

UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea in Informatica

Tesi di Laurea Triennale

## **COVID-19: dati, analisi e diffusione**

Progettazione e sviluppo di un'applicazione multiplatforma per analisi statistiche e diffusione dei dati relativi al COVID-19



**Relatori:**

prof. Arcangelo Castiglione  
prof. Fernando Antonio Pascuccio

**Candidato:**

Mario Egidio  
Matr. 0512105122

**ANNO ACCADEMICO 2019/2020**



## **Ringraziamenti**

Il ringraziamento più grande va alla mia famiglia e alla mia fidanzata, mi hanno sempre appoggiato e motivato durante il mio percorso universitario. Mi hanno aiutato nei momenti più difficili ed è solo grazie a loro se ho potuto raggiungere questo traguardo. Vorrei anche ringraziare tutti i miei amici, si sono sempre resi disponibili quando ne ho avuto bisogno. Grazie davvero a tutti voi, per me è impossibile elencarvi uno per uno ma sappiate che ognuno di voi è speciale per me.

Inoltre, un ringraziamento particolare va ai miei relatori, i professori Castiglione Arcangelo e Pascuccio Antonio. Il loro aiuto durante la stesura della tesi è stato veramente prezioso.

# Indice

Ringraziamenti .....	1
Introduzione.....	6
1.1    Panoramica.....	6
1.2    Obiettivi.....	7
1.3    Struttura dell’elaborato .....	7
Sviluppo Applicazioni Mobile .....	9
2.1    Cosa sono le App? .....	9
2.2    Perché è importante sviluppare App?.....	9
2.3    App native .....	10
2.4    App Ibride .....	11
2.5    Progressive Web App .....	12
2.6    App multiplatforma.....	13
2.7    Differenza tra app native e multiplatforma .....	14
Framework sviluppo Multiplatforma.....	17
3.1    Cos’è un framework? .....	17
3.2    Framework per App Multiplatforma più diffusi.....	17
3.2.1    React Native .....	18
3.2.2    Xamarin .....	19
3.2.3    Flutter .....	20
3.2.4    Flutter VS React Native VS Xamarin .....	22
Flutter.....	24
4.1    Motivazione scelta .....	24
4.2    Dart.....	25
4.3    Installazione e configurazione.....	26
Caso studio: Covid Analytics .....	29
5.1    Idea Iniziale.....	29

5.2	Analisi .....	30
5.2	Progettazione .....	32
5.3	Sviluppo .....	33
5.4	Realizzazione App Mobile .....	35
5.4.1	Home_screen .....	36
5.4.2	Grafici_screen.....	36
5.4.3	News_screen .....	37
5.4.4	Info_screen.....	37
5.5	Web .....	38
5.5.1	Firebase Hosting.....	38
5.5.1	App Web.....	38
	Sviluppi Futuri .....	42
6.1	Funzionalità aggiuntive .....	42
6.2	Integrazione con altre tecnologie .....	42
6.3	Diffusione informazioni sanitarie su altri tipi di pandemie.....	43
	Conclusioni.....	44
7.1	Multiplatforma in ambito medico.....	44
7.2	Considerazioni Finali .....	45
	Bibliografia .....	46

## Elenco delle figure

S.O. più utilizzati per dispositivi mobile.....	10
Capacità vs Portata .....	12
Figura che mostra una PWA eseguita da pc Windows.....	13
Figura che mostra una PWA eseguita da smartphone Android .....	13
App Native vs App Cross Platform.....	14
Framework Multipiattaforma più diffusi.....	18
Screen Home Android.....	36
Screen Grafici Android.....	36
Screen News Android .....	37
Screen Info Android .....	37
Screen Home Web .....	38
Screen Grafici Web .....	39
Screen News Web.....	40
Screen Info Web .....	41

**Listing**

Esempio “Hello World!” in React Native..... **19**

Esempio “Hello World!” in Xamarin ..... **20**

Esempio “Hello World!” in Flutter ..... **21**

Esempio dichiarazioni variabili in Flutter ..... **25**

Esempio chiamata metodo asincrono ..... **26**

# Capitolo 1

## Introduzione

### 1.1 Panoramica

La notevole evoluzione tecnologica, con il conseguente sviluppo nella produzione di software, ha permesso un'importante diffusione di informazioni tramite Internet. Tra le tecnologie più importanti troviamo gli smartphone.

La maggior parte delle persone ha a disposizione almeno uno di questi dispositivi, attraverso il quale ha la possibilità di accedere al Web. Infatti, proprio grazie al Web, è cambiato il modo in cui le persone vivono e, ancora più importante, è cambiato il modo in cui ci si informa.

Oggi viviamo nell'era digitale, usiamo i mezzi tecnologici (es. smartphone, tablet, pc, ecc.) per leggere i giornali, per lavorare in smart working<sup>1</sup>, per studiare, per restare in contatto con parenti o amici lontani, ecc. Ormai tutto questo è parte integrante della nostra vita.

Da questa diffusione tecnologica è nata l'esigenza, da parte degli sviluppatori, di realizzare software non vincolati ad un'unica piattaforma<sup>2</sup>. È molto importante che un software, una volta realizzato, venga reso disponibile per diversi dispositivi con diversi sistemi operativi (es. Android, iOS, MacOS, Windows, ecc.) in tempi brevi per poter raggiungere più utenti possibili. In informatica, sviluppare App Platform-Independent indica il fatto che un software, una volta realizzato, può essere utilizzato su diverse piattaforme.

---

<sup>1</sup> Il lavoro agile (o smart working) è una modalità di esecuzione del rapporto di lavoro subordinato caratterizzato dall'assenza di vincoli orari o spaziali e un'organizzazione per fasi, cicli e obiettivi, stabilita mediante accordo tra dipendente e datore di lavoro; una modalità che aiuta il lavoratore a conciliare i tempi di vita e lavoro e, al contempo, favorire la crescita della sua produttività. (lavoro.gov.it)

<sup>2</sup> Una piattaforma, in informatica, è una base hardware e/o software su cui sono sviluppati e/o eseguiti programmi o applicazioni; può indicare anche un ambiente di esecuzione che comprende hardware e sistema operativo ed eventualmente elementi middleware specifici, application server ed altri strumenti di supporto all'esecuzione di programmi. (wikipedia)



## 1.2 Obiettivi

L'obiettivo principale di questo elaborato è quello di analizzare le principali tecnologie volte alla progettazione e allo sviluppo di App Multiplatforma, al fine di valutare l'impatto che esse possono avere nel processo di ingegnerizzazione di apparecchiature biomedicali.

La realizzazione di questo progetto è stata svolta realizzando un'applicazione finalizzata alla visualizzazione, da parte degli utenti, dei dati giornalieri prodotti da apparecchiature biomedicali. Nel dettaglio vengono considerati i dati inerenti ai casi di COVID-19, ottenendo in tal modo un'analisi statistica tramite opportuni grafici ai fini di ricerca sperimentale di base.

## 1.3 Struttura dell'elaborato

Come già anticipato, l'obiettivo è quello di effettuare uno studio relativo all'impatto che può avere la realizzazione e l'utilizzo di un'applicazione indipendente dal tipo di piattaforma, quindi eseguibile su diversi dispositivi. Proprio per tale motivo, grande importanza è stata data allo studio degli strumenti tecnologici che ne permettono la realizzazione.

- Il primo capitolo è di introduzione;
- Il secondo capitolo definisce cosa sono le applicazioni mobile e mostra i vari modi per realizzarne una, evidenziando la differenza nella realizzazione di un App dipendente dal dispositivo utilizzato e di una Cross Platform;
- Nel terzo capitolo viene introdotto il concetto di framework, con una panoramica di quelli più diffusi impiegati nello sviluppo di App Multiplatforma. Nello specifico si discuterà in breve di React Native, Xamarin e Flutter;
- Nel quarto capitolo si approfondisce il Framework realizzato da Google, **Flutter**, e le motivazioni per cui è stato scelto per la realizzazione dell'App del caso di studio;

- Nel quinto capitolo si discute il processo di sviluppo del caso di studio **Covid Analytics**. Verranno affrontati diversi punti: idea, analisi, progettazione, sviluppo. A conclusione, verranno mostrate le immagini che dimostrano l'effettiva realizzazione dell'App;
- Nel sesto capitolo le proposte di possibili sviluppi futuri;
- Nel settimo capitolo vengono espone le conclusioni, introducendo una disamina teorica riguardante l'uso di queste tecnologie in ambito medico. Al termine, alcune considerazioni personali.

## **Capitolo 2**

### **Sviluppo Applicazioni Mobile**

#### **2.1 Cosa sono le App?**

In informatica, un'applicazione può essere definita come un programma software, destinato a un utente finale, così chiamato perché possiede appunto un'applicazione pratica per chi lo utilizza. Sono applicazioni per es. i programmi di word processing, i fogli di calcolo, i navigatori web e molti altri. Spesso il termine fa riferimento a terminali mobili (smartphone e tablet). [1]

Le App hanno come obiettivo principale quello di aggiungere funzionalità ad un dispositivo. In genere le applicazioni mobile sono diverse da quelle usate sul computer in quanto vanno adattate alle dimensioni degli smartphone o tablet, generalmente più piccole.

