UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Progettazione e implementazione di un'app Android per la recensione delle spiagge libere nelle Marche.

Design and implementation of an Android app for the review of free beaches in the Marche region

Candidato:
Luca Timperio

Relatore:
Prof. Domenico Ursino

Anno Accademico 2021-2022

Indice

Elenco delle figure

Capitolo 1

Introduzione

Lo scopo della nostra tesi è la creazione di un'applicazione Android tramite un'IDE e un software per la gestione dei database, utilizzando il linguaggio di programmazione Java.

L'applicazione ha come obiettivo la descrizione e recensione delle spiagge libere del litorale marchigiano.

L'IDE che utilizzeremo è Android Studio[1], il quale oltre che di Java, permette la codifica anche di altri linguaggi di programmazione.

Il sistema operativo Android viene utilizzato prevalentemente su dispositivi come smartphone o tablet, che sono dotati di touchscreen, anche se è possibile trovarlo anche su altri dipositivi, come le televisioni.

Con la riapertura delle attività dopo la pandemia, uno strumento in grado di raccogliere e descrivere informazioni sulle varie località balneari, come la nostra applicazione, può essere di grande aiuto per i turisti che stanno organizzando le proprie vacanze.

Le applicazioni che troviamo frequentemente sul web riguardano prevalentemente stabilimenti balneari privati, dove si è in grado di prenotare il proprio ombrellone con la sezione relativa alle recensioni

Infatti non sono molte le applicazioni che troviamo nei digital store o sul web che si occupano direttamente di spiagge libere, molto probabilmente ciò è legato al fatto che nessuna Pubblica Amministrazione e nessun privato vogliano finanziare questo tipo di applicazioni.

Possiamo trovare pero, alcune applicazioni sponsorizzate direttamente dalle amministrazioni comunali di alcune città, che si occupano di descrivere tutte le varie attività che si possono effettuare, e tra queste si trovano anche spiagge libere.

Quindi, un fattore importante che ha influenzato in maniera significativa la decisione di implementare la nostra applicazione è stata la scarsa presenza sul mercato di applicazioni con uguali finalità.

Tutti questi fattori che ci hanno portato all'implementazione della nostra applicazione verranno approfonditi nel capitolo sull'analisi SWOT che è la base per il marketing efficiente, ed è utilizzata dalle aziende per prendere delle decisioni sulle strategie di vendita e analisi di mercato per poter raggiungere gli obiettivi di vendita.

Analizzeremo le varie caratteristiche di forza, debolezza, opportunità e minacce del nostro programma dal punto di vista del mercato economico, valutando i processi interni ed esterni della nostra applicazione.

Prima dell'implementazione dell'applicazione, abbiamo sfruttato diversi strumenti, quali diagrammi dei casi d'uso, diagrammi delle attività e mock-up, che ci hanno permesso di rappresentarne la struttura logica, cosi da facilitarci il lavoro di codifica in seguito.

Uno degli obiettivi principali nell'implementare l'applicazione ha riguardato la facilità di utilizzo.

Con questo fattore in mente abbiamo iniziato con una facile programmazione dell'operazione di registrazione e accesso alla nostra applicazione, tramite le funzionalità di Authentication e Database di Firebase, utilizzando la propria e-mail e una password.

Firebase è un back-end service fornito da Google per gli svilupatori, che fornisce diverse funzionalità, come, ad esempio, un front-end data store nel cloud che permette lo sviluppo di applicazioni con avanzate funzionalità

La home page è costituita da una lista delle varie spiagge libere marchigiane con relative descrizioni e immagini che permettono all'utente di effettuare una prima scelta della spiaggia che si vuole visitare; successivamente con pochi click dell'apertura dell'applicazione, l'utente è in grado di individuare la spiaggia desiderata.

A questo punto, una volta aperta la pagina dedicata alla spiaggia di interesse, ci si trova di fronte ad una descrizione più dettagliata con l'immagine e una lista di recensioni di altri utenti.

Inoltre, in tale pagina, è prevista la possibilità, per ogni utente, di lasciare una propria recensione basata sull'esperienza visitando tale spiaggia.

Per rendere possibile la maggior parte delle funzionalità della nostra applicazione, come vedremo in seguito, abbiamo utilizzato Firebase Console, che ci permette non solo di costruire un database dinamico sia per le spiagge, con relative descrizioni, immagini e recensioni lasciate dagli utenti, e sia per registrare gli utenti che si iscrivono alla nostra applicazione.

Avendo implementato un'applicazione in grado di descrivere e recensire determinati luoghi, in futuro sarà possibile utilizzarla come base per la codifica di altre applicazioni con obiettivi simili.

Ad esempio, tramite relativi accorgimenti sul codice del programma, non sarà difficile implementare un'applicazione in grado di gestire stabilimenti balneari privati, in grado di prenotare ombrelloni, sdraio e lettini con l'opzione di recensione di tale località

Possiamo implementare anche altri tipi di applicazioni utilizzando la nostra come base; infatti, dato che l'obiettivo del nostro programma è la descrizione e recensione di luoghi, possiamo adattarla per altre tipologie di destinazioni, come, ad esempio, catene montuose e relativi percorsi per scalare le varie cime, oppure località sciistiche per i periodi invernali.

Capitolo 2

Android

2.1 Introduzione al software

Android è un sistema operativo per dispositivi mobili basato su una versione modificata del kernel di Linux e di altri open source software. Principalmente viene usato su dispositivi abilitati con touchscreen quali smartphone e tablet. Il suo codice sorgente è stato utilizzato su vari dispositivi, quali camere digitali, PC e game console, ognuna con, ovviamente, un'interfaccia diversa.

2.1.1 Android Studio

Android Studio[2] è l'IDE (Integrated Development Environment) ufficiale per il sistema operativo Android basato sul software IntelliJ IDEA di JetBrains e progettato specificamente per lo sviluppo Android esso è disponibile per il download su Windows, macOS e Linux. Per lo sviluppo di applicazioni nel sistema operativo Android, Android Studio utilizza un sistema di compilazione basato su Gradle, un emulatore, un template di codice e l'integrazione di GitHub. Android[3] Studio utilizza una funzione Instant Push per inviare modifiche di codice e risorse a un'applicazione in esecuzione. Un editor di codice assiste lo sviluppatore nella scrittura del codice e nell'offrire completamento, rifrazione e analisi del codice.

2.1.2 Installazione e preparazione

Inizialmente bisogna scaricare il software direttamente dal sito Android. Una volta installato sul proprio dispositivo, si procede alla preparazione dell'IDE.

