i campi che specificano i tipi di dati dei luoghi da restituire).

E' importante evidenziare che i campi sono divisi in tre categorie di fatturazione diverse, quali:

- **Base**, include i seguenti campi: formatted_address, geometry, icon, name, permanently_closed, photos, place_id, plus_code, types.
- **Contatto**, include i seguenti campi: opening_hours.
- **Atmosfera**, include i seguenti campi: price_level, rating.

- **Nearby Search requests:** Una ricerca nelle vicinanze consente di cercare luoghi all'interno di un'area specifica. E' possibile perfezionare

la richiesta di ricerca fornendo parole chiave o specificando il tipo di luogo che si sta cercando.

Una richiesta di ricerca nelle vicinanze è un URL HTTP nel formato mostrato nel codice 2.9.

```
1 https://maps.googleapis.com/maps/api/place/nearbysearch/  
    output?parameters
```

Listing 2.9: Esempio richiesta Nearby Search

Come nella richiesta precedente **output** può essere JSON o XML mentre i **parameters** richiesti sono: l'**API Key** dell'applicazione, una **location** che specifica la latitudine/longitudine attorno alla quale recuperare le informazioni sul luogo, un **radius** che definisce la distanza (in metri) entro la quale restituire i risultati del luogo e **rankby** che specifica l'ordine in cui sono elencati i risultati.

Si noti che **radius** non deve essere incluso se è specificato **rankby** e viceversa.

Anche in questo caso si hanno una serie di parametri opzionali come: **language**, **keyword**, **name**, **type** (Limita i risultati ai luoghi corrispondenti al tipo specificato.), **opennow** (restituisce solo i luoghi aperti per le attività commerciali al momento dell'invio della query) e **minprice/maxprice** (imita i risultati solo ai luoghi all'interno dell'intervallo specificato. I valori validi vanno da 0 (il più conveniente) a 4 (il più costoso), inclusi.).

- **Text Search requests:** Il servizio di ricerca di testo dell'API di **Google Places** è un servizio Web che restituisce informazioni su una

serie di luoghi in base a una stringa. esso risponde con un elenco di posizioni corrispondenti alla stringa di testo e all'eventuale distorsione di posizione impostata.

Il servizio è particolarmente utile per eseguire query di indirizzi ambigue in un sistema automatizzato e i componenti senza indirizzo della stringa possono corrispondere alle aziende e agli indirizzi.

La risposta della ricerca includerà un elenco di luoghi alla quale è possibile inviare una richiesta **Place Details** per ulteriori informazioni su uno qualsiasi dei luoghi nella risposta.

Una richiesta di ricerca di testo è un URL HTTP nel formato mostrato nel codice 2.10.

```
1 https://maps.googleapis.com/maps/api/place/textsearch/  
    output?parameters
```

Listing 2.10: Esempio richiesta Text Search

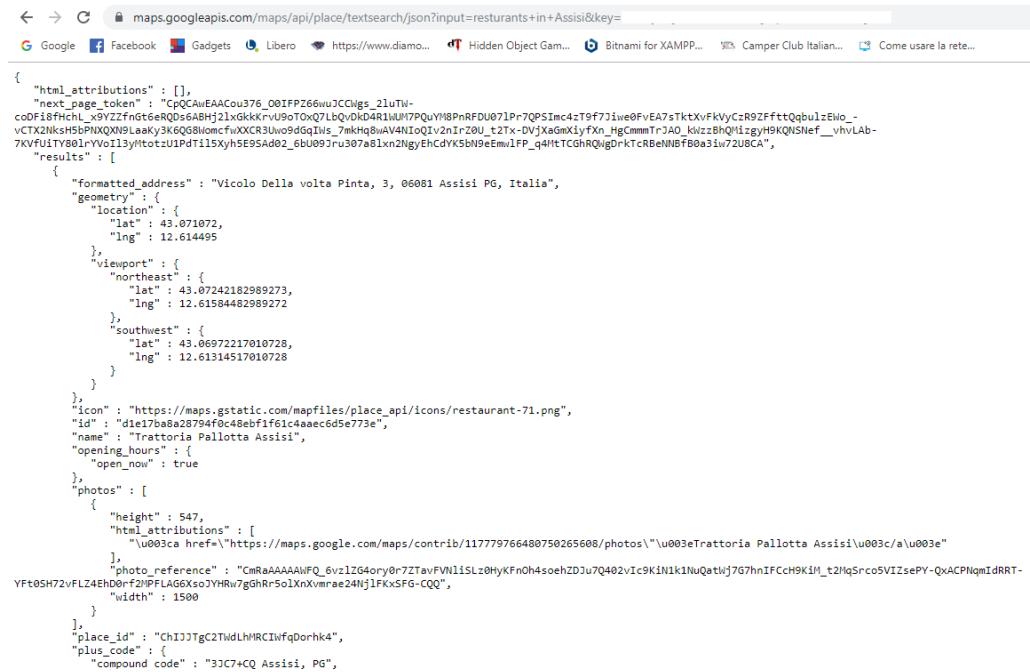
Anche in questa richiesta **output** può essere JSON o XML mentre i **parameters** richiesti sono: l'**API Key** dell'applicazione e una **query** ovvero la stringa di testo su cui cercare. I parametri opzionali sono gli stessi della richiesta precedente con l'aggiunta di **radius** e **region** (cioè il codice regionale che influenzera ma non limiterà completamente i risultati della ricerca).

Dunque proprio quest'ultima richiesta, **Text Search**, è stata utilizzata per ricercare le diverse aziende nella zona, come mostrato nel codice 2.11 e nella Figura 2.3.

Analisi dei requisiti e progettazione

```
1 https://maps.googleapis.com/maps/api/place/textsearch/json?  
    input=restaurants+in+Assisi&key="MY_API_KEY"
```

Listing 2.11: Esempio ricerca ristoranti in Assisi



The screenshot shows a browser window with the URL `https://maps.googleapis.com/maps/api/place/textsearch/json?input=restaurants+in+Assisi&key="MY_API_KEY"`. The page displays the JSON response for the search query. The response includes fields such as 'html_attributions' (empty), 'next_page_token' (a long string of characters), 'results' (a single result object), 'icon' (URL for a restaurant icon), 'id' (unique identifier), 'name' ('Trattoria Pallotta Assisi'), 'opening_hours' (open now), 'photos' (a list of photo details including height, width, and photo reference), 'place_id' ('ChIJJTgC2TdLhMRCIInfqDorhk4'), 'plus_code' ('3JJC7+CQ Assisi, PG'), and 'compound_code' ('3JJC7+CQ Assisi, PG').

```
{  
  "html_attributions" : [],  
  "next_page_token" : "CpcDAwEAAcou376_09IFPz66wu3CCWgs_2luTlu-  
coDF18fHchL_x9VZZfn6teR00d6ABHj2lxGkKxvU9OToxQ7LbQxDkD4R1VWu79QuVY8PnRFDU07lP-r7QPSImc4zT9f7jive0FvEA7sTktxvFkvycR9ZfftQpbulzEwo_-  
vCTX2Nks5bPNKQXNLaaky3K6Q9h0mcwfXXCR3luo9dgQ1is_7khkq8uA4VNIPQ1v2IrZ0U_t2Tx-DVjXeGmxiyfxN_hGcmmtTrjAO_kuzzBhQVigzyH9kQNSNef__vhvLab-  
7KVfUitY0lrvVo13yItotzu1pdt1l5yh5E9Sa6d82_6b1097ru307a81xn2NgxEhCdYK5bh@eEmw1FP_qd4ttCghRQvglDr-kTcRB@NlBF00a3iw72uBCA",  
  "results" : [  
    {  
      "formatted_address" : "Vicolo Della volta Pinta, 3, 06081 Assisi PG, Italia",  
      "geometry" : {  
        "location" : {  
          "lat" : 43.071072,  
          "lng" : 12.614495  
        },  
        "viewport" : {  
          "northeast" : {  
            "lat" : 43.0724182898273,  
            "lng" : 12.61584462989272  
          },  
          "southwest" : {  
            "lat" : 43.06972217010728,  
            "lng" : 12.61314517010728  
          }  
        }  
      },  
      "icon" : "https://maps.gstatic.com/mapfiles/place_api/icons/restaurant-71.png",  
      "id" : "d1e17ba8a28794fc048ebff61c4aec6d5e773e",  
      "name" : "Trattoria Pallotta Assisi",  
      "opening_hours" : {  
        "open_now" : true  
      },  
      "photos" : [  
        {  
          "height" : 547,  
          "html_attributions" : [  
            "\u003ca href=\"https://maps.google.com/maps/contrib/117779766480750265608/photos/\u003eTrattoria Pallotta Assisi\u003c/a\u003e"  
          ],  
          "photo_reference" : "CmRaAAAAAlFQ_6vz1ZG4oryr7ZTavFVn1iSLz0HyKFnOh4soeh2DJu7Q402vIc9KiN1k1NuQatkj7G7hnIFCc9K1M_t2Mq5rc05VIZsePY-QxACPNqm1dRRT-  
YFt0SH72vFLZ4Eh0rnf2MPFLAG6xsoJYHRw/gGhRr5o1XnXvmrae24Hj3lFKxFSG-CQQ",  
          "width" : 1500  
        }  
      ],  
      "place_id" : "ChIJJTgC2TdLhMRCIInfqDorhk4",  
      "plus_code" : {  
        "compound_code" : "3JJC7+CQ Assisi, PG"  
      }  
    }  
  ]  
}
```

Figura 2.3: Risultato della ricerca

Place Details

Come già detto questa Place request può richiedere maggiori informazioni su un particolare stabilimento o punto di interesse avviando una richiesta dettagliata del luogo.