Esistono diversi tipi di App: i social network come Facebook o Twitter; App di utilità come la calcolatrice o la fotocamera; App di svago, ad esempio i giochi; ecc.;

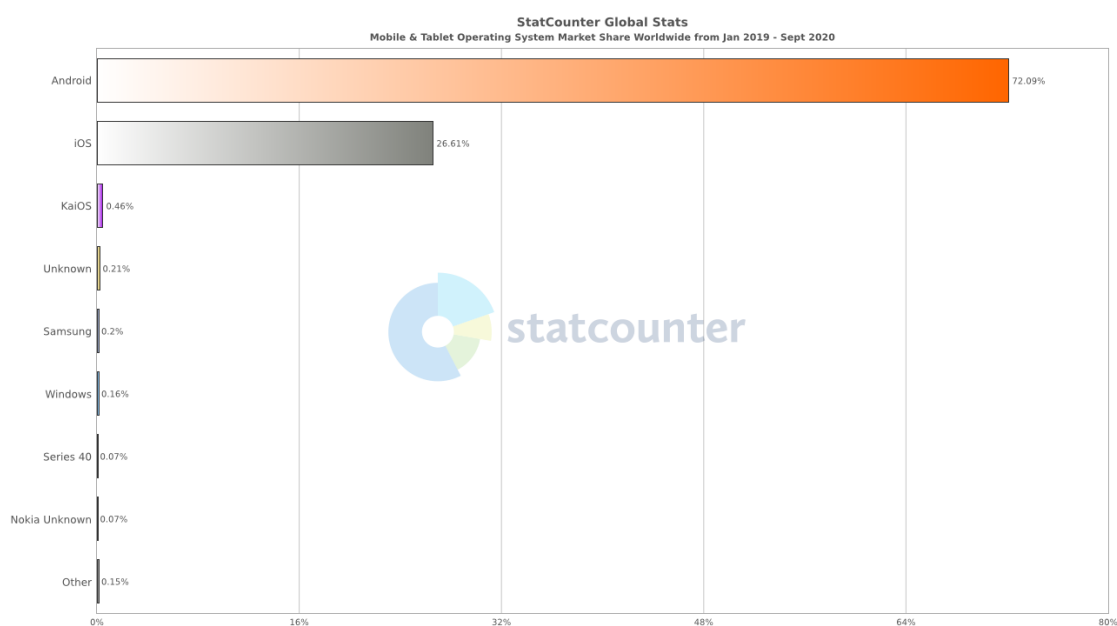
#### **2.2 Perché è importante sviluppare App?**

Ormai essenziali per ognuno di noi, da poco più di dieci anni le App fanno parte della nostra vita. Siamo tutti dei potenziali utilizzatori.

Per le aziende, questo è fondamentale, in quanto è proprio grazie ad alcune App che hanno la possibilità di aumentare la loro visibilità e raggiungere una clientela più vasta, promuovendo l'attività e pubblicizzando i singoli prodotti (es., sponsorizzazioni su Facebook o Instagram). Inoltre, nell'ambito degli acquisti online si è rilevata una preferenza da parte del cliente dell'App piuttosto che il sito Web. Ovviamente, per un'azienda, il non essere presente come App sullo store delle piattaforme più diffuse porta ad una sensibile perdita di possibili clienti.

Il processo di installazione delle App su uno smartphone avviene, in genere, con pochi click. Questa facilità di installazione ha reso molto semplice la loro diffusione. Quindi, col crescente utilizzo di App anche la necessità della loro produzione è aumentata. Gli sviluppatori devono considerare i gusti del pubblico che, negli ultimi tempi, è diventato sempre più esigente e difficile da accontentare.

I dispositivi mobile più utilizzati hanno come sistema operativo Android o iOS, per questo motivo l'obiettivo di chi produce App è quello di poterle rendere disponibili almeno questi due sistemi operativi principali.



Il grafico mostra in percentuale i S.O. più utilizzati nei dispositivi mobile. Fonte [gs.statcounter.com](https://gs.statcounter.com)

## 2.3 App native

Le applicazioni native sono sviluppate appositamente per uno specifico sistema operativo tramite un particolare linguaggio compatibile soltanto con la piattaforma di destinazione. Sono note per essere incredibilmente ricche e affidabili. Funzionano indipendentemente dalla connessione di rete. Possono leggere e scrivere dati dal "file system locale", accedere all'hardware del dispositivo, utilizzare i Bluetooth e possono persino interagire con i dati memorizzati (es., contatti, eventi del calendario, ecc.). Le

applicazioni native sembrano parte integrante del dispositivo su cui vengono eseguite.[2]

Il problema principale dello sviluppo di applicazioni native è che, per poterle eseguire su diversi sistemi operativi, devono essere sviluppate singolarmente per ognuno di essi.

Ad esempio, per sviluppare applicazioni compatibili con il sistema operativo Android, si utilizza il tool Android Studio e come linguaggio di programmazione Java o Kotlin. [3]

Invece, per realizzare applicazioni compatibili con iOS, si utilizza l'ambiente di sviluppo Xcode e come linguaggio di programmazione Swift oppure Objective-C. [4]

Il doverle sviluppare più volte comporta una notevole perdita di tempo sia per il rilascio iniziale che per i futuri aggiornamenti. Infatti, ogni aggiornamento reso pubblico per un determinato sistema operativo va sviluppato da zero per poter essere pubblicato per tutti gli altri tipi di piattaforme.

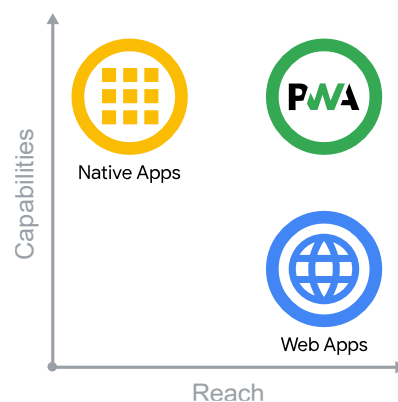
## 2.4 App Ibride

Si tratta di applicativi mobili che sfruttano sia componenti nativi sia tecnologie web per gestire l'interfaccia e la logica dell'applicazione. L'utilizzo delle tecnologie web avviene attraverso il componente **WebView**. Le app ibride sono una fase intermedia verso le Progressive Web App: imitano l'esperienza utente delle app native su mobile, e hanno ancora la necessità di un App store per essere scaricate. Come tali consumano spazio di memoria sul terminale. La combinazione di markup, fogli di stile e script ha consentito la creazione di elementi interattivi personalizzati senza l'uso di sistemi chiusi come "Flash". Pur essendo eseguite parzialmente nel browser del terminale mobile, le applicazioni ibride non hanno un URL, ma hanno un'interfaccia utente ricca, e accesso alle funzioni di sistema. [5]

## 2.5 Progressive Web App

Termine coniato in origine da Google. Le Progressive Web App (PWA) sono costruite e migliorate con API moderne per fornire funzionalità, affidabilità di tipo nativo raggiungendo chiunque, ovunque e su qualsiasi dispositivo con un singolo codice sorgente.

Una PWA è veloce e affidabile indipendentemente dalla rete. Inoltre, le PWA possono essere installate ed eseguite in una finestra autonoma anziché in una scheda del browser. Possono anche essere avviate dalla schermata iniziale dell'utente o dal menu delle applicazioni.



Capacità vs. Portata. Fonte webdev

Quando una PWA non è più eseguita attraverso una scheda del browser, ma si trova in una finestra autonoma, trasforma il modo in cui gli utenti la pensano e interagiscono con essa. [7]

Le applicazioni web vengono sviluppate e caricate come normali pagine web, ma assumono un comportamento simile alle applicazioni native.

Il termine "Progressive" si riferisce al fatto che, dal punto di vista dell'esperienza utente, queste applicazioni possono abilitare una serie di funzionalità aggiuntive alle normali pagine web. Ad esempio il browser può proporre all'utente di salvarle nella schermata home del terminale mobile, per essere percepite a tutti gli effetti come delle app native. [8]

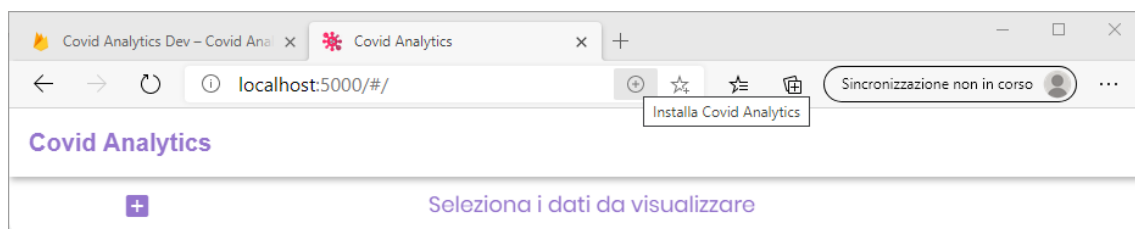


Figura che mostra una PWA eseguita da pc Windows

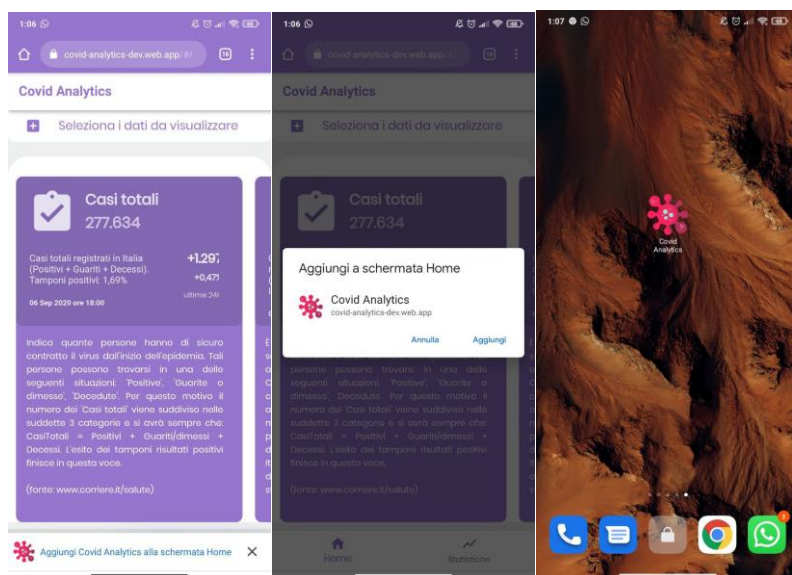


Figura che mostra una PWA eseguita da smartphone Android

## 2.6 App multiplatforma

Negli ultimi tempi, le soluzioni Cross Platform si stanno diffondendo sempre di più, in quanto la loro flessibilità riduce notevolmente i tempi di produzione per trasportarla su diverse piattaforme.