Come possiamo vedere nella figura 2.1, abbiamo, da una parte, la lista dei nostri progetti implementati nell'applicazione e nella parte principale, una lista di opzioni. Tra le varie opzioni si può scegliere quella di aprire un progetto esistente o di importarlo dal nostro dispositivo, nonchè, ovviamente, quella di iniziare un nuovo progetto. Quindi creiamo un nuovo progetto e accediamo alla schermata in Figura 2.2:

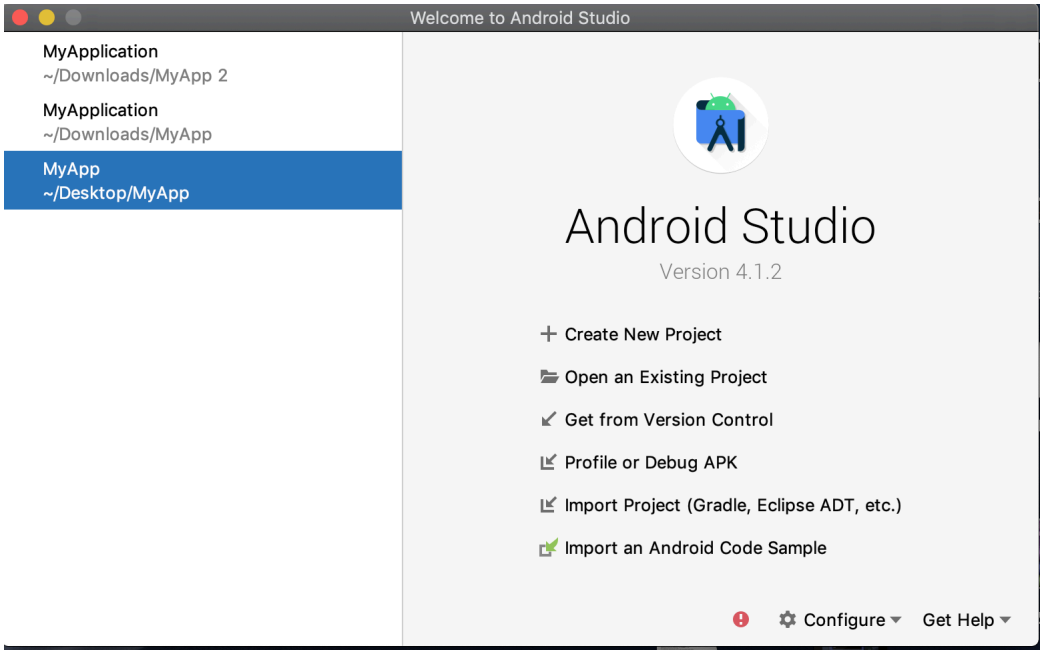

Figura 2.1: Schermata iniziale di Android Studio

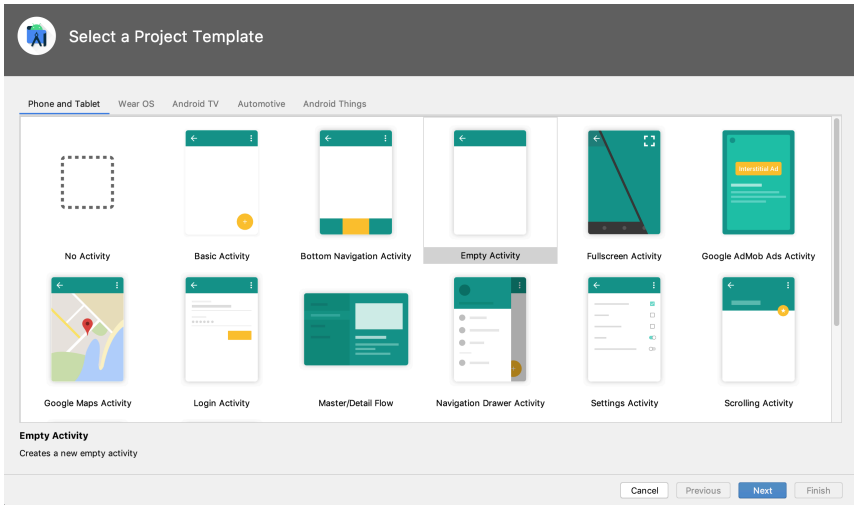

Figura 2.2: Template del Progetto

Qui ci viene presentata una serie di scelte in base al tipo di applicazione che vogliamo implementare. Inizialmente, come vediamo, possiamo implementare applicazioni per diversi dispositivi: smarphone, smartwatch,Tv Android etc. Una volta effettuata tale scelta, che, nel caso della nostra applicazione riguarderà smartphone e i tablet, ci viene chiesto di effettuare un'altra scelta sul tipo di interfaccia che debba avere la nostra activity. Android[4] gestisce questo tipo di interfacce in maniera diversa, per questo conviene avere un'interfaccia vuota in modo da avere il controllo sul nostro codice.

Ora Android Studio[5] ci chiede il nome del nostro progetto, del boundle e il tipo di linguaggio di programmazione che si vuole utilizzare per la codificazione della nostra applicazione. Noi abbiamo optato per il linguaggio Java che approfondiremo seguito. Poi è necessario scegliere l'API minima che supporterà la nostra applicazione. .evviamente più aumenteremo il livello dell'API, minori saranno i dispositivi su cui la nostra applicazione sarà supportata come si vede in figura 2.3.

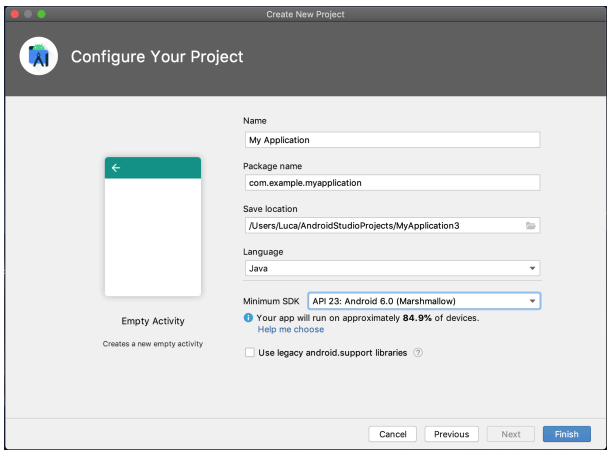

Figura 2.3: Configurazione del progetto

Nella Figura 2.4 vediamo come, una volta terminata la configurazione, arriviamo alla schermata della struttura, dove visualizziamo il nostro programma. Abbiamo, come primo elemento, il modulo "app" suddiviso in 3 cartelle: `Manifest`, `java`, `res`. Per quanto riguarda la cartella `Manifest`, al suo interno troveremo, appunto, il file `AndroidManifest.xml` definisce le proprietà della nosta applicazione, tra le proprietà definite in tale file abbiamo: Il package, ovvero l'ID univoco dell' applicazione; un tag "application" con all'interno, le varie proprietà della nostra applicazione, quali la forma dell'icona, il nome, il tema, e l'elenco delle nostre activity con alcune proprietà, come, ad esempio, la definizione della nostra main activity o di quale activity è "parente" di un'altra. Poi, nella cartella `java`, abbiamo i file della `MainActivity`, nella quale, durante l'implementazione, andremo a inserire le varie activity che andranno a comporre la nostra applicazione.L'ultima cartella è quella delle risorse o `res`, dove abbiamo varie sottocartelle che conterranno file molto importanti per l'implementazione. In questa cartella andremo a immettere i layout delle varie activity, le immagini che vogliamo inserire, come, ad esempio, il logo dell'applicazione, o values, dove possiamo inserire le stringhe, i colori o gli stili.