Una richiesta Place Details è un URL HTTP nel formato mostrato nel codice 2.12.

```
1 https://maps.googleapis.com/maps/api/place/details/output?  
    parameters
```

Listing 2.12: Esempio richiesta Place Details

Dove **output** può essere JSON o XML mentre i **parameters** richiesti sono: l'**API Key** dell'applicazione e **Place_id**, cioè un identificatore testuale che identifica in modo univoco un luogo, restituito da Place Search.

Infine si hanno una serie di parametri opzionali come: **language**, **region** e **field**.

Place Photos

Il servizio **Place Photo**, parte dell'API di **Places**, è un'API di sola lettura che consente di aggiungere contenuti fotografici di alta qualità alle applicazioni. Tale servizio dà accesso a milioni di foto archiviate nel database di Places. Quando si ottengono informazioni sul luogo utilizzando una richiesta **Places Details**, i riferimenti fotografici (**photo references**) verranno restituiti per il contenuto fotografico rilevante. Le richieste di **Nearby Search** e di **Text Search** restituiscono un singolo riferimento fotografico per ogni luogo, se pertinente. Utilizzando dunque questo servizio è quindi possibile accedere alle foto di riferimento e ridimensionare l'immagine alla dimensione ottimale per l'applicazione.

Tutte le richieste al servizio Place Photo devono includere **photoreference**, che viene ricevuto in risposta a una richiesta di **Nearby Search**, **Text Search** o **Places Details**. Se il luogo ha contenuti fotografici correlati, la risposta a queste richieste conterrà un campo **photos[]**, mostrato nel codice 2.13.

Ogni elemento **photos** conterrà i seguenti campi:

- **photo_reference**, è una stringa utilizzata per identificare la foto quando si esegue una richiesta di foto.
- **height**, l'altezza massima dell'immagine.
- **width**, la larghezza massima dell'immagine.
- **HTML_attributions[]**, contiene tutte le attribuzioni richieste. Questo campo è sempre presente, ma potrebbe essere vuoto.

```
1
2 "photos" : [
3   {
4     "HTML_attributions" : [],
5     "height" : 853,
6     "width" : 1280,
7     "photo_reference" : "CnRvAAAAwMpdHeWlXl-
8       1H0vp7lez4znKPIWSWvgvZFISdKx45AwJVP1Qp37Y0rH7sqHMJ8C -
9       vBDC546decipPHchJhHZL94RcTUFPa1jWzo -rSHaTlbNtjh -
10      N68RkcToUCuY9v2HNpo5mziqkir37WU8FJEqVBIQ4k938TI3e7bf8xq -
11      uwDZcxoUb0_ZJzPxremiQurAYzCTwRhE_V0"
12 }
```

Listing 2.13: Esempio elemento photos

Una richiesta Place Photo è un URL HTTP nel formato mostrato nel codice 2.14.

```
1 https://maps.googleapis.com/maps/api/place/photo?parameters
```

Listing 2.14: Esempio richiesta Place Photo

Dove i **parameters** richiesti sono: l'**API Key** dell'applicazione, **photoreference** e **maxheight/maxwidth**.

Place Autocomplete

Il servizio di completamento automatico del luogo è un servizio Web che restituisce previsioni di luogo in risposta a una richiesta HTTP. La richiesta specifica una stringa di ricerca testuale e limiti geografici opzionali. Il servizio può essere utilizzato per fornire funzionalità di completamento automatico per ricerche geografiche basate su testo, restituendo luoghi come attività commerciali, indirizzi e punti di interesse come tipi di utenti.

Il servizio può corrispondere a parole intere e sottostringhe, quindi le applicazioni possono inviare query man mano che l'utente digita, per fornire previsioni sul posto al volo.

Le previsioni restituite sono progettate per essere presentate all'utente per aiutarlo nella scelta del luogo desiderato. Inoltre è possibile inviare una richiesta Place Details per ulteriori informazioni su uno qualsiasi dei luoghi che vengono restituiti.

Una richiesta di Place Autocomplete è un URL HTTP nel formato mostrato nel codice 2.15.

```
1 https://maps.googleapis.com/maps/api/place/autocomplete/  
    output?parameters
```

Listing 2.15: Esempio richiesta Place Autocomplete

Dove **output** può essere JSON o XML mentre i **parameters** richiesti sono: l'**API Key** dell'applicazione e un **input**, cioè la stringa di testo su cui cercare.

Ulteriori parametri opzionali sono: **language**, **radius** (La distanza (in metri) entro la quale restituire i risultati del luogo), **type** (I tipi di risultati dei luoghi da restituire), **location** (il punto attorno al quale desideri recuperare le informazioni sul luogo), **sessiontoken** (Una stringa casuale che identifica una sessione di completamento automatico ai fini della fatturazione) e **offset** (La posizione, nel termine di input, dell'ultimo carattere che il servizio utilizza per abbinare le previsioni).

In risposta alla richiesta si ottiene un Json/XML contenente due elementi radice:

- **status**, che contiene metadati sulla richiesta. Esso può assumere diversi valori, come: OK, ZERO_RESULTS, OVER_QUERY_LIMIT, REQUEST_DENIED e INVALID_REQUEST.
- **predictions**, che contiene una serie di luoghi, con informazioni sul luogo. L'API Places restituisce fino a 5 risultati.

Ogni risultato di previsione contiene i seguenti campi: description, place_id, types, terms e matched_substrings.

Questo servizio di richiesta è stato dunque utilizzato all'interno dell'applicazione per poter semplificare e suggerire all'utente la ricerca di un luogo.

Query Autocomplete

Quest'ultima tipologia di richiesta di luogo può essere utilizzata per fornire una previsione delle query per ricerche geografiche basate su testo, restituendo le query suggerite durante la digitazione.

Dunque il servizio **Query Autocomplete** risulta avere una funzionalità molto simile a **Place Autocomplete** e consente di aggiungere previsioni di

query geografiche al volo all'applicazione. Invece di cercare una posizione specifica, un utente può digitare una ricerca categorica e il servizio risponde con un elenco di query suggerite corrispondenti alla stringa.

Poiché il servizio di completamento automatico delle query può corrispondere sia a parole intere sia a sottostringhe, le applicazioni possono inviare query mentre l'utente digita per fornire previsioni al volo.

Una richiesta di completamento automatico di una query è un URL HTTP nel seguente formato. Vedi il Listing 2.16.

```
1 https://maps.googleapis.com/maps/api/place/queryautocomplete  
/output?parameters
```

Listing 2.16: Esempio richiesta Query Autocomplete

Dove **output** può essere JSON o XML mentre i **parameters** richiesti sono: l'**API Key** dell'applicazione e un **input**, cioè la stringa di testo su cui cercare.

Ulteriori parametri opzionali che si differenziano da Place Autocomplete sono: **offset** (La posizione del carattere nel termine di input in cui il servizio utilizza il testo per le previsioni), **radius** (La distanza (in metri) entro la quale restituire i risultati del luogo) e **location** (Il punto attorno al quale desideri recuperare le informazioni sul luogo).

Anche in questo caso si ottiene come risposta alla richiesta un Json/XML contenente i due elementi **status** e **predictions**.

Aspetti negativi

L'API Place di Google risulta non essere gratuita ma utilizza un modello di prezzi **pay-as-you-go**, cioè il pagamento è rapportato al consumo del

servizio.