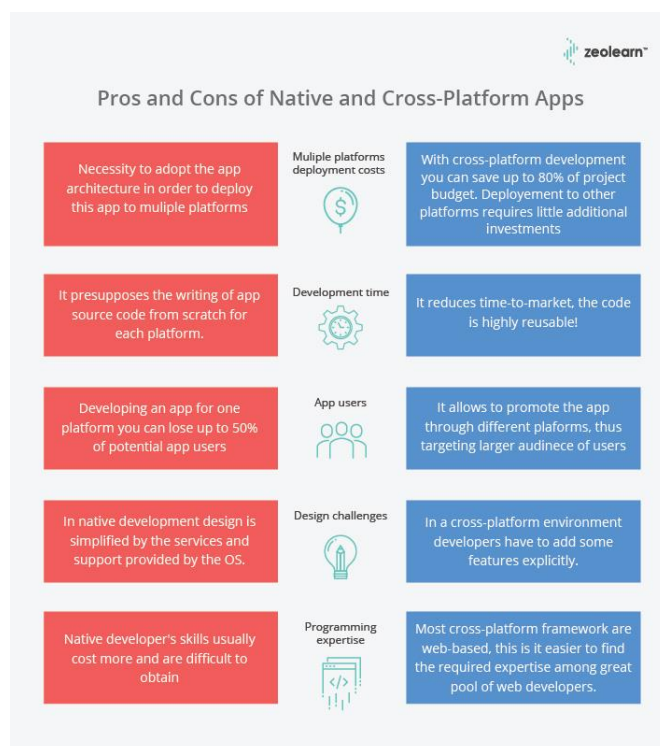
Lo sviluppo di app basate su soluzioni multiplatforma (es., Titanium, PhoneGap, Ionic) sono indipendenti dal tipo di sistema operativo. In questo modo, lo sviluppatore scrive il codice di un'applicazione una sola volta e lo pubblica su piattaforme hardware/software (supportate) diverse (es., iOS e Android). [6]

Il software multiplatforma può essere suddiviso in due tipi: uno richiede la creazione o la compilazione individuale per ciascuna piattaforma che supporta; l'altro può essere

eseguito direttamente su qualsiasi piattaforma senza una preparazione speciale (es., software scritto in un linguaggio interpretato o bytecode<sup>3</sup> portatile precompilato per il quale gli interpreti<sup>4</sup> o i pacchetti run-time sono componenti comuni o standard di tutte le piattaforme). [9]

## 2.7 Differenza tra app native e multiplatforma

Lo sviluppo di app multiplatforma garantisce costi più bassi e tempi di sviluppo brevi. Ciononostante, questo processo presenta dei limiti: non tutte le funzionalità dei dispositivi mobile sono accessibili (es., l'utilizzo di sensori); bisogna controllare il funzionamento dell'App su tutte le piattaforme per cui si vuole rendere disponibile l'App.



App Native vs App Cross Platform. Fonte zeolearn

<sup>3</sup> Il bytecode è un linguaggio intermedio più astratto tra il linguaggio macchina e il linguaggio di programmazione, serve per descrivere le operazioni che costituiscono un programma.

<sup>4</sup> Programma che consente di tradurre le istruzioni dai linguaggi letterali al linguaggio di macchina;



Inoltre, un App multiplatforma non ha le stesse prestazioni di un App nativa poiché non ha la possibilità di fare lo stesso uso di risorse del dispositivo.

Per progetti di grandi dimensioni, che necessitano molte funzionalità e alte prestazioni, è sempre meglio orientarsi verso lo sviluppo nativo poiché lo sviluppo Cross Platform prevede che un codice sorgente si adatti alle diverse piattaforme di destinazione, facendo in modo da non avere un'ottimizzazione su ogni singolo dispositivo.

Per la distribuzione di App native su piattaforme differenti è necessario progettare da zero l'App per renderla compatibile con la piattaforma di destinazione (ad esempio, bisogna avere familiarità con Swift / Objective-C e Xcode per iOS, Java / Kotlin e Android Studio per Android). Sviluppando app Cross Platform questo problema viene risolto ed è necessario apprendere soltanto un linguaggio di programmazione (es., il linguaggio Dart per il framework Flutter).

I vantaggi per lo sviluppo di app multiplatforma sono principalmente la manutenibilità e la velocità di sviluppo. Infatti, producendo un unico codice sorgente, i tempi di sviluppo si abbreviano e in caso di problemi o bug basta correggere l'errore e ripubblicare l'App su tutte le piattaforme desiderate.

<b>App native</b>	<b>App Multiplatforma</b>
<b>Vantaggi</b>	<b>Vantaggi</b>
Performance	Costi di sviluppo
Stabilità	Tempi di sviluppo
Progetti di Grandi dimensioni con molte funzionalità	Rapidità nella manutenzione e distribuzione di aggiornamenti
<b>Svantaggi</b>	<b>Svantaggi</b>
Costi di sviluppo	Non adatta a progetti di grandi dimensioni
Tempi di sviluppo	Performance minori rispetto le App Native

Tempi lunghi per manutenzione e aggiornamenti	Dipendenza dal framework
Possibilità di ascesa di uno specifico sistema operativo	

Quindi la scelta finale nell'utilizzo di uno di questi framework dipende dal tipo di app che si deve realizzare. In ogni caso non è da escludere l'ipotesi di impiegare framework multiplatforma nella prima fase dello sviluppo, in modo da raggiungere diverse piattaforme in tempi brevi e spiazzare la concorrenza. In una seconda fase è possibile passare allo sviluppo di app native per ottenere maggiore performance e stabilità. Ciò costituisce una strategia vincente per il programmatore.

## Capitolo 3

### Framework sviluppo Multiplatforma

#### 3.1 Cos'è un framework?

Un framework può essere definito come un sistema distribuito in cartelle e sotto forma di alcune porzioni di codice prestabilite che, connesse attraverso un design concettuale univoco, sono state create per la realizzazione di alcune specifiche funzionalità pronte all'uso. [10]

L'utilizzo di framework semplifica molto l'attività dello sviluppatore.

Un framework può essere inteso come una raccolta di librerie<sup>5</sup>. La differenza principale risiede nella cosiddetta "inversione del controllo": un framework non è chiamato ma esegue la chiamata, sulla quale ha quindi il controllo. Si tratta di una sorta di base per la programmazione. Il framework può includere una libreria, un compilatore e altri programmi utilizzati nel processo di sviluppo. Se dispone di un framework valido, uno sviluppatore non deve occuparsi delle parti di codice ricorrenti utilizzate in un dominio o in un'applicazione. In sostanza, rispetto a un'applicazione, la libreria rappresenta una determinata funzione e un framework è la sua struttura portante. [11]

#### 3.2 Framework per App Multiplatforma più diffusi

Tra i framework più diffusi nella realizzazione di App Cross Platform ci sono React Native, Flutter e Xamarin.

Verranno esposte in breve le caratteristiche principali di tali framework.

---

<sup>5</sup> Una libreria è una raccolta di risorse o routine precompilate e di utilizzo frequente (classi, modelli, dati di configurazione, ecc.) a disposizione di un programma informatico.

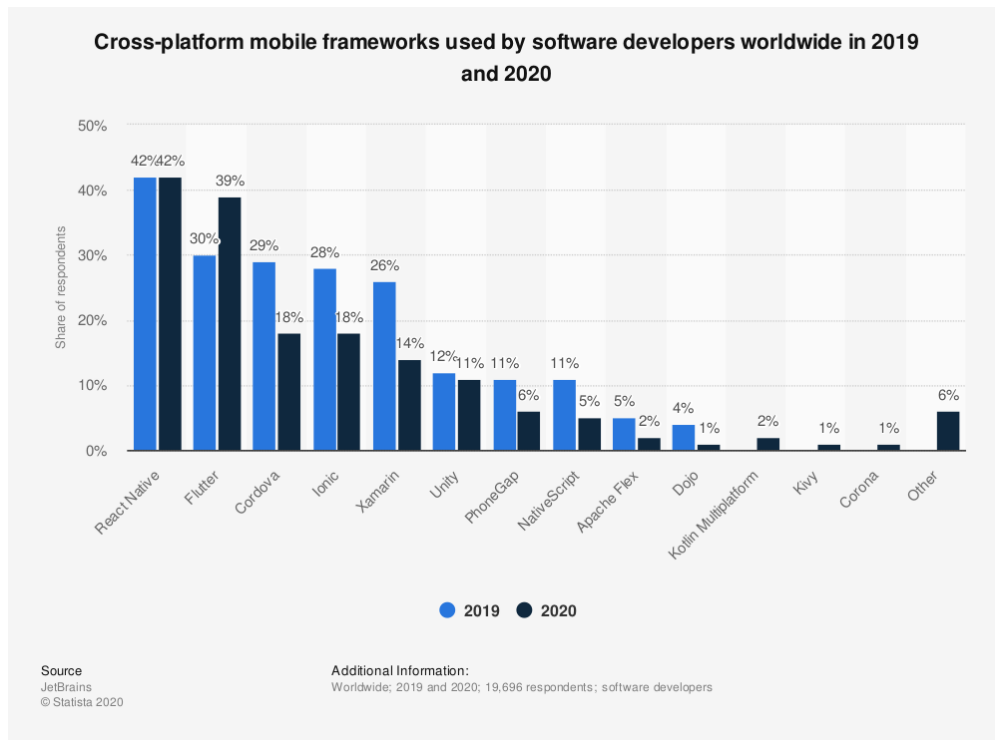


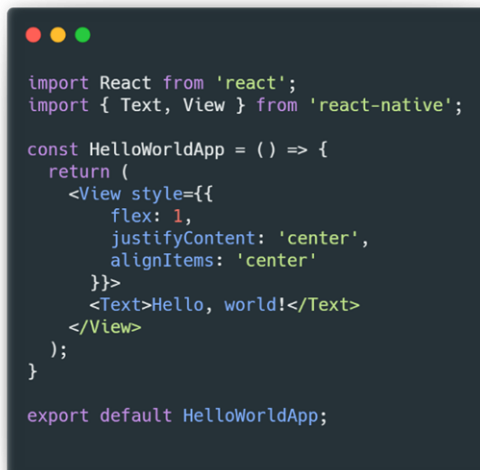
Grafico che mostra i framework più utilizzati nel 2019 e 2020. Fonte *statista*.