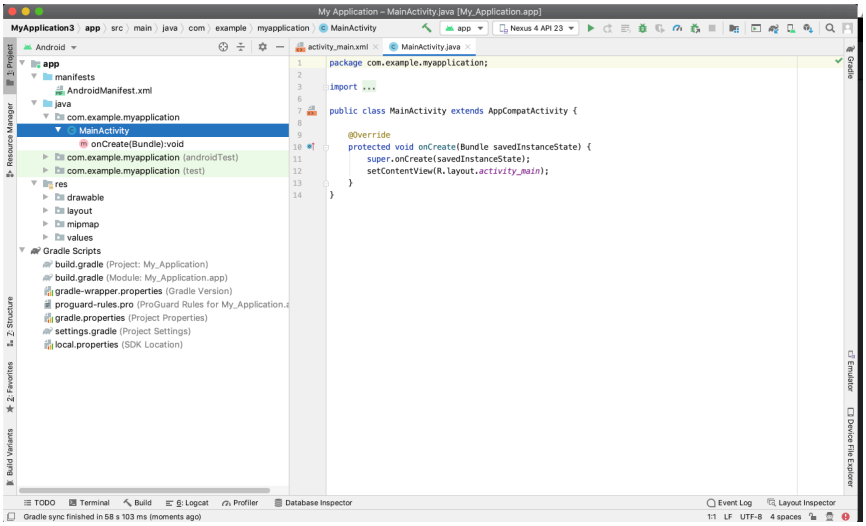

Figura 2.4: Struttura del progetto

2.1.3 Caratteristiche Software

Android Studio è un IDE che ha acquisito sempre più valore negli ultimi anni anche grazie alle varie feature che esso prevede.

Linguaggi di Programmazione

Come abbiamo visto in precedenza, AndroidStudio ci dà la possibilità di scegliere tra vari linguaggi di programmazione per la codificazione della nostra applicazione, tra i quali abbiamo Kotlin e Java. Nel nostro caso si è utilizzato il linguaggio Java. Java è un linguaggio di programmazione avanzato orientato agli oggetti, progettato per avere indipendenza dall'hardware su cui si andrà a eseguire il nostro codice, infatti uno, dei principi fondamentali di questo linguaggio è "write once, run anywhere", ovvero "scrivi una volta sola e esegui ovunque". Una volta scritto il nostro codice, esso non dovrà essere modificato se cambia il dispositivo su cui si vuole eseguirlo.

Emulatore (AVD Manager)

L'emulatore all'interno di Android Studio è una delle feature più importanti dell'IDE, infatti esso ci permette di testare le nostre applicazione su vari dispositivi con API di livelli diversi senza dover prendere il corrispettivo dispositivo fisico. L'Emulatore è in grado di simulare quasi tutte le caratteristiche di un dispositivo, infatti esso è in grado di simulare chiamate in entrata e uscita, come pure la rotazione del dispositivo,in modo da testare se la nostra applicazione si adatta alle varie grandezze di schermo e molto altro. Andiamo ora a vedere come impostare un dispositivo che ci permetterà di testare la nostra applicazione. Una volta aperto l'AVD manager, ci troveremo di fronte alla schermata in Figura 2.5.

Qui possiamo vedere una lista dei dispositivi già creati e l'opzione di crearne uno nuovo. Vediamo ora come crearne uno da zero.

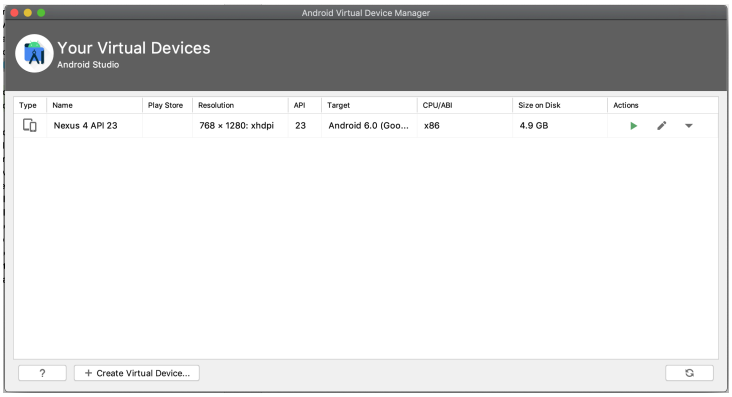

Figura 2.5: Creazione di un nuovo dispositivo

Nella Figura 2.6 vediamo come scegliere il tipo di dispositivo che vogliamo simulare; a sinistra abbiamo un elenco di tipologie di dispositivi, dagli smartphone alle tv etc, (come abbiamo già detto noi ci occuperemo di smartphone e tablet). Quindi ora abbiamo una serie di dispositivi ciascuno con la propria grandezza e la propria risoluzione dello schermo, l'icona che ci permette di individuare i dispositivi che suppportano le librerie di Google Play, in particolare quella del Play Services. Una volta effettuata questa scelta, ci troveremo di fronte alla schermata in Figura 2.7.

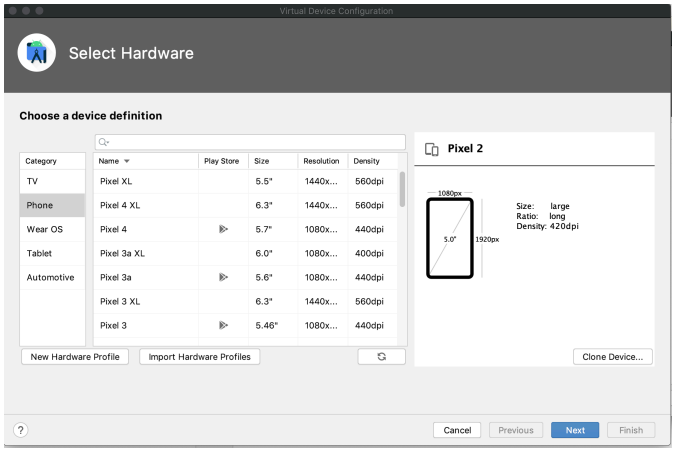

Figura 2.6: Scelta del dispositivo da simulare

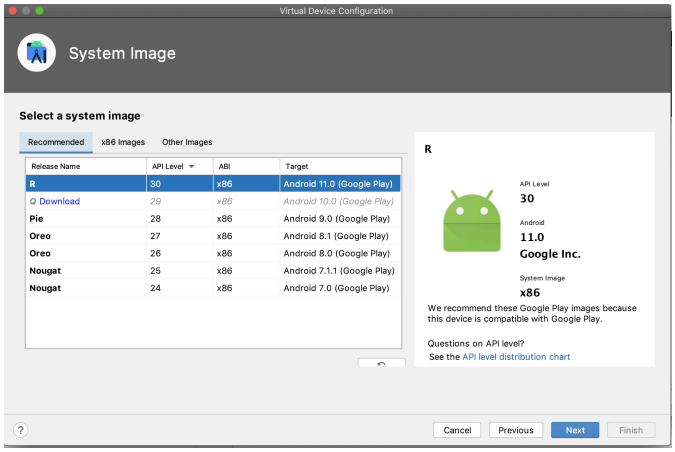

Figura 2.7: Scelta dell'API minima che dovrà essere supportata dal dispositivo

Qui potremo decidere l'API minima che dovrà supportare il nostro dispositivo. In base alla nostra scelta ci verrà permesso di scaricare le opportune librerie. Una volta fatto ciò e dato un nome al nostro emulatore, il programma è pronto per essere utilizzato.

