Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica e dell'Automazione

Tesi di Laurea

Progettazione e implementazione di un'app Android a supporto di uno stabilimento balneare nell'era del Covid-19

Design and implementation of an Android app to support a beach resort in the Covid-19 era

Relatore

Prof. Domenico Ursino

Candidato

Andrea Pinciaroli

Anno accademico 2019-2020

Indice

Elenco delle figure

Elenco dei listati

Introduzione

Il primo smartphone, chiamato Simon, fu progettato dalla IBM nel 1992. Oltre alle normali funzioni di un telefono incorporava calendario, rubrica, orologio, posta elettronica, ed altre ancora. In seguito, lo sviluppo degli smartphone è stato strettamente legato all'evoluzione degli standard di telefonia mobile cellulare, dall'UMTS all'LTE. Si è assistito, inoltre, ad un enorme sviluppo hardware che ha permesso di ottenere dispositivi dalle prestazioni molto elevate a dimensioni contenute. Tutto ciò ha portato alla nascita degli smartphone che, nello scenario odierno, dispongono di un'app per qualsiasi cosa. In futuro si assisterà sicuramente ad una diffusione sempre maggiore di tali dispositivi che permetterà ad essi di svolgere operazioni sempre più complesse con dimensioni che diventeranno sempre più piccole.

La situazione epidemica scaturita dal COVID-19 ha portato i governi ad adottare misure sempre più restrittive per contenere la diffusione del contagio, tra cui il distanziamento sociale e la quarantena. In questo scenario, gli smartphone, e quindi, le app, sono stati tra i mezzi più utilizzati per mantenere le comunicazioni. In molti si sono adoperati per sviluppare nuove applicazioni per permettere ai proprietari delle attività che hanno risentito maggiormente della pandemia (ristoratori, strutture ricettive turistiche, etc) di continuare a lavorare.

La presente tesi si colloca in tale contesto. Infatti, si è voluto realizzare una applicazione per la gestione di uno stabilimento balneare, in particolare, incentrata sulla possibilità di effettuare la prenotazione di un ombrellone direttamente dallo smartphone. Tutti coloro che lavorano a stretto contatto con il pubblico si sono dovuti reinventare in modo da poter continuare le attività garantendo la propria sicurezza e quella dei clienti. Per questo nasce MyBeachResort, un'applicazione che permette allo stabilimento balneare di ridurre la possibilità di assembramenti fornendo al cliente tutte le informazioni necessarie direttamente da remoto.

L'app presentata sarà composta da diverse parti, in modo da offrire all'utente una panoramica dello stabilimento balenare e dei servizi offerti. Essa dovrà fornire all'utente le informazioni di carattere generale, tra cui le foto dello stabilimento, la posizione, il menù del giorno, il WiFi, il parcheggio, etc. Una sezione sarà dedicata alla creazione di un account personale in modo da poter accedere alle funzionalità di prenotazione e inserimento di una recensione. L'utente, infatti, potrà effettuare una prenotazione giornaliera di un ombrellone nell'apposita finestra dell'applicazione e, una volta trascorso il soggiorno, avrà l'opportunità di lasciare una recensione

riguardo l'esperienza vissuta. L'applicazione, sebbene all'apparenza risulti molto semplice, è caratterizzata da una complessità implementativa notevole; infatti, essa deve gestire una grande quantità di dati, sia per quanto riguarda la registrazione di diversi utenti sia per la gestione delle prenotazioni ed il salvataggio delle recensioni.

Riassumendo, l'applicazione ha un aspetto semplice ed intuitivo, adatta ad ogni tipo di utente, ma al proprio interno gestisce molte problematiche. Inoltre, essa, deve essere dotata di una grafica accattivante ed intuitiva, capace di attrarre l'utente senza però rendere complicato l'utilizzo delle principali funzionalità messe a disposizione.

La presente tesi è strutturata come di seguito specificato:

- Nel Capitolo 1, dopo aver presentato il contesto di riferimento ed, in particolare, la situazione Italiana, si illustrano le norme per la regolarizzazione degli stabilimenti e, infine, si descrive l'utilizzo della tecnologia per far fronte alle problematiche create dalla pandemia.
- Nel Capitolo 2 si parlerà di Android, della sua storia, del suo sviluppo sul mercato, della sua architettura e degli strumenti messi a disposizione per gli sviluppatori.
- Il Capitolo 3 verterà sull'analisi dei requisiti e sulla progettazione. Si parlerà, in particolare, dei requisiti funzionali e non funzionali, per concentrarsi poi sulla fase di progettazione, in cui vengono presentati i diagrammi UML, la mappa dell'applicazione ed i mockup, per dare un'idea di come sarà l'app.
- Nel Capitolo 4 verranno mostrati l'implementazione del progetto e, di seguito, il manuale utente. Si parlerà, dapprima, di come è strutturato il progetto e saranno, quindi, mostrati i listati con la relativa spiegazione. Si passerà, poi, al manuale utente dove, verranno mostrati tutti i passaggi che un utente deve effettuare per poter utilizzare le funzionalità offerte dall'applicazione.
- Nel Capitolo 5 verrà presentata una discussione in merito al lavoro svolto, individuando le nozioni apprese durante la realizzazione del progetto e illustrando la SWOT Analysis, così da poter comprendere i punti di forza e di debolezza dell'app realizzata.
- Il Capitolo 6 espone le conclusioni riguardo al progetto realizzato, indicando possibili miglioramenti e sviluppi futuri.

1

Descrizione del contesto di riferimento

Dalle prime settimane di gennaio 2020, a partire dalla provincia di Wuhan in Cina, sono stati individuati dei soggetti affetti da una particolare polmonite causata da un nuovo coronavirus designato poi come SARS-CoV-2. In un primo momento ci sono state difficoltà nel capire come il virus si trasmettesse e sulla patogenicità, ma con il passare delle settimane il numero dei casi aumentò in maniera esponenziale non solo in Cina ma a livello globale. La malattia è stata poi riconoscita con il nome di COVID-19 e l'11 marzo 2020 l'OMS (Organizzazione Mondiale della Sanità) dichiara la pandemia.

1.1 La situazione in Italia

I primi segni della malattia in Italia si manifestarono verso la fine di Gennaio 2020 quando due turisti cinesi risultarono positivi al COVID-19. Da quel momento il governo italiano dichiarò lo stato di emergenza e la sospensione di tutti i voli provenienti dalla Cina. Purtroppo, verso la fine di febbraio, ci fu uno scoppio epidemico nel Lodigiano che fu la causa delle prime cosidette "zone rosse" ovvero aree metropolitane in cui ci fu l'obbligo della quarantena per la popolazione, la chiusura degli edifici pubblici e di ogni attività e servizi al di fuori delle primarie necessità.

1.1.1 Le misure di contenimento adottate

Con il passare dei giorni la situazione non migliorò, i numeri del contagio continuarono a salire e ciò obbligò il governo italiano ad estendere le misure di contenimento a tutta la regione Lombardia e ad altri comuni del Veneto.

Alla luce di questi eventi tra le persone si scatenò una sorta di panico generale, furono presi d'assalto i supermercati e ci fu una vera e propria "fuga" dal Nord verso il Sud.

Il 4 marzo è stata disposta la chiusura di tutte le scuole e università di ogni ordine e grado a livello nazionale, il 9 marzo tutti gli eventi sportivi sono stati completamente sospesi. L'11 marzo il presidente del consiglio Giuseppe Conte firmò il "dpcm" per il quale furono estese le misure di contenimento su tutto il

territorio nazionale; l'Italia entrava così nella "Fase 1" con l'obiettivo di ridurre la circolazione del virus in modo da evitare il sovraccarico e il collasso del sistema sanitario nazionale (Figura 1.1). A tal proposito furono dettate le norme igieniche di base come l'obbligo della mascherina in pubblico, il distanziamento sociale e le raccomandazioni di lavarsi bene le mani e evitare di toccarsi il volto.

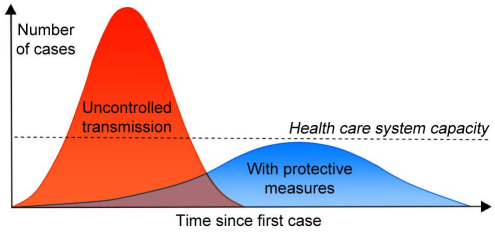

Figura 1.1. Appiattimento della curva epidemiologica.

Tra le misure adottate, citiamo l'ordinanza del 22 marzo 2020, firmata congiuntamente dal Ministro della Salute e dal Ministro dell'Interno, che vietava a tutte le persone fisiche di trasferirsi o spostarsi con mezzi di trasporto pubblici o privati in un comune diverso da quello in cui si trovavano, salvo che per comprovate esigenze lavorative, di assoluta urgenza ovvero per motivi di salute.

Il Governo ha poi emanato con il Dpcm 22 marzo 2020 nuove ulteriori misure in materia di contenimento e gestione dell'emergenza epidemiologica da COVID-19, applicabili sull'intero territorio nazionale. Il provvedimento prevedeva la chiusura delle attività produttive non essenziali o strategiche. Restavano aperti alimentari, farmacie, negozi di generi di prima necessità e i servizi essenziali. Le stesse disposizioni si applicarono, cumulativamente al Dpcm 11 marzo 2020, nonché a quelle previste dall'ordinanza del Ministro della Salute del 20 marzo 2020 i cui termini di efficacia, già fissati al 25 marzo 2020, sono state entrambi prorogati al 3 aprile 2020.

1.1.2 La fase due e la ripartenza economica

Il 18 maggio è stato il giorno che ha segnato la fine della "Fase 1" e ha dato l'inizio al periodo di convivenza con il virus denominato "Fase 2". Le misure di contenimento sono state allentate e pian piano le industrie e le attività commerciali si sono adoperate per ricominciare a lavorare. Le difficoltà per questa riapertura non sono state poche dovute al fatto che ovviamente è obbligatorio riuscire a garantire la salute dei lavoratori e dei clienti e proprio per questo sono state ridefinite completamente le modalità di lavoro in modo tale da rispettare il protocollo sanitario. Il governo per agevolare questa ripresa ha emanato un maxi-decreto chiamato "Decreto Rilancio" costituito da una serie di indennizzi, agevolazioni e incentivi per famiglie ed imprese a sostegno del reddito.

Dal 5 giugno 2020 la giunta della regione Marche ha approvato 50 misure facenti parte di una manovra straordinaria chiamata "Piattaforma 210" che stanzia un ulteriore importo di 210 milioni a supporto di attività commerciali e privati della regione.

1.2 Norme per la regolarizzazione degli stabilimenti balneari

Uno dei settori che più duramente è stato colpito dal COVID-19 è, sicuramente, il turismo, che corrisponde, per il nostro paese, a quasi il 13% del PIL. Di questa categoria fanno parte coloro che offrono servizi per la balneazione il cui lavoro è stato soggetto a documenti tecnici realizzati in collaborazione tra INAIL e Istituto Superiore della Sanità in modo tale da garantire la ripresa delle attività e la tutela della salute.

In riferimento alle linee guida nazionali, il 15 maggio 2020 la regione Marche ha pubblicato il decreto con le misure da adottare per gli stabilimenti balneari della regione. Tale decreto prevede le seguenti attività:

- *Formazione e informazione del personale.*
  L'impresa titolare dello stabilimento balneare provvede a formare ed informare il proprio personale tramite momenti formativi interni che includano le opportune linee guida e le procedure aziendali organizzative interne per la prevenzione della diffusione del virus responsabile del COVID-19.
  Ogni membro del personale, sia dipendente della struttura che dipendente di ditte terze operanti nella struttura, dovrà rispettare rigorosamente le misure indicate nelle linee guida.
  Tutti i dipendenti dell'azienda e i collaboratori devono essere dotati di un tesserino/elemento di riconoscimento (ad esempio maglietta staff o altro) esposto e visibile, in modo che i clienti possano avere punti di riferimento immediatamente visibili.
- *Screening del test del personale e dotazioni.*
  Ai sensi del DPCM 26/4/2020 Allegato 6 punto 2, il titolare dello stabilimento balneare può disporre in loco, verso tutti i lavoratori che operano all'interno dell' azienda, la misurazione della temperatura corporea prima di iniziare il turno lavorativo. In caso di febbre (superiore e 37.5° C), tosse o difficoltà respiratoria il lavoratore non potrà iniziare l'attività lavorativa e dovranno contattare immediatamente il proprio medico curante e seguire le sue indicazioni.
  Il personale deve essere dotato, da parte dei gestori di Dispositivi di Protezione Individuale (nel seguito, DPI) adeguati (mascherine, guanti, disinfettante, etc.) ed è obbligato all'adozione di DPI in caso di contatti ravvicinati con i bagnanti e attività a rischio (ad esempio contatto con rifiuti potenzialmente infetti, condizioni di formazione di aerosol durante la disinfezione).
- *Comunicazione.*
  È necessario predisporre strumenti di comunicazione finalizzati ad informare i clienti sulle disposizioni da rispettare all'interno dello stabilimento balneare.
  Tra gli strumenti di comunicazione, è raccomandata l'affissione di documenti e poster in posizione ben visibile, in diverse lingue, indicanti i punti salienti (distanze sociali, lavaggio delle mani, igiene respiratoria, altri comportamenti da tenere all'interno dello stabilimento e nei vari ambienti, obbligo di rimanere al proprio domicilio in presenza di febbre di oltre 37.5° o altri sintomi influenzali e di chiamare il proprio medico di famiglia, sia per i clienti che per il personale.

Oltre alle comunicazioni sopra descritte, il cliente verrà informato al fine di poter scaricare e utilizzare la APP *Immuni* anche usufruendo del servizio di *WiFi Regionale* gratuito.