Tutta via una volta impostato l'account di fatturazione si avrà diritto a un credito gratuito di 300\$ (utilizzabile per i prodotti della piattaforma Google Cloud) e un credito gratuito ricorrente di 200\$ mensile (esclusivo per i prodotti della piattaforma di Google Maps), dopo aver consumato i crediti, si riceverà un *OVER_QUERY_LIMIT* e non verrà addebitato niente. Se si desidera non ricevere un errore quando i crediti sono esauriti, si può aggiornare l'account di fatturazione e verrà addebitato di conseguenza per l'utilizzo dopo i crediti.

L'utilizzo e la fatturazione secondo il modello *pay-as-you-go* funziona come segue:

- Le API della piattaforma Google Maps sono fatturate da SKU (Stock Keeping Units).
- L'utilizzo viene monitorato per ogni SKU prodotto e un'API può avere più di un SKU prodotto.
- Il costo è calcolato da: SKU Utilizzo x Prezzo per ogni utilizzo.

A questo punto, consultando le relative tabelle dei costi, è stato sviluppato un preventivo.

Come già detto l'API Place utilizzata è stata **Text Search**, la quale non supporta la specifica dei campi di dati da restituire, fornendo sempre tutti i dati di **Places**.

Quindi oltre al costo per la richiesta **Text Search** viene addebitato un costo per le tre tipologie di dati fornite (base, contatto e atmosfera). In Figura 2.4 vengono riportati i prezzi inerenti a **Text Search**.

Analisi dei requisiti e progettazione

GAMMA VOLUME MENSILE (Prezzo per CHIAMATA)		
0-100,000	100,001-500,000	500.000+
0,032 USD per ciascuno (32,00 USD per 1000)	0,0256 USD per ciascuno (25,60 USD per 1000)	Contattare le vendite per i prezzi in volume

Figura 2.4: Tabella prezzi Places - Text Search.

Dati di Base: I campi nella categoria **Base** sono inclusi nel costo base della richiesta Places e non comportano costi aggiuntivi. Viene preso in considerazione quando uno di questi campi è obbligatorio: `address_component`, `adr_address`, `formatted_address`, `geometry`, `icon`, `name`, `permanently_closed`, `photo`, `place_id`, `plus_code`, `type`, `url`, `utc_offset`, `vicinity`.

In Figura 2.5 vengono mostrati i prezzi dei dati di base.

GAMMA VOLUME MENSILE (Prezzo per CHIAMATA)		
0-100,000	100,001-500,000	500.000+
La richiesta di posti costa + 0,00 USD	La richiesta di posti costa + 0,00 USD	La richiesta di posti costa + 0,00 USD

Figura 2.5: Tabella prezzi Dati di base.

Dati di Contatto: I campi nella categoria **Contatto** comportano un costo aggiuntivo. Viene preso in considerazione quando uno di questi campi è obbligatorio: `formatted_phone_number`, `international_phone_number`, `opening_hours`, `website`. In Figura 2.6 vengono mostrati i prezzi relativi a tale categoria.

Dati di Atmosfera: I campi nella categoria **Atmosfera** comportano un costo aggiuntivo. Viene preso in considerazione quando uno di questi campi è

Descrizione e Funzionamento applicazione

GAMMA VOLUME MENSILE (Prezzo per CHIAMATA)		
0-100,000	100,001-500,000	500.000+
Costo della richiesta di posti + 0,003 USD per ciascuno (+ 3,00 USD per 1000)	Costo della richiesta di posti + 0,0024 USD per ciascuno (+ 2,40 USD per 1000)	Contattare le vendite per i prezzi in volume

Figura 2.6: Tabella prezzi Dati di Contatto

obbligatorio: `price_level`, `rating`, `review`, `user_ratings_total`. I prezzi vengono mostrati in Figura 2.7.

GAMMA VOLUME MENSILE (Prezzo per CHIAMATA)		
0-100,000	100,001-500,000	500.000+
Costo della richiesta di posti + 0,005 USD per ciascuno (+ 5,00 USD per 1000)	Costo della richiesta di posti + 0,004 USD per ciascuno (+ 4,00 USD per 1000)	Contattare le vendite per i prezzi in volume

Figura 2.7: Tabella prezzi Dati di Atmosfera

Place Photos: Le richieste di caricamento foto hanno un costo, mostrato in Figura 2.8.

GAMMA VOLUME MENSILE (Prezzo per CHIAMATA)		
0-100,000	100,001-500,000	500.000+
0,007 USD per ciascuno (7,00 USD per 1000)	0,0056 USD per ciascuno (5,60 USD per 1000)	Contattare le vendite per i prezzi in volume

Figura 2.8: Tabella prezzi Place Photos

Place Autocomplete: Viene addebitato un costo per le richieste al servizio

Capitolo 3

Descrizione e Funzionamento applicazione

In questo ultimo capitolo verranno esposte ed analizzate particolari caratteristiche dell'applicazione.[9]

Infine verrà data una descrizione dell'applicazione e del suo funzionamento, illustrando dove e con quale logica sono state implementate le tecniche descritte nel capitolo precedente.

3.1 AsyncTask

Per la gestione della User Interface Thread, cioè la thread principale di esecuzione dell'applicazione, è stata messa a disposizione la classe astratta **AsyncTask**.

Tale classe permette quindi di controllare l'aggiornamento dell'interfaccia utente per rendere l'applicazione di un utilizzo semplice ed immediato.

Descrizione e Funzionamento applicazione

Un task asincrono è definito da una computazione che viene eseguita su una thread in background e il cui risultato viene pubblicato sulla UI Thread, come mostrato in Figura 3.1.

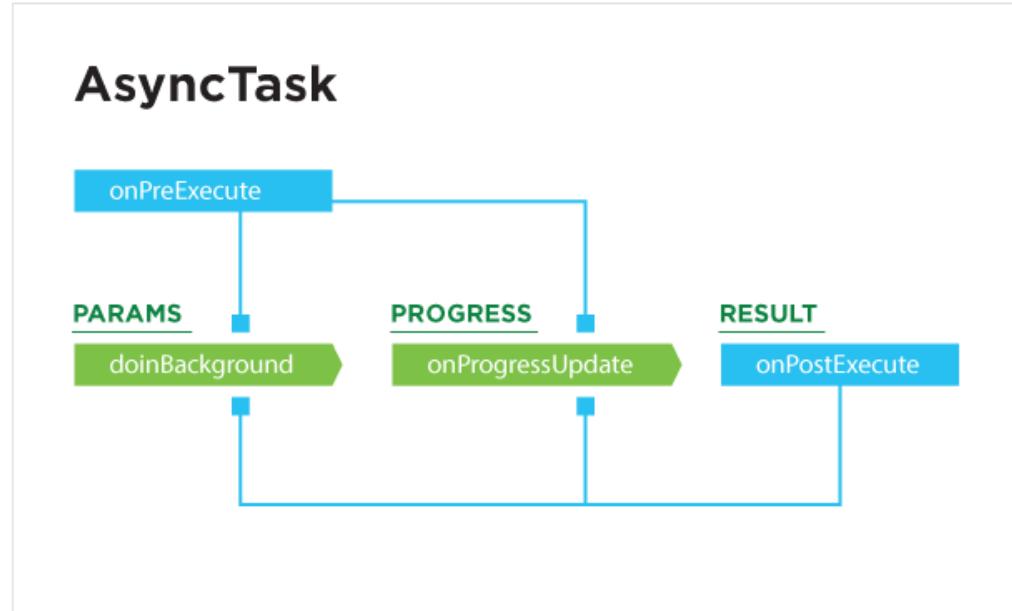


Figura 3.1: Funzionamento Asynctask

La classe `AsyncTask<P,P,R>` viene dunque definita come una classe generica con tre tipi parametrici:

- **Params**, il tipo di dato che viene passato al task prima di poter essere eseguito.
- **Progress**, il tipo di dato che viene pubblicato durante l'esecuzione della computazione in background.
- **Results**, il tipo di dato ottenuto alla fine della computazione in background.

Descrizione e Funzionamento applicazione

e con quattro metodi predefiniti (protected):

- **onPreExecute()**, prima che il task venga eseguito viene invocato sulla UI Thread. Questo passo viene solitamente chiamato per collocare il task, ad esempio mostrando una progress bar all'utente.
- **doInBackground(Params)**, viene chiamato dalla thread in background appena **onPreExecute()** termina. Il seguente passo è utilizzato per eseguire il compito destinato all'AsyncTask impiegando anche un tempo sufficientemente lungo e restituendo un risultato di tipo **Result**. Inoltre è possibile usare anche il metodo **publishProgress(Progress)** per rendere accessibili i valori che si stanno recuperando. Essi nel passo **onProgressUpdate()** sono poi pubblicati sulla UIThread.
- **onProgressUpdate(Progress)**, viene chiamato dalla UI Thread subito dopo che è stato invocato il metodo **publishProgress()**, tuttavia il tempo di esecuzione di questo passo non è definito. Questo metodo è usato di solito per rendere visibile all'utente il progresso della computazione che si sta compiendo in background.
- **onPostExecute(Result)**, viene chiamato dalla UI Thread subito dopo che il calcolo in background è concluso. Il risultato del task eseguito in precedenza viene passato come parametro in questo step.