Layout Editor

Il layout editor è uno degli strumenti fondamentali per la costruzione dei layout delle varie schermate della nostra applicazione. Esso, oltre alla codifica manuale di file XML di tali layout, ci permette di trascinare elementi per la creazione dell'interfaccia in un editor per la progettazione visiva del layout. Andiamo, ora, a vedere le varie features di quest'editor nella Figura 2.8.

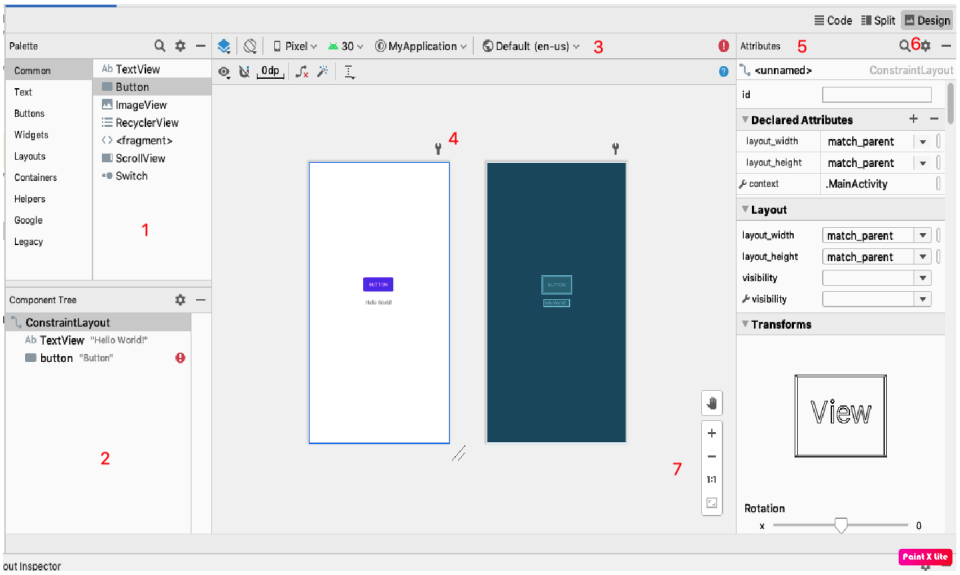

Figura 2.8: Il Layout Editor

1. *Palette*: contiene views che possiamo trascinare nel nostro layout.

2. *Component Tree*: mostra la gerarchia dei componenti del layout.

3. *Toolbar*: ci permette di configurare la parte visiva del layout e cambiare i suoi atributi.

4. *Attributes*: mostra i vari attributi del layout.

5. *View mode*: abbiamo la possibilità di visualizzare il nostro layout in modalità codice,modalità design o visiva o modalità split che mostra sia la modalità codice che quella del design.

6. *Zoom e pannello di controllo*: ci permette di controllare la grandezza e la posizione che avrà il nostro layout nell'editor.

Firebase

Firebase[6] è è un data store in tempo reale di default, focalizzato sul front-end, che esiste nel cloud, i clienti possono connettersi direttamente ad esso.
Firebase è un backend-as-a-Service (Baas). Fornisce agli sviluppatori una varietà di strumenti e servizi per aiutarli a sviluppare app di qualità, ed è costruito sull'infrastruttura di Google. Altri strumenti di Firebase sono i seguenti:

1. *Autenticazione*: supporta l'autenticazione degli utenti utilizzando password, numeri di telefono, Google, Facebook, Twitter e altro ancora. Firebase Authentication (SDK) può essere utilizzato per integrare manualmente uno o più metodi di accesso in un'app.

2. *Database in tempo reale*: i dati vengono sincronizzati tra tutti i client in tempo reale e rimangono disponibili anche quando un'applicazione va offline.

3. *Hosting*: Firebase Hosting fornisce un hosting veloce per una applicazione web.

Capitolo 3

Descrizione del problema e analisi dei requisiti

3.1 Descrizione del problema

Nella regione ci sono diverse spiagge libere che si estendono per un totale di 50km. In questo periodo di pandemia, data la loro estensione e l'obbligo di distanziamento sociale tra le varie postazioni, è stata data la possibilità ai bagnanti di godersi le spiagge e il mare, in una certa sicurezza. Da queste esigenze, nasce la nostra idea di creare un'applicazione che vuole descrivere e recensire queste spiagge, in modo da aiutare i bagnanti nella scelta di dove trascorrere le loro vacanze estive senza troppe preoccupazioni. I problemi che ci si pongono davanti sono molteplici; siamo partiti dalla classificazione ed elencazione delle varie spiagge libere in modo da avere un database iniziale su cui si baserà la nostra applicazione. L'utente che usufruisce della nostra applicazione, una volta effettuato l'accesso tramite la propria mail, avrà la possibilità di scegliere tra le varie spiagge nel nostro catalogo. Una volta effettuata tale scelta, egli avrà la possibilità di scorrere tra le recensioni lasciate da altri utenti e di lasciarne una, se lo vorrà. La attività CRUD dei vari item delle applicazioni, quali spiagge, utenti e recensioni, viene lasciata all'amministratore e dovrà essere effettuata in back-end tramite le varie funzionalità fornite dalla console di Firebase.

3.2 Requisiti dell'applicazione

In questa sezione elencheremo e spiegheremo i vari requisiti che la nostra applicazione dovrà rispettare. Possiamo dividere tale requisiti in due categorie, ovvero funzionali e non funzionali. In particolare:

1. *Requisiti funzionali*: sono servizi o funzionalità offerti dalla nostra applicazione.
2. *Requisiti non funzionali*: rappresentano i vincoli e le caratteristiche di qualità del sistema.

3.2.1 Requisiti funzionali

I requisiti funzionali della nostra applicazione sono i seguenti:

1. *Visualizzazione delle spiagge e delle recensioni degli utenti*
2. *Scrittura delle recensioni delle spiagge da parte dell'utente*
3. *Sign in/ Sign up da parte dell'utente*
4. *Attività CRUD relative agli utenti, alle spiagge e alle recensioni da parte dell'amministratore*

3.2.2 Requisiti non funzionali

Nella nostra applicazione avremo i seguenti requisiti non funzionali:

1. E'necessario un database di supporto per la gestione di utenti, spiagge e recensioni.
2. Per poter usufruire dell'applicazione bisogna registrarsi tramite la propria mail.
3. L'utente potrà visualizzare la descrizione di una spiaggia e le relative recensioni solo dopo aver scelto dalla lista la spiaggia desiderata.
4. L'utente potrà inserire una recensione solo dopo aver scelto la spiaggia.

3.3 Attori e casi d'uso

In questa sezione individueremo gli attori e i casi d'uso relativi ad essi. Un attore è un'entità (utente, admin) che interagisce col sistema di riferimento. Le azioni che vengono effettuate da questi attori vengono chiamate "casi d'uso", che vengono descritte nel diagramma dei casi d'uso.