- *Accesso allo stabilimento.*
  È fondamentale che gli accessi allo stabilimento avvengano in modo ordinato al fine di prevenire assembramenti.
  Gli utenti debbono indossare la mascherina al momento dell'arrivo, fino al raggiungimento della postazione assegnata, e analogamente all'uscita dallo stabilimento.
  La regolamentazione degli accessi e degli spostamenti sulle spiagge e negli arenili deve essere predisposta e attuata anche attraverso percorsi dedicati, prevedendo, ove necessario, la segnatura della distanza di 1 metro sulle parti comuni e i camminamenti con maggior passaggio e afflusso di clienti.
  La disposizione delle attrezzature all'interno dello stabilimento deve garantire in ogni circostanza il distanziamento sociale di almeno 1 mt.
  Il personale addetto alla reception e all'accompagnamento dei clienti viene dotato di dispositivi di protezione che limitino il contatto con droplets e aerosol e inviterà i clienti in arrivo ad informarsi tramite il materiale esposto e ad osservare tutte le disposizioni indicate all'interno dello stabilimento per prevenire e controllare i rischi.
  Essendo preferibile evitare la circolazione di monete e banconote si consiglia di incentivare i clienti all'utilizzo della moneta elettronica, possibilmente mediante card contactless o mediante pagamento anticipato con bonifico.
- *La distanza tra gli ombrelloni.*
  Al fine di garantire il corretto distanziamento sociale all'interno dello stabilimento balneare, le distanze minime tra gli ombrelloni posizionati sulla spiaggia in deroga all'attuale regolamento Regionale n.2/2004 art.4 lett.f) sono modificate come segue:
  La distanza tra ombrelloni della stessa fila e tra file deve in ogni caso garantire una superficie minima ad ombrellone di mq 10,5 a paletto.
  È consentita la possibilità di posizionare gli ombrelloni con distanza minima di 4,60 metri nella stessa fila e 3 metri tra le file per permettere, in caso di superamento delle condizioni critiche epidemiologiche, e successivamente ad eventuale specifico provvedimento inserimento di ulteriori ombrelloni a distanza minima di m.2.30 nella stessa fila.
  In caso di utilizzo di altri sistemi di ombreggio andranno, comunque, garantite aree di distanziamento equivalenti a quelle garantite dal posizionamento degli ombrelloni; la misura minima di 1 metro va garantita come distanza minima tra la proiezione di un sistema di ombreggio e l'altro.
  Le attrezzature complementari assegnate in dotazione all'ombrellone, quali sdraio, seggiola, lettino etc. potranno essere fornite in quantità limitata atta a garantire il distanziamento con le attrezzature dell'ombrellone contiguo di almeno 1,50 metri.
  Sotto gli ombrelloni, o altri sistemi di ombreggio, è fatto obbligo di osservare una distanza di sicurezza interpersonale di almeno un metro. L'obbligo è derogato per i soli membri del medesimo nucleo familiare, ovvero conviventi (potrà essere richiesta un'autocertificazione).

- *Distanziamento dei lettini in spiaggia.*
  Fermo restando l'obbligo di rispetto della fascia di mt. 5 dal bagnasciuga, i lettini posizionati singolarmente sulla spiaggia devono essere collocati orizzontalmente a distanza di almeno mt.2 l'uno dall'altro. L'obbligo è derogato per i soli membri del medesimo nucleo familiare, ovvero conviventi (potrà essere richiesta un'autocertificazione).
- *Accesso all'area di balneazione.*
  L'attività di balneazione deve rispettare le regole relative al distanziamento sociale senza mai derogare alle distanze consentite.
  Il personale abilitato quale *bagnino di salvataggio* dovrà essere impiegato esclusivamente per osservare lo specchio acqueo di competenza, sia per sensibilizzare l'utenza sull'obbligo di garantire il distanziamento fisico che per vigilare sulla salvaguardia della vita umana in mare dei bagnanti. Le ordinarie procedure di salvataggio dovranno essere adeguate con tecniche di intervento che tengano conto dell'emergenza Covid-19.
  Anche nella fase di accesso al mare dovrà essere prevista, ove opportuno, una regolamentazione degli accessi in modo da mantenere sempre il distanziamento prescritto.
- *Pulizia e disinfezione.*
  É necessario garantire una pulizia periodica, almeno giornaliera, con i normali detergenti, delle varie superfici e arredi di cabine e aree comuni.
  É inoltre, fatto obbligo di provvedere alla pulizia con acqua e detergenti comuni e alla disinfezione. Per la decontaminazione, si raccomanda l'uso di ipoclorito di sodio 0,1% dopo pulizia e, per le superfici che possono essere danneggiate dall'ipoclorito di sodio, di etanolo al 70% (Circolare del Ministero della salute n.005443 del 22/02/2020). La pulizia riguarda le attrezzature in dotazione, quali sedie, sdraio e lettini; essa deve avvenire periodicamente comunque ad ogni cambio di cliente.
  Deve essere assicurata più volte durante la giornata una pulizia accurata e frequente dei servizi igienici comuni in relazione alla quantità di flusso di accesso e disinfezione a fine giornata. In ogni caso, per le misure specifiche si rimanda al Rapporto ISS-COVID-19 n. 19/2020.
  Per la fruizione di servizi igienici e docce va rispettato il distanziamento sociale di almeno 2,00 metri, a meno che non siano previste barriere separatorie fra le postazioni.
  È consigliata la limitazione dell'utilizzo di strutture (ad esempio cabine docce singole, spogliatoi) per le quali non sia possibile assicurare una disinfezione intermedia tra gli utilizzi promiscui.
  É inoltre, necessario assicurare la non promiscuità nell'uso di lettini, sdraie e altre attrezzature, con divieto di scambiare le attrezzature tra ombrellone e ombrellone.
  All'ingresso delle aree adibite a servizi igienici deve essere messa a disposizione dei clienti una dotazione di soluzioni idroalcoliche per l'igiene delle mani in modo da detergersi prima dell'utilizzo dei servizi e all'uscita, con l'utilizzo di appositi dispenser collocati in punti facilmente individuabili. Per le cabine è vietato l'uso promiscuo ad eccezione dei membri del medesimo nucleo familiare o per soggetti

che condividano la medesima unità abitativa o ricettiva, prevedendo un'adeguata igienizzazione fra un utente e il successivo.

- *Responsabilità del titolare dello stabilimento.*
  Il titolare dello stabilimento pone in essere tutte le condizioni per il rispetto delle regole e dei comportamenti prescritti dalle presenti linee guida senza, tuttavia, essere direttamente responsabile di eventuali condotte contrarie da parte dei singoli clienti.
- *Servizi bar e ristorazione.*
  I Servizi di Bar e di Ristorazione forniti nell'ambito dello stabilimento balneare devono svolgersi nel rispetto delle normative vigenti e, in particolare, secondo le linee guida e le disposizioni specifiche per la categoria.
  In caso di consumo di bevande o pasti sotto l'ombrellone/gazebo, dovranno essere osservate scrupolosamente le disposizioni relative alle distanze di sicurezza.
  Potrà essere organizzato un servizio di prenotazione bar o ristorante mediante dispositivi informatici e consegna diretta all'ombrellone.

1.2.1 La tecnologia a supporto delle attività

In merito a quanto detto finora è facile capire quanto sia stato importante l'utilizzo di smatphone e Personal Computer di cui al giorno d'oggi tutti siamo dotati. La tecnologia ha permesso di continuare a lavorare da casa tramite lo "smart working" e di mantenere le comunicazioni. In molti si sono messi all'opera per creare software e applicazioni; il *Center for Systems Science and Engineering* della Johns Hopkins University ha sviluppato una dashboard online per visualizzare e tenere traccia dei casi di Covid-19 nel mondo in tempo reale, segnalati su base giornaliera. La mappa mostra i nuovi casi, i morti confermati e i guariti. Per la prima volta da quando abbiamo accesso alla rete, è possibile analizzare lo scenario di un contagio in tempo reale. Inoltre, il set completo di dati è scaricabile in formato Google sheets.

Alcune imprese hanno deciso di stare vicino alle persone, offrendo importanti sconti sui propri servizi o, addirittura, regalandoli; è nato così il fenomeno della solidarietà digitale.

Ad esempio, Google ha reso gratuite tutte le funzionalità "advanced" di Hangouts, l'app per scambiarsi messaggi in tempo reale e videochiamarsi.

Le compagnie di telefonia italiane hanno offerto sconti speciali e Giga gratis per affrontare il periodo di quarantena e favorire il lavoro da remoto.

Agli inizi di marzo l'azienda *Isinnova*, dopo aver realizzato delle valvole in stampa 3D per i macchinari di terapia intensiva, è stata contattata dall'ex primario dell'Ospedale di Gardone Valtrompia (Brescia), il Dott. Renato Favero, per sviluppare un prototipo di respiratore partendo da una maschera da snorkeling prodotta da Decathlon.

In aggiunta sono state sviluppate app per le consegne a domicilio del cibo, app gestionali per l'organizzazione delle prenotazioni per attività di parruchieri , estetista, ristorazione e, proprio, i sopra citati stabilimenti balneari.

2

Android

In questo capitolo verrà presentata una panoramica su Android, il sistema operativo per dispositivi mobile più diffuso al mondo. Verranno presentati alcuni cenni riguardo la storia, le sue versioni e la sua architettura fino all'introduzione di alcuni strumenti per lo sviluppo delle app.

2.1 Introduzione ad Android

Android (Figura 2.1) è il principale sistema operativo per dispositivi mobili al mondo. Secondo le statistiche, infatti, è il motore di ben tre dispositivi su quattro. Gran parte del suo successo deriva dal fatto che è open-source; ciò significa che chiunque voglia utilizzare tale sistema può scaricare l'intero codice sorgente gratuitamente. La sua popolarità è dovuta, inoltre, alla possibilità di personalizzazione che le aziende produttrici di dispositivi mobili possono effettuare per differenziare i propri prodotti dai competitor sul mercato.

Figura 2.1. Logo di Android Inc.

2.1.1 Un pò di storia

Nel 2003 viene fondata Android Inc. da Andy Rubin, Rich Miner, Nick Sears e Chris White. Solo nel 2005, con l'acquisizione di Google per la cifra di 50 milioni di dollari, Android inizia a diventare un sistema operativo per dispositivi mobile.

La presentazione ufficiale avvenne il 5 novembre del 2007 dalla neonata OHA (Open Handset Alliance), un consorzio di aziende che comprendeva produttori di smatphone e operatori di telefonia mobile. Il primo dispositivo con sistema operativo Android lanciato sul mercato fu l'HTC Dream (Figura 2.2) il 22 ottobre del 2008. Da quel momento è iniziato un lunghissimo processo di aggiornamenti e migliorie che hanno portato Android al successo.

Nell'aprile dell'anno successivo HTC dichiarò di aver venduto oltre 1 milione di unità.

Figura 2.2. HTC Dream.

2.1.2 Versioni

Numerose sono le versioni di Android (Figura 2.3), e ne viene rilasciata una nuova ogni anno.

Le versioni di Android (a partire dalla 1.5) adottano un simpatico sistema di naming: si segue l'ordine alfabetico e, in base alla lettera, viene associato alla nuova versione un nome di un dessert. Questo fino all'ultima versione cioè Android 10 (Figura 2.4 per la quale si è abbandonato l'utilizzo dei nomi dei dolci. Le versioni che sono state finora rilasciate sono le seguenti:

- Android 1.0, API 1: 2008;
- Android 1.1, API 2: 2009;

- Android 1.5, Cupcake: 2009;
- Android 1.6, Donut: 2009;
- Android 2.0-2.1, Eclair: 2009;
- Android 2.2-2.2.3, Froyo: 2010;
- Android 2.3-2.3.7, Gingerbread: 2010;
- Android 3.0-3.2.6, Honeycomb: 2011;
- Android 4.0-4.0.4, Ice Cream Sandwich: 2011;
- Android 4.1-4.3.1, Jelly Bean: 2012;
- Android 4.4-4.4.4, KitKat: 2013 ;
- Android 5.0-5.1.1, Lollipop: 2014;
- Android 6.0-6.0.1, Marshmallow: 2015 ;
- Android 7.0-7.1.2, Nougat: 2016;
- Android 8.0-8.1, Oreo: 2017 ;
- Android 9.0, Pie: 2018 ;
- Android 10 : 2019

Figura 2.3. Loghi delle versioni di Android in ordine di uscita

Figura 2.4. Logo Android 10

In commercio esistono contemporaneamente diverse versioni (Figura 2.5), poichè un'app viene progetatta in modo tale da funzionare sulla versione in cui viene sviluppata e su tutte le successive (backward compatibility). Questo permette un maggiore utilizzo da parte di un numero elevato di utenti. Questo modo di procedere fa si che le versioni recenti sono dotate di maggiori funzionalità ma possono essere utilizzate solo da un numero ristretto di dispositivi; mentre le versioni più datate

possono essere utilizzate da un numero maggiore di utenti ma pian piano perdono di funzionalità.

La scelta della versione su cui basare un progetto dipende principalmente da quello che si vuole sviluppare. Per il lavoro di tesi si è optato per la versione 6.0 Marshmallow poichè, essendo una versione più datata, è utilizzabile sulla maggior parte dei dispositivi in commercio e risulta essere più facile anche il debugging dell'applicazione.