Vediamo adesso in che modo viene svolta l'esecuzione di un **AsyncTask**:

1. Si definisce una **sottoclasse AT** che estenda **Asynctask** implementando i metodi che interessano (obbligatoriamente **doInBackground**) e specificando a cosa corrispondono i tipi P, P e R. Nel caso in cui non serve qualcuno di questi tipi, si usa il tipo **Void**.

Descrizione e Funzionamento applicazione

2. L'UI Thread, interessata ad eseguire un'operazione lunga ma impossibilitata a farlo direttamente, crea un oggetto di tipo AT e lo avvia con il metodo `execute(P)` a cui passa come parametri tutti i dati necessari di tipo `Params`.
3. Quando sta per partire il task, Android esegue nell'UI Thread il metodo `onPreExecute()` se implementato.
4. Viene creata una nuova thread T2 che esegue `doInBackground(P)` con gli stessi parametri presenti in `execute(P)`.
5. (Passo facoltativo) Con frequenza più regolare possibile, T2 esegue il metodo `publishProgress()` con cui indica tramite uno o più parametri di tipo `Progress` a che punto è arrivata.
6. Ad ogni chiamata di `publishProgress()`, Android esegue nell'UI Thread il metodo `onProgressUpdate(Progress)` sugli stessi parametri se implementato.
7. Quando T2 ha terminato, il risultato prodotto da `doInBackground()` è passato come argomento al metodo `onPostExecute(Result)`, se implementato, che Android esegue nell'UI Thread.

E' importante notare che la Thread secondaria esegue `doInBackground()` ed eventualmente `publishProgress()`, mentre gli altri metodi sono eseguiti nella Thread principale e quindi possono accedere all'interfaccia utente senza avere problemi.

Gli AsyncTask sono stati utilizzati all'interno dell'applicazione per poter eseguire in background:

Descrizione e Funzionamento applicazione

- La ricerca con Google Places API, passando come parametro l'URL HTTP di ricerca di un luogo. Vedi Listing 3.1.

```
1 new JsonTask().execute("https://maps.googleapis.com/
2   maps/api/place/textsearch/json?input=restaurants+in"+
3   posizione+"&key="+key);
```

Listing 3.1: Esempio AsyncTask utilizzato per la ricerca dei ristoranti.

All'interno del metodo `doInBackground()` viene effettuata la connessione a tale URL e memorizzato all'interno di un buffer il Json contenente le informazioni riga per riga. Vedi Listing 3.2.

```
1 protected String doInBackground(String... params) {
2     try {
3         URL url = new URL(params[0]);
4         HttpURLConnection connection = (HttpURLConnection)
5             url.openConnection();
6         connection.connect();
7         InputStream stream = connection.getInputStream();
8         BufferedReader reader = new BufferedReader(new
9             InputStreamReader(stream));
10        StringBuffer buffer = new StringBuffer();
11        String line = "";
12        while ((line = reader.readLine()) != null) {
13            buffer.append(line+"\n");
14        }
15        return buffer.toString();
16    } catch (MalformedURLException e) {
17        e.printStackTrace();
18    } catch (IOException e) {
19        e.printStackTrace();
20    } finally {
21        if (connection != null) {
```

Descrizione e Funzionamento applicazione

```
19         connection.disconnect();  
20     }  
21     try {  
22         if (reader != null) {  
23             reader.close();}  
24     } catch (IOException e) {  
25         e.printStackTrace();}  
26     } return null;  
}
```

Listing 3.2: Esempio metodo doInBackground

Mentre nel metodo `onPostExecute()` viene effettuato il parsing del Json e vengono memorizzati i dati all'interno di diversi ArrayList. Vedi Listing 3.3.

```
1 protected void onPostExecute(String result) {  
2     super.onPostExecute(result);  
3     progressBar.setVisibility(View.GONE);  
4     try{  
5         JSONObject json = new JSONObject(result);  
6         JSONArray jArray= json.getJSONArray("results");  
7         for( int i=0; i<jArray.length();i++){  
8             JSONObject ob = jArray.getJSONObject(i);  
9             name = ob.getString("name");  
10            rating = ob.getString("rating");  
11            address = ob.getString("formatted_address");  
12            try {  
13                JSONArray foto = ob.getJSONArray("photos");  
14                String photo_reference = "";  
15                for (int e = 0; e < foto.length(); e++){  
16                    JSONObject pic = foto.getJSONObject(e);  
17                    photo_reference += pic.getString("photo_reference") + ", ";  
18                }  
19            } catch (Exception e){  
20                e.printStackTrace();  
21            }  
22        }  
23    } catch (Exception e){  
24        e.printStackTrace();  
25    }  
26 }
```

Descrizione e Funzionamento applicazione

```
17         photo_reference = pic.getString("photo_reference");
18     }
19     imgUri.add("https://maps.googleapis.com/maps/api/place/photo?maxwidth=400&photoreference=" +
20     photo_reference+"&key="+key);
21     }catch (JSONException e){
22         e.printStackTrace();
23     imgUri.add("https://lh5.googleusercontent.com/...");}
24     nomi.add(name);
25     info.add(address);
26     info2.add("valutazione:"+rating+" su 5"+ "\n" +
27     "aperto adesso: "+orario); }
28 } catch (JSONException e) {
29     e.printStackTrace();}
```

Listing 3.3: Esempio metodo doInBackground

- La connessione al database MySql, sia per ricevere che per inserire dati. In questo caso all'interno del metodo `doInBackground()` avviene la connessione al file Php e successivamente la memorizzazione o l'inserimento delle informazioni all'interno di un buffer.

Mentre nel metodo `onPostExecute()` avviene il parsing del Json e la memorizzazione dei dati all'interno degli ArrayList. Vedi Listing 3.4.

```
1 protected String doInBackground(Void... voids) {
2 HttpURLConnection con = null;
3     try{
```

```
4     URL url = new URL("http://localhost/name/dati.php");
5
6     con = (HttpURLConnection) url.openConnection();
7
8     con.setRequestMethod("POST");
9
10    con.setDoOutput(true);
11
12    OutputStream outputStream = con.getOutputStream();
13
14    BufferedWriter bufferedWriter = new BufferedWriter(
15        new OutputStreamWriter(outputStream, "UTF-8"));
16
17    String post_data=URLEncoder.encode("nome", "UTF-8")+
18    "="+URLEncoder.encode(nome, "UTF-8");
19
20    bufferedWriter.write(post_data);
21
22    bufferedWriter.flush();
23
24    bufferedWriter.close();
25
26    outputStream.close();
27
28 }catch (Exception e){
29
30     return null;
31
32 }finally {
33
34     con.disconnect();
35 }
```

Listing 3.4: Esempio inserimento dati nel buffer

3.2 RecyclerView

Ottenute le informazioni dalle diverse fonti, si è posto il problema di come poter far visualizzare tali dati all’utente.

La risposta alla domanda è stata utilizzare per l’appunto **RecyclerView**, il quale permette di visualizzare un elenco a scorrimento di elementi basato su grandi set di dati.

Nel modello RecyclerView, componenti diversi lavorano insieme per visualizzare i dati:

- Iniziamo costruendo la RecyclerView, la quale contiene CardViews con 2 ImageView (una dell'azienda e una per segnalare la certificazione) e 3 TextViews ciascuna (le quali illustrano le informazioni dell'azienda).

Le CardViews sono oggetti personalizzati che permettono di mostrare informazioni all'interno di schede che hanno un aspetto coerente su tutta la piattaforma. Tali oggetti necessitano di una classe Java che li definisca e ne permetta l'utilizzo.

Per poter accedere ai due widget e aggiungerli ai Layouts è necessario aggiungere le librerie di supporto all'interno del file `build.gradle`.