### 3.2.1 React Native

React Native è un framework di Facebook per lo sviluppo di app multiplatforma compatibili con iOS e Android. Combina le parti migliori dello sviluppo di **App Native** con **React**, una libreria JavaScript per la creazione dell'interfaccia utente. È possibile utilizzare React Native nei progetti Android e iOS esistenti oppure è possibile creare un App completamente da zero. [12]

È scritto in JavaScript ma renderizzato utilizzando le API native, il che lo rende un framework molto flessibile e relativamente semplice per chi ha già esperienza nell'uso di JavaScript. Utilizzando questo framework, le App per Android e iOS condividono fino all'80% del codice sorgente.

### *Esempio "Hello World!" in React Native*



```
import React from 'react';
import { Text, View } from 'react-native';

const HelloWorldApp = () => {
  return (
    <View style={{
      flex: 1,
      justifyContent: 'center',
      alignItems: 'center'
    }}>
      <Text>Hello, world!</Text>
    </View>
  );
}

export default HelloWorldApp;
```

Le App vengono sviluppate attraverso un mix di JavaScript e markup XML, meglio noto come **JSX**.

React Native non è privo di svantaggi. Il rischio maggiore è probabilmente la maturità, poiché il progetto è ancora relativamente giovane. Il supporto iOS è stato rilasciato a marzo 2015 e il supporto Android è stato rilasciato a settembre 2015. La documentazione ha certamente margini di miglioramento e continua ad evolversi, inoltre alcune funzionalità su iOS e Android non sono ancora supportate. [13]

### **3.2.2 Xamarin**

Xamarin è una piattaforma open source per la compilazione di applicazioni moderne e a prestazioni elevate per iOS, Android e Windows con .NET. Xamarin è un livello di astrazione che gestisce la comunicazione del codice condiviso con il codice della piattaforma sottostante. Xamarin viene eseguito in un ambiente gestito che offre diversi vantaggi, ad esempio l'allocazione della memoria e la Garbage Collection.

Xamarin consente agli sviluppatori di condividere una media del 90% delle applicazioni su più piattaforme. Questo modello consente agli sviluppatori di scrivere tutta la logica di business in un unico linguaggio (o di riutilizzare il codice dell'applicazione esistente), replicando in ogni piattaforma le prestazioni e l'aspetto

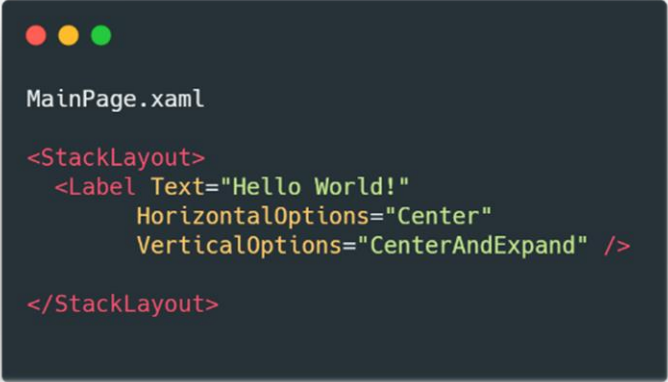
originari. Le applicazioni Xamarin possono essere scritte su PC o Mac e vengono compilate in pacchetti dell'applicazione nativa, ad esempio un file APK in Android o un file IPA in iOS.

Xamarin è destinato agli sviluppatori con gli obiettivi seguenti:

- Condividere il codice, i test e logica di business tra le piattaforme.
- Scrivere applicazioni multiplatforma in C# con Visual Studio.

Xamarin supporta sia uno stile dichiarativo di progettazione dell'interfaccia utente basato su file XML, sia la creazione dell'interfaccia utente a livello di codice nel codice. Quando si usa l'approccio dichiarativo, i file XML possono essere modificati manualmente o modificati visivamente utilizzando la finestra di progettazione **Xamarin.Android**. L'uso di una finestra di progettazione consente un riscontro immediato durante la creazione dell'interfaccia utente, velocizza lo sviluppo e rende il processo di creazione dell'interfaccia utente meno laborioso. [14]

*Esempio Hello World utilizzando Xamarin*



```
MainPage.xaml

<StackLayout>
  <Label Text="Hello World!"
    HorizontalOptions="Center"
    VerticalOptions="CenterAndExpand" />
</StackLayout>
```

### 3.2.3 Flutter

Flutter è il framework open-source per lo sviluppo App Cross Platform realizzato da Google per la creazione di interfacce native per iOS e Android. Fu rilasciato nel 2015 con il nome **Sky**. Inoltre, permette la creazione di applicazioni per il sistema operativo

Google Fuchsia. All'inizio, era compatibile solo con il sistema operativo Android, ma in poco tempo ha ampliato il suo supporto ad iOS e allo sviluppo di Web App. [18]

Flutter è composto da quattro componenti principali

- Dart Platform;
- Flutter engine, scritto in C++, fornisce supporto per il rendering a basso livello utilizzando la libreria Skia Graphics.
- Foundation library, un'interfaccia sugli SDK nativi di entrambe le piattaforme (iOS e Android);
- Design-specific widgets;

#### *Esempio Hello World utilizzando Flutter*

```
import 'package:flutter/material.dart';

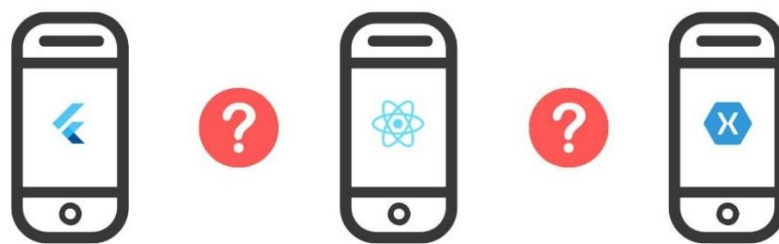
void main() {
  runApp(new HelloWorldApp());
}

class HelloWorldApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'HelloWorld APP',
      home: Scaffold(
        appBar: AppBar(
          title: Text('HelloWorld APP'),
        ),
        body: Center(
          child: Text('Hello World!'),
        ),
      ),
    );
  }
}
```

Una funzionalità molto importante offerta da Flutter è l'**Hot Reload**. Hot Reload utilizza la funzione di compilazione JIT di Dart. Il codice modificato viene iniettato nell'applicazione in esecuzione in modalità debug in pochi millisecondi,

mantenendo lo stato nella sua memoria. Questa è una funzionalità straordinaria per gli sviluppatori, poiché cambia il modo in cui funziona il flusso di lavoro di sviluppo. Infatti, in fase di sviluppo, ogni modifica al codice sorgente viene subito resa disponibile all'App in esecuzione in **debug mode**. Offre l'opportunità di scrivere il codice in un modo diverso e facendo sì che lo sviluppatore possa apportare più modifiche al codice dell'interfaccia utente, senza temere un lavoro rigoroso. [18]

### 3.2.4 Flutter VS React Native VS Xamarin



Fonte dzone.com

	Flutter	React Native	Xamarin
Linguaggio	Dart	JavaScript	C# con ambiente .NET
Piattaforme supportate	iOS, Android, Web Apps, Linux, MacOS	iOS, Android	iOS, Android, Windows, MacOS
Hot Reload	Si	Si	No
Architettura	Skia C++ engine	React Native Flux	Xamarin Mono Execution Environment
Performance	4/5	4.5/5	4/5



Community	5/5	4.5/5	3.5/5
IDE supportate	Android Studio, WebStorm, Visual Studio Code, Atom	Tutti gli IDE con supporto a JS	Microsoft Visual Studio
Distribuzione	Open Source	Open Source	Open Source + paid

[21]

## Capitolo 4

### Flutter

#### 4.1 Motivazione scelta

Flutter ha dietro una community molto vasta ed inoltre ha a disposizione sul sito ufficiale, <https://flutter.dev/docs>, un'ottima documentazione.

A differenza della maggior parte dei framework multiplatforma, non utilizza componenti nativi della piattaforma. In altre parole, quando si deve far visualizzare un pulsante, Flutter stesso esegue il rendering di quel pulsante. Ciò consente alle App prodotte di essere coerenti su tutte le piattaforme. Questo ha anche il vantaggio che i nuovi Componenti o Widget possono essere aggiunti a Flutter rapidamente e facilmente senza preoccuparsi del fatto che la piattaforma sottostante li supporti. Inoltre, fornisce widget specifici per la realizzazione della UI.

Nello specifico, offre due set:

- Widget di **Material**
- Widget di **Cupertino**. [15]

Quindi, uno dei motivi principali per cui è stato scelto questo framework al posto di altri è che permette la realizzazione di un App utilizzando il design originariamente concepito, senza comprometterne alcun aspetto. Possiamo quindi affermare che ***ha un'enorme potenza espressiva***.

Inoltre, è basato sul linguaggio Dart, realizzato da Google, orientato agli oggetti, con uno stile di sintassi basato su C il che rende l'apprendimento molto veloce per la maggior parte degli sviluppatori. [18]

Infine, è stato scelto Flutter anche per il semplice fatto che, oltre allo sviluppo di App Android e App iOS, grazie all'ultima release (Flutter 1.9) spostandosi sul "canale beta" permette la realizzazione di App Web e spostandosi sul "canale dev" permette la

realizzazione di App Desktop (per ora compatibili con MacOS e Linux). Il tutto a partire dallo stesso codice sorgente.

## 4.2 Dart

Dart è compilato ed è completamente orientato agli oggetti. Ha a disposizione il Garbage-Collector, quindi nessun problema di allocazione e deallocazione della memoria.

Ha uno stile di sintassi basato su C il che lo rende facile da apprendere per la maggior parte degli sviluppatori. Supporta le funzionalità più comuni agli altri linguaggi, come le classi, classi astratte e interfacce. Può essere usato in modo fortemente tipizzato ma anche in modo libero e senza alcun vincolo sulla tipizzazione.

Offre supporto alla concorrenza in modo da poter avere processi indipendenti che non condividono la memoria ma utilizzano invece la trasmissione dei messaggi per comunicare, ottenendo buone prestazioni riducendo il rischio tipicamente associato ai paradigmi di programmazione concorrente.