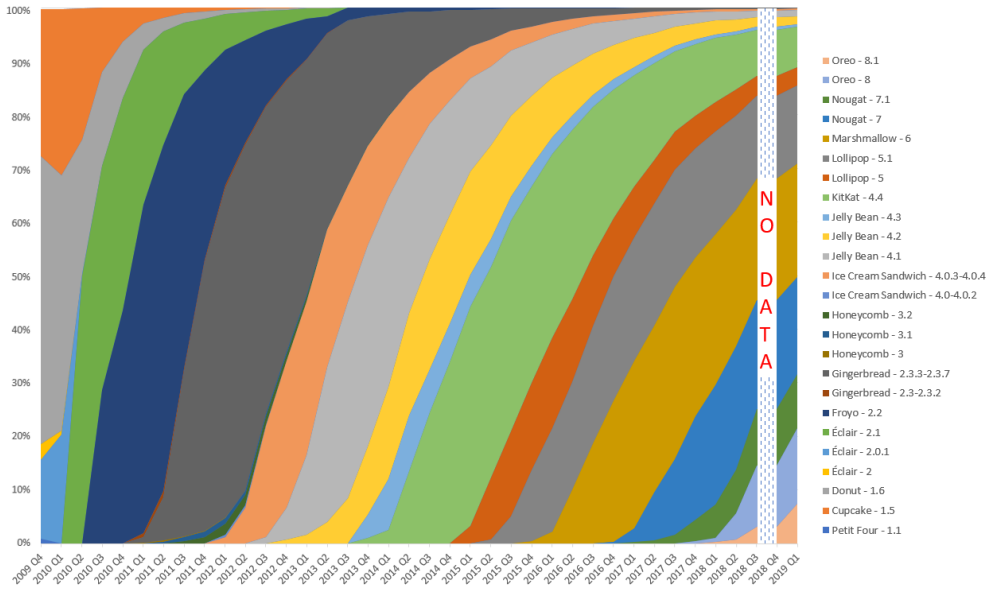

Figura 2.5. Utilizzo delle varie versioni di Android negli anni

2.2 Architettura di Android

Android comprende tutto lo stack degli strumenti necessari per la creazione di un'applicazione, tra cui un sistema operativo, un insieme di librerie native per le funzionalità core della piattaforma, un' implementazione della Virtual Machine (in seguito VM) e un insieme di librerie Java.

Tale architettura è un architettura a strati (Figura 2.6) dove i livelli inferiori offrono servizi ai livelli superiori permettendo un maggiore livello di astrazione.

Le sue componenti principali dell'architettura di Android sono le seguenti:

- *Kernel Linux*: alla base dell'architettura troviamo il kernel Linux che fornisce le funzioni di base del sistema. All'interno sono presenti i diversi driver per potersi interfacciare con l'hardware. La scelta del kernel Linux deriva dalla necessità di avere un sistema operativo affidabile e testato, che permetta ai produttori di definire i propri driver personalizzati.

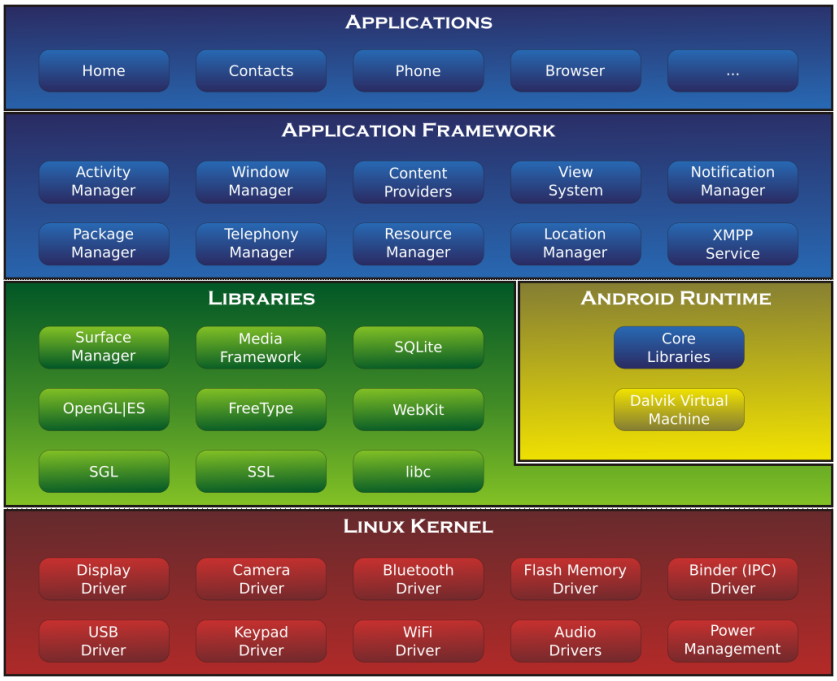

Figura 2.6. Architettura Android

- *Librerie Native*: sopra al kernel Linux si ha un livello formato da librerie native realizzate in C e C++ che rappresentano il core di Android. Le librerie sono necessarie per la gestione dei software, della grafica (Open GL ES), dei database (SQlite), del browser (WebKit), e della multimedialità (Media Framework).
- *Android Runtime*: l'esecuzione di un' applicazione Android è simile a ciò che avviene in ambiente Java, con la differenza che, in fase di compilazione serve necessariamente il file jar per la creazione di bytecode Java, mentre in esecuzione il dispositivo mette a disposizione la versione dex del runtime che viene eseguito dalla Dalvik Virtual Machine, (nel seguito DVM).
- *Application Framework*: si tratta di un insieme di API e componenti per l'esecuzione e la gestione di un'applicazione Android. Tale livello ha il compito di controllare il ciclo di vita di un'applicazione (Activity Manager), gestire il modo in cui le applicazioni possono recuperare informazioni relative ad altre applicazioni installate sul dispositivo (Package Manager), gestire le finestre delle diverse applicazioni (Window Manager), controllare la condivisione di informazioni tra i diversi processi (Content Provider), gestire la presenza di file di tipo diverso (Resource Manager), gestire la renderizzazione dei componenti e degli eventi (View System), etc.

2.3 Android per gli sviluppatori

L'ambiente di sviluppo integrato (Integrated Development Environment, IDE) più utilizzato per lo sviluppo di applicazioni Android è Android Studio (Figura 2.7), un IDE annunciato nel 2013 che si basa su IntelliJ di JetBrains. I primi sviluppatori Android che si sono formati su Eclipse hanno ben accettato l'arrivo di Android Studio sia per la sua intuitività ed efficienza sia perchè nasce appositamente per Android.

All'inizio di ogni nuovo progetto l'IDE offre diversi template per lo sviluppo dell'app e la configurazione è affidata ai file build Gradle. Quest'ultimo permette una configurazione molto flessibile e, soprattutto, la gestione delle dipendenze in stile Maven. Ciò da la possibilità, tramite inserimento di direttive, di recuperare librerie di terze parti direttamente in rete.

L'IDE è in continuo sviluppo e aggiornamento e, ovviamente, per un funzionamento ottimale sono richieste determinate componenti hardware e software.

Figura 2.7. Logo dell'IDE Android Studio

3

Analisi dei requisiti e progettazione

In questo capitolo verranno illustrate l'analisi dei requisiti, ovvero tutte le attività necessarie per identificare le diverse esigenze, e la progettazione. In particolare, verranno proposte una descrizione del progetto ed una rappresentazione del caso di studio tramite i diagrammi dei casi d'uso. Infine, per quanto riguarda la progettazione, saranno illustrati la mappa dell'applicazione e i mockup, per dare un'idea di come dovrà funzionare il sistema.

3.1 Descrizione del progetto

Il progetto consiste nell'implementare un'app che consenta di mostrare una panoramica a tutto tondo di uno stabilimento balneare e dei servizi che lo stesso può offrire; si tratta di una sorta di sistema gestionale. Lo scopo è quello di dare la possibilità all'utente di interfacciarsi a 360 gradi con lo stabilimento balneare ancor prima di entrare a contatto con il luogo fisico.

L'app, sarà quindi, composta da quattro sezioni principali:

- *Home*: pagina dedicata alle informazioni di carattere generale sullo stabilimento (ad esempio servizi offerti, meteo, immagini);
- *Prenotazioni*: pagina in cui l'utente può effettuare una prenotazione di un ombrellone;
- *Recensioni*: pagina in cui l'utente può lasciare una recensione sull'esperienza trascorsa nello stabilimento;
- *Profilo*: pagina tramite cui l'utente può creare un account.

3.2 Analisi dei requisiti

In questa sezione si farà un'analisi dei requisiti funzionali e non funzionali del progetto. I requisiti funzionali rappresentano l'insieme delle caratteristiche che dovranno essere implementate. Invece, i requisiti non funzionali indicano l'insieme dei vincoli realizzativi che l'applicazione dovrà rispettare.

Dopo l'analisi dei requisiti verranno mostrati i diagrammi dei casi d'uso, che descrivono le azioni che l'applicazione deve eseguire in seguito all'interazione da parte degli utenti esterni.

3.2.1 Descrizione dei requisiti

Requisiti funzionali

L'applicazione deve garantire le seguenti funzionalità fondamentali:

- *Creazione di un profilo*: l'utente deve avere la possibilità di creare un profilo in modo tale da poter accedere alle funzionalità di prenotazione ed inserimento di una recensione.
- *Effettuare una prenotazione*: l'utente registrato deve poter effettuare una prenotazione di un'ombrellone.
- *Scrivere una recensione*: l'utente registrato deve poter lasciare una recensione nell'apposita sezione.
- *Visualizzazione dei contenuti*: l'app deve permettere ad ogni utente la visualizzazione generale delle informazioni relative allo stabilimento.

Requisiti non funzionali

L'applicazione deve rispettare i seguenti vincoli:

- *Design intuitivo*: l'app necessita di un design intuitivo e accattivante in modo tale che l'utente possa muoversi con facilità nelle diverse sezioni.
- *Database*: i dati relativi alle prenotazioni, alle recensioni nonchè i dati personali di ogni utente devono essere salvati su di un apposito database implementato su Firebase.
- *Prenotazione solo per utente registrato*: la prenotazione di un ombrellone può essere effettuata solo da un utente registrato; l'utente senza profilo può solo visualizzare i contenuti senza apportare modifiche.
- *Recensione solo per utente registrato*: una recensione può essere scritta solo da un utente registrato.

Nella Figura 3.1 viene riassunto l'elenco dei requisiti funzionali e non funzionali che l'app dovrà rispettare.

Requisiti funzionali
+ Creazione di un profilo
+ Effettuare una prenotazione
+ Scrivere una recensione
+ Visualizzazione dei contenuti

Requisiti non funzionali
+ Design intuitivo
+ Database
+ Prenotazione solo per utente registrato
+ Recensione solo per utente registrato

Figura 3.1. Elenco dei requisiti funzionali e non funzionali

3.2.2 Diagramma dei casi d'uso

La fase successiva consiste nel trovare gli attori e i casi d'uso del progetto, cioè creare il diagramma dei casi d'uso (use case diagram).

Tale diagramma può essere definito come una rappresentazione dell'interazione dell'utente con i casi d'uso in cui esso è coinvolto. Tipicamente è il primo tipo di diagramma ad essere creato in un processo o ciclo di sviluppo per quanto riguarda l'analisi dei requisiti.

Un diagramma dei casi d'uso è formato, infatti, dai casi d'uso, che rappresentano le funzionalità che il sistema può offrire, e dagli attori, ovvero gli utenti che possono richiedere l'esecuzione dei casi d'uso.

Nella Figura 3.2 vengono mostrati i diagrammi dei casi d'uso relativi al progetto che si intende realizzare.

Nel caso del progetto sono presenti due attori, ovvero l'utente che utilizza l'app senza essere registrato (utente semplice) e l'utente registrato. I casi d'uso, invece, sono diversi e, a seconda dell'attore che si considera, possono essere eseguite più o meno funzionalità.

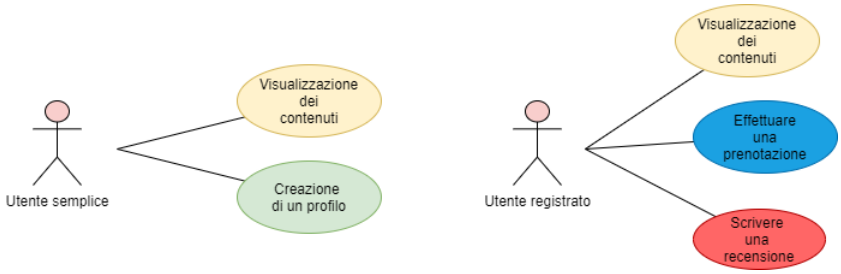

Figura 3.2. Diagramma dei casi d'uso relativo al progetto

3.3 Progettazione

La fase di progettazione consiste nel progettare le funzionalità software che l'applicazione deve offrire. In questa sezione verrà costruita la mappa dell'applicazione e verranno realizzati i mockup. In questo modo si vuole dare un'idea di come funzioni l'app e di che aspetto debba avere.

3.3.1 Mappa dell'applicazione

La mappa dell'applicazione è un modo semplice ed intuitivo per rappresentare ciò che è stato descritto nei casi d'uso. In essa sono descritti i passaggi che devono essere effettuati per compiere una determinata azione richiesta dall'attore.

Nel caso di progetto, la mappa dell'applicazione (Figura 3.3) illustra le diverse sezioni che compongono l'applicazione e quali sono le funzionalità che possono essere svolte a seconda dell'attore che si interfaccia.

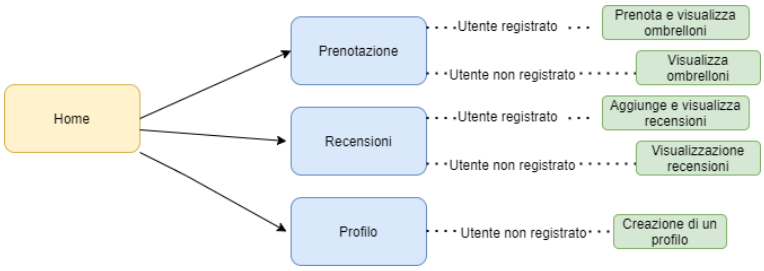

Figura 3.3. Mappa dell'applicazione relativa al progetto

3.3.2 Mockup

Un mockup è, in pratica, una simulazione della realtà che permette di illustrare i vari contenuti e l'aspetto grafico del progetto.

In base al grado di dettaglio si può avere un mockup di Livello 0, dove viene data un'idea molto approssimata dei contenuti del progetto, in pratica uno "schizzo"; nei mockup di Livello 1, vengono forniti ulteriori elementi dell'aspetto grafico ed, infine, i mockup di Livello 2 che sono una rappresentazione molto fedele di come poi si presenta il progetto finale.