3.3.1 Diagrammi dei casi d'uso

Prendendo in considerazione gli attori e le funzionalità dell'applicazione procediamo con la costruzione dei diagrammi dei casi d'uso. La Figura 3.1 riporta il diagramma dei casi d'uso degli utenti mentre nella Figura 3.2 troviamo il diagramma dei casi d'uso dell'amministratore.

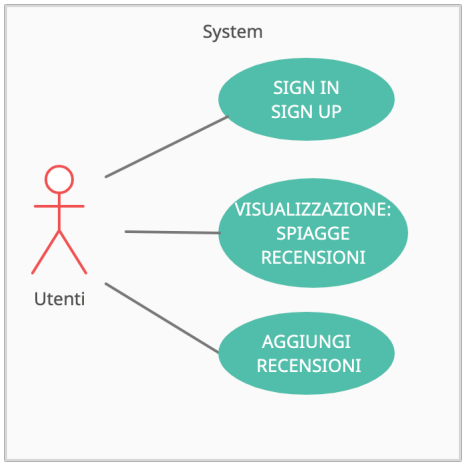

Figura 3.1: Diagramma dei casi d'uso degli utenti

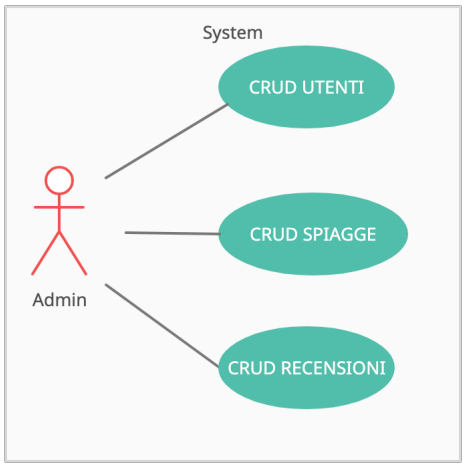

Figura 3.2: Diagramma dei casi d'uso degli amministratori

Capitolo 4

Progettazione dell'applicazione

4.1 Premessa

Normalmente ci viene da considerare che il modo migliore per lo sviluppo di un'applicazione viene determinato dalla scelta della piattaforma di sviluppo, ovviamente questa decisione è molto importante ma se consideriamo applicativi mobili, il fattore più importante è fornire la migliore fruizione possibile agli utenti. Quindi per progettare un'applicazione mobile che rispetti questo criterio, prendiamo in considerazione diverse proprietà fondamentali che un'applicazione mobile deve avere, in particolare, essa deve essere:

1. *Utile: è in grado di soddisfare i bisogni degli utenti?*
   Un'applicazione mobile deve fornire servizi e funzionalità che consentono all'utente di arrivare ai propri obiettivi.

2. *Utilizzabile: quanto facilmente è possibile raggiungere i vari obiettivi?*
   Ovviamente la facilità con cui si raggiunge un determinato obiettivo è importantissima, in quanto i dispositivi mobili sono strumenti attraverso i quali si è in grado di fornire servizi di grande utilità per un utente "on the go", che deve essere in grado di raggiungere tali obiettivi facilmente e nella minore velocità possibile. Un altro fattore che aggiunge importanza alla facilità con cui si arriva ad un obiettivo è, ovviamente, l'esistenza di altre applicazioni con le stesse funzionalità. Infatti, se un'applicazione di una società concorrente offre gli stessi servizi, ma è più facile da utilizzare, ovviamente gli utenti (a parità di altre condizioni, come, ad esempio, il prezzo) useranno l'applicazione più semplice.

3. *Desiderabile: le esperienze passate influiscono sulla scelta di un'app?*
   Gli utenti sceglieranno un'applicazione in base anche a varie emozioni date da esperienze passate e impatti visivi che il nostro applicativo scaturisce in loro.

4.2 Diagrammi delle attività

Dall'analisi dei requisiti, e conseguentemente dal diagramma dei casi d'uso visti nel capitolo precedente, abbiamo delineato la struttura logica della nostra applicazione,

attraverso lo strumento noto come diagramma delle attività. Questo tipo di diagramma mostrerà il flusso dei passaggi che un utente si troverà di fronte quando utilizzerà la nostra applicazione. Come si può vedere in Figura 4.1, il diagramma delle attività della nostra applicazione è alquanto lineare in quanto lo è anche l'obiettivo di tale app.

Figura 4.1: Diagramma delle attività dell'applicazione

4.3 Mockup

Un altro strumento utilizzato per la progettazione di un app è quello dei mockup; questi non sono altro che delle riproduzioni grafiche di oggetti. Nel nostro caso, essi saranno pagine della nostra applicazione, ovvero rappresentazioni grafiche delle varie interfacce di un sistema informatico.
Inizialmente vediamo i mockup delle pagine per la registrazione e l'accesso alla nostra applicazione. Come si vede nelle Figure 4.2 e 4.3, le pagine sono molto simili e ognuna ha un riferimento all'altra pagina.

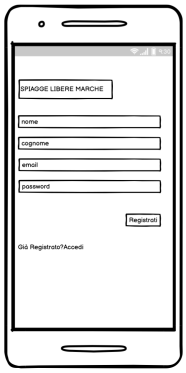

Figura 4.2: Mockup della pagina per la registrazione

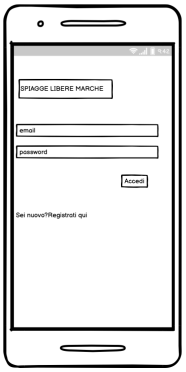

Figura 4.3: Mockup della pagina per il login

In Figura 4.4 vediamo la nostra home page, che sarà costruita con una lista delle spiagge in catalogo e un'immagine di repertorio per mostrare agli utenti una fotografia di riferimento della spiaggia in questione.

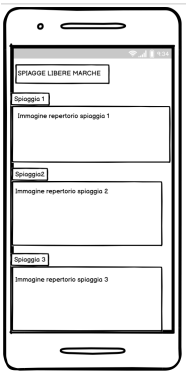

Figura 4.4: Mockup della home page

In Figura 4.5 vediamo la pagina che ci mostra la singola spiaggia, con una descrizione, un'immagine e le varie recensioni lasciate degli utenti nonchè la possibilità di lasciare una propria valutazione.

Figura 4.5: Diagramma delle attività dell'applicazione

Capitolo 5

Implementazione dell'applicazione e manuale utente

In questa sezione vedremo come abbiamo implementato le varie funzionalità della nostra applicazione. Uno strumento fondamentale è stato Firebas,e in particolare, l'utilizzo della sua console.

5.1 Firebase

Nei capitoli precedenti abbiamo già descritto le proprietà generali di Firebase, in questa sezione vedremo come la sua console è essenziale per l'implementazione di un'applicazione mobile.

5.1.1 Funzionalità implementate con Firebase

Nella Figura 5.1 vediamo la home page della console Firebase della nostra applicazione. Come si può osservare abbiamo l'Authentication mostrata in Figura 5.2 che, come suggerisce il nome, ci aiuterà nell'autenticazione nel momento della registrazione alla nostra applicazione. Qui troviamo anche il database degli utenti registrati e il metodo di sign-in che, nel nostro caso, avverrà tramite la mail. In seguito abbiamo lo storage con cui possiamo creare dei database utili nell' immagazinare dati per popolare la nostra applicazione, come possiamo vedere nella Figura 5.3.