Compila in codice ARM e x86, ma può anche essere trasferito in JavaScript in modo che il codice Dart possa essere eseguito anche sul web.

Dart è un linguaggio fortemente tipizzato, ma non è sempre necessario indicare il tipo di variabile:



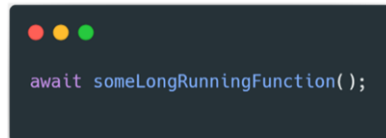
```
var val = "variabile 1";  
String var2 = "Variabile 2";  
dynamic var3 = "Variabile 3";  
Object var4 = "Variabile 4";
```

È possibile dichiarare una costante aggiungendo la parola chiave **const** oppure è possibile rendere final una variabile con l'aggiunta della parola chiave **final**.

Un grande vantaggio offerto da Dart è l'asincronia. Sono due le classi fondamentali che lo permettono, **Future** e **Stream**, insieme alle due parole chiave, **async** e **await**.

Entrambe le classi sono oggetti che le funzioni asincrone restituiscono quando inizia un'operazione di lunga durata. Nell'attesa del risultato, l'App può continuare a fare altre cose.

Per chiamare una funzione che restituisce un Future, bisogna usare la parola chiave **await**.<sup>[15]</sup>

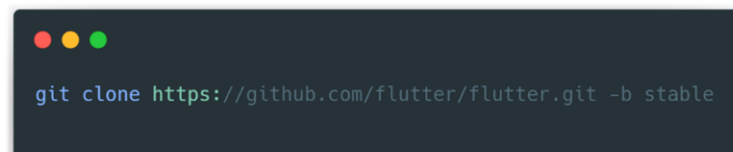


```
await someLongRunningFunction();
```

### 4.3 Installazione e configurazione

Per installare e configurare Flutter bisogna procedere come indicato sulla documentazione ufficiale. Di seguito verrà mostrata la procedura di configurazione tramite un Pc Windows.

- Per prima cosa bisogna ottenere il codice sorgente dal repository Flutter su GitHub e modificare branch o tag secondo la necessità. Per esempio per la versione stabile:



```
git clone https://github.com/flutter/flutter.git -b stable
```

- Come secondo passaggio bisogna aggiornare le variabili d'ambiente aggiungendo Flutter. Quindi, dalla barra di ricerca inserire la parola "env" e selezionare **Modifica variabili di ambiente**;
  - Qui, se è presente una voce chiamata **Path** va aggiunto al contenuto il percorso completo di **flutter\bin** utilizzando ";" come separatore dal valore già esistente.
  - Se la voce non esiste, va creata una nuova variabile denominata **Path** con il percorso completo di **flutter\bin** come valore.

- Dopodiché, è necessario chiudere e riaprire tutte le finestre della console esistenti affinché queste modifiche abbiano effetto;
- Procedere con l'esecuzione del comando **flutter doctor** per avere un rapporto sull'avvenuta installazione

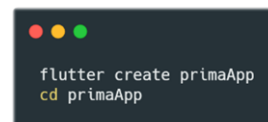


```
C:\src\flutter>flutter doctor
```

- Successivamente bisogna scaricare installare e installare **Android Studio** per fornire le dipendenze dalla piattaforma Android. Successivamente è possibile usare un qualsiasi editor a scelta per lo sviluppare App (es., Visual Studio Code, WebStorm, Atom, ecc.);
- Come ultimo passaggio è necessario configurare un dispositivo Android. Questa procedura può essere fatta in due modi diversi:
  - Abilitare la modalità sviluppatore e attivare il debug USB da un dispositivo fisico per poi collegarlo tramite un cavo USB al pc.
  - Utilizzare l'emulatore messo a disposizione da Android Studio.[18]

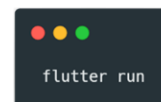
Dopo aver eseguito correttamente la procedura d'installazione è possibile creare la prima app.

- Per prima cosa va creato un nuovo progetto attraverso il comando **flutter create**;



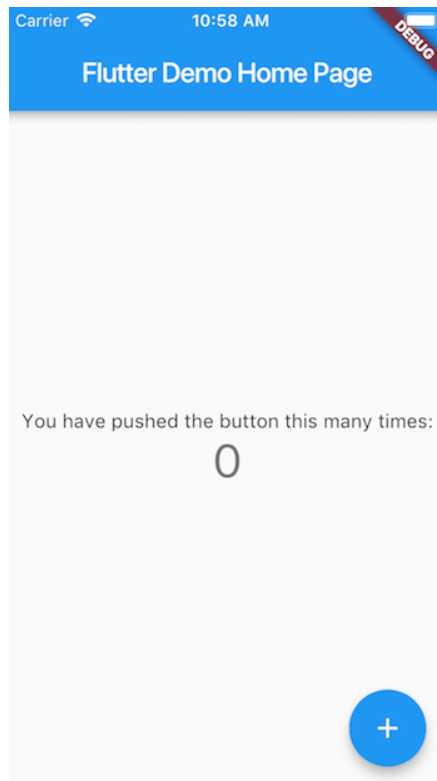
```
flutter create primaApp
cd primaApp
```

- Poi, dopo essersi assicurati di avere un dispositivo Android collegato (dispositivo fisico o emulatore), lanciare il comando **flutter run**;



```
flutter run
```

Schermata del dispositivo collegato



Fonte Flutter.dev

## Capitolo 5

### Caso studio: Covid Analytics

#### 5.1 Idea Iniziale

Covid Analytics è nata con l'idea di offrire informazioni inerenti ai casi di **COVID-19 in Italia** a tutti cittadini.

La malattia da coronavirus (COVID-19) è una malattia infettiva causata da un coronavirus appena scoperto. La maggior parte delle persone che contraggono il virus sviluppa sintomi lievi o moderati e guarisce senza aver bisogno di particolari cure.

Il virus che causa COVID-19 viene perlopiù trasmesso attraverso le goccioline prodotte dalle persone infette quando tossiscono, starnutiscono o espirano. Queste goccioline sono troppo pesanti per restare sospese nell'aria e si depositano rapidamente a terra o sulle superfici. Si può contrarre l'infezione respirando il virus se ci si trova nelle immediate vicinanze di una persona affetta da COVID-19, oppure toccando una superficie contaminata e poi toccandosi gli occhi, il naso o la bocca.

[20]

I dati vengono prelevati da una "fonte ufficiale", ovvero il Dipartimento della Protezione Civile (DPC <sup>6</sup>). Il DPC si occupa, ogni giorno, di pubblicare tutti i dati aggiornati su una repository presente su GitHub.

Pensata inizialmente solo come App Android, attraverso l'utilizzo del framework Flutter è diventata molto di più.

---

<sup>6</sup> Il Dipartimento della Protezione Civile è la struttura del governo della Repubblica Italiana preposta al coordinamento delle politiche e delle attività in tema di difesa e protezione civile, facente capo alla Presidenza del Consiglio dei ministri.

Grazie a questa applicazione l'utente finale può visionare la situazione attuale in Italia avendo l'opportunità di consultare dati e grafici.

La funzionalità principale dell'applicazione consiste nella visualizzazione dei dati di tutta la Nazione, delle Regioni e delle singole province. Inoltre, essendo queste ultime più numerose, è previsto un form adibito alla ricerca della provincia desiderata.

## 5.2 Analisi

Di seguito verranno illustrati alcuni scenari di utilizzo.

1. Marco, residente a Montella (AV), è molto preoccupato della situazione COVID-19 in Italia. Vuole vedere nella sua provincia, ovvero Avellino, com'è la situazione attuale. Allora apre l'App **Covid Analytics** dal suo smartphone. Clicca sul pulsante per selezionare il tipo di dati da visualizzare. Qui gli viene chiesto se scegliere tra i dati Nazionali, Regionali o provinciali. Marco seleziona le province. Dopo aver selezionato, gli compare un elenco di province. Si reca col dito sull'apposito form per la ricerca ed inserisce "Avellino" ed ottiene i dati richiesti.
2. Michela è interessata alle statistiche sui dati per cercare di capire se la variazione di contagi giornalieri è in crescita. Si reca sul sito web <https://covid-analytics-dev.web.app> per poter visionare i grafici. Una volta caricato il sito clicca sulla sezione "statistiche". Qui si reca sulla scritta per visionare il grafico inerente alla variazione percentuale giornaliera. Appena clicca gli viene mostrato il grafico con tutti i dati da lei richiesti.

Nel seguito si andranno ad analizzare i requisiti funzionali dell'applicazione.



L'app deve offrire le seguenti funzionalità:

- Possibilità di ottenere delle statistiche attraverso dei Grafici. Nello specifico ci dovranno essere:
  - Grafico per vedere l'andamento dei Casi totali
  - Variazione percentuale giornaliera
  - Incremento giornaliero dei contagi
  - Incremento giornaliero dei tamponi effettuati
  - Incremento giornaliero delle guarigioni
  - Incremento giornaliero del numero di persone attualmente positive
  - Incremento giornaliero del numero di decessi
  - Incremento giornaliero del numero di persone ricoverate
  - Tasso di letalità per regione
  - Regioni colpite maggiormente
  
- Possibilità di filtrare i dati scegliendo tra dati Nazionali, Regioni e Province
  - I dati di tutta la nazione devono essere suddivisi in:
    - #Casi totali;
    - #Persone Positive;
    - #Persone Guarite;
    - #Decessi;
    - #Ospedalizzati;
    - #Persone In Terapia Intensiva;
    - #Ricoverati;
    - #Tamponi
    - #Persone In Isolamento
  - Nell'elenco delle regioni deve esserci la possibilità di ordinamento. I tipo di ordinamento devono essere i seguenti:
    - Ordinamento per #contagi
    - Ordinamento per #decessi

- Ordinamento per percentuale contagi
  - Ordinamento per percentuale decessi
  - Ordinamento per nome
- Nell'elenco delle province deve esserci la possibilità di ricerca per nome provincia.
- I dati della singola regione, oltre a mostrare gli stessi campi dei dati nazionali, devono mostrare:
  - La percentuale di contagiati per abitanti
  - La percentuale di decessi per abitanti
- I dati della singola provincia devono mostrare il numero di contagiati più l'incremento avvenuto nelle ultime 24h.
- Possibilità di controllare le ultime notizie
- Possibilità di effettuare una donazione al conto della protezione civile.