Di seguito sono mostrati i mockup di Livello 0 e 1 per le diverse funzionalità dell'applicazione.

Home Page

La "Home Page" (Figure 3.4 e 3.5) è la prima schermata che viene vista all'apertura dell'app da parte dell'utente. Essa ha lo scopo di mostrare una panoramica generale sui servizi offerti dallo stabilimento.

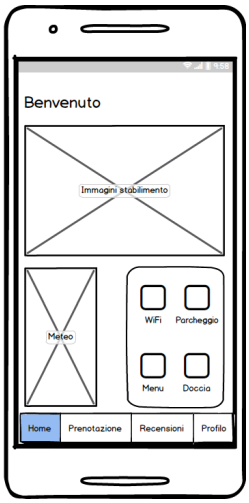

Figura 3.4. Mockup di Livello 0 riguardante la Home Page

Figura 3.5. Mockup di Livello 1 riguardante la Home Page

Prenotazione

Nella sezione "Prenotazione"(Figure 3.6 e 3.7) vengono mostrate la griglia degli ombrelloni e le loro disponibilità. Il rettangolo azzurro posizionato in basso sta ad indicare il mare (Figura 3.7) e, quindi, dove sono posizionati gli ombrelloni della prima fila. L'utente registrato può effettuare una prenotazione cliccando su di un ombrellone libero.

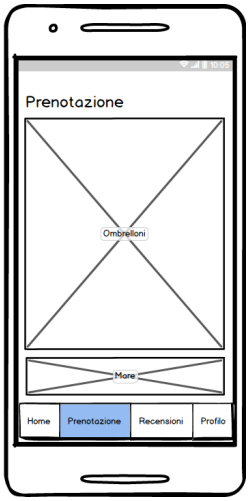

Figura 3.6. Mockup di Livello 0 riguardante la schermata delle prenotazioni

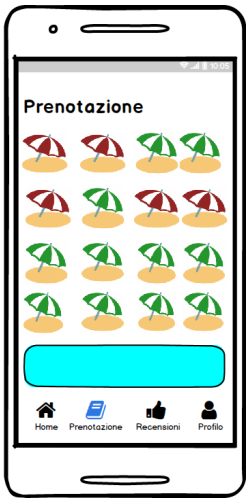

Figura 3.7. Mockup di Livello 1 riguardante la schermata delle prenotazioni

Conferma Prenotazione

Dopo aver cliccato su di un ombrellone disponibile, verrà mostrato all'utente una finestra di dialogo (Figure 3.8 e 3.9) in cui sarà chiesto di confermare la prenotazione, oppure di annullarla.

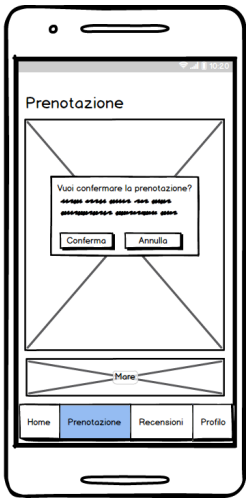

Figura 3.8. Mockup di Livello 0 riguardante la conferma della prenotazione

Figura 3.9. Mockup di Livello 1 riguardante la conferma della prenotazione

Recensioni

Nella sezione “Recensioni” (Figure 3.10 e 3.11) vengono mostrate tutte le recensioni lasciate dagli utenti nel corso del tempo. Tali recensioni sono disposte a formare una lista con scorrimento verticale;ciascuna recensione si trova in un apposito riquadro in cui troviamo la mail dell’utente, la valutazione dell’esperienza espressa tramite “RatingBar” (barra di valutazione) ed il testo.

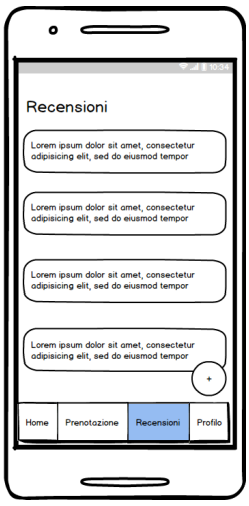

Figura 3.10. Mockup di Livello 0 riguardante la sezione delle recensioni

Figura 3.11. Mockup di Livello 1 riguardante la sezione delle recensioni

Inserimento di una recensione

L'utente registrato può aggiungere una recensione cliccando il pulsante di forma circolare posto in basso a destra. A seguito di tale attività si aprirà una finestra di dialogo (Figure 3.12 e 3.13) dove si deve indicare il numero di stelle e inserire il testo nell'apposito campo.

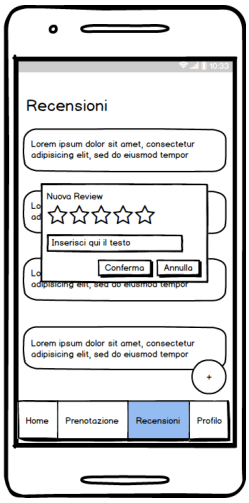

Figura 3.12. Mockup di Livello 0 riguardante l'inserimento di una recensione

Figura 3.13. Mockup di Livello 1 riguardante l'inserimento di una recensione

Profilo

L'ultima sezione è quella del "Profilo" (Figure 3.14 e 3.15) dove l'utente può autenticarsi attraverso l'apposito pulsante o, se non l'ha già fatto, può procedere alla registrazione e, quindi,creare il proprio account.

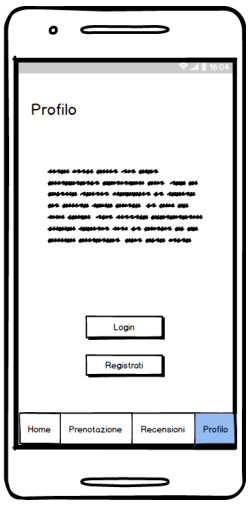

Figura 3.14. Mockup di Livello 0 riguardante la sezione Profilo

Figura 3.15. Mockup di Livello 1 riguardante la sezione Profilo

4

Implementazione e Manuale Utente

Questo capitolo tratta l'implementazione dell'app descritta nel capitolo precedente ed il manuale utente, ovvero verranno mostrate le linee guida per l'utilizzo dell'applicazione. In un primo momento sarà descritta la struttura del progetto; verranno, poi, mostrati gli script relativi alle varie parti del progetto; infine, si vedranno tutti i passaggi per utilizzare l'app facendo riferimento agli screen-shot dell'applicativo.

4.1 Struttura del progetto

Il progetto, sviluppato tramite l'IDE Android Studio, è strutturato come descritto nella Figura 4.1.

Le componenti di particolare interesse, dal punto di vista implementativo, sono le seguenti:

- `manifests`: directory che contiene il file `AndroidManifest.xml`. Questo contiene le caratteristiche basilari dell'app, necessarie per poter eseguire le diverse porzioni di codice.
- `com.example.progetto`: directory che contiene le classi `Note`, `MyDividerItem -Decoration`, `NotesAdapter`, `VerticalSpaceItemDecoration`, `ActivityOne`, `ActivityTwo`, `ActivityThree`, `MainActivity`, `RecyclerViewAdapter`, `Slide - Adapter`. Queste hanno lo scopo di implementare tutte le funzionalità descritte nel Capitolo 4.
- `drawable`: directory che contiene le immagini che verranno visualizzate dall'utente durante l'utilizzo dell'applicazione.
- `layout`: directory che contiene i file XML `activity_main.xml`, `activity_ - one.xml`, `activity_two.xml`, `activity_three.xml`, `busyumbrelladialog.xml`, `content_ - three.xml`, `freeumbrelladialog.xml`, `note_dialog.xml`, `note - _list_row.xml`, `servizi1.xml`, `servizi2.xml`, `servizi3.xml`, `servizi4.xml`, `single_view.xml` e `slideshow_layout.xml` . Questi file XML servono per descrivere cosa deve essere visualizzato sullo schermo e dove.
- `menu`: directory che contiene un file per la visualizzazione della bottom navigation bar.
- `mipmap`: directory che contiene l'icona dell'app, visibile dal menu di navigazione.

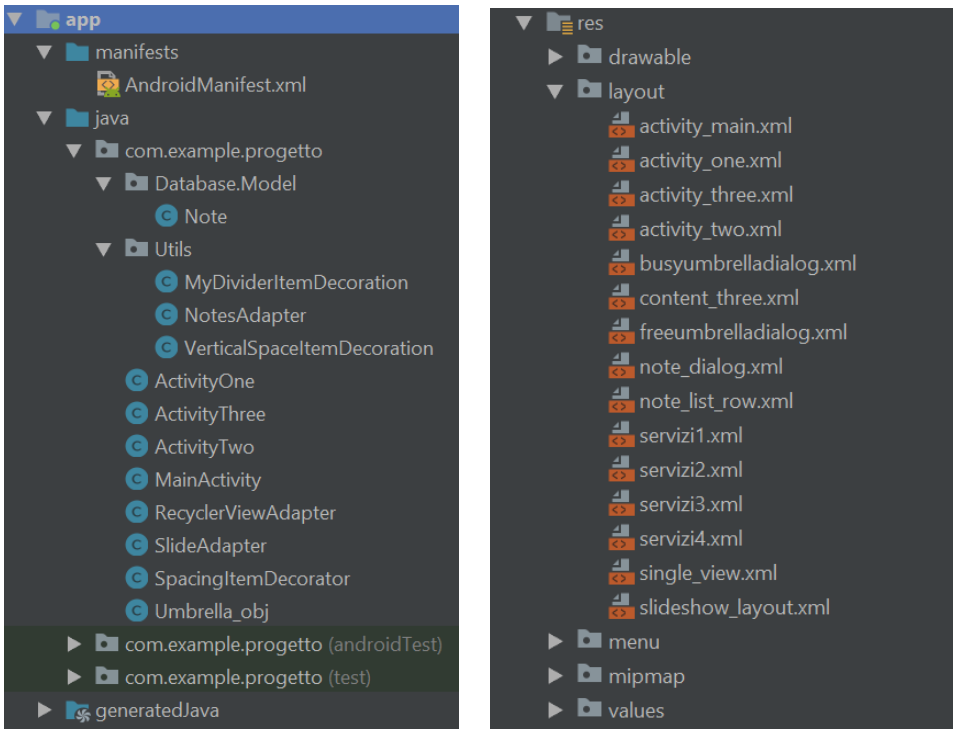

Figura 4.1. Struttura dell'applicazione

- `values`: directory che contiene dei file XML, necessari per etichettare alcune componenti fondamentali dell'app, come il nome o i colori primari e secondari.

4.2 Implementazione

In questa sezione si vedrà il codice relativo ad alcune delle componenti menzionate in precedenza, in particolare, si farà riferimento a quelle contenute nelle directory `manifests`, `com.example.progetto` e `layout`.

4.2.1 Manifest

Come già accennato in precedenza, il Manifest contiene le caratteristiche principali dell'applicazione, senza le quali non sarebbe possibile compilare alcuna porzione di codice.

Nel caso di progetto, all'interno del Manifest (dal nome `AndroidManifest.xml`) vanno inseriti i permessi relativi all'utilizzo della connessione di rete (Listato 4.1).

```
<uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Listato 4.1. Permessi per la connessione di rete

È presente, inoltre, il tag `application`, che contiene informazioni riguardanti il nome, l'icona, il tema e le dichiarazioni delle activity del progetto.

Il tag `application` è presentato nel Listato 4.2.

```
<application
        android:usesCleartextTraffic="true"
        android:allowBackup="true"
        android:icon="@mipmap/iconapp_round"
        android:label="Papeete␣APP"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:ignore="GoogleAppIndexingWarning"
      >
        <activity android:name="com.example.progetto.ActivityOne"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name="com.example.progetto.MainActivity"/>
        <activity android:name="com.example.progetto.ActivityTwo"/>
        <activity android:name="com.example.progetto.ActivityThree"/>

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


    </application>
```

Listato 4.2. Il tag `application` del manifest

Si noti come le activity possiedano l'attributo `android:screenOrientation="portrait"`. Questo parametro impedisce che l'app ruoti in seguito alla rotazione dello schermo da parte dell'utente; tale comportamento è necessario poiché il design dell'app è stato progettato esclusivamente per un utilizzo in verticale.

4.2.2 Le Activity

In questa sezione verranno analizzate le diverse activity che compongono il progetto. Si farà riferimento alla classe e al relativo file XML di ogni activity.

Le activity principali che caratterizzano il progetto sono quattro, e ognuna ha uno scopo ben definito:

- `Activityone`: è la prima activity, che viene mostrata all'utente all'avvio dell'app; contiene le informazioni di carattere generale sullo stabilimento balneare.
- `ActivityTwo`: quest'activity, invece, permette all'utente di effettuare una prenotazione di un ombrellone.
- `ActivityThree`: è l'activity in cui l'utente può leggere o inserire una recensione sull'esperienza trascorsa.
- `MainActivity`: in questa activity l'utente può autenticarsi o creare il proprio account.

ActivityOne

`ActivityOne` è la prima activity che viene visualizzata all'avvio dell'applicazione.

Nel codice viene implementato uno "slideshow" tramite la classe `SlideAdapter` (Listato 4.3) che permette la visualizzazione con scorrimento orizzontale di più immagini relative allo stabilimento.

Nell'activity vengono, inoltre, definite le finestre di dialogo che vengono mostrate quando l'utente seleziona uno dei servizi offerti (menu, WiFi, doccia, parcheggio).

Viene, infine, implementata la `Bottom Navigation Bar` con uno `switch` in modo che tutte le activity siano collegate.

Il file XML, invece, risulta essere composto da una `Textview` posta in alto a sinistra contenente il testo "Benvenuto", e da una serie di `ImageView`, di forma rettangolare, per racchiudere lo slideshow e le icone relative ai servizi.

I codici della classe e del file XML sono presentati, rispettivamente, nei Listati 4.4 e 4.5.