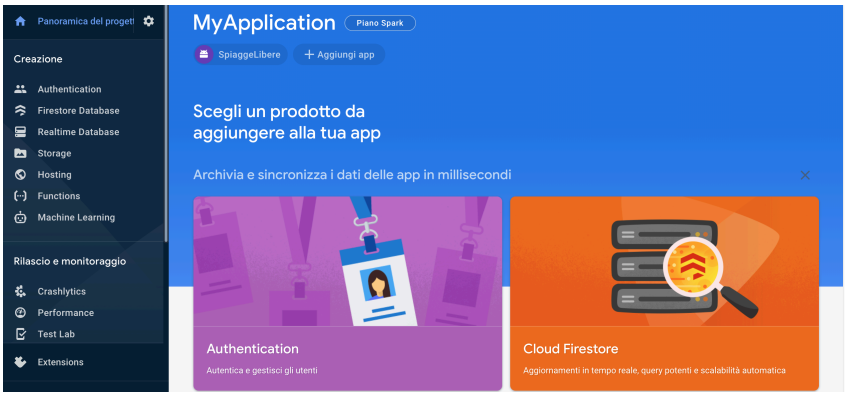

Figura 5.1: Home page della console di Firebase

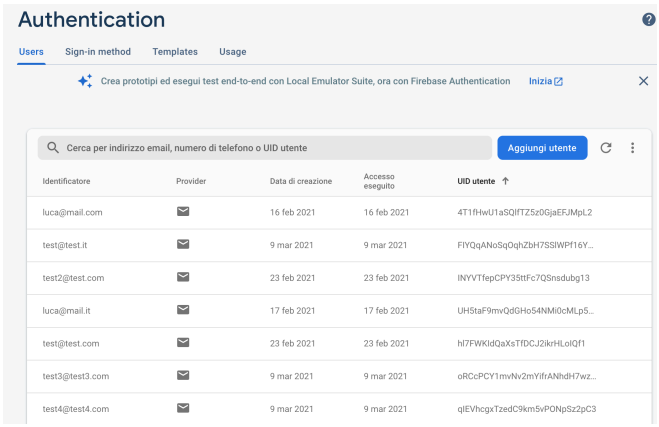

Figura 5.2: Pagina dell'Authentication della console di Firebase

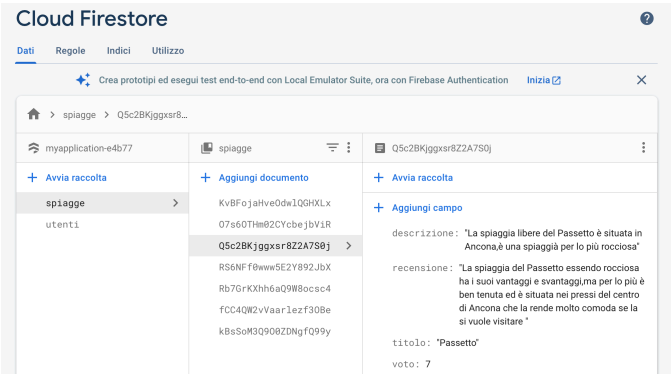

Figura 5.3: Pagina del database di Firebase

5.2 Android Studio

Vediamo ora alcune delle funzionalità della nostra applicazione implementate tramite la codificazione in Java nel nostro IDE. Come abbiamo già visto Android Studio è fondamentale grazie a tutte le proprietà intrinsiche per l'implementazione della nostra applicazione.

5.2.1 Funzionalità implementate con Android Studio

Android Manifest

Qualsiasi applicazione Android deve avere un file chiamato `AndroidManifest.xml` nella sua cartella principale.Esso contiene informazioni basilari sull'applicazione, necessarie al sistema per eseguire qualsiasi porzione di codice della stessa. Come vediamo in Figura 5.4, nel nostro Manifest abbiamo nelle righe 5, 6, 7 e 8, varie `permission`. Nella riga 5 abbiamo la `permission` per l'uso di internet da parte del dispositivo su cui la nostra applicazione viene usata; nella riga 6 ho la `permission` per l'uso della fotocamera; nella riga 7 si trova la `permission` per accedere a qualsiasi

file al di fuori delle directory specifiche dell'app sulla memoria esterna; infine nella riga 8, abbiamo la `permission` per scrivere su qualsiasi file al di fuori della directory specifica dell'app. Nelle righe succesive è presente una lista di impostazioni, come il nome della nostra applicazione, il suo logo e il suo tema. Infine, è presente, una lista delle activity e della loro gerarchia.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.myapplication">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <uses-feature
        android:name="android.hardware.camera2"
        android:required="false" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".NewRecensioneActivity"
            android:parentActivityName=".SingolaActivity"/>
        <activity android:name=".LoginActivity"
            android:parentActivityName=".MainActivity"/>
        <activity android:name=".MainActivity"
            android:launchMode="singleTop">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Login2Activity" />
        <activity android:name=".SingolaActivity"
            tools:ignore="Instantiatable" />
    </application>
</manifest>
```

Figura 5.4: File Manifest

Sign-in e Sign-up (trovare simbolo per mandare a capo)

Le prime pagine che un utente si trova di fronte aprendo la nostra applicazione saranno quelle per la registrazione e l'accesso a essa. Vediamo quindi, le parti fondamentali delle classi che ci permettono tali azioni. Nella Figura 5.5 vediamo la parte principale della classe di sign-up della nostra applicazione; come si evince dalla figura, facciamo affidamento su Firebase. In esso, infatti, registreremo gli utenti tramite la loro mail.

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    mAuth = FirebaseAuth.getInstance();
    textNome = findViewById(R.id.text_nome);
    textCognome = findViewById(R.id.text_cognome);
    textEmail = findViewById(R.id.text_email);
    textPassword = findViewById(R.id.text_password);
    btnRegistra = findViewById(R.id.btn_registra);
    mLoginBtn = findViewById(R.id.createText);
    btnRegistra.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            try {
                final String nome = textNome.getText().toString();
                final String cognome = textCognome.getText().toString();
                String email = textEmail.getText().toString();
                String password = textPassword.getText().toString();
                mAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if(task.isSuccessful()) {
                            final FirebaseUser user = mAuth.getCurrentUser();
                            UserProfileChangeRequest profileChangeRequest = new UserProfileChangeRequest.Builder()
                                    .setDisplayName(nome + "" + cognome)
                                    .build();
                            user.updateProfile(profileChangeRequest).addOnCompleteListener(new OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {
                                    writeUserToDb(nome, cognome, user.getUid());
                                    Intent intent = new Intent();
                                    intent.putExtra( name: "nome", textNome.getText().toString());
                                    intent.putExtra( name: "cognome", textCognome.getText().toString());
                                    setResult(RESULT_OK, intent);
                                    finish(); }}); } else{
                            task.getException().printStackTrace();
                            Toast.makeText( context: LoginActivity.this, "C' è stato un errore nella registrazione",Toast.LENGTH_SHORT).show(); } }});
            }catch (Exception e){
                Toast.makeText( context: LoginActivity.this, "Per favore,inserisci le informazioni richieste",Toast.LENGTH_SHORT).show(); }}});
```