Vedi Listing 3.5.

```
1 implementation 'com.android.support:recyclerview-v7:28'  
2 implementation 'com.android.support:cardview-v7:28'
```

Listing 3.5: Librerie di supporto RecyclerView e CardViews

- Proseguiamo impostando un `LayoutManager`, creando un `Adapter` personalizzato e infine completando la nostra RecyclerView in modo che possa visualizzare gli elementi delle CardViews.

```
1 public class MyActivity extends Activity {  
2     private RecyclerView recyclerView;  
3     private RecyclerView.Adapter mAdapter;  
4     private RecyclerView.LayoutManager layoutManager;  
5  
6     protected void onCreate(Bundle savedInstanceState) {  
7         super.onCreate(savedInstanceState);
```

Descrizione e Funzionamento applicazione

```
8         setContentView(R.layout.my_activity);
9
10        recyclerView = (RecyclerView) findViewById(R.id.
11                my_recycler_view);
12
13        recyclerView.setHasFixedSize(true);
14
15        layoutManager = new LinearLayoutManager(this);
16        recyclerView.setLayoutManager(layoutManager);
17        mAdapter = new MyAdapter(myDataset);
18        recyclerView.setAdapter(mAdapter);
19    }
20}
```

Listing 3.6: esempio RecyclerView applicazione

Il contenitore generale per l’interfaccia utente è dunque un oggetto `RecyclerView`, il quale viene riempito di views fornite da un `LayoutManager`, nel nostro caso è stato utilizzato un `LinearLayoutManager`.

Gli elementi delle `CardsViews` sono rappresentati da oggetti di tipo `view holder`, i quali sono delle istanze della sottoclassificazione di `RecyclerView.ViewHolder` e hanno l’unico compito di visualizzare un singolo oggetto con una view.

Mentre gli oggetti `view holder` sono gestiti attraverso un `adapter`, il quale viene creato estendendo la classe `RecyclerView.Adapter`.

L’`adapter` in base all’esigenza crea `view holder` per poi associargli i dati. Lo fa assegnando il `view holder` ad una posizione e richiamando il metodo `onBindViewHolder()` dell’`adapter`. Questo metodo usa la posizione del `view holder` per determinare quale dovrebbe essere il contenuto, in base alla sua posizione della lista. Vedi Listing 3.7.

Descrizione e Funzionamento applicazione

```
1 public class Adapter extends RecyclerView.Adapter<Adapter.
2     ViewHolder> {
3
4     private ArrayList<Elementi> mList;
5
6     private LayoutInflater mInflater;
7
8     Context mContext;
9
10
11     public static class ViewHolder extends RecyclerView.
12         ViewHolder{
13
14         public ImageView mImageView;
15
16         public ImageView mImageView2;
17
18         public TextView mTextView1;
19
20         public TextView mTextView2;
21
22         public TextView mTextView3;
23
24
25         public ViewHolder( View itemView) {
26
27             super(itemView);
28
29             mImageView= itemView.findViewById(R.id.imageView);
30
31             mImageView2= itemView.findViewById(R.id.imageView2);
32
33             mTextView1= itemView.findViewById(R.id.textView);
34
35             mTextView2= itemView.findViewById(R.id.textView2);
36
37             mTextView3= itemView.findViewById(R.id.textView3);
38
39         }
40
41     }
42
43     public Adapter( Context ctx,ArrayList<Elementi> lista){
44
45         this.mInflater= LayoutInflater.from(ctx);
46
47         this.mList = lista;
48
49         mContext=ctx;
50
51     }
52
53     public ViewHolder onCreateViewHolder( ViewGroup parent,
54         int viewType) {
55
56         View v=mInflater.inflate(R.layout.element,parent, false);
57
58         return new ViewHolder(v);
59     }
60
61     @Override
62     public void onBindViewHolder( ViewHolder holder, int position) {
63
64         holder.mTextView1.setText(mList.get(position).text1);
65
66         holder.mTextView2.setText(mList.get(position).text2);
67
68         holder.mTextView3.setText(mList.get(position).text3);
69
70         holder.mImageView.setImageResource(mList.get(position).image);
71
72         holder.mImageView2.setImageResource(mList.get(position).image2);
73
74     }
75
76     @Override
77     public int getItemCount() {
78
79         return mList.size();
80     }
81
82 }
```

Descrizione e Funzionamento applicazione

```
29     ViewHolder vh = new ViewHolder(v);
30
31     return vh;
32 }
33
34     public void onBindViewHolder( ViewHolder holder, int
35     position) {
36
37         Elementi elem = mList.get(position);
38
39         String imageUrl= elem.getImg();
40
41         Boolean cer= elem.getCertificato();
42
43         if(cer){
44
45             holder.mImageView2.setImageResource(R.drawable.
46             ic_star_black_24dp);
47
48         }
49
50         holder.mTextView1.setText(elem.getText1());
51
52         holder.mTextView2.setText(elem.getText2());
53
54         holder.mTextView3.setText(elem.getText3());
55
56         Picasso.with(mcontext).load(imageUrl).error(R.
57         drawable.ic_launcher_background).into(holder.mImageView);
58
59     }
60
61     public int getItemCount() {
62
63         return mList.size();
64     }
65 }
```

Listing 3.7: esempio Adapter applicazione

Dunque il layout manager chiama il metodo `onCreateViewHolder()` dell'adapter. Questo metodo deve costruire un ViewHolder e impostare la vista che utilizza per visualizzare i suoi contenuti, ovvero le CardsView. In genere avviene tramite il caricamento di un file di layout XML.

Infine il layout manager associa il view holder ai suoi dati e lo fa chiamando il metodo `onBindViewHolder()` dell'adapter e passando la posizione

del view holder nel RecyclerView. Il metodo deve quindi recuperare i dati appropriati e utilizzarli per riempire il layout del view holder.

3.3 Spinner

Gli Spinners forniscono un modo rapido per selezionare un valore da un insieme. Nello stato predefinito, un spinner mostra il valore attualmente selezionato. Toccando lo spinner viene visualizzato un menu a tendina con tutti gli altri valori disponibili, dal quale l'utente può selezionarne uno nuovo. La Figura 3.2 mostra un esempio di spinner.

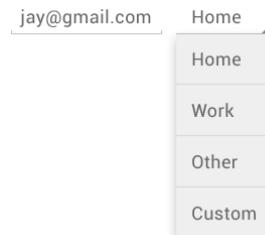


Figura 3.2: Esempio spinner

E' possibile aggiungere uno spinner nel layout XML con un elemento <Spinner>.

Vedi Listing 3.8.

```
1 <Spinner  
2     Android: id= "@+id/spinner"  
3     Android: layout_width= "fill_parent"  
4     android: layout_height= "wrap_content" />
```

Listing 3.8: esempio spinner nel layout XML

Descrizione e Funzionamento applicazione

Per popolare lo spinner con un elenco di scelte, è poi necessario specificare un **SpinnerAdapter** nella propria activity. Le scelte fornite per lo spinner possono provenire da qualsiasi fonte, ma devono essere fornite attraverso un Adapter, come ad esempio un ArrayAdapter se le scelte sono disponibili in un array. Vedi Listing 3.9.

```
1 Spinner spinner = (Spinner)findViewById(R.id.spinner);
2 ArrayAdapter<String> arrayAdapter = new ArrayAdapter(this,
3     android.R.layout.simple_spinner_item, nomiDb);
4 arrayAdapter.setDropDownViewResource(android.R.layout.
5     simple_spinner_dropdown_item);
6 spinner.setAdapter(arrayAdapter);
```

Listing 3.9: esempio spinnerAdapter

Una volta creato l'adapter è necessario richiamare il metodo predefinito **setDropDownViewResource(int)** in modo da specificare il layout che l'adapter stesso dovrebbe usare per visualizzare l'elenco delle scelte dello spinner. Infine viene chiamato **setAdapter()** per applicare l'adapter al proprio Spinner.

Quando l'utente seleziona un elemento dall'elenco a tendina, l'oggetto Spinner riceve un evento di selezione. Per definire l'handler di tale evento viene implementata l'interfaccia **AdapterView.OnItemSelectedListener** tramite il metodo **setOnItemSelectedListener()** e i corrispondenti metodi di callback **onItemSelected()** e **onNothingSelected()**. Vedi Listing 3.10.

```
1 spinner.setOnItemSelectedListener(new AdapterView.
2     OnItemSelectedListener() {
3         public void onItemSelected(AdapterView<?> adapterView,
4             View view, int i, long l) {
5             result = adapterView.getItemAtPosition(i).toString();
```

```
4 }public void onNothingSelected(AdapterView<?>  
5     adapterView) {}  
5 );
```

Listing 3.10: esempio selezione risultato

3.4 Descrizione Funzionale

Diamo adesso una breve descrizione sul funzionamento vero e proprio di Rightsapp, attraverso screenshot dell'applicazione e diagrammi dei casi d'uso.

Essa è strutturata in tre pagine principali:

1. La homepage, nella quale è possibile scegliere quale tipologia di aziende ricercare (Ristoranti, Hotel, Imprese e Negozи) e in quale luogo ricercarle, di default viene utilizzata la propria posizione. La Figura 3.3 mostra l'Homepage dell'applicazione.