```
package com.example.progetto;

import android.content.Context;
import androidx.annotation.NonNull;
import androidx.viewpager.widget.PagerAdapter;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.LinearLayout;

public class SlideAdapter extends PagerAdapter {

    private Context context;
    LayoutInflater inflater;

    public int [] images={
      R.drawable.lido4,
      R.drawable.lettinispiaggia,
      R.drawable.papetedonna
    };

    public SlideAdapter(Context context) {
        this.context = context;
    }

    @Override
    public int getCount() {
        return images.length;
    }

    @Override
    public boolean isViewFromObject(@NonNull View view, @NonNull Object object) {
        return (view==(LinearLayout)object);
    }

    @NonNull
    @Override
    public Object instantiateItem(@NonNull ViewGroup container, int position) {
        inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View view = inflater.inflate(R.layout.slideshow_layout, container,false);

        ImageView img = (ImageView) view.findViewById(R.id.imageView_id);
        img.setImageResource(images[position]);

        container.addView(view);
        return view;
    }

    @Override
    public void destroyItem(@NonNull ViewGroup container, int position, @NonNull Object object) {
        container.removeView((LinearLayout)object);


    }
}
```

Listato 4.3. Adapter per l'implementazione dello slideshow

```
package com.example.progetto;
import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
```

```
import android.widget.ImageView;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.viewpager.widget.ViewPager;

public class ActivityOne extends AppCompatActivity implements View.OnClickListener {


    ViewPager viewPager;
    SlideAdapter adapter;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one);

        ImageView wifiButtonALert = findViewById(R.id.imageView3);
        ImageView parkButtonALert = findViewById(R.id.imageView4);
        ImageView eatButtonALert = findViewById(R.id.imageView5);
        ImageView showerButtonALert = findViewById(R.id.imageView6);
        wifiButtonALert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                final AlertDialog.Builder alert = new AlertDialog.Builder(ActivityOne.this);
                View  mView = getLayoutInflater().inflate(R.layout.servizi1, null);
                alert.setView(mView);

                final AlertDialog alertDialog = alert.create();
                alertDialog.setCanceledOnTouchOutside(true);

                alertDialog.show();
            }
        });
        parkButtonALert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                final AlertDialog.Builder alert = new AlertDialog.Builder(ActivityOne.this);
                View  mView = getLayoutInflater().inflate(R.layout.servizi2, null);
                alert.setView(mView);

                final AlertDialog alertDialog = alert.create();
                alertDialog.setCanceledOnTouchOutside(true);

                alertDialog.show();
            }
        });
        eatButtonALert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                final AlertDialog.Builder alert = new AlertDialog.Builder(ActivityOne.this);
                View  mView = getLayoutInflater().inflate(R.layout.servizi3, null);
                alert.setView(mView);

                final AlertDialog alertDialog = alert.create();
                alertDialog.setCanceledOnTouchOutside(true);

                alertDialog.show();
            }
        });
        showerButtonALert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                final AlertDialog.Builder alert = new AlertDialog.Builder(ActivityOne.this);
                View  mView = getLayoutInflater().inflate(R.layout.servizi4, null);
                alert.setView(mView);

                final AlertDialog alertDialog = alert.create();
                alertDialog.setCanceledOnTouchOutside(true);

                alertDialog.show();
            }
        });


        BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);
        bottomNavigationView.setSelectedItemId(R.id.action_home);
        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.
        OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                switch (menuItem.getItemId()) {
                    case R.id.action_prenotazione:
                        Intent intent2 = new Intent(ActivityOne.this, ActivityTwo.class);
                        startActivity(intent2);
                        break;
                    case R.id.action_recensioni:
                        Intent intent3 = new Intent(ActivityOne.this, ActivityThree.class);
                        startActivity(intent3);
                        break;
                    case R.id.action_profilo:
```

```
                    Intent intent4 = new Intent(ActivityOne.this,MainActivity.class);
                    startActivity(intent4);
                }
                return false;
            }
        });

            viewPager= findViewById(R.id.viewPager_id);
            adapter= new SlideAdapter(this);
            viewPager.setAdapter(adapter);


        }

        @Override
        public void onClick(View v) { }


}
```

Listato 4.4. Il codice della classe associato ad `ActivityOne`

```
<?xml version="1.0" encoding="utf-8"?>
    <androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/white"
        tools:context=".MainActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0">

        <com.google.android.material.bottomnavigation.BottomNavigationView
            android:id="@+id/bottom_navigation"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="0dp"
            app:itemBackground="@color/white"
            app:itemIconTint="@drawable/selector"
            app:itemTextColor="@drawable/selector"
            app:labelVisibilityMode="labeled"

            app:menu="@menu/bottom_nav_menu" />

        <androidx.viewpager.widget.ViewPager
            android:id="@+id/viewPager_id"
            android:layout_width="372dp"
            android:layout_height="229dp"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_alignParentEnd="true"
            android:layout_marginStart="18dp"
            android:layout_marginTop="125dp"
            android:layout_marginEnd="18dp" />

        <ImageView
            android:layout_width="200dp"
            android:layout_height="250dp"
            android:layout_alignParentEnd="true"
            android:layout_alignParentBottom="true"
            android:layout_marginEnd="18dp"
            android:layout_marginBottom="100dp"
            android:background="@drawable/rettangolo_servizi" />


        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/titolo"
            android:layout_alignParentStart="true"
            android:layout_alignParentEnd="true"
            android:layout_alignParentBottom="true"
            android:layout_marginStart="16dp"
            android:layout_marginTop="40dp"
            android:layout_marginEnd="16dp"
            android:layout_marginBottom="372dp"
            android:background="@drawable/rettangolo_slideshow" />
```

```
    <ImageView
        android:layout_width="150dp"
        android:layout_height="250dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="18dp"
        android:layout_marginEnd="250dp"
        android:layout_marginBottom="100dp"
        android:background="@drawable/rettangolo_meteo" />

    <ImageView
        android:id="@+id/meteo"
        android:layout_width="77dp"
        android:layout_height="81dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="51dp"
        android:layout_marginBottom="243dp"
        app:srcCompat="@drawable/icona_meteo" />

    <TextView
        android:id="@+id/temperatura"
        android:layout_width="wrap_content"
        android:layout_height="73dp"
        android:layout_alignBottom="@+id/imageView6"
        android:layout_alignParentStart="true"
        android:layout_marginStart="61dp"
        android:layout_marginBottom="11dp"
        android:text="25°"
        android:textColor="@color/scritte_nere"
        android:textSize="45dp"
        android:textStyle="bold" />


    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="67dp"
        android:layout_height="63dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="213dp"
        android:layout_marginBottom="259dp"
        app:srcCompat="@drawable/wifi" />

    <ImageView
        android:id="@+id/imageView4"
        android:layout_width="67dp"
        android:layout_height="63dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="315dp"
        android:layout_marginEnd="29dp"
        android:layout_marginBottom="258dp"
        app:srcCompat="@drawable/car_parking" />

    <ImageView
        android:id="@+id/imageView5"
        android:layout_width="67dp"
        android:layout_height="63dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="129dp"
        android:layout_marginBottom="144dp"
        app:srcCompat="@drawable/wine" />

    <ImageView
        android:id="@+id/imageView6"
        android:layout_width="67dp"
        android:layout_height="63dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="29dp"
        android:layout_marginBottom="141dp"
        app:srcCompat="@drawable/shower" />


    <TextView
        android:id="@+id/titolo"
        android:layout_width="248dp"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginStart="19dp"
        android:layout_marginTop="23dp"
        android:text="Benvenuto"
        android:textColor="@color/scritte_nere"
        android:textSize="45dp"
        android:textStyle="bold" />


</RelativeLayout>
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

Listato 4.5. Il codice del file XML associato ad `ActivityOne`

ActivityTwo

`ActivityTwo` implementa le funzionalità per effettuare una prenotazione. L'activity deve, quindi:

- leggere dal database la lista degli ombrelloni;
- implementare le funzionalità per poter effettuare una prenotazione che deve essere, poi, memorizzata nel database

Nel codice viene definita una `RecyclerView` per la visualizzazione degli ombrelloni; si è scelto di procedere così in modo tale da rendere più semplice la possibilità di aggiungere o eliminare ombrelloni.

Tale `RecyclerView` viene popolata mediante il metodo `LoadDataFromFirestore` che estrae i dati relativi agli ombrelloni dal database e li inserisce in una lista `umbrellalist`. Fatto questo viene settato l'adapter, `RecyclerViewAdapter` (Listato 4.6) , dove viene implementato il codice che permette la prenotazione (apertura della finestra di dialogo, conferma della prenotazione, salvataggio della prenotazione nel database).

Nel file XML, invece, si definiscono la `RecyclerView` sopra citata e gli elementi grafici già visti nei mockup del Capitolo 3.

I codici della classe e del file XML sono presentati, rispettivamente, nei Listati 4.7 e 4.8.

```
package com.example.progetto;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.List;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.recyclerview.widget.RecyclerView;

public class RecyclerViewAdapter extends RecyclerView.Adapter<RecyclerViewAdapter.MyViewHolder2> {


    private List<Umbrella_obj> umbrellalist;
    ActivityTwo activityTwo;

    public RecyclerViewAdapter(ActivityTwo activityTwo, List<Umbrella_obj> umbrellalist){
        this.activityTwo = activityTwo;
        this.umbrellalist = umbrellalist;
    }

    @NonNull
    @Override
    public MyViewHolder2 onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.single_view, parent, false);
        MyViewHolder2 myViewHolder2 = new MyViewHolder2(view);
        return myViewHolder2;
    }
```

```
@Override
public void onBindViewHolder(@NonNull final MyViewHolder2 holder, final int position) {
    final Umbrella_obj umbrella_obj = umbrellalist.get(position);
    if(umbrella_obj.getPrenotato()){
        holder.imageView.setImageResource(R.drawable.umbrella_busy);
    } else {
        holder.imageView.setImageResource(R.drawable.umbrella_free);
    }

    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
            if(user != null){
                if(umbrella_obj.getPrenotato()){
                    LayoutInflater inflater= activityTwo.getLayoutInflater();

                    final AlertDialog.Builder alert = new AlertDialog.Builder(activityTwo);
                    View mView = inflater.inflate(R.layout.busyumbrelladialog, null);

                    Button btn_cancel = mView.findViewById(R.id.btn_cancel);

                    alert.setView(mView);

                    final AlertDialog alertDialog = alert.create();
                    alertDialog.setCanceledOnTouchOutside(false);

                    btn_cancel.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            alertDialog.dismiss();
                        }
                    });

                    alertDialog.show();

                } else {
                    LayoutInflater inflater= activityTwo.getLayoutInflater();

                    final AlertDialog.Builder alert = new AlertDialog.Builder(activityTwo);
                    View mView = inflater.inflate(R.layout.freeumbrelladialog, null);

                    Button btn_cancel = mView.findViewById(R.id.btn_cancel);
                    Button btn_confirm = mView.findViewById(R.id.btn_confirm);

                    alert.setView(mView);

                    final AlertDialog alertDialog = alert.create();
                    alertDialog.setCanceledOnTouchOutside(false);

                    alertDialog.show();

                    btn_cancel.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            Toast.makeText(activityTwo,"Prenotazione annullata",Toast.LENGTH_SHORT).show();
                            alertDialog.dismiss();
                        }
                    });

                    btn_confirm.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            FirebaseFirestore db = FirebaseFirestore.getInstance();
                            DocumentReference umbrellareference = db.collection("Ombrelloni")
                            .document(umbrellalist.get(position).getId());
                            umbrellareference.update("prenotato", true);
                            umbrellalist.get(position).setPrenotato(true);
                            notifyDataSetChanged();
                            Toast.makeText(activityTwo,"Penotazione Effettuata",Toast.LENGTH_SHORT).show();
                            alertDialog.dismiss();
                        }
                    });
                }
            } else {
    Toast.makeText(activityTwo,"Per prenotare un ombrellone devi essere autenticato",Toast.LENGTH_SHORT).show();
            }

        }
    });

}

@Override
public int getItemCount() {
    return umbrellalist.size();
}

public class MyViewHolder2 extends RecyclerView.ViewHolder {

    ImageView imageView;

    public MyViewHolder2(@NonNull View itemView) {
        super(itemView);
```

```
            imageView=itemView.findViewById(R.id.imageViewUmb);

        }

    }

}
```

Listato 4.6. Il codice della classe associata al `RecyclerViewAdapter`

```
package com.example.progetto;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.List;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

public class ActivityTwo extends AppCompatActivity {

    private List<Umbrella_obj> umbrellalist = new ArrayList<>();
    private RecyclerView recyclerView;
    RecyclerView.LayoutManager layoutManager;
    RecyclerViewAdapter recyclerViewAdapter;

    FirebaseFirestore db = FirebaseFirestore.getInstance();

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_two);

        recyclerView = findViewById(R.id.viewRecycler);
        layoutManager = new GridLayoutManager(this,4);
        recyclerView.setLayoutManager(layoutManager);
        SpacingItemDecorator itemDecorator = new SpacingItemDecorator(5);
        recyclerViewAdapter= new RecyclerViewAdapter(this,umbrellalist);
        recyclerView.setAdapter( recyclerViewAdapter);
        umbrellalist = new ArrayList<>();
        recyclerView.addItemDecoration(itemDecorator);
        LoadDataFromFirestore();

        BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);
        bottomNavigationView.setSelectedItemId(R.id.action_prenotazione);
        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.
        OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                switch (menuItem.getItemId()) {
                    case R.id.action_home:
                        Intent intent1 = new Intent(ActivityTwo.this, ActivityOne.class);
                        startActivity(intent1);
                        break;
                    case R.id.action_recensioni:
                        Intent intent3 = new Intent(ActivityTwo.this, ActivityThree.class);
                        startActivity(intent3);
                        break;
                    case R.id.action_profilo:
                        Intent intent4 = new Intent(ActivityTwo.this,MainActivity.class);
                        startActivity(intent4);
                }
                return false;
            }
        });

    }

    private void LoadDataFromFirestore() {
        db.collection("Ombrelloni").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                for(QueryDocumentSnapshot querySnapshot : task.getResult()){
                    Log.d("id",querySnapshot.getId());
                    Umbrella_obj umbrella_obj = new Umbrella_obj(querySnapshot.getBoolean("prenotato"),querySnapshot
```

```
                .getId());
                umbrellalist.add(umbrella_obj);
            }
            recyclerViewAdapter = new RecyclerViewAdapter(ActivityTwo.this,umbrellalist);
            recyclerView.setAdapter(recyclerViewAdapter);
        }
    });
}
```