Di seguito verranno discussi alcuni requisiti non funzionali

- Usabilità: considerando che il tipo di utenti che andranno ad utilizzare l'app, occorre che il sistema sia il più semplice e intuitivo possibile;
- Efficienza: Il tempo di risposta deve essere rapido;

## 5.2 Progettazione

L'app deve essere lineare ed intuitiva per garantire all'utente l'ottenimento di informazioni in modo rapido. L'obiettivo è quello di suddividere le diverse funzionalità attraverso un semplice menu, costituito da:

- Home;
- Statistiche;
- Notizie;
- Info.

Nella sezione **Home** dovrà esserci la possibilità di filtrare il tipo di dati da visualizzare:

- **Province:** la sezione che riguarda le province dovrà essere costituita da un elenco verticale con un campo dedicato appositamente alla ricerca della provincia desiderata.
- **Regioni:** la sezione che riguarda le regioni dovrà essere costituita da un elenco verticale con la possibilità di poterne ordinare il contenuto in diversi modi (es., nome, contagi, decessi, ecc.).
- **Dati nazionali:** la sezione che riguarda i dati specifici della nostra nazione dovrà contenere un elenco di dati. Per ogni dato è prevista una spiegazione dettagliata.

La sezione **Statistiche** dovrà contenere un elenco di grafici che mostrano l'andamento della pandemia secondo diversi parametri.

La sezione **Notizie** dovrà contenere un elenco di notizie riguardanti il COVID-19.

La sezione **Info**, oltre a mostrare le informazioni riguardanti lo sviluppo dell'app, dovrà contenere un apposito pulsante per permette di recarsi direttamente sul sito del DPC ed effettuare una donazione.

### 5.3 Sviluppo

La sviluppo dell'applicazione è stato molto lungo in quanto, durante la fase iniziale, determinati meccanismi inerenti a Flutter non erano ancora del tutto dettagliati. Ad esempio, Il pattern MVC non è stato implementato poiché le documentazioni riguardo la sua integrazione non erano sufficienti. Di seguito sono riportati i concetti principali dell'applicazione e come sono stati sviluppati.

In flutter i Widget, come accennato precedentemente nel capitolo 3, sono una delle componenti fondamentali. Sono costruiti utilizzando un framework moderno che prende ispirazione da React. L'idea centrale è la costruzione dell'interfaccia utente tramite appositi widget. Essi descrivono come dovrebbe essere la loro visualizzazione

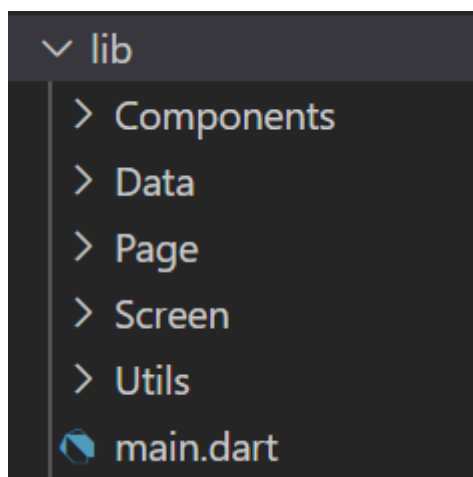
data la configurazione e lo stato corrente. Quando lo stato di un widget cambia, il widget viene ricostruito. [25]

Ogni schermata di **Covid Analytics** che non ha la necessità di conservare uno stato è stata realizzata attraverso l'uso di Stateless Widget (es., per la sezione Info).

Sono state realizzate come Stateful Widget tutte le schermate che necessitavano il mantenimento di uno stato. Questo per permettere di reagire ad eventi (es., risposte a chiamate http). Tre delle quattro schermate principali implementano un FutureBuilder, ovvero una Widget che permette di fare una chiamata http ed attendere la risposta per poi mostrare il risultato a schermo. Per la navigazione tra le varie schermate si è fatto uso di una semplice Bottom Bar.

Fondamentale per la visualizzazione dei dati è stata la chiamata di Web API, attraverso semplici richieste http. Tali chiamate restituiscono come risposta dei file JSON. Il corpo delle risposte è stato decodificato e istanziato come classi dell'applicazione.

La suddivisione del codice è stata effettuata cercando di renderlo quanto più possibile manutenibile e riutilizzabile.



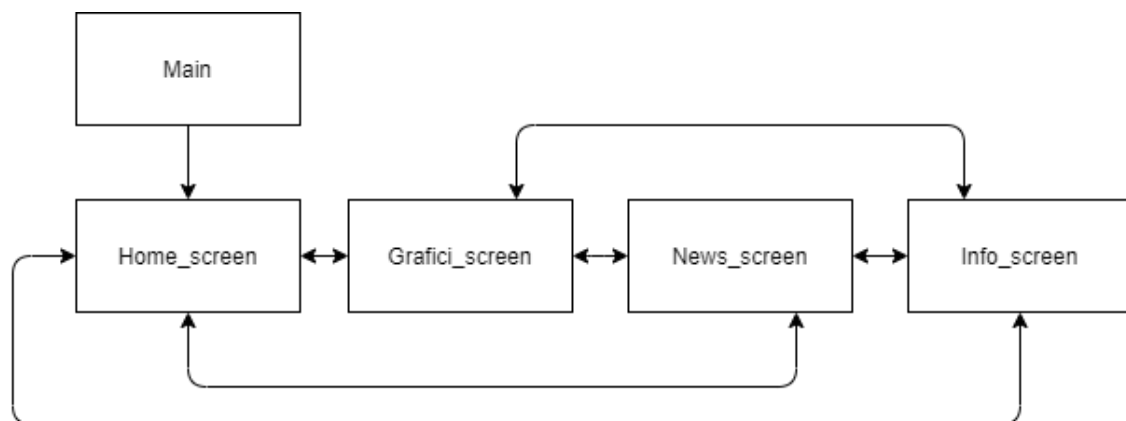
Struttura Codice

Il codice è stato suddiviso in:

- Components: qui sono stati inseriti widget riutilizzabili in diverse parti della stessa Page;
- Page: qui sono stati inseriti widget riutilizzabili in diversi Screen (es., elenco dei dati nazionali);
- Screen: qui sono state inserite le quattro schermate principali dell'App. (es., statistiche);
- Utils: qui sono state inserite classi di utilità. (es., Config, Style);
- Data: qui sono state inserite le Entity. Ovvero gli oggetti utilizzati per la memorizzazione di dati dopo la decodifica delle chiamate http. (es., DatiRegione).

## 5.4 Realizzazione App Mobile

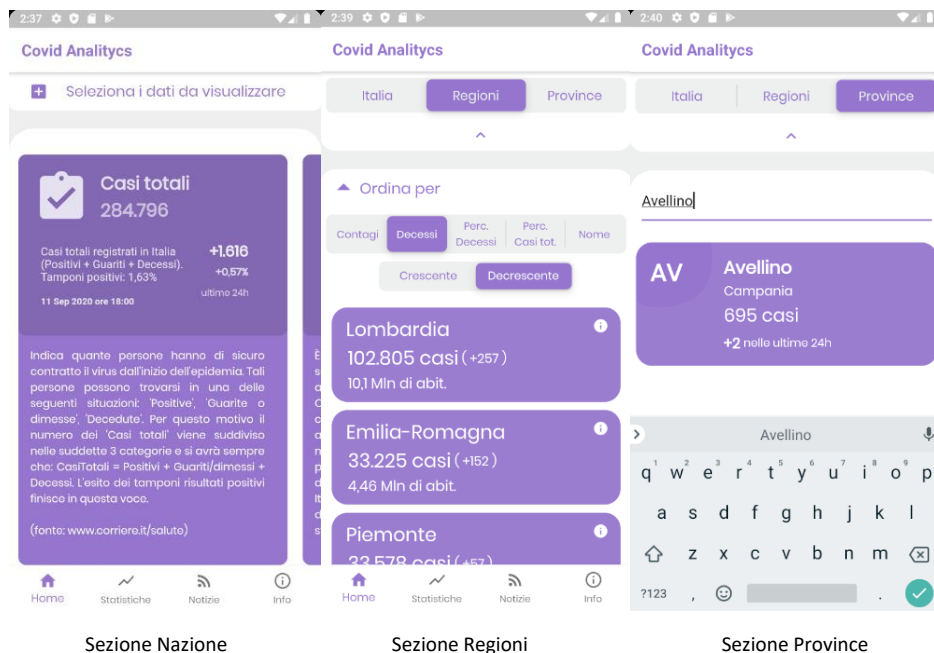
Di seguito verrà mostrata la struttura che indica l'interazione con l'utente finale. Grazie all'uso di una Bottom Bar la navigazione ottenuta dall'utente all'apertura dell'App è la seguente:



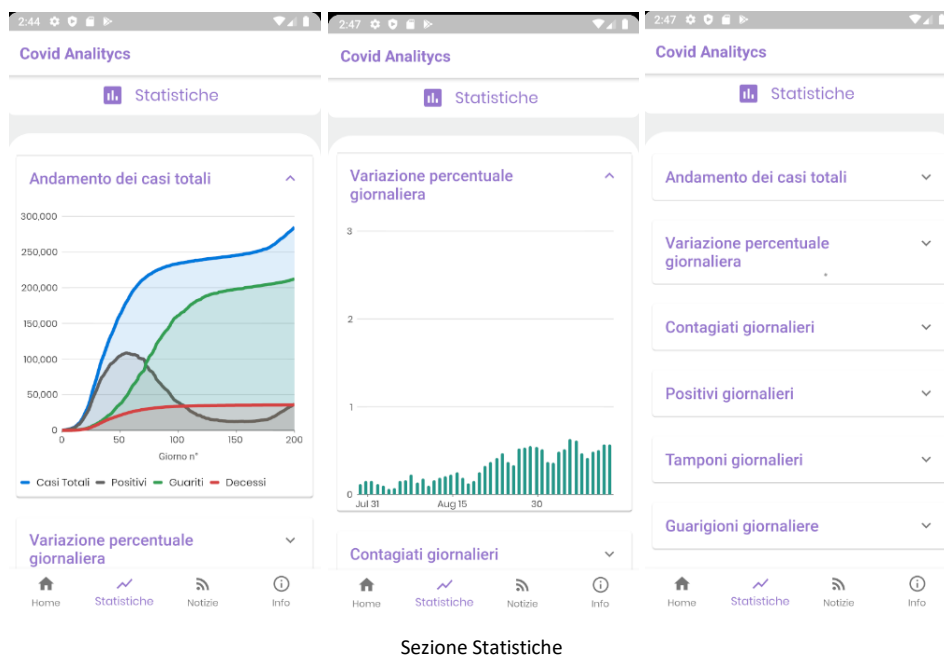
Quindi, da ogni Screen c'è la possibilità di passare ad un qualsiasi Screen a scelta. Gli Screen sono costituiti da Page (es., elenco dati nelle regioni), a loro volta costituite da Components (es., dati della singola regione).