Figura 5.5: File di registrazione della nostra app

Nella Figura 5.6, invece, vediamo come l'utente, tramite l'email e la password, possa accedere alla nostra applicazione, sempre grazie all'aiuto delle proprietà di Authentication e Database di Firebase, che vengono richiamate mediante alcune righe di codice.

```
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login2);
        mEmail = findViewById(R.id.text_email);
        mPassword = findViewById(R.id.text_password);
        mAuth = FirebaseAuth.getInstance();
        mLoginBtn = findViewById(R.id.loginBtn);
        mCreateBtn = findViewById(R.id.createText);
        mLoginBtn.setOnClickListener((view) → {
                String email = mEmail.getText().toString().trim();
                String password = mPassword.getText().toString().trim();
               mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener((task) → {
                       if(task.isSuccessful()){
Toast.makeText( context: Login2Activity.this, text: "Login effettuato con successo",Toast.LENGTH_SHORT).
            show();
         startActivity(new Intent(getApplicationContext(),MainActivity.class));
          }else {
Toast.makeText( context: Login2Activity.this, text: "Errore" + task.getException().
        getMessage(),Toast.LENGTH_SHORT).show();
          }
            }});
        });

        mCreateBtn.setOnClickListener((view) → {
                startActivity(new Intent(getApplicationContext(),LoginActivity.class));
        });

    }
}
```

Figura 5.6: File di login della nostra app

Home page

Nella Figura 5.7 viene visualizzato il file della nostra home page. Esaminiamo alcune parti fondamentali di questo file. Inizialmente vediamo come viene dato l'accesso a tale pagina solo se abbiamo effettuato il login; altrimenti, ovviamente, verrà caricata la pagina di login. Poi abbiamo implementato una `Recyclerview`, che ci permetterà di mostrare la lista delle spiaggie e le loro informazioni, che estrarremo da Firebase.

```
public MainActivity() {
}
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    initImageBitmaps();
    setUpRecyclerView();
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    button =(Button) findViewById(R.id.vai_a_spiagge);
    button.setOnClickListener(new View.OnClickListener() {
        @Override public void onClick(View view) { }});
    SharedPreferences preferences = getSharedPreferences( name: "login", MODE_PRIVATE);
    if(preferences.getBoolean( key: "firstrun",  defValue: true)) {
        Intent intent = new Intent( packageContext: MainActivity.this, LoginActivity.class);
        startActivityForResult(intent, LOGIN_REQUEST);} else{
        getSupportActionBar().setTitle("Benvenuto");
        fragmentFeed = new FragmentFeed();
        fragmentProfilo = new FragmentProfilo();
        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,fragmentFeed).commit();
        bottomNav = findViewById(R.id.bottom_menu);
        bottomNav.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                    Fragment selectedFragment = null;
                    switch (item.getItemId()){
                        case R.id.menu_feed:
                            selectedFragment = fragmentFeed;
                            break;
                        case R.id.menu_profilo:
                            selectedFragment = fragmentProfilo;
                            break; }
                    if(selectedFragment != null){
                        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,selectedFragment).commit(); }
                    return true; }}); } }
private void setUpRecyclerView(){
    Query query = spiaggeRef.orderBy( field: "voto",Query.Direction.DESCENDING);
    FirestoreRecyclerOptions<SingolaActivity> options = new FirestoreRecyclerOptions.Builder<SingolaActivity>()
            .setQuery(query,SingolaActivity.class)
            .build();
    adapter = new SpiaggeAdapter(options);
    RecyclerView recyclerView = findViewById(R.id.recycler_view);
    recyclerView.setHasFixedSize(true);
    recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
    recyclerView.setAdapter(adapter); }
@Override
protected void onStart() {
    super.onStart();
    adapter.startListening(); }
@Override
protected void onStop() {
    super.onStop();
    adapter.stopListening(); }
@Override
public void onPause() { super.onPause(); }
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);
    if(requestCode == LOGIN_REQUEST) {
        if(resultCode == RESULT_OK) {
            String nome = intent.getExtras().getString( key: "nome");
            String cognome = intent.getExtras().getString( key: "cognome");
            Map<String,Object> utenti = new HashMap<>();
            utenti.put(KEY_NOME,nome);
            utenti.put(KEY_COGNOME,cognome);
            db.collection( collectionPath: "utenti").document().set(utenti);
            getSupportActionBar().setTitle(nome + " " + cognome);
            SharedPreferences preferences = getSharedPreferences( name: "login", MODE_PRIVATE);
            SharedPreferences.Editor editor = preferences.edit();
            editor.putBoolean("firstrun", false);
            editor.apply();
        }
    }
}
```

Figura 5.7: File della home page della nostra app

Le pagine per la visualizzazione di una spiaggia singola e per la reccensione sono abbastanza semplici e con funzionalità e codice che abbiamo già visto nelle pagine precedenti.

Capitolo 6

Analisi SWOT e confronto con applicazioni correlate

In questo capitolo vedremo cosa è un'analisi SWOT[7], e che tipo di informazioni essa ci consente di trarre. Successivamente eseguiremo questo tipo di analisi sulla nostra applicazione, confrontandola con altre applicazioni di simile utilizzo.

6.1 Analisi SWOT

L'Analisi SWOT è uno strumento che ci permette di individuare i punti cruciali della nostra applicazione ed è la base per un marketing efficace. Essa ci consente di valutare i processi interni ed esterni della nostra applicazione, allo scopo di prendere delle decisioni che tengano conto degli obiettivi da raggiungere e del contesto in cui la nostra applicazione opererà. L'analisi SWOT quindi è uno strumento d'analisi utilizzato dalle aziende che tramite la creazione di una matrice ci permette di prendere decisioni strategiche. La matrice è divisa in quattro quadranti come si può vedere in Figura 6.1.

1. punti di forza (Strenghts)

2. punti di debolezza (Weaknesses)

3. opportunità (Opportunities)

4. minacce (Threats)

Figura 6.1: Matrice dell'analisi SWOT

Le ultime due, essendo fattori esterni, sono ovviamente più difficili da modificare, a differenza dei punti di forza e di debolezza che sono fattori endogeni e possono essere modificati facilmente.

6.1.1 Cause e conclusioni dell'Analisi SWOT

L'analisi SWOT ci consente di individuare i punti di forza sui quali affidarsi, i punti di debolezza da migliorare, le minacce da convertire in opportunità e le opportunità da ottimizzare. L'analisi si svolge seguendo un ordine ben preciso:

1. ci informiamo sul settore di interesse su cui la nostra applicazione opererà;
2. individuiamo i fattori esterni e li classifichiamo in opportunità e minacce;
3. costruiamo il piano commerciale risaltando i punti di forza e mitigando i punti di debolezza, ottimizzando (se possibile) le opportunità e attenuando le minacce;
4. controlliamo l'efficacia della strategia.