Listato 4.7. Il codice della classe associata ad `ActivityTwo`

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    tools:context=".MainActivity">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/relativeLayout2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout_editor_absoluteX="172dp">

        <com.google.android.material.bottomnavigation.BottomNavigationView
            android:id="@+id/bottom_navigation"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:itemBackground="@color/white"
            app:itemIconTint="@drawable/selector"
            app:itemTextColor="@drawable/selector"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:labelVisibilityMode="labeled"
            app:menu="@menu/bottom_nav_menu" />

        <TextView
            android:id="@+id/titolo2"
            android:layout_width="297dp"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_marginStart="16dp"
            android:layout_marginTop="23dp"
            android:text="Prenotazione"
            android:textColor="@color/scritte_nere"
            android:textSize="45dp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/viewRecycler"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="8dp"
            app:layout_constraintBottom_toTopOf="@+id/imageView"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/titolo2"
            app:layout_constraintVertical_bias="0.43" />

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="390dp"
            android:layout_height="58dp"
            android:layout_alignParentStart="true"
            android:layout_alignParentEnd="true"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="36dp"
            android:background="@drawable/rettangolo_mare"
            app:layout_constraintBottom_toTopOf="@+id/bottom_navigation"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.493"
            app:layout_constraintStart_toStartOf="parent" />

    </androidx.constraintlayout.widget.ConstraintLayout>
```

Listato 4.8. Il codice del file XML associato ad `ActivityTwo`

ActivityThree

`ActivityThree` è la pagina che permette la visualizzazione e l'inserimento delle recensioni. L'activity deve, quindi:

- leggere le recensioni salvate sul database;
- implementare le funzionalità per l'inserimento di una recensione da parte di un utente registrato e, di conseguenza, salvare l'inserimento nel database.

Come nell' `ActivityTwo`, anche in questo caso, nel codice viene implementata una `RecyclerView` per la visualizzazione delle recensioni sullo schermo.

La lettura delle recensioni dal database avviene tramite il metodo `LoadDataFromdatabase` che, tramite una query, riempie una lista chiamata `notesList` contenente tutte le recensioni salvate nel database. Quest'ultima viene, poi, passata all'adapter `NotesAdapter` (Listato 4.9) che popola la `RecyclerView` precedentemente descritta.

L'operazione di inserimento di una recensione avviene grazie alla funzione `showNoteDialog` che apre una finestra di dialogo in cui l'utente compila i campi. Il salvataggio avviene tramite il metodo `createNewReview"` che, con una query, salva la recensione sul database.

Una volta fatto ciò si riparte con l'operazione di lettura precedentemente descritta e, quindi, la `RecyclerView` viene aggiornata.

Nel file XML relativo all'`ActivityThree` si definisce la `RecyclerView` attraverso l'inclusione di un secondo file XML `content_three.xml` (Listato 4.12) ed il pulsante per poter accedere alla finestra di dialogo per l'inserimento.

I codici della classe e del file XML sono presentati, rispettivamente, nei Listati 4.10 e 4.11.

```
package com.example.progetto.Utils;


import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RatingBar;
import android.widget.TextView;

import com.example.progetto.Database.Model.Note;
import com.example.progetto.R;

import java.util.List;

import androidx.recyclerview.widget.RecyclerView;

public class NotesAdapter extends RecyclerView.Adapter<NotesAdapter.MyViewHolder> {

    private Context context;
    private List<Note> notesList;

    public class MyViewHolder extends RecyclerView.ViewHolder {
        public TextView note;
        public TextView IDuser;
        public RatingBar ratingBar;

        public MyViewHolder(View view) {
            super(view);
            note = view.findViewById(R.id.note);
            IDuser = view.findViewById(R.id.email);
            ratingBar = view.findViewById(R.id.rating_bar2);
        }
    }


    public NotesAdapter(Context context, List<Note> notesList) {
        this.context = context;
        this.notesList = notesList;
    }

    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
```

```
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.note_list_row, parent, false);

        return new MyViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(MyViewHolder holder, int position) {
        Note note = notesList.get(position);
        holder.note.setText(note.getTesto());
        holder.IDuser.setText(note.getUserID());
        holder.ratingBar.setRating(note.getRating());
    }

    public void update(List<Note> notesList){
        this.notesList = notesList;
        notifyDataSetChanged();
    }

    @Override
    public int getItemCount() {
        return notesList.size();
    }


}
```

Listato 4.9. Il codice della classe associata al `NotesAdapter`

```
package com.example.progetto;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.RatingBar;
import android.widget.TextView;
import android.widget.Toast;

import com.example.progetto.Database.Model.Note;
import com.example.progetto.Utils.NotesAdapter;
import com.example.progetto.Utils.VerticalSpaceItemDecoration;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.coordinatorlayout.widget.CoordinatorLayout;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;


public class ActivityThree extends AppCompatActivity
    implements View.OnClickListener {

    private NotesAdapter mAdapter;
    private List<Note> notesList = new ArrayList<>();
    private CoordinatorLayout coordinatorLayout;
    private RecyclerView recyclerView;
    private TextView noNotesView;
    private RatingBar ratingBar;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_three);

        BottomNavigationView bottomNavigationView = (BottomNavigationView) findViewById(R.id.bottom_navigation);
        bottomNavigationView.setSelectedItemId(R.id.action_recensioni);
```

```
        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView
        .OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                switch (menuItem.getItemId()) {
                    case R.id.action_home:
                        Intent intent1 = new Intent(ActivityThree.this, ActivityOne.class);
                        startActivity(intent1);
                        break;
                    case R.id.action_prenotazione:
                        Intent intent2 = new Intent(ActivityThree.this, ActivityTwo.class);
                        startActivity(intent2);
                        break;
                    case R.id.action_profilo:
                        Intent intent4 = new Intent(ActivityThree.this,MainActivity.class);
                        startActivity(intent4);
                }
                return false;
            }
        });

        coordinatorLayout = findViewById(R.id.coordinator_layout);
        recyclerView = findViewById(R.id.recycler_view);
        noNotesView = findViewById(R.id.empty_notes_view);
        ratingBar = findViewById(R.id.rating_bar);

        FloatingActionButton fab = findViewById(R.id.fab);

        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        if(user != null) {
            fab.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    showNoteDialog(null, -1,0);
                }
            });
        } else {
            Toast.makeText(this,"Per inserire una recensione devi aver effettuato il login",Toast.LENGTH_SHORT).show();
        }

        mAdapter = new NotesAdapter(this, notesList);
        VerticalSpaceItemDecoration itemDecoration = new VerticalSpaceItemDecoration(20);
        recyclerView.addItemDecoration(itemDecoration);
        RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager
                (getApplicationContext());
        recyclerView.setLayoutManager(mLayoutManager);
        recyclerView.setItemAnimator(new DefaultItemAnimator());
        recyclerView.setAdapter(mAdapter);
        LoadDataFromdatabase();

    }

    public void createNewReview (String note, float rating, int position){
        FirebaseFirestore db = FirebaseFirestore.getInstance();

        String userID = FirebaseAuth.getInstance().getCurrentUser().getEmail();

        Map<String,Object> review_data = new HashMap<>();
        review_data.put("testo",note);
        review_data.put("userID",userID);
        review_data.put("valutazione",rating);
        review_data.put("posizione",position);

        db.collection("Recensioni").add(review_data).addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
            @Override
            public void onSuccess(DocumentReference documentReference) {
                if(documentReference != null){
                    Toast.makeText(ActivityThree.this,"Inserimento riuscito",Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(ActivityThree.this, "Inserimento non riuscito", Toast.LENGTH_SHORT).show();
                }
            }
        });
        mAdapter.update(notesList);
        notesList.clear();
        LoadDataFromdatabase();
    }

    private void LoadDataFromdatabase(){
        FirebaseFirestore db = FirebaseFirestore.getInstance();
        db.collection("Recensioni").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                for(QueryDocumentSnapshot querySnapshot : task.getResult()){
                    Note n = new Note(querySnapshot.getString("testo"), querySnapshot.getString("userID"), querySnapshot
                    .getLong("valutazione"));
                    notesList.add(n);
                    notesList.size();
                    if(notesList.size() == 0){

                    } else {
                        noNotesView.setVisibility(View.INVISIBLE);
                    }
                }
```

```
                mAdapter = new NotesAdapter(ActivityThree.this, notesList);
                recyclerView.setAdapter(mAdapter);
            }
        });
    }

    private void showNoteDialog(final Note note, final int position, final float rating) {
        LayoutInflater layoutInflaterAndroid = LayoutInflater.from(getApplicationContext());
        View view = layoutInflaterAndroid.inflate(R.layout.note_dialog, null);

        AlertDialog.Builder alertDialogBuilderUserInput = new AlertDialog.Builder(ActivityThree.this);
        alertDialogBuilderUserInput.setView(view);


        final RatingBar ratingBar1 = view.findViewById(R.id.rating_bar);
        final EditText inputNote = view.findViewById(R.id.note);


        alertDialogBuilderUserInput
                .setCancelable(false)
                .setPositiveButton("Salva", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialogBox, int id) {
                        float rating1 = ratingBar1.getRating();
                        createNewReview(inputNote.getText().toString(),rating1,position);
                        dialogBox.dismiss();
                    }
                })
                .setNegativeButton("Cancella",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialogBox, int id) {
                                dialogBox.cancel();
                            }
                        });

        final AlertDialog alertDialog = alertDialogBuilderUserInput.create();
        alertDialog.show();
    }

    @Override
    public void onClick(View v) {

    }
}
```

Listato 4.10. Il codice della classe associato ad `ActivityThree`

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/coordinator_layout"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    tools:context="info.androidhive.sqlite.view.ActivityThree">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

    </com.google.android.material.appbar.AppBarLayout>

    <include
        android:id="@+id/include"
        layout="@layout/content_three" />

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/constraintLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.google.android.material.floatingactionbutton.FloatingActionButton
            android:id="@+id/fab"
            android:layout_width="wrap_content"
            android:layout_height="70dp"
            android:layout_gravity="bottom|right"
            android:layout_marginStart="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="72dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.976"
            app:layout_constraintStart_toStartOf="parent"
            app:srcCompat="@drawable/ic_action_plus"
            tools:ignore="MissingConstraints" />
```

```
        <com.google.android.material.bottomnavigation.BottomNavigationView
            android:id="@+id/bottom_navigation"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:itemBackground="@color/white"
            app:itemIconTint="@drawable/selector"
            app:itemTextColor="@drawable/selector"
            app:labelVisibilityMode="labeled"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:menu="@menu/bottom_nav_menu" />


    </androidx.constraintlayout.widget.ConstraintLayout>


</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Listato 4.11. Il codice del file XML associato ad `activity_three.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="info.androidhive.sqlite.view.ActivityThree"
    tools:showIn="@layout/activity_three">

    <TextView
        android:id="@+id/titolo"
        android:layout_width="248dp"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginStart="19dp"
        android:layout_marginTop="23dp"
        android:text="Recensioni"
        android:textColor="@color/scritte_nere"
        android:textSize="45dp"
        android:textStyle="bold" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="490dp"
        android:layout_below="@+id/titolo"
        android:layout_alignParentStart="true"
        android:layout_marginStart="0dp"
        android:layout_marginTop="28dp" />

    <TextView
        android:id="@+id/empty_notes_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="@dimen/margin_top_no_notes"
        android:fontFamily="sans-serif-light"
        android:text="@string/msg_no_notes"
        android:textColor="@color/msg_no_notes"
        android:textSize="@dimen/msg_no_notes" />

</RelativeLayout>
```

Listato 4.12. Il codice del file XML associato al `content_three.xml`

4.2.3 MainActivity

`MainActivity` è l'activity tramite cui l'utente si autentica o, se ancora non l'ha fatto, può crearsi il proprio account.

La procedura di autenticazione e registrazione è gestita dal metodo `onActivityResult` che, una volta inserite le credenziali, controlla che l'account esista; se tale account non esiste, invece, si procede con la registrazione.

L'utente ha due opzioni diverse per la creazione dell'account che, nel codice, sono definite nei `providers`; la registrazione può avvenire, infatti, mediante email e password o tramite l'utilizzo del proprio account Google.

Il file XML, invece, è formato da quattro blocchi:

- `TextView` contenente il titolo dell'activity;
- `TextView` con delle informazioni su quali funzionalità in più si hanno dopo aver effettuato la registrazione;
- `Button` per l'autenticazione o la registrazione ;
- `Button` per il logout.