Di seguito verranno mostrate le immagini che rappresentano gli screen dell'App.

### 5.4.1 Home\_screen



### 5.4.2 Grafici\_screen



### 5.4.3 News\_screen



Sezione Notizie

### 5.4.4 Info\_screen



Sezione Info

## 5.5 Web

### 5.5.1 Firebase Hosting

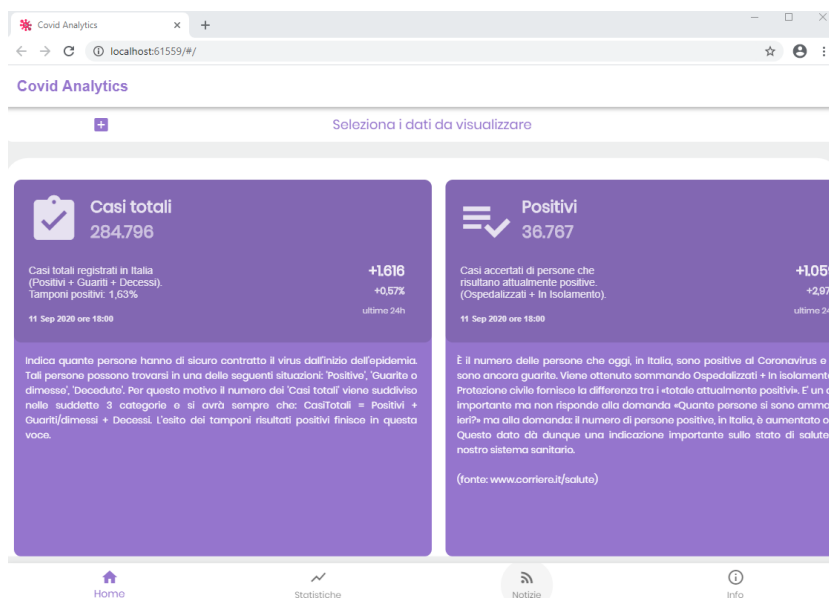
Firebase è una piattaforma per la creazione di applicazioni per dispositivi mobili e web sviluppata da Google. Originariamente era una compagnia indipendente fondata nel 2011. Nel 2014 Google ha acquisito la piattaforma ed è ora la sua offerta di punta per lo sviluppo di app.

Una delle funzionalità offerte da Firebase è l'hosting. Ha come caratteristica principale la possibilità di distribuire rapidamente i contenuti web. Con Firebase Hosting è possibile distribuire con gran facilità PWA o "pagine web a singola pagina". Dopo aver creato un account e configurato FirebaseCLI all'interno del proprio pc, la distribuzione avviene tramite un semplice linea di comando "firebase deploy".

### 5.5.1 App Web

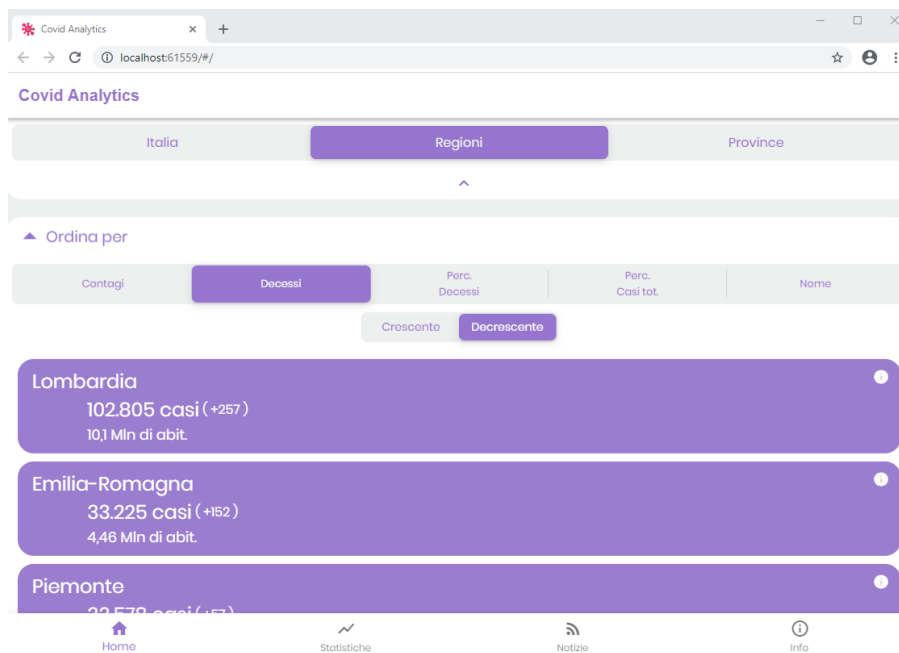
Di seguito verranno mostrate le immagini dell'applicazione in esecuzione tramite browser web.

#### 5.5.1.1 Home\_screen

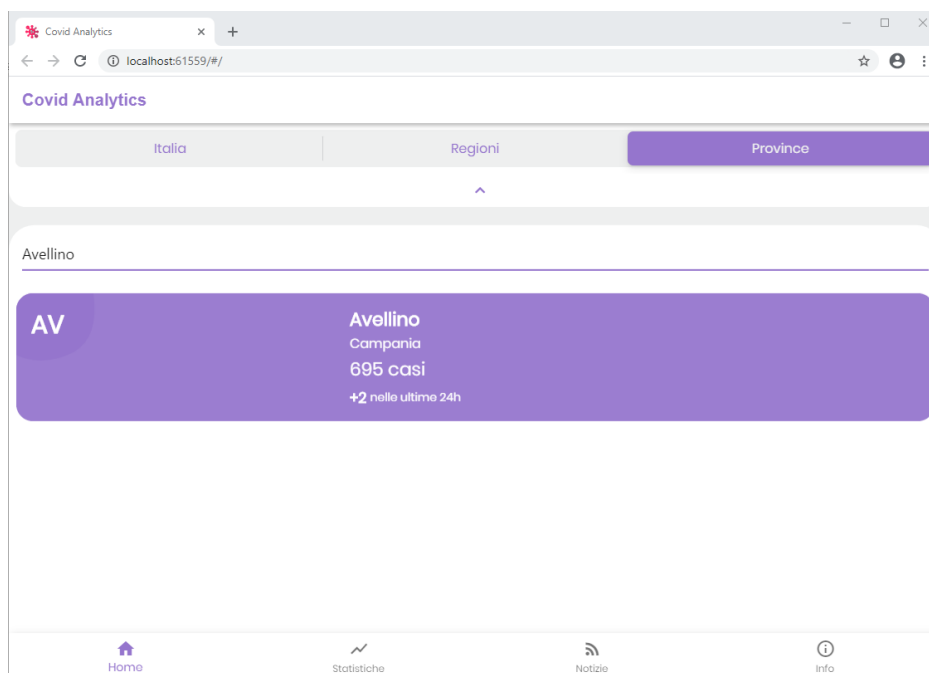


Sezione Nazione



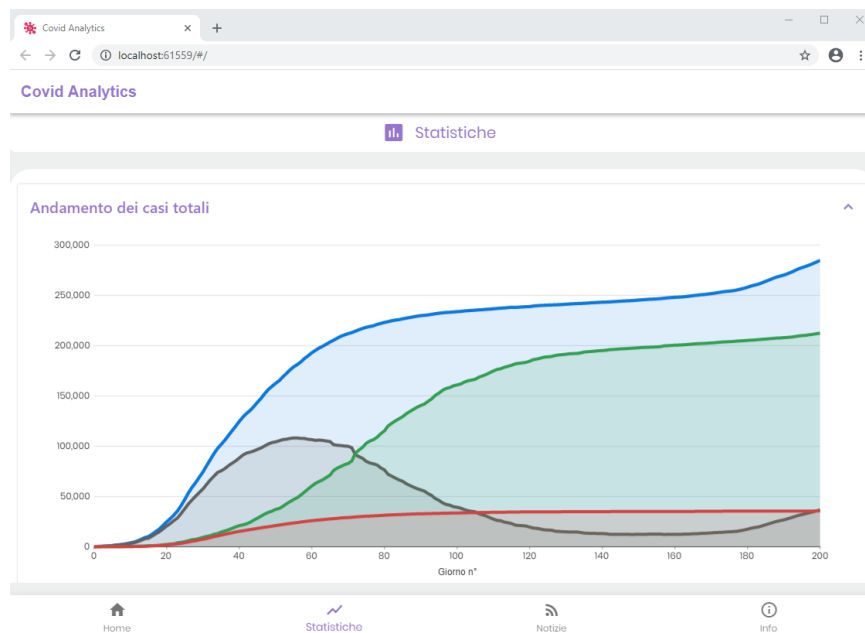


### Sezione Regioni



### Sezione Province

### 5.5.1.2 Grafici\_screen



Sezione Statistiche

### 5.5.1.3 News\_screen

Covid Analytics

Notizie (Fonte Sky News)

**Coronavirus, nuova ordinanza in Sardegna: test per chi arriva, mascherine obbligatorie h24**

Il provvedimento firmato dal governatore Christian Solinas dopo l'innalzamento dei casi dovuto alla movida estiva. Chi sbarca sull'isola è "invitato" a presentare un certificato di negatività al virus oppure dovrà sottoporsi a un tampone e restare in isolamento fiduciario in attesa del risultato

Nuova stretta contro il coronavirus in Sardegna dopo l'innalzamento dei contagi legata soprattutto alla movida estiva in Costa Smeralda, con l'arrivo di migliaia di turisti. Il governatore Christian Solinas ha firmato una nuova ordinanza restrittiva anti-covid: da lunedì, 14 settembre, chi arriva sull'isola è "invitato" a presentare un certificato di negatività al virus agli imbarchi di navi e aerei o a autocertificare di essere risultato negativo a un test sierologico, molecolare o antigenico. In assenza di questo "accetta" di effettuare un tampone entro le 48 ore dall'arrivo e a comunicarne l'esito alle autorità sanitarie locali. Nella stessa ordinanza viene imposto da subito l'obbligo di indossare le mascherine h24 anche all'aperto nel caso in cui non sia possibile rispettare la distanza di un metro (AGGIORNAMENTI IN DIRETTA - SPECIALE).