E' dunque possibile, mediante un'apposita barra di ricerca, selezionare il luogo desiderato tramite una ricerca automatizzata. Questa ricerca automatica avviene grazie l'utilizzo dell'**API Place Autocomplete di Google**, come accennato nelle sezioni precedenti. La Figura 3.4 mostra il cambio di posizione nella Homepage.

Il caso d'uso in Figura 3.5 mostra il funzionamento della homepage di Rhightsapp, in caso di assenza di internet viene visualizzato un errore di connessione.

2. Nella seconda pagina, vengono illustrati i risultati della ricerca tramite una RecyclerView. Inoltre nella parte inferiore della schermata è

Descrizione e Funzionamento applicazione

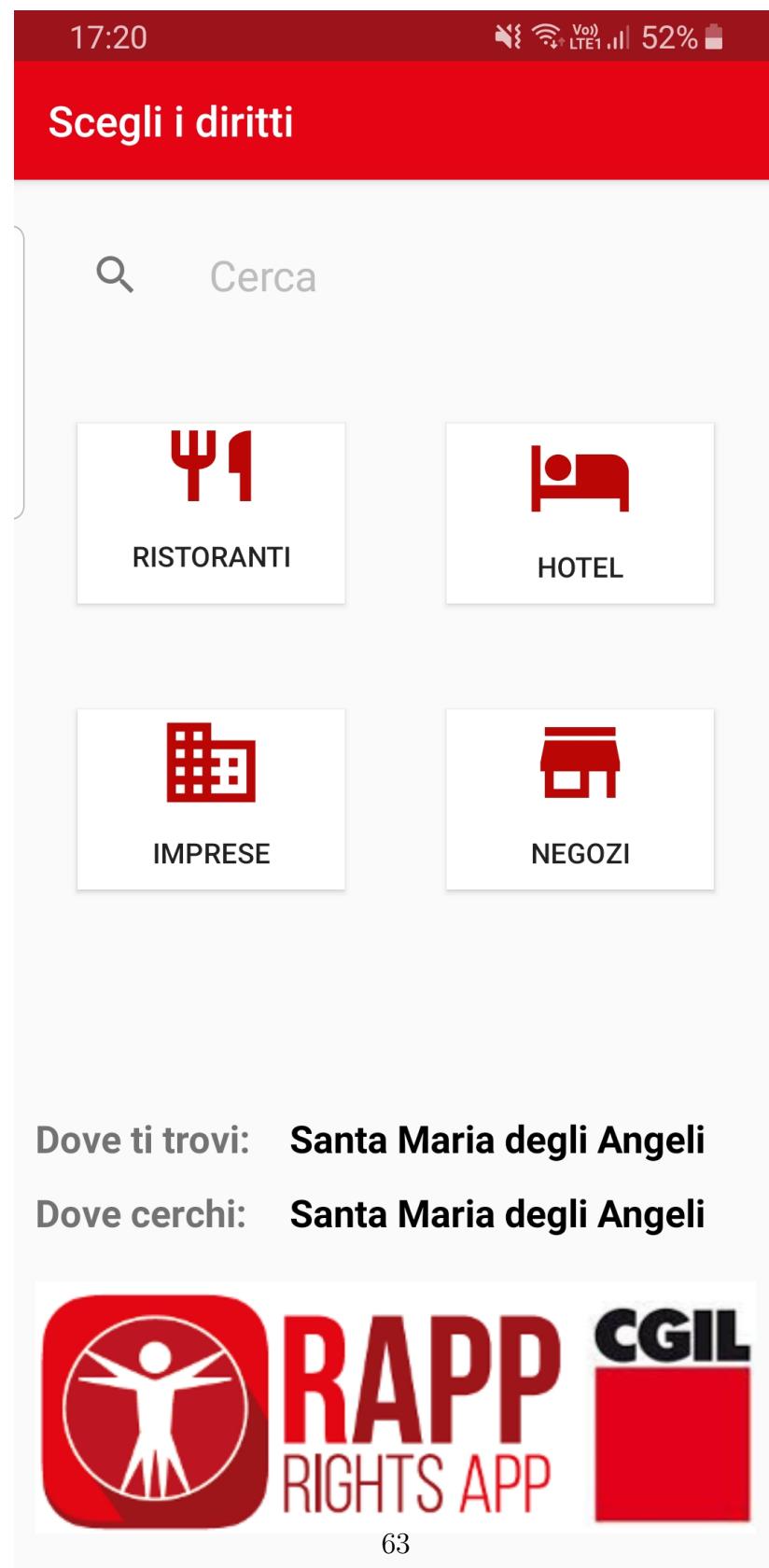


Figura 3.3: Homepage applicazione.

Descrizione e Funzionamento applicazione

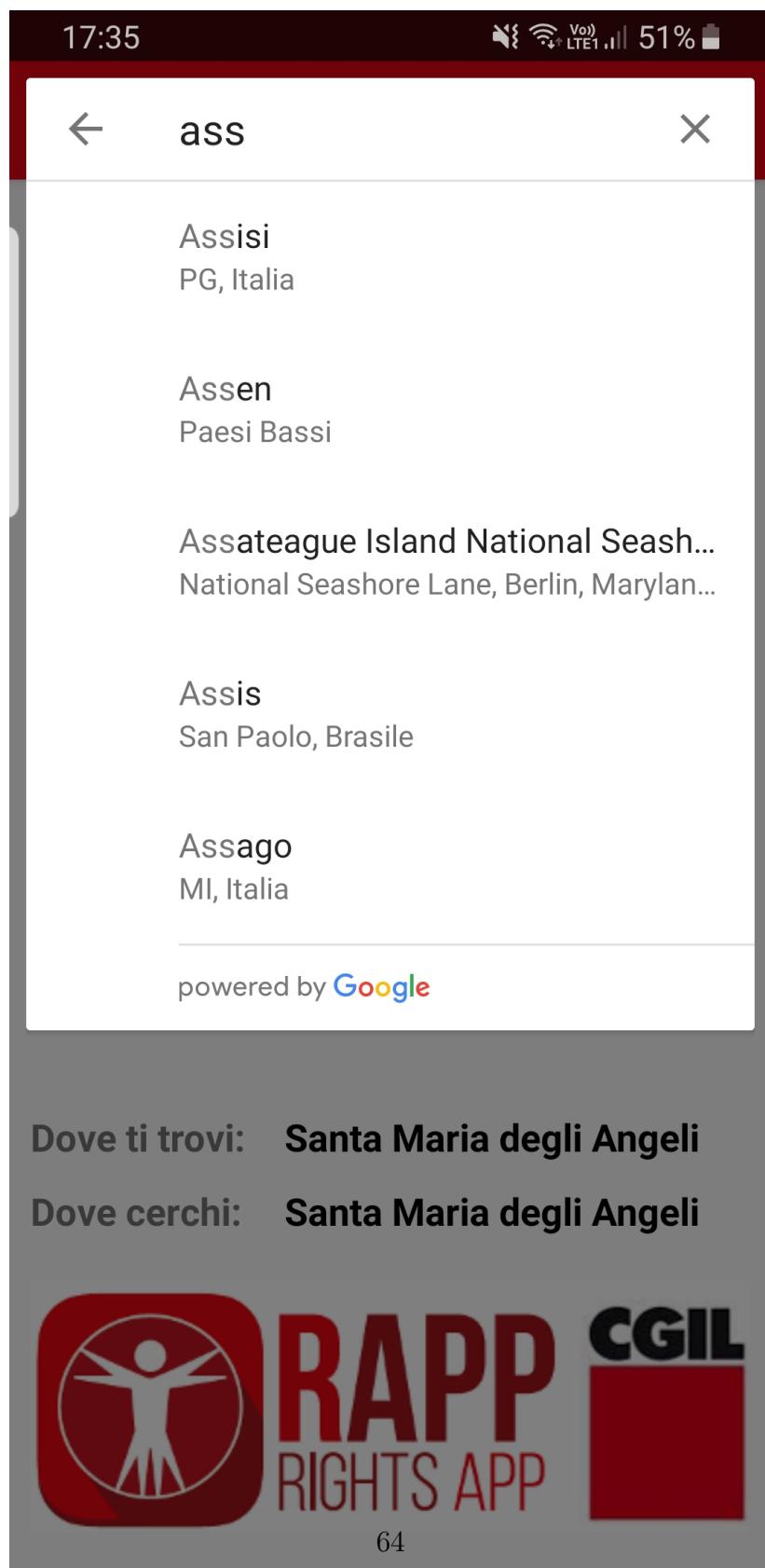


Figura 3.4: Cambio posizione nella Homepage.