Il codice della classe e del file XML sono presentati, rispettivamente, nei Listati 4.13 e 4.14.

```
package com.example.progetto;

import android.content.Intent;
import android.os.Bundle;
import android.os.StrictMode;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.firebase.ui.auth.AuthUI;
import com.firebase.ui.auth.IdpResponse;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import java.util.Arrays;
import java.util.List;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private static final int MY_REQUEST_CODE = 1234;
    private Button aggiungiUtentePost;

    private FirebaseAuth mAuth;

    List<AuthUI.IdpConfig> providers;
    Button btn_sign_out;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mAuth = FirebaseAuth.getInstance();

        btn_sign_out = findViewById(R.id.btn_sign_out);
        btn_sign_out.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                AuthUI.getInstance()
                        .signOut(MainActivity.this)
                        .addOnCompleteListener(new OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull Task<Void> task) {
                                btn_sign_out.setEnabled(false);
                                showSignInOptions();
                            }
                        }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(MainActivity.this, ""+e.getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });

            }
        });
        providers = Arrays.asList(
                new AuthUI.IdpConfig.EmailBuilder().build(),
                new AuthUI.IdpConfig.GoogleBuilder().build()
```

```
        );
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);

        aggiungiUtentePost = findViewById(R.id.aggiungi_utente_post);
        aggiungiUtentePost.setOnClickListener(this);

        BottomNavigationView bottomNavigationView = findViewById(R.id.bottom_navigation);
        bottomNavigationView.setSelectedItemId(R.id.action_profilo);
        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView
        .OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                switch (menuItem.getItemId()) {
                    case R.id.action_home:
                        Intent intent1 = new Intent(MainActivity.this, ActivityOne.class);
                        startActivity(intent1);
                        break;
                    case R.id.action_prenotazione:
                        Intent intent2 = new Intent(MainActivity.this, ActivityTwo.class);
                        startActivity(intent2);
                        break;
                    case R.id.action_recensioni:
                        Intent intent3 = new Intent(MainActivity.this, ActivityThree.class);
                        startActivity(intent3);
                        break;
                }
                return false;
            }
        });

    }

    private void showSignInOptions() {
        startActivityForResult(
                AuthUI.getInstance().createSignInIntentBuilder()
                .setAvailableProviders(providers)
                .setTheme(R.style.MyTheme)
                .build(), MY_REQUEST_CODE
        );
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode == MY_REQUEST_CODE)
        {
            IdpResponse response = IdpResponse.fromResultIntent(data);
            if(resultCode == RESULT_OK)
            {
                FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                Toast.makeText(this,""+user.getEmail(),Toast.LENGTH_SHORT).show();
                btn_sign_out.setEnabled(true);
                aggiungiUtentePost.setVisibility(View.GONE);
            } else
                {
                    Intent intent4 = new Intent(MainActivity.this,MainActivity.class);
                    startActivity(intent4);
                    //Toast.makeText(this, ""+response.getError().getMessage(), Toast.LENGTH_SHORT).show();
                }
        }
    }

    @Override
    public void onStart() {
        super.onStart();
        // Check if user is signed in (non-null) and update UI accordingly.
        FirebaseUser currentUser = mAuth.getCurrentUser();
        updateUI(currentUser);
    }

    public void updateUI(FirebaseUser account){
        if(account != null){
            aggiungiUtentePost.setVisibility(View.GONE);
            btn_sign_out.setEnabled(true);
        }else {

        }
    }

    @Override
    public void onClick(View v) {
            aggiungiUtentePost = findViewById(R.id.aggiungi_utente_post);
            showSignInOptions();
        }

}
```

Listato 4.13. Il codice della classe associato a `MainActivity`

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    tools:context=".MainActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0">

        <com.google.android.material.bottomnavigation.BottomNavigationView
            android:id="@+id/bottom_navigation"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="0dp"
            app:itemBackground="@color/white"
            app:itemIconTint="@drawable/selector"
            app:itemTextColor="@drawable/selector"
            app:labelVisibilityMode="labeled"
            app:menu="@menu/bottom_nav_menu" />

        <TextView
            android:id="@+id/titolo3"
            android:layout_width="248dp"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_marginStart="19dp"
            android:layout_marginTop="23dp"
            android:text="Profilo"
            android:textColor="@color/scritte_nere"
            android:textSize="45dp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/cosa_fare"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_alignParentEnd="true"
            android:layout_marginStart="56dp"
            android:layout_marginTop="158dp"
            android:layout_marginEnd="56dp"
            android:text="Registrandosi␣si␣ha␣la␣possibilità.."
            android:textColor="@color/scritte_nere"
            android:textSize="25dp"
            android:textStyle="normal" />

        <Button
            android:id="@+id/aggiungi_utente_post"
            android:layout_width="217dp"
            android:layout_height="wrap_content"
            android:layout_above="@+id/btn_sign_out"
            android:layout_alignParentStart="true"
            android:layout_marginStart="95dp"
            android:layout_marginBottom="15dp"
            android:allowUndo="false"
            android:autoLink="none"
            android:background="@drawable/roundedbutton"
            android:text="Registrati␣/␣Accedi"
            android:textColor="@color/white" />

        <Button
            android:id="@+id/btn_sign_out"
            android:layout_width="220dp"
            android:layout_height="45dp"
            android:layout_alignParentBottom="true"
            android:layout_marginStart="92dp"
            android:layout_marginBottom="130dp"
            android:background="@drawable/roundedbutton"
            android:enabled="false"
            android:text="Disconnetti"
            android:layout_above="@+id/bottom_navigation"
            android:textColor="@color/white" />
    </RelativeLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Listato 4.14. Il codice del file XML associato ad `activity_main.xml`

4.2.4 Collegamento alla console Firebase

Firebase è un backend, in particolare un servizio offerto da Google che viene utilizzato per semplificare la gestione dei dati della propria applicazione Android.

In pratica Firebase ci aiuta nella gestione dell'archivio dati, descritto in maniera molto intuitiva in modo da poter effettuare modifiche per soddisfare le proprie esigenze.

La console Firebase offre diversi servizi tra cui:

- Archiviazione dati;
- Autenticazione;
- Hosting.

Il collegamento della console all'applicazione è molto semplice. Basta, infatti, accedere nella pagina di Firebase con il proprio account Google e creare un nuovo progetto seguendo le istruzioni che vengono fornite.

Una volta fatto ciò si termina il collegamento direttamente nell'IDE Android Studio; selezionando la voce "Firebase" che si trova nel menù "Tools", si aprirà una finestra (Figura 4.2) in cui sono elencati tutti i servizi offerti dalla piattaforma e le relativi istruzioni per completare il collegamento dell'app con la console.

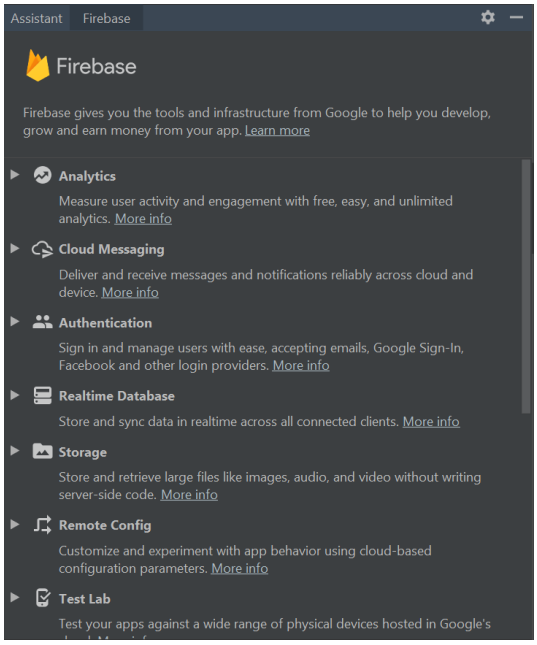

Figura 4.2. Finestra di configurazione di Firebase

Nel caso del progetto, come visto nelle lezioni precedenti, sono stati utilizzati i servizi di archiviazione dati e autenticazione.

4.2.5 Creazione database su Firebase

La creazione di un database su Firebase avviene direttamente nell'applicativo web della console.

Selezionando l'opzione "Database" (Figura 4.3) si apre una schermata (Figura 4.4) in cui si può avviare una "Raccolta" che sta a rappresentare il contenitore dei nostri dati. La raccolta viene popolata aggiungendo dei "Documenti", che rappresentano i dati veri e propri. Ogni documento sarà a sua volta caratterizzato da uno o più campi, i quali possono essere modificati a piacere da chi implementa il database.

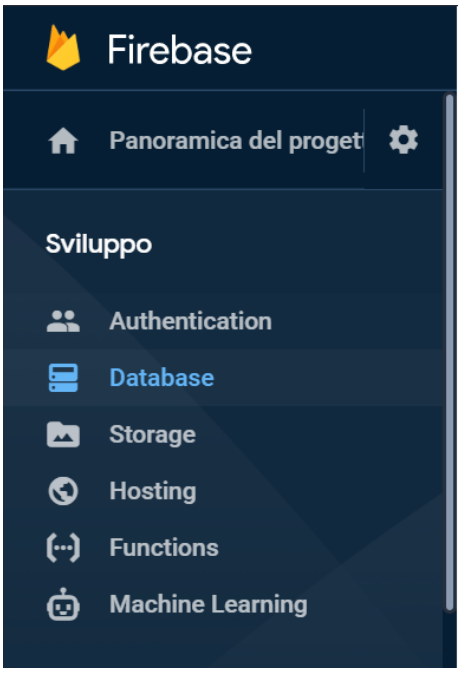

Figura 4.3. Menù dei servizi offerti da Firebase

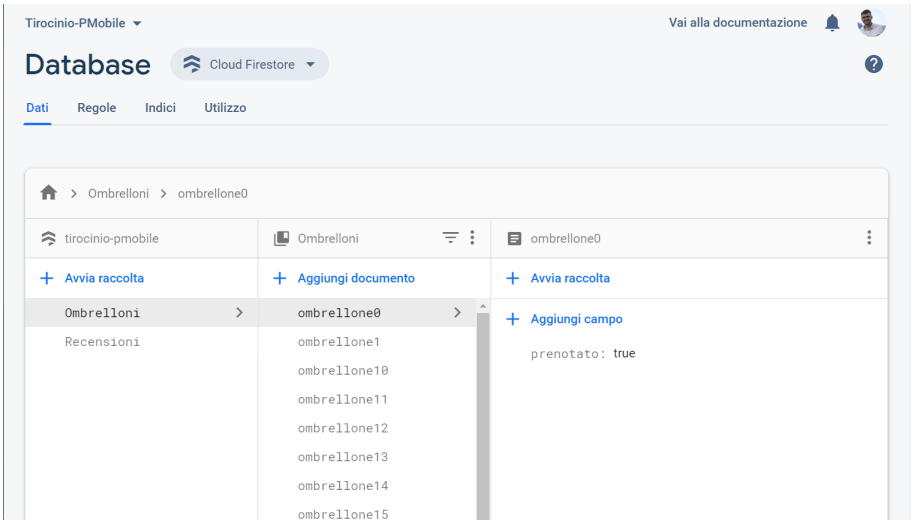

Figura 4.4. Struttura del database implementato tramite Firebase

Il servizio che è stato utilizzato è "Cloud Firestore", il quale permette aggiornamenti in tempo reale, query potenti e scalabilità automatica.

4.3 Manuale Utente

4.3.1 Creazione di un'account

Dopo aver installato l'app sul dispositivo, l'utente potrà accedere ad essa premendo sull'icona presente nel menu dello smartphone (Figura 4.5).

Figura 4.5. Icona visibile dal menù del dispositivo

In seguito all'apertura, all'utente verrà presentata la schermata di "Benvenuto". Da questa pagina egli può cliccare sull'icona relativa al "profilo" nella barra di navi-

gazione e spostarsi nella relativa schermata in cui potrà procedere con la creazione di un account. A questo punto l'utente troverà di fronte a sè due pulsanti ovvero "Registrati/Accedi" e "Disconnetti", come mostrato in Figura 4.6.

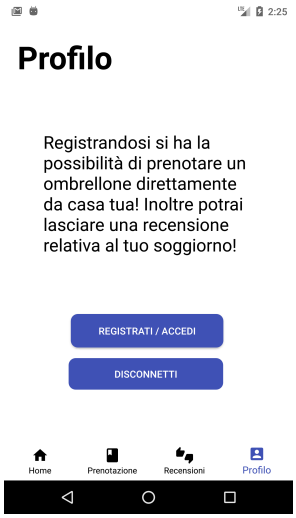

Figura 4.6. Schermata relativa alla sezione Profilo

Sia per la creazione dell'account che per l'autenticazione (in caso si fosse già registrati), l'utente dovrà cliccare sul pulsante "Registrati/Accedi". Facendo ciò egli vedrà apparire una schermata (Figura 4.7) in cui vengono mostrate le due opzioni per la registrazione/autenticazione.

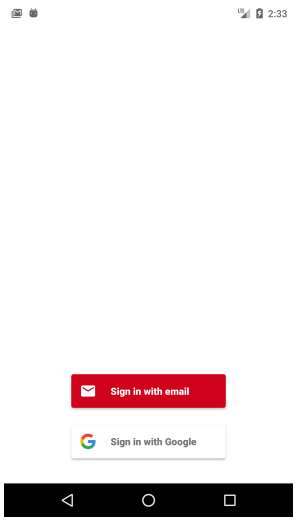

Figura 4.7. Opzioni per la creazione di un account ed autenticazione

L'utente una volta selezionata una delle due opzioni, nel caso non avesse ancora un account, procederà al completamento della registrazione inserendo i dati richiesti. Nel caso, invece, dell'autenticazione, sarà sufficiente inserire le proprie credenziali e si verrà riconosciuti dal sistema.

Per disconnettersi basterà cliccare sull'apposito pulsante "Disconnetti".

4.3.2 Prenotazione di un ombrellone

Per la prenotazione di un ombrellone l'utente dovrà spostarsi nell'apposita schermata "Prenotazione" (Figura 4.8) muovendosi tramite la barra di navigazione.

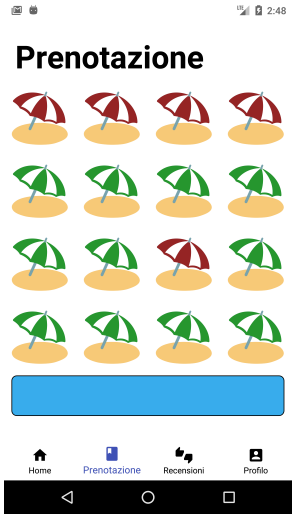

Figura 4.8. Schermata per effettuare una prenotazione

Le icone verdi indicano gli ombrelloni liberi mentre le icone rosse denotano gli ombrelloni già prenotati.