<https://tg24.sky.it/cronaca/2020/09/12/sardegna-covid-ordinanza>

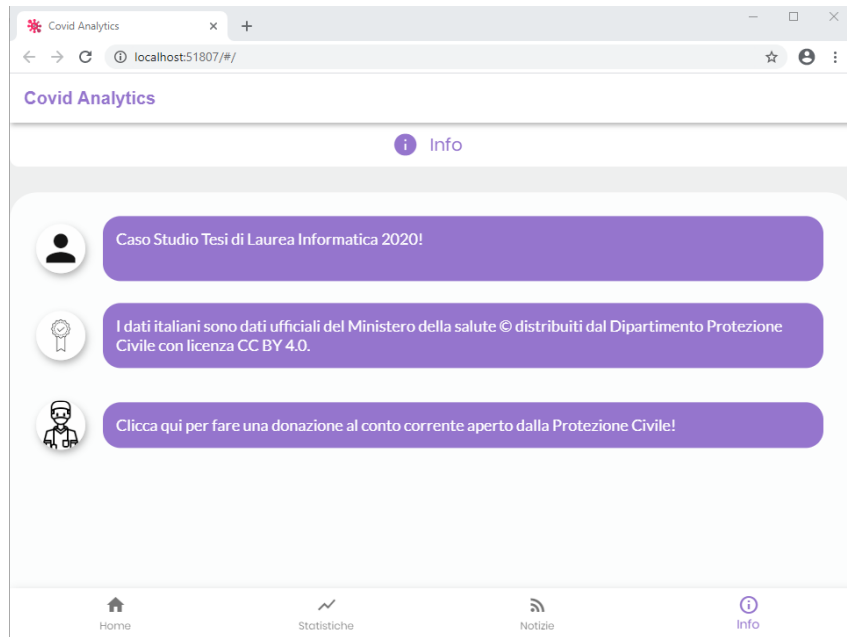
**Coronavirus, quali sono i nuovi focolai in Italia regione per regione**

I nuovi casi di Covid-19 indicati nel bollettino dell'11 settembre sono 1.616, con 98.880 tamponi analizzati. Nel report settimanale dell'Iss viene segnalato un aumento dei contagi per la sesta settimana consecutiva. Attualmente sono 2.280 i focolai attivi nel nostro Paese, di cui 691 nuovi:

Home Statistiche Notizie Info

Sezione Notizie

#### 5.5.1.4 Info\_screen



Sezione Info

## Capitolo 6

### Sviluppi Futuri

#### 6.1 Funzionalità aggiuntive

Alcune delle funzionalità che potrebbero essere aggiunte per migliorare quella che è l'efficienza dell'App sono le seguenti:

- Migliorie grafiche, (es., cercare di rendere la visualizzazione web migliore);
- Ricerca per giorno: questa funzionalità potrebbe permettere di ottenere i dati relativi ad un singolo giorno (scelto dall'utente);
- Ricerca per periodo di tempo: questa funzionalità potrebbe permettere all'utente di visionare i grafici selezionando un determinato intervallo di tempo;
- Assistenza: questa funzionalità potrebbe risultare utile, da parte dell'utente, per contattare i numeri di pubblica utilità in caso di necessità;
- Mappa: questa funzionalità permetterebbe la visione semplificata dei dati in generale. Grazie alla mappa, l'individuazione di eventuali focolai risulta facilitata.

#### 6.2 Integrazione con altre tecnologie

Uno degli sviluppi futuri più interessanti potrebbe essere l'integrazione con le API per il tracciamento realizzate da Google e Apple.

I tradizionali metodi di tracciamento dei contatti sono essenziali per contenere la diffusione dell'infezione. La tecnologia può supportare e migliorare questi metodi consentendo alle autorità sanitarie di informare tempestivamente coloro che potrebbero essere entrati in contatto con qualcuno che ha contratto il COVID-19.[22]

### 6.3 Diffusione informazioni sanitarie su altri tipi di pandemie

Una **pandemia** (dal greco *pan-demos*, "tutto il popolo") è una malattia epidemica che, diffondendosi rapidamente tra le persone e si espande su scala globale. L'OMS ha utilizzato una classificazione in sei stadi per descrivere il processo mediante il quale un virus procede dalle infezioni iniziali arrivando ad una pandemia. [17]

In passato, il mondo, purtroppo, ha visto il prendere vita di diverse pandemie (es., la Spagnola, Ebola, Colera, Sifilide, Malaria, ecc.).

Un buon utilizzo della tecnologia, e nel dettaglio dello sviluppo di App Multiplatforma, potrebbe essere quello di diffondere le **giuste informazioni** riguardante le varie pandemie. Ad esempio, indicando le zone più a rischio contagio o spiegando come comportarsi e chi contattare in caso di bisogno.

Siamo tutti i giorni collegati tramite lo smartphone... Perché non farne buon uso?

Quindi, un altro degli sviluppi futuri potrebbe essere quello di tenere sotto controllo altre pandemie presenti o future.

## Capitolo 7

### Conclusioni

#### 7.1 Multipiattaforma in ambito medico

Oltre alla diffusione dei dati per garantire a tutti una corretta informazione, lo sviluppo di App Cross Platform in ambito medico apporta dei notevoli risvolti positivi. Basti pensare alla “comunicazione” tra il personale sanitario stesso oppure tra quest’ultimo e i pazienti.

Ad esempio, grazie all’utilizzo dello standard **Fast Healthcare Interoperability Resources (FHIR)** è possibile realizzare in tempi brevi applicazioni veloci, flessibili e compatibili con diverse piattaforme.

FHIR è:

- una API per scambiare dati relativi alla cartella clinica elettronica;
- uno standard che descrive il modello e il formato dei dati.

Consente lo scambio di dati medicali usando protocolli web standard in maniera semplice, veloce ed efficace tra sistemi diversi (il tipo di dati utilizzato nelle comunicazioni è a scelta tra JSON, XML o RDF). [23]

Lo standard è stato creato dall'organizzazione per gli standard sanitari **Health Level Seven International (HL7)**. [24]

Tramite l’utilizzo di questa API medici, infermieri e pazienti avrebbero la possibilità, attraverso un pc o uno smartphone, di accedere ai dati di loro interesse nel modo più semplice possibile e in tutta sicurezza. Ad esempio, un App del genere potrebbe garantire al personale sanitario un quadro dettagliato ed organizzato delle cartelle cliniche di tutti i pazienti, fornendo addirittura eventuali statistiche. Il tutto a portata di qualche click. Da non escludere è la possibilità di ricevere ed inviare avvisi attraverso delle semplici notifiche (ad esempio, utilizzando Firebase).

## **7.2 Considerazioni Finali**

In conclusione, l'esperienza che ha portato alla stesura di questo elaborato è stata positiva da tutti i punti di vista. È stata fondamentale per capire quali sono metodologie di sviluppo delle applicazioni multiplatforma e per poter accumulare conoscenze tecniche essenziali per l'ingresso nel mondo del lavoro.

Importante è stato anche capire il ruolo che un'applicazione di questo genere possa avere in ambito medico. Ci sono diversi aspetti vantaggiosi da considerare.

Il primo riguarda la possibilità di diffondere informazioni a tutti coloro che possiedono uno smartphone in modo semplice e preciso.

Il secondo aspetto da tener presente è che l'utilizzo di tecnologie simili in ambito medico rende molto più semplice il lavoro dovuto all'organizzazione dei dati restituiti da apparecchiature. Il tutto è automatizzato e la visualizzazione dei dati, oltre ad essere organizzata, è accessibile a portata di click in qualsiasi istante da qualsiasi dispositivo.

# Bibliografia

- [1] *Applicazione.*
- [2] *App Native.*
- [3] *Sviluppo app Android.*
- [4] *Sviluppo app iOS.*
- [5] *App Ibride.*
- [6] *Cauro, Maria & Francese, Rita & Scanniello, Giuseppe & Spera, Antonio. (2019).  
Does the Migration of Cross-Platform Apps Towards the Android Platform Matter?  
An Approach and a User Study. 10.1007/978-3-030-35333-9\_9.*
- [7] *Cosa sono le PWA.*
- [8] *PWA.*
- [9] *Cross Platform.*
- [10] *Framework.*
- [11] *Framework.*
- [12] *React Native.*
- [13] *Svantaggi React.*
- [14] *Xamarin.*
- [15] *Frank Zammetti — Practical Flutter.*
- [16] *Flutter Doc.*



[17] *Pandemia.*

[18] *Prajyot Mainkar, Salvatore Giordano — Google Flutter Mobile Development Quick*

*Start Guide*

[19] *DPC.*

[20] *COVID-19.*

[21] *Flutter vs React Native vs Xamarin.*

[22] *Exposure Notifications.*

[23] *FHIR.*

[24] *HL7.*

[25] *Widget in Flutter.*

[26] *Firebase.*

[27] *Firebase Hosting.*