Descrizione e Funzionamento applicazione

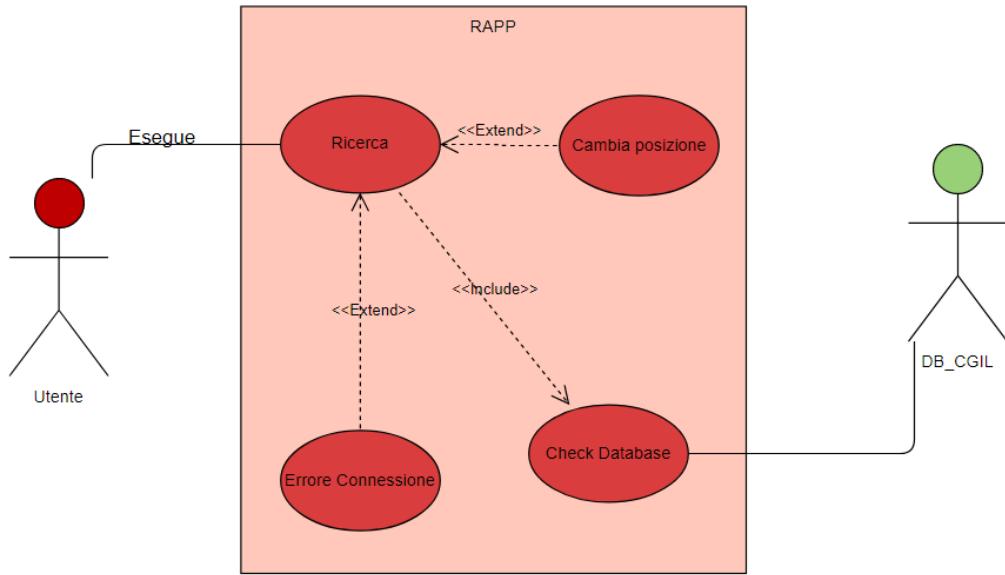


Figura 3.5: Diagramma caso d’uso Homepage

presente un bottone che permette di visualizzare ulteriori aziende certificate, mentre nella parte superiore risiede una freccia che permette di tornare alla Homepage. La Figura 3.6 mostra la seconda pagina dell’applicazione.

Il caso d’uso in Figura 3.7 mostra il funzionamento della seconda pagina di Rhightsapp. In particolare viene mostrata un’eccezione, che si verifica quando non vi sono aziende certificate nella zona desiderata. La Figura 3.8 mostra il diagramma relativo al caso d’uso della seconda pagina.

3. In questa ultima pagina vengono visualizzate le aziende certificate che corrispondono alla ricerca effettuata precedentemente, come mostrato in Figura 3.9.

Descrizione e Funzionamento applicazione

The screenshot shows a mobile application interface. At the top, there is a red header bar with the time "18:05" on the left and signal strength, battery level (48%), and other icons on the right. Below the header, the word "Ricerca" (Search) is displayed in white on a red background. The main content area has a white background and features three restaurant entries, each enclosed in a light gray box.

Ristoranti - Santa Maria degli Angeli

Ristorante Valerie
Via della Repubblica, 8,
06081 Santa Maria degli
Angeli PG, Italy
valutazione: 4.6 su 5
aperto adesso: chiuso

**Ristorante Pizzera
San Francesco**
Viale Patrono d'Italia, 22/
a, 06081 Santa Maria degli
Angeli, Assisi PG, Italy
valutazione: 4.4 su 5
aperto adesso: chiuso

**La Pregiutteria Casa
Norcia**
Via Alcide De Gasperi, 1,
06081 Santa Maria degli
Angeli PG, Italy
valutazione: 4.4 su 5
aperto adesso: chiuso

**Scopri altri Ristoranti
certificati nella zona di Santa
Maria degli Angeli**

SCOPRI

66

Figura 3.6: Seconda pagina applicazione.

Descrizione e Funzionamento applicazione

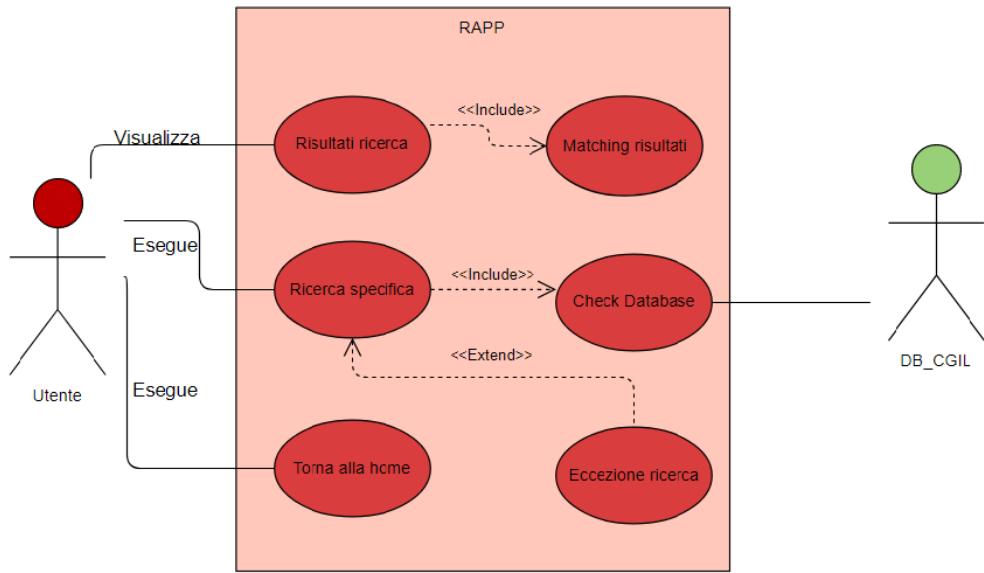


Figura 3.7: Diagramma caso d’uso seconda pagina

Questo è stato fatto per permettere di mostrare eventuali aziende che non sono state riportate dalla ricerca effettuata da Google. Infatti la ricerca restituisce fino ad un massimo di venti aziende, dunque è possibile che aziende certificate non ottengano la giusta visibilità.

Infine nella parte inferiore della schermata è presente uno spinner, il quale mostra le aziende certificate e ne permette la loro segnalazione, mentre nella parte superiore una freccia che permette di tornare alla Homepage. La Figura 3.10 mostra la pagina per ls segnalazione di un’azienda.

Il caso d’uso in Figura 3.11 mostra il funzionamento della terza pagina di Rhightsapp. In particolare viene mostrata un’eccezione, che si verifica quando si prova a effettuare più segnalazioni consecutive. La

Descrizione e Funzionamento applicazione

18:06 VoIP LTE1 48%

← Ricerca

Ristoranti - Perugia

★ **Hotel la Rosetta**
Piazza Italia, 19, 06121
Perugia PG, Italy

valutazione: 4 su 5
aperto adesso: aperto

★ **Osteria a Priori**
Via dei Priori, 39, 06123
Perugia PG, Italy

valutazione: 4.5 su 5
aperto adesso: chiuso

★ **Ristorante del Sole**
Via della Rupe, 1, 06121
Perugia PG, Italy

valutazione: 4.2 su 5
aperto adesso: chiuso

★ **La Taverna**
non è stato trovato nessun ristorante nella zona di Perugia

Scopri altri Ristoranti certificati nella zona di Perugia

68

SCOPRI

Figura 3.8: Eccezione seconda pagina

Descrizione e Funzionamento applicazione

The screenshot shows a mobile application interface with a red header bar. The top left displays the time (18:55) and the top right shows signal strength, battery level (60%), and a battery icon. The header bar contains a back arrow and the text "DB RApp". Below the header, there are four cards, each representing a different establishment:

- Umami beer**: Located at Via Los Angeles, 145, 06081 Santa Maria degli Angeli PG, Italy. The card features a photograph of several dark beer bottles.
- Testone**: Located at Via Sandro Pertini, 3, 06081 Santa Maria degli Angeli PG, Italy. The card features a photograph of a restaurant interior with wooden tables and chairs.
- Papavero Street Gourmet**: Located at Via Gianmaria Santarelli, 13, 06083 Santa Maria degli Angeli PG, Italy. The card features a photograph of the exterior of the restaurant.
- Momò G.A.P.**: Located at Via S. Pio X, 2/4, 06081 Santa Maria degli Angeli. The card features a photograph of the interior of the restaurant.

At the bottom of the screen, there is a call-to-action button labeled "SEGNALA" and the number 69.

Figura 3.9: Terza pagina applicazione.