L'utente, quindi, una volta cliccato su di un ombrellone disponibile, vedrà comparire una finestra di dialogo (Figura 4.9) in cui gli verrà richiesto di confermare la prenotazione. Fatto ciò l'ombrellone selezionato cambierà colore e, tramite un `tpast` (notifica di forma rettangolare nera che appare nella parte bassa del display), egli riceverà conferma dell' avvenuta prenotazione.

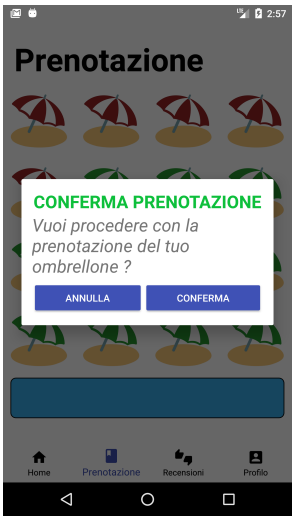

Figura 4.9. Finestra di dialogo per la conferma della prenotazione

4.3.3 Inserimento di una recensione

Per l'inserimento di una recensione, l'utente dovrà muoversi nell'apposita sezione "Recensioni" (Figura 4.10) tramite la barra di navigazione.

Figura 4.10. Schermata per l'inserimento di una recensione

L'utente, una volta cliccato il pulsante blu di forma circolare posto in basso a destra, vedrà apparire una finestra di dialogo (Figura 4.11) in cui dovrà inserire la valutazione, tramite l'apposita "Rating Bar", e il testo relativo alla sua esperienza.

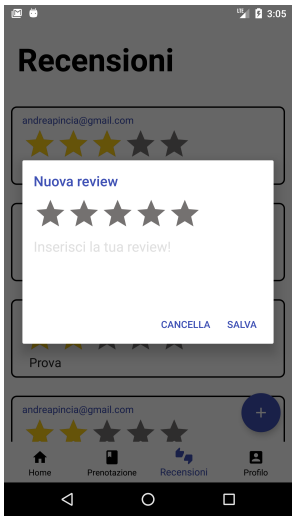

Figura 4.11. Finestra di dialogo per l'inserimento di una recensione

Per concludere l'inserimento l'utente cliccherà sul pulsante "Salva", situato nella parte bassa della finestra di dialogo; anche in questo caso verrà visualizzato un `toast` che darà conferma dell' avvenuto inserimento. A questo punto, sarà visibile insieme a tutte le altre già presenti nella schermata.

5

Discussione in merito al lavoro svolto

In questo capitolo verrà effettuata una valutazione dell'esperienza avuta e verrà proposta una panoramica sulle nozioni apprese durante lo sviluppo del progetto. Verrà presentata, infine, la SWOT analysis che ci consentirà di effettuare una valutazione del sistema realizzato.

5.1 Nozioni apprese sull'utilizzo di Android Studio e Firebase

La realizzazione di questo progetto ha comportato un grande lavoro, sia dal punto di vista progettuale che, sopratutto, implementativo.

Per quanto riguarda la progettazione sono state approfondite tematiche come, per esempio, l'analisi dei requisiti, i digrammi UML, la mappa dell'applicazione, i mockup. Tali elementi fanno parte di tutta la serie di studi che vengono fatti a monte dello sviluppo di un'applicazione. È stato, quindi, molto importante realizzare il progetto da zero poiché ha permesso di comprendere come avviene lo sviluppo di un progetto in una situazione reale.

Per quanto riguarda l'implementazione, diversi sono stati i problemi riscontrati. L'utilizzo dell'IDE Android Studio ha permesso di acquisire le conoscenze fondamentali per lo sviluppo di un'applicazione nativa. Dall'utilizzo dell'IDE derivano, anche, le conoscenze relative alla programmazione ad oggetti e l'utilizzo del linguaggio di "markup" XML per la programmazione dell'interfaccia utente.

La connessione con "Firebase", infine, ha dato all'applicazione una controparte server di qualità in grado di fornire servizi come l'archiviazione dati e l'autenticazione. L'utilizzo di "Firebase" permette di avere un backend affidabile e già pronto all'interazione con il software, evitando allo sviluppatore problemi di carattere tecnico nello sviluppo delle componenti lato server.

5.2 SWOT analysis

L'analisi SWOT (Figura 5.1) è uno strumento per effettuare la valutazione di un progetto. Viene utilizzata per analizzare i punti di forza (Strength), di debolezza (Weaknesses), le opportunità (Opportunities) e le minacce (Threats).

Figura 5.1. Matrice dell'analisi SWOT

I punti di forza e di debolezza fanno parte degli elementi interni al progetto, mentre le opportunità e le minacce riguardano le condizioni all'esterno del progetto. Questa analisi risulta essere molto importante per poter comprendere quali sono gli aspetti positivi del progetto (punti di forza e opportunità), in modo da valorizzarli, e quali, invece, quelli negativi (debolezze e minacce), in modo da minimizzarli.

5.2.1 Punti di forza

I punti di forza sono gli aspetti del progetto che ne determinano il successo, ovvero quelle motivazioni che dovrebbero indurre un utente ad utilizzare l'app. I punti di forza individuati per il progetto sono i seguenti:

- La *funzionalità*, in quanto, per la situazione creatasi a causa del COVID-19, un progetto di questo tipo risulta essere vantaggioso sia per l'utente sia per il gestore dello stabilimento, permettendo un aumento della clientela nel rispetto delle norme vigenti.
- La *semplicità*, in quanto l'app risulta essere molto diretta ed intuitiva; ciò permette all'utente di avere una visione totale dello stabilimento in maniera efficace.
- La *grafica*, in quanto il design dell'app è studiato in modo tale da poter essere comprensibile anche dall'utente meno esperto. L'interfaccia utente, inoltre, pone molta attenzione al Material Design e si adatta alla totalità dei dispositivi presenti in commercio.

5.2.2 Punti di debolezza

I punti di debolezza rappresentano l'aspetto contrario rispetto ai punti di forza, cioè sono aspetti del progetto che possono limitarne il successo; di conseguenza devono essere minimizzati il più possibile. I punti di debolezza riscontrati nel progetto sono i seguenti:

- La presenza di molte immagini di alta qualità potrebbe rappresentare un problema nell'utilizzo dell'app in dispositivi meno recenti.
- Per accedere alla funzionalità più importanti, come la prenotazione, l'inserimento delle recensioni e la creazione di un profilo, è necessario che l'utente disponga di una connessione dati. L'assenza di una connessione implica una possibilità limitata di utilizzo dell'app da parte dell'utente.

5.2.3 Opportunità

Le opportunità rappresentano i vantaggi che possono venire dall'esterno, cioè, per esempio, sviluppo tecnologico, nuove normative, etc. Più in generale, sono tutti quegli aspetti esterni al progetto, ma da cui il progetto può trarne vantaggio. Una possibile opportunità deriva proprio dalla situazione creatasi con il COVID-19. In un prossimo futuro, per evitare di ripetere errori commessi con questa pandemia, applicazioni di questo tipo potrebbero diventare lo standard per ogni azienda che lavora a stretto contatto con il pubblico.

5.2.4 Minacce

Una minaccia è un fattore esterno che potrebbe limitare o danneggiare l'applicazione.

Poiché l'applicazione gestisce l'archiviazione dati e l'autenticazione mediante un servizio in cloud (Firebase), un malfunzionamento della piattaforma potrebbe portare al quasi completo inutilizzo delle principali funzioni che l'app fornisce.

Per quanto riguarda la sicurezza è, invece, molto importante che lo sviluppatore definisca le regole per la scrittura e lettura dei dati nel database Firebase in modo tale che solo un utente autenticato possa usufruire di tali funzionalità.

6

Conclusioni

In questa tesi è stato descritto il progetto MyBeachResort, che consiste nella progettazione e realizzazione di un'app per la gestione di uno stabilimento balneare.

Prima di tutto è stata presentata una panoramica sull'emergenza epidemiologica in Italia. Si è passato, poi, ad una breve introduzione del sistema operativo Android, fornendo alcuni dati storici e presentandone l'architettura e l'ambiente di sviluppo.

Nel Capitolo 3 è stata fatta l'analisi dei requisiti, illustrando le funzionalità da implementare e i vincoli da rispettare, per poi passare alla fase di progettazione, in cui sono stati presentati i diagrammi dei casi d'uso, la mappa dell'applicazione ed i mockup, utili per capire il funzionamento dell'app.

Il Capitolo 4 ha trattato l'implementazione ed il manuale utente. Si è parlato, quindi, della struttura dell'app, illustrandone le parti che la compongono ed i listati con relativa spiegazione. Per quanto riguarda il manuale utente, invece, è stata presentata una guida ben dettagliata in modo tale che un utente riesca a svolgere le principali funzionalità offerte dall'app.

Infine, nel Capitolo 5, sono state presentate le nozioni acquisite durante lo sviluppo del progetto, per concludere con un analisi che mostra una panoramica generale dei vantaggi e degli svantaggi dell'elaborato.

Questo progetto nasce con lo scopo di migliorare l'esperienza tra cliente e stabilimento balneare, permettendo la visualizzazione in remoto dei servizi offerti ed una maggiore praticità per la prenotazione di un ombrellone. Con lo sviluppo di quest'app si è cercato, inoltre, di aiutare coloro che lavorano nel settore della balneazione nel poter riprendere l'attività adeguandosi alle norme anti contagio dovute all'epidemia. La realizzazione del progetto è stata molto utile per comprendere come avviene l'implementazione di un' applicazione nativa Android e l'utilizzo del servizio in cloud Firebase.

L'app, sebbene sia funzionante, può essere migliorata in diversi aspetti; in particolare, al momento l'app è progettata per un solo stabilimento balneare, si potrebbe fare in modo che, in futuro, tramite l'applicazione, possono essere gestiti più stabilimenti balneari. Si potrebbero aumentare i metodi per la creazione di un account (ad esempio gestendo l'accesso tramite Facebook o altri social media). Si potrebbe, infine, implementare un metodo di pagamento dopo la conferma della prenotazione, in modo da rendere l'intera operazione completamente eseguibile da remoto.

Riferimenti bibliografici

1. Android. `https://it.wikipedia.org/wiki/Android`, 2020.
2. Android Developer: Android Studio. `https://developer.android.com/studio/intro`, 2020.
3. COVID-19. `https://it.wikipedia.org/wiki/Pandemia_di_COVID-19_del_2019-2020`, 2020.
4. Firebase. `https://it.wikipedia.org/wiki/Firebase`, 2020.
5. Smartphone. `https://it.wikipedia.org/wiki/Smartphone`, 2020.
6. B. Cruz Zapata. *Android Studio Application Development*. Packt Publishing, 2013.
7. Guglielmo Briscese, Nicola Lacetera, Mario Macis, and Mirco Tonin. Compliance with covid-19 social-distancing measures in italy: the role of expectations and duration. Technical report, National Bureau of Economic Research, 2020.
8. C. De Sio Cesari. *Manuale di Java 8: Programmazione orientata agli oggetti con Java standard edition 8*. Hoepli, 2014.
9. M. Carli. *Android 9. Guida per lo sviluppatore*. Apogeo, 2017.
10. E. Cisotti and M. Giannino. *Android. Guida completa*. Edizioni LSWR, 2015.
11. F. Collini, M. Bonifazi, A. Martellucci, and S. Sanna. *Android: Programmazione avanzata*. Edizioni LSWR, 2015.
12. A. Lorenzi e A. Rizzi. *Java. Programmazione ad ogetti e applicazioni Android*. Atlas, 2013.
13. G. Grandinetti. *Android studio. Sviluppare vere applicazione Android partendo da zero: 2*. Edizionefutura.com, 2016.
14. T. Hagos. *Learn Android Studio 3: Efficient Android App Development* . Apress, 2018.
15. C. Horstmann. *Concetti di Informatica e fondamenti di Java (IV Edizione)*. Apogeo S.R.L., 2007.
16. Mohd Javaid, Abid Haleem, Raju Vaishya, Shashi Bahl, Rajiv Suman, and Abhishek Vaish. Industry 4.0 technologies and their applications in fighting covid-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, 2020.
17. Leonardo López and Xavier Rodó. The end of social confinement and covid-19 re-emergence risk. *Nature Human Behaviour*, 4(7):746–755, 2020.
18. G. Nudelman. *Android Design Patterns: Interaction Design Solutions for Developers*. Wiley, 2013.
19. H. Schildt. *Java. La guida completa*. McGraw-Hill Education, 2012.
20. MD[1] Shalini Shah, MD Sudhir Diwan, MD Lynn Kohan, MD David Rosenblum, MD Christopher Gharibo, MD Amol Soin, and MD Adrian Sulindro. The technological impact of covid-19 on the future of education and health care delivery. *Pain Physician*, 23:S367–S380, 2020.
21. A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2011.

Ringraziamenti

Con questa tesi sono giunto alla fine del mio percorso formativo, durante il quale non poche sono state le difficoltà incontrate.

Innanzitutto, vorrei ringraziare i componenti della mia famiglia che, attraverso i loro sacrifici, mi hanno permesso di svolgere questo cammino con estrema serenità. Sono sempre riusciti a darmi sostegno nei momenti di sconforto, fornendomi la motivazione necessaria per non arrendermi.

Voglio fare un ulteriore ringraziamento a tutti gli amici del mio paese, Sant'Angelo in Pontano, per aver fatto il tifo da lontano per me. Ringrazio, inoltre, tutte le persone che ho incontrato in questo viaggio con cui ho passato momenti di studio e di svago. Ringrazio tutti i coinquilini con cui sono stato, in particolare quelli di casa Rossini.

Vorrei ringraziare di cuore la mia ragazza Alessia, che nelle insicurezze è riuscita sempre ad infondermi calma e tranquillità.

Faccio un sentito ringraziamento al mio compagno di corso Ettore, con cui ho condiviso un'esperienza formativa dal primo giorno di liceo ad oggi.

Infine, un enorme ringraziamento va al mio relatore, il Prof. Domenico Ursino. Lo ringrazio per avermi fatto appassionare al suo corso, il che mi ha conseguito una notevole crescita professionale per il domani.