Una volta effettuata questa analisi si possono sviluppare diverse strategie che dipenderanno soprattutto alla valutazione dei fattori esterni e interni.
La schematizzazione, di tali strategie viene illustrata in Figura 6.2.
Tali strategie sono:

1. strategia S-O (punti di forza-opportunità) è la situazione ottimale, dove non c'è bisogno di alcuna modifica;
2. strategia W-O (punti di debolezza-opportunità) attraverso tale strategia vengono ridotti i punti di debolezza e ci si concentra sui punti di forza;
3. strategia W-T (punti di debolezza-minacce) è la situazione peggiore in cui ci si può trovare, in quanto si devono riuscire a eliminare le minacce e si devono trasformare i punti di debolezza in punti di forza;
4. strategia S-T (punti di forza-minacce) si convertono le minacce in opportunità.

Figura 6.2: Matrice delle strategie dell'analisi SWOT

6.2 Analisi SWOT della nostra applicazione

Come si può vedere in Figura 6.3 abbiamo effettuato un'analisi SWOT sulla nostra applicazione. Spieghiamo in dettaglio le motivazioni che ci hanno portato alla costruzione di questa specifica matrice SWOT.

Figura 6.3: Matrice SWOT della nostra applicazione

6.2.1 Punti di forza

La nostra applicazione si posiziona in un mercato libero da competitori, in quanto l'obiettivo è molto specifico. Infatti prendendo in considerazione le spiagge libere di una regione italiana, entriamo in profondità in quello che può essere un obiettivo più generale. La linearità di utilizzo e la facilità di navigazione sono alla base dei punti di forza della nostra applicazione, cosi da permettere l'utilizzo al maggior numero di utenti, non avendo barriere all'ingresso dovute all'impiego. Aggiungiamo, poi, che l'applicazione sarà gratuita, cosi da abbattere ancora di più le barriere d'entrata per usufruire dei servizi offerti.

6.2.2 Debolezze

Come abbiamo visto poco fa, la nostra applicazione si posiziona in un mercato alquanto libero da altri concorrenti; ciò è causato molto probabilmente alla limitata domanda per questo tipo di applicazione dovuta all'utilizzo in un luogo limitato e specifico, in questo caso la regione Marche. Altro fattore da considerare è la stagionalità dell'utilizzo delle spiagge in Italia e, quindi, anche della nostra applicazione.

6.2.3 Opportunità

Dopo un lungo periodo di chiusura delle infrastrutture, tra cui anche le spiagge dovute alla pandemia mondiale, quali anche le spiagge, ci avviciniamo sempre di più alla riapertura completa; questo aiuterà la diffusione della nostra applicazione che supporterà i turisti nella scelta della destinazione per le loro vacanze.

6.2.4 Minacce

Come già visto in precedenza, essendo quello della nostra applicazione un mercato alquanto privo di competitori e, di conseguenza, di applicazioni rivali, ovviamente la più grande minaccia è quella dell'entrata nel mercato di un nuovo programma con finalità uguali alle nostre.

6.3 Applicazioni correlate

L'idea su cui si basa la nostra applicazione non è ovviamente originale, in quanto possiamo trovare programmi e siti web che offrono simili servizi in altre località. Molte delle applicazioni trovate sul web si occupano di una specifica città; in esse oltre al servizio riguardante le spiagge, che, nella maggior parte dei casi, si occupano di stabilimenti balneari e non di spiagge libere, tali applicazioni comprendono anche varie altre funzionalità, strettamente legate alle le attività che si possono effettuare nella città. Un'applicazione di questo genere è WebPesaro, l'applicazione web del comune di Pesaro che, oltre alla possibilità di prenotare ombrelloni, lettini nei vari lidi della città, offre altri servizi, come la prenotazione per ristoranti e bollettini news

locali. Un altro esempio è l'applicazione Spiagge.it, che permette la prenotazione di ombrelloni, lettini e sdraio nei lidi in tutte le località balneari italiane con opportunità di recensire il servizo.

6.3.1 Confronto

Dagli esempi visti in precedenza, notiamo come le varie applicazioni che troviamo sul web, siano simili e, allo stesso tempo, diverse dalla nostra. Infatti, anche se il minimo comun denominatore sono le spiagge, l'obiettivo principale è totalmente diverso, dal momento che la nostra applicazione si occupa di recensire spiagge libere che non hanno bisogno di prenotazione, mentre le altre, essendo gli stabilimenti privati, oltre a poter recensire tali lidi ci danno l'opportunità di prenotare i vari servizi offerti. Ovviamente tutte le applicazioni di questo tipo sono gratuite, dal momento che devono raggiungere il maggior numero possibile di utenti. Esse, inoltre, sono per lo più servizi che un'amministrazione o un'organizzazione eroga per facilitare il turismo nelle località desiderate.

Capitolo 7

Conclusioni

L'idea del nostro software nasce dalla consapevolezza che il litorale marchigiano manca di un'applicazione che gestisca le recensioni riguardanti le spiagge libere, il cui mercato è sostanzialmente inesistente. Partendo da tale idea, abbiamo iniziato la progettazione del nostro software. Dopo una ricerca sulla suddivisione, nei nominativi e nelle descrizioni approfondite delle spiagge libere per la costruzione del nostro database abbiamo iniziato con l'implementazione dell'applicazione.

I servizi che offre la nostra applicazione non sono molti e sono alquanto modesti; partendo da questa base in futuro sarà possibile implementare nuovi servizi o si avrà la possibilità di estendere il raggio geografico di applicabilità. Poichè la base della nostra applicazione è alquanto semplice, sarà possibile aggiungere qualsiasi tipo di servizio relativo all'obiettivo generale senza troppi sforzi. Ad esempio, non dovrebbe essere difficile spostarsi nell'ambito delle spiagge private e aggiungere funzionalità che ci permettano di prenotare la postazione dove trascorrere le proprie vacanze.

E' anche possibile utilizzare la nostra applicazione per litorali di altre regioni, essenzialmente senza alcuna modifica al codice ma con un database diverso; effettuando alcune modifiche sarà possibile ampliare il raggio di utilizzo, ad esempio prendendo in considerazione tutte le spiagge libere dei vari litorali italiani. E' possibile, inoltre, anche se con molti accorgimenti, utilizzare la nostra applicazione come base per implementare un software di organizzazione di eventi o per poter aiutare gli utenti a darsi appuntamento per un ritrovo di gruppo in una determinata data.

Inoltre. anche se già esistente in alcune regioni, è possibile trasferire la base della nostra applicazione per la descrizione e recensione delle varie catene montuose e dei relativi percorsi turistici delle varie località italiane.

Concludendo, la nostra applicazione, oltre ad avere una propria finalità, è una buona base su cui appoggiarsi per lo sviluppo di diverse applicazioni future.

Bibliografia

[1] Ian Lake Reto Meier. *Professional Android*. 2018.

[2] Google. Android studio features, 2013.

[3] Kristin Marsicano Bill Phillips, Chris Stewart. *Android Programming: The Big Nerd Ranch Guide*. 2017.

[4] Erik Hellman. *Android Programming: Pushing the Limits*. 2013.

[5] MR Mark L Murphy. *The Busy Coder's Guide to Advanced Android Development*. 2011.

[6] Google. Firebase, 2013.

[7] Elena Renga. Come si fa un'analisi swot: guida completa con esempio di progetto, 2012.