Descrizione e Funzionamento applicazione

The screenshot shows a mobile application interface with a red header bar. The top left of the header shows the time (18:56) and the top right shows battery level (61%) and signal strength. The header text "DB RApp" is centered, with a back arrow icon to its left.

The main content area displays four business entries, each in a white card:

- Umami beer**: Located at Via Los Angeles, 145, 06081 Santa Maria degli Angeli PG, Italy. It features a photograph of several beer bottles.
- Testone**: Located at Via Sandro Pertini, 3, 06081 Santa Maria degli Angeli PG, Italy. It features a photograph of a restaurant interior with wooden tables and chairs.
- Papavero Street Gourmet**: Located at Via Gianmaria Santarelli, 13, 06083 Santa Maria degli Angeli PG, Italy. It features a photograph of a restaurant entrance with outdoor seating.
- Momò G.A.P.**: Located at Via S. Pio X, 2/4, 06081 Santa Maria degli Angeli. It features a photograph of a restaurant interior with a curved ceiling.

At the bottom, there is a call-to-action section with the text "La tua opinione conta!" and a button labeled "segnalazione riuscita". Below this, the name "Umami beer" is listed next to a downward arrow, and a red button labeled "SEGNALA" is shown.

Figura 3.10: Segnalazione azienda.

Descrizione e Funzionamento applicazione

Figura 3.12 mostra l'eccezione che si verifica nella terza pagina.

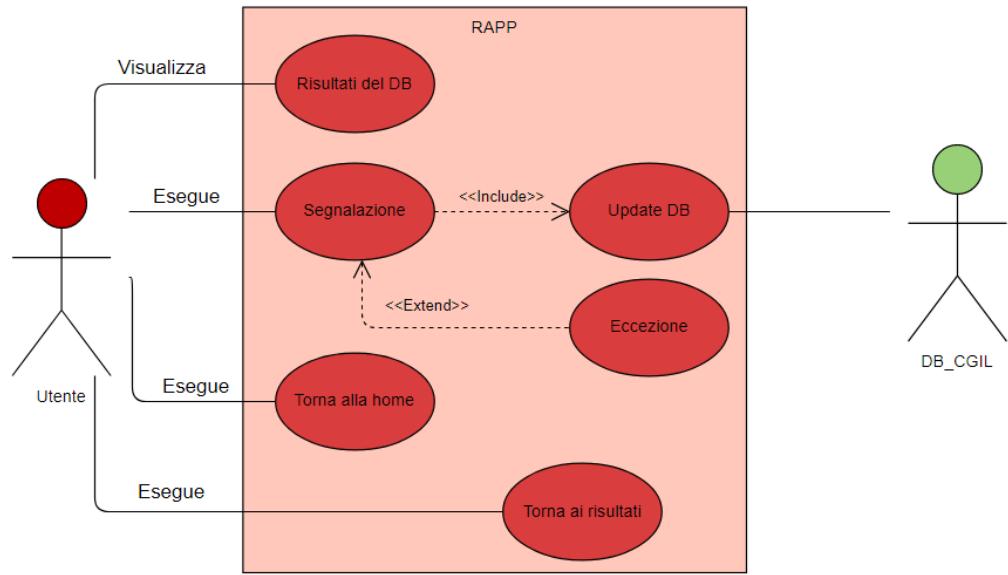


Figura 3.11: Diagramma caso d'uso terza pagina

Descrizione e Funzionamento applicazione

The screenshot shows a mobile application interface with a red header bar. The top left corner displays the time '18:56'. On the right side of the header are icons for signal strength, battery level at 61%, and a power symbol. Below the header, the title 'DB RApp' is centered above a back arrow icon.

The main content area displays four establishment cards, each featuring a yellow star icon followed by the establishment's name and address:

- Umami beer**
Via Los Angeles, 145,
06081 Santa Maria degli
Angeli PG, Italy
- Testone**
Via Sandro Pertini, 3,
06081 Santa Maria degli
Angeli PG, Italy
- Papavero Street
Gourmet**
Via Gianmaria Santarelli,
13, 06083 Santa Maria
degli Angeli PG, Italy
- Momò G.A.P.**
Via S. Pio X, 2/4, 06081
Santa Maria degli Angeli

Below these cards, a dark grey rounded rectangle contains the text 'La tua opinione conta!' and 'Hai già effettuato una segnalazione!'. To the left of this message, there is a small image of a restaurant interior.

At the bottom left, the name 'Umami beer' is displayed next to a downward-pointing arrow. At the bottom right, a large red button with the white text 'SEGNALA' is visible. The page number '72' is located at the very bottom center.

Figura 3.12: Eccezione terza pagina

Conclusioni

In questa tesi sono state presentate le attività svolte per lo sviluppo dell'applicazione Android della CGIL.

In sintesi, dal problema generale - ottenere un'applicazione fruibile - sono stati ricavati dei sotto problemi, qui eleganti, svolti e sviluppati in maniera gerarchica per il conseguimento e la risoluzione del problema padre:

1. Studio di fattibilità e analisi dei requisiti.
2. Suddivisione del lavoro.
3. Analisi dell'approccio da seguire per l'evoluzione dell'applicazione.
4. Geolocalizzazione del dispositivo.
5. Recupero informazioni tramite Google.
6. Connessione al Database.

Il punto 5 ha portato allo sviluppo di due applicazioni simili che recuperassero le informazioni tramite Google con modalità differenti.

La prima riguarda l'utilizzo della già citata libreria Jsoup, ben funzionante e totalmente gratuita. Tuttavia questo metodo di lavoro, cioè leggere

l'HTML fornito da Google, potrebbe non essere la soluzione migliore, per due motivi:

- Il primo è che il lavoro svolto risulta essere lungo e complesso.
- Il secondo è che non esiste garanzia che tutto ciò funzioni, perché se in futuro Google modifichi un tag HTML nella pagine delle ricerche, la nostra applicazione potrebbe immediatamente smettere di funzionare.

Dunque la seconda versione dell'applicazione vede utilizzare le [API di Google](#), che permettono in modo semplice ed efficace il raggiungimento del nostro obiettivo, a costo di una spesa annuale.

Come conclusioni di questa tesi è possibile affermare di aver raggiunto gli obiettivi proposti, ovvero la possibilità di realizzare un'applicazione Android in grado di valorizzare le aziende che fanno della qualità del lavoro un fattore imprescindibile.

Gli sviluppi futuri possibili su questa tesi possono essere:

- Possibilità di mostrare il percorso per raggiungere un'azienda.
- Possibilità di scrivere una recensione.
- Possibilità di contattare le aziende.
- Funzionalità per effettuare prenotazioni.
- Sviluppare un eventuale porting per iOS.

Ringraziamenti

Arrivato alla fine di questo percorso di studi mi sento di ringraziare alcune persone, innanzitutto il Prof. Gervasi e il Dott. Perri per avermi aiutato in questi ultimi mesi e per avermi dato la possibilità di lavorare insieme a loro.

Ringrazio la mia famiglia per non avermi mai fatto mancare nulla, in particolar modo l'affetto. Li ringrazio perché mi hanno sempre aiutato, in tutto, mi hanno sostenuto e mi hanno dato la possibilità di arrivare dove sono arrivato.

È solamente grazie a loro che oggi sono quello che sono.

Ringrazio la mia ragazza, Natalie che nei momenti di sconforto e di difficoltà era sempre li vicino a me ad ascoltarmi ed a sostenermi. Infine un grazie va ai miei amici che sono e saranno sempre la mia seconda famiglia.

Bibliografia

- [1] Iggy Krajci and Darren Cummings. History and evolution of the android os. In *Android on x86*, pages 1–8. Springer, 2013.
- [2] Peter Curwen. Has the smartphone market hit the rocks? a regular column on the information industries. *Digital Policy, Regulation and Governance*, 21(2):193–195, 2019.
- [3] Noor Ali-Hasan and Yuan Hang Li. Behind the scenes of researching android 9 pie’s system navigation. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*, page 80. ACM, 2019.
- [4] Android Studio. Android studio. *The Official IDE for Android*, 2017.
- [5] Apache Friends. Xampp apache+ mariadb+ php+ perl. *Apache Friends*, 2017.
- [6] Manav Singhal and Anupam Shukla. Implementation of location based services in android using gps and web services. *International Journal of Computer Science Issues (IJCSI)*, 9(1):237, 2012.

Bibliografia

- [7] Jonathan Hedley. jsoup: Java html parser, 2019. [Online; in data 3-settembre-2019].
- [8] Google. Introducing the places api, 2019. [Online; in data 20-settembre-2019].
- [9] Charlie Collins, Michael Galpin, and Matthias Käppler. *Android in practice*. Manning Publications Co., 2011.